IMAGE and VIDEO COMPRESSION for MULTIMEDIA ENGINEERING

Fundamentals, Algorithms, and Standards

Yun Q. Shi
Huifang Sun

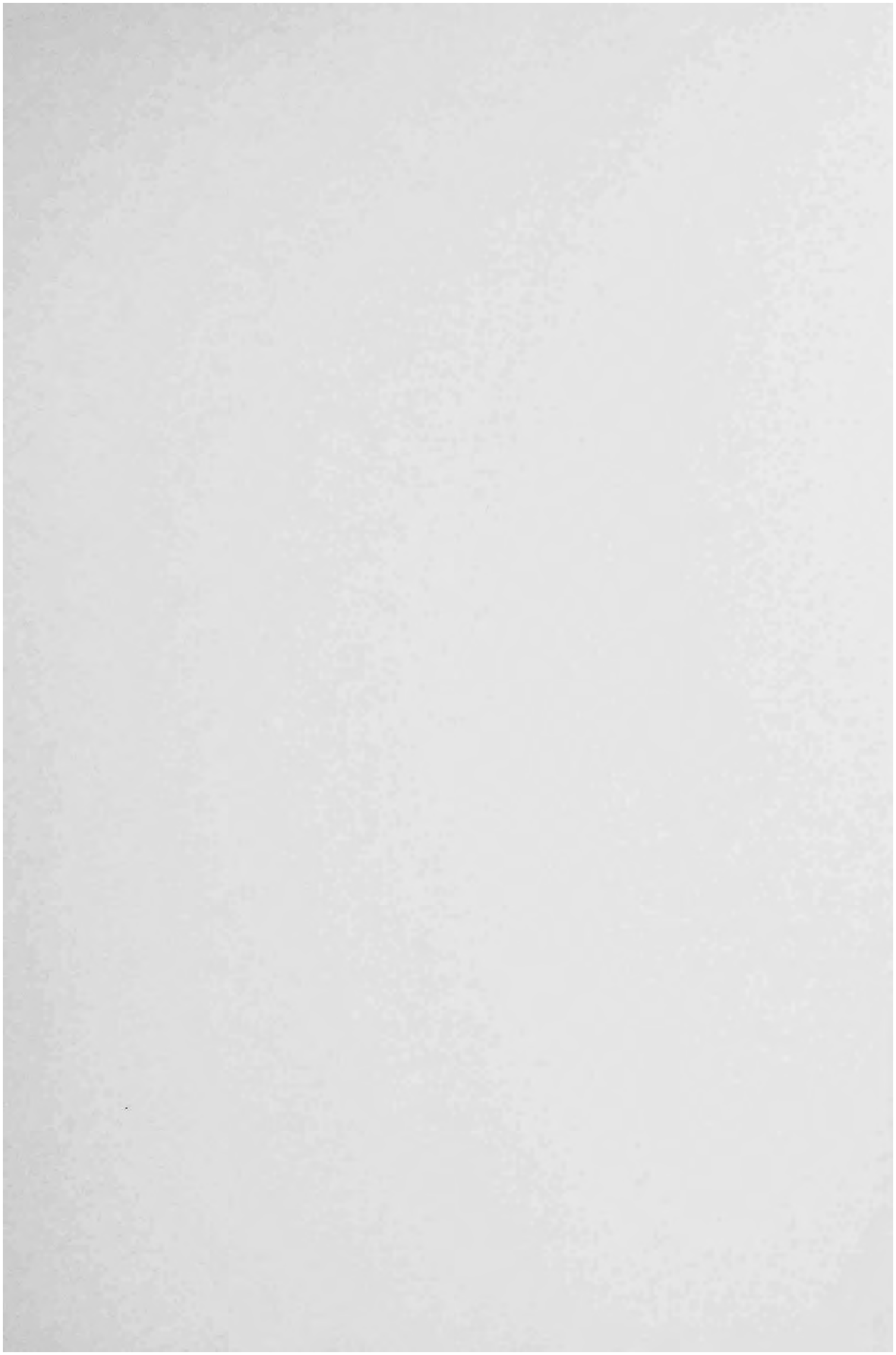


IMAGE and VIDEO COMPRESSION for MULTIMEDIA ENGINEERING

Fundamentals, Algorithms, and Standards

IMAGE PROCESSING SERIES

Series Editor: Phillip A. Laplante

Forthcoming Titles

Adaptive Image Processing: A Computational Intelligence Perspective
Ling Guan, Hau-San Wong, and Stuart William Perry

Shape Analysis and Classification: Theory and Practice
Luciano da Fontoura Costa and Roberto Marcondes Cesar, Jr.

IMAGE and VIDEO COMPRESSION for MULTIMEDIA ENGINEERING

Fundamentals, Algorithms, and Standards

Yun Q. Shi
New Jersey Institute of Technology
Newark, NJ

Huifang Sun
Mitsubishi Electric Information Technology Center
America Advanced Television Laboratory
New Providence, NJ

CRC Press
Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Shi, Yun Q.
Image and video compression for multimedia engineering : fundamentals, algorithms, and standards /
Yun Q. Shi, Huifang Sun.
p. cm.
Includes bibliographical references and index.
ISBN 0-8493-3491-8 (alk. paper)
1. Multimedia systems. 2. Image compression. 3. Video compression I. Sun, Huifang. II. Title.
QA76.575.S555 1999
006.7—dc21

99-047137

International Standard Book Number 0-8493-3491-8
Library of Congress Card Number 99-047137
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

Preface

It is well known that in the 1960s the advent of the semiconductor computer and the space program swiftly brought the field of digital image processing into public focus. Since then the field has experienced rapid growth and has entered into every aspect of modern technology. Since the early 1980s, digital image sequence processing has been an attractive research area because an image sequence, as a collection of images, may provide more information than a single image frame. The increased computational complexity and memory space required for image sequence processing are becoming more attainable. This is due to more advanced, achievable computational capability resulting from the continuing progress made in technologies, especially those associated with the VLSI industry and information processing.

In addition to image and image sequence processing in the digitized domain, facsimile transmission has switched from analog to digital since the 1970s. However, the concept of high definition television (HDTV) when proposed in the late 1970s and early 1980s continued to be analog. This has since changed. In the U.S., the first digital system proposal for HDTV appeared in 1990. The Advanced Television Standards Committee (ATSC), formed by the television industry, recommended the digital HDTV system developed jointly by the seven Grand Alliance members as the standard, which was approved by the Federal Communication Commission (FCC) in 1997. Today's worldwide prevailing concept of HDTV is digital. Digital television (DTV) provides the signal that can be used in computers. Consequently, the marriage of TV and computers has begun. Direct broadcasting by satellite (DBS), digital video disks (DVD), video-on-demand (VOD), video games, and other digital video related media and services are available now, or soon will be.

As in the case of image and video transmission and storage, audio transmission and storage through some media have changed from analog to digital. Examples include entertainment audio on compact disks (CD) and telephone transmission over long and medium distances. Digital TV signals, mentioned above, provide another example since they include audio signals. Transmission and storage of audio signals through some other media are about to change to digital. Examples of this include telephone transmission through local area and cable TV.

Although most signals generated from various sensors are analog in nature, the switching from analog to digital is motivated by the superiority of digital signal processing and transmission over their analog counterparts. The principal advantage of the digital signal is its robustness against various noises. Clearly, this results from the fact that only binary digits exist in digital format and it is much easier to distinguish one state from the other than to handle analog signals.

Another advantage of being digital is ease of signal manipulation. In addition to the development of a variety of digital signal processing techniques (including image, video, and audio) and specially designed software and hardware that may be well known, the following development is an example of this advantage. The digitized information format, i.e., the bitstream, often in a compressed version, is a revolutionary change in the video industry that enables many manipulations which are either impossible or very complicated to execute in analog format. For instance, video, audio, and other data can be first compressed to separate bitstreams and then combined to form a signal bitstream, thus providing a multimedia solution for many practical applications. Information from different sources and to different devices can be multiplexed and demultiplexed in terms of the bitstream. Bitstream conversion in terms of bit rate conversion, resolution conversion, and syntax conversion becomes feasible. In digital video, content-based coding, retrieval, and manipulation and the ability to edit video in the compressed domain become feasible. All system-timing signals

in the digital systems can be included in the bitstream instead of being transmitted separately as in traditional analog systems.

The digital format is well suited to the recent development of modern telecommunication structures as exemplified by the Internet and World Wide Web (WWW). Therefore, we can see that digital computers, consumer electronics (including television and video games), and telecommunications networks are combined to produce an information revolution. By combining audio, video, and other data, multimedia becomes an indispensable element of modern life. While the pace and the future of this revolution cannot be predicted, one thing is certain: this process is going to drastically change many aspects of our world in the next several decades.

One of the enabling technologies in the information revolution is digital data compression, since the digitization of analog signals causes data expansion. In other words, storage and/or transmission of digitized signals require more storage space and/or bandwidth than the original analog signals.

The focus of this book is on image and video compression encountered in multimedia engineering. Fundamentals, algorithms, and standards are the three emphases of the book. It is intended to serve as a senior/graduate-level text. Its material is sufficient for a one-semester or one-quarter graduate course on digital image and video coding. For this purpose, at the end of each chapter there is a section of exercises containing problems and projects for practice, and a section of references for further reading.

Based on this book, a short course entitled "Image and Video Compression for Multimedia," was conducted at Nanyang Technological University, Singapore in March and April, 1999. The response to the short course was overwhelmingly positive.

Authors

Dr. Yun Q. Shi has been a professor with the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology, Newark, NJ since 1987. Before that he obtained his B.S. degree in Electronic Engineering and M.S. degree in Precision Instrumentation from the Shanghai Jiao Tong University, Shanghai, China and his Ph.D. in Electrical Engineering from the University of Pittsburgh. His research interests include motion analysis from image sequences, video coding and transmission, digital image watermarking, computer vision, applications of digital image processing and pattern recognition to industrial automation and biomedical engineering, robust stability, spectral factorization, multidimensional systems and signal processing. Prior to entering graduate school, he worked in a radio factory as a design and test engineer in digital control manufacturing and in electronics.

He is the author or coauthor of about 90 journal and conference proceedings papers in his research areas and has been a formal reviewer of the *Mathematical Reviews* since 1987, an IEEE senior member since 1993, and the chairman of Signal Processing Chapter of IEEE North Jersey Section since 1996. He was an associate editor for *IEEE Transactions on Signal Processing* responsible for Multidimensional Signal Processing from 1994 to 1999, the guest editor of the special issue on Image Sequence Processing for the *International Journal of Imaging Systems and Technology*, published as Volumes 9.4 and 9.5 in 1998, one of the contributing authors in the area of Signal and Image Processing to the *Comprehensive Dictionary of Electrical Engineering*, published by the CRC Press LLC in 1998. His biography has been selected by Marquis Who's Who for inclusion in the 2000 edition of *Who's Who in Science and Engineering*.

Dr. Huifang Sun received the B.S. degree in Electrical Engineering from Harbin Engineering Institute, Harbin, China, and the Ph.D. in Electrical Engineering from University of Ottawa, Ottawa, Canada. In 1986 he jointed Fairleigh Dickinson University, Teaneck, NJ as an assistant professor and was promoted to an associate professor in electrical engineering. From 1990 to 1995, he was with the David Sarnoff Research Center (Sarnoff Corp.) in Princeton as a member of technical staff and later promoted to technology leader of Digital Video Technology where his activities included MPEG video coding, AD-HDTV, and Grand Alliance HDTV development. He joined the Advanced Television Laboratory, Mitsubishi Electric Information Technology Center America (ITA), New Providence, NJ in 1995 as a senior principal technical staff and was promoted to deputy director in 1997 working in advanced television development and digital video processing. He has been active in MPEG video standards for many years and holds 10 U.S. patents with several pending. He has authored or coauthored more than 80 journal and conference papers and obtained the 1993 best paper award of IEEE Transactions on Consumer Electronics, and 1997 best paper award of International Conference on Consumer Electronics. For his contributions to HDTV development, he obtained the 1994 Sarnoff technical achievement award. He is currently the associate editor of *IEEE Transactions on Circuits and Systems for Video Technology*.

Acknowledgments

We are pleased to express our gratitude here for the support and help we received in the course of writing this book.

The first author thanks his friend and former colleague, Dr. C. Q. Shu, for fruitful technical discussions related to some contents of the book. Sincere thanks also are directed to several of his friends and former students, Drs. J. N. Pan, X. Xia, S. Lin, and Y. Shi, for their technical contributions and computer simulations related to some subjects of the book. He is grateful to Ms. L. Fitton for her English editing of 11 chapters, and to Dr. Z. F. Chen for her help in preparing many graphics.

The second author expresses his appreciation to his colleagues, Anthony Vetro and Ajay Divakaran, for fruitful technical discussion related to some contents of the book and for proofreading nine chapters. He also extends his appreciation to Dr. Xiaobing Lee for his help in providing some useful references, and to many friends and colleagues of the MPEGers who provided wonderful MPEG documents and tutorial materials that are cited in some chapters of this book. He also would like to thank Drs. Tommy Poor, Jim Foley, and Toshiaki Sakaguchi for their continuing support and encouragement.

Both authors would like to express their deep appreciation to Dr. Z. F. Chen for her great help in formatting all the chapters of the book. They also thank Dr. F. Chichester for his help in preparing the book.

Special thanks go to the editor-in-chief of the Image Processing book series of CRC Press, Dr. P. Laplante, for his constant encouragement and guidance. Help from the editors at CRC Press, N. Konopka, M. Mogck, and other staff, is appreciated.

The first author acknowledges the support he received associated with writing this book from the Electrical and Computer Engineering Department at the New Jersey Institute of Technology. In particular, thanks are directed to the department chairman, Professor R. Haddad, and the associate chairman, Professor K. Sohn. He is also grateful to the Division of Information Engineering and the Electrical and Electronic Engineering School at Nanyang Technological University (NTU), Singapore for the support he received during his sabbatical leave. It was in Singapore that he finished writing the manuscript. In particular, thanks go to the dean of the school, Professor Er Meng Hwa, and the division head, Professor A. C. Kot. With pleasure, he expresses his appreciation to many of his colleagues at the NTU for their encouragement and help. In particular, his thanks go to Drs. G. Li and J. S. Li, and Dr. G. A. Bi. Thanks are also directed to many colleagues, graduate students, and some technical staff from industrial companies in Singapore who attended the short course which was based on this book in March/April 1999 and contributed their enthusiastic support and some fruitful discussion.

Last but not least, both authors thank their families for their patient support during the course of the writing. Without their understanding and support we would not have been able to complete this book.

Yun Q. Shi
Huifang Sun

Content and Organization of the Book

The entire book consists of 20 chapters which can be grouped into four sections:

I. Fundamentals,
II. Still Image Compression,
III. Motion Estimation and Compensation, and
IV. Video Compression.

In the following, we summarize the aim and content of each chapter and each part, and the relationships between some chapters and between the four parts.

Section I includes the first six chapters. It provides readers with a solid basis for understanding the remaining three parts of the book. In Chapter 1, the practical needs for image and video compression is demonstrated. The feasibility of image and video compression is analyzed. Specifically, both statistical and psychovisual redundancies are analyzed and the removal of these redundancies leads to image and video compression. In the course of the analysis, some fundamental characteristics of the human visual system are discussed. Visual quality measurement as another important concept in the compression is addressed in both subjective and objective quality measures. The new trend in combining the virtues of the two measures also is presented. Some information theory results are presented as the final subject of the chapter.

Quantization, as a crucial step in lossy compression, is discussed in Chapter 2. It is known that quantization has a direct impact on both the coding bit rate and quality of reconstructed frames. Both uniform and nonuniform quantization are covered. The issues of quantization distortion, optimum quantization, and adaptive quantization are addressed. The final subject discussed in the chapter is pulse code modulation (PCM) which, as the earliest, best-established, and most frequently applied coding system normally serves as a standard against which other coding techniques are compared.

Two efficient coding schemes, differential coding and transform coding (TC), are discussed in Chapters 3 and 4, respectively. Both techniques utilize the redundancies discussed in Chapter 1, thus achieving data compression. In Chapter 3, the formulation of general differential pulse code modulation (DPCM) systems is described first, followed by discussions of optimum linear prediction and several implementation issues. Then, delta modulation (DM), an important, simple, special case of DPCM, is presented. Finally, application of the differential coding technique to interframe coding and information-preserving differential coding are covered.

Chapter 4 begins with the introduction of the Hotelling transform, the discrete version of the optimum Karhunen and Loeve transform. Through statistical, geometrical, and basis vector (image) interpretations, this introduction provides a solid understanding of the transform coding technique. Several linear unitary transforms are then presented, followed by performance comparisons between these transforms in terms of energy compactness, mean square reconstruction error, and computational complexity. It is demonstrated that the discrete cosine transform (DCT) performs better than others, in general. In the discussion of bit allocation, an efficient adaptive scheme is presented using thresholding coding devised by Chen and Pratt in 1984, which established a basis for the international still image coding standard, Joint Photographic (image) Experts Group (JPEG). The

comparison between DPCM and TC is given. The combination of these two techniques (hybrid transform/waveform coding), and its application in image and video coding also are described.

The last two chapters in the first part cover some coding (codeword assignment) techniques. In Chapter 5, two types of variable-length coding techniques, Huffman coding and arithmetic coding, are discussed. First, an introduction to some basic coding theory is presented, which can be viewed as a continuation of the information theory results presented in Chapter 1. Then the Huffman code, as an optimum and instantaneous code, and a modified version are covered. Huffman coding is a systematic procedure for encoding a source alphabet with each source symbol having an occurrence probability. As a block code (a fixed codeword having an integer number of bits is assigned to a source symbol), it is optimum in the sense that it produces minimum coding redundancy. Some limitations of Huffman coding are analyzed. As a stream-based coding technique, arithmetic coding is distinct from and is gaining more popularity than Huffman coding. It maps a string of source symbols into a string of code symbols. Free of the integer-bits-per-source-symbol restriction, arithmetic coding is more efficient. The principle of arithmetic coding and some of its implementation issues are addressed.

While the two types of variable-length coding techniques introduced in Chapter 5 can be classified as fixed-length to variable-length coding techniques, both run-length coding (RLC) and dictionary coding, discussed in Chapter 6, can be classified as variable-length to fixed-length coding techniques. The discrete Markov source model (another portion of the information theory results) that can be used to characterize 1-D RLC, is introduced at the beginning of Chapter 6. Both 1-D RLC and 2-D RLC are then introduced. The comparison between 1-D and 2-D RLC is made in terms of coding efficiency and transmission error effect. The digital facsimile coding standards based on 1-D and 2-D RLC are introduced. Another focus of Chapter 6 is on dictionary coding. Two groups of adaptive dictionary coding techniques, the LZ77 and LZ78 algorithms, are presented and their applications are discussed. At the end of the chapter, a discussion of international standards for lossless still image compression is given. For both lossless bilevel and multilevel still image compression, the respective standard algorithms and their performance comparisons are provided.

Section II of the book (Chapters 7, 8, and 9) is devoted to still image compression. In Chapter 7, the international still image coding standard, JPEG, is introduced. Two classes of encoding: lossy and lossless; and four modes of operation: sequential DCT-based mode, progressive DCT-based mode, lossless mode, and hierarchical mode are covered. The discussion in the first part of the book is very useful in understanding what is introduced here for JPEG.

Due to its higher coding efficiency and superior spatial and quality scalability features over the DCT coding technique, the discrete wavelet transform (DWT) coding has been adopted by JPEG-2000 still image coding standards as the core technology. Chapter 8 begins with an introduction to wavelet transform (WT), which includes a comparison between WT and the short-time Fourier transform (STFT), and presents WT as a unification of several existing techniques known as filter bank analysis, pyramid coding, and subband coding. Then the DWT for still image coding is discussed. In particular, the embedded zerotree wavelet (EZW) technique and set partitioning in hierarchical trees (SPIHT) are discussed. The updated JPEG-2000 standard activity is presented.

Chapter 9 presents three nonstandard still image coding techniques: vector quantization (VQ), fractal, and model-based image coding. All three techniques have several important features such as very high compression ratios for certain kinds of images, and very simple decoding procedures. Due to some limitations, however, they have not been adopted by the still image coding standards. On the other hand, the facial model and face animation technique have been adopted by the MPEG-4 video standard.

Section III, consisting of Chapters 10 through 14, addresses the motion estimation and motion compensation — key issues in modern video compression. In this sense, Section III is a prerequisite to Section IV, which discusses various video coding standards. The first chapter in Section III, Chapter 10, introduces motion analysis and compensation in general. The chapter begins with the concept of imaging space, which characterizes all images and all image sequences in temporal and

spatial domains. Both temporal and spatial image sequences are special proper subsets of the imaging space. A single image becomes merely a specific cross section of the imaging space. Two techniques in video compression utilizing interframe correlation, both developed in the late 1960s and early 1970s, are presented. Frame replenishment is relatively simpler in modeling and implementation. However, motion compensated coding achieves higher coding efficiency and better quality in reconstructed frames with a 2-D displacement model. Motion analysis is then viewed from the signal processing perspective. Three techniques in motion analysis are briefly discussed. They are block matching, pel recursion, and optical flow, which are presented in detail in Chapters 11, 12, and 13, respectively. Finally, other applications of motion compensation to image sequence processing are discussed.

Chapter 11 addresses the block matching technique, which presently is the most frequently used motion estimation technique. The chapter first presents the original block matching technique proposed by Jain and Jain. Several different matching criteria and search strategies are then discussed. A thresholding multiresolution block matching algorithm is described in some detail so as to provide an insight into the technique. Then, the limitations of block matching techniques are analyzed, from which several new improvements are presented. They include hierarchical block matching, multigrid block matching, predictive motion field segmentation, and overlapped block matching. All of these techniques modify the nonoverlapped, equally spaced, fix-sized, small rectangular block model proposed by Jain and Jain in some way so that the motion estimation is more accurate and has fewer block artifacts and less overhead side information.

The pel recursive technique is discussed in Chapter 12. First, determination of 2-D displacement vectors is converted via the use of the displaced frame difference (DFD) concept to a minimization problem. Second, descent methods in optimization theory are discussed. In particular, the steepest descent method and Newton-Raphson method are addressed in terms of algorithm, convergence, and implementation issues such as selection of step-size and initial value. Third, the first pel recursive techniques proposed by Netravali and Robbins are presented. Finally, several improvement algorithms are described.

Optical flow, the third technique in motion estimation for video coding, is covered in Chapter 13. First, some fundamental issues in motion estimation are addressed. They include the difference and relationships between 2-D motion and optical flow, the aperture problem, and the ill-posed nature of motion estimation. The gradient-based and correlation-based approaches to optical flow determination are then discussed in detail. For the former, the Horn and Schunck algorithm is illustrated as a representative technique and some other algorithms are briefly introduced. For the latter, the Singh method is introduced as a representative technique. In particular, the concepts of conservation information and neighborhood information are emphasized. A correlation-feedback algorithm is presented in detail to provide an insight into the correlation technique. Finally, multiple attributes for conservation information are discussed.

Chapter 14, the last chapter in Section III, provides a further discussion and summary of 2-D motion estimation. First, a few features common to all three major techniques discussed in Chapters 11, 12, and 13 are addressed. They are the aperture and ill-posed inverse problems, conservation and neighborhood information, occlusion and disocclusion, rigid and nonrigid motion. Second, a variety of different classifications of motion estimation techniques is presented. Frequency domain methods are discussed as well. Third, a performance comparison between the three major techniques in motion estimation is made. Finally, the new trends in motion estimation are presented.

Section IV, discussing various video coding standards, is covered in Chapters 15 through 20. Chapter 15 presents fundamentals of video coding. First, digital video representation is discussed. Second, the rate distortion function of the video signal is covered — the fourth portion of the information theory results presented in this book. Third, various digital video formats are discussed. Finally, the current digital image/video coding standards are summarized. The full names and abbreviations of some organizations, the completion time, and the major features of various image/video coding standards are listed in two tables.

Chapter 16 is devoted to video coding standards MPEG-1/2, which are the most widely used video coding standards at the present. The basic technique of MPEG-1/2 is a full-motion-compensated DCT and DPCM hybrid coding algorithm. The features of MPEG-1 (including layered data structure) and the MPEG-2 enhancements (including field/frame modes for supporting the interlaced video input and scalability extension) are described. Issues of rate control, optimum mode decision, and multiplexing are discussed.

Chapter 17 presents several application examples of MPEG-1/2 video standards. They are the ATSC DTV standard approved by the FCC in the U.S., transcoding, the down-conversion decoder, and error concealment. Discussion of these applications can enhance the understanding and mastering of MPEG-1/2 standards. Some research work is reported that may be helpful for graduate students to broaden their knowledge of digital video processing — an active research field.

Chapter 18 presents the MPEG-4 video standard. The predominant feature of MPEG-4, content-based manipulation, is emphasized. The underlying concept of audio/visual objects (AVOs) is introduced. The important functionalities of MPEG-4: content-based interactivity (including bit-stream editing, synthetic and natural hybrid coding [SNHC]), content-based coding efficiency, and universal access (including content-based scalability), are discussed. Since neither MPEG-1 nor MPEG-2 includes synthetic video and content-based coding, the most important application of MPEG-4 is in a multimedia environment.

Chapter 19 introduces ITU-T video coding standards H.261 and H.263, which are utilized mainly for videophony and videoconferencing. The basic technical details of H.261, the earliest video coding standard, are presented. The technical improvements by which H.263 achieves high coding efficiency are discussed. Features of H.263+, H.263++, and H.26L are presented.

Chapter 20 covers the systems part of MPEG — multiplexing/demultiplexing and synchronizing the coded audio and video as well as other data. Specifically, MPEG-2 systems and MPEG-4 systems are introduced. In MPEG-2 systems, two forms: Program Stream and Transport Stream, are described. In MPEG-4 systems, some multimedia application related issues are discussed.

Contents

Section I Fundamentals

Section IV Video Compression

Dedication

To beloved Kong Wai Shih and Wen Su,
Yi Xi Li and Shu Jun Zheng,
Xian Hong Li,
and
To beloved Xuedong, Min, Yin, Andrew, and Haixin

Section 1

Fundamentals

1 Introduction

Image and video data compression* refers to a process in which the amount of data used to represent image and video is reduced to meet a bit rate requirement (below or at most equal to the maximum available bit rate), while the quality of the reconstructed image or video satisfies a requirement for a certain application and the complexity of computation involved is affordable for the application. The block diagram in Figure 1.1 shows the functionality of image and video data compression in visual transmission and storage. Image and video data compression has been found to be necessary in these important applications, because the huge amount of data involved in these and other applications usually greatly exceeds the capability of today's hardware despite rapid advancements in the semiconductor, computer, and other related industries.

It is noted that information and data are two closely related yet different concepts. Data represent information, and the quantity of data can be measured. In the context of digital image and video, data are usually measured by the number of binary units (bits). Information is defined as knowledge, facts, and news according to the Cambridge International Dictionary of English. That is, while data are the *representations* of knowledge, facts, and news, information *is* the knowledge, facts, and news. Information, however, may also be quantitatively measured.

The bit rate (also known as the coding rate), is an important parameter in image and video compression and is often expressed in a unit of bits per second, which is suitable in visual communication. In fact, an example in Section 1.1 concerning videophony (a case of visual transmission) uses the bit rate in terms of bits per second (bits/sec, or simply bps). In the application of image storage, the bit rate is usually expressed in a unit of bits per pixel (bpp). The term pixel is an abbreviation for picture element and is sometimes referred to as pel. In information source coding, the bit rate is sometimes expressed in a unit of bits per symbol. In Section 1.4.2, when discussing noiseless source coding theorem, we consider the bit rate as the average length of codewords in the unit of bits per symbol.

The required quality of the reconstructed image and video is application dependent. In medical diagnoses and some scientific measurements, we may need the reconstructed image and video to mirror the original image and video. In other words, only reversible, information-preserving schemes are allowed. This type of compression is referred to as lossless compression. In applications such as motion pictures and television (TV), a certain amount of information loss is allowed. This type of compression is called lossy compression.

From its definition, one can see that image and video data compression involves several fundamental concepts including information, data, visual quality of image and video, and computational complexity. This chapter is concerned with several fundamental concepts in image and video compression. First, the necessity as well as the feasibility of image and video data compression are discussed. The discussion includes the utilization of several types of redundancies inherent in image and video data, and the visual perception of the human visual system (HVS). Since the quality of the reconstructed image and video is one of our main concerns, the subjective and objective measures of visual quality are addressed. Then we present some fundamental information theory results, considering that they play a key role in image and video compression.

* In this book, the terms image and video data compression, image and video compression, and image and video coding are synonymous.

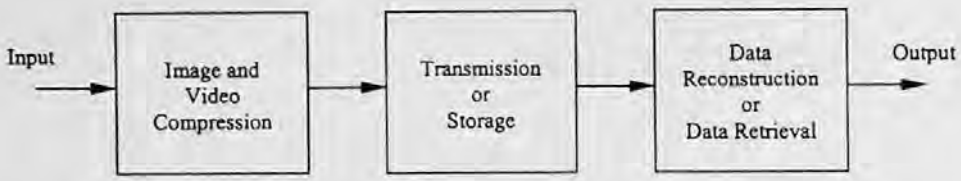


FIGURE 1.1 Image and video compression for visual transmission and storage.

1.1 PRACTICAL NEEDS FOR IMAGE AND VIDEO COMPRESSION

Needless to say, visual information is of vital importance if human beings are to perceive, recognize, and understand the surrounding world. With the tremendous progress that has been made in advanced technologies, particularly in very large scale integrated (VLSI) circuits, and increasingly powerful computers and computations, it is becoming more than ever possible for video to be widely utilized in our daily lives. Examples include videophony, videoconferencing, high definition TV (HDTV), and the digital video disk (DVD), to name a few.

Video as a sequence of video frames, however, involves a huge amount of data. Let us take a look at an illustrative example. Assume the present switch telephone network (PSTN) modem can operate at a maximum bit rate of 56,600 bits per second. Assume each video frame has a resolution of 288 by 352 (288 lines and 352 pixels per line), which is comparable with that of a normal TV picture and is referred to as common intermediate format (CIF). Each of the three primary colors RGB (red, green, blue) is represented for 1 pixel with 8 bits, as usual, and the frame rate in transmission is 30 frames per second to provide a continuous motion video. The required bit rate, then, is $288 \times 352 \times 8 \times 3 \times 30 = 72{,}990{,}720$ bps. Therefore, the ratio between the required bit rate and the largest possible bit rate is about 1289. This implies that we have to compress the video data by at least 1289 times in order to accomplish the transmission described in this example. Note that an audio signal has not yet been accounted for yet in this illustration.

With increasingly complex video services such as 3-D movies and 3-D games, and high video quality such as HDTV, advanced image and video data compression is necessary. It becomes an enabling technology to bridge the gap between the required huge amount of video data and the limited hardware capability.

1.2 FEASIBILITY OF IMAGE AND VIDEO COMPRESSION

In this section we shall see that image and video compression is not only a necessity for the rapid growth of digital visual communications, but it is also feasible. Its feasibility rests with two types of redundancies, i.e., statistical redundancy and psychovisual redundancy. By eliminating these redundancies, we can achieve image and video compression.

1.2.1 Statistical Redundancy

Statistical redundancy can be classified into two types: interpixel redundancy and coding redundancy. By interpixel redundancy we mean that pixels of an image frame and pixels of a group of successive image or video frames are not statistically independent. On the contrary, they are correlated to various degrees. (Note that the differences and relationships between image and video sequences are discussed in Chapter 10, when we begin to discuss video compression.) This type of interpixel correlation is referred to as interpixel redundancy. Interpixel redundancy can be divided into two categories, spatial redundancy and temporal redundancy. By coding redundancy we mean the statistical redundancy associated with coding techniques.

1.2.1.1 Spatial Redundancy

Spatial redundancy represents the statistical correlation between pixels within an image frame. Hence it is also called intraframe redundancy.

It is well known that for most properly sampled TV signals the normalized autocorrelation coefficients along a row (or a column) with a one-pixel shift is very close to the maximum value of 1. That is, the intensity values of pixels along a row (or a column) have a very high autocorrelation (close to the maximum autocorrelation) with those of pixels along the same row (or the same column), but shifted by a pixel. This does not come as a surprise because most of the intensity values change continuously from pixel to pixel within an image frame except for the edge regions. This is demonstrated in Figure 1.2. Figure 1.2(a) is a normal picture — a boy and a girl in a park, and is of a resolution of 883 by 710. The intensity profiles along the 318th row and the 262nd column are depicted in Figures 1.2(b) and (c), respectively. For easy reference, the positions of the 318th row and 262nd column in the picture are shown in Figure 1.2(d). That is, the vertical axis represents intensity values, while the horizontal axis indicates the pixel position within the row or the column. These two plots (shown in Figures 1.2(b) and 1.2(c)) indicate that intensity values often change gradually from one pixel to the other along a row and along a column.

The study of the statistical properties of video signals can be traced back to the 1950s. Knowing that we must study and understand redundancy in order to remove redundancy, Kretzmer designed some experimental devices such as a picture autocorrelator and a probabiloscope to measure several statistical quantities of TV signals and published his outstanding work in (Kretzmer, 1952). He found that the autocorrelation in both horizontal and vertical directions exhibits similar behaviors, as shown in Figure 1.3. Autocorrelation functions of several pictures with different complexities were measured. It was found that from picture to picture, the shape of the autocorrelation curves ranges from remarkably linear to somewhat exponential. The central symmetry with respect to the vertical axis and the bell-shaped distribution, however, remains generally the same. When the pixel shifting becomes small, it was found that the autocorrelation is high. This "local" autocorrelation can be as high as 0.97 to 0.99 for one- or two-pixel shifting. For very detailed pictures, it can be from 0.43 to 0.75. It was also found that autocorrelation generally has no preferred direction.

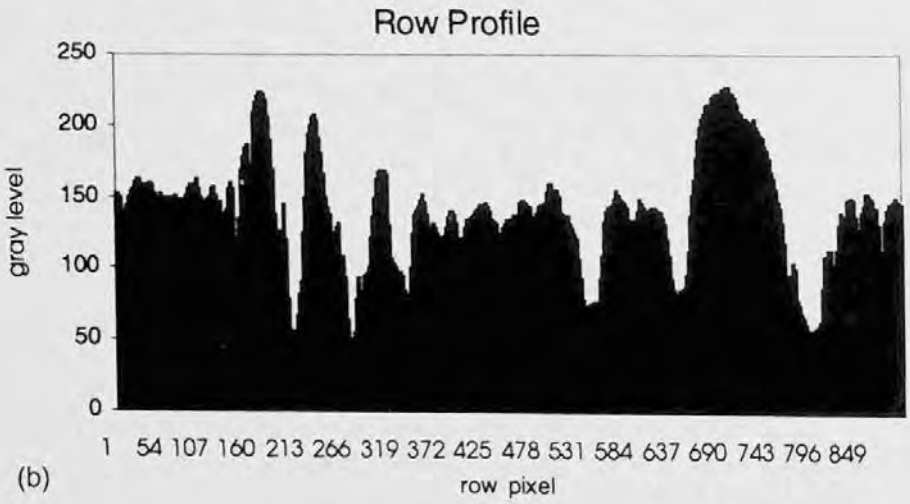
The Fourier transform of autocorrelation, the power spectrum, is known as another important function in studying statistical behavior. Figure 1.4 shows a typical power spectrum of a television signal (Fink, 1957; Connor et al., 1972). It is reported that the spectrum is quite flat until 30 kHz for a broadcast TV signal. Beyond this line frequency the spectrum starts to drop at a rate of around 6 dB per octave. This reveals the heavy concentration of video signals in low frequencies, considering a nominal bandwidth of 5 MHz.

Spatial redundancy implies that the intensity value of a pixel can be *guessed* from that of its neighboring pixels. In other words, it is not necessary to represent each pixel in an image frame independently. Instead, one can predict a pixel from its neighbors. Predictive coding, also known as differential coding, is based on this observation and is discussed in Chapter 3. The direct consequence of recognition of spatial redundancy is that by removing a large amount of the redundancy (or utilizing the high correlation) within an image frame, we may save a lot of data in representing the frame, thus achieving data compression.

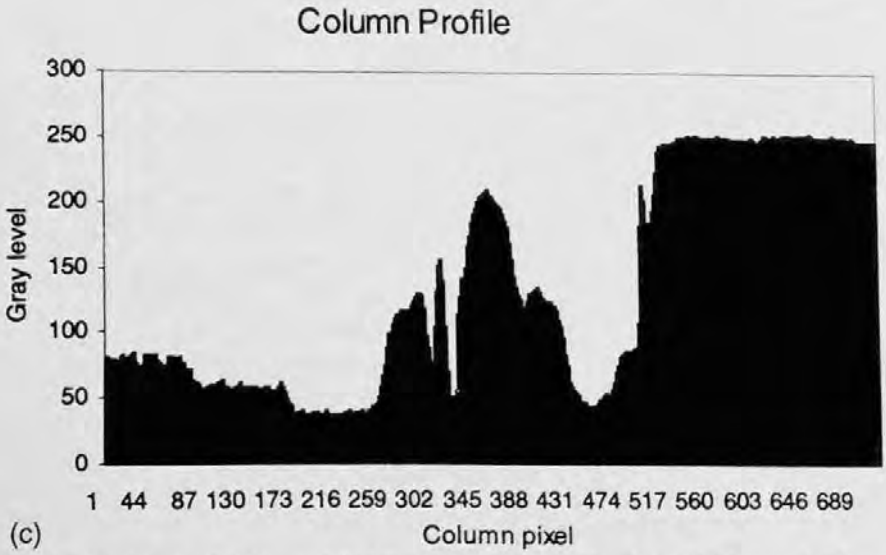
1.2.1.2 Temporal Redundancy

Temporal redundancy is concerned with the statistical correlation between pixels from successive frames in a temporal image or video sequence. Therefore, it is also called interframe redundancy.

Consider a temporal image sequence. That is, a camera is fixed in the 3-D world and it takes pictures of the scene one by one as time goes by. As long as the time interval between two consecutive pictures is short enough, i.e., the pictures are taken densely enough, we can imagine that the similarity between two neighboring frames is strong. Figures 1.5(a) and (b) show, respectively,



(b)



(c)

FIGURE 1.2 (a) A picture of "Boy and Girl," (b) Intensity profile along 318th row, (c) Intensity profile along 262nd column, (d) Positions of 318th row and 262nd column.

FIGURE 1.2 (continued)

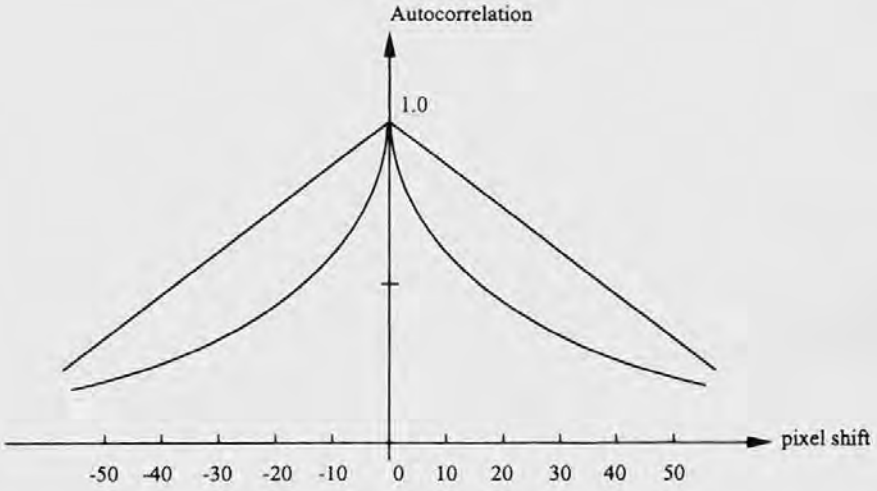


FIGURE 1.3 Autocorrelation in the horizontal direction for some pictures. (After Kretzmer, 1952.)

the 21st and 22nd frames of the "Miss America" sequence. The frames have a resolution of 176 by 144. Among the total of 25,344 pixels, only 3.4% change their gray value by more than 1% of the maximum gray value (255 in this case) from the 21st frame to the 22nd frame. This confirms an observation made in (Mounts, 1969): for a videophone-like signal with moderate motion in the scene, on average, less than 10% of pixels change their gray values between two consecutive frames by an amount of 1% of the peak signal. The high interframe correlation was reported in (Kretzmer, 1952). There, the autocorrelation between two adjacent frames was measured for two typical motion-picture films. The measured autocorrelations are 0.80 and 0.86. In summary, pixels within successive frames usually bear a strong similarity or correlation.

As a result, we may predict a frame from its neighboring frames along the temporal dimension. This is referred to as interframe predictive coding and is discussed in Chapter 3. A more precise, hence, more efficient interframe predictive coding scheme, which has been in development since

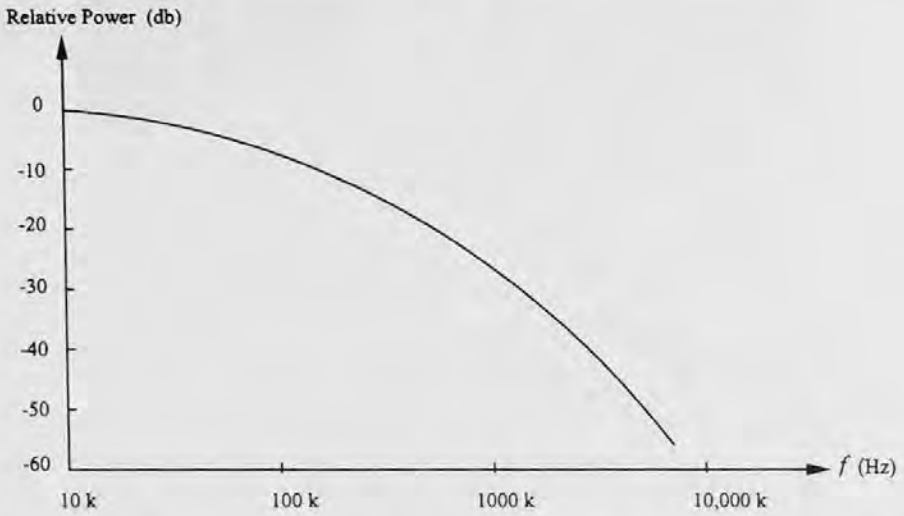


FIGURE 1.4 Typical power spectrum of a TV broadcast signal. (Adapted from Fink, D.G., *Television Engineering Handbook*, McGraw-Hill, New York, 1957.)

FIGURE 1.5 (a) The 21st frame, and (b) 22nd frame of the "Miss America" sequence.

the 1980s, uses motion analysis. That is, it considers that the changes from one frame to the next are mainly due to the motion of some objects in the frame. Taking this motion information into consideration, we refer to the method as motion compensated predictive coding. Both interframe correlation and motion compensated predictive coding are covered in detail in Chapter 10.

Removing a large amount of temporal redundancy leads to a great deal of data compression. At present, all the international video coding standards have adopted motion compensated predictive coding, which has been a vital factor to the increased use of digital video in digital media.

1.2.1.3 Coding Redundancy

As we discussed, interpixel redundancy is concerned with the correlation between pixels. That is, some information associated with pixels is redundant. The psychovisual redundancy, which is discussed in the next subsection, is related to the information that is psychovisually redundant, i.e., to which the HVS is not sensitive. Hence, it is clear that both the interpixel and psychovisual redundancies are somehow associated with some information contained in the image and video. Eliminating these redundancies, or utilizing these correlations, by using fewer bits to represent the

information results in image and video data compression. In this sense, the coding redundancy is different. It has nothing to do with information redundancy but with the representation of information, i.e., coding itself. To see this, let us take a look at the following example.

TABLE 1.1
An Illustrative Example

Symbol	Occurrence Probability	Code 1	Code 2
a_1	0.1	000	0000
a_2	0.2	001	01
a_3	0.5	010	1
a_4	0.05	011	0001
a_5	0.15	100	001

One illustrative example is provided in Table 1.1. The first column lists five distinct symbols that need to be encoded. The second column contains occurrence probabilities of these five symbols. The third column lists code 1, a set of codewords obtained by using uniform-length codeword assignment. (This code is known as the natural binary code.) The fourth column shows code 2, in which each codeword has a variable length. Therefore, code 2 is called the variable-length code. It is noted that the symbol with a higher occurrence probability is encoded with a shorter length. Let us examine the efficiency of the two different codes. That is, we will examine which one provides a shorter average length of codewords. It is obvious that the average length of codewords in code 1, $L_{avg,1}$, is three bits. The average length of codewords in code 2, $L_{avg,2}$, can be calculated as follows.

$$L_{avg,2} = 4 \times 0.1 + 2 \times 0.2 + 1 \times 0.5 + 4 \times 0.05 + 3 \times 0.15 = 1.95 \ \textit{bits per symbol} \qquad (1.1)$$

Therefore, it is concluded that code 2 with variable-length coding is more efficient than code 1 with natural binary coding.

From this example, we can see that for the same set of symbols different codes may perform differently. Some may be more efficient than others. For the same amount of information, code 1 contains some redundancy. That is, some data in code 1 are not necessary and can be removed without any effect. Huffman coding and arithmetic coding, two variable-length coding techniques, will be discussed in Chapter 5.

From the study of coding redundancy, it is clear that we should search for more efficient coding techniques in order to compress image and video data.

1.2.2 Psychovisual Redundancy

While interpixel redundancy inherently rests in image and video data, psychovisual redundancy originates from the characteristics of the human visual system (HVS).

It is known that the HVS perceives the outside world in a rather complicated way. Its response to visual stimuli is not a linear function of the strength of some physical attributes of the stimuli, such as intensity and color. HVS perception is different from camera sensing. In the HVS, visual information is not perceived equally; some information may be more important than other information. This implies that if we apply fewer data to represent less important visual information, perception will not be affected. In this sense, we see that some visual information is psychovisually redundant. Eliminating this type of psychovisual redundancy leads to data compression.

In order to understand this type of redundancy, let us study some properties of the HVS. We may model the human vision system as a cascade of two units (Lim, 1990), as depicted in Figure 1.6.

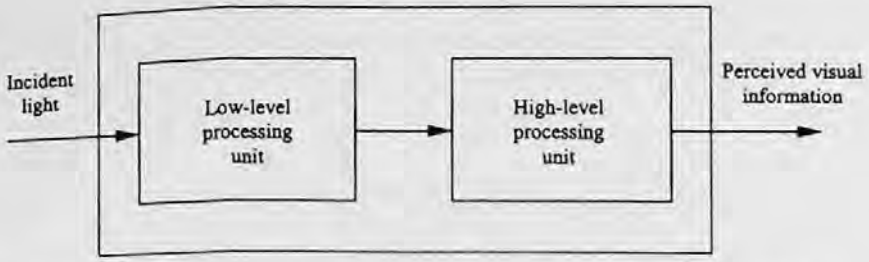


FIGURE 1.6 A two-unit cascade model of the human visual system (HVS).

The first one is a low-level processing unit which converts incident light into a neural signal. The second one is a high-level processing unit, which extracts information from the neural signal. While much research has been carried out to investigate low-level processing, high-level processing remains wide open. The low-level processing unit is known as a nonlinear system (approximately logarithmic, as shown below). While a great body of literature exists, we will limit our discussion only to video compression-related results. That is, several aspects of the HVS which are closely related to image and video compression are discussed in this subsection. They are luminance masking, texture masking, frequency masking, temporal masking, and color masking. Their relevance in image and video compression is addressed. Finally, a summary is provided in which it is pointed out that all of these features can be unified as one: differential sensitivity. This seems to be the most important feature of human visual perception.

1.2.2.1 Luminance Masking

Luminance masking concerns the brightness perception of the HVS, which is the most fundamental aspect among the five to be discussed here. Luminance masking is also referred to as *luminance dependence* (Connor et al., 1972), and *contrast masking* (Legge and Foley, 1980, Watson, 1987). As pointed in (Legge and Foley, 1980), the term *masking* usually refers to a destructive interaction or interference among stimuli that are closely coupled in time or space. This may result in a failure in detection, or errors in recognition. Here, we are mainly concerned with the detectability of one stimulus when another stimulus is present simultaneously. The effect of one stimulus on the detectability of another, however, does not have to decrease detectability. Indeed, there are some cases in which a low-contrast masker increases the detectability of a signal. This is sometimes referred to as *facilitation*, but in this discussion we only use the term masking.

Consider the monochrome image shown in Figure 1.7. There, a uniform disk-shaped object with a gray level (intensity value) I_1 is imposed on a uniform background with a gray level I_2. Now the question is under what circumstances can the disk-shaped object be discriminated from the background by the HVS? That is, we want to study the effect of one stimulus (the background in this example, the masker) on the detectability of another stimulus (in this example, the disk). Two extreme cases are obvious. That is, if the difference between the two gray levels is quite large, the HVS has no problem with discrimination, or in other words the HVS notices the object from the background. If, on the other hand, the two gray levels are the same, the HVS cannot identify the existence of the object. What we are concerned with here is the critical threshold in the gray level difference for discrimination to take place.

If we define the threshold ΔI as such a gray level difference $\Delta I = I_1 - I_2$ that the object can be noticed by the HVS with a 50% chance, then we have the following relation, known as *contrast sensitivity function*, according to Weber's law:

$$\frac{\Delta I}{I} \approx constant \tag{1.2}$$

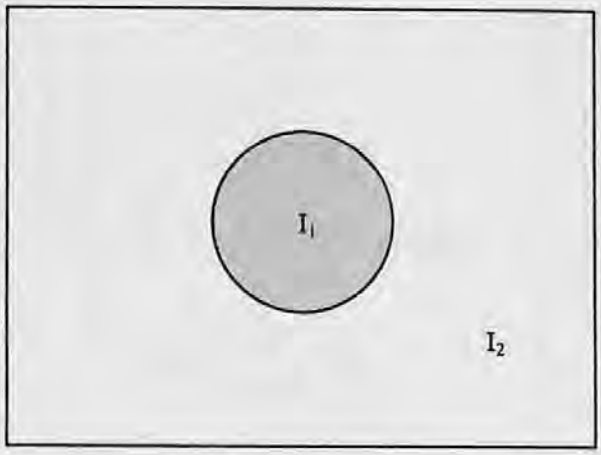


FIGURE 1.7 A uniform object with gray level I_1 imposed on a uniform background with gray level I_2.

where the constant is about 0.02. Weber's law states that for a relatively very wide range of I, the threshold for discrimination, ΔI, is directly proportional to the intensity I. The implication of this result is that when the background is bright, a larger difference in gray levels is needed for the HVS to discriminate the object from the background. On the other hand, the intensity difference required could be smaller if the background is relatively dark. It is noted that Equation 1.2 implies a logarithmic response of the HVS, and Weber's law holds for all other human senses as well.

Further research has indicated that the luminance threshold ΔI increases more slowly than is predicted by Weber's law. Some more accurate contrast sensitivity functions have been presented in the literature. In (Legge and Foley, 1980), it was reported that an exponential function replaces the linear relation in Weber's law. The following exponential expression is reported in (Watson, 1987).

$$\Delta I = I_0 \cdot \max\left\{1, \left(\frac{I}{I_0}\right)^{\alpha}\right\}, \tag{1.3}$$

where I_0 is the luminance detection threshold when the gray level of the background is equal to zero, i.e., $I = 0$, and α is a constant, approximately equal to 0.7.

Figure 1.8 shows a picture uniformly corrupted by additive white Gaussian noise (AWGN). It can be observed that the noise is more visible in the dark areas than in the bright areas if comparing, for instance, the dark portion and the bright portion of the cloud above the bridge. This indicates that noise filtering is more necessary in the dark areas than in the bright areas. The lighter areas can accommodate more additive noise before the noise becomes visible. This property has found application in embedding digital watermarks (Huang and Shi, 1998).

The direct impact that luminance masking has on image and video compression is related to quantization, which is covered in detail in the next chapter. Roughly speaking, quantization is a process that converts a continuously distributed quantity into a set of many finitely distinct quantities. The number of these distinct quantities (known as quantization levels) is one of the keys in quantizer design. It significantly influences the resulting bit rate and the quality of the reconstructed image and video. An effective quantizer should be able to minimize the visibility of quantization error. The contrast sensitivity function provides a guideline in analysis of the visibility of quantization error. Therefore, it can be applied to quantizer design. Luminance masking suggests a nonuniform quantization scheme that takes the contrast sensitivity function into consideration. One such example was presented in (Watson, 1987).

FIGURE 1.8 The Burrard bridge in Vancouver. (a) Original picture (courtesy of Minhuai Shi). (b) Picture uniformly corrupted by additive white Gaussian noise.

1.2.2.2 Texture Masking

Texture masking is sometimes also called *detail dependence* (Connor et al., 1972), *spatial masking* (Netravali and Presada, 1977; Lim, 1990), or *activity masking* (Mitchell et al., 1997). It states that the discrimination threshold increases with increasing picture detail. That is, the stronger the texture, the larger the discrimination threshold. In Figure 1.8, it can be observed that the additive random noise is less pronounced in the strongly textured area than in the smooth area if comparing, for instance, the dark portion of the cloud (the upper-right corner of the picture) with the water area (the lower right corner of the picture). This is a confirmation of texture masking.

In Figure 1.9(b), the number of quantization levels decreases from 256, as in Figure 1.9(a), to 16. That is, we use only four bits instead of eight bits to represent the intensity value for each pixel.

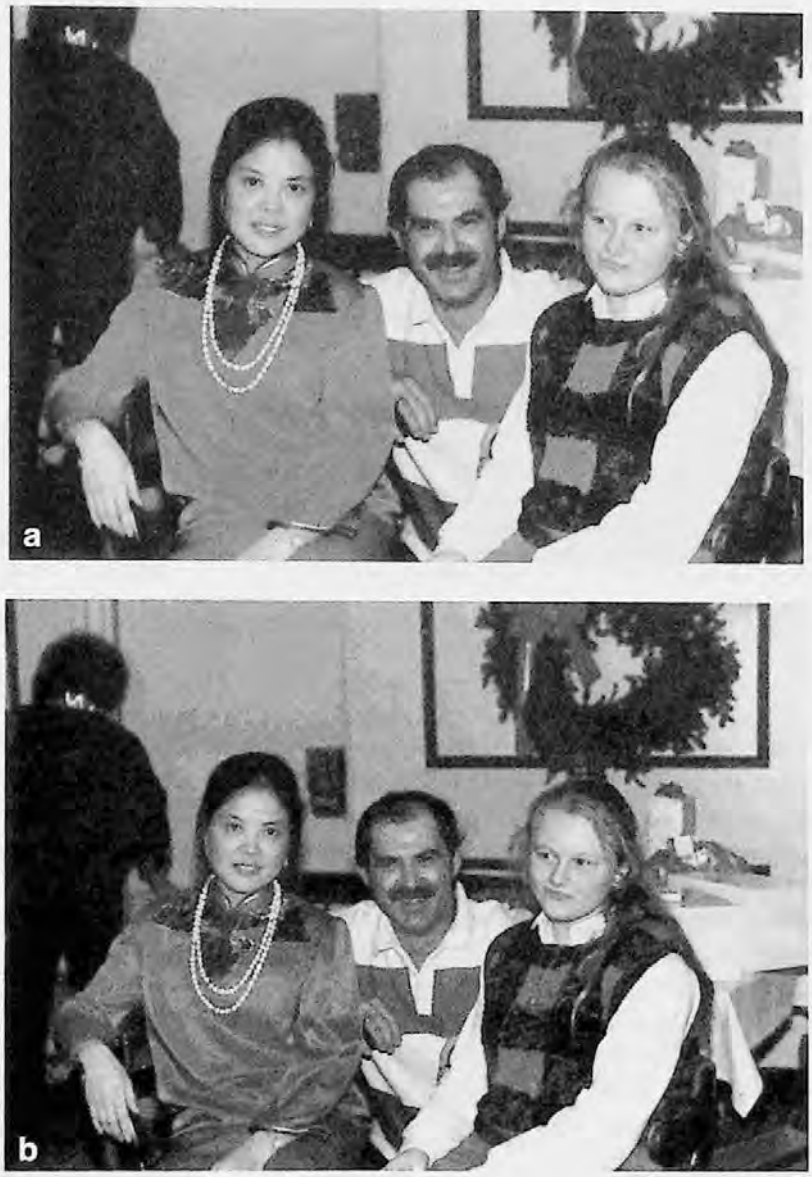


FIGURE 1.9 Christmas in Winorlia. (a) Original. (b) Four-bit quantized. (c) Improved IGS quantized with four bits.

The unnatural contours caused by coarse quantization can be noticed in the relatively uniform regions, compared with Figure 1.9(a). This phenomenon was first noted in (Goodall, 1951) and is called *false contouring* (Gonzalez and Woods, 1992). Now we see that the false contouring can be explained by using texture masking, since texture masking indicates that the human eye is more sensitive to the smooth region than to the textured region, where intensity exhibits a high variation. A direct impact on image and video compression is that the number of quantization levels, which affects the bit rate significantly, should be adapted according to the intensity variation of image regions.

1.2.2.3 Frequency Masking

While the above two characteristics are picture dependent in nature, frequency masking is picture independent. It states that the discrimination threshold increases with frequency increase. It is also referred to as *frequency dependence*.

FIGURE 1.9 (continued)

Frequency masking can be well illustrated in using Figure 1.9 above. In Figure 1.9(c), high-frequency random noise has been added to the original image before quantization. This method is referred to as the improved gray-scale (IGS) quantization (Gonzalez and Woods, 1992). With the same number of quantization levels, 16, as in Figure 1.9(b), the picture quality of Figure 1.9(c) is improved dramatically compared with that of Figure 1.9(b): the annoying false contours have disappeared despite an increase in the root mean square value of the total noise in Figure 1.9(c). This is due to the fact that the low-frequency quantization error is converted to the high-frequency noise, and that the HVS is less sensitive to the high-frequency content. We thus see, as pointed out in (Connor, 1972), that our human eyes function like a low-pass filter.

Owing to frequency masking in the transform domain, say, the discrete cosine transform (DCT) domain, we can drop some high-frequency coefficients with small magnitudes to achieve data compression without noticeably affecting the perception of the HVS. This leads to what is called transform coding, which is discussed in Chapter 4.

1.2.2.4 Temporal Masking

Temporal masking is another picture-independent feature of the HVS. It states that it takes a while for the HVS to adapt itself to the scene when the scene changes abruptly. During this transition the HVS is not sensitive to details. The masking takes place both before and after the abrupt change. It is called forward temporal masking if it happens after the scene change. Otherwise, it is referred to backward temporal masking (Mitchell et al., 1997).

This implies that one should take temporal masking into consideration when allocating data in image and video coding.

1.2.2.5 Color Masking

Digital color image processing is gaining increasing popularity due to the wide application of color images in modern life. As mentioned at the beginning of the discussion about psychovisual redundancy, we are not going to cover all aspects of the perception of the HVS. Instead, we cover only those aspects related to psychovisual redundancy, thus to image and video compression. Therefore, our discussion here on color perception is by no means exhaustive.

In physics, it is known that any visible light corresponds to an electromagnetic spectral distribution. Therefore, a color, as a sensation of visible light, is an energy with an intensity as well as a set of wavelengths associated with the electromagnetic spectrum. Obviously, intensity is an attribute of visible light. The composition of wavelengths is another attribute: chrominance. There are two elements in the chrominance attribute: *hue* and *saturation*. The hue of a color is characterized by the dominant wavelength in the composition. Saturation is a measure of the purity of a color. A pure color has a saturation of 100%, whereas white light has a saturation of 0.

RGB model — The red-green-blue (RGB) primary color system is the best known of several color systems. This is due to the following feature of the human perception of color. The color-sensitive area in the HVS consists of three different sets of cones and each set is sensitive to the light of one of the three primary colors: red, green, and blue. Consequently, any color sensed by the HVS can be considered as a particular linear combination of the three primary colors. Many research studies are available, the CIE (Commission Internationale de l'Eclairage) chromaticity diagram being a well-known example. These results can be easily found in many classic optics and digital image processing texts.

The RGB model is used mainly in color image acquisition and display. In color signal processing including image and video compression, however, the luminance-chrominance color system is more efficient and, hence, widely used. This has something to do with the color perception of the HVS. It is known that the HVS is more sensitive to green than to red, and is least sensitive to blue. An equal representation of red, green, and blue leads to inefficient data representation when the HVS is the ultimate viewer. Allocating data only to the information that the HVS can perceive, on the other hand, can make video coding more efficient.

Luminance is concerned with the perceived brightness, while chrominance is related to the perception of hue and saturation of color. Roughly speaking, the luminance-chrominance representation agrees more with the color perception of the HVS. This feature makes the luminance-chrominance color models more suitable for color image processing. A good example was presented in (Gonzalez and Woods, 1992), about histogram equalization. It is well known that applying histogram equalization can bring out some details originally hidden in dark regions. Applying histogram equalization to the RGB components separately can certainly achieve the goal. In doing so, however, the chrominance elements, hue and saturation, have been changed, thus leading to color distortion. With a luminance-chrominance model, histogram equalization can be applied to the luminance component only. Hence, the details in the dark regions are brought out, whereas the chrominance elements remain unchanged, producing no color distortion. With the luminance component Y serving as a black-white signal, a luminance-chrominance color model offers compatibility with black and white TV systems. This is another virtue of luminance-chrominance color models.

It is known that a nonlinear relationship (basically a power function) exists between electrical signal magnitude and light intensity for both cameras and CRT-based display monitors (Haskell et al., 1997). That is, the light intensity is a linear function of the signal voltage raised to the power of γ. It is a common practice to correct this nonlinearity before transmission. This is referred to as gamma correction. The gamma-corrected RGB components are denoted by R′, G′, and B′, respectively. They are used in the discussion on various color models. For the sake of notational brevity, we simply use R, G, and B instead of R′, G′ and B′ in the following discussion, while keeping the gamma-correction in mind.

Discussed next are several different luminance-chrominance color models: HSI, YUV, YCbCr, and YDbDr.

HSI model — In this model, I stands for the intensity component, H for the hue component, and S for saturation. One merit of this color system is that the intensity component is decoupled from the chromatic components. As analyzed above, this decoupling usually facilitates color image processing tasks. Another merit is that this model is closely related to the way the HVS perceives color pictures. Its main drawback is the complicated conversion between RGB and HSI models. A

detailed derivation of the conversion may be found in (Gonzalez and Woods, 1992). Because of this complexity, the HSI model is not used in any TV systems.

YUV model — In this model, Y denotes the luminance component, and U and V are the two chrominance components. The luminance Y can be determined from the RGB model via the following relation:

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \tag{1.4}$$

It is noted that the three weights associated with the three primary colors, R, G, and B, are not the same. Their different magnitudes reflect different responses of the HVS to different primary colors.

Instead of being directly related to hue and saturation, the other two chrominance components, U and V, are defined as color differences as follows.

$$U = 0.492(B - Y) \tag{1.5}$$

$$V = 0.877(R - Y) \tag{1.6}$$

In this way, the YUV model lowers computational complexity. It has been used in PAL (Phase Alternating Line) TV systems. Note that PAL is an analog composite color TV standard and is used in most European countries, some Asian countries, and Australia. By composite systems, we mean both the luminance and chrominance components of the TV signals are multiplexed within the same channel. For completeness, an expression of YUV in terms of RGB is listed below.

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{1.7}$$

YIQ model — This color space has been utilized in NTSC (National Television Systems Committee) TV systems for years. Note that NTSC is an analog composite color TV standard and is used in North America and Japan. The Y component is still the luminance. The two chrominance components are the linear transformation of the U and V components defined in the YUV model. Specifically,

$$I = -0.545U + 0.839V \tag{1.8}$$

$$Q = 0.839U + 0.545V \tag{1.9}$$

Substituting the U and V expressed in Equations 1.4 and 1.5 into the above two equations, we can express YIQ directly in terms of RGB. That is,

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{1.10}$$

YDbDr model — The YDbDr model is used in the SECAM (Sequential Couleur a Memoire) TV system. Note that SECAM is used in France, Russia, and some eastern European countries. The relationship between YDbDr and RGB appears below.

$$\begin{pmatrix} Y \\ Db \\ Dr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.450 & -0.883 & 1.333 \\ -1.333 & 1.116 & -0.217 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{1.11}$$

That is,

$$Db = 3.059U \tag{1.12}$$

$$Dr = -2.169V \tag{1.13}$$

YCbCr model — From the above, we can see that the U and V chrominance components are differences between the gamma-corrected color B and the luminance Y, and the gamma-corrected R and the luminance Y, respectively. The chrominance component pairs I and Q, and Db and Dr are both linear transforms of U and V. Hence they are very closely related to each other. It is noted that U and V may be negative as well. In order to make chrominance components nonnegative, the Y, U, and V are scaled and shifted to produce the YCbCr model, which is used in the international coding standards JPEG and MPEG. (These two standards are covered in Chapters 7 and 16, respectively.)

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 16 \\ 128 \\ 128 \end{pmatrix} \tag{1.14}$$

1.2.2.6 Color Masking and Its Application in Video Compression

It is well-known that the HVS is much more sensitive to the luminance component than to the chrominance components. Following Van Ness and Bouman (1967) and Mullen (1985), there is a figure in Mitchell et al. (1997) to quantitatively illustrate the above statement. A modified version is shown in Figure 1.10. There, the abscissa represents spatial frequency in the unit of cycles per degree (cpd), while the ordinate is the contrast threshold for a just detectable change in the sinusoidal testing signal. Two observations are in order. First, for each of the three curves, i.e., curves for the luminance component Y and the opposed-color chrominance, the contrast sensitivity increases when spatial frequency increases, in general. This agrees with frequency masking discussed above. Second, for the same contrast threshold, we see that the luminance component corresponds to a much higher spatial frequency. This is an indication that the HVS is much more sensitive to luminance than to chrominance. This statement can also be confirmed, perhaps more easily, by examining those spatial frequencies at which all three curves have data available. Then we can see that the contrast threshold of luminance is much lower than that of the chrominance components.

The direct impact of color masking on image and video coding is that by utilizing this psychovisual feature we can allocate more bits to the luminance component than to the chrominance components. This leads to a common practice in color image and video coding: using full resolution for the intensity component, while using a 2 by 1 subsampling both horizontally and vertically for the two chrominance components. This has been adopted in related international coding standards, which will be discussed in Chapter 16.

1.2.2.7 Summary: Differential Sensitivity

In this subsection we discussed luminance masking, texture masking, frequency masking, temporal masking, and color masking. Before we enter the next subsection, let us summarize what we have discussed so far.

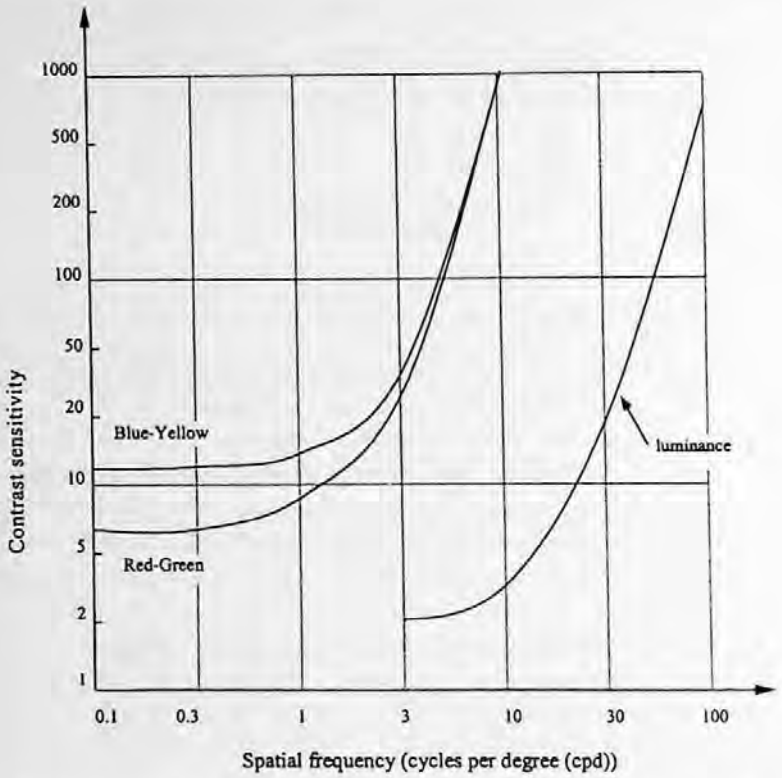


FIGURE 1.10 Contrast sensitivity vs. spatial frequency. (Revised from Van Ness and Bouman [1967] and Mullen [1985].)

We see that luminance masking, also known as contrast masking, is of fundamental importance among several types of masking. It states that the sensitivity of the eyes to a stimulus depends on the intensity of another stimulus. Thus it is a differential sensitivity. Both the texture (detail or activity) and frequency of another stimulus significantly influence this differential sensitivity. The same mechanism exists in color perception, where the HVS is much more sensitive to luminance than to chrominance. Therefore, we conclude that differential sensitivity is the key in studying human visual perception. These features can be utilized to eliminate psychovisual redundancy, and thus compress image and video data.

It is also noted that variable quantization, which depends on activity and luminance in different regions, seems to be reasonable from a data compression point of view. Its practical applicability, however, is somehow questionable. That is, some experimental work does not support this expectation (Mitchell et al., 1997).

It is noted that this differential sensitivity feature of the HVS is common to human perception. For instance, there is also forward and backward temporal masking in human audio perception.

1.3 VISUAL QUALITY MEASUREMENT

As the definition of image and video compression indicates, image and video quality is an important factor in dealing with image and video compression. For instance, in evaluating two different compression methods we have to base the evaluation on some definite image and video quality. When both methods achieve the same quality of reconstructed image and video, the one that requires less data is considered to be superior to the other. Alternatively, with the same amount of data the method providing a higher-quality reconstructed image or video is considered the better method. Note that here we have not considered other performance criteria, such as computational complexity.

Surprisingly, however, it turns out that the measurement of image and video quality is not straightforward. There are two types of visual quality assessments. One is objective assessment (using electrical measurements), and the other is subjective assessment (using human observers). Each has its merits and drawbacks. A combination of these two methods is now widely utilized in practice. In this section we first discuss subjective visual quality measurement, followed by objective quality measurement.

1.3.1 Subjective Quality Measurement

It is natural that the visual quality of reconstructed video frames should be judged by human viewers if they are to be the ultimate receivers of the data (see Figure 1.1). Therefore, the subjective visual quality measure plays an important role in visual communications.

In subjective visual quality measurement, a set of video frames is generated with varying coding parameters. Observers are invited to subjectively evaluate the visual quality of these frames. Specifically, observers are asked to rate the pictures by giving some measure of picture quality. Alternatively, observers are requested to provide some measure of impairment to the pictures. A five-scale rating system of the degree of impairment, used by Bell Laboratories, is listed below (Sakrison, 1979). It has been adopted as one of the standard scales in CCIR Recommendation 500-3 (CCIR, 1986). Note that CCIR is now ITU-R (International Telecommunications Union — Recommendations).

1. Impairment is not noticeable
2. Impairment is just noticeable
3. Impairment is definitely noticeable, but not objectionable
4. Impairment is objectionable
5. Impairment is extremely objectionable

In regard to the subjective evaluation, there are a few things worth mentioning. In most applications there is a whole array of pictures simultaneously available for evaluation. These pictures are generated with different encoding parameters. By keeping some parameters fixed while making one parameter (or a subset of parameters) free to change, the resulting quality rating can be used to study the effect of the one parameter (or the subset of parameters) on encoding. An example using this method for studying the effect of varying numbers of quantization levels on image quality can be found in (Gonzalez and Woods, 1992).

Another possible way to study the effect is to identify pictures with the same subjective quality measure from the whole array of pictures. From this subset of test pictures we can produce, in the encoding parameter space, isopreference curves that can be used to study the effect of the parameter(s) under investigation. An example using this method to study the effect of varying both image resolution and numbers of quantization levels on image quality can be found in (Huang, 1965).

In this rating, a whole array of pictures is usually divided into columns, with each column sharing some common conditions. The evaluation starts within each column with a pairwise comparison. This is because a pairwise comparison is relatively easy for the eyes. As a result, pictures in one column are arranged in an order according to visual quality, and quality or impairment measures are then assigned to the pictures in that one column. After each column has been rated, a unification between columns is necessary. That is, different columns need to have a unified quality measurement. As pointed out in (Sakrison, 1979), this task is not easy since it means we may need to equate impairment that results from different types of errors.

One thing can be understood from the above discussion: subjective evaluation of visual quality is costly. It needs a large number of pictures and observers. The evaluation takes a long time because human eyes are easily fatigued and bored. Some special measures have to be taken in order to arrive at an accurate subjective quality measurement. Examples in this regard include averaging

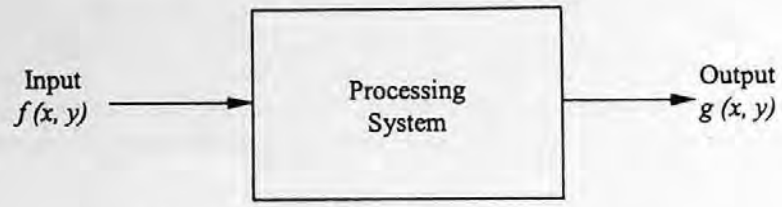


FIGURE 1.11 An image processing system.

subjective ratings and taking their deviation into consideration. For further details on subjective visual quality measurement, readers may refer to Sakrison (1979), Hidaka and Ozawa (1990), and Webster et al. (1993).

1.3.2 Objective Quality Measurement

In this subsection, we first introduce the concept of signal-to-noise ratio (SNR), which is a popularly utilized objective quality assessment. Then we present a promising new objective visual quality assessment technique based on human visual perception.

1.3.2.1 Signal-to-Noise Ratio

Consider Figure 1.11, where $f(x, y)$ is the input image to a processing system. The system can be a low-pass filter, a subsampling system, or a compression system. It can even represent a process in which additive white Gaussian noise corrupts the input image. The $g(x, y)$ is the output of the system. In evaluating the quality of $g(x, y)$, we define an error function $e(x, y)$ as the difference between the input and the output. That is,

$$e(x,y) = f(x,y) - g(x,y) \tag{1.15}$$

The mean square error is defined as E_{ms}:

$$E_{ms} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} e(x,y)^2 \tag{1.16}$$

where M and N are the dimensions of the image in the horizontal and vertical directions. Note that it is sometimes denoted by *MSE*. The root mean square error is defined as E_{rms}:

$$E_{rms} = \sqrt{E_{ms}} \tag{1.17}$$

It is sometimes denoted by *RMSE*.

As noted earlier, *SNR* is widely used in objective quality measurement. Depending whether mean square error or root mean square error is used, the *SNR* may be called the mean square signal-to-noise ratio, SNR_{ms}, or the root mean square signal-to-noise ratio, SNR_{rms}. We have

$$SNR_{ms} = 10\log_{10}\left(\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} g(x,y)^2}{MN \cdot E_{ms}}\right), \tag{1.18}$$

and

$$SNR_{rms} = \sqrt{SNR_{ms}} \tag{1.19}$$

In image and video data compression, another closely related term, *PSNR* (peak signal-to-noise ratio), which is essentially a modified version of SNR_{ms}, is widely used. It is defined as follows.

$$PSNR = 10 \log_{10} \left(\frac{255^2}{E_{ms}} \right) \tag{1.20}$$

The interpretation of the *SNRs* is that the larger the *SNR* (SNR_{ms}, SNR_{rms}, or *PSNR*) the better the quality of the processed image, $g(x, y)$; that is, the closer the processed image $g(x,y)$ is to the original image $f(x, y)$. This seems to be correct. However, from our above discussion about the features of the HVS, we know that the HVS does not respond to visual stimuli in a straightforward way. Its low-level processing unit is known to be nonlinear. Several masking phenomena exist. Each confirms that the visual perception of the HVS is not simple. It is worth noting that our understanding of the high-level processing unit of the HVS is far from complete. Therefore, we may understand that the *SNR* does not always provide us with reliable assessments of image quality. One good example is presented in Section 1.2.2.3, which uses the IGS quantization technique to achieve high compression (using only four bits for quantization instead of the usual eight bits) without introducing noticeable false contouring. In this case, the subjective quality is improved, and the *SNR* decreases due to the additive high-frequency random noise. Another example, drawn from our discussion about the masking phenomena, is that some additive noise in bright areas or in highly textured regions may be masked, while some minor artifacts in dark and uniform regions may turn out to be quite annoying. In this case, the *SNR* cannot truthfully reflect visual quality, as well.

On the one hand, we see that objective quality measurement does not always provide reliable picture quality assessment. On the other hand, however, its implementation is much faster and easier than that of the subjective quality measurement. Furthermore, objective assessment is repeatable. Owing to these merits, objective quality assessment is still widely used despite this drawback.

It is noted that combining subjective and objective assessments has been a common practice in international coding-standard activity.

1.3.2.2 Objective Quality Measurement Based on Human Visual Perception

Introduced here is a new development in visual quality assessment, which is an objective quality measurement based on human visual perception (Webster et al., 1993). Since it belongs to the category of objective assessment, it possesses virtues such as repeatability and fast and easy implementation. Because it is based on human visual perception, on the other hand, its assessment of visual quality agrees closely to that of subjective assessment. In this sense the new method attempts to combine the merits of the two different types of assessment.

Motivation — Visual quality assessment is best conducted via the subjective approach since in this case the HVS is the ultimate viewer. The implementation of subjective assessment, however, is time-consuming, costly, and lacks repeatability. On the other hand, although not always accurate, objective assessment is fast, easy, and repeatable. The motivation here is to develop an objective quality measurement system such that its quality assessment is very close to that obtained by using subjective assessment. In order to achieve this goal, this objective system is based on subjective assessment. That is, it uses the rating achieved via subjective assessment as a criterion to search for new objective measurements so as to have the objective rating as close to the subjective one as possible.

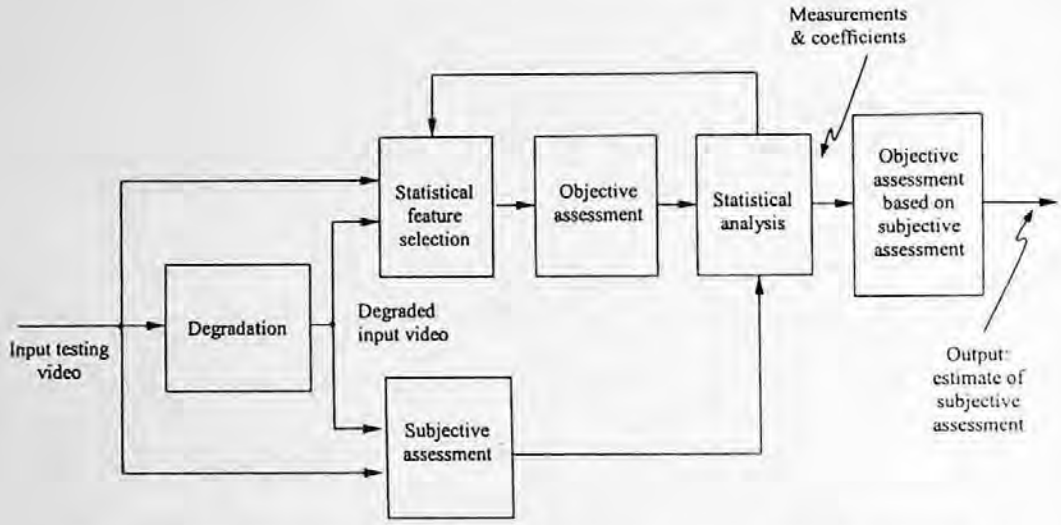


FIGURE 1.12 Block diagram of objective assessment based on subjective assessment.

Methodology — The derivation of the objective quality assessment system is shown in Figure 1.12. The input testing video goes through a degradation block, resulting in degraded input video. The degradation block, or impairment generator, includes various video compression codecs (coder-decoder pairs) with bit rates ranging from 56 kb/sec to 45 Mb/sec and other video operations. The input video and degraded input video form a pair of testing videos, which is sent to a subjective assessment block as well as a statistical feature selection block.

A normal subjective visual quality assessment as introduced in the previous subsection is performed in the subjective assessment block, which involves a large panel of observers, e.g., 48 observers in Webster et al. (1993). In the statistical feature selection block, a variety of statistical operations are conducted and various statistical features are selected. Examples cover Sobel filtering, Laplacian operator, first-order differencing, moment calculation, fast Fourier transform, etc. Statistical measurements are then selected based on these statistical operations and features. An objective assessment is formed as follows:

$$\hat{s} = a_0 + \sum_{i=1}^{l} a_i n_i , \tag{1.21}$$

where $\hat{s}$ denotes the output rating of the object assessment, or simply the objective measure, which is supposed to be a good estimate of the corresponding subjective score. The n_i, $i = 1, \cdots, l$ are selected objective measurements. The a_0, a_i, $i = 1, \cdots, l$ are coefficients in the linear model of the objective assessment.

The results of the objective assessment and subjective assessment are applied to a statistical analysis block. In the statistical analysis block, the objective assessment rating is compared with that of the subjective assessment. The result of the comparison is fed back to the statistical feature selection block. The statistical measurements obtained in the statistical feature selection block are examined according to their performance in the assessment. A statistical measurement is regarded to be good if it can reduce by a significant amount of the difference between the objective assessment and the subjective assessment. The best measurement is determined via an exhaustive search among the various measurements. Note that the coefficients in Equation 1.21 are also examined in the statistical analysis block in a similar manner to that used for the measurements.

The measurements and coefficients determined after iterations result in an optimal objective assessment via Equation 1.21, which is finally passed to the last block as the output of the system. The whole process will become much clearer below.

Results

The results reported by Webster (1993) are introduced here.

Information features — As mentioned in Section 1.2.2, differential sensitivity is a key in human visual perception. Two selected features: perceived spatial information (the amount of spatial detail) and perceived temporal information (the amount of temporal luminance variation), involve pixel differencing. Spatial information (SI) is defined as shown below.

$$SI(f_n) = STD_s\{Sobel(f_n)\} \tag{1.22}$$

where STD_s stands for the standard deviation operator in the spatial domain, *Sobel* denotes the Sobel operation, and f_n represents the nth video frame. Temporal information (TI) is defined similarly:

$$TI(f_n) = STD_s\{\Delta f_n\} \tag{1.23}$$

where $\Delta f_n = f_n - f_{n-1}$, i.e., the successive frame difference.

Determined Measurements

The parameter l in Equation 1.21 is chosen as three. That is

$$\hat{s} = a_0 + a_1 n_1 + a_2 n_2 + a_3 n_3 \tag{1.24}$$

The measurements n_1, n_2, and n_3 are formulated based on the above-defined information features, SI and TI, as follows:

1. Measurement n_1:

$$n_1 = RMS_t\left(5.81\left|\frac{SI(of_n) - SI(df_n)}{SI(of_n)}\right|\right) \tag{1.25}$$

   where RMS_t represents the root mean square value taken over the time dimension, and of_n and df_n denote the original nth frame and the degraded nth frame, respectively. It is observed that n_1 is a measure of the relative change in the spatial information between the original frame and the degraded frame.

2. Measurement n_2:

$$n_2 = \Im_t\left\{0.108 \cdot MAX\left\{\left[TI(of_n) - TI(df_n)\right], 0\right\}\right\} \tag{1.26}$$

   where

$$\Im_t\{y_t\} = STD_t\{CONV(y_t, [-1, 2, -1])\} \tag{1.27}$$

   where STD_t denotes the standard deviation operator with respect to time, and *CONV* indicates the convolution operation between its two arguments. It is understood that

temporal information, TI, measures temporal luminance variation (temporal motion) and the convolution kernel, [-1, 2, -1], enhances the variation due to its high-pass filter nature. Therefore, n_2 measures the difference of TI between the original and graded frames.

3. Measurement n_3:

$$n_3 = MAX_t\left\{4.23 \cdot \log_{10}\left(\frac{TI(df_n)}{TI(of_n)}\right)\right\} \tag{1.28}$$

where MAX_t indicates the taking of the maximum value over time. Therefore, measurement n_3 responds to the ratio between the temporal information of the degraded video and that of the original video. Distortions such as block artifacts (discussed in Chapter 11) and motion jerkiness (discussed in Chapter 10), which occur in video coding, will cause n_3 to be large.

Objective estimator — The least square error procedure is applied to testing video sequences with measurements n_i, $i = 1,2,3$, determined above, to minimize the difference between the rating scores obtained from the subjective assessment and the objective assessment, resulting in the estimated coefficients a_0 and a_i, $i = 1,2,3$. Consequently, the objective assessment of visual quality $\hat{s}$ becomes

$$\hat{s} = 4.77 - 0.992n_1 - 0.272n_2 - 0.356n_3 \tag{1.29}$$

Reported experimental results — It was reported that the correlation coefficient between the subjective assessment score and the objective assessment score (an estimate of the subjective score) is in the range of 0.92 to 0.94. It is noted that a set of 36 testing scenes containing various amounts of spatial and temporal information was used in the experiment. Hence, it is apparent that quite good performance was achieved. Though there is surely room for further improvement, this work does open a new and promising way to assess visual quality by combining subjective and objective approaches. Since it is objective it is fast and easy; and because it is based on the subjective measurement, it is more accurate in terms of the high correlation to human perception. Theoretically, the spatial information measure and temporal information measure defined on differencing are very important. They reflect the most important aspect of human visual perception.

1.4 INFORMATION THEORY RESULTS

In the beginning of this chapter it was noted that the term information is considered one of the fundamental concepts in image and video compression. We will now address some information theory results. In this section, the measure of information and the entropy of an information source are covered first. We then introduce some coding theorems, which play a fundamental role in studying image and video compression.

1.4.1 Entropy

Entropy is a very important concept in information theory and communications. So is it in image and video compression. We first define the information content of a source symbol. Then we define entropy as average information content per symbol for a discrete memoriless source.

1.4.1.1 Information Measure

As mentioned at the beginning of this chapter, information is defined as knowledge, fact, and news. It can be measured quantitatively. The carriers of information are symbols. Consider a symbol with

an occurrence probability p. Its information content (i.e., the amount of information contained in the symbol), I, is defined as follows.

$$I = \log_2 \frac{1}{p} \quad bits \qquad or \qquad I = -\log_2 p \quad bits \tag{1.30}$$

where the *bit* is a contraction of *binary unit*. In the above equations we set the base of the logarithmic function to equal 2. It is noted that these results can be easily converted as follows for the case where the r-ary digits are used for encoding. Hence, from now on, we restrict our discussion to binary encoding.

$$I = -\log_r 2 \cdot \log_2 p \quad bits \tag{1.31}$$

According to Equation 1.30, the information contained within a symbol is a logarithmic function of its occurrence probability. The smaller the probability, the more information the symbol contains. This agrees with common sense. The occurrence probability is somewhat related to the uncertainty of the symbol. A small occurrence probability means large uncertainty. In this way, we see that the information content of a symbol is about the uncertainty of the symbol. It is noted that the information measure defined here is valid for both equally probable symbols and nonequally probable symbols (Lathi, 1998).

1.4.1.2 Average Information per Symbol

Now consider a discrete memoriless information source. By discreteness, we mean the source is a countable set of symbols. By memoriless, we mean the occurrence of a symbol in the set is independent of that of its preceding symbol. Take a look at a source of this type that contains m possible symbols: $\{s_i, i = 1,2\cdots,m\}$. The corresponding occurrence probabilities are denoted by $\{p_i, i = 1,2,\cdots,m\}$. According to the discussion above, the information content of a symbol s_i, I_i, is equal to $I_i = -\log_2 p_i$ bits. Entropy is defined as the average information content per symbol of the source. Obviously, the entropy, H, can be expressed as follows.

$$H = -\sum_{i=1}^{m} p_i \log_2 p_i \quad bits \tag{1.32}$$

From this definition, we see that the entropy of an information source is a function of occurrence probabilities. It is straightforward to show that the entropy reaches the maximum when all symbols in the set are equally probable.

1.4.2 Shannon's Noiseless Source Coding Theorem

Consider a discrete, memoriless, stationary information source. In what is called source encoding, a *codeword* is assigned to each symbol in the source. The number of bits in the codeword is referred to as the length of the codeword. The average length of codewords is referred to as the bit rate, expressed in the unit of bits per symbol.

Shannon's noiseless source coding theorem states that for a discrete, memoriless, stationary information source, the minimum bit rate required to encode a symbol, on average, is equal to the entropy of the source. This theorem provides us with a lower bound in source coding. Shannon showed that the lower bound can be achieved when the *encoding delay* extends to infinity. By encoding delay, we mean the encoder waits and then encodes a certain number of symbols at once. Fortunately, with finite encoding delay, we can already achieve an average codeword length fairly

close to the entropy. That is, we do not have to actually sacrifice bit rate much to avoid long encoding delay, which involves high computational complexity and a large amount of memory space.

Note that the discreteness assumption is not necessary. We assume a discrete source simply because digital image and video are the focus in this book. Stationarity assumption is necessary in deriving the noiseless source coding theorem. This assumption may not be satisfied in practice. Hence, Shannon's theorem is a theoretical guideline only. There is no doubt, however, that it is a fundamental theoretical result in information theory.

In summary, the noiseless source coding theorem, Shannon's first theorem, which was published in his celebrated paper (Shannon, 1948), is concerned with the case where both the channel and the coding system are noise free. The aim under these circumstances is coding compactness. The more compact it is, the better the coding. This theorem specifies the lower bound, which is the source entropy, and how to reach this lower bound.

One way to evaluate the efficiency of a coding scheme is to determine its *efficiency* with respect to the lower bound, i.e., entropy. The efficiency η is defined as follows.

$$\eta = \frac{H}{L_{avg}} \tag{1.33}$$

where H is entropy, and L_{avg} denotes the average length of the codewords in the code. Since the entropy is the lower bound, the efficiency never exceeds the unity, i.e., $\eta \leq 1$. The same definition can be generalized to calculate the relative efficiency between two codes. That is

$$\eta = \frac{L_{avg,1}}{L_{avg,2}} \tag{1.34}$$

where $L_{avg,1}$ and $L_{avg,2}$ represent the average codeword length for code 1 and code 2, respectively. We usually put the larger of the two in the denominator, and η is called the efficiency of code 2 with respect to code 1. A complementary parameter of coding efficiency is coding *redundancy*, ζ, which is defined as

$$\zeta = 1 - \eta \tag{1.35}$$

1.4.3 Shannon's Noisy Channel Coding Theorem

If a code has an efficiency of $\eta = 1$, i.e., it reaches the lower bound of source encoding, then coding redundancy is $\zeta = 0$. Now consider a noisy transmission channel. In transmitting the coded symbol through the noisy channel, the received symbols may be erroneous due to the lack of redundancy. On the other hand, it is well known that by adding redundancy (e.g., parity check bits) some errors occurring during the transmission over the noisy channel may be corrected or identified. In the latter, the coded symbols are then resent. In this way, we see that adding redundancy may combat noise.

Shannon's noisy channel coding theorem states that it is possible to transmit symbols over a noisy channel without error if the bit rate is below a *channel capacity*, C. That is

$$R < C \tag{1.36}$$

where R denotes the bit rate. The channel capacity is determined by the noise and signal power.

In conclusion, the noisy channel coding theorem, Shannon's second theorem (Shannon, 1948), is concerned with a noisy, memoriless channel. By memoriless, we mean the channel output

corresponding to the current input is independent of the output corresponding to previous input symbols. Under these circumstances, the aim is reliable communication. To be error free, the bit rate cannot exceed channel capacity. That is, channel capacity sets an upper bound on the bit rate.

1.4.4 Shannon's Source Coding Theorem

As seen in the previous two subsections, the noiseless source coding theorem defines the lowest possible bit rate for noiseless source coding and noiseless channel transmission; whereas the noisy channel coding theorem defines the highest possible coding bit rate for error-free transmission. Therefore, both theorems work for reliable (no error) transmission. In this subsection, we continue to deal with discrete memoriless information sources, but we discuss the situation in which lossy coding is encountered. As a result, distortion of the information source takes place. For instance, quantization, which is covered in the next chapter, causes information loss. Therefore, it is concluded that if an encoding procedure involves quantization, then it is lossy coding. That is, errors occur during the coding process, even though the channel is error free. We want to find the lower bound of the bit rate for this case.

The source coding theorem (Shannon, 1948) states that for a given distortion D, there exists a rate distortion function $R(D)$ (Berger, 1971), which is the minimum bit rate required to transmit the source with distortion less than or equal to D. That is, in order to have distortion not larger than D, the bit rate R must satisfy the following condition:

$$R \geq R(D) \tag{1.37}$$

A more detailed discussion about this theorem and the rate distortion function is given in Chapter 15, when we introduce video coding.

1.4.5 Information Transmission Theorem

It is clear that by combining the noisy channel coding theorem and the source coding theorem we can derive the following relationship:

$$C \geq R(D) \tag{1.38}$$

This is called the information transmission theorem (Slepian, 1973). It states that if the channel capacity of a noisy channel, C, is larger than the rate distortion function $R(D)$, then it is possible to transmit an information source with distortion D over a noisy channel.

1.5 SUMMARY

In this chapter, we first discussed the necessity for image and video compression. It is shown that image and video compression becomes an enabling technique in today's exploding number of digital multimedia applications. Then, we show that the feasibility of image and video compression rests in redundancy removal. Two types of redundancies: statistical redundancy and psychovisual redundancy are studied. Statistical redundancy comes from interpixel correlation and coding redundancy. By interpixel correlation, we mean correlation between pixels either located in one frame (spatial or intraframe redundancy) or pixels located in successive frames (temporal or interframe redundancy). Coding redundancy is related to coding technique. Psychovisual redundancy is based on the features (several types of masking phenomena) of human visual perception. That is, visual information is not perceived equally from the human visual point of view. In this sense, some information is psychovisually redundant.

The visual quality of the reconstructed image and video is a crucial criterion in the evaluation of the performance of visual transmission or storage systems. Both subjective and objective assessments are discussed. A new and promising objective technique based on subjective assessment is introduced. Since it combines the merits of both types of visual quality assessment, it achieves a quite satisfactory performance. The selected statistical features reveal some possible mechanism of the human visual perception. Further study in this regard would be fruitful.

In the last section, we introduced some fundamental information theory results, relevant to image and video compression. The results introduced include information measurement, entropy, and several theorems. All the theorems assume discrete, memoriless, and stationary information sources. The noiseless source coding theorem points out that the entropy of an information source is the lower bound of the coding bit rate that a source encoder can achieve. The source coding theorem deals with lossy coding applied in a noise-free channel. It states that for a given distortion, D, there is a rate distortion function, $R(D)$. When the bit rate in the source coding is greater than $R(D)$, the reconstructed source at the receiving end may satisfy the fidelity requirement defined by D. The noisy channel coding theorem states that, in order to achieve error-free performance, the source coding bit rate must be smaller than the channel capacity. Channel capacity is a function of noise and signal power. The information transmission theorem combines the noisy channel coding theorem and the source coding theorem. It states that it is possible to have a reconstructed waveform at the receiving end, satisfying the fidelity requirement corresponding to distortion D if the channel capacity, C, is larger than the rate distortion function $R(D)$. Though some of the assumptions on which these theorems were developed may not be valid in complicated practical situations, these theorems provide important theoretical limits for image and video coding. They can also be used for evaluation of the performance of different coding techniques.

1.6 EXERCISES

1-1. Using your own words, define spatial and temporal redundancy, and psychovisual redundancy, and state the impact they have on image and video compression.

1-2. Why is differential sensitivity considered the most important feature in human visual perception?

1-3. From the description of the newly developed objective assessment technique based on subjective assessment, discussed in Section 1.3, what points do you think are related to and support the statement made in Exercise 1-2?

1-4. Interpret Weber's law using your own words.

1-5. What is the advantage possessed by color models that decouple the luminance component from chrominance components.

1-6. Why has the HIS model not been adopted by any TV systems?

1-7. What is the problem with the objective visual quality measure of PSNR?

REFERENCES

Berger, T. *Rate Distortion Theory*, Englewood Cliffs, NJ, Prentice-Hall, 1971.

CCIR Recommendation 500-3, Method for the subjective assessment of the quality of television pictures, Recommendations and Reports of the CCIR, 1986, XVIth Plenary Assembly, Volume XI, Part 1.

Connor, D. J., R. C. Brainard, and J. O. Limb, Interframe coding for picture transmission, *Proc. IEEE*, 60(7), 779-790, 1972.

Fink, D. G. *Television Engineering Handbook*, New York, McGraw-Hill, 1957, Sect. 10.7.

Goodall, W. M. Television by pulse code modulation, *Bell Syst. Tech. J.*, 33-49, 1951.

Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Reading, MA, Addison-Wesley, 1992.

Haskell, B. G., A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, Chapman and Hall, New York, 1997.

Hidaka, T. and K. Ozawa, Subjective assessment of redundancy-reduced moving images for interactive application: test methodology and report, *Signal Process. Image Commun.*, 2, 201-219, 1990.

Huang, T. S. PCM picture transmission, *IEEE Spectrum*, 2(12), 57-63, 1965.

Huang, J. and Y. Q. Shi, Adaptive image watermarking scheme based on visual masking, *IEE Electron. Lett.*, 34(8), 748-750, 1998.

Kretzmer, E. R. Statistics of television signal, *Bell Syst. Tech. J.*, 31(4), 751-763, 1952.

Lathi, B. P. *Modern Digital and Analog Communication Systems*, 3rd ed., Oxford University Press, New York, 1998.

Legge, G. E. and J. M. Foley, Contrast masking in human vision, *J. Opt. Soc. Am.*, 70(12), 1458-1471, 1980.

Lim, J. S. *Two-Dimensional Signal and Image Processing*, Englewood Cliffs, NJ, Prentice Hall, 1990.

Mitchell, J. L., W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, Chapman and Hall, New York, 1997.

Mounts, F. W. A video encoding system with conditional picture-element replenishment, *Bell Syst. Tech. J.*, 48(7), 2545-2554, 1969.

Mullen, K.T. The contrast sensitivity of human color vision to red-green and blue-yellow chromatic gratings, *J. Physiol.*, 359, 381-400, 1985.

Netravali, A. N. and B. Prasada, Adaptive quantization of picture signals using spatial masking, *Proc. IEEE*, 65, 536-548, 1977.

Sakrison, D. J. Image coding applications of vision model, in *Image Transmission Techniques*, W. K. Pratt (Ed.), 21-71, New York, Academic Press, 1979.

Seyler, A. J. The coding of visual signals to reduce channel-capacity requirements, *IEE Monogr.*, 533E, July 1962.

Seyler, A. J. Probability distributions of television frame difference, *Proc. IREE (Australia)*, 26, 335, 1965.

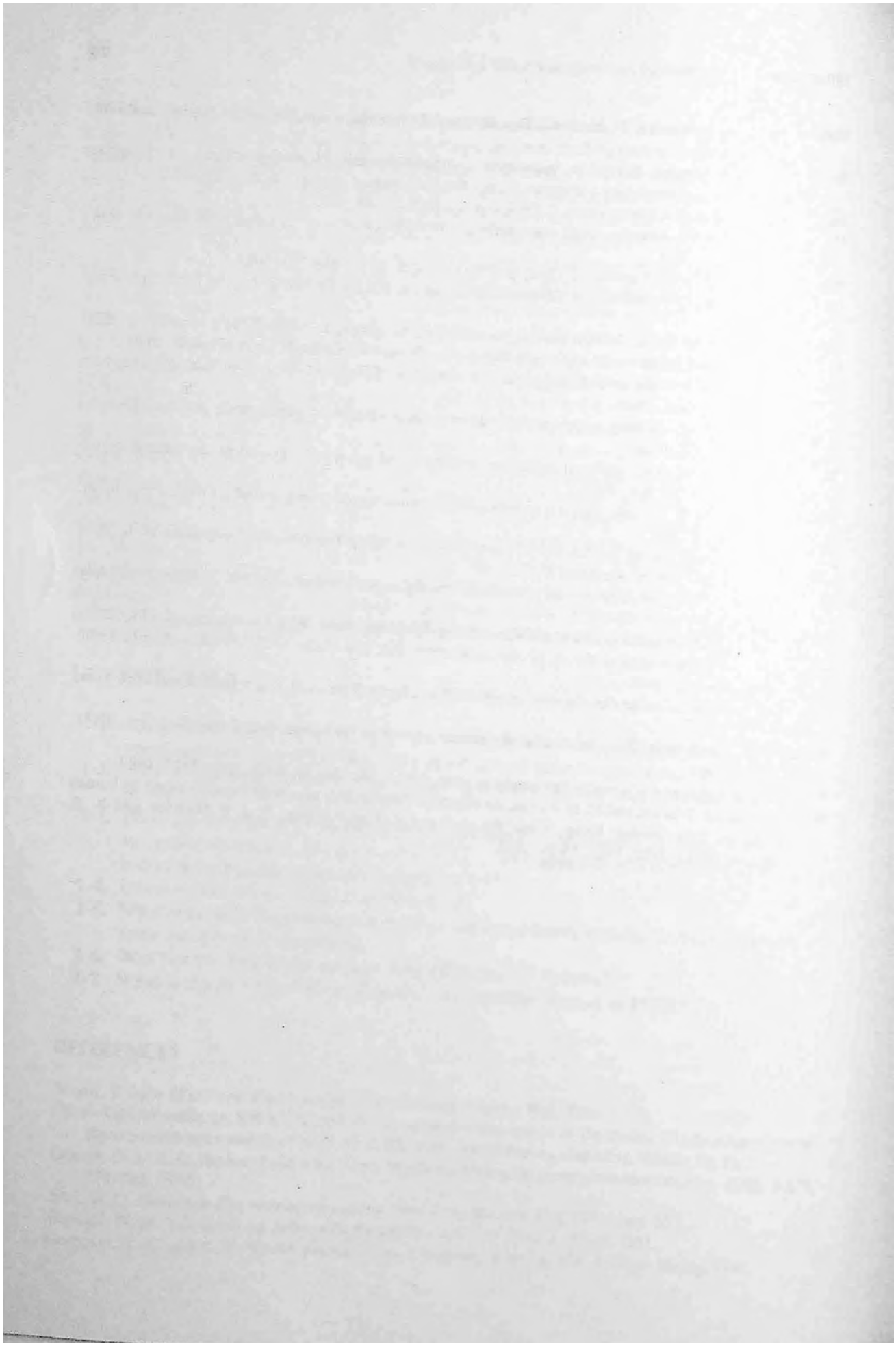
Shannon, C. E. A mathematical theory of communication, *Bell Syst. Tech. J.*, 27, 379-423 (Part I), 1948; 623-656 (Part II), 1948.

Slepian, D. *Key Papers in the Development of Information Theory*, Slepian, D. (Ed.), IEEE Press, New York, 1973.

Van Ness, F. I. and M. A. Bouman, Spatial modulation transfer in the human eye, *J. Opt. Soc. Am.*, 57(3), 401-406, 1967.

Watson, A. B. Efficiency of a model human image code, *J. Opt. Soc. Am., A*, 4(12), 2401-2417, 1987.

Webster, A. A., C. T. Jones, and M. H. Pinson, An objective video quality assessment system based on human perception, *Proc. Human Vision, Visual Processing and Digital Display IV*, J. P. Allebach and B. E. Rogowitz (Eds.), SPIE, 1913, 15-26, 1993.



2 Quantization

After the introduction to image and video compression presented in Chapter 1, we now address several fundamental aspects of image and video compression in the remaining chapters of Section I. Chapter 2, the first chapter in the series, concerns quantization. Quantization is a necessary component in lossy coding and has a direct impact on the bit rate and the distortion of reconstructed images or videos. We discuss concepts, principles and various quantization techniques which include uniform and nonuniform quantization, optimum quantization, and adaptive quantization.

2.1 QUANTIZATION AND THE SOURCE ENCODER

Recall Figure 1.1, in which the functionality of image and video compression in the applications of visual communications and storage is depicted. In the context of visual communications, the whole system may be illustrated as shown in Figure 2.1. In the transmitter, the input analog information source is converted to a digital format in the A/D converter block. The digital format is compressed through the image and video source encoder. In the channel encoder, some redundancy is added to help combat noise and, hence, transmission error. Modulation makes digital data suitable for transmission through the analog channel, such as air space in the application of a TV broadcast. At the receiver, the counterpart blocks reconstruct the input visual information. As far as storage of visual information is concerned, the blocks of channel, channel encoder, channel decoder, modulation, and demodulation may be omitted, as shown in Figure 2.2. If input and output are required to be in the digital format in some applications, then the A/D and D/A converters are omitted from the system. If they are required, however, other blocks such as encryption and decryption can be added to the system (Sklar, 1988). Hence, what is conceptualized in Figure 2.1 is a fundamental block diagram of a visual communication system.

In this book, we are mainly concerned with source encoding and source decoding. To this end, we take it a step further. That is, we show block diagrams of a source encoder and decoder in Figure 2.3. As shown in Figure 2.3(a), there are three components in source encoding: transformation, quantization, and codeword assignment. After the transformation, some form of an input information source is presented to a quantizer. In other words, the transformation block decides which types of quantities from the input image and video are to be encoded. It is not necessary that the original image and video waveform be quantized and coded: we will show that some formats obtained from the input image and video are more suitable for encoding. An example is the difference signal. From the discussion of interpixel correlation in Chapter 1, it is known that a pixel is normally highly correlated with its immediate horizontal or vertical neighboring pixel. Therefore, a better strategy is to encode the difference of gray level values between a pixel and its neighbor. Since these data are highly correlated, the difference usually has a smaller dynamic range. Consequently, the encoding is more efficient. This idea is discussed in Chapter 3 in detail.

Another example is what is called transform coding, which is addressed in Chapter 4. There, instead of encoding the original input image and video, we encode a transform of the input image and video. Since the redundancy in the transform domain is greatly reduced, the coding efficiency is much higher compared with directly encoding the original image and video.

Note that the term transformation in Figure 2.3(a) is sometimes referred to as *mapper* and *signal processing* in the literature (Gonzalez and Woods, 1992; Li and Zhang, 1995). Quantization refers to a process that converts input data into a set of finitely different values. Often, the input data to a quantizer are continuous in magnitude.

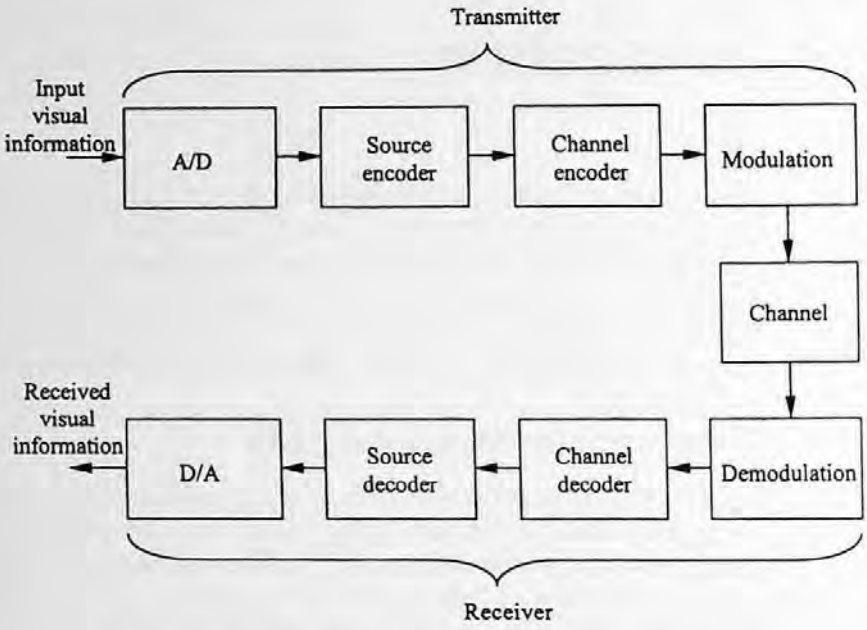


FIGURE 2.1 Block diagram of a visual communication system.

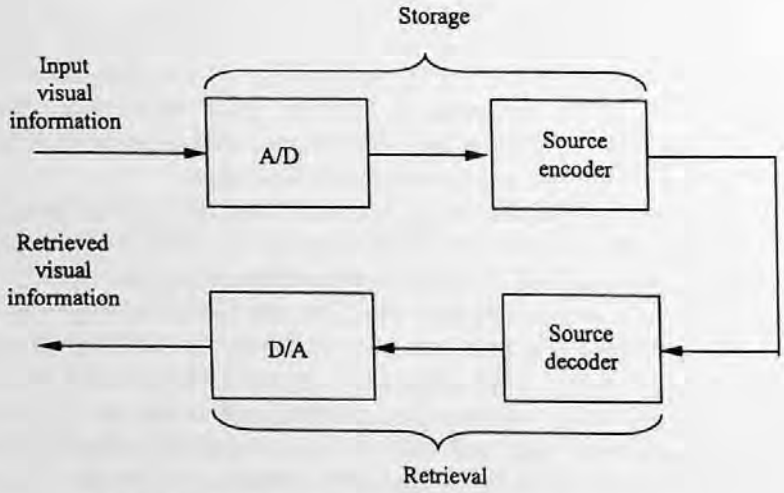


FIGURE 2.2 Block diagram of a visual storage system.

Hence, quantization is essentially discretization in magnitude, which is an important step in the lossy compression of digital image and video. (The reason that the term lossy compression is used here will be shown shortly.) The input and output of quantization can be either scalars or vectors. The quantization with scalar input and output is called *scalar quantization*, whereas that with vector input and output is referred to as *vector quantization*. In this chapter we discuss scalar quantization. Vector quantization will be addressed in Chapter 9.

After quantization, codewords are assigned to the many finitely different values from the output of the quantizer. Natural binary code (NBC) and variable-length code (VLC), introduced in Chapter 1, are two examples of this. Other examples are the widely utilized entropy code (including Huffman code and arithmetic code), dictionary code, and run-length code (RLC) (frequently used in facsimile transmission), which are covered in Chapters 5 and 6.

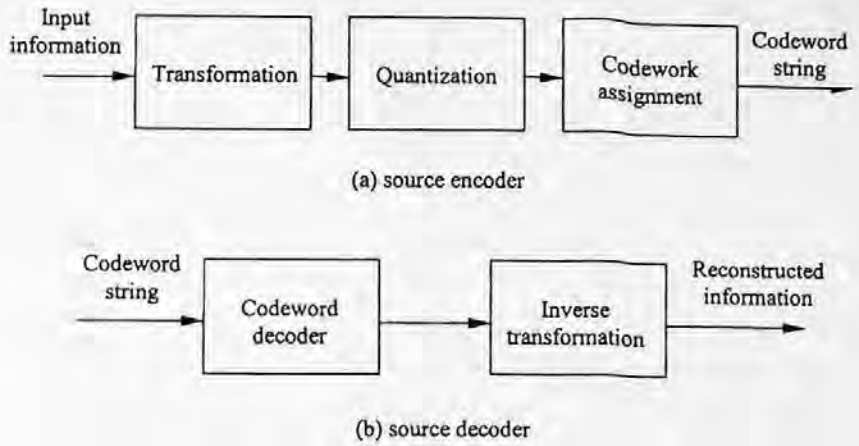


FIGURE 2.3 Block diagram of a source encoder and a source decoder.

The source decoder, as shown in Figure 2.3(b), consists of two blocks: codeword decoder and inverse transformation. They are counterparts of the codeword assignment and transformation in the source encoder. Note that there is no block that corresponds to quantization in the source decoder. The implication of this observation is the following. First, quantization is an irreversible process. That is, in general there is no way to find the original value from the quantized value. Second, quantization, therefore, is a source of information loss. In fact, quantization is a critical stage in image and video compression. It has significant impact on the distortion of reconstructed image and video as well as the bit rate of the encoder. Obviously, coarse quantization results in more distortion and a lower bit rate than fine quantization.

In this chapter, uniform quantization, which is the simplest yet the most important case, is discussed first. Nonuniform quantization is covered after that, followed by optimum quantization for both uniform and nonuniform cases. Then a discussion of adaptive quantization is provided. Finally, pulse code modulation (PCM), the best established and most frequently implemented digital coding method involving quantization, is described.

2.2 UNIFORM QUANTIZATION

Uniform quantization is the simplest and most popular quantization technique. Conceptually, it is of great importance. Hence, we start our discussion on quantization with uniform quantization. Several fundamental concepts of quantization are introduced in this section.

2.2.1 Basics

This subsection concerns several basic aspects of uniform quantization. These are some fundamental terms, quantization distortion, and quantizer design.

2.2.1.1 Definitions

Take a look at Figure 2.4. The horizontal axis denotes the input to a quantizer, while the vertical axis represents the output of the quantizer. The relationship between the input and the output best characterizes this quantizer; this type of configuration is referred to as the input-output characteristic of the quantizer. It can be seen that there are nine intervals along the x-axis. Whenever the input falls in one of the intervals, the output assumes a corresponding value. The input-output characteristic of the quantizer is staircase-like and, hence, clearly nonlinear.

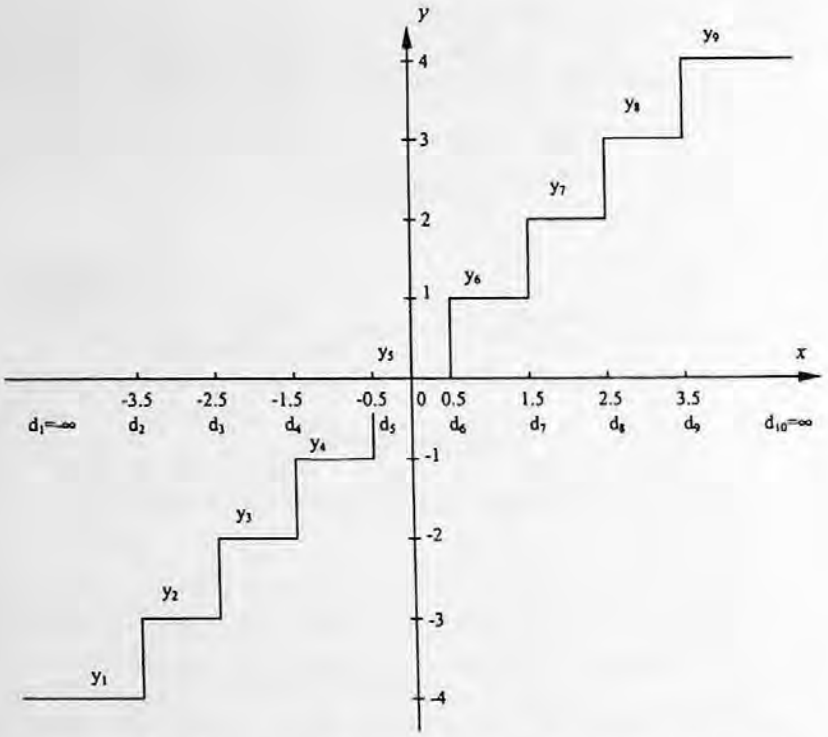


FIGURE 2.4 Input-output characteristic of a uniform midtread quantizer.

The end points of the intervals are called *decision levels*, denoted by d_i with i being the index of intervals. The output of the quantization is referred to as the *reconstruction level* (also known as *quantizing level* [Musmann, 1979]), denoted by y_i with i being its index. The length of the interval is called the *step size* of the quantizer, denoted by Δ. With the above terms defined, we can now mathematically define the function of the quantizer in Figure 2.4 as follows.

$$y_i = Q(x) \quad if \quad x \in \left(d_i, d_{i+1}\right) \tag{2.1}$$

where $i = 1,2,\cdots,9$ and $Q(x)$ is the output of the quantizer with respect to the input x.

It is noted that in Figure 2.4, $\Delta = 1$. The decision levels and reconstruction levels are evenly spaced. It is a uniform quantizer because it possesses the following two features.

1. Except for possibly the right-most and left-most intervals, all intervals (hence, decision levels) along the x-axis are uniformly spaced. That is, each inner interval has the same length.
2. Except for possibly the outer intervals, the reconstruction levels of the quantizer are also uniformly spaced. Furthermore, each inner reconstruction level is the arithmetic average of the two decision levels of the corresponding interval along the x-axis.

The uniform quantizer depicted in Figure 2.4 is called *midtread* quantizer. Its counterpart is called a *midrise* quantizer, in which the reconstructed levels do not include the value of zero. A midrise quantizer having step size $\Delta = 1$ is shown in Figure 2.5. Midtread quantizers are usually utilized for an odd number of reconstruction levels and midrise quantizers are used for an even number of reconstruction levels.

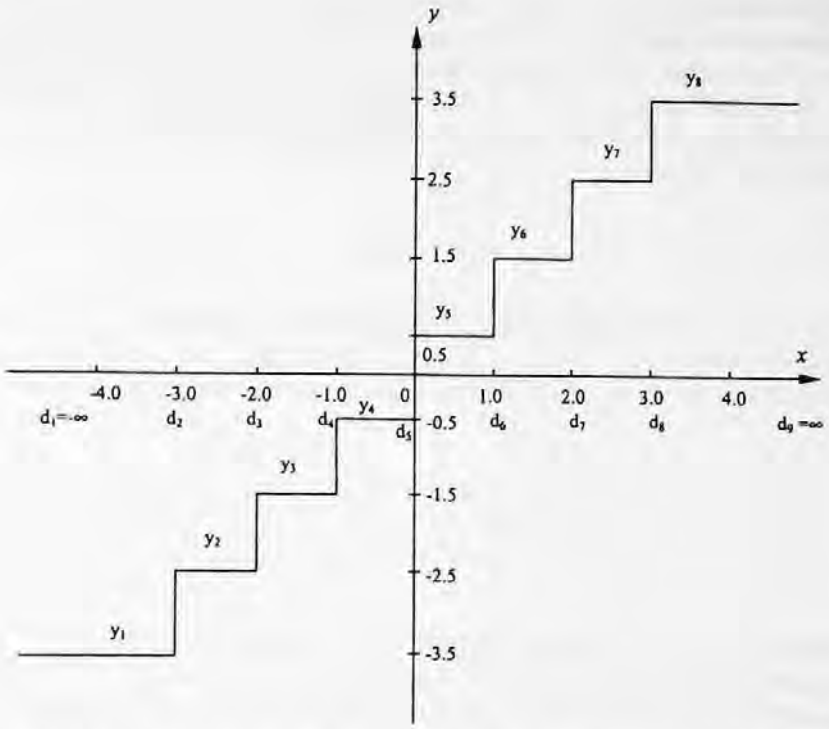


FIGURE 2.5 Input-output characteristic of a uniform midrise quantizer.

Note that the input-output characteristic of both the midtread and midrise uniform quantizers as depicted in Figures 2.4 and 2.5, respectively, is odd symmetric with respect to the vertical axis $x = 0$. In the rest of this chapter, our discussion develops under this symmetry assumption. The results thus derived will not lose generality since we can always subtract the statistical mean of input x from the input data and thus achieve this symmetry. After quantization, we can add the mean value back.

Denote by N the total number of reconstruction levels of a quantizer. A close look at Figure 2.4 and 2.5 reveals that if N is even, then the decision level $d_{(N/2)+1}$ is located in the middle of the input x-axis. If N is odd, on the other hand, then the reconstruction level $y_{(N+1)/2} = 0$. This convention is important in understanding the design tables of quantizers in the literature.

2.2.1.2 Quantization Distortion

The source coding theorem presented in Chapter 1 states that for a certain distortion D, there exists a rate distortion function $R(D)$, such that as long as the bit rate used is larger than $R(D)$ then it is possible to transmit the source with a distortion smaller than D. Since we cannot afford an infinite bit rate to represent an original source, some distortion in quantization is inevitable. In other words, we can say that since quantization causes information loss irreversibly, we encounter *quantization error* and, consequently, an issue: how do we evaluate the quality or, equivalently, the distortion of quantization. According to our discussion on visual quality assessment in Chapter 1, we know that there are two ways to do so: subjective evaluation and objective evaluation.

In terms of subjective evaluation, in Section 1.3.1 we introduced a five-scale rating adopted in CCIR Recommendation 500-3. We also described the false contouring phenomenon, which is caused by coarse quantization. That is, our human eyes are more sensitive to the relatively uniform regions in an image plane. Therefore an insufficient number of reconstruction levels results in

annoying false contours. In other words, more reconstruction levels are required in relatively uniform regions than in relatively nonuniform regions.

In terms of objective evaluation, in Section 1.3.2 we defined mean square error (*MSE*) and root mean square error (*RMSE*), signal-to-noise ratio (*SNR*), and peak signal-to-noise ratio (*PSNR*). In dealing with quantization, we define quantization error, e_q, as the difference between the input signal and the quantized output:

$$e_q = x - Q(x), \tag{2.2}$$

where x and $Q(x)$ are input and quantized output, respectively. Quantization error is often referred to as *quantization noise*. It is a common practice to treat input x as a random variable with a probability density function (*pdf*) $f_x(x)$. Mean square quantization error, MSE_q, can thus be expressed as

$$MSE_q = \sum_{i=1}^{N} \int_{d_i}^{d_{i+1}} \left(x - Q(x)\right)^2 f_x(x)dx \tag{2.3}$$

where N is the total number of reconstruction levels. Note that the outer decision levels may be $-\infty$ or ∞, as shown in Figures 2.4 and 2.5. It is clear that when the *pdf*, $f_x(x)$, remains unchanged, fewer reconstruction levels (smaller N) result in more distortion. That is, coarse quantization leads to large quantization noise. This confirms the statement that quantization is a critical component in a source encoder and significantly influences both bit rate and distortion of the encoder. As mentioned, the assumption we made above that the input-output characteristic is odd symmetric with respect to the $x = 0$ axis implies that the mean of the random variable, x, is equal to zero, i.e., $E(x) = 0$. Therefore the mean square quantization error MSE_q is the variance of the quantization noise equation, i.e., $MSE_q = \sigma_q^2$.

The quantization noise associated with the midtread quantizer depicted in Figure 2.4 is shown in Figure 2.6. It is clear that the quantization noise is signal dependent. It is observed that, associated with the inner intervals, the quantization noise is bounded by $\pm 0.5\Delta$. This type of quantization noise is referred to as *granular noise*. The noise associated with the right-most and the left-most

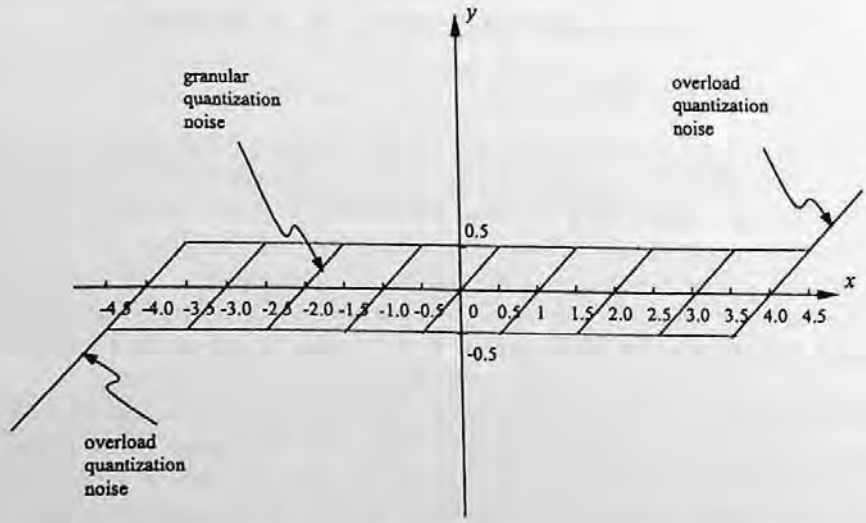


FIGURE 2.6 Quantization noise of the uniform midtread quantizer shown in Figure 2.4.

intervals are unbounded as the input x approaches either $-\infty$ or ∞. This type of quantization noise is called *overload noise*. Denoting the mean square granular noise and overload noise by $MSE_{q,g}$ and $MSE_{q,o}$, respectively, we then have the following relations:

$$MSE_q = MSE_{q,g} + MSE_{q,o} \tag{2.4}$$

and

$$MSE_{q,g} = \sum_{i=2}^{N-1} \int_{d_i}^{d_{i+1}} (x - Q(x))^2 \, f_X(x) dx \tag{2.5}$$

$$MSE_{q,o} = 2 \int_{d_1}^{d_2} (x - Q(x))^2 \, f_X(x) dx \tag{2.6}$$

2.2.1.3 Quantizer Design

The design of a quantizer (either uniform or nonuniform) involves choosing the number of reconstruction levels, N (hence, the number of decision levels, $N+1$), and selecting the values of decision levels and reconstruction levels (deciding where to locate them). In other words, the design of a quantizer is equivalent to specifying its input-output characteristic.

The *optimum* quantizer design can be stated as follows. For a given probability density function of the input random variable, $f_X(x)$, determine the number of reconstruction levels, N, choose a set of decision levels $\{d_i, i = 1, \cdots, N+1\}$ and a set of reconstruction levels $\{y_i, i = 1, \cdots, N\}$ such that the mean square quantization error, MSE_q, defined in Equation 2.3, is minimized.

In the uniform quantizer design, the total number of reconstruction levels, N, is usually given. According to the two features of uniform quanitzers described in Section 2.2.1.1, we know that the reconstruction levels of a uniform quantizer can be derived from the decision levels. Hence, only one of these two sets is independent. Furthermore, both decision levels and reconstruction levels are uniformly spaced except possibly the outer intervals. These constraints together with the symmetry assumption lead to the following observation: There is in fact only one parameter that needs to be decided in uniform quantizer design, which is the step size Δ. As to the optimum uniform quantizer design, a different *pdf* leads to a different step size.

2.2.2 Optimum Uniform Quantizer

In this subsection, we first discuss optimum uniform quantizer design when the input x obeys uniform distribution. Then, we cover optimum uniform quantizer design when the input x has other types of probabilistic distributions.

2.2.2.1 Uniform Quantizer with Uniformly Distributed Input

Let us return to Figure 2.4, where the input-output characteristic of a nine reconstruction-level midtread quantizer is shown. Now, consider that the input x is a uniformly distributed random variable. Its input-output characteristic is shown in Figure 2.7. We notice that the new characteristic is restricted within a finite range of x, i.e., $-4.5 \le x \le 4.5$. This is due to the definition of uniform distribution. Consequently, the overload quantization noise does not exist in this case, which is shown in Figure 2.8.

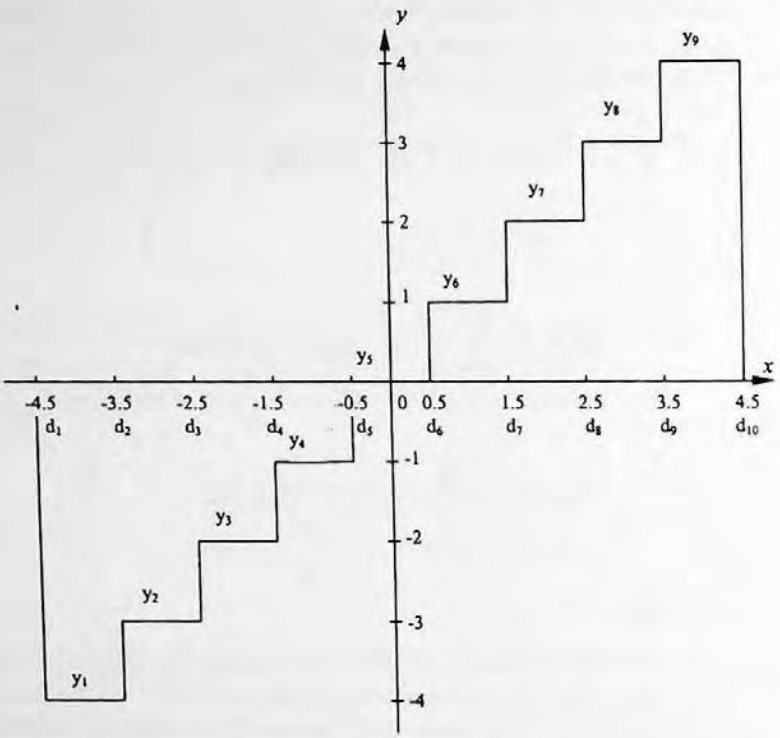


FIGURE 2.7 Input-output characteristic of a uniform midtread quantizer with input x having uniform distribution in [-4.5, 4.5].

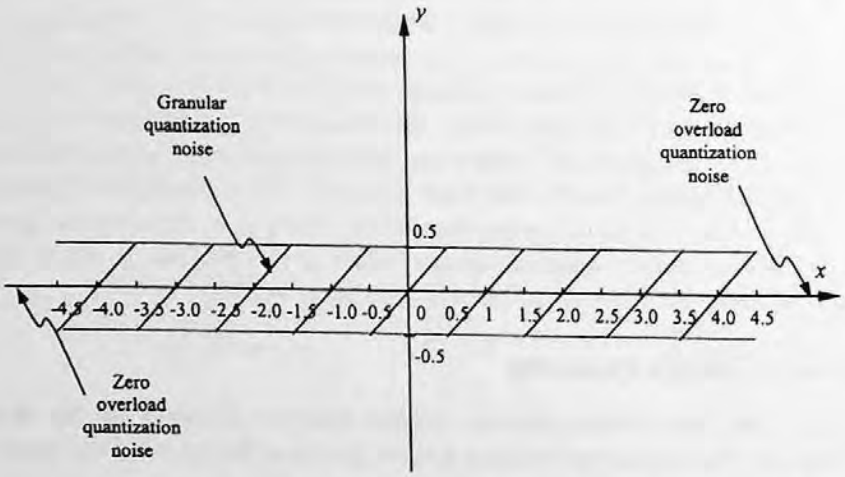


FIGURE 2.8 Quantization noise of the quantizer shown in Figure 2.7.

The mean square quantization error is found to be

$$MSE_q = N\int_{d_1}^{d_2} \left(x - Q(x)\right)^2 \frac{1}{N\Delta} dx \tag{2.7}$$

$$MSE_q = \frac{\Delta^2}{12}$$

This result indicates that if the input to a uniform quantizer has a uniform distribution and the number of reconstruction levels is fixed, then the mean square quantization error is directly proportional to the square of the quantization step size. Or, in other words, the root mean square quantization error (the standard deviation of the quantization noise) is directly proportional to the quantization step. The larger the step size, the larger (according to square law) the mean square quantization error. This agrees with our previous observation: coarse quantization leads to large quantization error.

As mentioned above, the mean square quantization error is equal to the variance of the quantization noise, i.e., $MSE_q = \sigma_q^2$. In order to find the signal-to-noise ratio of the uniform quantization in this case, we need to determine the variance of the input x. Note that we assume the input x to be a zero mean uniform random variable. So, according to probability theory, we have

$$\sigma_x^2 = \frac{(N\Delta)^2}{12} \tag{2.8}$$

Therefore, the mean square signal-to-noise ratio, SNR_{ms}, defined in Chapter 1, is equal to

$$SNR_{ms} = 10\log_{10}\frac{\sigma_x^2}{\sigma_q^2} = 10\log_{10} N^2. \tag{2.9}$$

Note that here we use the subscript *ms* to indicate the signal-to-noise ratio in the mean square sense, as defined in the previous chapter. If we assume $N = 2^n$, we then have

$$SNR_{ms} = 20\log_{10} 2^n = 6.02n \quad dB. \tag{2.10}$$

The interpretation of the above result is as follows. If we use the natural binary code to code the reconstruction levels of a uniform quantizer with a uniformly distributed input source, then every increased bit in the coding brings out a 6.02-dB increase in the SNR_{ms}. An equivalent statement can be derived from Equation 2.7. That is, whenever the step size of the uniform quantizer decreases by a half, the mean square quantization error decreases four times.

2.2.2.2 Conditions of Optimum Quantization

The conditions under which the mean square quantization error MSE_q is minimized were derived (Lloyd, 1982; Max, 1960) for a given probability density function of the quantizer input, $f_X(x)$.

The mean square quantization error MSE_q was given in Equation 2.3. The necessary conditions for optimum (minimum mean square error) quantization are as follows. That is, the derivatives of MSE_q with respect to the d_i and y_i have to be zero.

$$\left(d_i - y_{i-1}\right)^2 f_x(d_i) - \left(d_i - y_i\right)^2 f_x(d_i) = 0 \qquad i = 2, \cdots, N \tag{2.11}$$

$$-\int_{d_i}^{d_{i+1}} \left(x - y_i\right) f_x(x)\,dx = 0 \qquad i = 1, \cdots, N \tag{2.12}$$

The sufficient conditions can be derived accordingly by involving the second-order derivatives (Max, 1960; Fleischer, 1964). The symmetry assumption of the input-output characteristic made earlier holds here as well. These sufficient conditions are listed below.

$$1.\quad x_1 = -\infty \text{ and } x_{N+1} = +\infty \tag{2.13}$$

$$2.\quad \int_{d_i}^{d_{i+1}} (x - y_i) f_x(x) dx = 0 \qquad i = 1, 2, \cdots, N \tag{2.14}$$

$$3.\quad d_i = \frac{1}{2}(y_{i-1} + y_i) \qquad i = 2, \cdots, N \tag{2.15}$$

Note that the first condition is for an input x whose range is $-\infty < x < \infty$. The interpretation of the above conditions is that each decision level (except for the outer intervals) is the arithmetic average of the two neighboring reconstruction levels, and each reconstruction level is the centroid of the area under the probability density function $f_X(x)$ and between the two adjacent decision levels.

Note that the above conditions are general in the sense that there is no restriction imposed on the *pdf*. In the next subsubsection, we discuss the optimum uniform quantization when the input of quantizer assumes different distributions.

2.2.2.3 Optimum Uniform Quantizer with Different Input Distributions

Let's return to our discussion on the optimum quantizer design whose input has uniform distribution. Since the input has uniform distribution, the outer intervals are also finite. For uniform distribution, Equation 2.14 implies that each reconstruction level is the arithmetic average of the two corresponding decision levels. Considering the two features of a uniform quantizer, presented in Section 2.2.1.1, we see that a uniform quantizer is optimum (minimizing the mean square quantization error) when the input has uniform distribution.

When the input x is uniformly distributed in [-1,1], the step size Δ of the optimum uniform quantizer is listed in Table 2.1 for the number of reconstruction levels, N, equal to 2, 4, 8, 16, and 32. From the table, we notice that the MSE_q of the uniform quantization with a uniformly distributed input decreases four times as N doubles. As mentioned in Section 2.2.2.1, this is equivalent to an increase of SNR_{ms} by 6.02 dB as N doubles.

The derivation above is a special case, i.e., the uniform quantizer is optimum for a uniformly distributed input. Normally, if the probability density function is not uniform, the optimum quantizer is not a uniform quantizer. Due to the simplicity of uniform quantization, however, it may sometimes be desirable to design an optimum uniform quantizer for an input with an other-than-uniform distribution.

Under these circumstances, however, Equations 2.13, 2.14, and 2.15 are not a set of simultaneous equations one can hope to solve with any ease. Numerical procedures were suggested to solve for design of optimum uniform quantizers. Max derived uniform quantization step size Δ for an input with a Gaussian distribution (Max, 1960). Paez and Glisson (1972) found step size Δ for Laplacian- and Gamma-distributed input signals. These results are listed in Table 2.1. Note that all three distributions have a zero mean and unit standard deviation. If the mean is not zero, only a shift in input is needed when applying these results. If the standard deviation is not unity, the tabulated step size needs to be multiplied by the standard deviation. The theoretical *MSE* is also listed in Table 2.1. Note that the subscript q associated with *MSE* has been dropped from now on in the chapter for the sake of notational brevity as long as it does not cause confusion.

2.3 NONUNIFORM QUANTIZATION

It is not difficult to see that, except for the special case of the uniformly distributed input variable x, the optimum (minimum *MSE*, also denoted sometimes by *MMSE*) quantizers should be nonuniform.

TABLE 2.1
Optimal Symmetric Uniform Quantizer for Uniform Gaussian, Laplacian, and Gamma Distributions [a]

	Uniform			Gaussian			Laplacian			Gamma		
N	d_i	y_i	MSE	d_i	y_i	MSE	d_i	y_i	MSE	d_i	y_i	MSE
2	−1.000	−0.500	8.33	−1.596	−0.798	0.363	−1.414	−0.707	0.500	−1.154	−0.577	0.668
	0.000	0.500	$\times 10^{-2}$	0.000	0.798		0.000	0.707		0.000	0.577	
	1.000			**1.596**			**1.414**			**1.154**		
4	−1.000	−0.750	2.08	−1.991	−1.494	0.119	−2.174	−1.631	1.963	−2.120	−1.590	0.320
	−0.500	−0.250	$\times 10^{-2}$	−0.996	−0.498		−1.087	−0.544	$\times 10^{-1}$	−1.060	−0.530	
	0.000	0.250		0.000	0.498		0.000	0.544		0.000	0.530	
	0.500	0.750		**0.996**	1.494		**1.087**	1.631		**1.060**	1.590	
	1.000			1.991			2.174			2.120		
8	−1.000	−0.875	5.21	−2.344	−2.051	3.74	−2.924	−2.559	7.17	−3.184	−2.786	0.132
	−0.750	−0.625	$\times 10^{-3}$	−1.758	−1.465	$\times 10^{-2}$	−2.193	−1.828	$\times 10^{-2}$	−2.388	−1.990	
	−0.500	−0.375		−1.172	−0.879		−1.462	−1.097		−1.592	−1.194	
	−0.250	−0.125		−0.586	−0.293		−0.731	−0.366		−0.796	−0.398	
	0.000	0.125		0.000	0.293		0.000	0.366		0.000	0.398	
	0.250	0.375		**0.586**	0.879		**0.731**	1.097		**0.796**	1.194	
	0.500	0.625		1.172	1.465		1.462	1.828		1.592	1.990	
	0.750	0.875		1.758	2.051		2.193	2.559		2.388	2.786	
	1.000			2.344			2.924			3.184		
16	−1.000	−0.938	1.30	−2.680	−2.513	1.15	−3.648	−3.420	2.54	−4.320	−4.050	5.01
	−0.875	−0.813	$\times 10^{-3}$	−2.345	−2.178	$\times 10^{-2}$	−3.192	−2.964	$\times 10^{-2}$	−3.780	−3.510	$\times 10^{-2}$
	−0.750	−0.688		−2.010	−1.843		−2.736	−2.508		−3.240	−2.970	
	−0.625	−0.563		−1.675	−1.508		−2.280	−2.052		−2.700	−2.430	
	−0.500	−0.438		−1.340	−1.173		−1.824	−1.596		−2.160	−1.890	
	−0.375	−0.313		−1.005	−0.838		−1.368	−1.140		−1.620	−1.350	
	−0.250	−0.188		−0.670	−0.503		−0.912	−0.684		−1.080	−0.810	
	−0.125	−0.063		−0.335	−0.168		−0.456	−0.228		−0.540	−0.270	
	0.000	0.063		0.000	0.168		0.000	0.228		0.000	0.270	
	0.125	0.188		**0.335**	0.503		**0.456**	0.684		**0.540**	0.810	
	0.250	0.313		0.670	0.838		0.912	1.140		1.080	1.350	
	0.375	0.438		1.005	1.173		1.368	1.596		1.620	1.890	
	0.500	0.563		1.340	1.508		1.824	2.052		2.160	2.430	
	0.625	0.688		1.675	1.843		2.280	2.508		2.700	2.970	
	0.750	0.813		2.010	2.178		2.736	2.964		3.240	3.510	
	0.875	0.938		2.345	2.513		3.192	3.420		3.780	4.050	
	1.000			2.680			3.648			4.320		

Note: The uniform distribution is between [−1,1], the other three distributions have zero mean and unit variance. The numbers in bold type are the step sizes.

[a] Data from (Max, 1960; Paez and Glisson, 1972).

Consider a case in which the input random variable obeys the Gaussian distribution with a zero mean and unit variance, and the number of reconstruction levels is finite. We naturally consider that having decision levels more densely located around the middle of the x-axis, $x = 0$ (high-probability density region), and choosing decision levels more coarsely distributed in the range far away from the center of the x-axis (low-probability density region) will lead to less *MSE*. The strategy adopted here is analogous to the superiority of variable-length code over fixed-length code discussed in the previous chapter.

2.3.1 Optimum (Nonuniform) Quantization

Conditions for optimum quantization were discussed in Section 2.2.2.2. With some constraints, these conditions were solved in a closed form (Panter and Dite, 1951). The equations characterizing these conditions, however, cannot be solved in a closed form in general. Lloyd and Max proposed an iterative procedure to numerically solve the equations. The optimum quantizers thus designed are called Lloyd-Max quantizers.

TABLE 2.2
Optimal Symmetric Quantizer for Uniform, Gaussian, Laplacian, and Gamma Distributions[a]

	Uniform			Gaussian			Laplacian			Gamma		
N	d_i	y_i	MSE	d_i	y_i	MSE	d_i	y_i	MSE	d_i	y_i	MSE
2	−1.000	−0.500	8.33	−∞	−0.799	0.363	−∞	−0.707	0.500	−∞	−0.577	0.668
	0.000	0.500	$\times 10^{-2}$	0.000	0.799		0.000	0.707		0.000	0.577	
	1.000			∞			∞			∞		
4	−1.000	−0.750	2.08	−∞	−1.510	0.118	−∞	−1.834	1.765	−∞	−2.108	0.233
	−0.500	−0.250	$\times 10^{-2}$	−0.982	−0.453		−1.127	−0.420	$\times 10^{-1}$	−1.205	−0.302	
	0.000	0.250		0.000	0.453		0.000	0.420		0.000	0.302	
	0.500	0.750		−0.982	1.510		1.127	1.834		1.205	2.108	
	1.000			∞			∞			∞		
8	−1.000	−0.875	5.21	−∞	−2.152	3.45	−∞	−3.087	5.48	−∞	−3.799	7.12
	−0.750	−0.625	$\times 10^{-3}$	−1.748	−1.344	$\times 10^{-2}$	−2.377	−1.673	$\times 10^{-2}$	−2.872	−1.944	$\times 10^{-2}$
	−0.500	−0.375		−1.050	−0.756		−1.253	−0.833		−1.401	−0.859	
	−0.250	−0.125		−0.501	−0.245		−0.533	−0.233		−0.504	−0.149	
	0.000	0.125		0.000	0.245		0.000	0.233		0.000	0.149	
	0.250	0.375		0.501	0.756		0.533	0.833		0.504	0.859	
	0.500	0.625		1.050	1.344		1.253	1.673		1.401	1.944	
	0.750	0.875		1.748	2.152		2.377	3.087		2.872	3.799	
	1.000			∞			∞			∞		
16	−1.000	−0.938	1.30	−∞	−2.733	9.50	−∞	−4.316	1.54	−∞	−6.085	1.96
	−0.875	−0.813	$\times 10^{-3}$	−2.401	−2.069	$\times 10^{-3}$	−3.605	−2.895	$\times 10^{-2}$	−5.050	−4.015	$\times 10^{-2}$
	−0.750	−0.688		−1.844	−1.618		−2.499	−2.103		−3.407	−2.798	
	−0.625	−0.563		−1.437	−1.256		−1.821	−1.540		−2.372	−1.945	
	−0.500	−0.438		−1.099	−0.942		−1.317	−1.095		−1.623	−1.300	
	−0.375	−0.313		−0.800	−0.657		−0.910	−0.726		−1.045	−0.791	
	−0.250	−0.188		−0.522	−0.388		−0.566	−0.407		−0.588	−0.386	
	−0.125	−0.063		−0.258	−0.128		−0.266	−0.126		−0.229	−0.072	
	0.000	0.063		0.000	0.128		0.000	0.126		0.000	0.072	
	0.125	0.188		0.258	0.388		0.266	0.407		0.229	0.386	
	0.250	0.313		0.522	0.657		0.566	0.726		0.588	0.791	
	0.375	0.438		0.800	0.942		0.910	1.095		1.045	1.300	
	0.500	0.563		1.099	1.256		1.317	1.540		1.623	1.945	
	0.625	0.688		1.437	1.618		1.821	2.103		2.372	2.798	
	0.750	0.813		1.844	2.069		2.499	2.895		3.407	4.015	
	0.875	0.938		2.401	2.733		3.605	4.316		5.050	6.085	
	1.000			∞			∞			∞		

Note: The uniform distribution is between [-1, 1], the other three distributions have zero mean and unit variance.

[a] Data from Lloyd, 1957, 1982; Max, 1990; and Paez, 1972.

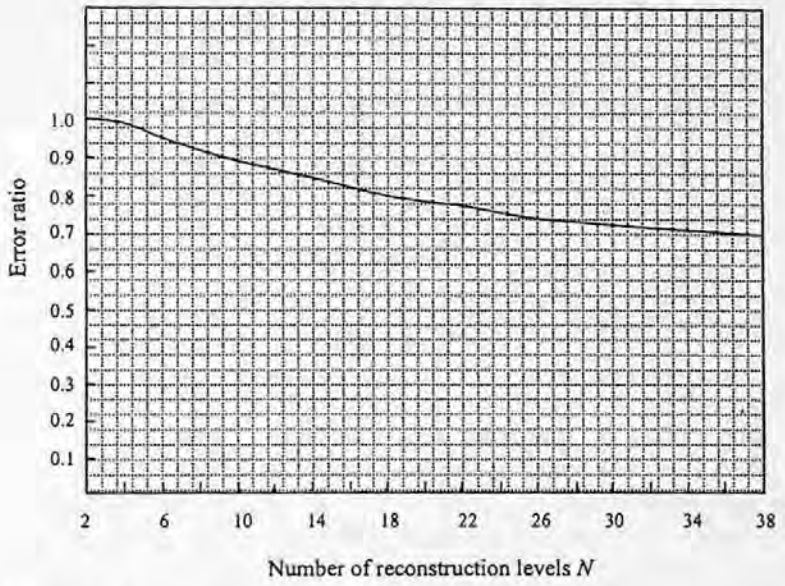


FIGURE 2.9 Ratio of the error for the optimal quantizer to the error for the optimum uniform quantizer vs. the number of reconstruction levels N. (Minimum mean square error for Gaussian-distributed input with a zero mean and unit variance). (Data from Max, 1960.)

The solution to optimum quantizer design for many finite reconstruction levels N when input x obeys Gaussian distribution was obtained (Lloyd, 1982; Max, 1960). That is, the decision levels and reconstruction levels together with theoretical minimum *MSE* and optimum *SNR* have been determined. Following this procedure, the design for Laplacian and Gamma distribution were tabulated in (Paez and Glisson, 1972). These results are contained in Table 2.2. As stated before, we see once again that uniform quantization is optimal if the input x is a uniform random variable.

Figure 2.9 (Max, 1960) gives a performance comparison between optimum uniform quantization and optimum quantization for the case of a Gaussian-distributed input with a zero mean and unit variance. The abscissa represents the number of reconstruction levels, N, and the ordinate the ratio between the error of the optimum quantizer and the error of the optimum uniform quantizer. It can be seen that when N is small, the ratio is close to one. That is, the performances are close. When N increases, the ratio decreases. Specifically, when N is large the nonuniform quantizer is about 20 to 30% more efficient than the uniform optimum quantizer for the Gaussian distribution with a zero mean and unit variance.

2.3.2 Companding Quantization

It is known that a speech signal usually has a large dynamic range. Moreover, its statistical distribution reveals that very low speech volumes predominate most voice communications. Specifically, by a 50% chance, the voltage characterizing detected speech energy is less than 25% of the root mean square (rms) value of the signal. Large amplitude values are rare: only by a 15% chance does the voltage exceed the rms value (Sklar, 1988). These statistics naturally lead to the need for nonuniform quantization with relatively dense decision levels in the small-magnitude range and relatively coarse decision levels in the large-magnitude range.

When the bit rate is eight bits per sample, the following companding technique (Smith, 1957), which realizes nonuniform quantization, is found to be extremely useful. Though speech coding is not the main focus of this book, we briefly discuss the companding technique here as an alternative way to achieve nonuniform quantization.

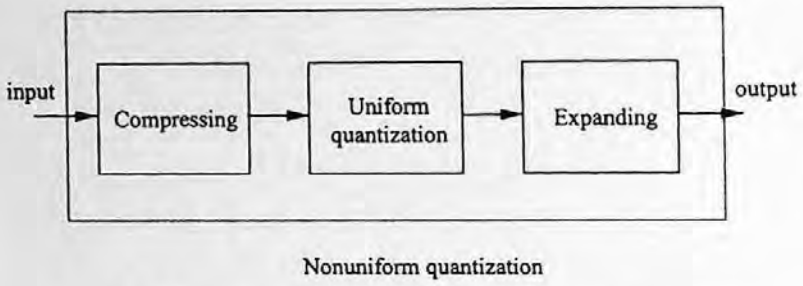


FIGURE 2.10 Companding technique in achieving quantization.

The companding technique, also known as logarithmic quantization, consists of the following three stages: compressing, uniform quantization, and expanding (Gersho, 1977), as shown in Figure 2.10. It first compresses the input signal with a logarithmic characteristic, and then it quantizes the compressed input using a uniform quantizer. Finally, the uniformly quantized results are expanded inversely. An illustration of the characteristics of these three stages and the resultant nonuniform quantization are shown in Figure 2.11.

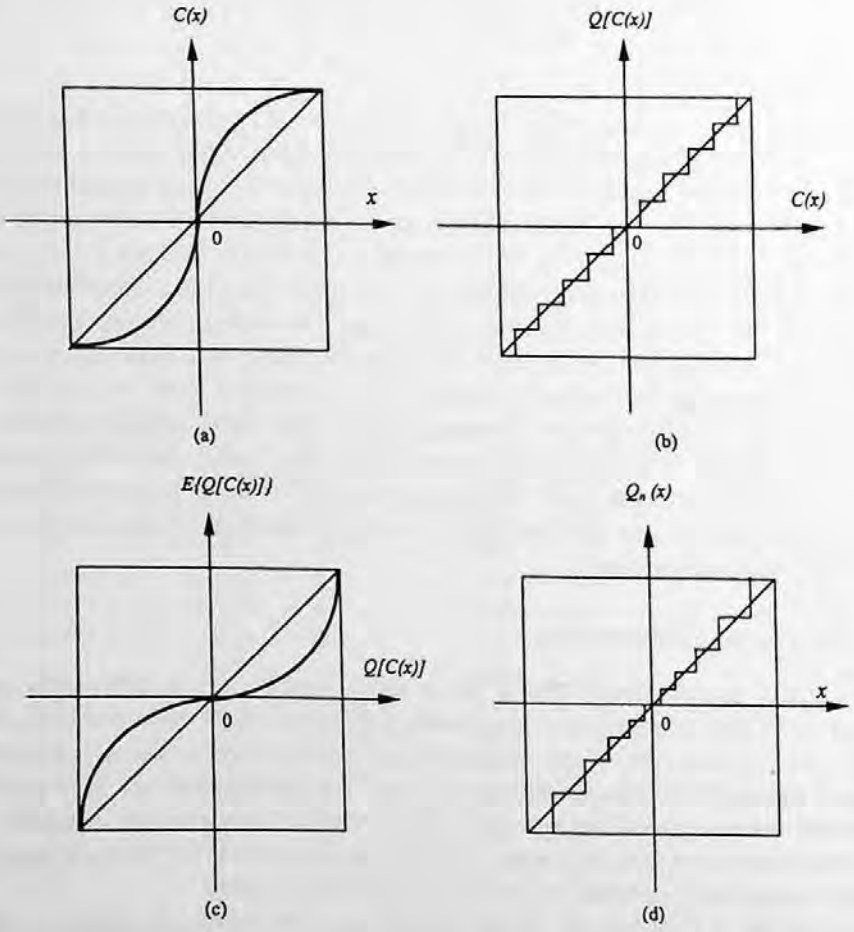
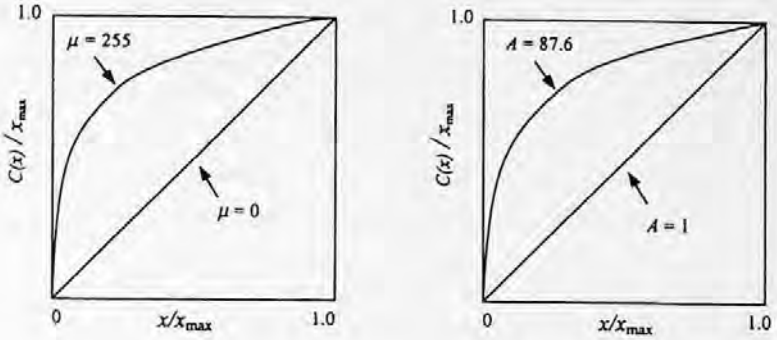


FIGURE 2.11 Characteristics of companding techniques. (a) Compressing characteristic. (b) Uniform quantizer characteristic. (c) Expanding characteristic. (d) Nonuniform quantizer characteristic.



(a) μ law (b) A law

FIGURE 2.12 Compression characteristics.

In practice, a piecewise linear approximation of the logarithmic compression characteristic is used. There are two different ways. In North America, a μ-law compression characteristic is used, which is defined as follows:

$$c(x) = x_{\max} \frac{\ln\left[1+\mu\left(|x|/x_{\max}\right)\right]}{\ln(1+\mu)} \operatorname{sgn} x, \tag{2.16}$$

where sgn is a sign function defined as

$$\operatorname{sgn} x = \begin{cases} +1 & \text{if} \quad x \geq 0 \\ -1 & \text{if} \quad x < 0 \end{cases} \tag{2.17}$$

The μ-law compression characteristic is shown in Figure 2.12(a). The standard value of μ is 255. Note from the figure that the case of μ = 0 corresponds to uniform quantization.

In Europe, the A-law characteristic is used. The A-law characteristic is depicted in Figure 2.12(b), and is defined as follows:

$$c(x) = \begin{cases} x_{\max} \dfrac{A\left(|x|/x_{\max}\right)}{1+\ln A} \operatorname{sgn} x & 0 < \dfrac{|x|}{x_{\max}} \leq \dfrac{1}{A} \\ x_{\max} \dfrac{1+\ln\left[A\left(|x|/x_{\max}\right)\right]}{1+\ln A} \operatorname{sgn} x & \dfrac{1}{A} < \dfrac{|x|}{x_{\max}} < 1 \end{cases} \tag{2.18}$$

It is noted that the standard value of A is 87.6. The case of A = 1 corresponds to uniform quantization.

2.4 ADAPTIVE QUANTIZATION

In the previous section, we studied nonuniform quantization, whose motivation is to minimize the mean square quantization error MSE_q. We found that nonuniform quantization is necessary if the *pdf* of the input random variable x is not uniform. Consider an optimum quantizer for a Gaussian-distributed input when the number of reconstruction levels N is eight. Its input-output characteristic can be derived from Table 2.2 and is shown in Figure 2.13. This plot reveals that the decision levels are densely located in the central region of the x-axis and coarsely elsewhere. In other words, the

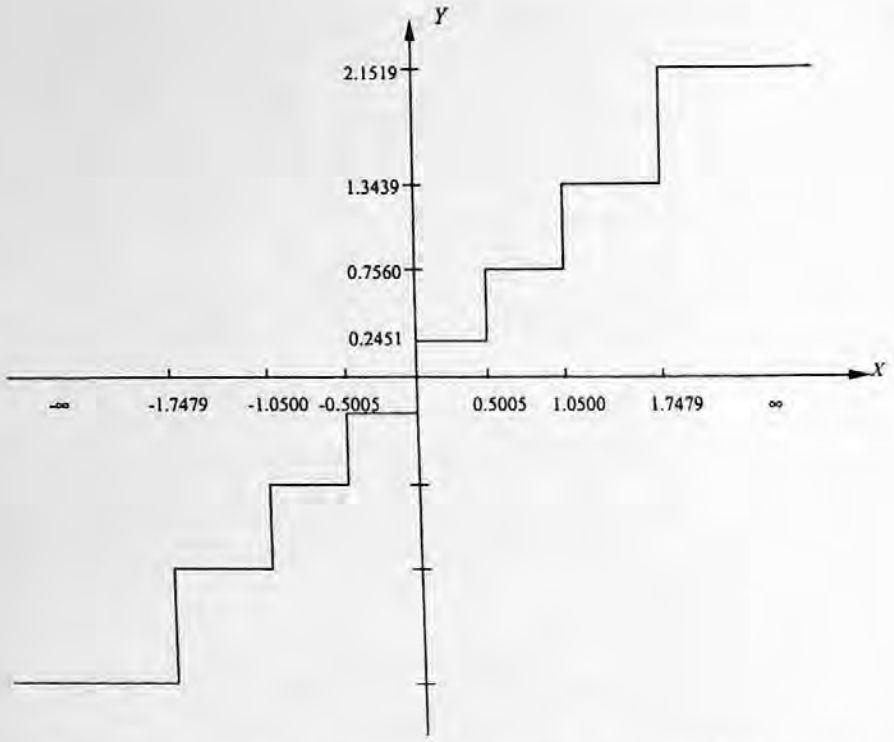


FIGURE 2.13 Input-output characteristic of the optimal quantizer for Gaussian distribution with zero mean, unit variance, and N = 8.

decision levels are densely distributed in the region having a higher probability of occurrence and coarsely distributed in other regions. A logarithmic companding technique also allocates decision levels densely in the small-magnitude region, which corresponds to a high occurrence probability, but in a different way. We conclude that nonuniform quantization achieves minimum mean square quantization error by distributing decision levels according to the statistics of the input random variable.

These two types of nonuniform quantizers are both time-invariant. That is, they are not designed for nonstationary input signals. Moreover, even for a stationary input signal, if its *pdf* deviates from that with which the optimum quantizer is designed, then what is called *mismatch* will take place and the performance of the quantizer will deteriorate. There are two main types of mismatch. One is called variance mismatch. That is, the *pdf* of input signal is matched, while the variance is mismatched. Another type is *pdf* mismatch. Noted that these two kinds of mismatch also occur in optimum uniform quantization, since there the optimization is also achieved based on the input statistics assumption. For a detailed analysis of the effects of the two types of mismatch on quantization, readers are referred to (Jayant and Noll, 1984).

Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance. It is a means to combat the mismatch problem discussed above. By statistics, we mean the statistical mean, variance (or the dynamic range), and type of input *pdf*. When the mean of the input changes, differential coding (discussed in the next chapter) is a suitable method to handle the variation. For other types of cases, adaptive quantization is found to be effective. The price paid for adaptive quantization is processing delays and an extra storage requirement as seen below.

There are two different types of adaptive quantization: forward adaptation and backward adaptation. Before we discuss these, however, let us describe an alternative way to define quantization (Jayant and Noll, 1984). Look at Figure 2.14. Quantization can be viewed as a two-stage

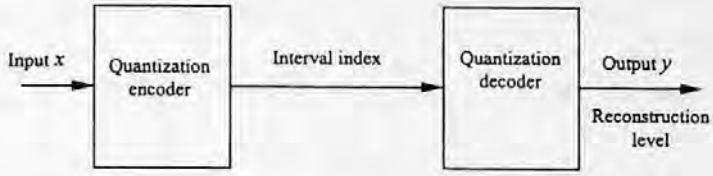


FIGURE 2.14 A two-stage model of quantization.

process. The first stage is the quantization encoder and the second stage is the quantization decoder. In the encoder, the input to quantization is converted to the index of an interval into which the input x falls. This index is mapped to (the codeword that represents) the reconstruction level corresponding to the interval in the decoder. Roughly speaking, this definition considers a quantizer as a communication system in which the quantization encoder is on the transmitter side while the quantization decoder is on the receiver side. In this sense, this definition is broader than that for quantization defined in Figure 2.3(a).

2.4.1 Forward Adaptive Quantization

A block diagram of forward adaptive quantization is shown in Figure 2.15. There, the input to the quantizer, x, is first split into blocks, each with a certain length. Blocks are stored in a buffer one at a time. A statistical analysis is then carried out with respect to the block in the buffer. Based on the analysis, the quantization encoder is set up, and the input data within the block are assigned indexes of respective intervals. In addition to these indexes, the encoder setting parameters, derived from the statistical analysis, are sent to the quantization decoder as *side* information. The term *side* comes from the fact that the amount of bits used for coding the setting parameter is usually a small fraction of the total amount of bits used.

Selection of the block size is a critical issue. If the size is small, the adaptation to the local statistics will be effective, but the side information needs to be sent frequently. That is, more bits are used for sending the side information. If the size is large, the bits used for side information decrease. On the other hand, the adaptation becomes less sensitive to changing statistics, and both processing delays and storage required increase. In practice, a proper compromise between the quantity of side information and the effectiveness of adaptation produces a good selection of the block size.

Examples of using the forward approach to adapt quantization to a changing input variance (to combat variance mismatch) can be found in (Jayant and Noll, 1984; Sayood, 1996).

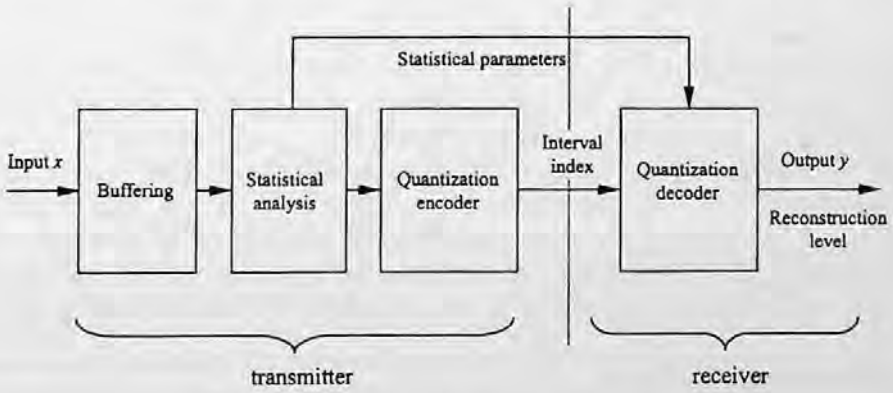


FIGURE 2.15 Forward adaptive quantization.

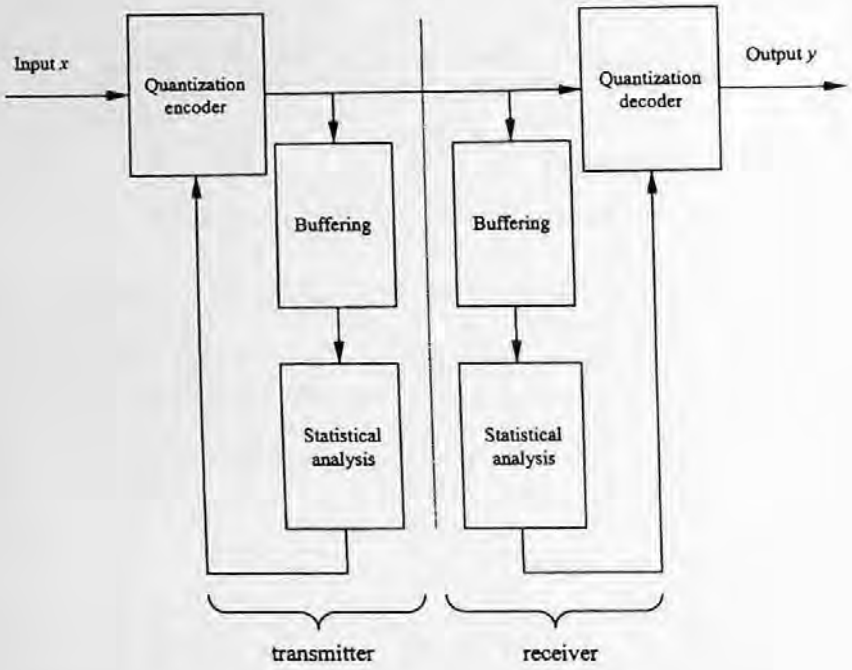


FIGURE 2.16 Backward adaptive quantization.

2.4.2 Backward Adaptive Quantization

Figure 2.16 shows a block diagram of backward adaptive quantization. A close look at the block diagram reveals that in both the quantization encoder and decoder the buffering and the statistical analysis are carried out with respect to the output of the quantization encoder. In this way, there is no need to send side information. The sensitivity of adaptation to the changing statistics will be degraded, however, since instead of the original input, only the output of the quantization encoder is used in the statistical analysis. That is, the quantization noise is involved in the statistical analysis.

2.4.3 Adaptive Quantization with a One-Word Memory

Intuitively, it is expected that observance of a sufficiently large number of input or output (quantized) data is necessary in order to track the changing statistics and then adapt the quantizer setting in adaptive quantization. Through an analysis, Jayant showed that effective adaptations can be realized with an explicit memory of only one word. That is, either one input sample, x, in forward adaptive quantization or a quantized output, y, in backward adaptive quantization is sufficient (Jayant, 1973).

In (Jayant, 1984), examples on step-size adaptation (with the number of total reconstruction levels larger than four) were given. The idea is as follows. If at moment t_i the input sample x_i falls into the outer interval, then the step size at the next moment t_{i+1} will be enlarged by a factor of m_i (multiplying the current step size by m_i, $m_i > 1$). On the other hand, if the input x_i falls into an inner interval close to $x = 0$ then, the multiplier is less than 1, i.e., $m_i < 1$. That is, the multiplier m_i is small in the interval near $x = 0$ and monotonically increases for an increased x. Its range varies from a small positive number less than 1 to a number larger than 1. In this way, the quantizer adapts itself to the input to avoid *overload* as well as *underload* to achieve better performance.

2.4.4 Switched Quantization

This is another adaptive quantization scheme. A block diagram is shown in Figure 2.17. It consists of a bank of L quantizers. Each quantizer in the bank is fixed, but collectively they form a bank

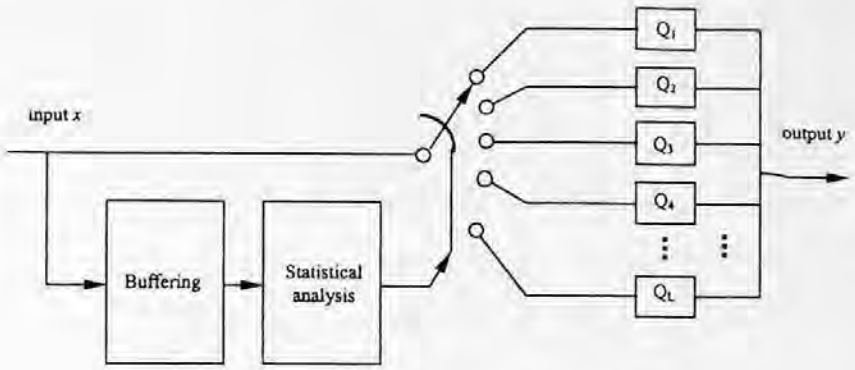


FIGURE 2.17 Switched quantization.

of quantizers with a variety of input-output characteristics. Based on a statistical analysis of recent input or output samples, a switch connects the current input to one of the quantizers in the bank such that the best possible performance may be achieved. It is reported that in both video and speech applications, this scheme has shown improved performance even when the number of quantizers in the bank, L, is two (Jayant and Noll, 1984). Interestingly, it is noted that as $L \to \infty$, the switched quantization converges to the adaptive quantizer discussed above.

2.5 PCM

Pulse code modulation (PCM) is closely related to quantization, the focus of this chapter. Furthermore, as pointed out in (Jayant, 1984), PCM is the earliest, best established, and most frequently applied coding system despite the fact that it is the most bit-consuming digitizing system (since it encodes each pixel independently) as well as being a very demanding system in terms of the bit error rate on the digital channel. Therefore, we discuss the PCM technique in this section.

PCM is now the most important form of pulse modulation. The other forms of pulse modulation are pulse amplitude modulation (PAM), pulse width modulation (PWM), and pulse position modulation (PPM), which are covered in most communications texts. Briefly speaking, pulse modulation links an analog signal to a pulse train in the following way. The analog signal is first sampled (a discretization in the time domain). The sampled values are used to modulate a pulse train. If the modulation is carried out through the amplitude of the pulse train, it is called PAM. If the modified parameter of the pulse train is the pulse width, we then have PWM. If the pulse width and magnitude are constant — only the position of pulses is modulated by the sample values — we then encounter PPM. An illustration of these pulse modulations is shown in Figure 2.18.

In PCM, an analog signal is first sampled. The sampled value is then quantized. Finally the quantized value is encoded, resulting in a bit steam. Figure 2.19 provides an example of PCM. We see that through a sampling and a uniform quantization the PCM system converts the input analog signal, which is continuous in both time and magnitude, into a digital signal (discretized in both time and magnitude) in the form of a natural binary code sequence. In this way, an analog signal modulates a pulse train with a natural binary code.

By far, PCM is more popular than other types of pulse modulation since the *code* modulation is much more robust against various noises than amplitude modulation, width modulation, and position modulation. In fact, almost all coding techniques include a PCM component. In digital image processing, given digital images usually appear in PCM format. It is known that an acceptable PCM representation of a monochrome picture requires six to eight bits per pixel (Huang, 1975). It is used so commonly in practice that its performance normally serves as a standard against which other coding techniques are compared.

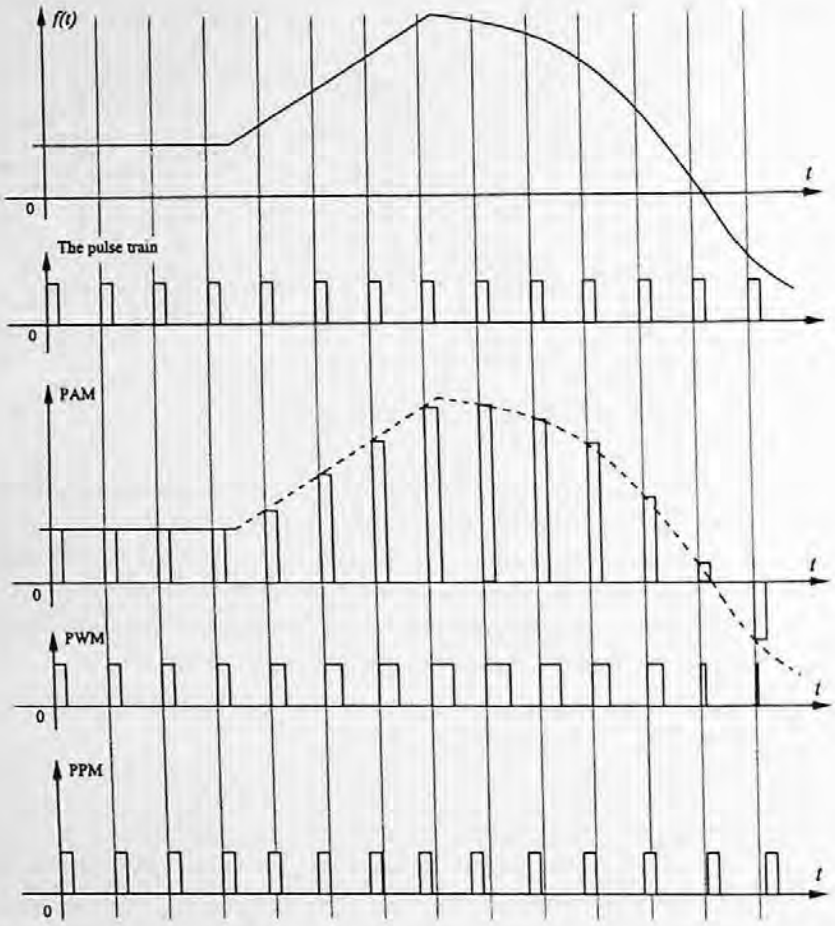


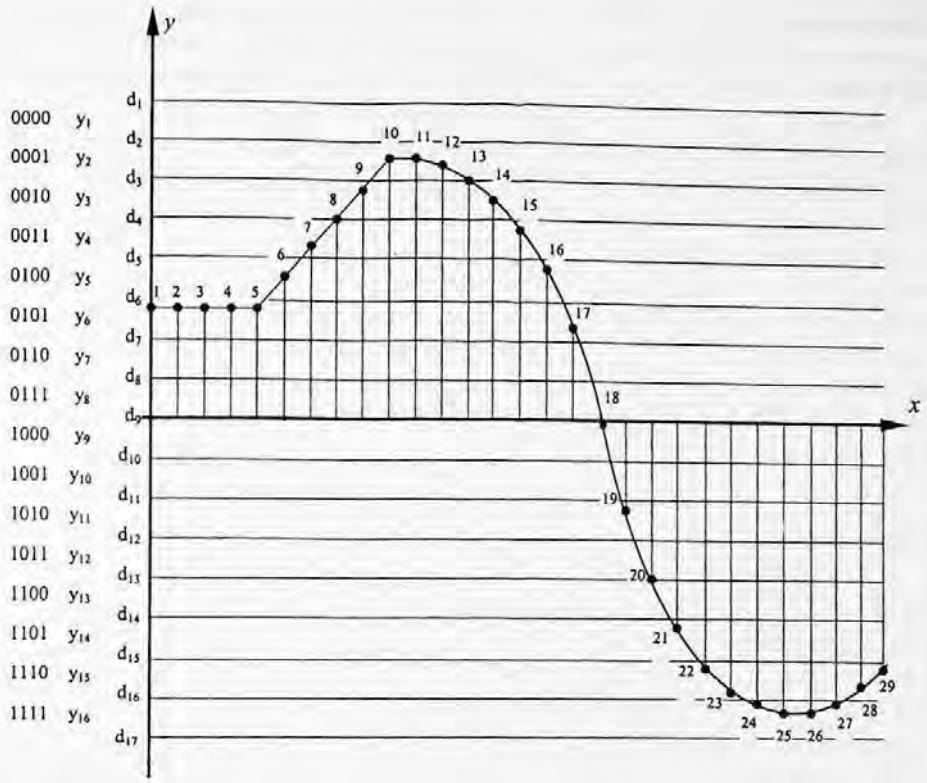
FIGURE 2.18 Pulse modulation.

Recall the false contouring phenomenon discussed in Chapter 1, when we discussed texture masking. It states that our eyes are more sensitive to relatively uniform regions in an image plane. If the number of reconstruction levels is not large enough (coarse quantization), then some unnatural contours will appear. When frequency masking was discussed, it was noted that by adding some high-frequency signal before quantization, the false contouring can be eliminated to a great extent. This technique is called dithering. The high-frequency signal used is referred to as a dither signal. Both false contouring and dithering were first reported in (Goodall, 1951).

2.6 SUMMARY

Quantization is a process in which a quantity having possibly an infinite number of different values is converted to another quantity having only finite many values. It is an important element in source encoding that has significant impact on both bit rate and distortion of reconstructed images and video in visual communication systems. Depending on whether the quantity is a scalar or a vector, quantization is called either scalar quantization or vector quantization. In this chapter we considered only scalar quantization.

Uniform quantization is the simplest and yet the most important case. In uniform quantization, except for outer intervals, both decision levels and reconstruction levels are uniformly spaced. Moreover, a reconstruction level is the arithmetic average of the two corresponding decision levels. In uniform quantization design, the step size is usually the only parameter that needs to be specified.



Output code (from left to right, from top to bottom):

0101	0101	0101	0101	0101	0100	0011	0011	0010	0001	0001	0001	0010	0010	0011
0100	0101	1000	1010	1100	1101	1110	1110	1111	1111	1111	1111	1110	1110	

FIGURE 2.19 Pulse code modulation (PCM).

Optimum quantization implies minimization of the mean square quantization error. When the input has a uniform distribution, uniform quantization is optimum. For the sake of simplicity, a uniform optimum quantizer is sometimes desired even when the input does not obey uniform distribution. The design under these circumstances involves an iterative procedure. The design problem in cases where the input has Gaussian, Lapacian, or Gamma distribution was solved and the parameters are available.

When the constraint of uniform quantization is removed, the conditions for optimum quantization are derived. The resultant optimum quantizer is normally nonuniform. An iterative procedure to solve the design is established and the optimum design parameters for Gaussian, Laplacian, and Gamma distribution are tabulated.

The companding technique is an alternative way to implement nonuniform quantization. Both nonuniform quantization and companding are time-invariant and hence not suitable for nonstationary input. Adaptive quantization deals with nonstationary input and combats the mismatch that occurs in optimum quantization design.

In adaptive quantization, buffering is necessary to store some recent input or sampled output data. A statistical analysis is carried out with respect to the stored recent data. Based on the analysis, the quantizer's parameters are adapted to changing input statistics to achieve better quantization performance. There are two types of adaptive quantization: forward and backward adaptive quantization. With the forward type, the statistical analysis is derived from the original input data, while with the backward type, quantization noise is involved in the analysis. Therefore, the forward

technique usually achieves more effective adaptation than the backward manner. The latter, however, does not need to send quantizer setting parameters as side information to the receiver side, since the output values of the quantization encoder (based on which the statistics are analyzed and the quantizer's parameters are adapted) are available in both the transmitter and receiver sides.

Switched quantization is another type of adaptive quantization. In this scheme, a bank of fixed quantizers is utilized, each quantizer having different input-output characteristics. A statistical analysis based on recent input decides which quantizer in the bank is suitable for the present input. The system then connects the input to this particular quantizer.

Nowadays, pulse code modulation is the most frequently used form of pulse modulation due to its robustness against noise. PCM consists of three stages: sampling, quantization, and encoding. Analog signals are first sampled with a proper sampling frequency. The sampled data are then quantized using a uniform quantizer. Finally, the quantized values are encoded with natural binary code. It is the best established and most applied coding system. Despite its bit-consuming feature, it is utilized in almost all coding systems.

2.7 EXERCISES

2-1. Using your own words, define quantization and uniform quantization. What are the two features of uniform quantization?

2-2. What is optimum quantization? Why is uniform quantization sometimes desired, even when the input has a *pdf* different from uniform? How was this problem solved? Draw an input-output characteristic of an optimum uniform quantizer with an input obeying Gaussian *pdf* having zero mean, unit variance, and the number of reconstruction levels, N, equal to 8.

2-3. What are the conditions of optimum nonuniform quantization? From Table 2.2, what observations can you make?

2-4. Define variance mismatch and *pdf* mismatch. Discuss how you can resolve the mismatch problem.

2-5. What is the difference between forward and backward adaptive quantization? Comment on the merits and drawbacks for each.

2-6. What are PAM, PWM, PPM, and PCM? Why is PCM the most popular type of pulse modulation?

REFERENCES

Fleischer, P. E. Sufficient conditions for achieving minimum distortion in quantizer, *IEEE Int. Convention Rec.*, 12, 104-111, 1964.

Gersho, A, Quantization, *IEEE Commun. Mag.*, pp. 6-29, September, 1977.

Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

Goodall, W. M. Television by pulse code modulation, *Bell Syst. Tech. J.*, 33-49, January 1951.

Huang, T. S. PCM picture transmission, *IEEE Spectrum*, 2, 57-63, 1965.

Jayant, N. S. Adaptive delta modulation with one-bit memory, *Bell Syst. Tech. J.*, 49, 321-342, 1970.

Jayant, N. S. Adaptive quantization with one word memory, *Bell Syst. Tech. J.*, 52, 1119-1144, 1973.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Li, W. and Y.-Q. Zhang, Vector-based signal processing and qunatization for image and video compression, *Proc. IEEE*, 83(2), 317-335, 1995.

Lloyd, S. P. Least squares quantization in PCM, Inst. Mathematical Statistics Meet., Atlantic City, NJ, September 1957; *IEEE Trans. Inf. Theory*, pp. 129-136, March 1982.

Max, J. Quantizing for minimum distortion, *IRE Trans. Inf. Theory*, it-6, 7-12, 1960.

Musmann, H. G. Predictive Image Coding, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

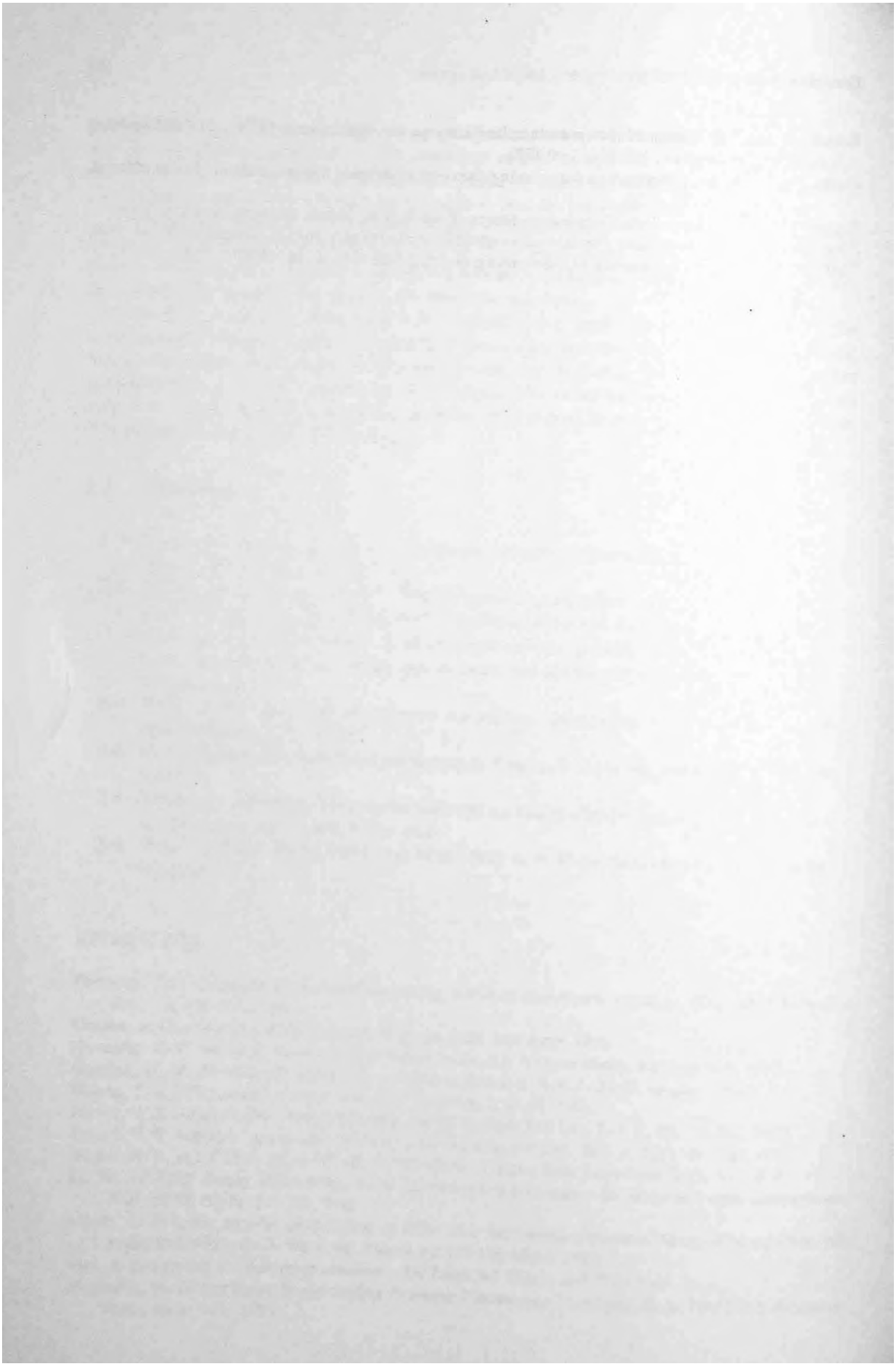
Paez, M. D. and T. H. Glisson, Minimum mean squared error qunatization in speech PCM and DPCM Systems, *IEEE Trans. Commun.*, 225-230, April 1972.

Panter, P. F. and W. Dite, Quantization distortion in pulse count modulation with nonuniform spacing of levels, *Proc. IRE*, 39, 44-48, 1951.

Sayood, K. *Introduction to Data Compression*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.

Sklar, B. *Digital Communications: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Smith, B. Instantaneous companding of quantized signals, *Bell Syst. Tech. J.*, 36, 653-709, 1957.



3 Differential Coding

Instead of encoding a signal directly, the *differential coding* technique codes the difference between the signal itself and its prediction. Therefore it is also known as *predictive coding*. By utilizing spatial and/or temporal interpixel correlation, differential coding is an efficient and yet computationally simple coding technique. In this chapter, we first describe the differential technique in general. Two components of differential coding, prediction and quantization, are discussed. There is an emphasis on (optimum) prediction, since quantization was discussed in Chapter 2. When the difference signal (also known as prediction error) is quantized, the differential coding is called differential pulse code modulation (DPCM). Some issues in DPCM are discussed, after which delta modulation (DM) as a special case of DPCM is covered. The idea of differential coding involving image sequences is briefly discussed in this chapter. More detailed coverage is presented in Sections III and IV, starting from Chapter 10. If quantization is not included, the differential coding is referred to as information-preserving differential coding. This is discussed at the end of the chapter.

3.1 INTRODUCTION TO DPCM

As depicted in Figure 2.3, a source encoder consists of the following three components: transformation, quantization, and codeword assignment. The transformation converts input into a format for quantization followed by codeword assignment. In other words, the component of transformation decides which format of input is to be encoded. As mentioned in the previous chapter, input itself is not necessarily the most suitable format for encoding.

Consider the case of monochrome image encoding. The input is usually a 2-D array of gray level values of an image obtained via PCM coding. The concept of spatial redundancy, discussed in Section 1.2.1.1, tells us that neighboring pixels of an image are usually highly correlated. Therefore, it is more efficient to encode the gray difference between two neighboring pixels instead of encoding the gray level values of each pixel. At the receiver, the decoded difference is added back to reconstruct the gray level value of the pixel. Since neighboring pixels are highly correlated, their gray level values bear a great similarity. Hence, we expect that the variance of the difference signal will be smaller than that of the original signal. Assume uniform quantization and natural binary coding for the sake of simplicity. Then we see that for the same bit rate (bits per sample) the quantization error will be smaller, i.e., a higher quality of reconstructed signal can be achieved. Or, for the same quality of reconstructed signal, we need a lower bit rate.

3.1.1 Simple Pixel-to-Pixel DPCM

Denote the gray level values of pixels along a row of an image as z_i, $i = 1, \cdots, M$, where M is the total number of pixels within the row. Using the immediately preceding pixel's gray level value, z_{i-1}, as a prediction of that of the present pixel, $\hat{z}_i$, i.e.,

$$\hat{z}_i = z_{i-1} \tag{3.1}$$

we then have the difference signal

$$d_i = z_i - \hat{z}_i = z_i - z_{i-1} \tag{3.2}$$

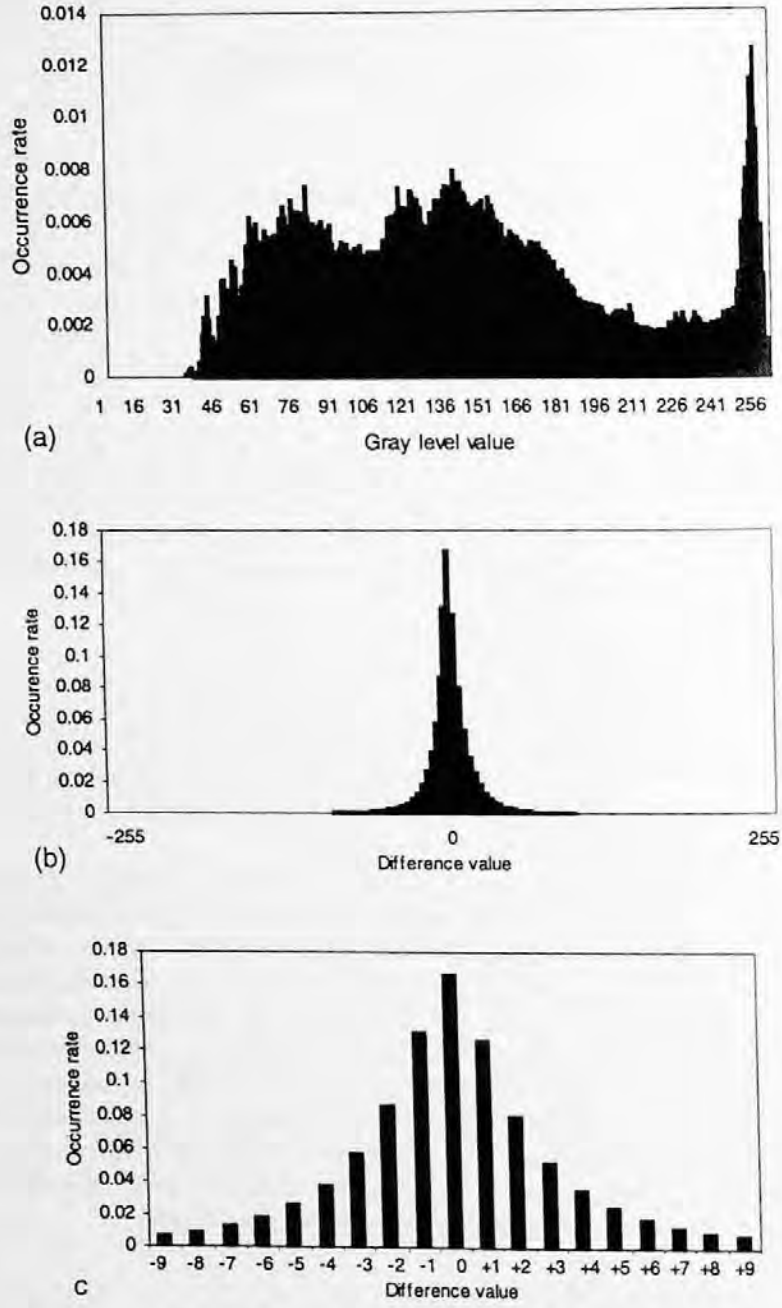


FIGURE 3.1 (a) Histogram of the original "boy and girl" image. (b) Histogram of the difference image obtained by using horizontal pixel-to-pixel differencing. (c) A close-up of the central portion of the histogram of the difference image.

Assume a bit rate of eight bits per sample in the quantization. We can see that although the dynamic range of the difference signal is theoretically doubled, from 256 to 512, the variance of the difference signal is actually much smaller. This can be confirmed from the histograms of the "boy and girl" image (refer to Figure 1.1) and its difference image obtained by horizontal pixel-to-pixel differencing, shown in Figure 3.1(a) and (b), respectively. Figure 3.1(b) and its close-up (c) indicate that by a rate of 42.44% the difference values fall into the range of −1, 0, and +1. In other words, the histogram of the difference signal is much more narrowly concentrated than that of the original signal.

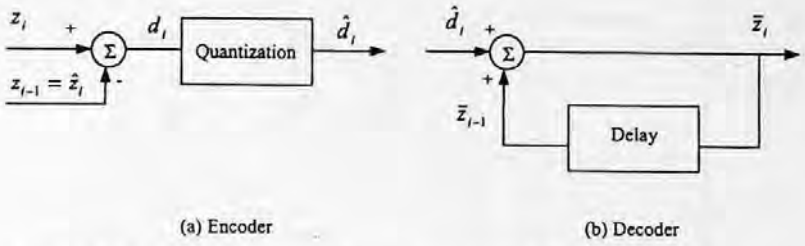


FIGURE 3.2 Block diagram of a pixel-to-pixel differential coding system.

A block diagram of the scheme described above is shown in Figure 3.2. There z_i denotes the sequence of pixels along a row, d_i is the corresponding difference signal, and $\hat{d}_i$ is the quantized version of the difference, i.e.,

$$\hat{d}_i = Q(d_i) = d_i + e_q \tag{3.3}$$

where e_q represents the quantization error. In the decoder, $\bar{z}_i$ represents the reconstructed pixel gray value, and we have

$$\bar{z}_i = \bar{z}_{i-1} + \hat{d}_i \tag{3.4}$$

This simple scheme, however, suffers from an accumulated quantization error. We can see this clearly from the following derivation (Sayood, 1996), where we assume the initial value z_0 is available for both the encoder and the decoder.

$$\text{as} \quad i = 1, \quad d_1 = z_1 - z_0$$

$$\hat{d}_1 = d_1 + e_{q,1} \tag{3.5}$$

$$\bar{z}_1 = z_0 + \hat{d}_1 = z_0 + d_1 + e_{q,1} = z_1 + e_{q,1}$$

Similarly, we can have

$$as \quad i = 2, \quad \bar{z}_2 = z_2 + e_{q,1} + e_{q,2} \tag{3.6}$$

and, in general,

$$\bar{z}_i = z_i + \sum_{j=1}^{i} e_{q,j} \tag{3.7}$$

This problem can be remedied by the following scheme, shown in Figure 3.3. Now we see that in both the encoder and the decoder, the reconstructed signal is generated in the same way, i.e.,

$$\bar{z}_i = \bar{z}_{i-1} + \hat{d}_i \tag{3.8}$$

and in the encoder the difference signal changes to

$$d_i = z_i - \bar{z}_{i-1} \tag{3.9}$$

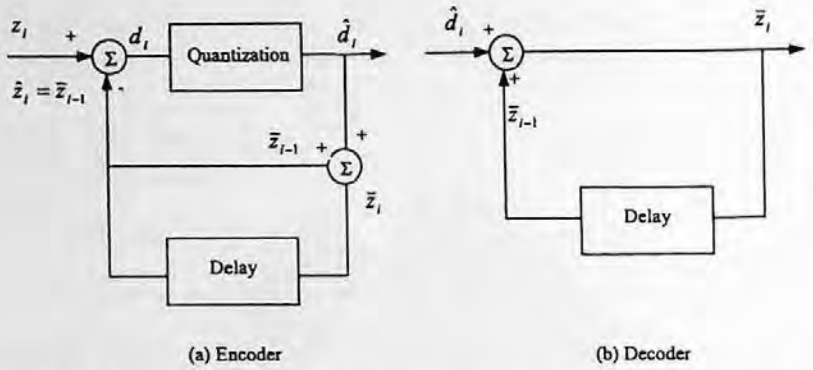


FIGURE 3.3 Block diagram of a practical pixel-to-pixel differential coding system.

Thus, the previously reconstructed $\bar{z}_{i-1}$ is used as the predictor, $\hat{z}_i$, i.e.,

$$\hat{z}_i = \bar{z}_{i-1}. \tag{3.10}$$

In this way, we have

$$\begin{aligned} as \quad i=1, \quad & d_1 = z_1 - z_0 \\ & \hat{d}_1 = d_1 + e_{q,1} \\ & \bar{z}_1 = z_0 + \hat{d}_1 = z_0 + d_1 + e_{q,1} = z_1 + e_{q,1} \end{aligned} \tag{3.11}$$

Similarly, we have

$$\begin{aligned} as \quad i=2, \quad & d_2 = z_2 - \bar{z}_1 \\ & \hat{d}_2 = d_2 + e_{q,2} \\ & \bar{z}_2 = \bar{z}_1 + \hat{d}_2 = z_2 + e_{q,2} \end{aligned} \tag{3.12}$$

In general,

$$\bar{z}_i = z_i + e_{q,i} \tag{3.13}$$

Thus, we see that the problem of the quantization error accumulation has been resolved by having both the encoder and the decoder work in the same fashion, as indicated in Figure 3.3, or in Equations 3.3, 3.9, and 3.10.

3.1.2 General DPCM Systems

In the above discussion, we can view the reconstructed neighboring pixel's gray value as a prediction of that of the pixel being coded. Now, we generalize this simple pixel-to-pixel DPCM. In a general DPCM system, a pixel's gray level value is first predicted from the preceding reconstructed pixels' gray level values. The difference between the pixel's gray level value and the predicted value is then quantized. Finally, the quantized difference is encoded and transmitted to the receiver. A block

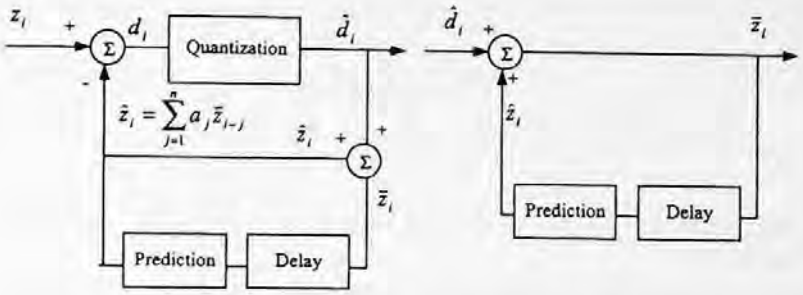


FIGURE 3.4 Block diagram of a general DPCM system.

diagram of this general differential coding scheme is shown in Figure 3.4, where the codeword assignment in the encoder and its counterpart in decoder are not included.

It is noted that, instead of using the previously reconstructed sample, $\bar{z}_{i-1}$, as a predictor, we now have the predicted version of z_i, $\hat{z}_i$, as a function of the n previously reconstructed samples, $\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}$. That is,

$$\hat{z}_i = f\left(\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}\right) \tag{3.14}$$

Linear prediction, i.e., that the function f in Equation 3.14 is linear, is of particular interest and is widely used in differential coding. In linear prediction, we have

$$\hat{z}_i = \sum_{j=1}^{n} a_j \bar{z}_{i-j} \tag{3.15}$$

where a_j are real parameters. Hence, we see that the simple pixel-to-pixel differential coding is a special case of general differential coding with linear prediction, i.e., $n = 1$ and $a_1 = 1$.

In Figure 3.4, d_i is the difference signal and is equal to the difference between the original signal, z_i, and the prediction $\hat{z}_i$. That is,

$$d_i = z_i - \hat{z}_i \tag{3.16}$$

The quantized version of d_i is denoted by $\hat{d}_i$. The reconstructed version of z_i is represented by $\bar{z}_i$, and

$$\bar{z}_i = \hat{z}_i + \hat{d}_i \tag{3.17}$$

Note that this is true for both the encoder and the decoder. Recall that the accumulation of the quantization error can be remedied by using this method.

The difference between the original input and the predicted input is called prediction error, which is denoted by e_p. That is,

$$e_p = z_i - \hat{z}_i \tag{3.18}$$

where the e_p is understood as the prediction error associated with the index i. Quantization error, e_q, is equal to the reconstruction error or coding error, e_r, defined as the difference between the original signal, z_i, and the reconstructed signal, $\bar{z}_i$, when the transmission is error free:

$$\begin{aligned} e_q &= d_i - \hat{d}_i \\ &= \left(z_i - \hat{z}_i\right) - \left(\bar{z}_i - \hat{z}_i\right) \\ &= z_i - \bar{z}_i = e_r \end{aligned} \tag{3.19}$$

This indicates that quantization error is the only source of information loss with an error-free transmission channel.

The DPCM system depicted in Figure 3.4 is also called closed-loop DPCM with feedback around the quantizer (Jayant, 1984). This term reflects the feature in DPCM structure.

Before we leave this section, let us take a look at the history of the development of differential image coding. According to an excellent early article on differential image coding (Musmann, 1979), the first theoretical and experimental approaches to image coding involving linear prediction began in 1952 at the Bell Telephone Laboratories (Oliver, 1952; Kretzmer, 1952; Harrison, 1952). The concepts of DPCM and DM were also developed in 1952 (Cutler, 1952; Dejager, 1952). Predictive coding capable of preserving information for a PCM signal was established at the Massachusetts Institute of Technology (Elias, 1955).

The differential coding technique has played an important role in image and video coding. In the international coding standard for still images, JPEG (covered in Chapter 7), we can see that differential coding is used in the lossless mode and in the DCT-based mode for coding DC coefficients. Motion-compensated (MC) coding has been a major development in video coding since the 1980s and has been adopted by all the international video coding standards such as H.261 and H.263 (covered in Chapter 19), MPEG 1 and MPEG 2 (covered in Chapter 16). MC coding is essentially a predictive coding technique applied to video sequences involving displacement motion vectors.

3.2 OPTIMUM LINEAR PREDICTION

Figure 3.4 demonstrates that a differential coding system consists of two major components: prediction and quantization. Quantization was discussed in the previous chapter. Hence, in this chapter we emphasize prediction. Below, we formulate an optimum linear prediction problem and then present a theoretical solution to the problem.

3.2.1 FORMULATION

Optimum linear prediction can be formulated as follows. Consider a discrete-time random process z. At a typical moment i, it is a random variable z_i. We have n previous observations $\bar{z}_{i-1}, \bar{z}_{i-2}, \cdots, \bar{z}_{i-n}$ available and would like to form a prediction of z_i, denoted by $\hat{z}_i$. The output of the predictor, $\hat{z}_i$, is a linear function of the n previous observations. That is,

$$\hat{z}_i = \sum_{j=1}^{n} a_j \bar{z}_{i-j} \tag{3.20}$$

with a_j, $j = 1,2,\cdots,n$ being a set of real coefficients. An illustration of a linear predictor is shown in Figure 3.5. As defined above, the prediction error, e_p, is

$$e_p = z_i - \hat{z}_i \tag{3.21}$$

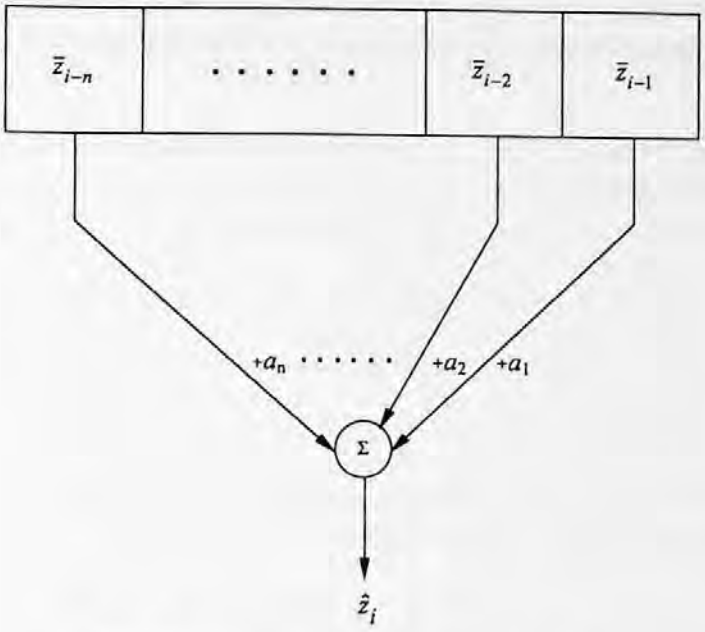


FIGURE 3.5 An illustration of a linear predictor.

The mean square prediction error, MSE_p, is

$$MSE_p = E\left[\left(e_p\right)^2\right] = E\left[\left(z_i - \hat{z}_i\right)^2\right] \tag{3.22}$$

The optimum prediction, then, refers to the determination of a set of coefficients a_j, $j = 1,2,\cdots,n$ such that the mean square prediction error, MSE_p, is minimized.

This optimization problem turns out to be computationally intractable for most practical cases due to the feedback around the quantizer shown in Figure 3.4, and the nonlinear nature of the quantizer. Therefore, the optimization problem is solved in two separate stages. That is, the best linear predictor is first designed ignoring the quantizer. Then, the quantizer is optimized for the distribution of the difference signal (Habibi, 1971). Although the predictor thus designed is suboptimal, ignoring the quantizer in the optimum predictor design allows us to substitute the reconstructed $\bar{z}_{i-j}$ by z_{i-j} for $j = 1,2,\cdots,n$, according to Equation 3.20. Consequently, we can apply the theory of optimum linear prediction to handle the design of the optimum predictor as shown below.

3.2.2 Orthogonality Condition and Minimum Mean Square Error

By taking the differentiation of MSE_p with respect to coefficient a_js, one can derive the following necessary conditions, which are usually referred to as the *orthogonality condition*:

$$E\left[e_p \cdot z_{i-j}\right] = 0 \quad for \quad j = 1,2,\cdots,n \tag{3.23}$$

The interpretation of Equation 3.23 is that the prediction error, e_p, must be orthogonal to all the observations, which are now the preceding samples: z_{i-j}, $j = 1,2,\cdots,n$ according to our discussion in Section 3.2.1. These are equivalent to

$$R_z(m) = \sum_{j=1}^{n} a_j R_z(m-j) \quad for \quad m = 1,2,\cdots,n \tag{3.24}$$

where R_z represents the autocorrelation function of z. In a vector-matrix format, the above orthogonal conditions can be written as

$$\begin{bmatrix} R_z(1) \\ R_z(2) \\ \vdots \\ \vdots \\ R_z(n) \end{bmatrix} = \begin{bmatrix} R_z(0) & R_z(1) & \cdots & \cdots & R_z(n-1) \\ R_z(1) & R_z(0) & \cdots & \cdots & R_z(n-2) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ R_z(n-1) & R_z(n) & \cdots & \cdots & R_z(0) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} \tag{3.25}$$

Equations 3.24 and 3.25 are called Yule-Walker equations.

The minimum mean square prediction error is then found to be

$$MSE_p = R_z(0) - \sum_{j=1}^{n} a_j R_z(j) \tag{3.26}$$

These results can be found in texts dealing with random processes, e.g., in (Leon-Garcia, 1994).

3.2.3 Solution to Yule-Walker Equations

Once autocorrelation data are available, the Yule-Walker equation can be solved by matrix inversion. A recursive procedure was developed by Levinson to solve the Yule-Walker equations (Leon-Garcia, 1993). When the number of previous samples used in the linear predictor is large, i.e., the dimension of the matrix is high, the Levinson recursive algorithm becomes more attractive. Note that in the field of image coding the autocorrelation function of various types of video frames is derived from measurements (O'Neal, 1966; Habibi, 1971).

3.3 SOME ISSUES IN THE IMPLEMENTATION OF DPCM

Several related issues in the implementation of DPCM are discussed in this section.

3.3.1 Optimum DPCM System

Since DPCM consists mainly of two parts, prediction and quantization, its optimization should not be carried out separately. The interaction between the two parts is quite complicated, however, and thus combined optimization of the whole DPCM system is difficult. Fortunately, with the mean square error criterion, the relation between quantization error and prediction error has been found:

$$MSE_q \approx \frac{9}{2N^2} MSE_p \tag{3.27}$$

where N is the total number of reconstruction levels in the quantizer (O'Neal, 1966; Musmann, 1979). That is, the mean square error of quantization is approximately proportional to the mean square error of prediction. With this approximation, we can optimize the two parts separately, as mentioned in Section 3.2.1. While the optimization of quantization was addressed in Chapter 2, the

optimum predictor was discussed in Section 3.2. A large amount of work has been done on this subject. For instance, the optimum predictor for color image coding was designed and tested in (Pirsch and Stenger, 1977).

3.3.2 1-D, 2-D, AND 3-D DPCM

In Section 3.1.2, we expressed linear prediction in Equation 3.15. However, so far we have not really discussed how to predict a pixel's gray level value by using its neighboring pixels' coded gray level values.

Recall that a practical pixel-to-pixel differential coding system was discussed in Section 3.1.1. There, the reconstructed intensity of the immediately preceding pixel along the same scan line is used as a prediction of the pixel intensity being coded. This type of differential coding is referred to as 1-D DPCM. In general, 1-D DPCM may use the reconstructed gray level values of more than one of the preceding pixels within the same scan line to predict that of a pixel being coded. By far, however, the immediately preceding pixel in the same scan line is most frequently used in 1-D DPCM. That is, pixel A in Figure 3.6 is often used as a prediction of pixel Z, which is being DPCM coded.

Sometimes in DPCM image coding, both the decoded intensity values of adjacent pixels within the same scan line and the decoded intensity values of neighboring pixels in different scan lines are involved in the prediction. This is called 2-D DPCM. A typical pixel arrangement in 2-D predictive coding is shown in Figure 3.6. Note that the pixels involved in the prediction are restricted to be either in the lines above the line where the pixel being coded, Z, is located or on the left-hand side of pixel Z if they are in the same line. Traditionally, a TV frame is scanned from top to bottom and from left to right. Hence, the above restriction indicates that only those pixels, which have been coded and available in both the transmitter and the receiver, are used in the prediction. In 2-D system theory, this support is referred to as recursively computable (Bose, 1982). An often-used 2-D prediction involves pixels A, D, and E.

Obviously, 2-D predictive coding utilizes not only the spatial correlation existing within a scan line but also that existing in neighboring scan lines. In other words, the spatial correlation is utilized both horizontally and vertically. It was reported that 2-D predictive coding outperforms 1-D predictive coding by decreasing the prediction error by a factor of two, or equivalently, 3dB in *SNR*. The improvement in subjective assessment is even larger (Musmann, 1979). Furthermore, the transmission error in 2-D predictive image coding is much less severe than in 1-D predictive image coding. This is discussed in Section 3.6.

In the context of image sequences, neighboring pixels may be located not only in the same image frame but also in successive frames. That is, neighboring pixels along the time dimension are also involved. If the prediction of a DPCM system involves three types of neighboring pixels: those along the same scan line, those in the different scan lines of the same image frame, and those

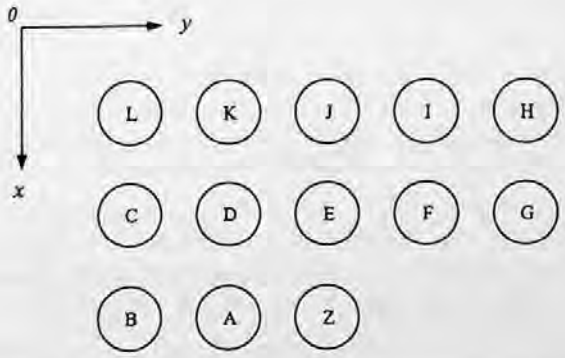


FIGURE 3.6 Pixel arrangement in 1-D and 2-D prediction.

in the different frames, the DPCM is then called 3-D differential coding. It will be discussed in Section 3.5.

3.3.3 Order of Predictor

The number of coefficients in the linear prediction, n, is referred to as the order of the predictor. The relation between the mean square prediction error, MSE_p, and the order of the predictor, n, has been studied. As shown in Figure 3.7, the MSE_p decreases quite effectively as n increases, but the performance improvement becomes negligible as $n > 3$ (Habibi, 1971).

3.3.4 Adaptive Prediction

Adaptive DPCM means adaptive prediction and adaptive quantization. As adaptive quantization was discussed in Chapter 2, here we discuss adaptive prediction only.

Similar to the discussion on adaptive quantization, adaptive prediction can be done in two different ways: forward adaptive and backward adaptive prediction. In the former, adaptation is based on the input of a DPCM system, while in the latter, adaptation is based on the output of the DPCM. Therefore, forward adaptive prediction is more sensitive to changes in local statistics. Prediction parameters (the coefficients of the predictor), however, need to be transmitted as side information to the decoder. On the other hand, quantization error is involved in backward adaptive prediction. Hence, the adaptation is less sensitive to local changing statistics. But, it does not need to transmit side information.

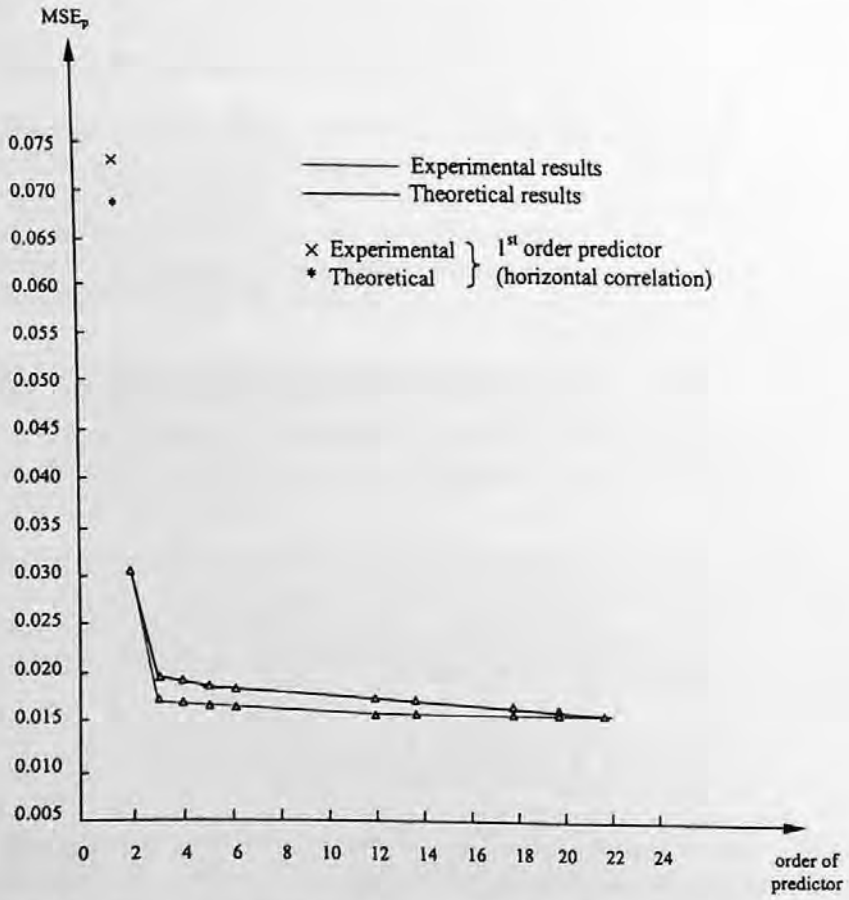


FIGURE 3.7 Mean square prediction error vs. order of predictor. (Data from Habibi, 1971.)

In either case, the data (either input or output) have to be buffered. Autocorrelation coefficients are analyzed, based on which prediction parameters are determined.

3.3.5 Effect of Transmission Errors

Transmission errors caused by channel noise may reverse the binary bit information from 0 to 1 or 1 to 0 with what is known as *bit error probability*, or *bit error rate*. The effect of transmission errors on reconstructed images varies depending on different coding techniques.

In the case of the PCM-coding technique, each pixel is coded independently of the others. Therefore bit reversal in the transmission only affects the gray level value of the corresponding pixel in the reconstructed image. It does not affect other pixels in the reconstructed image.

In DPCM, however, the effect caused by transmission errors becomes more severe. Consider a bit reversal occurring in transmission. It causes error in the corresponding pixel. But, this is not the end of the effect. The affected pixel causes errors in reconstructing those pixels towards which the erroneous gray level value was used in the prediction. In this way, the transmission error propagates.

Interestingly, it is reported that the error propagation is more severe in 1-D differential image coding than in 2-D differential coding. This may be explained as follows. In 1-D differential coding, usually only the immediate preceding pixel in the same scan line is involved in prediction. Therefore, an error will be propagated along the scan line until the beginning of the next line, where the pixel gray level value is reinitialized. In 2-D differential coding, the prediction of a pixel gray level value depends not only on the reconstructed gray level values of pixels along the same scan line but also on the reconstructed gray level values of the vertical neighbors. Hence, the effect caused by a bit reversal transmission error is less severe than in the 1-D differential coding.

For this reason, the bit error rate required by DPCM coding is lower than that required by PCM coding. For instance, while a bit error rate less than $5 \cdot 10^{-6}$ is normally required for PCM to provide broadcast TV quality, for the same application a bit error rate less than 10^{-7} and 10^{-9} are required for DPCM coding with 2-D prediction and 1-D prediction, respectively (Musmann, 1979).

Channel encoding with an error correction capability was applied to lower the bit error rate. For instance, to lower the bit error rate from the order of 10^{-6} to the order of 10^{-9} for DPCM coding with 1-D prediction, an error correction by adding 3% redundancy in channel coding has been used (Bruders, 1978).

3.4 DELTA MODULATION

Delta modulation (DM) is an important, simple, special case of DPCM, as discussed above. It has been widely applied and is thus an important coding technique in and of itself.

The above discussion and characterization of DPCM systems are applicable to DM systems. This is because DM is essentially a special type of DPCM, with the following two features.

1. The linear predictor is of the first order, with the coefficient a_1 equal to 1.
2. The quantizer is a one-bit quantizer. That is, depending on whether the difference signal is positive or negative, the output is either $+\Delta/2$ or $-\Delta/2$.

To perceive these two features, let us take a look at the block diagram of a DM system and the input-output characteristic of its one-bit quantizer, shown in Figures 3.8 and 3.9, respectively. Due to the first feature listed above, we have:

$$\hat{z}_i = \bar{z}_{i-1} \tag{3.28}$$

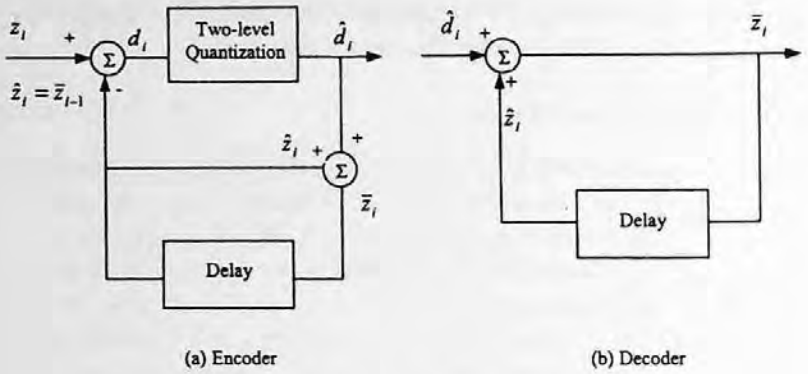


FIGURE 3.8 Block diagram of DM systems.

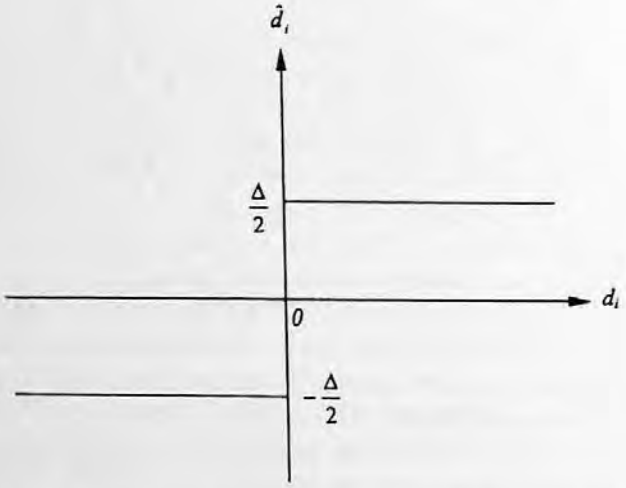


FIGURE 3.9 Input-output characteristic of two-level quantization in DM.

Next, we see that there are only two reconstruction levels in quantization because of the second feature. That is,

$$\hat{d}_i = \begin{cases} +\Delta/2 & if \quad z_i > \bar{z}_{i-1} \\ -\Delta/2 & if \quad z_i < \bar{z}_{i-1} \end{cases} \tag{3.29}$$

From the relation between the reconstructed value and the predicted value of DPCM, discussed above, and the fact that DM is a special case of DPCM, we have

$$\bar{z}_i = \hat{z}_i + \hat{d}_i \tag{3.30}$$

Combining Equations 3.28, 3.29, and 3.30, we have

$$\bar{z}_i = \begin{cases} \bar{z}_{i-1} + \Delta/2 & if \quad z_i > \bar{z}_{i-1} \\ \bar{z}_{i-1} - \Delta/2 & if \quad z_i < \bar{z}_{i-1} \end{cases} \tag{3.31}$$

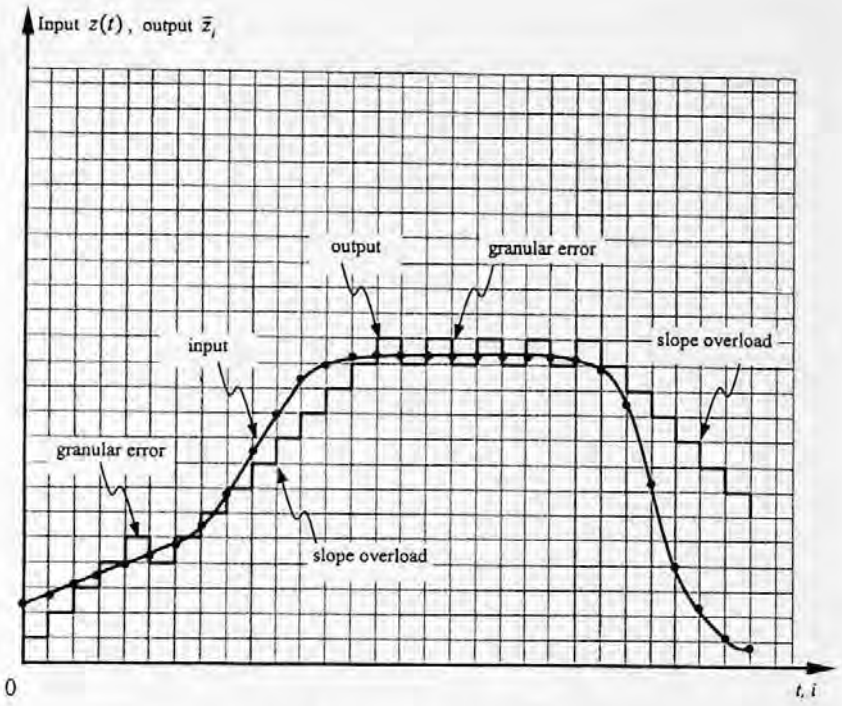


FIGURE 3.10 DM with fixed step size.

The above mathematical relationships are of importance in understanding DM systems. For instance, Equation 3.31 indicates that the step size Δ of DM is a crucial parameter. We notice that a large step size compared with the magnitude of the difference signal causes granular error, as shown in Figure 3.10. Therefore, in order to reduce the granular error we should choose a small step size. On the other hand, a small step size compared with the magnitude of the difference signal will lead to the overload error discussed in Chapter 2 for quantization. Since in DM systems it is the difference signal that is quantized, the overload error in DM becomes *slope overload* error, as shown in Figure 3.10. That is, it takes a while (multiple steps) for the reconstructed samples to catch up with the sudden change in input. Therefore, the step size should be large in order to avoid the slope overload. Considering these two conflicting factors, a proper compromise in choosing the step size is common practice in DM.

To improve the performance of DM, an oversampling technique is often applied. That is, the input is oversampled prior to the application of DM. By oversampling, we mean that the sampling frequency is higher than the sampling frequency used in obtaining the original input signal. The increased sample density caused by oversampling decreases the magnitude of the difference signal. Consequently, a relatively small step size can be used so as to decrease the granular noise without increasing the slope overload error. Thus, the resolution of the DM-coded image is kept the same as that of the original input (Jayant, 1984; Lim, 1990).

To achieve better performance for changing inputs, an adaptive technique can be applied in DM. That is, either input (forward adaptation) or output (backward adaptation) data are buffered and the data variation is analyzed. The step size is then chosen accordingly. If it is forward adaptation, side information is required for transmission to the decoder. Figure 3.11 demonstrates step size adaptation. We see the same input as that shown in Figure 3.10. But, the step size is now not fixed. Instead, the step size is adapted according to the varying input. When the input changes with a large slope, the step size increases to avoid the slope overload error. On the other hand, when the input changes slowly, the step size decreases to reduce the granular error.

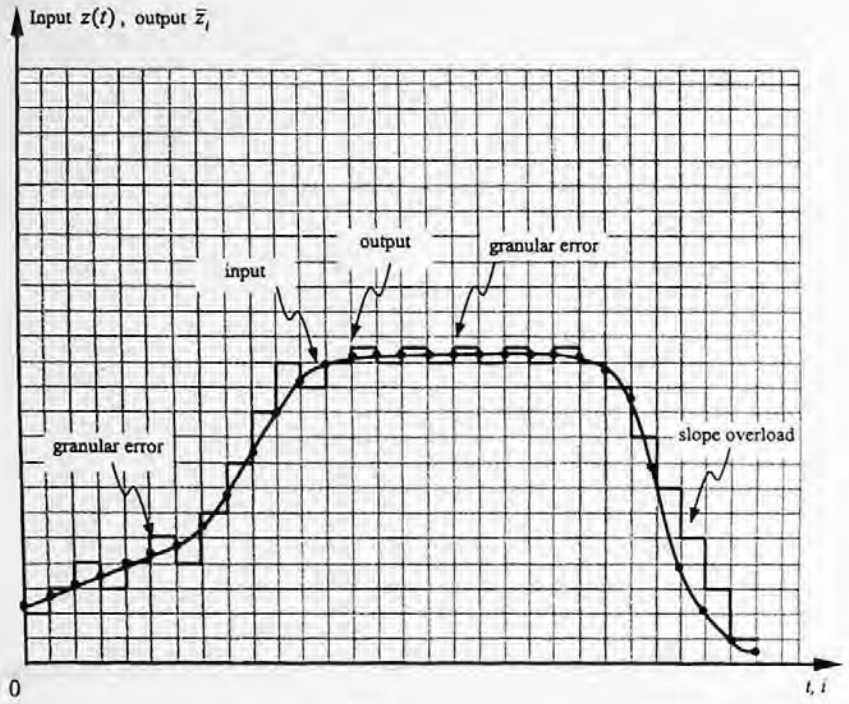


FIGURE 3.11 Adaptive DM.

3.5 INTERFRAME DIFFERENTIAL CODING

As was mentioned in Section 3.3.2, 3-D differential coding involves an image sequence. Consider a sensor located in 3-D world space. For instance, in applications such as videophony and video-conferencing, the sensor is fixed in position for a while and it takes pictures. As time goes by, the images form a temporal image sequence. The coding of such an image sequence is referred to as interframe coding. The subject of image sequence and video coding is addressed in Sections III and IV. In this section, we briefly discuss how differential coding is applied to interframe coding.

3.5.1 CONDITIONAL REPLENISHMENT

Recognizing the great similarity between consecutive TV frames, a conditional replenishment coding technique was proposed and developed (Mounts, 1969). It was regarded as one of the first real demonstrations of interframe coding exploiting interframe redundancy (Netravali and Robbins, 1979).

In this scheme, the previous frame is used as a reference for the present frame. Consider a pair of pixels: one in the previous frame, the other in the present frame — both occupying the same spatial position in the frames. If the gray level difference between the pair of pixels exceeds a certain criterion, then the pixel is considered a *changing* pixel. The present pixel gray level value and its position information are transmitted to the receiving side, where the pixel is replenished. Otherwise, the pixel is considered *unchanged*. At the receiver its previous gray level is repeated. A block diagram of conditional replenishment is shown in Figure 3.12. There, a frame memory unit in the transmitter is used to store frames. The differencing and thresholding of corresponding pixels in two consecutive frames can then be conducted there. A buffer in the transmitter is used to smooth the transmission data rate. This is necessary because the data rate varies from region to region within an image frame and from frame to frame within an image sequence. A buffer in the receiver is needed for a similar consideration. In the frame memory unit, the replenishment is

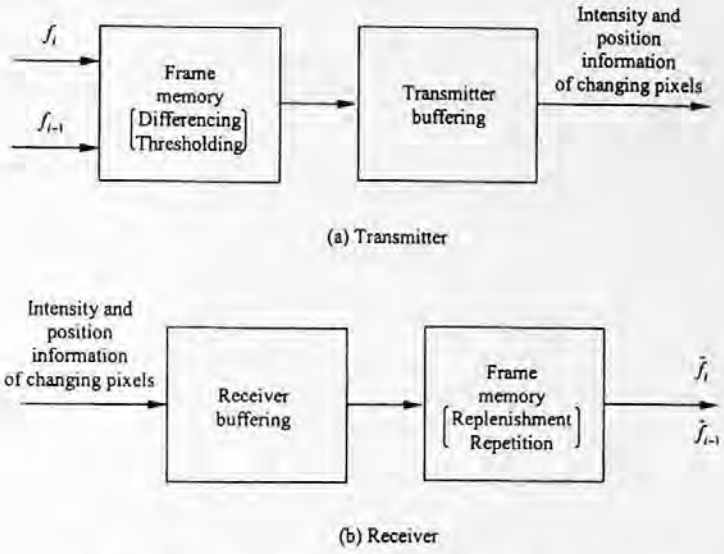


FIGURE 3.12 Block diagram of conditional replenishment.

carried out for the changing pixels and the gray level values in the receiver are repeated for the unchanged pixels.

With conditional replenishment, a considerable savings in bit rate was achieved in applications such as videophony, videoconferencing, and TV broadcasting. Experiments in real time, using the head-and-shoulder view of a person in animated conversation as the video source, demonstrated an average bit rate of 1 bit/pixel with a quality of reconstructed video comparable with standard 8 bit/pixel PCM transmission (Mounts, 1969). Compared with pixel-to-pixel 1-D DPCM, the most popularly used coding technique at the time, the conditional replenishment technique is more efficient due to the exploitation of high interframe redundancy. As pointed in (Mounts, 1969), there is more correlation between television pixels along the frame-to-frame temporal dimension than there is between adjacent pixels within a signal frame. That is, the temporal redundancy is normally higher than spatial redundancy for TV signals.

Tremendous efforts have been made to improve the efficiency of this rudimentary technique. For an excellent review, readers are referred to (Haskell et al., 1972, 1979). 3-D DPCM coding is among the improvements and is discussed next.

3.5.2 3-D DPCM

It was soon realized that it is more efficient to transmit the gray level difference than to transmit the gray level itself, resulting in interframe differential coding. Furthermore, instead of treating each pixel independently of its neighboring pixels, it is more efficient to utilize spatial redundancy as well as temporal redundancy, resulting in 3-D DPCM.

Consider two consecutive TV frames, each consisting of an odd and an even field. Figure 3.13 demonstrates the small neighborhood of a pixel, Z, in the context. As with the 1-D and 2-D DPCM discussed before, the prediction can only be based on the previously encoded pixels. If the pixel under consideration, Z, is located in the even field of the present frame, then the odd field of the present frame and both odd and even fields of the previous frame are available. As mentioned in Section 3.3.2, it is assumed that in the even field of the present frame, only those pixels in the lines above the line where pixel Z lies and those pixels left of the Z in the line where Z lies are used for prediction.

Table 3.1 lists several utilized linear prediction schemes. It is recognized that the case of *element difference* is a 1-D predictor since the immediately preceding pixel is used as the predictor. The

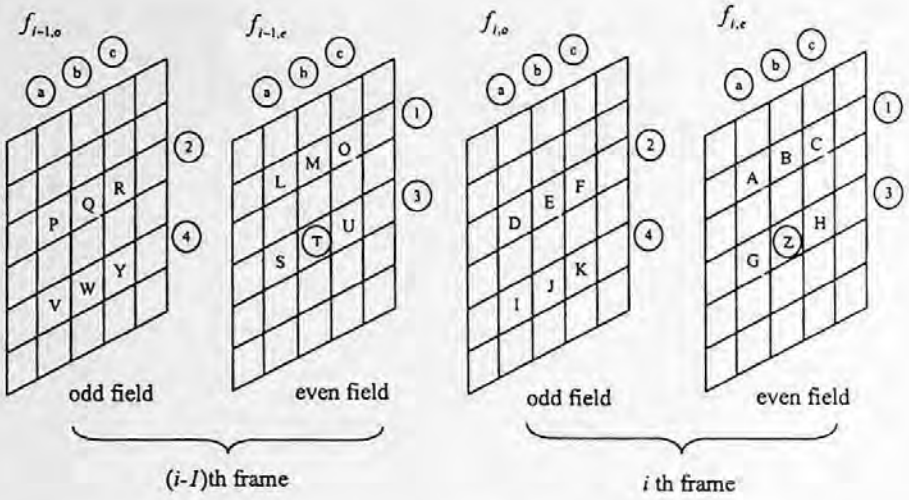


FIGURE 3.13 Pixel arrangement in two TV frames. (After Haskell, 1979.)

field difference is defined as the arithmetic average of two immediately vertical neighboring pixels in the previous odd field. Since the odd field is generated first, followed by the even field, this predictor cannot be regarded as a pure 2-D predictor. Instead, it should be considered a 3-D predictor. The remaining cases are all 3-D predictors. One thing is common in all the cases: the gray levels of pixels used in the prediction have already been coded and thus are available in both the transmitter and the receiver. The prediction error of each changing pixel Z identified in thresholding process is then quantized and coded.

An analysis of the relationship between the entropy of moving areas (bits per changing pixel) and the speed of the motion (pixels per frame interval) in an image containing a moving mannequin's head was studied with different linear predictions, as listed in Table 3.1 in Haskell (1979). It was found that the element difference of field difference generally corresponds to the lowest entropy, meaning that this prediction is the most efficient. The frame difference and element difference correspond to higher entropy. It is recognized that, in the circumstances, transmission error will be propagated if the pixels in the previous line are used in prediction (Connor, 1973). Hence, the linear predictor should use only pixels from the same line or the same line in the previous frame when bit reversal error in transmission needs to be considered. Combining these two factors, the element difference of frame difference prediction is preferred.

TABLE 3.1
Some Linear Prediction Schemes. (After Haskell, 1979).

	Original signal (Z)	Prediction signal ($\hat{Z}$)	Differential signal (d_z)
Element difference	Z	G	$Z-G$
Field difference	Z	$\frac{E+J}{2}$	$Z-\frac{E+J}{2}$
Frame difference	Z	T	$Z-T$
Element difference of frame difference	Z	$T+G-S$	$(Z-G)-(T-S)$
Line difference of frame difference	Z	$T+B-M$	$(Z-B)-(T-M)$
Element difference of field difference	Z	$T+\frac{E+J}{2}-\frac{Q+W}{2}$	$\left(Z-\frac{E+J}{2}\right)-\left(T-\frac{Q+W}{2}\right)$

3.5.3 Motion-Compensated Predictive Coding

When frames are taken densely enough, changes in successive frames can be attributed to the motion of objects during the interval between frames. Under this assumption, if we can analyze object motion from successive frames, then we should be able to predict objects in the next frame based on their positions in the previous frame and the estimated motion. The difference between the original frame and the predicted frame thus generated and the motion vectors are then quantized and coded. If the motion estimation is accurate enough, the motion-compensated prediction error can be smaller than 3-D DPCM. In other words, the variance of the prediction error will be smaller, resulting in more efficient coding. Take motion into consideration — this differential technique is called motion compensated predictive coding. This has been a major development in image sequence coding since the 1980s. It has been adopted by all international video coding standards. A more detailed discussion is provided in Chapter 10.

3.6 INFORMATION-PRESERVING DIFFERENTIAL CODING

As emphasized in Chapter 2, quantization is not reversible in the sense that it causes permanent information loss. The DPCM technique, discussed above, includes quantization, and hence is lossy coding. In applications such as those involving scientific measurements, information preservation is required. In this section, the following question is addressed: under these circumstances, how should we apply differential coding in order to reduce the bit rate while preserving information?

Figure 3.14 shows a block diagram of information-preserving differential coding. First, we see that there is no quantizer. Therefore, the irreversible information loss associated with quantization does not exist in this technique. Second, we observe that prediction and differencing are still used. That is, the differential (predictive) technique still applies. Hence it is expected that the variance of the difference signal is smaller than that of the original signal, as explained in Section 3.1. Consequently, the higher-peaked histograms make coding more efficient. Third, an efficient lossless coder is utilized. Since quantizers cannot be used here, PCM with natural binary coding is not used here. Since the histogram of the difference signal is narrowly concentrated about its mean, lossless coding techniques such as an efficient Huffman coder (discussed in Chapter 5) is naturally a suitable choice here.

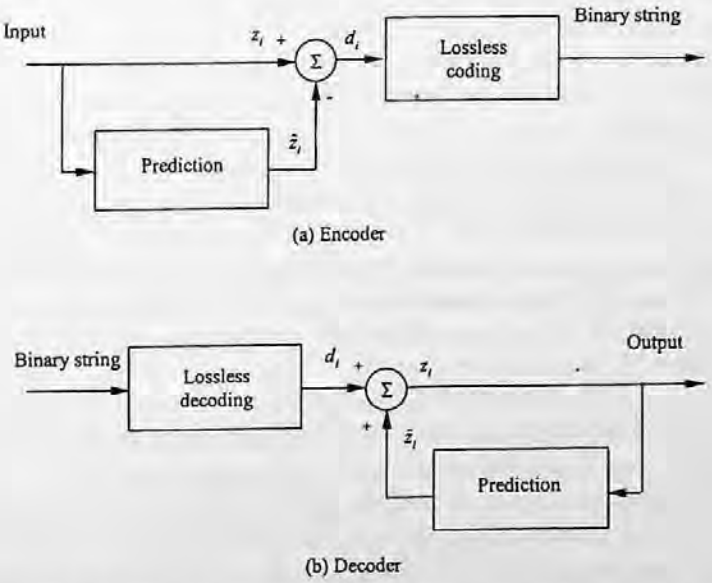


FIGURE 3.14 Block diagram of information-preserving differential coding.

As mentioned before, input images are normally in a PCM coded format with a bit rate of eight bits per pixel for monochrome pictures. The difference signal is therefore integer-valued. Having no quantization and using an efficient lossless coder, the coding system depicted in Figure 3.14, therefore, is an information-preserving differential coding technique.

3.7 SUMMARY

Rather than coding the signal itself, differential coding, also known as predictive coding, encodes the difference between the signal and its prediction. Utilizing spatial and/or temporal correlation between pixels in the prediction, the variance of the difference signal can be much smaller than that of the original signal, thus making differential coding quite efficient.

Among differential coding methods, differential pulse code modulation (DPCM) is used most widely. In DPCM coding, the difference signal is quantized and codewords are assigned to the quantized difference. Prediction and quantization are therefore two major components in the DPCM systems. Since quantization was addressed in Chapter 2, this chapter emphasizes prediction. The theory of optimum linear prediction is introduced. Here, optimum means minimization of the mean square prediction error. The formulation of optimum linear prediction, the orthogonality condition, and the minimum mean square prediction error are presented. The orthogonality condition states that the prediction error must be orthogonal to each observation, i.e., to the reconstructed sample intensity values used in the linear prediction. By solving the Yule-Walker equation, the optimum prediction coefficients may be determined.

In addition, some fundamental issues in implementing the DPCM technique are discussed. One issue is the dimensionality of the predictor in DPCM. We discussed 1-D, 2-D, and 3-D predictors. DPCM with a 2-D predictor demonstrates better performance than a 1-D predictor since 2-D DPCM utilizes more spatial correlation, i.e., not only horizontally but also vertically. As a result, a 3-dB improvement in *SNR* was reported. 3-D prediction is encountered in what is known as interframe coding. There, temporal correlation exists. 3-D DPCM utilizes both spatial and temporal correlation between neighboring pixels in successive frames. Consequently, more redundancy can be removed. Motion-compensated predictive coding is a very powerful technique in video coding related to differential coding. It uses a more advanced translational motion model in the prediction, however, and it is covered in Sections III and IV.

Another issue is the order of predictors and its effect on the performance of prediction in terms of mean square prediction error. Increasing the prediction order can lower the mean square prediction error effectively, but the performance improvement becomes insignificant after the third order.

Adaptive prediction is another issue. Similar to adaptive quantization, discussed in Chapter 2, we can adapt the prediction coefficients in the linear predictor to varying local statistics.

The last issue is concerned with the effect of transmission error. Bit reversal in transmission causes a different effect on reconstructed images depending on the type of coding technique used. PCM is known to be bit-consuming. (An acceptable PCM representation of monochrome images requires six to eight bits per pixel.) But a one-bit reversal only affects an individual pixel. For the DPCM coding technique, however, a transmission error may propagate from one pixel to the other. In particular, DPCM with a 1-D predictor suffers from error propagation more severely than DPCM with a 2-D predictor.

Delta modulation is an important special case of DPCM in which the predictor is of the first order. Specifically, the immediately preceding coded sample is used as a prediction of the present input sample. Furthermore, the quantizer has only two reconstruction levels.

Finally, an information-preserving differential coding technique is discussed. As mentioned in Chapter 2, quantization is an irreversible process: it causes information loss. In order to preserve information, there is no quantizer in this type of system. To be efficient, lossless codes such as Huffman code or arithmetic code should be used for difference signal encoding.

3.8 EXERCISES

3-1. Justify the necessity of the closed-loop DPCM with feedback around quantizers. That is, convince yourself why the quantization error will be accumulated if, instead of using the reconstructed preceding samples, we use the immediately preceding sample as the prediction of the sample being coded in DPCM.

3-2. Why does the overload error encountered in quantization appear to be the slope overload in DM?

3-3. What advantage does oversampling bring up in the DM technique?

3-4. What are the two features of DM that make it a subclass of DPCM?

3-5. Explain why DPCM with a 1-D predictor suffers from bit reversal transmission error more severely than DPCM with a 2-D predictor.

3-6. Explain why no quantizer can be used in information-preserving differential coding, and why the differential system can work without a quantizer.

3-7. Why do all the pixels involved in prediction of differential coding have to be in a recursively computable order from the point of view of the pixel being coded?

3-8. Discuss the similarity and dissimilarity between DPCM and motion compensated predictive coding.

REFERENCES

Bose, N. K. *Applied Multidimensional System Theory*, Van Nostrand Reinhold, New York, 1982.

Bruders, R., T. Kummerow, P. Neuhold, and P. Stamnitz, Ein versuchssystem zur digitalen ubertragung von fernsehsignalen unter besonderer berucksichtigung von ubertragungsfehlern, Festschrift 50 Jahre Heinrich-Hertz-Institut, Berlin, 1978.

Connor, D. J. *IEEE Trans. Commun.*, com-21, 695-706, 1973.

Cutler, C. C. U. S. Patent 2,605,361, 1952.

DeJager, F. *Philips Res. Rep.*, 7, 442-466, 1952.

Elias, P. *IRE Trans. Inf. Theory*, it-1, 16-32, 1955.

Habibi, A. Comparison of nth-order DPCM encoder with linear transformations and block quantization techniques, *IEEE Trans. Commun. Technol.*, COM-19(6), 948-956, 1971.

Harrison, C. W. *Bell Syst. Tech. J.*, 31, 764-783, 1952.

Haskell, B. G., F. W. Mounts, and J. C. Candy, Interframe coding of videotelephone pictures, *Proc. IEEE*, 60, 7, 792-800, 1972.

Haskell, B. G. Frame replenishment coding of television, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Upper Saddle River, NJ, 1984.

Kretzmer, E. R. Statistics of television signals, *Bell Syst. Tech. J.*, 31, 751-763, 1952.

Leon-Garcia, A. *Probability and Random Processes for Electrical Engineering*, 2nd ed., Addison-Wesley, Reading, MA, 1994.

Lim, J. S. *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Mounts, F. W. A video encoding system with conditional picture-element replenishment, *Bell Syst. Tech. J.*, 48, 7, 1969.

Musmann, H. G. Predictive Image Coding, in *Image Transmission Techniques*, W. K. Pratt (Ed.), Academic Press, New York, 1979.

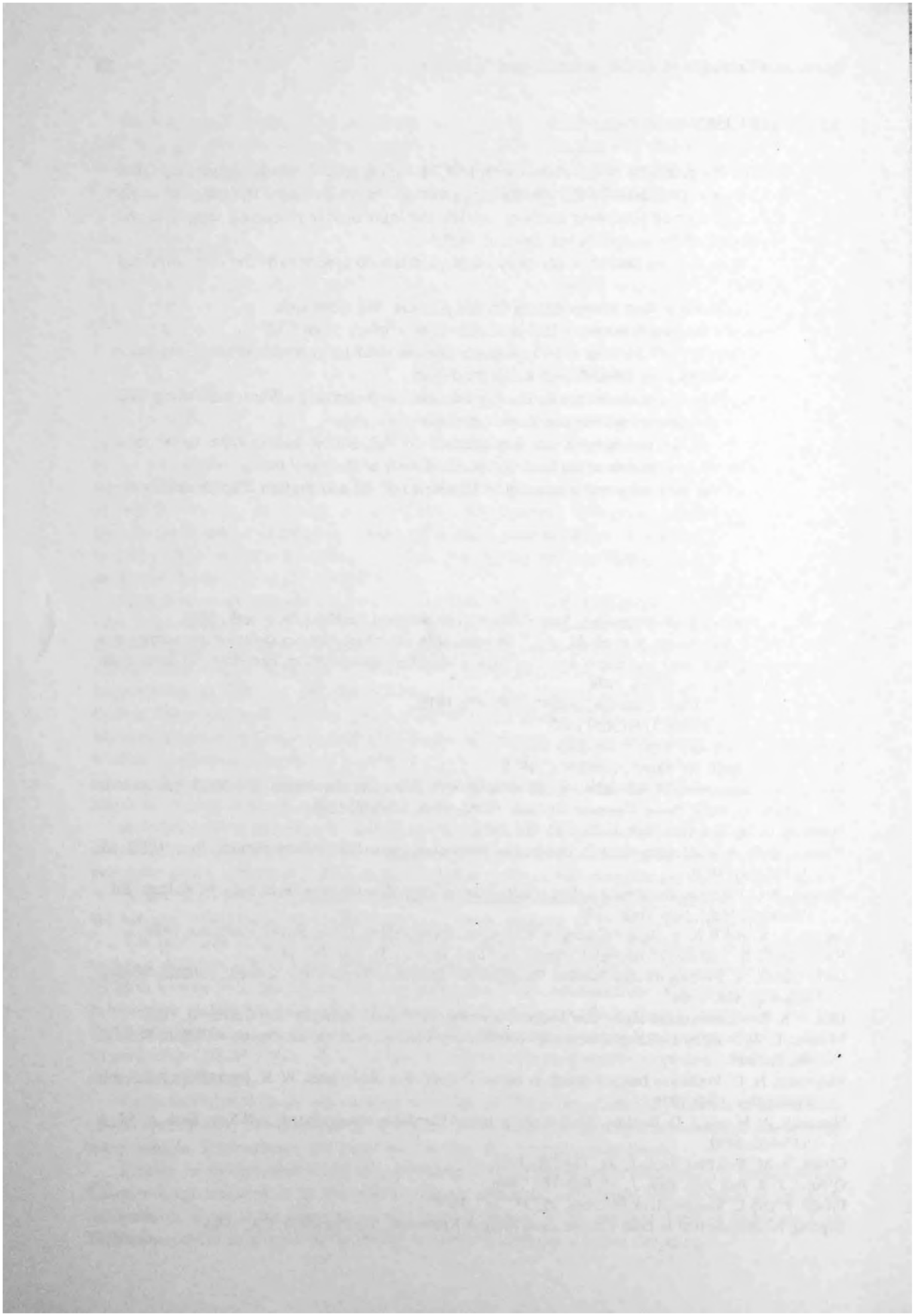
Netravali, A. N. and J. D. Robbins, Motion-compensated television coding. Part I, *Bell Syst. Tech. J.*, 58, 3, 631-670, 1979.

Oliver, B. M. *Bell Syst. Tech. J.*, 31, 724-750, 1952.

O'Neal, J. B. *Bell Syst. Tech. J.*, 45, 689-721, 1966.

Pirsch, P. and L. Stenger, *Acta Electron.*, 19, 277-287, 1977.

Sayood, K. *Introduction to Data Compression*, Morgan Kaufmann, San Francisco, CA, 1996.



4 Transform Coding

As introduced in the previous chapter, differential coding achieves high coding efficiency by utilizing the correlation between pixels existing in image frames. Transform coding (TC), the focus of this chapter, is another efficient coding scheme based on utilization of interpixel correlation. As we will see in Chapter 7, TC has become a fundamental technique recommended by the international still image coding standard, JPEG. Moreover, TC has been found to be efficient in coding prediction error in motion-compensated predictive coding. As a result, TC was also adopted by the international video coding standards such as H.261, H.263, and MPEG 1, 2, and 4. This will be discussed in Section IV.

4.1 INTRODUCTION

Recall the block diagram of source encoders shown in Figure 2.3. There are three components in a source encoder: transformation, quantization, and codeword assignment. It is the transformation component that decides which format of input source is quantized and encoded. In DPCM, for instance, the difference between an original signal and a predicted version of the original signal is quantized and encoded. As long as the prediction error is small enough, i.e., the prediction resembles the original signal well (by using correlation between pixels), differential coding is efficient.

In transform coding, the main idea is that if the transformed version of a signal is less correlated compared with the original signal, then quantizing and encoding the transformed signal may lead to data compression. At the receiver, the encoded data are decoded and transformed back to reconstruct the signal. Therefore, in transform coding, the transformation component illustrated in Figure 2.3 is a transform. Quantization and codeword assignment are carried out with respect to the transformed signal, or, in other words, carried out in the transform domain.

We begin with the Hotelling transform, using it as an example of how a transform may decorrelate a signal in the transform domain.

4.1.1 Hotelling Transform

Consider an N-dimensional vector $\vec{z}_s$. The ensemble of such vectors, $\{\vec{z}_s\}$ $s \in I$, where I represents the set of all vector indexes, can be modeled by a random vector $\bar{z}$ with each of its component z_i $i = 1, 2, \cdots, N$ as a random variable. That is,

$$\bar{z} = (z_1, z_2, \cdots, z_N)^T \tag{4.1}$$

where T stands for the operator of matrix transposition. The mean vector of the population, $m_{\bar{z}}$, is defined as

$$m_{\bar{z}} = E[\bar{z}] = (m_1, m_2, \cdots, m_N)^T \tag{4.2}$$

where E stands for the expectation operator. Note that $m_{\bar{z}}$ is an N-dimensional vector with the ith component, m_i, being the expectation value of the ith random variable component in $\bar{z}$.

$$m_i = E[z_i] \quad i = 1, 2, \cdots, N \tag{4.3}$$

The covariance matrix of the population, denoted by $C_{\vec{z}}$, is equal to

$$C_{\vec{z}} = E\left[\left(\vec{z} - m_{\vec{z}}\right)\left(\vec{z} - m_{\vec{z}}\right)^T\right]. \tag{4.4}$$

Note that the product inside the E operator is referred to as the *outer product* of the vector $(\vec{z} - m_{\vec{z}})$. Denote an entry at the ith row and jth column in the covariance matrix by $c_{i,j}$. From Equation 4.4, it can be seen that $c_{i,j}$ is the covariance between the ith and jth components of the random vector $\vec{z}$. That is,

$$c_{i,j} = E\left[\left(z_i - m_i\right)\left(z_j - m_j\right)\right] = Cov\left(z_i, z_j\right). \tag{4.5}$$

On the main diagonal of the covariance matrix $C_{\vec{z}}$, the element $c_{i,i}$ is the variance of the ith component of $\vec{z}$, z_i. Obviously, the covariance matrix $C_{\vec{z}}$ is a real and symmetric matrix. It is real because of the definition of random variables. It is symmetric because $Cov(z_i, z_j) = Cov(z_j, z_i)$. According to the theory of linear algebra, it is always possible to find a set of N orthonormal eigenvectors of the matrix $C_{\vec{z}}$, with which we can convert the real symmetric matrix $C_{\vec{z}}$ into a fully ranked diagonal matrix. This statement can be found in texts of linear algebra, e.g., in (Strang, 1998).

Denote the set of N orthonormal eigenvectors and their corresponding eigenvalues of the covariance matrix $C_{\vec{z}}$ by $\vec{e}_i$ and λ_i, $i = 1,2,\cdots,N$, respectively. Note that eigenvectors are column vectors. Form a matrix Φ such that its rows comprise the N transposed eigenvectors. That is,

$$\Phi = \left(\vec{e}_1, \vec{e}_2, \cdots, \vec{e}_N\right)^T. \tag{4.6}$$

Now, consider the following transformation:

$$\vec{y} = \Phi\left(\vec{z} - m_{\vec{z}}\right) \tag{4.7}$$

It is easy to verify that the transformed random vector $\vec{y}$ has the following two characteristics:

1. The mean vector, $m_{\vec{y}}$, is a zero vector. That is,

$$m_{\vec{y}} = 0. \tag{4.8}$$

2. The covariance matrix of the transformed random vector $C_{\vec{y}}$ is

$$C_{\vec{y}} = \Phi C_{\vec{z}} \Phi^T = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}. \tag{4.9}$$

This transform is called the Hotelling transform (Hotelling, 1933), or eigenvector transform (Tasto, 1971; Wintz, 1972).

The inverse Hotelling transform is defined as

$$\vec{z} = \Phi^{-1}\vec{y} + m_{\vec{z}}, \tag{4.10}$$

where Φ^{-1} is the inverse matrix of Φ. It is easy to see from its formation discussed above that the matrix Φ is orthogonal. Therefore, we have $\Phi^T = \Phi^{-1}$. Hence, the inverse Hotelling transform can be expressed as

$$\vec{z} = \Phi^T \vec{y} + m_{\vec{z}}. \tag{4.11}$$

Note that in implementing the Hotelling transform, the mean vector $m_{\vec{z}}$ and the covariance matrix $C_{\vec{z}}$ can be calculated approximately by using a given set of K sample vectors (Gonzalez and Woods, 1992).

$$m_{\vec{z}} = \frac{1}{K}\sum_{s=1}^{K} \vec{z}_s \tag{4.12}$$

$$C_{\vec{z}} = \frac{1}{K}\sum_{s=1}^{K} \vec{z}_s \vec{z}_s^T - m_{\vec{z}} m_{\vec{z}}^T \tag{4.13}$$

The analogous transform for continuous data was devised by Karhunen and Loeve (Karhunen, 1947; Loeve, 1948). Alternatively, the Hotelling transform can be viewed as the discrete version of the Karhunen-Loeve transform (KLT). We observe that the covariance matrix $C_{\vec{y}}$ is a diagonal matrix. The elements in the diagonal are the eigenvalues of the covariance matrix $C_{\vec{z}}$. That is, the two covariance matrices have the same eigenvalues and eigenvectors because the two matrices are similar. The fact that zero values are everywhere except along the main diagonal in $C_{\vec{y}}$ indicates that the components of the transformed vector $\vec{y}$ are uncorrelated. That is, the correlation previously existing between the different components of the random vector $\vec{z}$ has been removed in the transformed domain. Therefore, if the input is split into *blocks* and the Hotelling transform is applied blockwise, the coding may be more efficient since the data in the transformed block are uncorrelated. At the receiver, we may produce a replica of the input with an inverse transform. This basic idea behind transform coding will be further illustrated next. Note that transform coding is also referred to as block quantization (Huang, 1963).

4.1.2 Statistical Interpretation

Let's continue our discussion of the 1-D Hotelling transform. Recall that the covariance matrix of the transformed vector $\vec{y}$, $C_{\vec{y}}$, is a diagonal matrix. The elements in the main diagonal are eigenvalues of the covariance matrix $C_{\vec{y}}$. According to the definition of a covariance matrix, these elements are the variances of the components of vector $\vec{y}$, denoted by $\sigma_{y,1}^2, \sigma_{y,2}^2, \cdots, \sigma_{y,N}^2$. Let us arrange the eigenvalues (variances) in a nonincreasing order. That is, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. Choose an integer L, and $L < N$. Using the corresponding L eigenvectors, $\vec{e}_1, \vec{e}_2, \cdots, \vec{e}_L$, we form a matrix $\overline{\Phi}$ with these L eigenvectors (transposed) as its L rows. Obviously, the matrix $\overline{\Phi}$ is of $L \times N$. Hence, using the matrix $\overline{\Phi}$ in Equation 4.7 we will have the transformed vector $\vec{y}$ of $L \times 1$. That is,

$$\vec{y} = \overline{\Phi}\left(\vec{z} - m_{\vec{z}}\right). \tag{4.14}$$

The inverse transform changes accordingly:

$$\vec{z}' = \overline{\Phi}^T \vec{y} + m_{\vec{z}}. \tag{4.15}$$

Note that the reconstructed vector $\vec{z}$, denoted by $\vec{z}'$, is still an $N \times 1$ column vector. It can be shown (Wintz, 1972) that the mean square reconstruction error between the original vector $\vec{z}$ and the reconstructed vector $\vec{z}$ is given by

$$MSE_r = \sum_{i=L+1}^{N} \sigma_{y,i}^2. \tag{4.16}$$

This equation indicates that the mean square reconstruction error equals the sum of variances of the discarded components. Note that although we discuss the reconstruction error here, we have not considered the quantization error and transmission error involved. Equation 4.15 implies that if, in the transformed vector $\vec{y}$, the first L components have their variances occupy a large percentage of the total variances, the mean square reconstruction error will not be large even though only the first L components are kept, i.e., the $(N - L)$ remaining components in the $\vec{y}$ are discarded. Quantizing and encoding only L components of vector $\vec{y}$ in the transform domain lead to higher coding efficiency. This is the basic idea behind transform coding.

4.1.3 Geometrical Interpretation

Transforming a set of statistically dependent data into another set of uncorrelated data, then discarding the insignificant transform coefficients (having small variances) illustrated above using the Hotelling transform, can be viewed as a statistical interpretation of transform coding. Here, we give a geometrical interpretation of transform coding. For this purpose, we use 2-D vectors instead of N-D vectors.

Consider a binary image of a car in Figure 4.1(a). Each pixel in the shaded object region corresponds to a 2-D vector with its two components being coordinates z_1 and z_2, respectively. Hence, the set of all pixels associated with the object forms a population of vectors. We can determine its mean vector and covariance matrix using Equations 4.12 and 4.13, respectively. We can then apply the Hotelling transform by using Equation 4.7. Figure 4.1(b) depicts the same object after the application of the Hotelling transform in the y_1-y_2 coordinate system. We notice that the origin of the new coordinate system is now located at the centroid of the binary object. Furthermore, the new coordinate system is aligned with the two eigenvectors of the covariance matrix $C_{\vec{z}}$.

As mentioned, the elements along the main diagonal of $C_{\vec{y}}$ (two eigenvalues of the $C_{\vec{y}}$ and $C_{\vec{z}}$) are the two variances of the two components of the $\vec{y}$ population. Since the covariance matrix

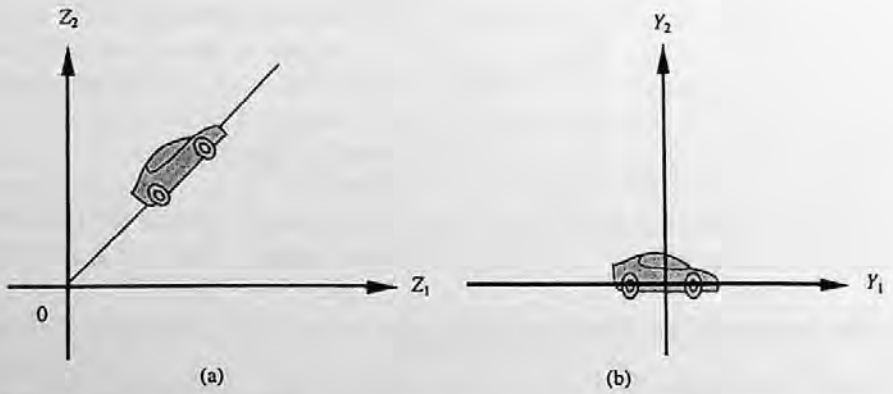


FIGURE 4.1 (a) A binary object in the z_1-z_2 coordinate system. (b) After the Hotelling transform, the object is aligned with its principal axes.

$C_{\vec{y}}$ is a diagonal matrix, the two components are uncorrelated after the transform. Since one variance (along the y_1 direction) is larger than the other (along the y_2 direction), it is possible for us to achieve higher coding efficiency by ignoring the component associated with the smaller variance without too much sacrifice of the reconstructed image quality.

It is noted that the alignment of the object with the eigenvectors of the covariance matrix is of importance in pattern recognition (Gonzalez and Woods, 1992).

4.1.4 Basis Vector Interpretation

Basis vector expansion is another interpretation of transform coding. For simplicity, in this subsection we assume a zero mean vector. Under this assumption, the Hotelling transform and its inverse transform become

$$\vec{y} = \Phi\vec{z} \tag{4.17}$$

$$\vec{z} = \Phi^T\vec{y} \tag{4.18}$$

Recall that the row vectors in the matrix Φ are the transposed eigenvectors of the covariance matrix $C_{\vec{z}}$. Therefore, Equation 4.18 can be written as

$$\vec{z} = \sum_{i=1}^{N} y_i\vec{e}_i. \tag{4.19}$$

In the above equation, we can view vector $\vec{z}$ as a linear combination of *basis vectors* $\vec{e}_i$, $i = 1,2,\cdots,N$. The components of the transformed vector $\vec{y}$, y_i, $i = 1,2,\cdots,N$ serve as coefficients in the linear combination, or weights in the weighted sum of basis vectors. The coefficient y_i, $i = 1,2,\cdots,N$ can be produced according to Equation 4.17:

$$y_i = \vec{e}_i^{\,T}\vec{z}. \tag{4.20}$$

That is, y_i is the *inner product* between vectors $\vec{e}_i$ and $\vec{z}$. Therefore, the coefficient y_i can be interpreted as the amount of correlation between the basis vector $\vec{e}_i$ and the original signal $\vec{z}$.

In the Hotelling transform the coefficients y_i, $i = 1,2,\cdots,N$ are uncorrelated. The variance of y_i can be arranged in a nonincreasing order. For $i > L$, the variance of the coefficient becomes insignificant. We can then discard these coefficients without introducing significant error in the linear combination of basis vectors and achieve higher coding efficiency.

In the above three interpretations of transform coding, we see that the linear unitary transform can provide the following two functions:

1. Decorrelate input data; i.e., transform coefficients are less correlated than the original data, and
2. Have some transform coefficients more significant than others (with large variance, eigenvalue, or weight in basis vector expansion) such that transform coefficients can be treated differently: some can be discarded, some can be coarsely quantized, and some can be finely quantized.

Note that the definition of *unitary* transform is given shortly in Section 4.2.1.3.

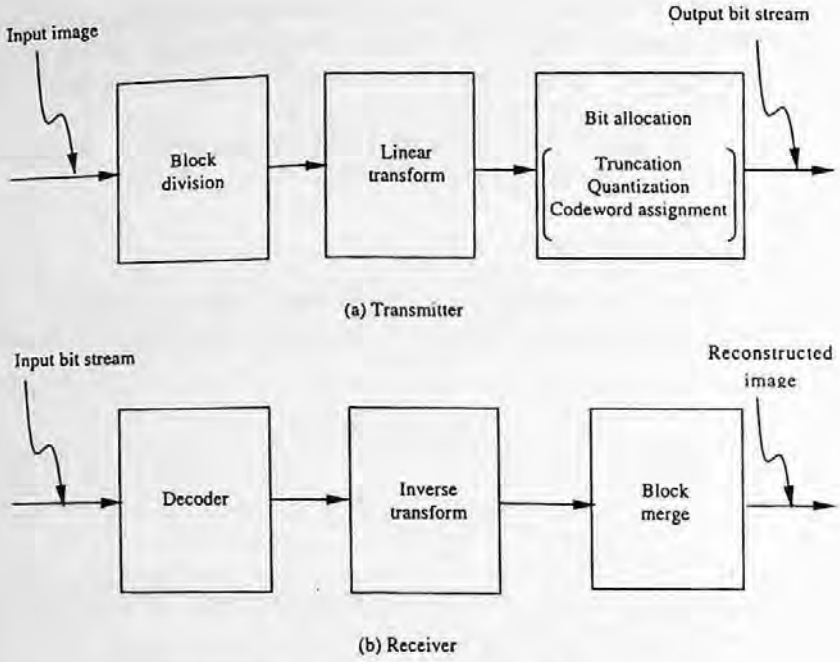


FIGURE 4.2 Block diagram of transform coding.

4.1.5 Procedures of Transform Coding

Prior to leaving this section, we summarize the procedures of transform coding. There are three steps in transform coding as shown in Figure 4.2. First, the input data (frame) are divided into blocks (subimages). Each block is then linearly transformed. The transformed version is then truncated, quantized, and encoded. These last three functions, which are discussed in Section 4.4, can be grouped and termed as bit allocation. The output of the encoder is a bitstream.

In the receiver, the bitstream is decoded and then inversely transformed to form reconstructed blocks. All the reconstructed blocks collectively produce a replica of the input image.

4.2 LINEAR TRANSFORMS

In this section, we first discuss a general formulation of a linear unitary 2-D image transform. Then, a basis image interpretation of TC is given.

4.2.1 2-D Image Transformation Kernel

There are two different ways to handle image transformation. In the first way, we convert a 2-D array representing a digital image into a 1-D array via row-by-row stacking, for example. That is, from the second row on, the beginning of each row in the 2-D array is cascaded to the end of its previous row. Then we transform this 1-D array using a 1-D transform. After the transformation, we can convert the 1-D array back to a 2-D array. In the second way, a 2-D transform is directly applied to the 2-D array corresponding to an input image, resulting in a transformed 2-D array. These two ways are essentially the same. It can be straightforwardly shown that the difference between the two is simply a matter of notation (Wintz, 1972). In this section, we use the second way to handle image transformation. That is, we work on 2-D image transformation.

Assume a digital image is represented by a 2-D array $g(x, y)$, where (x, y) is the coordinates of a pixel in the 2-D array, while g is the gray level value (also often called intensity or brightness) of the pixel. Denote the 2-D transform of $g(x, y)$ by $T(u, v)$, where (u, v) is the coordinates in the transformed domain. Assume that both $g(x, y)$ and $T(u, v)$ are a square 2-D array of $N \times N$; i.e., $0 \le x, y, u, v \le N - 1$.

The 2-D forward and inverse transforms are defined as

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x,y) f(x,y,u,v) \tag{4.21}$$

and

$$g(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) i(x,y,u,v) \tag{4.22}$$

where $f(x, y, u, v)$ and $i(x, y, u, v)$ are referred to as the forward and inverse *transformation kernels*, respectively.

A few characteristics of transforms are discussed below.

4.2.1.1 Separability

A transformation kernel is called separable (hence, the transform is said to be separable) if the following conditions are satisfied.

$$f(x,y,u,v) = f_1(x,u) f_2(y,v), \tag{4.23}$$

and

$$i(x,y,u,v) = i_1(x,u) i_2(y,v). \tag{4.24}$$

Note that a 2-D separable transform can be decomposed into two 1-D transforms. That is, a 2-D transform can be implemented by a 1-D transform rowwise followed by another 1-D transform columnwise. That is,

$$T_1(x,v) = \sum_{y=0}^{N-1} g(x,y) f_2(y,v), \tag{4.25}$$

where $0 \le x, v \le N - 1$, and

$$T(u,v) = \sum_{x=0}^{N-1} T_1(x,v) f_1(x,u), \tag{4.26}$$

where $0 \le u, v \le N - 1$. Of course, the 2-D transform can also be implemented in a reverse order with two 1-D transforms, i.e., columnwise first, followed by rowwise. The counterparts of Equations 4.25 and 4.26 for the inverse transform can be derived similarly.

4.2.1.2 Symmetry

The transformation kernel is symmetric (hence, the transform is symmetric) if the kernel is separable and the following condition is satisfied:

$$f_1(y,v) = f_2(y,v). \tag{4.27}$$

That is, f_1 is functionally equivalent to f_2.

4.2.1.3 Matrix Form

If a transformation kernel is symmetric (hence, separable) then the 2-D image transform discussed above can be expressed compactly in the following matrix form. Denote an *image matrix* by G and $G = \{g_{i,j}\} = \{g(i-1, j-1)\}$. That is, a typical element (at the ith row and jth column) in the matrix G is the pixel gray level value in the 2-D array $g(x, y)$ at the same geometrical position. Note that the subtraction of one in the notation $g(i-1, j-1)$ comes from Equations 4.21 and 4.22. Namely, the indexes of a square 2-D image array are conventionally defined from 0 to N-1, while the indexes of a square matrix are from 1 to N. Denote the *forward transform matrix* by F and $F = \{f_{i,j}\} = \{f_1(i-1, j-1)\}$. We then have the following matrix form of a 2-D transform:

$$T = F^T GF \tag{4.28}$$

where T on the left-hand side of the equation denotes the matrix corresponding to the transformed 2-D array in the same fashion as that used in defining the G matrix. The inverse transform can be expressed as

$$G = I^T TI \tag{4.29}$$

where the matrix I is the *inverse transform matrix* and $I = \{i_{j,k}\} = \{i_1(j-1, k-1)\}$. The forward and inverse transform matrices have the following relation:

$$I = F^{-1} \tag{4.30}$$

Note that all of the matrices defined above, G, T, F, and I are of $N \times N$.

It is known that the discrete Fourier transform involves complex quantities. In this case, the counterparts of Equations 4.28, 4.29, and 4.30 become Equations 4.31, 4.32, and 4.33, respectively:

$$T = F^{*T} GF \tag{4.31}$$

$$G = I^{*T} TI \tag{4.32}$$

$$I = F^{-1} = F^{*T} \tag{4.33}$$

where * indicates complex conjugation. Note that the transform matrices F and I contain complex quantities and satisfy Equation 4.33. They are called unitary matrices and the transform is referred to as a unitary transform.

4.2.1.4 Orthogonality

A transform is said to be orthogonal if the transform matrix is orthogonal. That is,

$$F^T = F^{-1} \tag{4.34}$$

Note that an orthogonal matrix (orthogonal transform) is a special case of a unitary matrix (unitary transform), where only real quantities are involved. We will see that all the 2-D image transforms, presented in Section 4.3, are separable, symmetric, and unitary.

4.2.2 Basis Image Interpretation

Here we study the concept of *basis images* or *basis matrices*. Recall that we discussed basis vectors when we considered the 1-D transform. That is, the components of the transformed vector (also referred to as the transform coefficients) can be interpreted as the coefficients in the basis vector expansion of the input vector. Each coefficient is essentially the amount of correlation between the input vector and the corresponding basis vector. The concept of basis vectors can be extended to basis images in the context of 2-D image transforms.

Recall that the 2-D inverse transform introduced at the beginning of this section is defined as

$$g(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)i(x,y,u,v) \tag{4.35}$$

where $0 \le x, y \le N-1$. This equation can be viewed as a *component* form of the inverse transform. As defined above in Section 4.2.1.3, the whole image $\{g(x,y)\}$ is denoted by the image matrix G of $N \times N$. We now denote the "image" formed by the inverse transformation kernel $\{i(x,y,u,v), 0 \le x, y \le N-1\}$ as a 2-D array $I_{u,v}$ of $N \times N$ for a specific pair of (u,v) with $0 \le u, v \le N-1$. Recall that a digital image can be represented by a 2-D array of gray level values. In turn the 2-D array can be arranged into a matrix. Namely, we treat the following three: a digital image, a 2-D array (with proper resolution), and a matrix (with proper indexing), interchangeably. We then have

$$I_{u,v} = \begin{bmatrix} i(0,0,u,v) & i(0,1,u,v) & \cdots & \cdots & i(0,N-1,u,v) \\ i(1,0,u,v) & i(1,1,u,v) & \cdots & \cdots & i(1,N-1,u,v) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ i(N-1,0,u,v) & i(N-1,1,u,v) & \cdots & \cdots & i(N-1,N-1,u,v) \end{bmatrix} \tag{4.36}$$

The 2-D array $I_{u,v}$ is referred to as a basis image. There are N^2 basis images in total since $0 \le u,v \le N-1$. The inverse transform expressed in Equation 4.35 can then be written in a *collective* form as

$$G = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)I_{u,v}. \tag{4.37}$$

We can interpret this equation as a series expansion of the original image G into a set of N^2 basis images $I_{u,v}$. The transform coefficients $T(u,v)$, $0 \le u, v \le N-1$, become the coefficients of the expansion. Alternatively, the image G is said to be a weighted sum of basis images. Note that,

similar to the 1-D case, the coefficient or the weight $T(u,v)$ is a correlation measure between the image G and the basis image $I_{u,v}$ (Wintz, 1972).

Note that basis images have nothing to do with the input image. Instead, it is completely defined by the transform itself. That is, basis images are the attribute of 2-D image transforms. Different transforms have different sets of basis images.

The motivation behind transform coding is that with a proper transform, hence, a proper set of basis images, the transform coefficients are more independent than the gray scales of the original input image. In the ideal case, the transform coefficients are statistically independent. We can then optimally encode the coefficients independently, which can make coding more efficient and simple. As pointed out in (Wintz, 1972), however, this is generally impossible because of the following two reasons. First, it requires the joint probability density function of the N^2 pixels, which have not been deduced from basic physical laws and cannot be measured. Second, even if the joint probability density functions were known, the problem of devising a reversible transform that can generate independent coefficients is unsolved. The optimum linear transform we can have results in uncorrelated coefficients. When Gaussian distribution is involved, we can have independent transform coefficients. In addition to the uncorrelatedness of coefficients, the variance of the coefficients varies widely. Insignificant coefficients can be ignored without introducing significant distortion in the reconstructed image. Significant coefficients can be allocated more bits in encoding. The coding efficiency is thus enhanced.

As shown in Figure 4.3, TC can be viewed as expanding the input image into a set of basis images, then quantizing and encoding the coefficients associated with the basis images separately. At the receiver the coefficients are reconstructed to produce a replica of the input image. This strategy is similar to that of subband coding, which is discussed in Chapter 8. From this point of view, transform coding can be considered a special case of subband coding, though transform coding was devised much earlier than subband coding.

It is worth mentioning an alternative way to define basis images. That is, a basis image with indexes (u, v), $I_{u,v}$, of a transform can be constructed as the *outer product* of the uth basis vector, $\vec{b}_u$, and the vth basis vector, $\vec{b}_v$, of the transform. The basis vector, $\vec{b}_u$, is the uth column vector of the inverse transform matrix I (Jayant and Noll, 1984). That is,

$$I_{u,v} = \vec{b}_u \vec{b}_v^T. \tag{4.38}$$

4.2.3 Subimage Size Selection

The selection of subimage (block) size, N, is important. Normally, the larger the size the more decorrelation the transform coding can achieve. It has been shown, however, that the correlation between image pixels becomes insignificant when the distance between pixels becomes large, e.g., it exceeds 20 pixels (Habibi, 1971a). On the other hand, a large size causes some problems. In adaptive transform coding, a large block cannot adapt to local statistics well. As will be seen later in this chapter, a transmission error in transform coding affects the whole associated subimage. Hence a large size implies a possibly severe effect of transmission error on reconstructed images. As will be shown in video coding (Section III and Section IV), transform coding is used together with motion-compensated coding. Consider that large block size is not used in motion estimation; subimage sizes of 4, 8, and 16 are used most often. In particular, $N = 8$ is adopted by the international still image coding standard, JPEG, as well as video coding standards H.261, H.263, MPEG 1, and MPEG 2.

4.3 TRANSFORMS OF PARTICULAR INTEREST

Several commonly used image transforms are discussed in this section. They include the discrete Fourier transform, the discrete Walsh transform, the discrete Hadamard transform, and the discrete Cosine and Sine transforms. All of these transforms are symmetric (hence, separable as well),

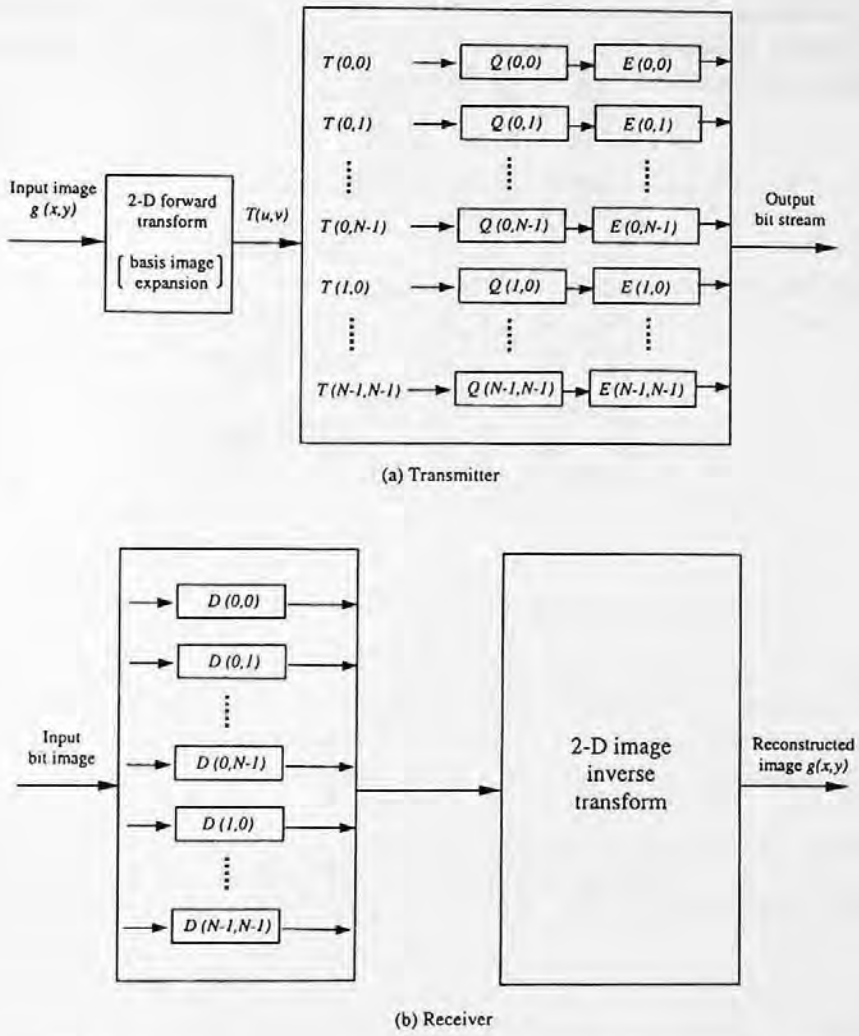


FIGURE 4.3 Basis image interpretation of TC (Q: quantizer, E: encoder, D: decoder).

unitary, and reversible. For each transform, we define its transformation kernel and discuss its basis images.

4.3.1 Discrete Fourier Transform (DFT)

The DFT is of great importance in the field of digital signal processing. Owing to the fast Fourier transform (FFT) based on the algorithm developed in (Cooley, 1965), the DFT is widely utilized for various tasks of digital signal processing. It has been discussed in many signal and image processing texts. Here we only define it by using the transformation kernel just introduced above. The forward and inverse transformation kernels of the DFT are

$$f(x,y,u,v)=\frac{1}{N}\exp\{-j2\pi(xu+yv)/N\} \tag{4.39}$$

and

$$i(x,y,u,v)=\frac{1}{N}\exp\{j2\pi(xu+yv)/N\} \tag{4.40}$$

Clearly, since complex quantities are involved in the DFT transformation kernels, the DFT is generally complex. Hence, we use the unitary matrix to handle the DFT (refer to Section 4.2.1.3). The basis vector of the DFT $\vec{b}_u$ is an $N \times 1$ column vector and is defined as

$$\vec{b}_u = \frac{1}{\sqrt{N}}\left[1, \exp\left(j2\pi\frac{u}{N}\right), \exp\left(j2\pi\frac{2u}{N}\right), \cdots, \exp\left(j2\pi\left(\frac{(N-1)u}{N}\right)\right)\right]^T \tag{4.41}$$

As mentioned, the basis image with index (u,v), $I_{u,v}$, is equal to $\vec{b}_u \vec{b}_v^T$. A few basis images are listed below for $N = 4$.

$$I_{0,0} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.42}$$

$$I_{0,1} = \frac{1}{4}\begin{pmatrix} 1 & j & -1 & -j \\ 1 & j & -1 & -j \\ 1 & j & -1 & -j \\ 1 & j & -1 & -j \end{pmatrix} \tag{4.43}$$

$$I_{1,2} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & -j \\ j & -j & j & -j \\ -1 & 1 & -1 & 1 \\ -j & -j & -j & j \end{pmatrix} \tag{4.44}$$

$$I_{3,3} = \frac{1}{4}\begin{pmatrix} 1 & -j & -1 & j \\ -j & -1 & j & 1 \\ -1 & j & 1 & -j \\ j & 1 & -j & -1 \end{pmatrix} \tag{4.45}$$

4.3.2 Discrete Walsh Transform (DWT)

The transformation kernels of the DWT (Walsh, 1923) are defined as

$$f(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{n-1}\left[(-1)^{p_i(x)p_{n-1-i}(u)}(-1)^{p_i(y)p_{n-1-i}(v)}\right] \tag{4.46}$$

and

$$i(x,y,u,v) = f(x,y,u,v). \tag{4.47}$$

where $n = \log_2 N$, $p_i(\text{arg})$ represents the ith bit in the natural binary representation of the arg, the oth bit corresponds to the least significant bit, and the $(n\text{-}1)$th bit corresponds to the most significant

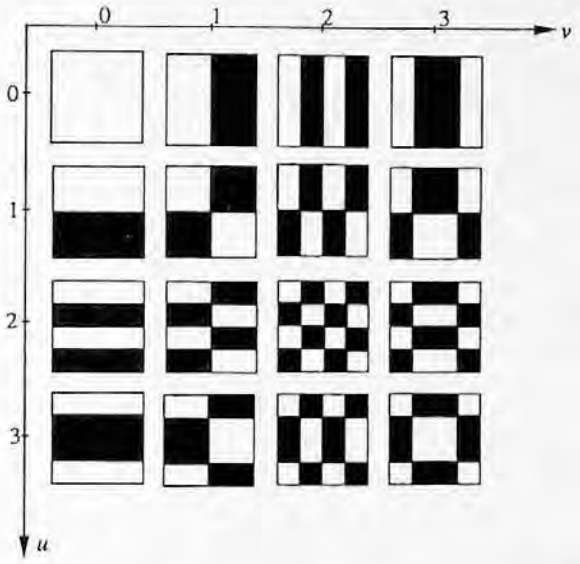


FIGURE 4.4 When N = 4: a set of the 16 basis images of DWT.

bit. For instance, consider $N = 16$, then $n = 4$. The natural binary code of number 8 is 1000. Hence, $p_0(8) = p_1(8) = p_2(8) = 0$, and $p_3(8) = 1$. We see that if the factor $1/N$ is put aside then the forward transformation kernel is always an integer: either +1 or −1. In addition, the inverse transformation kernel is the same as the forward transformation kernel. Therefore, we conclude that the implementation of the DWT is simple.

When $N = 4$, the 16 basis images of the DWT are shown in Figure 4.4. Each corresponds to a specific pair of (u, v) and is of resolution 4×4 in the x-y coordinate system. They are binary images, where the bright represents +1, while the dark −1. The transform matrix of the DWT is shown below for $N = 4$.

$$F = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{4.48}$$

4.3.3 Discrete Hadamard Transform (DHT)

The DHT (Hadamard, 1893) is closely related to the DWT. This can be seen from the following definition of the transformation kernels.

$$f(x, y, u, v) = \frac{1}{N}\prod_{i=0}^{n}\left[(-1)^{p_i(x)p_i(u)}(-1)^{p_i(y)p_i(v)}\right] \tag{4.49}$$

and

$$i(x, y, u, v) = f(x, y, u, v) \tag{4.50}$$

where the definitions of n, i, and p_i(arg) are the same as in the DWT. For this reason, the term Walsh-Hadamard transform (DWHT) is frequently used to represent either of the two transforms.

When N is a power of 2, the transform matrices of the DWT and DHT have the same row (or column) vectors except that the order of row (or column) vectors in the matrices are different. This is the only difference between the DWT and DHT under the circumstance $N = 2^n$. Because of this difference, while the DWT can be implemented by using the FFT algorithm with a straightforward modification, the DHT needs more work to use the FFT algorithm. On the other hand, the DHT possesses the following recursive feature, while the DWT does not:

$$F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.51}$$

and

$$F_{2N} = \begin{bmatrix} F_N & F_N \\ F_N & -F_N \end{bmatrix} \tag{4.52}$$

where the subscripts indicate the size of the transform matrices. It is obvious that the transform matrix of the DHT can be easily derived by using the recursion.

Note that the number of sign changes between consecutive entries in a row (or a column) of a transform matrix (from positive to negative and from negative to positive) is known as *sequency*. It is observed that the sequency does not monotonically increase as the order number of rows (or columns) increases in the DHT. Since sequency bears some similarity to frequency in the Fourier transform, sequency is desired as an increasing function of the order number of rows (or columns). This is realized by the *ordered* Hadamard transform (Gonzalez, 1992).

The transformation kernel of the ordered Hadamard transform is defined as

$$f(x,y,u,v) = \frac{1}{N}\prod_{i=0}^{N-1}\left[(-1)^{p_i(x)d_i(u)}(-1)^{p_i(y)d_i(v)}\right] \tag{4.53}$$

where the definitions of i, $p_i(\arg)$ are the same as defined above for the DWT and DHT. The $d_i(\arg)$ is defined as below.

$$\begin{aligned} d_0(\arg) &= b_{n-1}(\arg) \\ d_1(\arg) &= b_{n-1}(\arg) + b_{n-2}(\arg) \\ d_{n-1}(\arg) &= b_1(\arg) + b_0(\arg) \end{aligned} \tag{4.54}$$

The 16 basis images of the ordered Hadamard transform are shown in Figure 4.5 for $N = 4$. It is observed that the variation of the binary basis images becomes more frequent monotonically when u and v increase. Also we see that the basis image expansion is similar to the frequency expansion of the Fourier transform in the sense that an image is decomposed into components with different variations. In transform coding, these components with different coefficients are treated differently.

4.3.4 Discrete Cosine Transform (DCT)

The DCT is the most commonly used transform for image and video coding.

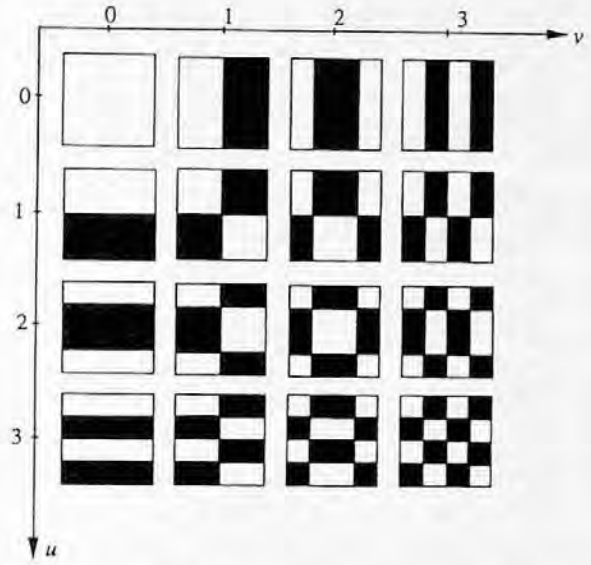


FIGURE 4.5 When N = 4: a set of the 16 basis images of the ordered DHT.

4.3.4.1 Background

The DCT, which plays an extremely important role in image and video coding, was established by Ahmed et al. (1974). There, it was shown that the *basis member* $\cos[(2x + 1)u\pi/2N]$ is the uth Chebyshev polynomial $T_U(\xi)$ evaluated at the xth zero of $T_N(\xi)$. Recall that the Chebyshev polynomials are defined as

$$T_0(\xi) = 1/\sqrt{2} \tag{4.55}$$

$$T_k(\xi) = \cos\left[k\cos^{-1}(\xi)\right] \tag{4.56}$$

where $T_K(\xi)$ is the kth order Chebyshev polynomial and it has k zeros, starting from the 1st zero to the kth zero. Furthermore, it was demonstrated that the basis vectors of 1-D DCT provide a good approximation to the eigenvectors of the class of Toeplitz matrices defined as

$$\begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix}, \tag{4.57}$$

where $0 < \rho < 1$.

4.3.4.2 Transformation Kernel

The transformation kernel of the 2-D DCT can be extended straightforwardly from that of 1-D DCT as follows:

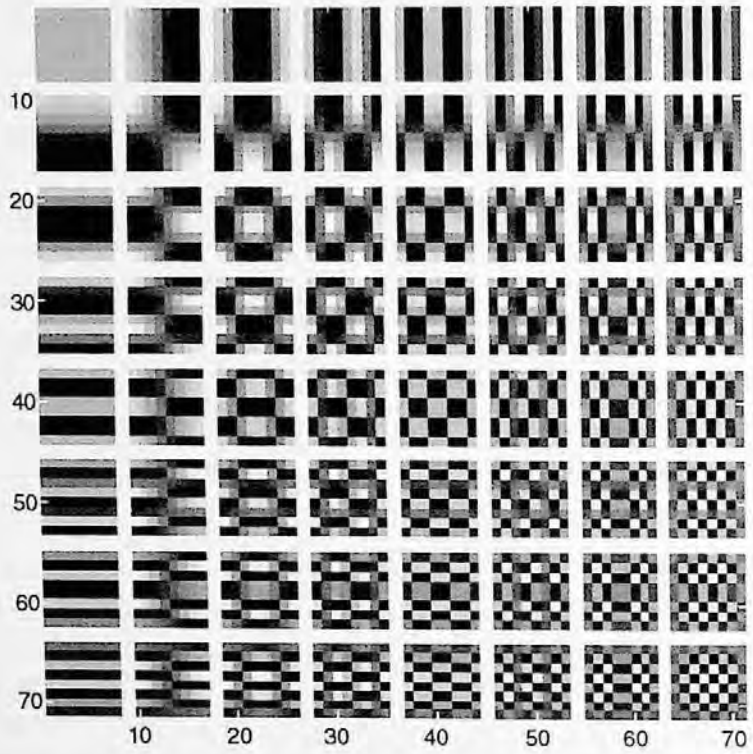


FIGURE 4.6 When N = 8: a set of the 64 basis images of the DCT.

$$f(x,y,u,v) = C(u)C(v)\cos\left(\frac{(2x+1)u\pi}{2N}\right)\cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{4.58}$$

where

$$C(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u = 1,2,\cdots,N-1 \end{cases} \tag{4.59}$$

$$i(x,y,u,v) = f(x,y,u,v). \tag{4.60}$$

Note that the $C(v)$ is defined the same way as in Equation 4.59. The 64 basis images of the DCT are shown in Figure 4.6 for $N = 8$.

4.3.4.3 Relationship with DFT

The DCT is closely related to the DFT. This can be examined from an alternative method of defining the DCT. It is known that applying the DFT to an N-point sequence $g_N(n)$, $n = 0,1,\cdots,N-1$, is equivalent to the following:

1. Repeating $g_N(n)$ every N points, form a periodic sequence, $\tilde{g}_N(n)$, with a fundamental period N. That is,

$$\tilde{g}_N(n) = \sum_{i=-\infty}^{\infty} g_N(n - iN). \tag{4.61}$$

2. Determine the Fourier series expansion of the periodic sequence $\tilde{g}_N(n)$. That is, determine all the coefficients in the Fourier series which are known to be periodic with the same fundamental period N.
3. Truncate the sequence of the Fourier series coefficients so as to have the same support as that of the given sequence $g_N(n)$. That is, only keep the N coefficients with indexes $0,1,\cdots,N-1$ and set all the others to equal zero. These N Fourier series coefficients form the DFT of the given N-point sequence $g_N(n)$.

An N-point sequence $g_N(n)$ and the periodic sequence $\tilde{g}_N(n)$, generated from $g_N(n)$, are shown in Figure 4.7(a) and (b), respectively. In summary, the DFT can be viewed as a correspondence between two periodic sequences. One is the periodic sequence $\tilde{g}_N(n)$, which is formed by periodically repeating $g_N(n)$. The other is the periodic sequence of Fourier series coefficients of $\tilde{g}_N(n)$.

The DCT of an N-point sequence is obtained via the following three steps:

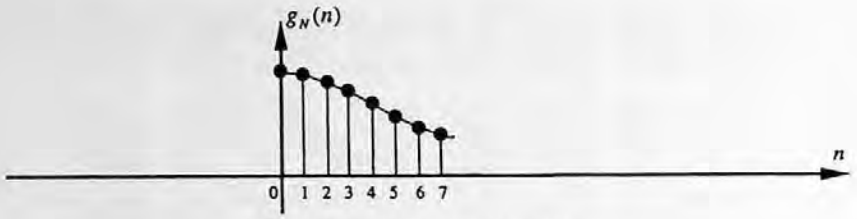
1. Flip over the given sequence with respect to the end point of the sequence to form a $2N$-point sequence, $g_{2N}(n)$, as shown in Figure 4.7(c). Then form a periodic sequence $\tilde{g}_{2N}(n)$, shown in Figure 4.7(d), according to

$$\tilde{g}_{2N}(n) = \sum_{i=-\infty}^{\infty} g_{2N}(n - 2iN) \tag{4.62}$$

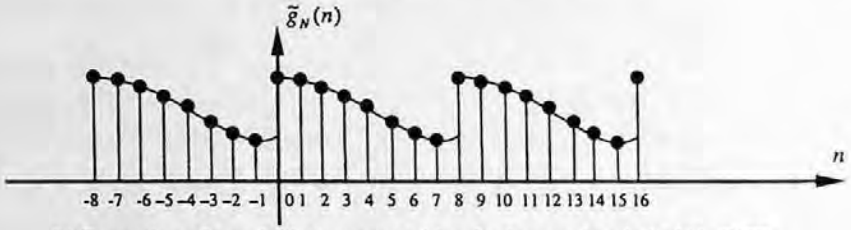
2. Find the Fourier series coefficients of the periodic sequences $\tilde{g}_{2N}(n)$.
3. Truncate the resultant periodic sequence of the Fourier series coefficients to have the support of the given finite sequence $g_N(n)$. That is, only keep the N coefficients with indexes $0,1,\cdots,N-1$ and set all the others to equal zero. These N Fourier series coefficients form the DCT of the given N-point sequence $g_N(n)$.

A comparison between Figure 4.7(b) and (d) reveals that the periodic sequence $\tilde{g}_N(n)$ is not smooth. There usually exist discontinuities at the beginning and end of each period. These end-head discontinuities cause a high-frequency distribution in the corresponding DFT. On the contrary, the periodic sequence $\tilde{g}_{2N}(n)$ does not have this type of discontinuity due to flipping over the given finite sequence. As a result, there is no high-frequency component corresponding to the end-head discontinuities. Hence, the DCT possesses better energy compaction in the low frequencies than the DFT. By better energy compaction, we mean more energy is compacted in a fraction of transform coefficients. For instance, it is known that the most energy of an image is contained in a small region of low frequency in the DFT domain. Vivid examples can be found in (Gonzalez and Woods, 1992). In terms of energy compaction, when compared with the Karhunen-Loeve transform (the Hotelling transform is its discrete version), which is known as the optimal, the DCT is the best among the DFT, DWT, DHT, and discrete Haar transform.

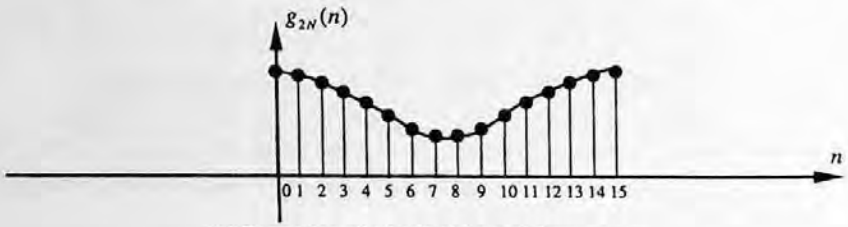
Besides this advantage, the DCT can be implemented using the FFT. This can be seen from the above discussion. There, it has been shown that the DCT of an N-point sequence, $g_N(n)$, can be obtained from the DFT of the $2N$-point sequence $g_{2N}(n)$. Furthermore, the even symmetry



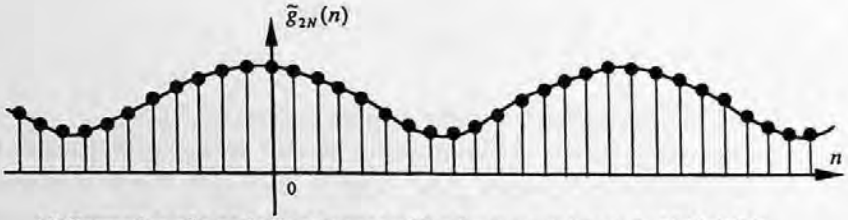
(a) Original 1-D input sequence



(b) Formation of a periodic sequence with a fundamental period of N (DFT)



(c) Formation of a back-to-back $2N$ sequence



(d) Formation of a periodic sequence with a fundamental period of $2N$ (DCT)

FIGURE 4.7 An example to illustrate the differences and similarities between DFT and DCT.

in $\tilde{g}_{2N}(n)$ makes the computation required for the DCT of an N-point equal to that required for the DFT of the N-point sequence. Because of these two merits, the DCT is the most popular image transform used in image and video coding nowadays.

4.3.5 Performance Comparison

In this subsection, we compare the performance of a few commonly used transforms in terms of energy compaction, mean square reconstruction error, and computational complexity.

4.3.5.1 Energy Compaction

Since all the transforms we discussed are symmetric (hence separable) and unitary, the matrix form of the 2-D image transform can be expressed as $T = F^T GF$ as discussed in Section 4.2.1.3. In the 1-D case, the transform matrix F is the counterpart of the matrix Φ discussed in the Hotelling

transform. Using the F, one can transform a 1-D column vector $\vec{z}$ into another 1-D column vector $\vec{y}$. The components of the vector $\vec{y}$ are transform coefficients. The variances of these transform coefficients, and therefore the signal energy associated with the transform coefficients, can be arranged in a nondecreasing order. It can be shown that the total energy before and after the transform remains the same. Therefore, the more energy compacted in a fraction of total coefficients, the better energy compaction the transform has. One measure of energy compaction is the *transform coding gain* G_{TC}, which is defined as the ratio between the arithmetic mean and the geometric mean of the variances of all the components in the transformed vector (Jayant, 1984).

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_i^2}{\left(\prod_{i=0}^{N-1}\sigma_i^2\right)^{\frac{1}{N}}} \tag{4.63}$$

A larger G_{TC} indicates higher energy compaction. The transform coding gains for a first-order autoregressive source with $\rho = 0.95$ achieved by using the DCT, DFT, and KLT was reported in (Zelinski and Noll, 1975; Jayant and Noll, 1984). The transform coding gain afforded by the DCT compares very closely to that of the optimum KLT.

4.3.5.2 Mean Square Reconstruction Error

The performance of the transforms can be compared in terms of the mean square reconstruction error as well. This was mentioned in Section 4.1.2 when we provided a statistical interpretation for transform coding. That is, after arranging all the N transformed coefficients according to their variances in a nonincreasing order, if $L < N$ and we discard the last $(N - L)$ coefficients to reconstruct the original input signal $\vec{z}$ (similar to what we did with the Hotelling transform), then the mean square reconstruction error is

$$MSE_r = E\left[\|\vec{z} - \vec{z}'\|^2\right] = \sum_{i=L+1}^{N}\sigma_i^2, \tag{4.64}$$

where $\vec{z}'$ denotes the reconstructed vector. Note that in the above-defined mean square reconstruction error, the quantization error and transmission error have not been included. Hence, it is sometimes referred to as the mean square approximation error. Therefore it is desired to choose a transform so that the transformed coefficients are "more independent" and more energy is concentrated in the first L coefficients. Then it is possible to discard the remaining coefficients to save coding bits without causing significant distortion in input signal reconstruction.

In terms of the mean square reconstruction error, the performance of the DCT, KLT, DFT, DWT, and discrete Haar transform for the 1-D case was reported in Ahmed et al. (1974). The variances of the 16 transform coefficients are shown in Figure 4.8 when $N = 16$, $\rho = 0.95$. Note that N stands for the dimension of the 1-D vector, while the parameter ρ is shown in the Toeplitz matrix (refer to Equation 4.57). We can see that the DCT compares most closely to the KLT, which is known to be optimum.

Note that the unequal variance distribution among transform coefficients has also found application in the field of pattern recognition. Similar results to those in Ahmed et al. (1974) for the DFT, DWT, and Haar transform were reported in (Andrews, 1971).

A similar analysis can be carried out for the 2-D case (Wintz, 1972). Recall that an image $g(x, y)$ can be expressed as a weighted sum of basis images $I_{u,v}$. That is,

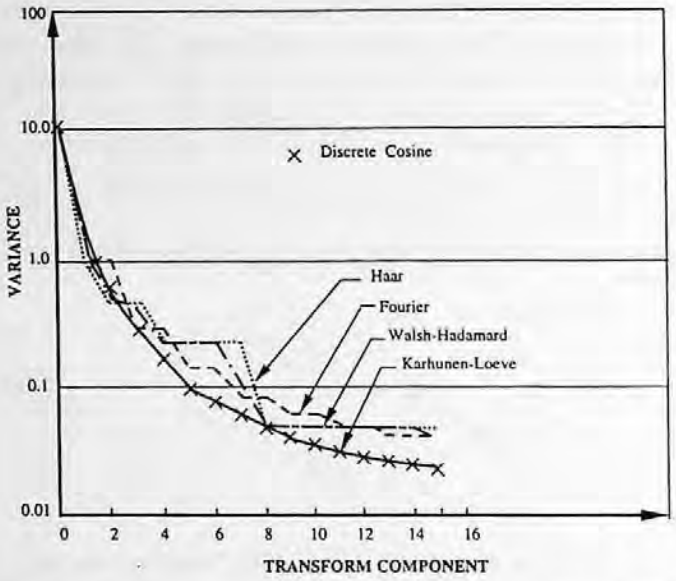


FIGURE 4.8 Transform coefficient variances when $N = 16$, $\rho = 0.95$. (From Ahmed, N. et al., *IEEE Trans. Comput.*, 90, 1974. With permission.)

$$G = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v) I_{u,v} \tag{4.65}$$

where the weights are transform coefficients. We arrange the coefficients according to their variances in a nonincreasing order. For some choices of the transform (hence basis images), the coefficients become insignificant after the first L terms, and the image can be approximated well by truncating the coefficients after L. That is,

$$G = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v) I_{u,v} \approx \sum_{u=0}^{L}\sum_{v=0}^{L} T(u,v) I_{u,v} \tag{4.66}$$

The mean square reconstruction error is given by

$$MSE_r = \sum_{L}^{N-1}\sum^{N-1} \sigma_{u,v}^2 \tag{4.67}$$

A comparison among the KLT, DHT, and DFT in terms of the mean square reconstruction error for 2-D array of 16×16 (i.e., 256 transform coefficients) was reported in (Figure 5, Wintz, 1972). Note that the discrete KLT is image dependent. In the comparison, the KLT is calculated with respect to an image named "Cameraman." It shows that while the KLT achieves the best performance, the other transforms perform closely.

In essence, the criteria of mean square reconstruction error and energy compaction are closely related. It has been shown that the discrete Karhunen transform (KLT), also known as the Hotelling transform, is the optimum in terms of energy compaction and mean square reconstruction error. The DWT, DHT, DFT, and DCT are close to the optimum (Wintz, 1972; Ahmed et al., 1974); however, the DCT is the best among these several *suboptimum* transforms.

Note that the performance comparison among various transforms in terms of bit rate vs. distortion in the reconstructed image was reported in (Pearl et al., 1972; Ahmed et al., 1974). The same conclusion was drawn. That is, the KLT is optimum, while the DFT, DWT, DCT, and Haar transforms are close in performance. Among the suboptimum transforms, the DCT is the best.

4.3.5.3 Computational Complexity

Note that while the DWT, DHT, DFT, and DCT are input image independent, the discrete KLT (the Hotelling transform) is input dependent. More specifically, the row vectors of the Hotelling transform matrix are transposed eigenvectors of the covariance matrix of the input random vector. So far there is no fast transform algorithm available. This computational complexity prohibits the Hotelling transform from practical usage. It can be shown that the DWT, DFT, and DCT can be implemented using the FFT algorithm.

4.3.5.4 Summary

As pointed out above, the DCT is the best among the suboptimum transforms in terms of energy compaction. Moreover, the DCT can be implemented using the FFT. Even though a $2N$-point sequence is involved, the even symmetry makes the computation involved in the N-point DCT equivalent to that of the N-point FFT. For these two reasons, the DCT finds the widest application in image and video coding.

4.4 BIT ALLOCATION

As shown in Figure 4.2, in transform coding, an input image is first divided into blocks (subimages). Then a 2-D linear transform is applied to each block. The transformed blocks go through truncation, quantization, and codeword assignment. The last three functions: truncation, quantization, and codeword assignment, are combined and called bit allocation.

From the previous section, it is known that the applied transform decorrelates subimages. Moreover, it redistributes image energy in the transform domain in such a way that most of the energy is compacted into a small fraction of coefficients. Therefore, it is possible to discard the majority of transform coefficients without introducing significant distortion.

As a result, we see that in transform coding there are mainly three types of errors involved. One is due to truncation. That is, the majority of coefficients are truncated to zero. Others come from quantization. (Note that truncation can also be considered a special type of quantization). Transmission errors are the third type of error. Recall that the mean square reconstruction error discussed in Section 4.3.5.2 is in fact only related to truncation error. For this reason, it was referred to more precisely as a mean square approximation error. In general, the reconstruction error, i.e., the error between the original image signal and the reconstructed image at the receiver, includes three types of errors: truncation error, quantization error, and transmission error.

There are two different ways to truncate transform coefficients. One is called *zonal coding*, while the other is *threshold coding*. They are discussed below.

4.4.1 Zonal Coding

In zonal coding, also known as *zonal sampling*, a zone in the transformed block is predefined according to a statistical average obtained from many blocks. All transform coefficients in the zone are retained, while all coefficients outside the zone are set to zero. As mentioned in Section 4.3.5.1, the total energy of the image remains the same after applying the transforms discussed there. Since it is known that the DC and low-frequency AC coefficients of the DCT occupy most of the energy, the zone is located in the top-left portion of the transformed block when the transform coordinate

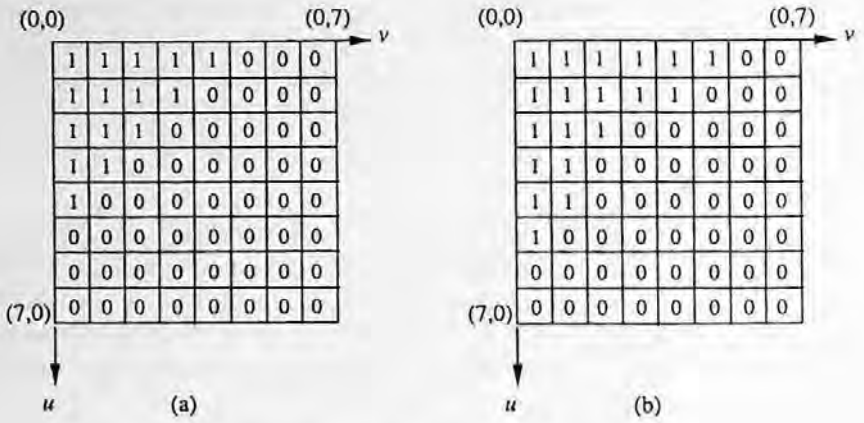


FIGURE 4.9 Two illustrations of zonal coding.

system is set conventionally. (Note that by DC we mean $u = v = 0$. By AC we mean u and v do not equal zero simultaneously.) That is, the origin is at the top-left corner of the transformed block. Two typical zones are shown in Figure 4.9. The simplest uniform quantization with natural binary coding can be used to quantize and encode the retained transform coefficients. With this simple technique, there is no overhead side information that needs to be sent to the receiver, since the structure of the zone, the scheme of the quantization, and encoding are known at both the transmitter and receiver.

The coding efficiency, however, may not be very high. This is because the zone is predefined based on average statistics. Therefore some coefficients outside the zone might be large in magnitude, while some coefficients inside the zone may be small in quantity. Uniform quantization and natural binary encoding are simple, but they are known not to be efficient enough.

For further improvement of coding efficiency, an adaptive scheme has to be used. There, a two-pass procedure is applied. In the first pass, the variances of transform coefficients are measured or estimated. Based on the statistics, the quantization and encoding schemes are determined. In the second pass, quantization and encoding are carried out (Habibi, 1971a; Chen and Smith, 1977).

4.4.2 Threshold Coding

In threshold coding, also known as threshold sampling, there is not a predefined zone. Instead, each transform coefficient is compared with a threshold. If it is smaller than the threshold, then it is set to zero. If it is larger than the threshold, it will be retained for quantization and encoding. Compared with zonal coding, this scheme is adaptive in truncation in the sense that the coefficients with more energy are retained no matter where they are located. The addresses of these retained coefficients, however, have to be sent to the receiver as side information. Furthermore, the threshold is determined after an evaluation of all coefficients. Hence, it was usually a two-pass adaptive technique.

Chen and Pratt (1984) devised an efficient adaptive scheme to handle threshold coding. It is a one-pass adaptive scheme, in contrast to the two-pass adaptive schemes. Hence it is fast in implementation. With several effective techniques that will be addressed here, it achieved excellent results in transform coding. Specifically, it demonstrated a satisfactory quality of reconstructed frames at a bit rate of 0.4 bits per pixel for coding of color images, which corresponds to real-time color television transmission over a 1.5-Mb/sec channel. This scheme has been adopted by the international still coding standard JPEG. A block diagram of the threshold coding proposed by Chen and Pratt is shown in Figure 4.10. More details and modification made by JPEG will be described in Chapter 7.

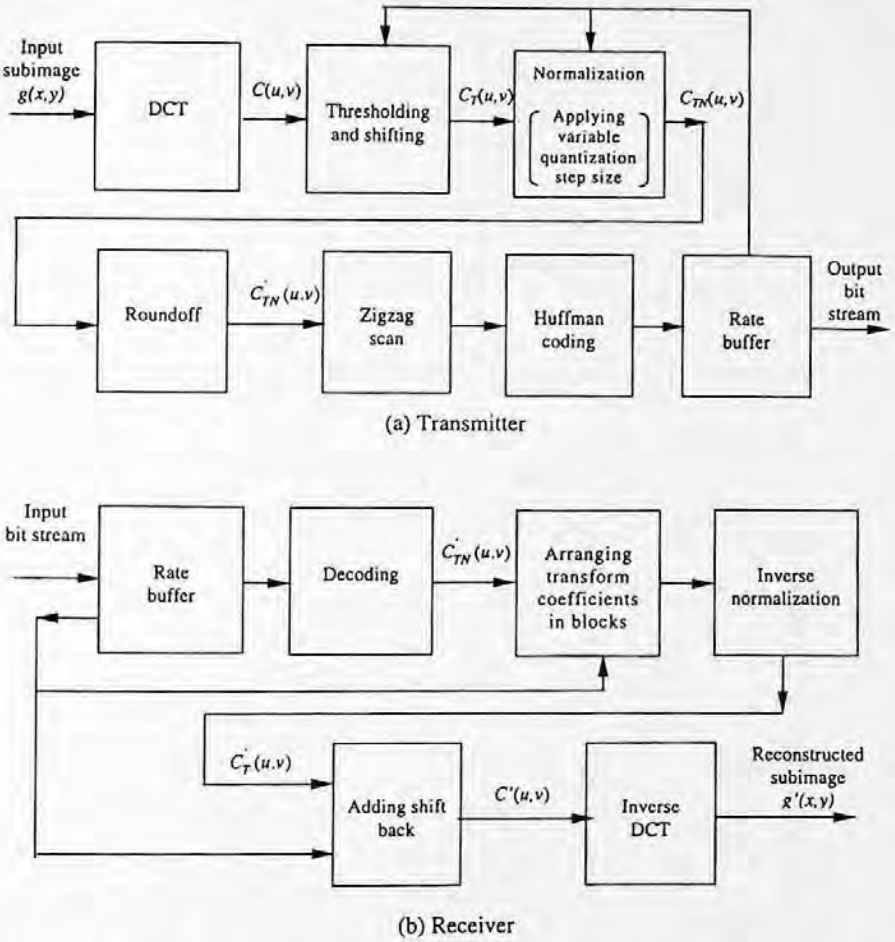


FIGURE 4.10 Block diagram of the algorithm proposed by Chen and Pratt (1984).

4.4.2.1 Thresholding and Shifting

The DCT is used in the scheme because of its superiority, described in Section 4.3. Here we use $C(u,v)$ to denote the DCT coefficients. The DC coefficient, $C(0,0)$, is processed differently. As mentioned in Chapter 3, the DC coefficients are encoded with a differential coding technique. For more details, refer to Chapter 7. For all the AC coefficients, the following thresholding and shifting are carried out:

$$C_T(u,v) = \begin{cases} C(u,v) - T & \text{if} \quad C(u,v) > T \\ 0 & \text{if} \quad C(u,v) \le T \end{cases} \tag{4.68}$$

where T on the right-hand side is the threshold. Note that the above equation also implies a shifting of transform coefficients by T when $C(u, v) > T$. The input-output characteristic of the thresholding and shifting is shown in Figure 4.11.

Figure 4.12 demonstrates that more than 60% of the DCT coefficients normally fall below a threshold value as low as 5. This indicates that with a properly selected threshold value it is possible to set most of the DCT coefficients equal to zero. The threshold value is adjusted by the feedback from the rate buffer, or by the desired bit rate.

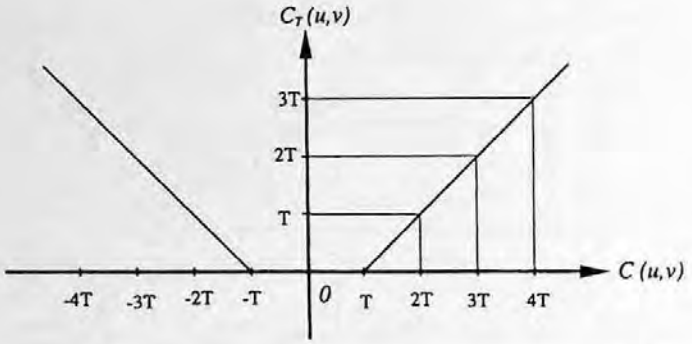


FIGURE 4.11 Input-output characteristic of thresholding and shifting.

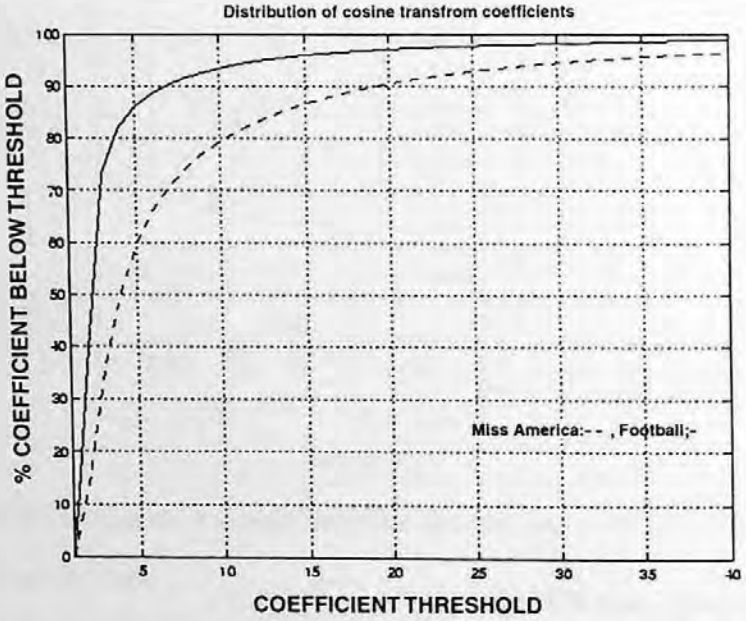


FIGURE 4.12 Amplitude distribution of the DCT coefficients.

4.4.2.2 Normalization and Roundoff

The threshold subtracted transform coefficients $C_T(u,v)$ are normalized before roundoff. The normalization is implemented as follows:

$$C_{TN}(u,v) = \frac{C_T(u,v)}{\Gamma_{u,v}} \tag{4.69}$$

where the normalization factor $\Gamma_{u,v}$ is controlled by the rate buffer. The roundoff process converts floating point to integer as follows.

$$R\left[C_{TN}(u,v)\right] = C_{TN}(u,v) = \begin{cases} \lfloor C_{TN}(u,v) + 0.5 \rfloor & \text{if} \quad C_{TN}(u,v) \geq 0 \\ \lceil C_{TN}(u,v) - 0.5 \rceil & \text{if} \quad C_{TN}(u,v) < 0 \end{cases} \tag{4.70}$$

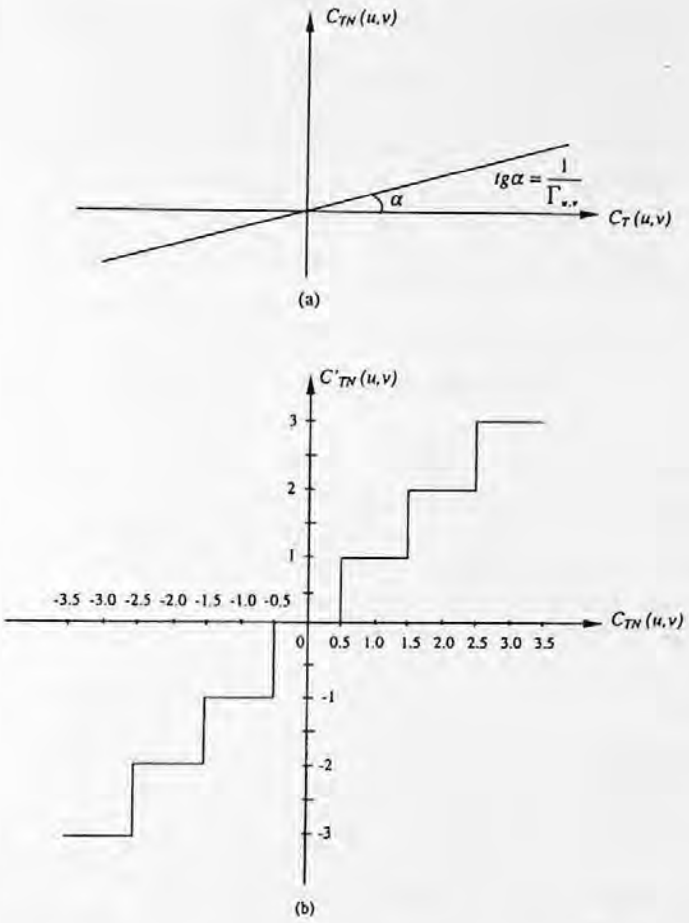


FIGURE 4.13 Input-output characteristic of (a) normalization, (b) roundoff.

where the operator $\lfloor x \rfloor$ means the largest integer smaller than or equal to x, the operator $\lceil x \rceil$ means the smallest integer larger than or equal to x. The input-output characteristics of the normalization and roundoff are shown in Figure 4.13(a) and (b), respectively.

From these input-output characteristics, we can see that the roundoff is a uniform midtread quantizer with a unit quantization step. The combination of normalization and roundoff is equivalent to a uniform midtread quantizer with the quantization step size equal to the normalization factor $\Gamma_{u,v}$. Normalization is a scaling process, which makes the resultant uniform midtread quantizer adapt to the dynamic range of the associated transform coefficient. It is therefore possible for one quantizer design to be applied to various coefficients with different ranges. Obviously, by adjusting the parameter $\Gamma_{u,v}$ (quantization step size) a variable bit rate and mean square quantization error can be achieved. Hence, the selection of the normalization factors for different transform coefficients can take the statistical feature of the images and the characteristics of the human visual system (HVS) into consideration. In general, most image energy is contained in the DC and low-frequency AC transform coefficients. The HVS is more sensitive to a relatively uniform region than to a relatively detailed region, as discussed in Chapter 1. Chapter 1 also mentions that, with regard to the color image, the HVS is more sensitive to the luminance component than to the chrominance components.

These have been taken into consideration in JPEG. A matrix consisting of all the normalization factors is called a quantization table in JPEG. A luminance quantization table and a chrominance quantization table used in JPEG are shown in Figure 4.14. We observe that in general in both tables

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a) Luminance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b) Chrominance quantization table

FIGURE 4.14 Quantization tables.

the small normalization factors are assigned to the DC and low-frequency AC coefficients. The large Γs are associated with the high-frequency transform coefficients. Compared with the luminance quantization table, the chrominance quantization table has larger quantization step sizes for the low- and middle-frequency coefficients and almost the same step sizes for the DC and high-frequency coefficients, indicating that the chrominance components are relatively coarsely quantized, compared with the luminance component.

4.4.2.3 Zigzag Scan

As mentioned at the beginning of this section, while threshold coding is adaptive to the local statistics and hence is more efficient in truncation, threshold coding needs to send the addresses of retained coefficients to the receiver as overhead side information. An efficient scheme, called the zigzag scan, was proposed by Chen and Pratt (1984) and is shown in Figure 4.14. As shown in Figure 4.12, a great majority of transform coefficients have magnitudes smaller than a threshold of 5. Consequently, most quantized coefficients are zero. Hence, in the 1-D sequence obtained by zigzag scanning, most of the numbers are zero. A code known as run-length code, discussed in Chapter 6, is very efficient under these circumstances to encode the address information of nonzero coefficients. Run-length of zero coefficients is understood as the number of consecutive zeros in the zigzag scan. Zigzag scanning minimizes the use of run-length codes in the block.

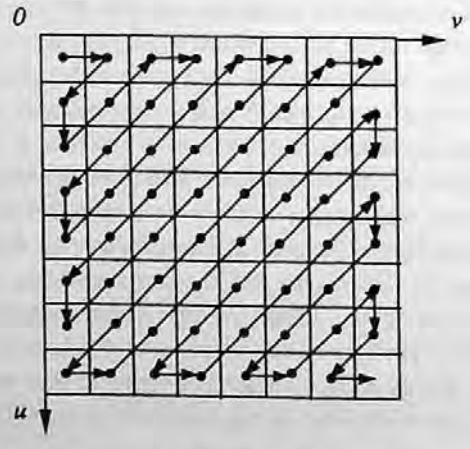


FIGURE 4.15 Zigzag scan of DCT coefficients within an 8 × 8 block.

4.4.2.4 Huffman Coding

Statistical studies of the magnitude of nonzero DCT coefficients and the run-length of zero DCT coefficients in zigzag scanning were conducted in (Chen and Pratt, 1984). The domination of the coefficients with small amplitudes and the short run-lengths was found and is shown in Figures 4.16 and 4.17. This justifies the application of the Huffman coding to the magnitude of nonzero transform coefficients and run-lengths of zeros.

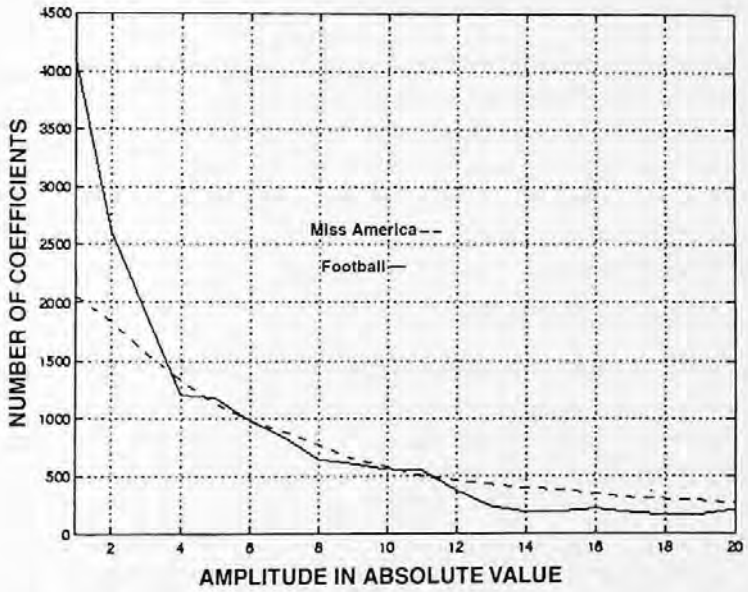


FIGURE 4.16 Histogram of DCT coefficients in absolute amplitude.

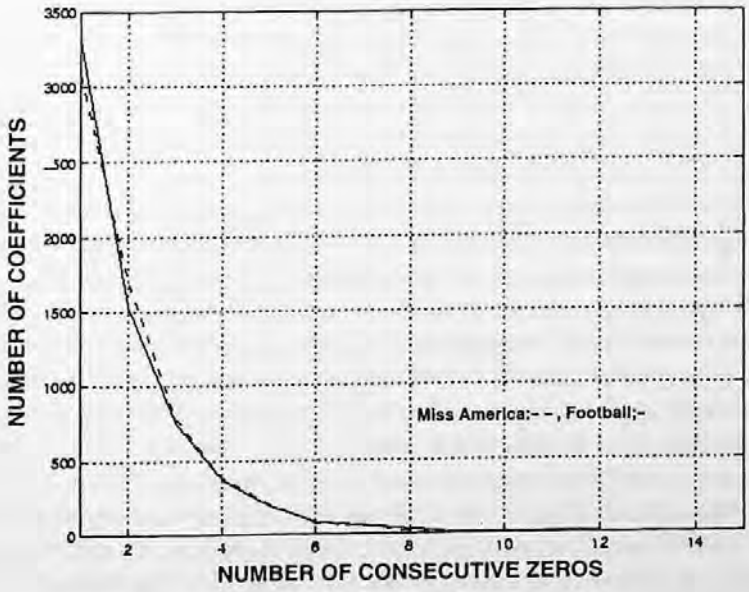


FIGURE 4.17 Histogram of zero run length.

4.4.2.5 Special Codewords

Two special codewords were used by Chen and Pratt (1984). One is called *end of block* (EOB). Another is called *run-length prefix*. Once the last nonzero DCT coefficients in the zigzag is coded, EOB is appended, indicating the termination of coding the block. This further saves bits used in coding. A run-length prefix is used to discriminate the run-length codewords from the amplitude codewords.

4.4.2.6 Rate Buffer Feedback and Equalization

As shown in Figure 4.10, a rate buffer accepts a variable-rate data input from the encoding process and provides a fixed-rate data output to the channel. The status of the rate buffer is monitored and fed back to control the threshold and the normalization factor. In this fashion a one-pass adaptation is achieved.

4.5 SOME ISSUES

4.5.1 Effect of Transmission Errors

In transform coding, each pixel in the reconstructed image relies on all transform coefficients in the subimage where the pixel is located. Hence, a bit reversal transmission error will spread. That is, an error in a transform coefficient will lead to errors in all the pixels within the subimage. As discussed in Section 4.2.3, this is one of the reasons the selected subimage size cannot be very large. Depending on which coefficient is in error, the effect caused by a bit reversal error on the reconstructed image varies. For instance, an error in the DC or a low-frequency AC coefficient may be objectionable, while an error in the high-frequency coefficient may be less noticeable.

4.5.2 Reconstruction Error Sources

As discussed, three sources: truncation (discarding transform coefficients with small variances), quantization, and transmission contribute to the reconstruction error. It is noted that in TC the transform is applied block by block. Quantization and encoding of transform coefficients are also conducted blockwise. At the receiver, reconstructed blocks are put together to form the whole reconstructed image. In the process, block artifacts are produced. Sometimes, even though it may not severely affect an objective assessment of the reconstructed image quality, block artifacts can be annoying to the HVS, especially when the coding rate is low.

To alleviate the blocking effect, several techniques have been proposed. One is to overlap blocks in the source image. Another is to postfilter the reconstructed image along block boundaries. The selection of advanced transforms is an additional possible method (Lim, 1990).

In the block-overlapping method, when the blocks are finally organized to form the reconstructed image, each pixel in the overlapped regions takes an average value of all its reconstructed gray level values from multiple blocks. In this method, extra bits are used for those pixels involved in the overlapped regions. For this reason, the overlapped region is usually only one pixel wide.

Due to the sharp transition along block boundaries, block artifacts are of high frequency in nature. Hence, low-pass filtering is normally used in the postfiltering method. To avoid the blurring effect caused by low-pass filtering on the nonboundary image area, low-pass postfiltering is only applied to block boundaries. Unlike the block-overlapping method, the postfiltering method does not need extra bits. Moreover, it has been shown that the postfiltering method can achieve better results in combating block artifacts (Reeve and Lim, 1984; Ramamurthi and Gersho, 1986). For these two reasons, the postfiltering method has been adopted by the international coding standards.

4.5.3 Comparison Between DPCM and TC

As mentioned at the beginning of the chapter, both differential coding and transform coding utilize interpixel correlation and are efficient coding techniques. Comparisons between these two techniques have been reported (Habibi, 1971b). Take a look at the techniques discussed in the previous chapter and in this chapter. We can see that differential coding is simpler than TC. This is because the linear prediction and differencing involved in differential coding are simpler than the 2-D transform involved in TC. In terms of the memory requirement and processing delay, differential coding such as DPCM is superior to TC. That is, DPCM needs less memory and has less processing delay than TC. The design of the DPCM system, however, is sensitive to image-to-image variation, and so is its performance. That is, an optimum DPCM design is matched to the statistics of a certain image. When the statistics change, the performance of the DPCM will be affected. On the contrary, TC is less sensitive to the variation in the image statistics. In general, the optimum DPCM coding system with a third or higher order predictor performs better than TC when the bit rate is about two to three bits per pixel for single images. When the bit rate is below two to three bits per pixel, TC is normally preferred. As a result, the international still image coding standard JPEG is based on TC, whereas, in JPEG, DPCM is used for coding the DC coefficients of DCT, and information-preserving differential coding is used for lossless still image coding.

4.5.4 Hybrid Coding

A method called hybrid transform/waveform coding, or simply hybrid coding, was devised in order to combine the merits of the two methods. By waveform coding, we mean coding techniques that code the waveform of a signal instead of the transformed signal. DPCM is a waveform coding technique. Hybrid coding combines TC and DPCM coding. That is, TC can be first applied rowwise, followed by DPCM coding columnwise, or vice versa. In this way, the two techniques complement each other. That is, the hybrid coding technique simultaneously has TC's small sensitivity to variable image statistics and DPCM's simplicity in implementation.

Worth mentioning is a successful hybrid coding scheme in interframe coding: predictive coding along the temporal domain. Specifically, it uses motion-compensated predictive coding. That is, the motion analyzed from successive frames is used to more accurately predict a frame. The prediction error (in the 2-D spatial domain) is transform coded. This hybrid coding scheme has been very efficient and was adopted by the international video coding standards H.261, H.263, and MPEG 1, 2, and 4.

4.6 SUMMARY

In transform coding, instead of the original image or some function of the original image in the spatial and/or temporal domain, the image in the transform domain is quantized and encoded. The main idea behind transform coding is that the transformed version of the image is less correlated. Moreover, the image energy is compacted into a small proper subset of transform coefficients.

The basis vector (1-D) and the basis image (2-D) provide a meaningful interpretation of transform coding. This type of interpretation considers the original image to be a weighted sum of basis vectors or basis images. The weights are the transform coefficients, each of which is essentially a correlation measure between the original image and the corresponding basis image. These weights are less correlated than the gray level values of pixels in the original image. Furthermore they have a great disparity in variance distribution. Some weights have large variances. They are retained and finely quantized. Some weights have small energy. They are retained and coarsely quantized. A vast majority of weights are insignificant and discarded. In this way, a high coding efficiency is achieved in transform coding. Because the quantized nonzero coefficients have a very nonuniform

probability distribution, they can be encoded by using efficient variable-length codes. In summary, three factors: truncation (discarding a great majority of transform coefficients), adaptive quantization, and variable-length coding contribute mainly to a high coding efficiency of transform coding.

Several linear, reversible, unitary transforms have been studied and utilized in transform coding. They include the discrete Karhunen-Loeve transform (the Hotelling transform), the discrete Fourier transform, the Walsh transform, the Hadamard transform, and the discrete cosine transform. It is shown that the KLT is the optimum. The transform coefficients of the KLT are uncorrelated. The KLT can compact the most energy in the smallest fraction of transform coefficients. However, the KLT is image dependent. There is no fast algorithm to implement it. This prohibits the KLT from practical use in transform coding. While the rest of the transforms perform closely, the DCT appears to be the best. Its energy compaction is very close to the optimum KLT and it can be implemented using the fast Fourier transform. The DCT has been found to be efficient not only for still images coding but also for coding residual images (predictive error) in motion-compensated interframe predictive coding. These features make the DCT the most widely used transform in image and video coding.

There are two ways to truncate transform coefficients: zonal coding and threshold coding. In zonal coding, a zone is predefined based on average statistics. The transform coefficients within the zone are retained, while those outside the zone are discarded. In threshold coding, each transform coefficient is compared with a threshold. Those coefficients larger than the threshold are retained, while those smaller are discarded. Threshold coding is adaptive to local statistics. A two-pass procedure is usually taken. That is, the local statistics are measured or estimated in the first pass. The truncation takes place in the second pass. The addresses of the retained coefficients need to be sent to the receiver as overhead side information.

A one-step adaptive framework of transform coding has evolved as a result of the tremendous research efforts in image coding. It has become a base of the international still image coding standard JPEG. Its fundamental components include the DCT transform, thresholding and adaptive quantization of transform coefficients, zigzag scan, Huffman coding of the magnitude of the nonzero DCT coefficients and run-length of zeros in the zigzag scan, the codeword of EOB, and rate buffer feedback control.

The threshold and the normalization factor are controlled by rate buffer feedback. Since the threshold decides how many transform coefficients are retained and the normalization factor is actually the quantization step size, the rate buffer has direct impact on the bit rate of the transform coding system. Selection of quantization steps takes the energy compaction of the DCT and the characteristics of the HVS into consideration. That is, it uses not only statistical redundancy, but also psychovisual redundancy to enhance coding efficiency.

After thresholding, normalization and roundoff are applied to the DCT transform coefficients in a block; a great majority of transform coefficients are set to zero. A zigzag scan can convert the 2-D array of transform coefficients into a 1-D sequence. The number of consecutive zero-valued coefficients in the 1-D sequence is referred to as the run-length of zeros and is used to provide address information of nonzero DCT coefficients. Both the magnitude of nonzero coefficients and run-length information need to be coded. Statistical analysis has demonstrated that a small magnitude and short run-length are dominant. Therefore, efficient lossless entropy coding methods such as Huffman coding and arithmetic coding (the focus of the next chapter) can be applied to magnitude and run-length.

In a reconstructed subimage, there are three types of errors involved: truncation error (some transform coefficients have been set to zero), quantization error, and transmission error. In a broad sense, the truncation can be viewed as a part of the quantization. That is, these truncated coefficients are quantized to zero. The transmission error in terms of bit reversal will affect the whole reconstructed subimage. This is because, in the inverse transform (such as the inverse DCT), each transform coefficient makes a contribution.

In reconstructing the original image all the subimages are organized to form the whole image. Therefore the independent processing of individual subimages causes block artifacts. Though they may not severely affect the objective assessment of reconstructed image quality, block artifacts can be annoying, especially in low bit rate image coding. Block overlappling and postfiltering are two effective ways to alleviate block artifacts. In the former, neighboring blocks are purposely overlapped by one pixel. In reconstructing the image, those pixels that have been coded more than once take an average of the multiple decoded values. Extra bits are used. In the latter technique, a low-pass filter is applied along boundaries of blocks. No extra bits are required in the process and the effect of combating block artifacts is better than with the former technique.

The selection of subimage size is an important issue in the implementation of transform coding. In general, a large size will remove more interpixel redundancy. But it has been shown that the pixel correlation becomes insignificant when the distance of pixels exceeds 20. On the other hand, a large size is not suitable for adaptation to local statistics, while adaptation is required in handling nonstationary images. A large size also makes the effect of a transmission error spread more widely. For these reasons, subimage size should not be large. In motion-compensated predictive interframe coding, motion estimation is normally carried out in sizes of 16×16 or 8×8. To be compatible, the subimage size in transform coding is normally chosen as 8×8.

Both predictive codings, say, DPCM and TC, utilize interpixel correlation and are efficient coding schemes. Compared with TC, DPCM is simpler in computation. It needs less storage and has less processing delay. But it is more sensitive to image-to-image variation. On the other hand, TC provides higher adaptation to statistical variation. TC is capable of removing more interpixel correlation, thus providing higher coding efficiency. Traditionally, predictive coding is preferred if the bit rate is in the range of two to three bits per pixel, while TC is preferred when bit rate is below two to three bits per pixel. However, the situation changes. TC becomes the core technology in image and video coding. Many special VLSI chips are designed and manufactured for reducing computational complexity. Consequently, predictive coding such as DPCM is only used in some very simple applications.

In the context of interframe coding, 3-D (two spatial dimensions and one temporal dimension) transform coding has not found wide application in practice due to the complexity in computation and storage. Hybrid transform/waveform coding has proven to be very efficient in interframe coding. There, motion-compensated predictive coding is used along the temporal dimension, while transform coding is used to code the prediction error in two spatial dimensions.

4.7 EXERCISES

4-1. Consider the following eight points in a 3-D coordinate system: $(0,0,0)^T$, $(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T$, $(0,1,1)^T$, $(1,0,1)^T$, $(1,1,0)^T$, $(1,1,1)^T$. Find the mean vector and covariance matrix using Equations 4.12 and 4.13.

4-2. For N = 4, find the basis images of the DFT, $I_{u,v}$ when (a) u = 0, v = 0; (b) u = 1, v = 0; (c) u = 2, v = 2; (d) u = 3, v = 2. Use both methods discussed in the text; i.e., the method with basis image and the method with basis vectors.

4-3. For N = 4, find the basis images of the ordered discrete Hadamard transform when (a) u = 0, v = 2; (b) u = 1, v = 3; (c) u = 2, v = 3; (d) u = 3, v = 3. Verify your results by comparing them with Figure 4.5.

4-4. Repeat the previous problem for the DWT, and verify your results by comparing them with Figure 4.4.

4-5. Repeat problem 4-3 for the DCT and N = 4.

4-6. When $N = 8$, draw the transform matrix F for the DWT, DHT, the order DHT, DFT, and DCT.

4-7. The matrix form of forward and inverse 2-D symmetric image transforms are expressed in texts such as (Jayant and Noll, 1984) as $T = FGF^T$ and $G = ITI^T$, which are different from Equations 4.28 and 4.29. Can you explain this discrepancy?
4-8. Derive Equation 4.64. (Hint: use the concept of basis vectors and the orthogonality of basis vectors.)
4-9. Justify that the normalization factor is the quantization step.
4-10. The transform used in TC has two functions: decorrelation and energy compaction. Does decorrelation automatically lead to energy compaction? Comment.
4-11. Using your own words, explain the main idea behind transform coding.
4-12. Read the techniques by Chen and Pratt presented in Section 4.4.2. Compare them with JPEG discussed in Chapter 7. Comment on the similarity and dissimilarity between them.
4-13. How is one-pass adaptation to local statistics in the Chen and Pratt algorithm achieved?
4-14. Explain why the DCT is superior to the DFT in terms of energy compaction.
4-15. Why is the subimage size of 8×8 widely used?

REFERENCES

Ahmed, N., T. Nararajan, and K. R. Rao, Discrete cosine transform, *IEEE Trans. Comput.*, 90-93, 1974.

Andrews, H. C. Multidimensional rotations in feature selection, *IEEE Trans. Comput.*, c-20, 1045-1051, 1971.

Chen, W. H. and C. H. Smith, Adaptive coding of monochrome and color images, *IEEE Trans. Commun.*, COM-25, 1285-1292, 1977.

Chen, W. H. and W. K. Pratt, Scene adaptive coder, *IEEE Trans. Commun.*, COM-32, 225-232, 1984.

Cooley, J. W. and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. Comput.*, 19, 297-301, 1965.

Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

Hadamard, J. Resolution d'une question relative aux determinants, *Bull. Sci. Math., Ser. 2*, 17, Part I, 240-246, 1893.

Habibi, A. and P. A. Wintz, Image coding by linear transformations and block quantization, *IEEE Trans. Commun. Technol.*, com-19, 50-60, 1971.

Habibi, A. Comparison of nth-order DPCM encoder with linear transformations and block quantization techniques, *IEEE Trans. Commun. Technol.*, com-19(6), 948-956, 1971.

Huang, J.-Y. and P. M. Schultheiss, Block quantization of correlated Gaussian random variables, *IEEE Trans. Commun. Syst.*, cs-11, 289-296, 1963.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Karhunen, K., On Linear Methods in Probability Theory, Ann. Acad. Sci. Fennicae, Ser. A137. (Translated into English.) The Rand Corp., Santa Monica, CA, 1960.

Lim, J. S. *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Loeve, M., Fonctions Aléatoires de Second Ordre, in P. Levy, *Processus Stochastique et Mouvement Brownien*, Hermann, Paris.

Pearl, J., H. C. Andrews, and W. K. Pratt, Performance measures for transform data coding, *IEEE Trans. Commun. Technol.*, com-20, 411-415, 1972.

Reeve, H. C. III and J. S. Lim, Reduction of blocking effects in image coding, *J. Opt. Eng.*, 23, 34-37, 1984.

Ramamurthi, B. and A. Gersho, Nonlinear space-variant postprocessing of block coded images, *IEEE Trans. Acoust. Speech Signal Process.*, 34, 1258-1267, 1986.

Strang, G., *Introduction to Linear Algebra*, Wellesley-Cambridge Press, 2nd ed., 1998.

Tasto, M. and P. A. Wintz, Image coding by adaptive block quantization, *IEEE Trans. Commun. Technol.*, com-19(6), 957-972, 1971.

Walsh, J. L. A closed set of normal orthogonal functions, *Am. J. Math.*, 45(1), 5-24, 1923.

Wintz, P. A. Transform picture coding, *Proc. IEEE*, 60(7), 809-820, 1972.

Zelinski, R. and P. Noll, Adaptive block quantization of speech signals, (in German), Tech. Rep. no. 181, Heinrich Hertz Institut, Berlin, 1975.

5 Variable-Length Coding: Information Theory Results (II)

Recall the block diagram of encoders shown in Figure 2.3. There are three stages that take place in an encoder: transformation, quantization, and codeword assignment. Quantization was discussed in Chapter 2. Differential coding and transform coding using two different transformation components were covered in Chapters 3 and 4, respectively. In differential coding it is the difference signal that is quantized and encoded, while in transform coding it is the transformed signal that is quantized and encoded. In this chapter and the next chapter, we discuss several codeword assignment (encoding) techniques. In this chapter we cover two types of variable-length coding: Huffman coding and arithmetic coding.

First we introduce some fundamental concepts of encoding. After that, the rules that must be obeyed by all optimum and instantaneous codes are discussed. Based on these rules, the Huffman coding algorithm is presented. A modified version of the Huffman coding algorithm is introduced as an efficient way to dramatically reduce codebook memory while keeping almost the same optimality.

The promising arithmetic coding algorithm, which is quite different from Huffman coding, is another focus of the chapter. While Huffman coding is a *block-oriented* coding technique, arithmetic coding is a *stream-oriented* coding technique. With improvements in implementation, arithmetic coding has gained increasing popularity. Both Huffman coding and arithmetic coding are included in the international still image coding standard JPEG (Joint Photographic [image] Experts Group coding). The adaptive arithmetic coding algorithms have been adopted by the international bilevel image coding standard JBIG (Joint Bi-level Image experts Group coding). Note that the material presented in this chapter can be viewed as a continuation of the information theory results presented in Chapter 1.

5.1 SOME FUNDAMENTAL RESULTS

Prior to presenting Huffman coding and arithmetic coding, we first provide some fundamental concepts and results as necessary background.

5.1.1 Coding an Information Source

Consider an information source, represented by a *source alphabet S*.

$$S = \{s_1, s_2, \cdots, s_m\} \tag{5.1}$$

where s_i, $i = 1,2,\cdots,m$ are *source symbols*. Note that the terms source symbol and information message are used interchangeably in the literature. In this book, however, we would like to distinguish between them. That is, an information message can be a source symbol, or a combination of source symbols. We denote *code alphabet* by A and

$$A = \{a_1, a_2, \cdots, a_r\} \tag{5.2}$$

where a_j, $j = 1,2,\cdots,r$ are *code symbols*. A *message code* is a sequence of code symbols that represents a given information message. In the simplest case, a message consists of only a source symbol. Encoding is then a procedure to assign a *codeword* to the source symbol. Namely,

$$s_i \rightarrow A_i = \left(a_{i1}, a_{i2}, \cdots, a_{ik}\right) \tag{5.3}$$

where the codeword A_i is a string of k code symbols assigned to the source symbol s_i. The term message ensemble is defined as the entire set of messages. A code, also known as an ensemble code, is defined as a mapping of all the possible sequences of symbols of S (message ensemble) into the sequences of symbols in A.

Note that in binary coding, the number of code symbols r is equal to 2, since there are only two code symbols available: the binary digits "0" and "1". Two examples are given below to illustrate the above concepts.

Example 5.1
Consider an English article and the ASCII code. Refer to Table 5.1. In this context, the source alphabet consists of all the English letters in both lower and upper cases and all the punctuation marks. The code alphabet consists of the binary 1 and 0. There are a total of 128 7-bit binary codewords. From Table 5.1, we see that the codeword assigned to the capital letter A is 1000001. That is, A is a source symbol, while 1000001 is its codeword.

Example 5.2
Table 5.2 lists what is known as the (5,2) code. It is a linear block code. In this example, the source alphabet consists of the four (2^2) source symbols listed in the left column of the table: 00, 01, 10, and 11. The code alphabet consists of the binary 1 and 0. There are four codewords listed in the right column of the table. From the table, we see that the code assigns a 5-bit codeword to each source symbol. Specifically, the codeword of the source symbol 00 is 00000. The source symbol 01 is encoded as 10100; 01111 is the codeword assigned to 10. The symbol 11 is mapped to 11011.

5.1.2 Some Desired Characteristics

To be practical in use, codes need to have some desirable characteristics (Abramson, 1963). Some of the characteristics are addressed in this subsection.

5.1.2.1 Block Code

A code is said to be a block code if it maps each source symbol in S into a fixed codeword in A. Hence, the codes listed in the above two examples are block codes.

5.1.2.2 Uniquely Decodable Code

A code is uniquely decodable if it can be unambiguously decoded. Obviously, a code has to be uniquely decodable if it is to be of use.

Example 5.3
Table 5.3 specifies a code. Obviously it is not uniquely decodable since if a binary string "00" is received we do not know which of the following two source symbols has been sent out: s_1 or s_3.

Nonsingular Code
A block code is nonsingular if all the codewords are distinct (see Table 5.4).

Example 5.4
Table 5.4 gives a nonsingular code since all four codewords are distinct. If a code is not a nonsingular code, i.e., at least two codewords are identical, then the code is not uniquely decodable. Notice, however, that a nonsingular code does not guarantee unique decodability. The code shown in Table 5.4 is such an example in that it is nonsingular while it is not uniquely decodable. It is not

TABLE 5.1
Seven-Bit American Standard Code for Information Interchange (ASCII)

Bits				5	0	1	0	1	0	1	0	1
				6	0	0	1	1	0	0	1	1
1	2	3	4	7	0	0	0	0	1	1	1	1
0	0	0	0	NUL		DLE	SP	0	@	P	`	p
1	0	0	0	SOH		DC1	!	1	A	Q	a	q
0	1	0	0	STX		DC2	"	2	B	R	b	r
1	1	0	0	ETX		DC3	#	3	C	S	c	s
0	0	1	0	EOT		DC4	$	4	D	T	d	t
1	0	1	0	ENQ		NAK	%	5	E	U	e	u
0	1	1	0	ACK		SYN	&	6	F	V	f	v
1	1	1	0	BEL		ETB	'	7	G	W	g	w
0	0	0	1	BS		CAN	(	8	H	X	h	x
1	0	0	1	HT		EM	)	9	I	Y	i	y
0	1	0	1	LF		SUB	*	:	J	Z	j	z
1	1	0	1	VT		ESC	+	;	K	[	k	{
0	0	1	1	FF		FS	,	<	L	\	l	|
1	0	1	1	CR		GS	-	=	M	]	m	}
0	1	1	1	SO		RS	.	>	N	^	n	~
1	1	1	1	SI		US	/	?	O	—	o	DEL

NUL	Null, or all zeros	DC1	Device control 1
SOH	Start of heading	DC2	Device control 2
STX	Start of text	DC3	Device control 3
ETX	End of text	DC4	Device control 4
EOT	End of transmission	NAK	Negative acknowledgment
ENQ	Enquiry	SYN	Synchronous idle
ACK	Acknowledge	ETB	End of transmission block
BEL	Bell, or alarm	CAN	Cancel
BS	Backspace	EM	End of medium
HT	Horizontal tabulation	SUB	Substitution
LF	Line feed	ESC	Escape
VT	Vertical tabulation	FS	File separator
FF	Form feed	GS	Group separator
CR	Carriage return	RS	Record separator
SO	Shift out	US	Unit separator
SI	Shift in	SP	Space
DLE	Data link escape	DEL	Delete

TABLE 5.2
A (5,2) Linear Block Code

Source Symbol	Codeword
S_1 (0 0)	0 0 0 0 0
S_2 (0 1)	1 0 1 0 0
S_3 (1 0)	0 1 1 1 1
S_4 (1 1)	1 1 0 1 1

TABLE 5.3
A Not Uniquely Decodable Code

Source Symbol	Codeword
S_1	0 0
S_2	1 0
S_3	0 0
S_4	1 1

TABLE 5.4
A Nonsingular Code

Source Symbol	Codeword
S_1	1
S_2	1 1
S_3	0 0
S_4	0 1

uniquely decodable because once the binary string "11" is received, we do not know if the source symbols transmitted are s_1 followed by s_1 or simply s_2.

The *n*th Extension of a Block Code
The nth extension of a block code, which maps the source symbol s_i into the codeword A_i, is a block code that maps the sequences of source symbols $s_{i1}s_{i2}\cdots s_{in}$ into the sequences of codewords $A_{i1}A_{i2}\cdots A_{in}$.

A Necessary and Sufficient Condition of a Block Code's Unique Decodability
A block code is uniquely decodable if and only if the nth extension of the code is nonsingular for every finite n.

Example 5.5
The second extension of the nonsingular block code shown in Example 5.4 is listed in Table 5.5. Clearly, this second extension of the code is not a nonsingular code, since the entries s_1s_2 and s_2s_1 are the same. This confirms the nonunique decodability of the nonsingular code in Example 5.4.

TABLE 5.5
The Second Extension of the Nonsingular Block Code in Example 5.4

Source Symbol	Codeword	Source Symbol	Codeword
$S_1\,S_1$	1 1	$S_3\,S_1$	0 0 1
$S_1\,S_2$	1 1 1	$S_3\,S_2$	0 0 1 1
$S_1\,S_3$	1 0 0	$S_3\,S_3$	0 0 0 0
$S_1\,S_4$	1 0 1	$S_3\,S_4$	0 0 0 1
$S_2\,S_1$	1 1 1	$S_4\,S_1$	0 1 1
$S_2\,S_2$	1 1 1 1	$S_4\,S_2$	0 1 1 1
$S_2\,S_3$	1 1 0 0	$S_4\,S_3$	0 1 0 0
$S_2\,S_4$	1 1 0 1	$S_4\,S_4$	0 1 0 1

TABLE 5.6
Three Uniquely Decodable Codes

Source Symbol	Code 1	Code 2	Code 3
S_1	0 0	1	1
S_2	0 1	0 1	10
S_3	1 0	0 0 1	1 0 0
S_4	1 1	0 0 0 1	1 0 0 0

5.1.2.3 Instantaneous Codes

Definition of Instantaneous Codes
A uniquely decodable code is said to be instantaneous if it is possible to decode each codeword in a code symbol sequence without knowing the succeeding codewords.

Example 5.6
Table 5.6 lists three uniquely decodable codes. The first one is in fact a two-bit natural binary code. In decoding, we can immediately tell which source symbols are transmitted since each codeword has the same length. In the second code, code symbol "1" functions like a comma. Whenever we see a "1", we know it is the end of the codeword. The third code is different from the previous two codes in that if we see a "10" string we are not sure if it corresponds to s_2 until we see a succeeding "1". Specifically, if the next code symbol is "0", we still cannot tell if it is s_3 since the next one may be "0" (hence s_4) or "1" (hence s_3). In this example, the next "1" belongs to the succeeding codeword. Therefore we see that code 3 is uniquely decodable. It is not instantaneous, however.

Definition of the *j*th Prefix
Assume a codeword $A_i = a_{i1}a_{i2}\cdots a_{ik}$. Then the sequences of code symbols $a_{i1}a_{i2}\cdots a_{ij}$ with $1 \le j \le k$ is the jth order prefix of the codeword A_i.

Example 5.7
If a codeword is 11001, it has the following five prefixes: 11001, 1100, 110, 11, 1. The first-order prefix is 1, while the fifth-order prefix is 11001.

A Necessary and Sufficient Condition of Being an Instantaneous Code
A code is instantaneous if and only if no codeword is a prefix of some other codeword. This condition is often referred to as the *prefix condition*. Hence, the instantaneous code is also called the prefix condition code or sometimes simply the prefix code. In many applications, we need a block code that is nonsingular, uniquely decodable, and instantaneous.

5.1.2.4 Compact Code

A uniquely decodable code is said to be compact if its average length is the minimum among all other uniquely decodable codes based on the same source alphabet S and code alphabet A. A compact code is also referred to as a *minimum redundancy* code, or an *optimum* code.

Note that the average length of a code was defined in Chapter 1 and is restated below.

5.1.3 Discrete Memoryless Sources

This is the simplest model of an information source. In this model, the symbols generated by the source are independent of each other. That is, the source is memoryless or it has a zero memory.

Consider the information source expressed in Equation 5.1 as a discrete memoryless source. The occurrence probabilities of the source symbols can be denoted by $p(s_1), p(s_2), \cdots, p(s_m)$. The

lengths of the codewords can be denoted by $l_1, l_2, \cdots, l_m$. The average length of the code is then equal to

$$L_{avg} = \sum_{i=1}^{m} l_i p(s_i) \tag{5.4}$$

Recall Shannon's first theorem, i.e., the noiseless coding theorem described in Chapter 1. The average length of the code is bounded below by the entropy of the information source. The entropy of the source S is defined as $H(S)$ and

$$H(S) = -\sum_{i=1}^{m} p(s_i) \log_2 p(s_i) \tag{5.5}$$

Recall that entropy is the average amount of information contained in a source symbol. In Chapter 1 the efficiency of a code, η, is defined as the ratio between the entropy and the average length of the code. That is, $\eta = H(S)/L_{avg}$. The redundancy of the code, ζ, is defined as $\zeta = 1 - \eta$.

5.1.4 Extensions of a Discrete Memoryless Source

Instead of coding each source symbol in a discrete source alphabet, it is often useful to code blocks of symbols. It is, therefore, necessary to define the nth extension of a discrete memoryless source.

5.1.4.1 Definition

Consider the zero-memory source alphabet S defined in Equation 5.1. That is, $S = \{s_1, s_2, \cdots, s_m\}$. If n symbols are grouped into a block, then there is a total of m^n blocks. Each block is considered as a new source symbol. These m^n blocks thus form an information source alphabet, called the nth extension of the source S, which is denoted by S^n.

5.1.4.2 Entropy

Let each block be denoted by β_i and

$$\beta_i = (s_{i1}, s_{i2}, \cdots, s_{in}) \tag{5.6}$$

Then we have the following relation due to the memoryless assumption:

$$p(\beta_i) = \prod_{j=1}^{n} p(s_{ij}) \tag{5.7}$$

Hence, the relationship between the entropy of the source S and the entropy of its nth extension is as follows:

$$H(S^n) = n \cdot H(S) \tag{5.8}$$

Example 5.8
Table 5.7 lists a source alphabet. Its second extension is listed in Table 5.8.

TABLE 5.7
A Discrete Memoryless Source Alphabet

Source Symbol	Occurrence Probability
S_1	0.6
S_2	0.4

TABLE 5.8
The Second Extension of the Source Alphabet Shown in Table 5.7

Source Symbol	Occurrence Probability
$S_1 S_1$	0.36
$S_2 S_2$	0.24
$S_2 S_1$	0.24
$S_2 S_2$	0.16

The entropy of the source and its second extension are calculated below.

$$H(S) = -0.6 \cdot \log_2(0.6) - 0.4 \cdot \log_2(0.4) \approx 0.97$$

$$H(S^2) = -0.36 \cdot \log_2(0.36) - 2 \cdot 0.24 \cdot \log_2(0.24) - 0.16 \cdot \log_2(0.16) \approx 1.94$$

It is seen that $H(S^2) = 2H(S)$.

5.1.4.3 Noiseless Source Coding Theorem

The noiseless source coding theorem, also known as Shannon's first theorem, defining the minimum average codeword length per source pixel, was presented in Chapter 1, but without a mathematical expression. Here, we provide some mathematical expressions in order to give more insight about the theorem.

For a discrete zero-memory information source S, the noiseless coding theorem can be expressed as

$$H(S) \le L_{avg} < H(S) + 1 \tag{5.9}$$

That is, there exists a variable-length code whose average length is bounded below by the entropy of the source (that is encoded) and bounded above by the entropy plus 1. Since the nth extension of the source alphabet, S^n, is itself a discrete memoryless source, we can apply the above result to it. That is,

$$H(S^n) \le L^n_{avg} < H(S^n) + 1 \tag{5.10}$$

where L^n_{avg} is the average codeword length of a variable-length code for the S^n. Since $H(S^n) = nH(S)$ and $L^n_{avg} = nL^n$avg, we have

$$H(S) \le L_{avg} < H(S) + \frac{1}{n} \tag{5.11}$$

Therefore, when coding blocks of n source symbols, the noiseless source coding theory states that for an arbitrary positive number ε, there is a variable-length code which satisfies the following:

$$H(S) \le L_{avg} < H(S) + \varepsilon \tag{5.12}$$

as n is large enough. That is, the average number of bits used in coding per source symbol is bounded below by the entropy of the source and is bounded above by the sum of the entropy and an arbitrary positive number. To make ε arbitrarily small, i.e., to make the average length of the code arbitrarily close to the entropy, we have to make the block size n large enough. This version of the noiseless coding theorem suggests a way to make the average length of a variable-length code approach the source entropy. It is known, however, that the high coding complexity that occurs when n approaches infinity makes implementation of the code impractical.

5.2 HUFFMAN CODES

Consider the source alphabet defined in Equation 5.1. The method of encoding source symbols according to their probabilities, suggested in (Shannon, 1948; Fano, 1949), is not optimum. It approaches the optimum, however, when the block size n approaches infinity. This results in a large storage requirement and high computational complexity. In many cases, we need a direct encoding method that is optimum and instantaneous (hence uniquely decodable) for an information source with finite source symbols in source alphabet S. Huffman code is the first such optimum code (Huffman, 1952), and is the technique most frequently used at present. It can be used for r-ary encoding as $r > 2$. For notational brevity, however, we discuss only the Huffman coding used in the binary case presented here.

5.2.1 Required Rules for Optimum Instantaneous Codes

Let us rewrite Equation 5.1 as follows:

$$S = (s_1, s_2, \cdots, s_m) \tag{5.13}$$

Without loss of generality, assume the occurrence probabilities of the source symbols are as follows:

$$p(s_1) \ge p(s_2) \ge \cdots \ge p(s_{m-1}) \ge p(s_m) \tag{5.14}$$

Since we are seeking the optimum code for S, the lengths of codewords assigned to the source symbols should be

$$l_1 \le l_2 \le \cdots \le l_{m-1} \le l_m . \tag{5.15}$$

Based on the requirements of the optimum and instantaneous code, Huffman derived the following rules (restrictions):

1. $l_1 \le l_2 \le \cdots \le l_{m-1} = l_m$ (5.16)

   Equations 5.14 and 5.16 imply that when the source symbol occurrence probabilities are arranged in a nonincreasing order, the length of the corresponding codewords should be in a nondecreasing order. In other words, the codeword length of a more probable source

symbol should not be longer than that of a less probable source symbol. Furthermore, the length of the codewords assigned to the two least probable source symbols should be the same.
2. The codewords of the two least probable source symbols should be the same except for their last bits.
3. Each possible sequence of length $l_m - 1$ bits must be used either as a codeword or must have one of its prefixes used as a codeword.

Rule 1 can be justified as follows. If the first part of the rule, i.e., $l_1 \leq l_2 \leq \cdots \leq l_{m-1}$ is violated, say, $l_1 > l_2$, then we can exchange the two codewords to shorten the average length of the code. This means the code is not optimum, which contradicts the assumption that the code is optimum. Hence it is impossible. That is, the first part of Rule 1 has to be the case. Now assume that the second part of the rule is violated, i.e., $l_{m-1} < l_m$. (Note that $l_{m-1} > l_m$ can be shown to be impossible by using the same reasoning we just used in proving the first part of the rule.) Since the code is instantaneous, codeword A_{m-1} is not a prefix of codeword A_m. This implies that the last bit in the codeword A_m is redundant. It can be removed to reduce the average length of the code, implying that the code is not optimum. This contradicts the assumption, thus proving Rule 1.

Rule 2 can be justified as follows. As in the above, A_{m-1} and A_m are the codewords of the two least probable source symbols. Assume that they do not have the identical prefix of the order $l_m - 1$. Since the code is optimum and instantaneous, codewords A_{m-1} and A_m cannot have prefixes of any order that are identical to other codewords. This implies that we can drop the last bits of A_{m-1} and A_m to achieve a lower average length. This contradicts the optimum code assumption. It proves that Rule 2 has to be the case.

Rule 3 can be justified using a similar strategy to that used above. If a possible sequence of length $l_m - 1$ has not been used as a codeword and any of its prefixes have not been used as codewords, then it can be used in place of the codeword of the mth source symbol, resulting in a reduction of the average length L_{avg}. This is a contradiction to the optimum code assumption and it justifies the rule.

5.2.2 Huffman Coding Algorithm

Based on these three rules, we see that the two least probable source symbols have codewords of equal length. These two codewords are identical except for the last bits, the binary 0 and 1, respectively. Therefore, these two source symbols can be combined to form a single new symbol. Its occurrence probability is the sum of two source symbols, i.e., $p(s_{m-1}) + p(s_m)$. Its codeword is the common prefix of order $l_m - 1$ of the two codewords assigned to s_m and s_{m-1}, respectively. The new set of source symbols thus generated is referred to as the first auxiliary source alphabet, which is one source symbol less than the original source alphabet. In the first auxiliary source alphabet, we can rearrange the source symbols according to a nonincreasing order of their occurrence probabilities. The same procedure can be applied to this newly created source alphabet. A binary 0 and a binary 1, respectively, are assigned to the last bits of the two least probable source symbols in the alphabet. The second auxiliary source alphabet will again have one source symbol less than the first auxiliary source alphabet. The procedure continues. In some step, the resultant source alphabet will have only two source symbols. At this time, we combine them to form a single source symbol with a probability of 1. The coding is then complete.

Let's go through the following example to illustrate the above Huffman algorithm.

Example 5.9

Consider a source alphabet whose six source symbols and their occurrence probabilities are listed in Table 5.9. Figure 5.1 demonstrates the Huffman coding procedure applied. In the example, among the two least probable source symbols encountered at each step we assign binary 0 to the top symbol and binary 1 to the bottom symbol.

TABLE 5.9
Source Alphabet and Huffman Codes in Example 5.9

Source Symbol	Occurrence Probability	Codeword Assigned	Length of Codeword
S_1	0.3	00	2
S_2	0.1	101	3
S_3	0.2	11	2
S_4	0.05	1001	4
S_5	0.1	1000	4
S_6	0.25	01	2

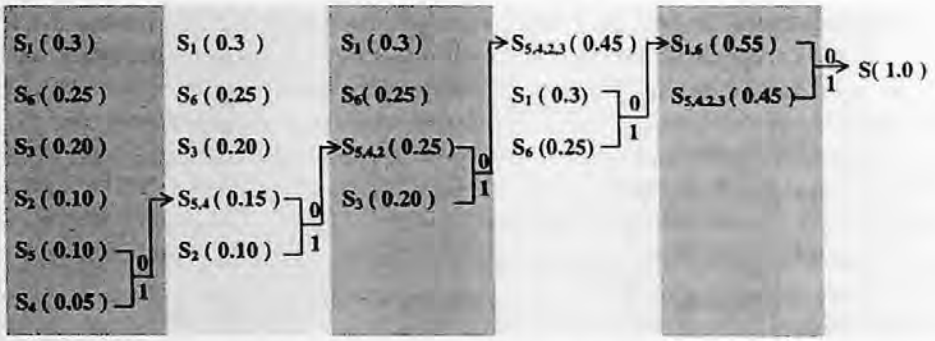


FIGURE 5.1 Huffman coding procedure in Example 5.9.

5.2.2.1 Procedures

In summary, the Huffman coding algorithm consists of the following steps.

1. Arrange all source symbols in such a way that their occurrence probabilities are in a nonincreasing order.
2. Combine the two least probable source symbols:
   - Form a new source symbol with a probability equal to the sum of the probabilities of the two least probable symbols.
   - Assign a binary 0 and a binary 1 to the two least probable symbols.
3. Repeat until the newly created auxiliary source alphabet contains only one source symbol.
4. Start from the source symbol in the last auxiliary source alphabet and trace back to each source symbol in the original source alphabet to find the corresponding codewords.

5.2.2.2 Comments

First, it is noted that the assignment of the binary 0 and 1 to the two least probable source symbols in the original source alphabet and each of the first $(u - 1)$ auxiliary source alphabets can be implemented in two different ways. Here u denotes the total number of the auxiliary source symbols in the procedure. Hence, there is a total of 2^u possible Huffman codes. In Example 5.9, there are five auxiliary source alphabets, hence a total of $2^5 = 32$ different codes. Note that each is optimum: that is, each has the same average length.

Second, in sorting the source symbols, there may be more than one symbol having equal probabilities. This results in multiple arrangements of symbols, hence multiple Huffman codes. While all of these Huffman codes are optimum, they may have some other different properties.

For instance, some Huffman codes result in the minimum codeword length variance (Sayood, 1996). This property is desired for applications in which a constant bit rate is required.

Third, Huffman coding can be applied to r-ary encoding with $r > 2$. That is, code symbols are r-ary with $r > 2$.

5.2.2.3 Applications

As a systematic procedure to encode a finite discrete memoryless source, the Huffman code has found wide application in image and video coding. Recall that it has been used in differential coding and transform coding. In transform coding, as introduced in Chapter 4, the magnitude of the quantized transform coefficients and the run-length of zeros in the zigzag scan are encoded by using the Huffman code. This has been adopted by both still image and video coding standards.

5.3 MODIFIED HUFFMAN CODES

5.3.1 Motivation

As a result of Huffman coding, a set of all the codewords, called a codebook, is created. It is an agreement between the transmitter and the receiver. Consider the case where the occurrence probabilities are skewed, i.e., some are large, while some are small. Under these circumstances, the improbable source symbols take a disproportionately large amount of memory space in the codebook. The size of the codebook will be very large if the number of the improbable source symbols is large. A large codebook requires a large memory space and increases the computational complexity. A modified Huffman procedure was therefore devised in order to reduce the memory requirement while keeping almost the same optimality (Hankamer, 1979).

Example 5.10

Consider a source alphabet consisting of 16 symbols, each being a 4-bit binary sequence. That is, $S = \{s_i, i = 1,2,\cdots,16\}$. The occurrence probabilities are

$$p(s_1) = p(s_2) = 1/4,$$

$$p(s_3) = p(s_4) = \cdots = p(s_{16}) = 1/28.$$

The source entropy can be calculated as follows:

$$H(S) = 2 \cdot \left(-\frac{1}{4}\log_2\frac{1}{4}\right) + 14 \cdot \left(-\frac{1}{28}\log_2\frac{1}{28}\right) \approx 3.404 \quad \textit{bits per symbol}$$

Applying the Huffman coding algorithm, we find that the codeword lengths associated with the symbols are: $l_1 = l_2 = 2$, $l_3 = 4$, and $l_4 = l_5 = \cdots = l_{16} = 5$, where l_i denotes the length of the ith codeword. The average length of Huffman code is

$$L_{avg} = \sum_{i=1}^{16} p(s_i) l_i = 3.464 \quad \textit{bits per symbol}$$

We see that the average length of Huffman code is quite close to the lower entropy bound. It is noted, however, that the required codebook memory, M (defined as the sum of the codeword lengths), is quite large:

$$M = \sum_{i=1}^{16} l_i = 73 \quad bits$$

This number is obviously larger than the average codeword length multiplied by the number of codewords. This should not come as a surprise since the average here is in the statistical sense instead of in the arithmetic sense. When the total number of improbable symbols increases, the required codebook memory space will increase dramatically, resulting in a great demand on memory space.

5.3.2 Algorithm

Consider a source alphabet S that consists of 2^v binary sequences, each of length v. In other words, each source symbol is a v-bit codeword in the natural binary code. The occurrence probabilities are highly skewed and there is a large number of improbable symbols in S. The modified Huffman coding algorithm is based on the following idea: lumping all the improbable source symbols into a category named ELSE (Weaver, 1978). The algorithm is described below.

1. Categorize the source alphabet S into two disjoint groups, S_1 and S_2, such that

$$S_1 = \left\{ s_i \middle| p(s_i) > \frac{1}{2^v} \right\} \tag{5.17}$$

and

$$S_2 = \left\{ s_i \middle| p(s_i) \le \frac{1}{2^v} \right\} \tag{5.18}$$

2. Establish a source symbol ELSE with its occurrence probability equal to $p(S_2)$.
3. Apply the Huffman coding algorithm to the source alphabet S_3 with $S_3 = S_1 \cup ELSE$.
4. Convert the codebook of S_3 to that of S as follows.
   - Keep the same codewords for those symbols in S_1.
   - Use the codeword assigned to ELSE as a prefix for those symbols in S_2.

5.3.3 Codebook Memory Requirement

Codebook memory M is the sum of the codeword lengths. The M required by Huffman coding with respect to the original source alphabet S is

$$M = \sum_{i \in S} l_i = \sum_{i \in S_1} l_i + \sum_{i \in S_2} l_i \tag{5.19}$$

where l_i denotes the length of the ith codeword, as defined previously. In the case of the modified Huffman coding algorithm, the memory required M_{mH} is

$$M_{mH} = \sum_{i \in S_3} l_i = \sum_{i \in S_1} l_i + l_{ELSE} \tag{5.20}$$

where l_{ELSE} is the length of the codeword assigned to ELSE. The above equation reveals the big savings in memory requirement when the probability is skewed. The following example is used to illustrate the modified Huffman coding algorithm and the resulting dramatic memory savings.

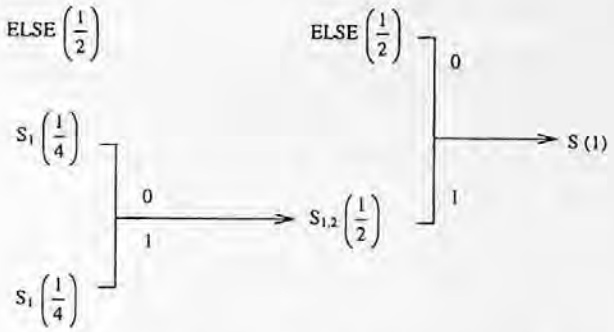


FIGURE 5.2 The modified Huffman coding procedure in Example 5.11.

Example 5.11

In this example, we apply the modified Huffman coding algorithm to the source alphabet presented in Example 5.10. We first lump the 14 symbols having the least occurrence probabilities together to form a new symbol ELSE. The probability of ELSE is the sum of the 14 probabilities. That is, $p(ELSE) = \frac{1}{28} \cdot 14 = \frac{1}{2}$.

Apply Huffman coding to the new source alphabet $S_3 = \{s_1, s_2, ELSE\}$, as shown in Figure 5.2. From Figure 5.2, it is seen that the codewords assigned to symbols s_1, s_2, and *ELSE*, respectively, are 10, 11, and 0. Hence, for every source symbol lumped into ELSE, its codeword is 0 followed by the original 4-bit binary sequence. Therefore, $M_{mH} = 2 + 2 + 1 = 5$ *bits*, i.e., the required codebook memory is only 5 bits. Compared with 73 bits required by Huffman coding (refer to Example 5.10), there is a savings of 68 bits in codebook memory space. Similar to the comment made in Example 5.10, the memory savings will be even larger if the probability distribution is skewed more severely and the number of improbable symbols is larger. The average length of the modified Huffman algorithm is $L_{avg,mH} = \frac{1}{4} \cdot 2 \cdot 2 + \frac{1}{28} \cdot 5 \cdot 14 = 3.5$ *bits per symbol*. This demonstrates that modified Huffman coding retains almost the same coding efficiency as that achieved by Huffman coding.

5.3.4 Bounds on Average Codeword Length

It has been shown that the average length of the modified Huffman codes satisfies the following condition:

$$H(S) \le L_{avg} < H(S) + 1 - p \log_2 p \tag{5.21}$$

where $p = \Sigma_{s_i \in s_2} p(s_i)$. It is seen that, compared with the noiseless source coding theorem, the upper bound of the code average length is increased by a quantity of $-p \log_2 p$. In Example 5.11 it is seen that the average length of the modified Huffman code is close to that achieved by the Huffman code. Hence the modified Huffman code is almost optimum.

5.4 ARITHMETIC CODES

Arithmetic coding, which is quite different from Huffman coding, is gaining increasing popularity. In this section, we will first analyze the limitations of Huffman coding. Then the principle of arithmetic coding will be introduced. Finally some implementation issues are discussed briefly.

5.4.1 Limitations of Huffman Coding

As seen in Section 5.2, Huffman coding is a systematic procedure for encoding a source alphabet, with each source symbol having an occurrence probability. Under these circumstances, Huffman coding is optimum in the sense that it produces a minimum coding redundancy. It has been shown that the average codeword length achieved by Huffman coding satisfies the following inequality (Gallagher, 1978).

$$H(S) \leq L_{avg} < H(S) + p_{\max} + 0.086 \tag{5.22}$$

where $H(S)$ is the entropy of the source alphabet, and $p_{\max}$ denotes the maximum occurrence probability in the set of the source symbols. This inequality implies that the upper bound of the average codeword length of Huffman code is determined by the entropy and the maximum occurrence probability of the source symbols being encoded.

In the case where the probability distribution among source symbols is skewed (some probabilities are small, while some are quite large), the upper bound may be large, implying that the coding redundancy may not be small. Imagine the following extreme situation. There are only two source symbols. One has a very small probability, while the other has a very large probability (very close to 1). The entropy of the source alphabet in this case is close to 0 since the uncertainty is very small. Using Huffman coding, however, we need two bits: one for each. That is, the average codeword length is 1, which means that the redundancy is very close to 1. This agrees with Equation 5.22. This inefficiency is due to the fact that Huffman coding always encodes a source symbol with an integer number of bits.

The noiseless coding theorem (reviewed in Section 5.1) indicates that the average codeword length of a block code can approach the source alphabet entropy when the block size approaches infinity. As the block size approaches infinity, the storage required, the codebook size, and the coding delay will approach infinity, however, and the complexity of the coding will be out of control.

The fundamental idea behind Huffman coding and Shannon-Fano coding (devised a little earlier than Huffman coding [Bell et al., 1990]) is block coding. That is, some codeword having an integral number of bits is assigned to a source symbol. A message may be encoded by cascading the relevant codewords. It is the *block-based* approach that is responsible for the limitations of Huffman codes.

Another limitation is that when encoding a message that consists of a sequence of source symbols, the nth extension Huffman coding needs to enumerate all possible sequences of source symbols having the same length, as discussed in coding the nth extended source alphabet. This is not computationally efficient.

Quite different from Huffman coding, arithmetic coding is *stream-based*. It overcomes the drawbacks of Huffman coding. A string of source symbols is encoded as a string of code symbols. Hence, it is free of the integral-bits-per-source symbol restriction and is more efficient. Arithmetic coding may reach the theoretical bound to coding efficiency specified in the noiseless source coding theorem for any information source. Below, we introduce the principle of arithmetic coding, from which we can see the stream-based nature of arithmetic coding.

5.4.2 Principle of Arithmetic Coding

To understand the different natures of Huffman coding and arithmetic coding, let us look at Example 5.12, where we use the same source alphabet and the associated occurrence probabilities used in Example 5.9. In this example, however, a string of source symbols $s_1 s_2 s_3 s_4 s_5 s_6$ is encoded. Note that we consider the terms *string* and *stream* to be slightly different. By stream, we mean a message, or possibly several messages, which may correspond to quite a long sequence of source symbols. Moreover, stream gives a dynamic "flavor." Later on we will see that arithmetic coding

TABLE 5.10
Source Alphabet and Cumulative Probabilities in Example 5.12

Source Symbol	Occurrence Probability	Associated Subintervals	CP
S_1	0.3	[0, 0.3)	0
S_2	0.1	[0.3, 0.4)	0.3
S_3	0.2	[0.4, 0.6)	0.4
S_4	0.05	[0.6, 0.65)	0.6
S_5	0.1	[0.65, 0.75)	0.65
S_6	0.25	[0.75, 1.0)	0.75

is implemented in an incremental manner. Hence stream is a suitable term to use for arithmetic coding. In this example, however, only six source symbols are involved. Hence we consider the term *string* to be suitable, aiming at distinguishing it from the term *block*.

Example 5.12
The set of six source symbols and their occurrence probabilities are listed in Table 5.10. In this example, the string to be encoded using arithmetic coding is $s_1s_2s_3s_4s_5s_6$. In the following four subsections we will use this example to illustrate the principle of arithmetic coding and decoding.

5.4.2.1 Dividing Interval [0,1) into Subintervals

As pointed out by Elias, it is not necessary to sort out source symbols according to their occurrence probabilities. Therefore in Figure 5.3(a) the six symbols are arranged in their natural order, from symbols s_1, s_2, $\cdots$, up to s_6. The real interval between 0 and 1 is divided into six subintervals, each having a length of $p(s_i)$, $i = 1,2,\cdots,6$. Specifically, the interval denoted by [0,1) — where 0 is included in (the left end is closed) and 1 is excluded from (the right end is open) the interval — is divided into six subintervals. The first subinterval [0, 0.3) corresponds to s_1 and has a length of $P(s_1)$, i.e., 0.3. Similarly, the subinterval [0, 0.3) is said to be closed on the left and open on the right. The remaining five subintervals are similarly constructed. All six subintervals thus formed are disjoint and their union is equal to the interval [0, 1). This is because the sum of all the probabilities is equal to 1.

We list the sum of the preceding probabilities, known as *cumulative probability* (Langdon, 1984), in the right-most column of Table 5.10 as well. Note that the concept of cumulative probability (CP) is slightly different from that of cumulative distribution function (CDF) in probability theory. Recall that in the case of discrete random variables the CDF is defined as follows.

$$CDF(s_i) = \sum_{j=1}^{i} p(s_j) \tag{5.23}$$

The CP is defined as

$$CP(s_i) = \sum_{j=1}^{i-1} p(s_j) \tag{5.24}$$

where $CP(s_1) = 0$ is defined. Now we see each subinterval has its lower end point located at $CP(s_i)$. The width of each subinterval is equal to the probability of the corresponding source symbol. A subinterval can be completely defined by its lower end point and its width. Alternatively, it is

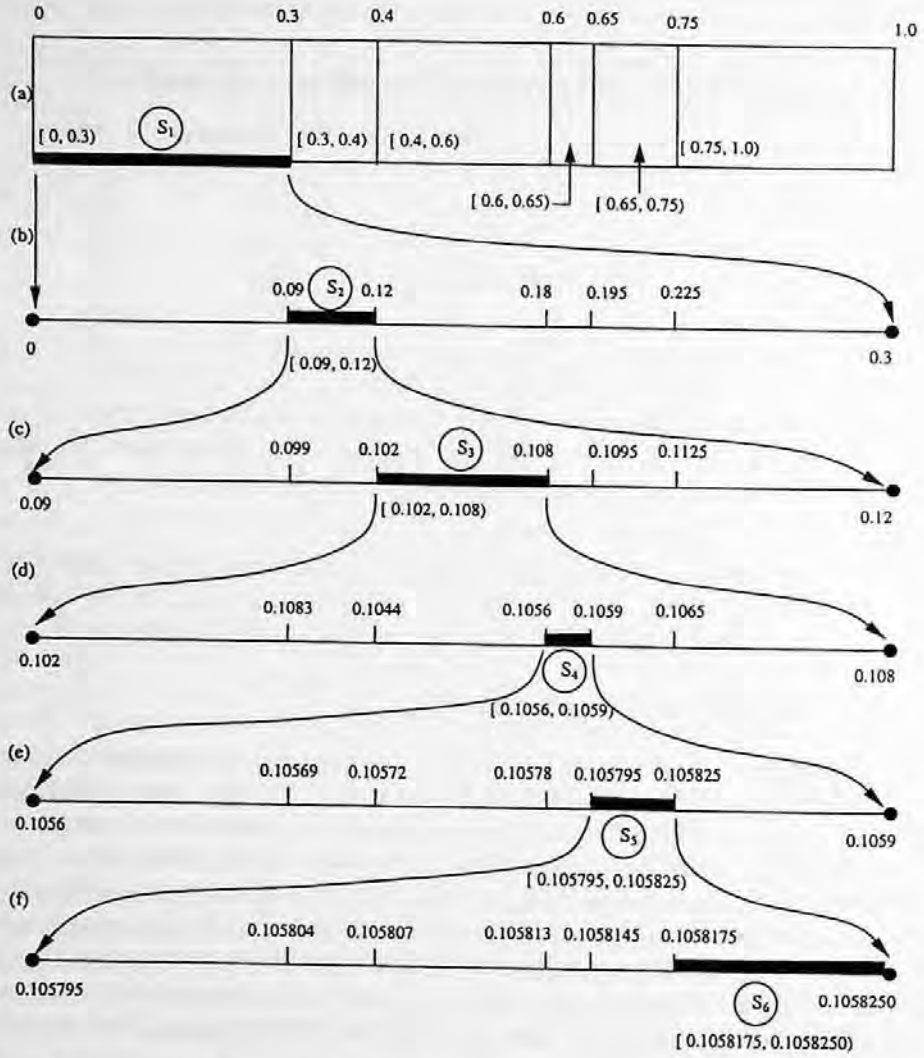


FIGURE 5.3 Arithmetic coding working on the same source alphabet as that in Example 5.9. The encoded symbol string is $S_1\ S_2\ S_3\ S_4\ S_5\ S_6$.

determined by its two end points: the lower and upper end points (sometimes also called the left and right end points).

Now we consider encoding the string of source symbols $s_1s_2s_3s_4s_5s_6$ with the arithmetic coding method.

5.4.2.2 Encoding

Encoding the First Source Symbol

Refer to Figure 5.3(a). Since the first symbol is s_1, we pick up its subinterval [0, 0.3). Picking up the subinterval [0, 0.3) means that any real number in the subinterval, i.e., any real number equal to or greater than 0 and smaller than 0.3, can be a pointer to the subinterval, thus representing the source symbol s_1. This can be justified by considering that all the six subintervals are disjoint.

Encoding the Second Source Symbol

Refer to Figure 5.3(b). We use the same procedure as used in part (a) to divide the interval [0, 0.3) into six subintervals. Since the second symbol to be encoded is s_2, we pick up its subinterval [0.09, 0.12).

Notice that the subintervals are recursively generated from part (a) to part (b). It is known that an interval may be completely specified by its lower end point and width. Hence, the subinterval recursion in the arithmetic coding procedure is equivalent to the following two recursions: end point recursion and width recursion.

From interval [0, 0.3) derived in part (a) to interval [0.09, 0.12) obtained in part (b), we can conclude the following lower end point recursion:

$$L_{new} = L_{current} + W_{current} \cdot CP_{new} \tag{5.25}$$

where L_{new}, $L_{current}$ represent, respectively, the lower end points of the new and current recursions, and the $W_{current}$ and the CP_{new} denote, respectively, the width of the interval in the current recursion and the cumulative probability in the new recursion. The width recursion is

$$W_{new} = W_{current} \cdot p(s_i) \tag{5.26}$$

where W_{new}, and $p(s_i)$ are, respectively, the width of the new subinterval and the probability of the source symbol s_i that is being encoded. These two recursions, also called double recursion (Langdon, 1984), play a central role in arithmetic coding.

Encoding the Third Source Symbol

Refer to Figure 5.3(c). When the third source symbol is encoded, the subinterval generated above in part (b) is similarly divided into six subintervals. Since the third symbol to encode is s_3, its subinterval [0.102, 0.108) is picked up.

Encoding the Fourth, Fifth, and Sixth Source Symbols

Refer to Figure 5.3(d,e,f). The subinterval division is carried out according to Equations 5.25 and 5.26. The symbols s_4, s_5, and s_6 are encoded. The final subinterval generated is [0.1058175, 0.1058250).

That is, the resulting subinterval [0.1058175, 0.1058250) can represent the source symbol string $s_1s_2s_3s_4s_5s_6$. Note that in this example decimal digits instead of binary digits are used. In binary arithmetic coding, the binary digits 0 and 1 are used.

5.4.2.3 Decoding

As seen in this example, for the encoder of arithmetic coding, the input is a source symbol string and the output is a subinterval. Let us call this the final subinterval or the resultant subinterval. Theoretically, any real numbers in the interval can be the code string for the input symbol string since all subintervals are disjoint. Often, however, the lower end of the final subinterval is used as the code string. Now let us examine how the decoding process is carried out with the lower end of the final subinterval.

Decoding sort of reverses what encoding has done. The decoder knows the encoding procedure and therefore has the information contained in Figure 5.3(a). It compares the lower end point of the final subinterval 0.1058175 with all the end points in (a). It is determined that $0 < 0.1058175 < 0.3$. That is, the lower end falls into the subinterval associated with the symbol s_1. Therefore, the symbol s_1 is first decoded.

Once the first symbol is decoded, the decoder may know the partition of subintervals shown in Figure 5.3(b). It is then determined that $0.09 < 0.1058175 < 0.12$. That is, the lower end is contained in the subinterval corresponding to the symbol s_2. As a result, s_2 is the second decoded symbol.

The procedure repeats itself until all six symbols are decoded. That is, based on Figure 5.3(c), it is found that $0.102 < 0.1058175 < 0.108$. The symbol s_3 is decoded. Then, the symbols s_4, s_5, s_6 are subsequently decoded because the following inequalities are determined:

$$0.1056 < 0.1058175 < 0.1059$$

$$0.105795 < 0.1058175 < 0.1058250$$

$$0.1058145 < 0.1058175 < 0.1058250$$

Note that a terminal symbol is necessary to inform the decoder to stop decoding.

The above procedure gives us an idea of how decoding works. The decoding process, however, does not need to construct parts (b), (c), (d), (e), and (f) of Figure 5.3. Instead, the decoder only needs the information contained in Figure 5.3(a). Decoding can be split into the following three steps: *comparison*, *readjustment* (subtraction), and *scaling* (Langdon, 1984).

As described above, through comparison we decode the first symbol s_1. From the way Figure 5.3(b) is constructed, we know the decoding of s_2 can be accomplished as follows. We subtract the lower end of the subinterval associated with s_1 in part (a), that is, 0 in this example, from the lower end of the final subinterval 0.1058175, resulting in 0.1058175. Then we divide this number by the width of the subinterval associated with s_1, i.e., the probability of s_1, 0.3, resulting in 0.352725. Looking at part (a) of Figure 5.3, it is found that $0.3 < 0.352725 < 0.4$. That is, the number is within the subinterval corresponding to s_2. Therefore the second decoded symbol is s_2. Note that these three decoding steps exactly "undo" what encoding did.

To decode the third symbol, we subtract the lower end of the subinterval with s_2, 0.3 from 0.352725, obtaining 0.052725. This number is divided by the probability of s_2, 0.1, resulting in 0.52725. The comparison of 0.52725 with end points in part (a) reveals that the third decoded symbol is s_3.

In decoding the fourth symbol, we first subtract the lower end of the s_3's subinterval in part (a), 0.4 from 0.52725, getting 0.12725. Dividing 0.12725 by the probability of s_3, 0.2, results in 0.63625. Referring to part (a), we decode the fourth symbol as s_4 by comparison.

Subtraction of the lower end of the subinterval of s_4 in part (a), 0.6, from 0.63625 leads to 0.03625. Division of 0.03625 by the probability of s_4, 0.05, produces 0.725. The comparison between 0.725 and the end points in part (a) decodes the fifth symbol as s_5.

Subtracting 0.725 by the lower end of the subinterval associated with s_5 in part (a), 0.65, gives 0.075. Dividing 0.075 by the probability of s_5, 0.1, generates 0.75. The comparison indicates that the sixth decoded symbol is s_6.

In summary, considering the way in which parts (b), (c), (d), (e), and (f) of Figure 5.3 are constructed, we see that the three steps discussed in the decoding process: comparison, readjustment, and scaling, exactly "undo" what the encoding procedure has done.

5.4.2.4 Observations

Both encoding and decoding involve only arithmetic operations (addition and multiplication in encoding, subtraction and division in decoding). This explains the name *arithmetic coding*.

We see that an input source symbol string $s_1s_2s_3s_4s_5s_6$, via encoding, corresponds to a subinterval [0.1058175, 0.1058250). Any number in this interval can be used to denote the string of the source symbols.

We also observe that arithmetic coding can be carried out in an *incremental* manner. That is, source symbols are fed into the encoder one by one and the final subinterval is refined continually, i.e., the code string is generated continually. Furthermore, it is done in a manner called *first in first out* (FIFO). That is, the source symbol encoded first is decoded first. This manner is superior to that of *last in first out* (LIFO). This is because FIFO is suitable for adaptation to the statistics of the symbol string.

It is obvious that the width of the final subinterval becomes smaller and smaller when the length of the source symbol string becomes larger and larger. This causes what is known as the precision

problem. It is this problem that prohibited arithmetic coding from practical usage for quite a long period of time. Only after this problem was solved in the late 1970s, did arithmetic coding become an increasingly important coding technique.

It is necessary to have a termination symbol at the end of an input source symbol string. In this way, an arithmetic coding system is able to know when to terminate decoding.

Compared with Huffman coding, arithmetic coding is quite different. Basically, Huffman coding converts each source symbol into a fixed codeword with an integral number of bits, while arithmetic coding converts a source symbol string to a code symbol string. To encode the same source symbol string, Huffman coding can be implemented in two different ways. One way is shown in Example 5.9. We construct a fixed codeword for each source symbol. Since Huffman coding is instantaneous, we can cascade the corresponding codewords to form the output, a 17-bit code string 00.101.11.1001.1000.01, where, for easy reading, the five periods are used to indicate different codewords. As we see that for the same source symbol string, the final subinterval obtained by using arithmetic coding is [0.1058175, 0.1058250). It is noted that a decimal in binary number system, 0.000110111111111, which is of 15 bits, is equal to the decimal in decimal number system, 0.1058211962, which falls into the final subinterval representing the string $s_1s_2s_3s_4s_5s_6$. This indicates that, for this example, arithmetic coding is more efficient than Huffamn coding.

Another way is to form a sixth extension of the source alphabet as discussed in Section 5.1.4: treat each group of six source symbols as a new source symbol; calculate its occurrence probability by multiplying the related six probabilities; then apply the Huffman coding algorithm to the sixth extension of the discrete memoryless source. This is called the sixth extension of Huffman block code (refer to Section 5.1.2.2). In other words, in order to encode the source string $s_1s_2s_3s_4s_5s_6$, Huffman coding encodes all of the $6^6 = 46{,}656$ codewords in the sixth extension of the source alphabet. This implies a high complexity in implementation and a large codebook. It is therefore not efficient.

Note that we use the decimal fraction in this section. In binary arithmetic coding, we use the binary fraction. In (Langdon, 1984) both binary source and code alphabets are used in binary arithmetic coding.

Similar to the case of Huffman coding, arithmetic coding is also applicable to r-ary encoding with $r > 2$.

5.4.3 Implementation Issues

As mentioned, the final subinterval resulting from arithmetic encoding of a source symbol string becomes smaller and smaller as the length of the source symbol string increases. That is, the lower and upper bounds of the final subinterval become closer and closer. This causes a growing precision problem. It is this problem that prohibited arithmetic coding from practical usage for a long period of time. This problem has been resolved and the finite precision arithmetic is now used in arithmetic coding. This advance is due to the incremental implementation of arithmetic coding.

5.4.3.1 Incremental Implementation

Recall Example 5.12. As source symbols come in one by one, the lower and upper ends of the final subinterval get closer and closer. In Figure 5.3, these lower and upper ends in Example 5.12 are listed. We observe that after the third symbol, s_3, is encoded, the resultant subinterval is [0.102, 0.108). That is, the two most significant decimal digits are the same and they remain the same in the encoding process. Hence, we can transmit these two digits without affecting the final code string. After the fourth symbol s_4 is encoded, the resultant subinterval is [0.1056, 0.1059). That is, one more digit, 5, can be transmitted. Or we say the cumulative output is now .105. After the sixth symbol is encoded, the final subinterval is [0.1058175, 0.1058250). The cumulative output is 0.1058. Refer to Table 5.11. This important observation reveals that we are able to incrementally transmit output (the code symbols) and receive input (the source symbols that need to be encoded).

TABLE 5.11
Final Subintervals and Cumulative Output in Example 5.12

	Final Subinterval		
Source Symbol	Lower End	Upper End	Cumulative Output
S_1	0	0.3	—
S_2	0.09	0.12	—
S_3	0.102	0.108	0.10
S_4	0.1056	0.1059	0.105
S_5	0.105795	0.105825	0.105
S_6	0.1058175	0.1058250	0.1058

5.4.3.2 Finite Precision

With the incremental manner of transmission of encoded digits and reception of input source symbols, it is possible to use finite precision to represent the lower and upper bounds of the resultant subinterval, which gets closer and closer as the length of the source symbol string becomes long.

Instead of floating-point math, integer math is used. The potential problems known as underflow and overflow, however, need to be carefully monitored and controlled (Bell et al., 1990).

5.4.3.3 Other Issues

There are some other problems that need to be handled in implementation of binary arithmetic coding. Two of them are listed below (Langdon and Rissanen, 1981).

Eliminating Multiplication
The multiplication in the recursive division of subintervals is expensive in hardware as well as software. It can be avoided in binary arithmetic coding so as to simplify the implementation of binary arithmetic coding. The idea is to approximate the lower end of the interval by the closest binary fraction 2^{-Q}, where Q is an integer. Consequently, the multiplication by 2^{-Q} becomes a right shift by Q bits. A simpler approximation to eliminate multiplication is used in the Skew Coder (Langdon and Rissanen, 1982) and the Q-Coder (Pennebaker et al., 1988).

Carry-Over Problem
Carry-over takes place in the addition required in the recursion updating the lower end of the resultant subintervals. A carry may *propagate* over q bits. If the q is larger than the number of bits in the fixed-length register utilized in finite precision arithmetic, the carry-over problem occurs. To block the carry-over problem, a technique known as "bit stuffing" is used, in which an additional buffer register is utilized.

For a detailed discussion on the various issues involved, readers are referred to (Langdon et al., 1981, 1982, 1984; Pennebaker et al., 1988, 1992). Some computer programs of arithmetic coding in C language can be found in (Bell et al., 1990; Nelson and Gailley, 1996).

5.4.4 History

The idea of encoding by using cumulative probability in some ordering, and decoding by comparison of magnitude of binary fraction, was introduced in Shannon's celebrated paper (Shannon, 1948). The recursive implementation of arithmetic coding was devised by Elias. This unpublished result was first introduced by Abramson as a note in his book on information theory and coding

(Abramson, 1963). The result was further developed by Jelinek in his book on information theory (Jelinek, 1968). The growing precision problem prevented arithmetic coding from attaining practical usage, however. The proposal of using finite precision arithmetic was made independently by Pasco (Pasco, 1976) and Rissanen (Rissanen, 1976). Practical arithmetic coding was developed by several independent groups (Rissanen and Langdon, 1979; Rubin, 1979; Guazzo, 1980). A well-known tutorial paper on arithmetic coding appeared in (Langdon, 1984). The tremendous efforts made in IBM led to a new form of adaptive binary arithmetic coding known as the Q-coder (Pennebaker et al., 1988). Based on the Q-coder, the activities of the international still image coding standards JPEG and JBIG combined the best features of the various existing arithmetic coders and developed the binary arithmetic coding procedure known as the QM-coder (Pennebaker and Mitchell, 1992).

5.4.5 Applications

Arithmetic coding is becoming popular. Note that in text and bilevel image applications there are only two source symbols (black and white), and the occurrence probability is skewed. Therefore binary arithmetic coding achieves high coding efficiency. It has been successfully applied to bilevel image coding (Langdon and Rissanen, 1981) and adopted by the international standards for bilevel image compression, JBIG. It has also been adopted by the international still image coding standard, JPEG. More in this regard is covered in the next chapter when we introduce JBIG.

5.5 SUMMARY

So far in this chapter, not much has been explicitly discussed regarding the term variable-length codes. It is known that if source symbols in a source alphabet are equally probable, i.e., their occurrence probabilities are the same, then fixed-length codes such as the natural binary code are a reasonable choice. When the occurrence probabilities, however, are unequal, variable-length codes should be used in order to achieve high coding efficiency. This is one of the restrictions on the minimum redundancy codes imposed by Huffman. That is, the length of the codeword assigned to a probable source symbol should not be larger than that associated with a less probable source symbol. If the occurrence probabilities happen to be the integral powers of 1/2, then choosing the codeword length equal to $-\log_2 p(s_i)$ for a source symbol s_i having the occurrence probability $p(s_i)$ results in minimum redundancy coding. In fact, the average length of the code thus generated is equal to the source entropy.

Huffman devised a systematic procedure to encode a source alphabet consisting of finitely many source symbols, each having an occurrence probability. It is based on some restrictions imposed on the optimum, instantaneous codes. By assigning codewords with variable lengths according to variable probabilities of source symbols, Huffman coding results in minimum redundancy codes, or optimum codes for short. These have found wide applications in image and video coding and have been adopted in the international still image coding standard JPEG and video coding standards H.261, H.263, and MPEG 1 and 2.

When some source symbols have small probabilities and their number is large, the size of the codebook of Huffman codes will require a large memory space. The modified Huffman coding technique employs a special symbol to lump all the symbols with small probabilities together. As a result, it can reduce the codebook memory space drastically while retaining almost the same coding efficiency as that achieved by the conventional Huffman coding technique.

On the one hand, Huffman coding is optimum as a block code for a fixed-source alphabet. On the other hand, compared with the source entropy (the lower bound of the average codeword length) it is not efficient when the probabilities of a source alphabet are skewed with the maximum probability being large. This is caused by the restriction that Huffman coding can only assign an integral number of bits to each codeword.

Another limitation of Huffman coding is that it has to enumerate and encode all the possible groups of n source symbols in the nth extension Huffman code, even though there may be only one such group that needs to be encoded.

Arithmetic coding can overcome the limitations of Huffman coding because it is stream-oriented rather than block-oriented. It translates a stream of source symbols into a stream of code symbols. It can work in an incremental manner. That is, the source symbols are fed into the coding system one by one and the code symbols are output continually. In this stream-oriented way, arithmetic coding is more efficient. It can approach the lower coding bounds set by the noiseless source coding theorem for various sources.

The recursive subinterval division (equivalently, the double recursion: the lower end recursion and width recursion) is the heart of arithmetic coding. Several measures have been taken in the implementation of arithmetic coding. They include the incremental manner, finite precision, and the elimination of multiplication. Due to its merits, binary arithmetic coding has been adopted by the international bilevel image coding standard, JBIG, and still image coding standard, JPEG. It is becoming an increasingly important coding technique.

5.6 EXERCISES

5-1. What does the noiseless source coding theorem state (using your own words)? Under what condition does the average code length approach the source entropy? Comment on the method suggested by the noiseless source coding theorem.

5-2. What characterizes a block code? Consider another definition of block code in (Blahut, 1986): a block code breaks the input data stream into blocks of fixed length n and encodes each block into a codeword of fixed length m. Are these two definitions (the one above and the one in Section 5.1, which comes from [Abramson, 1963]) essentially the same? Explain.

5-3. Is a uniquely decodable code necessarily a prefix condition code?

5-4. For text encoding, there are only two source symbols for black and white. It is said that Huffman coding is not efficient in this application. But it is known as the optimum code. Is there a contradiction? Explain.

5-5. A set of source symbols and their occurrence probabilities is listed in Table 5.12. Apply the Huffman coding algorithm to encode the alphabet.

5-6. Find the Huffman code for the source alphabet shown in Example 5.10.

5-7. Consider a source alphabet $S = \{s_i, i = 1,2,\cdots,32\}$ with $p(s_1) = 1/4$, $p(s_i) = 3/124$, $i = 2,3,\cdots,32$. Determine the source entropy and the average length of Huffman code if applied to the source alphabet. Then apply the modified Huffman coding algorithm. Calculate the average length of the modified Huffman code. Compare the codebook memory required by Huffman code and the modified Huffman code.

5-8. A source alphabet consists of the following four source symbols: s_1, s_2, s_3, and s_4, with their occurrence probabilities equal to 0.25, 0.375, 0.125, and 0.25, respectively. Applying arithmetic coding as shown in Example 5.12 to the source symbol string $s_2s_1s_3s_4$, determine the lower end of the final subinterval.

5-9. For the above problem, show step by step how we can decode the original source string from the lower end of the final subinterval.

5-10. In Problem 5.8, find the codeword of the symbol string $s_2s_1s_3s_4$ by using the fourth extension of the Huffman code. Compare the two methods.

5-11. Discuss how modern arithmetic coding overcame the growing precision problem.

TABLE 5.12
Source Alphabet in Problem 5.5

Source Symbol	Occurrence Probability	Codeword Assigned
S_1	0.20	
S_2	0.18	
S_3	0.10	
S_4	0.10	
S_5	0.10	
S_6	0.06	
S_7	0.06	
S_8	0.04	
S_9	0.04	
S_{10}	0.04	
S_{11}	0.04	
S_{12}	0.04	

REFERENCES

Abramson, N. *Information Theory and Coding*, New York: McGraw-Hill, 1963.

Bell, T. C., J. G. Cleary, and I. H. Witten, *Text Compression*, Englewood, NJ: Prentice-Hall, 1990.

Blahut, R. E. *Principles and Practice of Information Theory*, Reading, MA: Addison-Wesley, 1986.

Fano, R. M. The transmission of information, Tech. Rep. 65, Research Laboratory of Electronics, MIT, Cambridge, MA, 1949.

Gallagher, R. G. Variations on a theme by Huffman, *IEEE Trans. Inf. Theory*, IT-24(6), 668-674, 1978.

Guazzo, M. A general minimum-redundacy source-coding algorithm, *IEEE Trans. Inf. Theory*, IT-26(1), 15-25, 1980.

Hankamer, M. A modified Huffman procedure with reduced memory requirement, *IEEE Trans. Commun.*, COM-27(6), 930-932, 1979.

Huffman, D. A. A method for the construction of minimum-redundancy codes, *Proc. IRE*, 40, 1098-1101, 1952.

Jelinek, F. *Probabilistic Information Theory*, New York: McGraw-Hill, 1968.

Langdon, G. G., Jr. and J. Rissanen, Compression of black-white images with arithmetic coding, *IEEE Trans. Commun.*, COM-29(6), 858-867, 1981.

Langdon, G. G., Jr. and J. Rissanen, A simple general binary source code, *IEEE Trans. Inf. Theory*, IT-28, 800, 1982.

Langdon, G. G., Jr., An introduction to arithmetic coding, *IBM J. Res. Dev.*, 28(2), 135-149, 1984.

Nelson, M. and J. Gailly, *The Data Compression Book*, 2nd ed., New York: M&T Books, 1996.

Pasco, R. Source Coding Algorithms for Fast Data Compression, Ph.D. dissertation, Stanford University, Stanford, CA, 1976.

Pennebaker, W. B., J. L. Mitchell, G. G. Langdon, Jr., and R. B. Arps, An overview of the basic principles of the Q-coder adaptive binary arithmetic Coder, *IBM J. Res. Dev.*, 32(6), 717-726, 1988.

Pennebaker, W. B. and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1992.

Rissanen, J. J. Generalized Kraft inequality and arithmetic coding, *IBM J. Res. Dev.*, 20, 198-203, 1976.

Rissanen, J. J. and G. G. Landon, Arithmetic coding, *IBM J. Res. Dev.*, 23(2), 149-162, 1979.

Rubin, F. Arithmetic stream coding using fixed precision registers, *IEEE Trans. Inf. Theory*, IT-25(6), 672-675, 1979.

Sayood, K. *Introduction to Data Compression*, San Francisco, CA: Morgan Kaufmann Publishers, 1996.

Shannon, C. E. A mathematical theory of communication, *Bell Syst. Tech. J.*, 27, 379-423, 1948; 623-656, 1948.

Weaver, C. S., Digital ECG data compression, in *Digital Encoding of Electrocardiograms*, H. K. Wolf, Ed., Springer-Verlag, Berlin/New York, 1979.

6 Run-Length and Dictionary Coding: Information Theory Results (III)

As mentioned at the beginning of Chapter 5, we are studying some codeword assignment (encoding) techniques in Chapters 5 and 6. In this chapter, we focus on run-length and dictionary-based coding techniques. We first introduce Markov models as a type of dependent source model in contrast to the memoryless source model discussed in Chapter 5. Based on the Markov model, run-length coding is suitable for facsimile encoding. Its principle and application to facsimile encoding are discussed, followed by an introduction to dictionary-based coding, which is quite different from Huffman and arithmetic coding techniques covered in Chapter 5. Two types of adaptive dictionary coding techniques, the LZ77 and LZ78 algorithms, are presented. Finally, a brief summary of and a performance comparison between international standard algorithms for lossless still image coding are presented.

Since the Markov source model, run-length, and dictionary-based coding are the core of this chapter, we consider this chapter as a third part of the information theory results presented in the book. It is noted, however, that the emphasis is placed on their applications to image and video compression.

6.1 MARKOV SOURCE MODEL

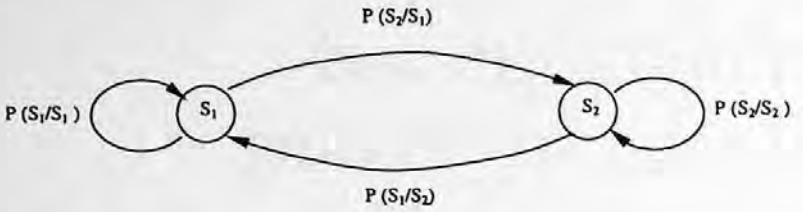
In the previous chapter we discussed the discrete memoryless source model, in which source symbols are assumed to be independent of each other. In other words, the source has zero memory, i.e., the previous status does not affect the present one at all. In reality, however, many sources are dependent in nature. Namely, the source has memory in the sense that the previous status has an influence on the present status. For instance, as mentioned in Chapter 1, there is an interpixel correlation in digital images. That is, pixels in a digital image are not independent of each other. As will be seen in this chapter, there is some dependence between characters in text. For instance, the letter u often follows the letter q in English. Therefore it is necessary to introduce models that can reflect this type of dependence. A Markov source model is often used in this regard.

6.1.1 Discrete Markov Source

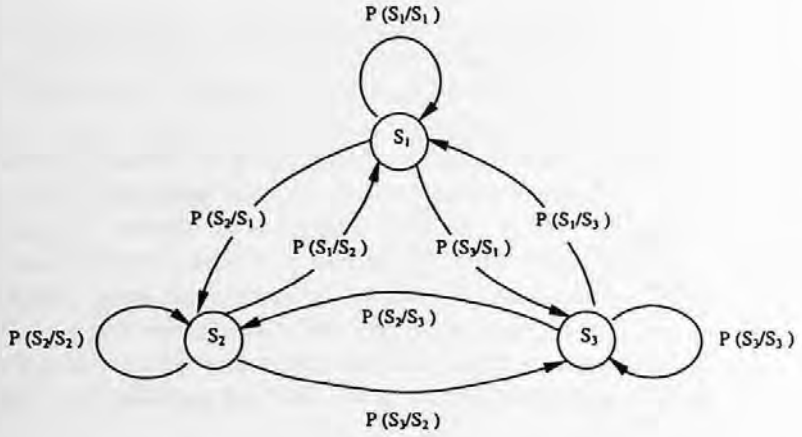
Here, as in the previous chapter, we denote a source alphabet by $S = \{s_1, s_2, \cdots, s_m\}$ and the occurrence probability by p. An lth order Markov source is characterized by the following equation of conditional probabilities.

$$p\left(s_j \middle| s_{i1}, s_{i2}, \cdots, s_{il}, \cdots\right) = p\left(s_j \middle| s_{i1}, s_{i2}, \cdots, s_{il}\right), \tag{6.1}$$

where $j, i1, i2, \cdots, il, \cdots \in \{1,2,\cdots,m\}$, i.e., the symbols $s_j, s_{i1}, s_{i2}, \cdots, s_{il}, \cdots$ are chosen from the source alphabet S. This equation states that the source symbols are not independent of each other. The occurrence probability of a source symbol is determined by some of its previous symbols. Specifically, the probability of s_j given its history being $s_{i1}, s_{i2}, \cdots, s_{il}, \cdots$ (also called the transition probability), is determined completely by the immediately previous l symbols $s_{i1}, \cdots, s_{il}$. That is,



(a)



(b)

FIGURE 6.1 State diagrams of the first-order Markov sources with their source alphabets having (a) two symbols and (b) three symbols.

the knowledge of the entire sequence of previous symbols is equivalent to that of the l symbols immediately preceding the current symbol s_j.

An lth order Markov source can be described by what is called a *state diagram*. A state is a sequence of $(s_{i1}, s_{i2}, \cdots, s_{il})$ with $i1, i2, \cdots, il \in \{1,2,\cdots,m\}$. That is, any group of l symbols from the m symbols in the source alphabet S forms a state. When $l = 1$, it is called a first-order Markov source. The state diagrams of the first-order Markov sources, with their source alphabets having two and three symbols, are shown in Figure 6.1(a) and (b), respectively. Obviously, an lth order Markov source with m symbols in the source alphabet has a total of m^l different states. Therefore, we conclude that a state diagram consists of all the m^l states. In the diagram, all the transition probabilities together with appropriate arrows are used to indicate the state transitions.

The source entropy at a state $(s_{i1}, s_{i2}, \cdots, s_{il})$ is defined as

$$H\left(S|s_{i1},s_{i2},\cdots,s_{il}\right)=-\sum_{j=1}^{m}p\left(s_j|s_{i1},s_{i2},\cdots,s_{il}\right)\log_2 p\left(s_j|s_{i1},s_{i2},\cdots,s_{il}\right) \tag{6.2}$$

The source entropy is defined as the statistical average of the entropy at all the states. That is,

$$H(S)=\sum_{(s_{i1},s_{i2},\cdots,s_{il})\in S^l}p\left(s_{i1},s_{i2},\cdots,s_{il}\right)H\left(S|s_{i1},s_{i2},\cdots,s_{il}\right), \tag{6.3}$$

where, as defined in the previous chapter, S^l denotes the lth extension of the source alphabet S. That is, the summation is carried out with respect to all l-tuples taking over the S^l. Extensions of a Markov source are defined below.

6.1.2 Extensions of a Discrete Markov Source

An extension of a Markov source can be defined in a similar way to that of an extension of a memoryless source in the previous chapter. The definition of extensions of a Markov source and the relation between the entropy of the original Markov source and the entropy of the nth extension of the Markov source are presented below without derivation. For the derivation, readers are referred to (Abramson, 1963).

6.1.2.1 Definition

Consider an lth order Markov source $S = \{s_1, s_2, \cdots, s_m\}$ and a set of conditional probabilities $p(s_j | s_{i1}, s_{i2}, \cdots, s_{il})$, where $j,i1, i2, \cdots, il \in \{1,2,\cdots,m\}$. Similar to the memoryless source discussed in Chapter 5, if n symbols are grouped into a block, then there is a total of m^n blocks. Each block can be viewed as a new source symbol. Hence, these m^n blocks form a new information source alphabet, called the nth extension of the source S and denoted by S^n. The nth extension of the lth-order Markov source is a kth-order Markov source, where k is the smallest integer greater than or equal to the ratio between l and n. That is,

$$k = \left\lceil \frac{l}{n} \right\rceil, \tag{6.4}$$

where the notation $|a|$ represents the operation of taking the smallest integer greater than or equal to the quantity a.

6.1.2.2 Entropy

Denote, respectively, the entropy of the lth order Markov source S by $H(S)$, and the entropy of the nth extension of the lth order Markov source, S^n, by $H(S^n)$. The following relation between the two entropies can be shown:

$$H(S^n) = nH(S) \tag{6.5}$$

6.1.3 Autoregressive (AR) Model

The Markov source discussed above represents a kind of dependence between source symbols in terms of the transition probability. Concretely, in determining the transition probability of a present source symbol given all the previous symbols, only the set of finitely many immediately preceding symbols matters. The autoregressive model is another kind of dependent source model that has been used often in image coding. It is defined below.

$$s_j = \sum_{k=1}^{l} a_k s_{ik} + x_j, \tag{6.6}$$

where s_j represents the currently observed source symbol, while s_{ik} with $k = 1,2,\cdots,l$ denote the l preceding observed symbols, a_k's are coefficients, and x_j is the current input to the model. If $l = 1$,

the model defined in Equation 6.6 is referred to as the first-order AR model. Clearly, in this case, the current source symbol is a linear function of its preceding symbol.

6.2 RUN-LENGTH CODING (RLC)

The term *run* is used to indicate the repetition of a symbol, while the term *run-length* is used to represent the number of repeated symbols, in other words, the number of consecutive symbols of the same value. Instead of encoding the consecutive symbols, it is obvious that encoding the run-length and the value that these consecutive symbols commonly share may be more efficient. According to an excellent early review on binary image compression by Arps (1979), RLC has been in use since the earliest days of information theory (Shannon and Weaver, 1949; Laemmel, 1951).

From the discussion of the JPEG in Chapter 4 (with more details in Chapter 7), it is seen that most of the DCT coefficients within a block of 8×8 are zero after certain manipulations. The DCT coefficients are zigzag scanned. The nonzero DCT coefficients and their addresses in the 8×8 block need to be encoded and transmitted to the receiver side. There, the nonzero DCT values are referred to as labels. The position information about the nonzero DCT coefficients is represented by the run-length of zeros between the nonzero DCT coefficients in the zigzag scan. The labels and the run-length of zeros are then Huffman coded.

Many documents such as letters, forms, and drawings can be transmitted using facsimile machines over the general switched telephone network (GSTN). In digital facsimile techniques, these documents are quantized into binary levels: black and white. The resolution of these binary tone images is usually very high. In each scan line, there are many consecutive white and black pixels, i.e., many alternate white runs and black runs. Therefore it is not surprising to see that RLC has proven to be efficient in binary document transmission. RLC has been adopted in the international standards for facsimile coding: the CCITT Recommendations T.4 and T.6.

RLC using only the horizontal correlation between pixels on the same scan line is referred to as 1-D RLC. It is noted that the first-order Markov source model with two symbols in the source alphabet depicted in Figure 6.1(a) can be used to characterize 1-D RLC. To achieve higher coding efficiency, 2-D RLC utilizes both horizontal and vertical correlation between pixels. Both the 1-D and 2-D RLC algorithms are introduced below.

6.2.1 1-D Run-Length Coding

In this technique, each scan line is encoded independently. Each scan line can be considered as a sequence of alternating, independent white runs and black runs. As an agreement between encoder and decoder, the first run in each scan line is assumed to be a white run. If the first actual pixel is black, then the run-length of the first white run is set to be zero. At the end of each scan line, there is a special codeword called end-of-line (EOL). The decoder knows the end of a scan line when it encounters an EOL codeword.

Denote run-length by r, which is integer-valued. All of the possible run-lengths construct a source alphabet R, which is a random variable. That is,

$$R = \{r : r \in 0, 1, 2, \cdots\} \tag{6.7}$$

Measurements on typical binary documents have shown that the maximum compression ratio, ζ_{max}, which is defined below, is about 25% higher when the white and black runs are encoded separately (Hunter and Robinson, 1980). The average white run-length, $\bar{r}_W$, can be expressed as

$$\bar{r}_W = \sum_{r=0}^{m} r \cdot P_W(r) \tag{6.8}$$

where m is the maximum value of the run-length, and $P_W(r)$ denotes the occurrence probability of a white run with length r. The entropy of the white runs, H_W, is

$$H_W = -\sum_{r=0}^{m} P_W(r) \log_2 P_W(r) \tag{6.9}$$

For the black runs, the average run-length $\bar{r}_B$ and the entropy H_B can be defined similarly. The maximum theoretical compression factor ζ_{max} is

$$\zeta_{max} = \frac{\bar{r}_W + \bar{r}_B}{H_W + H_B} \tag{6.10}$$

Huffman coding is then applied to two source alphabets. According to CCITT Recommendation T.4, A4 size (210 × 297 mm) documents should be accepted by facsimile machines. In each scan line, there are 1728 pixels. This means that the maximum run-length for both white and black runs is 1728, i.e., $m = 1728$. Two source alphabets of such a large size imply the requirement of two large codebooks, hence the requirement of large storage space. Therefore, some modification was made, resulting in the "modified" Huffman (MH) code.

In the modified Huffman code, if the run-length is larger than 63, then the run-length is represented as

$$r = M \times 64 + T \quad as \quad r > 63, \tag{6.11}$$

where M takes integer values from 1, 2 to 27, and $M \times 64$ is referred to as the makeup run-length; T takes integer values from 0, 1 to 63, and is called the terminating run-length. That is, if $r \leq 63$, the run-length is represented by a terminating codeword only. Otherwise, if $r > 63$, the run-length is represented by a makeup codeword and a terminating codeword. A portion of the modified Huffman code table (Hunter and Robinson, 1980) is shown in Table 6.1. In this way, the requirement of large storage space is alleviated. The idea is similar to that behind modified Huffman coding, discussed in Chapter 5.

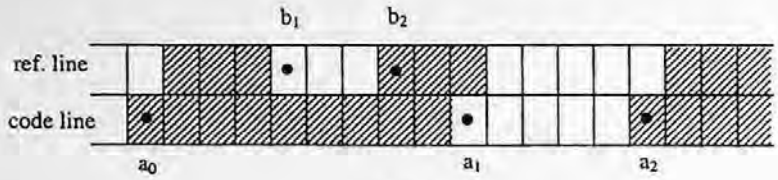
6.2.2 2-D Run-Length Coding

The 1-D run-length coding discussed above only utilizes correlation between pixels within a scan line. In order to utilize correlation between pixels in neighboring scan lines to achieve higher coding efficiency, 2-D run-length coding was developed. In Recommendation T.4, the modified relative element address designate (READ) code, also known as the modified READ code or simply the MR code, was adopted.

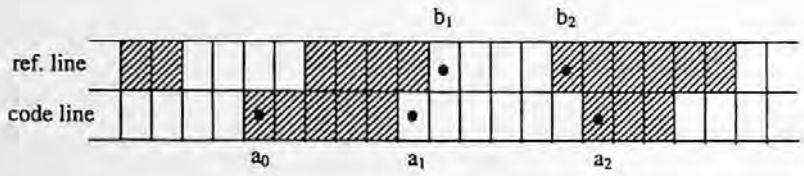
The modified READ code operates in a line-by-line manner. In Figure 6.2, two lines are shown. The top line is called the reference line, which has been coded, while the bottom line is referred to as the coding line, which is being coded. There are a group of five changing pixels, a_0, a_1, a_2, b_1, b_2, in the two lines. Their relative positions decide which of the three coding modes is used. The starting changing pixel a_0 (hence, five changing points) moves from left to right and from top to bottom as 2-D run-length coding proceeds. The five changing pixels and the three coding modes are defined below.

6.2.2.1 Five Changing Pixels

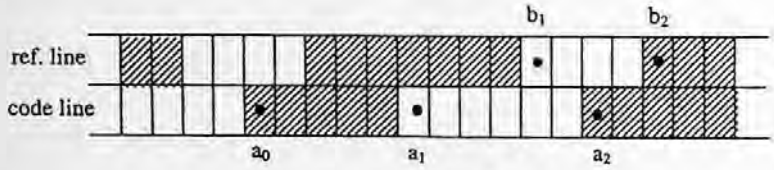
By a changing pixel, we mean the first pixel encountered in white or black runs when we scan an image line-by-line, from left to right, and from top to bottom. The five changing pixels are defined below.



(a) Pass mode



(b) Vertical mode



(c) Horizontal mode

FIGURE 6.2 2-D run-length coding.

a_0: The reference-changing pixel in the coding line. Its position is defined in the previous coding mode, whose meaning will be explained shortly. At the beginning of a coding line, a_0 is an imaginary white changing pixel located before the first actual pixel in the coding line.

a_1: The next changing pixel in the coding line. Because of the above-mentioned left-to-right and top-to-bottom scanning order, it is at the right-hand side of a_0. Since it is a changing pixel, it has an opposite "color" to that of a_0.

a_2: The next changing pixel after a_1 in the coding line. It is to the right of a_1 and has the same color as that of a_0.

b_1: The changing pixel in the reference line that is closest to a_0 from the right and has the same color as a_1.

b_2: The next changing pixel in the reference line after b_1.

6.2.2.2 Three Coding Modes

Pass Coding Mode — If the changing pixel b_2 is located to the left of the changing pixel a_1, it means that the run in the reference line starting from b_1 is not adjacent to the run in the coding line starting from a_1. Note that these two runs have the same color. This is called the pass coding mode. A special codeword, "0001", is sent out from the transmitter. The receiver then knows that the run starting from a_0 in the coding line does not end at the pixel below b_2. This pixel (below b_2 in the coding line) is identified as the reference-changing pixel a_0 of the new set of five changing pixels for the next coding mode.

Vertical Coding Mode — If the relative distance along the horizontal direction between the changing pixels a_1 and b_1 is not larger than three pixels, the coding is conducted in vertical coding

TABLE 6.1
Modified Huffman Code Table (Hunter and Robinson, 1980)

Run-Length	White Runs	Black Runs
	Terminating Codewords	
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010
7	1111	00011
8	10011	000101
⋮	⋮	⋮
60	01001011	000000101100
61	00110010	000001011010
62	00110011	000001100110
63	00110100	000001100111
	Makeup Codewords	
64	11011	0000001111
128	10010	000011001000
192	010111	000011001001
256	0110111	000001011011
⋮	⋮	⋮
1536	010011001	0000001011010
1600	010011010	0000001011011
1664	011000	0000001100100
1728	010011011	0000001100101
EOL	000000000001	000000000001

TABLE 6.2
2-D Run-Length Coding Table

Mode	Conditions	Output Codeword	Position of New a_0
Pass coding mode	$b_2a_1 < 0$	0001	Under b_2 in coding line
Vertical coding mode	$a_1b_1 = 0$	1	a_1
	$a_1b_1 = 1$	011	
	$a_1b_1 = 2$	000011	
	$a_1b_1 = 3$	0000011	
	$a_1b_1 = -1$	010	
	$a_1b_1 = -2$	000010	
	$a_1b_1 = -3$	0000010	
Horizontal coding mode	$\|a_1b_1\| > 3$	$001 + (a_0a_1) + (a_1a_2)$	a_2

Note: $|x_iy_j|$: distance between x_i and y_j, $x_iy_j > 0$: x_i is right to y_j, $x_iy_j < 0$: x_i is left to y_j. (x_iy_j): codeword of the run denoted by x_iy_j taken from the modified Huffman code.

Source: From Hunter and Robinson (1980).

mode. That is, the position of a_1 is coded with reference to the position of b_1. Seven different codewords are assigned to seven different cases: the distance between a_1 and b_1 equals 0, ±1, ±2, ±3, where + means a_1 is to the right of b_1, while – means a_1 is to the left of b_1. The a_1 then becomes the reference changing pixel a_0 of the new set of five changing pixels for the next coding mode.

Horizontal Coding Mode — If the relative distance between the changing pixels a_1 and b_1 is larger than three pixels, the coding is conducted in horizontal coding mode. Here, 1-D run-length coding is applied. Specifically, the transmitter sends out a codeword consisting the following three parts: a flag "001"; a 1-D run-length codeword for the run from a_0 to a_1; a 1-D run-length codeword for the run from a_1 to a_2. The a_2 then becomes the reference changing pixel a_0 of the new set of five changing pixels for the next coding mode. Table 6.2 contains three coding modes and the corresponding output codewords. There, (a_0a_1) and (a_1a_2) represent 1-D run-length codewords of run-length a_0a_1 and a_1a_2, respectively.

6.2.3 Effect of Transmission Error and Uncompressed Mode

In this subsection, effect of transmission error in the 1-D and 2-D RLC cases and uncompressed mode are discussed.

6.2.3.1 Error Effect in the 1-D RLC Case

As introduced above, the special codeword EOL is used to indicate the end of each scan line. With the EOL, 1-D run-length coding encodes each scan line independently. If a transmission error occurs in a scan line, there are two possibilities that the effect caused by the error is limited within the scan line. One possibility is that *resynchronization* is established after a few runs. One example is shown in Figure 6.3. There, the transmission error takes place in the second run from the left. Resynchronization is established in the fifth run in this example. Another possibility lies in the EOL, which forces resynchronization.

In summary, it is seen that the 1-D run-length coding will not propagate transmission error between scan lines. In other words, a transmission error will be restricted within a scan line. Although error detection and retransmission of data via an automatic repeat request (ARQ) system is supposed to be able to effectively handle the error susceptibility issue, the ARQ technique was not included in Recommendation T.4 due to the computational complexity and extra transmission time required.

Once the number of decoded pixels between two consecutive EOL codewords is not equal to 1728 (for an A4 size document), an error has been identified. Some *error concealment* techniques can be used to reconstruct the scan line (Hunter and Robinson, 1980). For instance, we can repeat

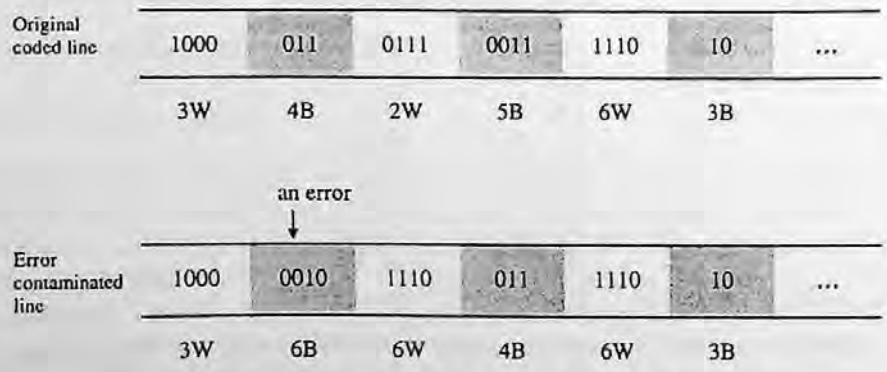


FIGURE 6.3 Establishment of resynchronization after a few runs.

the previous line, or replace the damaged line by a white line, or use a correlation technique to recover the line as much as possible.

6.2.3.2 Error Effect in the 2-D RLC Case

From the above discussion, we realize that 2-D RLC is more efficient than 1-D RLC on the one hand. The 2-D RLC is more susceptible to transmission errors than the 1-D RLC on the other hand. To prevent error propagation, there is a parameter used in 2-D RLC, known as the *K-factor*, which specifies the number of scan lines that are 2-D RLC coded.

Recommendation T.4 defined that no more than K-1 consecutive scan lines be 2-D RLC coded after a 1-D RLC coded line. For binary documents scanned at normal resolution, K = 2. For documents scanned at high resolution, K = 4.

According to Arps (1979), there are two different types of algorithms in binary image coding, *raster* algorithms and *area* algorithms. Raster algorithms only operate on data within one or two raster scan lines. They are hence mainly 1-D in nature. Area algorithms are truly 2-D in nature. They require that all, or a substantial portion, of the image is in random access memory. From our discussion above, we see that both 1-D and 2-D RLC defined in T.4 belong to the category of raster algorithms. Area algorithms require large memory space and are susceptible to transmission noise.

6.2.3.3 Uncompressed Mode

For some detailed binary document images, both 1-D and 2-D RLC may result in data expansion instead of data compression. Under these circumstances the number of coding bits is larger than the number of bilevel pixels. An uncompressed mode is created as an alternative way to avoid data expansion. Special codewords are assigned for the uncompressed mode.

For the performances of 1-D and 2-D RLC applied to eight CCITT test document images, and issues such as "fill bits" and "minimum scan line time (MSLT)," to name only a few, readers are referred to an excellent tutorial paper by Hunter and Robinson (1980).

6.3 DIGITAL FACSIMILE CODING STANDARDS

Facsimile transmission, an important means of communication in modern society, is often used as an example to demonstrate the mutual interaction between widely used applications and standardization activities. Active facsimile applications and the market brought on the necessity for international standardization in order to facilitate interoperability between facsimile machines worldwide. Successful international standardization, in turn, has stimulated wider use of facsimile transmission and, hence, a more demanding market. Facsimile has also been considered as a major application for binary image compression.

So far, facsimile machines are classified in four different groups. Facsimile apparatuses in groups 1 and 2 use analog techniques. They can transmit an A4 size (210 × 297 mm) document scanned at 3.85 lines/mm in 6 and 3 min, respectively, over the GSTN. International standards for these two groups of facsimile apparatus are CCITT (now ITU) Recommendations T.2 and T.3, respectively. Group 3 facsimile machines use digital techniques and hence achieve high coding efficiency. They can transmit the A4 size binary document scanned at a resolution of 3.85 lines/mm and sampled at 1728 pixels per line in about 1 min at a rate of 4800 b/sec over the GSTN. The corresponding international standard is CCITT Recommendations T.4. Group 4 facsimile apparatuses have the same transmission speed requirement as that for group 3 machines, but the coding technique is different. Specifically, the coding technique used for group 4 machines is based on 2-D run-length coding, discussed above, but modified to achieve higher coding efficiency. Hence it is referred to as the modified modified READ coding, abbreviated MMR. The corresponding standard is CCITT Recommendations T.6. Table 6.3 summarizes the above descriptions.

TABLE 6.3 FACSIMILE CODING STANDARDS

Group of Facsimile Apparatuses	Speed Requirement for A-4 Size Document	Analog or Digital Scheme	CCITT Recommendation	Compression Technique		
				Model	Basic Coder	Algorithm Acronym
G_1	6 min	Analog	T.2	—	—	—
G_2	3 min	Analog	T.3	—	—	—
G_3	1 min	Digital	T.4	1-D RLC	Modified Huffman	MH
				2-D RLC (optional)		MR
G_4	1 min	Digital	T.6	2-D RLC	Modified Huffman	MMR

6.4 DICTIONARY CODING

Dictionary coding, the focus of this section, is different from Huffman coding and arithmetic coding, discussed in the previous chapter. Both Huffman and arithmetic coding techniques are based on a statistical model, and the occurrence probabilities play a particular important role. Recall that in the Huffman coding the shorter codewords are assigned to more frequently occurring source symbols. In dictionary-based data compression techniques a symbol or a string of symbols generated from a source alphabet is represented by an index to a dictionary constructed from the source alphabet. A dictionary is a list of symbols and strings of symbols. There are many examples of this in our daily lives. For instance, the string "September" is sometimes represented by an index "9," while a social security number represents a person in the U.S.

Dictionary coding is widely used in text coding. Consider English text coding. The source alphabet includes 26 English letters in both lower and upper cases, numbers, various punctuation marks, and the space bar. Huffman or arithmetic coding treats each symbol based on its occurrence probability. That is, the source is modeled as a memoryless source. It is well known, however, that this is not true in many applications. In text coding, *structure* or *context* plays a significant role. As mentioned earlier, it is very likely that the letter u appears after the letter q. Likewise, it is likely that the word "concerned" will appear after "As far as the weather is." The strategy of the dictionary coding is to build a dictionary that contains frequently occurring symbols and string of symbols. When a symbol or a string is encountered and it is contained in the dictionary, it is encoded with an index to the dictionary. Otherwise, if not in the dictionary, the symbol or the string of symbols is encoded in a less efficient manner.

6.4.1 Formulation of Dictionary Coding

To facilitate further discussion, we define dictionary coding in a precise manner (Bell et al., 1990). We denote a source alphabet by S. A dictionary consisting of two elements is defined as $D = (P, C)$, where P is a finite set of phrases generated from the S, and C is a coding function mapping P onto a set of codewords.

The set P is said to be complete if any input string can be represented by a series of phrases chosen from the P. The coding function C is said to obey the prefix property if there is no codeword that is a prefix of any other codeword. For practical usage, i.e., for reversible compression of any input text, the phrase set P must be complete and the coding function C must satisfy the prefix property.

6.4.2 Categorization of Dictionary-Based Coding Techniques

The heart of dictionary coding is the formulation of the dictionary. A successfully built dictionary results in data compression; the opposite case may lead to data expansion. According to the ways

in which dictionaries are constructed, dictionary coding techniques can be classified as static or adaptive.

6.4.2.1 Static Dictionary Coding

In some particular applications, the knowledge about the source alphabet and the related strings of symbols, also known as phrases, is sufficient for a fixed dictionary to be produced before the coding process. The dictionary is used at both the transmitting and receiving ends. This is referred to as static dictionary coding. The merit of the static approach is its simplicity. Its drawbacks lie in its relatively lower coding efficiency and less flexibility compared with adaptive dictionary techniques. By less flexibility, we mean that a dictionary built for a specific application is not normally suitable for utilization in other applications.

An example of static algorithms occurring is *digram* coding. In this simple and fast coding technique, the dictionary contains all source symbols and some frequently used pairs of symbols. In encoding, two symbols are checked at once to see if they are in the dictionary. If so, they are replaced by the index of the two symbols in the dictionary, and the next pair of symbols is encoded in the next step. If not, then the index of the first symbol is used to encode the first symbol. The second symbol is combined with the third symbol to form a new pair, which is encoded in the next step.

The digram can be straightforwardly extended to *n-gram*. In the extension, the size of the dictionary increases and so does its coding efficiency.

6.4.2.2 Adaptive Dictionary Coding

As opposed to the static approach, with the adaptive approach a completely defined dictionary does not exist prior to the encoding process and the dictionary is not fixed. At the beginning of coding, only an initial dictionary exists. It adapts itself to the input during the coding process. All the adaptive dictionary coding algorithms can be traced back to two different original works by Ziv and Lempel (1977, 1978). The algorithms based on Ziv and Lempel (1977) are referred to as the LZ77 algorithms, while those based on their 1978 work are the LZ78 algorithms. Prior to introducing the two landmark works, we will discuss the parsing strategy.

6.4.3 Parsing Strategy

Once we have a dictionary, we need to examine the input text and find a string of symbols that matches an item in the dictionary. Then the index of the item to the dictionary is encoded. This process of segmenting the input text into disjoint strings (whose union equals the input text) for coding is referred to as *parsing*. Obviously, the way to segment the input text into strings is not unique.

In terms of the highest coding efficiency, optimal parsing is essentially a shortest-path problem (Bell et al., 1990). In practice, however, a method called *greedy* parsing is used most often. In fact, it is used in all the LZ77 and LZ78 algorithms. With greedy parsing, the encoder searches for the longest string of symbols in the input that matches an item in the dictionary at each coding step. Greedy parsing may not be optimal, but it is simple in its implementation.

Example 6.1

Consider a dictionary, D, whose phrase set $P = \{a, b, ab, ba, bb, aab, bbb\}$. The codewords assigned to these strings are $C(a) = 10$, $C(b) = 11$, $C(ab) = 010$, $C(ba) = 0101$, $C(bb) = 01$, $C(abb) = 11$, and $C(bbb) = 0110$. Now the input text is *abbaab*.

Using greedy parsing, we then encode the text as $C(ab).C(ba).C(ab)$, which is a 10-bit string: 010.0101.010. In the above representations, the periods are used to indicate the division of segments in the parsing. This, however, is not an optimum solution. Obviously, the following parsing will be more efficient, i.e., $C(a).C(bb).C(aab)$, which is a 6-bit string: 10.01.11.

6.4.4 Sliding Window (LZ77) Algorithms

As mentioned earlier, LZ77 algorithms are a group of adaptive dictionary coding algorithms rooted in the pioneering work of Ziv and Lempel (1977). Since they are adaptive, there is no complete and fixed dictionary before coding. Instead, the dictionary changes as the input text changes.

6.4.4.1 Introduction

In the LZ77 algorithms, the dictionary used is actually a portion of the input text, which has been recently encoded. The text that needs to be encoded is compared with the strings of symbols in the dictionary. The longest matched string in the dictionary is characterized by a *pointer* (sometimes called a *token*), which is represented by a triple of data items. Note that this triple functions as an index to the dictionary, as mentioned above. In this way, a variable-length string of symbols is mapped to a fixed-length pointer.

There is a sliding window in the LZ77 algorithms. The window consists of two parts: a search buffer and a look-ahead buffer. The search buffer contains the portion of the text stream that has recently been encoded which, as mentioned, is the dictionary; while the look-ahead buffer contains the text to be encoded next. The window slides through the input text stream from beginning to end during the entire encoding process. This explains the term *sliding* window. The size of the search buffer is much larger than that of the look-ahead buffer. This is expected because what is contained in the search buffer is in fact the adaptive dictionary. The sliding window is usually on the order of a few thousand symbols, whereas the look-ahead buffer is on the order of several tens to one hundred symbols.

6.4.4.2 Encoding and Decoding

Below we present more details about the sliding window dictionary coding technique, i.e., the LZ77 approach, via a simple illustrative example.

Example 6.2

Figure 6.4 shows a sliding window. The input text stream is *ikaccbadaccbaccbaccgikmoabc*. In part (a) of the figure, a search buffer of nine symbols and a look-ahead buffer of six symbols are shown. All the symbols in the search buffer, *accbadacc*, have just been encoded. All the symbols in the look-ahead buffer, *baccba*, are to be encoded. (It is understood that the symbols before the

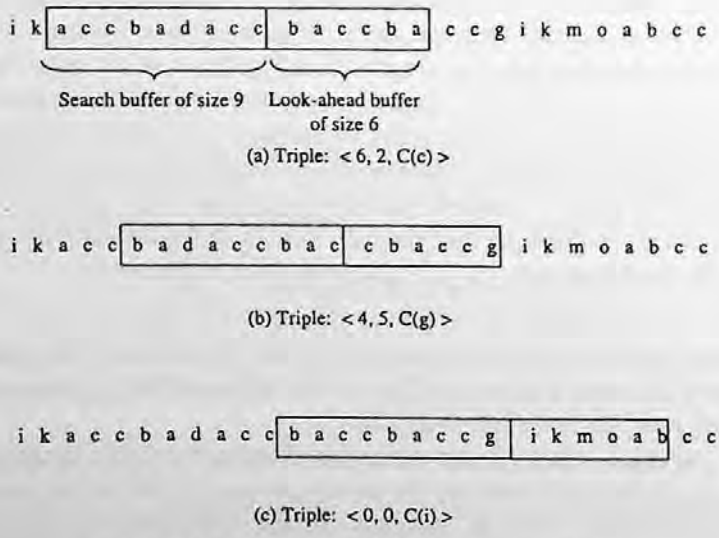


FIGURE 6.4 An encoding example using LZ77.

search buffer have been encoded and the symbols after the look-ahead buffer are to be encoded.) The strings of symbols, *ik* and *ccgikmoabcc*, are not covered by the sliding window at the moment.

At the moment, or in other words, in the first step of encoding, the symbol(s) to be encoded begin(s) with the symbol *b*. The pointer starts searching for the symbol *b* from the last symbol in the search buffer, *c*, which is immediately to the left of the first symbol *b* in the look-ahead buffer. It finds a match at the sixth position from *b*. It further determines that the longest string of the match is *ba*. That is, the maximum matching length is two. The pointer is then represented by a triple, <i,j,k>. The first item, "i", represents the distance between the first symbol in the look-ahead buffer and the position of the pointer (the position of the first symbol of the matched string). This distance is called *offset*. In this step, the offset is six. The second item in the triple, "j", indicates the length of the matched string. Here, the length of the matched string *ba* is two. The third item, "k", is the codeword assigned to the symbol immediately following the matched string in the look-ahead buffer. In this step, the third item is C(c), where C is used to represent a function to map symbol(s) to a codeword, as defined in Section 6.4.1. That is, the resulting triple after the first step is: <6, 2, C(c)>.

The reason to include the third item "k" into the triple is as follows. In the case where there is no match in the search buffer, both "i" and "j" will be zero. The third item at this moment is the codeword of the first symbol in the look-ahead buffer itself. This means that even in the case where we cannot find a match string, the sliding window still works. In the third step of the encoding process described below, we will see that the resulting triple is: <0, 0, C(i)>. The decoder hence understands that there is no matching, and the single symbol *i* is decoded.

The second step of the encoding is illustrated in part (b) of Figure 6.4. The sliding window has been shifted to the right by three positions. The first symbol to be encoded now is *c*, which is the left-most symbol in the look-ahead buffer. The search pointer moves towards the left from the symbol *c*. It first finds a match in the first position with a length of one. It then finds another match in the fourth position from the first symbol in the look-ahead buffer. Interestingly, the maximum matching can exceed the boundary between the search buffer and the look-ahead buffer and can enter the look-ahead buffer. Why this is possible will be seen shortly, when we discuss the decoding process. In this manner, it is found that the maximum length of matching is five. The last match is found at the fifth position. The length of the matched string, however, is only one. Since greedy parsing is used, the match with a length five is chosen. That is, the offset is four and the maximum match length is five. Consequently, the triple resulting from the second step is <4, 5, C(g)>.

The sliding window is then shifted to the right by six positions. The third step of the encoding is depicted in Part (c). Obviously, there is no matching of *i* in the search buffer. The resulting triple is hence <0, 0, C(i)>.

The encoding process can continue in this way. The possible cases we may encounter in the encoding, however, are described in the first three steps. Hence we end our discussion of the encoding process and discuss the decoding process. Compared with the encoding, the decoding is simpler because there is no need for matching, which involves many comparisons between the symbols in the look-ahead buffer and the symbols in the search buffer. The decoding process is illustrated in Figure 6.5.

In the above three steps, the resulting triples are <6, 2, C(c)>, <4, 5, C(g)>, and <0, 0, C(i)>. Now let us see how the decoder works. That is, how the decoder recovers the string *baccbaccgi* from these three triples.

In part (a) of Figure 6.5, the search buffer is the same as that in part (a) of Figure 6.4. That is, the string *accbadacc* stored in the search window is what was just decoded.

Once the first triple <6, 2, C(c)> is received, the decoder will move the decoding pointer from the first position in the look-ahead buffer to the left by six positions. That is, the pointer will point to the symbol *b*. The decoder then copies the two symbols starting from *b*, i.e., *ba*, into the look-ahead buffer. The symbol *c* will be copied right to *ba*. This is shown in part (b) of Figure 6.5. The window is then shifted to the right by three positions, as shown in part (c) of Figure 6.5.

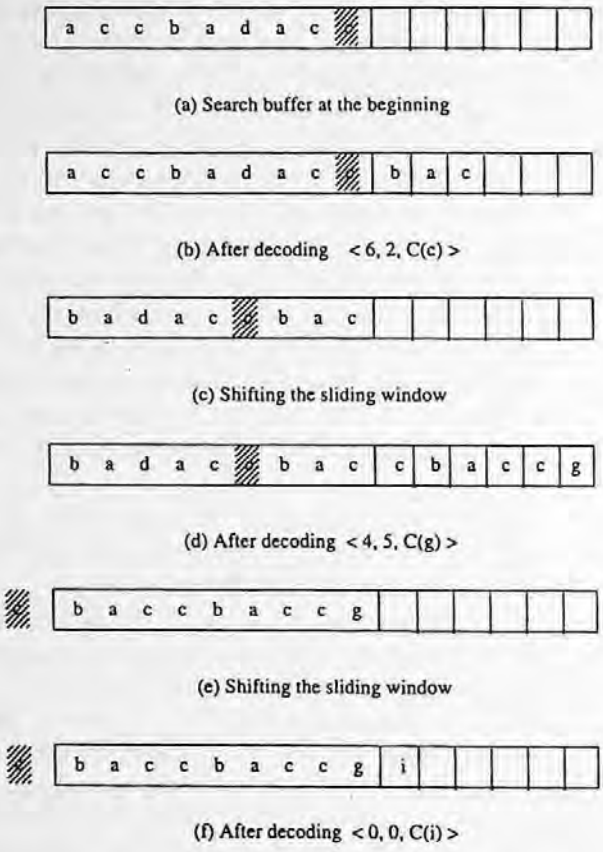


FIGURE 6.5 A decoding example using LZ77.

After the second triple <4, 5, C(g)> is received, the decoder moves the decoding pointer from the first position of the look-ahead buffer to the left by four positions. The pointer points to the symbol *c*. The decoder then copies five successive symbols starting from the symbol *c* pointed by the pointer. We see that at the beginning of this copying process there are only four symbols available for copying. Once the first symbol is copied, however, all five symbols are available. After copying, the symbol *g* is added to the end of the five copied symbols in the look-ahead buffer. The results are shown in part (c) of Figure 6.5. Part (d) then shows the window shifting to the right by six positions.

After receiving the triple <0, 0, C(i)>, the decoder knows that there is no match and a single symbol *i* is encoded. Hence, the decoder adds the symbol *i* following the symbol *g*. This is shown in part (f) of Figure 6.5.

In Figure 6.5, for each part, the last previously encoded symbol *c* prior to the receiving of the three triples is shaded. From part (f), we see that the string added after the symbol *c* due to the three triples is *baccbaccgi*. This agrees with the sequence mentioned at the beginning of our discussion about the decoding process. We thus conclude that the decoding process has correctly decoded the encoded sequence from the last encoded symbol and the received triples.

6.4.4.3 Summary of the LZ77 Approach

The sliding window consists of two parts: the search buffer and the look-ahead buffer. The most recently encoded portion of the input text stream is contained in the search buffer, while the portion of the text that needs to be encoded immediately is in the look-ahead buffer. The first symbol in the look-ahead buffer, located to the right of the boundary between the two buffers, is the symbol

or the beginning of a string of symbols to be encoded at the moment. Let us call it the symbol s. The size of the search buffer is usually much larger than that of the look-ahead buffer.

In encoding, the search pointer moves to the left, away from the symbol s, to find a match of the symbol s in the search buffer. Once a match is found, the encoding process will further determine the length of the matched string. When there are multiple matches, the match that produces the longest matched string is chosen. The match is denoted by a triple <i, j, k>. The first item in the triple, "i", is the offset, which is the distance between the pointer pointing to the symbol giving the maximum match and the symbol s. The second item, "j", is the length of the matched string. The third item, "k", is the codeword of the symbol following the matched string in the look-ahead buffer. The sliding window is then shifted to the right by j+1 positions before the next coding step takes place.

When there is no matching in the search buffer, the triple is represented by <0, 0, C(s)>, where C(s) is the codeword assigned to the symbol s. The sliding window is then shifted to the right by one position.

The sliding window is shifted along the input text stream during the encoding process. The symbol s moves from the beginning symbol to the ending symbol of the input text stream.

At the very beginning, the content of the search buffer can be arbitrarily selected. For instance, the symbols in the search buffer may all be the space symbol.

Let us denote the size of the search buffer by SB, the size of the look-ahead buffer by L, and the size of the source alphabet by A. Assume that the natural binary code is used. Then we see that the LZ77 approach encodes variable-length strings of symbols with fixed-length codewords. Specifically, the offset "i" is of coding length $\lceil \log_2 SB \rceil$, the length of matched string "j" is of coding length $\lceil \log_2 (SB + L) \rceil$, and the codeword "k" is of coding length $\lceil \log_2 (A) \rceil$, where the sign $\lceil a \rceil$ denotes the smallest integer larger than a.

The length of the matched string is equal to $\lceil \log_2 (SB + L) \rceil$ because the search for the maximum matching can enter into the look-ahead buffer as shown in Example 6.2.

The decoding process is simpler than the encoding process since there are no comparisons involved in the decoding.

The most recently encoded symbols in the search buffer serve as the dictionary used in the LZ77 approach. The merit of doing so is that the dictionary is well adapted to the input text. The limitation of the approach is that if the distance between the repeated patterns in the input text stream is larger than the size of the search buffer, then the approach cannot utilize the structure to compress the text. A vivid example can be found in (Sayood, 1996).

A window with a moderate size, say, $SB + L \leq 8192$, can compress a variety of texts well. Several reasons have been analyzed by Bell et al. (1990).

Many variations have been made to improve coding efficiency of the LZ77 approach. The LZ77 produces a triple in each encoding step; i.e., the offset (position of the matched string), the length of the matched string, and the codeword of the symbol following the matched string. The transmission of the third item in each coding step is not efficient. This is true especially at the beginning of coding. A variant of the LZ77, referred to as the LZSS algorithm, improves this inefficiency.

6.4.5 LZ78 Algorithms

6.4.5.1 Introduction

As mentioned above, the LZ77 algorithms use a sliding window of fixed size, and both the search buffer and the look-ahead buffer have a fixed size. This means that if the distance between two repeated patterns is larger than the size of the search buffer, the LZ77 algorithms cannot work efficiently. The fixed size of both the buffers implies that the matched string cannot be longer than the sum of the sizes of the two buffers, placing another limitation on coding efficiency. Increasing the sizes of the search buffer and the look-ahead buffer seemingly will resolve the problem. A close

look, however, reveals that it also leads to increases in the number of bits required to encode the offset and matched string length, as well as an increase in processing complexity.

The LZ78 algorithms (Ziv and Lempel, 1978) eliminate the use of the sliding window. Instead, these algorithms use the encoded text as a dictionary which, potentially, does not have a fixed size. Each time a pointer (token) is issued, the encoded string is included in the dictionary. Theoretically, the LZ78 algorithms reach optimal performance as the encoded text stream approaches infinity. In practice, however, as mentioned above with respect to the LZ77, a very large dictionary will affect coding efficiency negatively. Therefore, once a preset limit to the dictionary size has been reached, either the dictionary is fixed for the future (if the coding efficiency is good), or it is reset to zero, i.e., it must be restarted.

Instead of the triples used in the LZ77, only pairs are used in the LZ78. Specifically, only the position of the pointer to the matched string and the symbol following the matched string need to be encoded. The length of the matched string does not need to be encoded since both the encoder and the decoder have exactly the same dictionary, i.e., the decoder knows the length of the matched string.

6.4.5.2 Encoding and Decoding

Like the discussion of the LZ77 algorithms, we will go through an example to describe the LZ78 algorithms.

Example 6.3

Consider the text stream: *baccbaccacbcabccbbacc*. Table 6.4 shows the coding process. We see that for the first three symbols there is no match between the individual input symbols and the entries in the dictionary. Therefore, the doubles are, respectively, <0, C(b)>, <0, C(a)>, and <0, C(c)>, where 0 means no match, and C(b), C(a), and C(c) represent the codewords of *b*, *a*, and *c*, respectively. After symbols b, a, c, comes *c*, which finds a match in the dictionary (the third entry). Therefore, the next symbol *b* is combined to be considered. Since the string *cb* did not appear before, it is encoded as a double and it is appended as a new entry into the dictionary. The first item in the double is the index of the matched entry *c*, 3, the second item is the index/codeword of the symbol following the match *b*, 1. That is, the double is <3, 1>. The following input symbol is *a*, which appeared in the dictionary. Hence, the next symbol *c* is taken into consideration. Since the string *ac* is not an entry of the dictionary, it is encoded with a double. The first item in the double is the index of symbol *a*, 2; the second item is the index of symbol *c*, 3, i.e., <2, 3>. The encoding proceeds in this way. Take a look at Table 6.4. In general, as the encoding proceeds, the entries in the dictionary become longer and longer. First, entries with single symbols come out, but later, more and more entries with two symbols show up. After that, more and more entries with three symbols appear. This means that coding efficiency is increasing.

Now consider the decoding process. Since the decoder knows the rule applied in the encoding, it can reconstruct the dictionary and decode the input text stream from the received doubles. When the first double <0, C(b)> is received, the decoder knows that there is no match. Hence, the first entry in the dictionary is *b*. So is the first decoded symbol. From the second double <0, C(a)>, symbol *a* is known as the second entry in the dictionary as well as the second decoded symbol. Similarly, the next entry in the dictionary and the next decoded symbol are known as *c*. When the following double <3, 1> is received. The decoder knows from two items, 3 and 1, that the next two symbols are the third and the first entries in the dictionary. This indicates that the symbols *c* and *b* are decoded, and the string *cb* becomes the fourth entry in the dictionary.

We omit the next two doubles and take a look at the double <4, 3>, which is associated with Index 7 in Table 6.4. Since the first item in the double is 4, it means that the maximum matched string is *cb*, which is associated with Index 4 in Table 6.4. The second item in the double, 3, implies that the symbol following the match is the third entry *c*. Therefore the decoder decodes a string *cbc*. Also the string *cbc* becomes the seventh entry in the reconstructed dictionary. In this way, the

TABLE 6.4
An Encoding Example Using the LZ78 Algorithm

Index	Doubles	Encoded Symbols
1	< 0, C(b) >	b
2	< 0, C(a) >	a
3	< 0, C(c) >	c
4	< 3, 1 >	cb
5	< 2, 3 >	ac
6	< 3, 2 >	ca
7	< 4, 3 >	cbc
8	< 2, 1 >	ab
9	< 3, 3 >	cc
10	< 1, 1 >	bb
11	< 5, 3 >	acc

decoder can reconstruct the exact same dictionary as that established by the encoder and decode the input text stream from the received doubles.

6.4.5.3 LZW Algorithm

Both the LZ77 and LZ78 approaches, when published in 1977 and 1978, respectively, were theory oriented. The effective and practical improvement over the LZ78 by Welch (1984) brought much attention to the LZ dictionary coding techniques. The resulting algorithm is referred to the LZW algorithm. It removed the second item in the double (the index of the symbol following the longest matched string) and, hence, it enhanced coding efficiency. In other words, the LZW only sends the indexes of the dictionary to the decoder. For the purpose, the LZW first forms an initial dictionary, which consists of all the individual source symbols contained in the source alphabet. Then, the encoder examines the input symbol. Since the input symbol matches to an entry in the dictionary, its succeeding symbol is cascaded to form a string. The cascaded string does not find a match in the initial dictionary. Hence, the index of the matched symbol is encoded and the enlarged string (the matched symbol followed by the cascaded symbol) is listed as a new entry in the dictionary. The encoding process continues in this manner.

For the encoding and decoding processes, let us go through an example to see how the LZW algorithm can encode only the indexes and the decoder can still decode the input text string.

Example 6.4

Consider the following input text stream: *accbadaccbaccbacc*. We see that the source alphabet is $S = \{a,b,c,d,\}$. The top portion of Table 6.5 (with indexes 1,2,3,4) gives a possible initial dictionary used in the LZW. When the first symbol *a* is input, the encoder finds that it has a match in the dictionary. Therefore the next symbol *c* is taken to form a string *ac*. Because the string *ac* is not in the dictionary, it is listed as a new entry in the dictionary and is given an index, 5. The index of the matched symbol *a*, 1, is encoded. When the second symbol, *c*, is input the encoder takes the following symbol *c* into consideration because there is a match to the second input symbol *c* in the dictionary. Since the string *cc* does not match any existing entry, it becomes a new entry in the dictionary with an index, 6. The index of the matched symbol (the second input symbol), *c*, is encoded. Now consider the third input symbol *c*, which appeared in the dictionary. Hence, the following symbol *b* is cascaded to form a string *cb*. Since the string *cb* is not in the dictionary, it becomes a new entry in the dictionary and is given an index, 7. The index of matched symbol *c*, 3, is encoded. The process proceeds in this fashion.

TABLE 6.5
An Example of the Dictionary Coding Using the LZW Algorithm

Index	Entry	Input Symbols	Encoded Index
1	a	Initial dictionary	
2	b		
3	c		
4	d		
5	ac	a	1
6	cc	c	3
7	cb	c	3
8	ba	b	2
9	ad	a	1
10	da	d	4
11	acc	a,c	5
12	cba	c,b	7
13	accb	a,c,c	11
14	bac	b,a,	8
15	cc...	c,c,...	

Take a look at entry 11 in the dictionary shown in Table 6.5. The input symbol at this point is *a*. Since it has a match in the previous entries, its next symbol *c* is considered. Since the string *ac* appeared in entry 5, the succeeding symbol *c* is combined. Now the new enlarged string becomes *acc* and it does not have a match in the previous entries. It is thus added to the dictionary. And a new index, 11, is given to the string *acc*. The index of the matched string *ac*, 5, is encoded and transmitted. The final sequence of encoded indexes is 1, 3, 3, 2, 1, 4, 5, 7, 11, 8. Like the LZ78, the entries in the dictionary become longer and longer in the LZW algorithm. This implies high coding efficiency since long strings can be represented by indexes.

Now let us take a look at the decoding process to see how the decoder can decode the input text stream from the received index. Initially, the decoder has the same dictionary (the top four rows in Table 6.5) as that in the encoder. Once the first index 1 comes, the decoder decodes a symbol *a*. The second index is 3, which indicates that the next symbol is *c*. From the rule applied in encoding, the decoder knows further that a new entry *ac* has been added to the dictionary with an index 5. The next index is 3. It is known that the next symbol is also *c*. It is also known that the string *cc* has been added into the dictionary as the sixth entry. In this way, the decoder reconstructs the dictionary and decodes the input text stream.

6.4.5.4 Summary

The LZW algorithm, as a representative of the LZ78 approach, is summarized below.

The initial dictionary contains the indexes for all the individual source symbols. At the beginning of encoding, when a symbol is input, since it has a match in the initial dictionary, the next symbol is cascaded to form a two-symbol string. Since the two-symbol string cannot find a match in the initial dictionary, the index of the former symbol is encoded and transmitted, and the two-symbol string is added to the dictionary with a new, incremented index. The next encoding step starts with the latter symbol of the two.

In the middle, the encoding process starts with the last symbol of the latest added dictionary entry. Since it has a match in the previous entries, its succeeding symbol is cascaded after the symbol to form a string. If this string appeared before in the dictionary (i.e., the string finds a

match), the next symbol is cascaded as well. This process continues until such an enlarged string cannot find a match in the dictionary. At this moment, the index of the last matched string (the longest match) is encoded and transmitted, and the enlarged and unmatched string is added into the dictionary as a new entry with a new, incremented index.

Decoding is a process of transforming the index string back to the corresponding symbol string. In order to do so, however, the dictionary must be reconstructed in exactly the same way as that established in the encoding process. That is, the initial dictionary is constructed first in the same way as that in the encoding. When decoding the index string, the decoder reconstructs the same dictionary as that in the encoder according to the rule used in the encoding.

Specifically, at the beginning of the decoding, after receiving an index, a corresponding single symbol can be decoded. Via the next received index, another symbol can be decoded. From the rule used in the encoding, the decoder knows that the two symbols should be cascaded to form a new entry added into the dictionary with an incremented index. The next step in the decoding will start from the latter symbol among the two symbols.

Now consider the middle of the decoding process. The presently received index is used to decode a corresponding string of input symbols according to the reconstructed dictionary at the moment. (Note that this string is said to be with the present index.) It is known from the encoding rule that the symbols in the string associated with the next index should be considered. (Note that this string is said to be with the next index.) That is, the first symbol in the string with the next index should be appended to the last symbol in the string with the present index. The resultant combination, i.e., the string with the present index followed by the first symbol in the string with the next index, cannot find a match to an entry in the dictionary. Therefore, the combination should be added to the dictionary with an incremented index. At this moment, the next index becomes the new present index, and the index following the next index becomes the new next index. The decoding process then proceeds in the same fashion in a new decoding step.

Compared with the LZ78 algorithm, the LZW algorithm eliminates the necessity of having the second item in the double, an index/codeword of the symbol following a matched string. That is, the encoder only needs to encode and transmit the first item in the double. This greatly enhances the coding efficiency and reduces the complexity of the LZ algorithm.

6.4.5.5 Applications

The CCITT Recommendation V.42 bis is a data compression standard used in modems that connect computers with remote users via the GSTN. In the compressed mode, the LZW algorithm is recommended for data compression.

In image compression, the LZW finds its application as well. Specifically, it is utilized in the graphic interchange format (GIF) which was created to encode graphical images. GIF is now also used to encode natural images, though it is not very efficient in this regard. For more information, readers are referred to Sayood (1996). The LZW algorithm is also used in the UNIX Compress command.

6.5 INTERNATIONAL STANDARDS FOR LOSSLESS STILL IMAGE COMPRESSION

In the previous chapter, we studied Huffman and arithmetic coding techniques. We also briefly discussed the international standard for bilevel image compression, known as the JBIG. In this chapter, so far we have discussed another two coding techniques: the run-length and dictionary coding techniques. We also introduced the international standards for facsimile compression, in which the techniques known as the MH, MR, and MMR were recommended. All of these techniques involve lossless compression. In the next chapter, the international still image coding standard JPEG will be introduced. As we will see, the JPEG has four different modes. They can be divided into

two compression categories: lossy and lossless. Hence, we can talk about the lossless JPEG. Before leaving this chapter, however, we briefly discuss, compare, and summarize various techniques used in the international standards for lossless still image compression. For more details, readers are referred to an excellent survey paper by Arps and Truong (1994).

6.5.1 Lossless Bilevel Still Image Compression

6.5.1.1 Algorithms

As mentioned above, there are four different international standard algorithms falling into this category.

MH (Modified Huffman coding) — This algorithm is defined in CCITT Recommendation T.4 for facsimile coding. It uses the 1-D run-length coding technique followed by the "modified" Huffman coding technique.

MR (Modified READ [Relative Element Address Designate] coding) — Defined in CCITT Recommendation T.4 for facsimile coding. It uses the 2-D run-length coding technique followed by the "modified" Huffman coding technique.

MMR (Modified Modified READ coding) — Defined in CCITT Recommendation T.6. It is based on MR, but is modified to maximize compression.

JBIG (Joint Bilevel Image experts Group coding) — Defined in CCITT Recommendation T.82. It uses an adaptive 2-D coding model, followed by an adaptive arithmetic coding technique.

6.5.1.2 Performance Comparison

The JBIG test image set was used to compare the performance of the above-mentioned algorithms. The set contains scanned business documents with different densities, graphic images, digital halftones, and mixed (document and halftone) images.

Note that digital halftones, also named (digital) halftone images, are generated by using only binary devices. Some small black units are imposed on a white background. The units may assume different shapes: a circle, a square, and so on. The more dense the black units in a spot of an image, the darker the spot appears. The digital halftoning method has been used for printing gray-level images in newspapers and books. Digital halftoning through character overstriking, used to generate digital images in the early days for the experimental work associated with courses on digital image processing, has been described by Gonzalez and Woods (1992).

The following two observations on the performance comparison were made after the application of the several techniques to the JBIG test image set.

1. For bilevel images excluding digital halftones, the compression ratio achieved by these techniques ranges from 3 to 100. The compression ratio increases monotonically in the order of the following standard algorithms: MH, MR, MMR, JBIG.
2. For digital halftones, MH, MR, and MMR result in data expansion, while JBIG achieves compression ratios in the range of 5 to 20. This demonstrates that among the techniques, JBIG is the only one suitable for the compression of digital halftones.

6.5.2 Lossless Multilevel Still Image Compression

6.5.2.1 Algorithms

There are two international standards for multilevel still image compression:

JBIG (Joint Bilevel Image experts Group coding) — Defined in CCITT Recommendation T.82. It uses an adaptive arithmetic coding technique. To encode multilevel images, the JIBG decomposes multilevel images into bit-planes, then compresses these bit-planes using its bilevel

image compression technique. To further enhance the compression ratio, it uses Gary coding to represent pixel amplitudes instead of weighted binary coding.

JPEG (Joint Photographic (image) Experts Group coding) — Defined in CCITT Recommendation T.81. For lossless coding, it uses the differential coding technique. The predictive error is encoded using either Huffman coding or adaptive arithmetic coding techniques.

6.5.2.2 Performance Comparison

A set of color test images from the JPEG standards committee was used for performance comparison. The luminance component (Y) is of resolution 720 × 576 pixels, while the chrominance components (U and V) are of 360 × 576 pixels. The compression ratios calculated are the combined results for all the three components. The following observations have been reported.

1. When quantized in 8 bits per pixel, the compression ratios vary much less for multilevel images than for bilevel images, and are roughtly equal to 2.
2. When quantized with 5 bits per pixel down to 2 bits per pixel, compared with the lossless JPEG the JBIG achieves an increasingly higher compression ratio, up to a maximum of 29%.
3. When quantized with 6 bits per pixel, JBIG and lossless JPEG achieve similar compression ratios.
4. When quantized with 7 bits per pixel to 8 bits per pixel, the lossless JPEG achieves a 2.4 to 2.6% higher compression ratio than JBIG.

6.6 SUMMARY

Both Huffman coding and arithmetic coding, discussed in the previous chapter, are referred to as variable-length coding techniques, since the lengths of codewords assigned to different entries in a source alphabet are different. In general, a codeword of a shorter length is assigned to an entry with higher occurrence probabilities. They are also classified as fixed-length to variable-length coding techniques (Arps, 1979), since the entries in a source alphabet have the same fixed length. Run-length coding (RLC) and dictionary coding, the focus of this chapter, are opposite, and are referred to as variable-length to fixed-length coding techniques. This is because the runs in the RLC and the string in the dictionary coding are variable and are encoded with codewords of the same fixed length.

Based on RLC, the international standard algorithms for facsimile coding, MH, MR, and MMR have worked successfully except for dealing with digital halftones. That is, these algorithms result in data expansion when applied to digital halftones. The JBIG, based on an adaptive arithmetic coding technique, not only achieves a higher coding efficiency than MH, MR, and MMR for facsimile coding, but also compresses the digital halftones effectively.

Note that 1-D RLC utilizes the correlation between pixels within a scan line, whereas 2-D RLC utilizes the correlation between pixels within a few scan lines. As a result, 2-D RLC can obtain higher coding efficiency than 1-D RLC. On the other hand, 2-D RLC is more susceptible to transmission errors than 1-D RLC.

In text compression, the dictionary-based techniques have proven to be efficient. All the adaptive dictionary-based algorithms can be classified into two groups. One is based on a work by Ziv and Lempel in 1977, and another is based on their pioneering work in 1978. They are called the LZ77 and LZ78 algorithms, respectively. With the LZ77 algorithms, a fixed-size window slides through the input text stream. The sliding window consists of two parts: the search buffer and the look-ahead buffer. The search buffer contains the most recently encoded portion of the input text, while the look-ahead buffer contains the portion of the input text to be encoded immediately. For the symbols to be encoded, the LZ77 algorithms search for the longest match in the search buffer. The

information about the match: the distance between the matched string in the search buffer and that in the look-ahead buffer, the length of the matched string, and the codeword of the symbol following the matched string in the look-ahead buffer are encoded. Many improvements have been made in the LZ77 algorithms.

The performance of the LZ77 algorithms is limited by the sizes of the search buffer and the look-ahead buffer. With a finite size for the search buffer, the LZ77 algorithms will not work well in the case where repeated patterns are apart from each other by a distance longer than the size of the search buffer. With a finite size for the sliding window, the LZ77 algorithms will not work well in the case where matching strings are longer than the window. In order to be efficient, however, these sizes cannot be very large.

In order to overcome the problem, the LZ78 algorithms work in a different way. They do not use the sliding window at all. Instead of using the most recently encoded portion of the input text as a dictionary, the LZ78 algorithms use the index of the longest matched string as an entry of the dictionary. That is, each matched string cascaded with its immediate next symbol is compared with the existing entries of the dictionary. If this combination (a new string) does not find a match in the dictionary constructed at the moment, the combination will be included as an entry in the dictionary. Otherwise, the next symbol in the input text will be appended to the combination and the enlarged new combination will be checked with the dictionary. The process continues until the new combination cannot find a match in the dictionary. Among the several variants of the LZ78 algorithms, the LZW algorithm is perhaps the most important one. It only needs to encode the indexes of the longest matched strings to the dictionary. It can be shown that the decoder can decode the input text stream from the given index stream. In doing so, the same dictionary as that established in the encoder needs to be reconstructed at the decoder, and this can be implemented since the same rule used in the encoding is known in the decoder.

The size of the dictionary cannot be infinitely large because, as mentioned above, the coding efficiency will not be high. The common practice of the LZ78 algorithms is to keep the dictionary fixed once a certain size has been reached and the performance of the encoding is satisfactory. Otherwise, the dictionary will be set to empty and will be reconstructed from scratch.

Considering the fact that there are several international standards concerning still image coding (for both bilevel and multilevel images), a brief summary of them and a performance comparison have been presented in this chapter. At the beginning of this chapter, a description of the discrete Markov source and its nth extensions was provided. The Markov source and the autoregressive model serve as important models for the dependent information sources.

6.7 EXERCISES

6-1. Draw the state diagram of a second-order Markov source with two symbols in the source alphabet. That is, $S = \{s_1, s_2\}$. It is assumed that the conditional probabilities are

$$p(s_1|s_1s_1) = p(s_2|s_2s_2) = 0.7,$$

$$p(s_2|s_1s_1) = p(s_1|s_2s_2) = 0.3, \text{ and}$$

$$p(s_1|s_1s_2) = p(s_1|s_2s_1) = p(s_2|s_1s_2) = p(s_2|s_2s_1) = 0.5.$$

6-2. What are the definitions of raster algorithm and area algorithm in binary image coding? To which category does 1-D RLC belong? To which category does 2-D RLC belong?

6-3. What effect does a transmission error have on 1-D RLC and 2-D RLC, respectively? What is the function of the codeword EOL?

6-4. Make a convincing argument that the "modified" Huffman (MH) algorithm reduces the requirement of large storage space.

6-5. Which three different modes does 2-D RLC have? How do you view the vertical mode?

6-6. Using your own words, describe the encoding and decoding processes of the LZ77 algorithms. Go through Example 6.2.

6-7. Using your own words, describe the encoding and decoding processes of the LZW algorithm. Go through Example 6.3.

6-8. Read the reference paper (Arps and Truong, 1994), which is an excellent survey on the international standards for lossless still image compression. Pay particular attention to all the figures and to Table 1.

REFERENCES

Abramson, N. *Information Theory and Coding*, New York: McGraw-Hill, 1963.

Arps, R. B. Binary Image Compression, in *Image Transmission Techniques*, W. K. Pratt (Ed.), New York: Academic Press, 1979.

Arps, R. B. and T. K. Truong, Comparison of international standards for lossless still image compression, *Proc. IEEE*, 82(6), 889-899, 1994.

Bell, T. C., J. G. Cleary, and I. H. Witten, *Text Compression*, Englewood Cliffs, NJ: Prentice-Hall, 1990.

Gonzalez, R. C. and R. E. Woods, *Digital Image Processing*, Reading, MA: Addison-Wesley, 1992.

Hunter, R. and A. H. Robinson, International digital facsimile coding standards, *Proc. IEEE*, 68(7), 854-867, 1980.

Laemmel, A. E. Coding Processes for Bandwidth Reduction in Picture Transmission, Rep. R-246-51, PIB-187, Microwave Res. Inst., Polytechnic Institute of Brooklyn, New York.

Nelson, M. and J.-L. Gailly, *The Data Compression Book*, 2nd ed., New York: M&T Books, 1995.

Sayood, K. *Introduction to Data Compression*, San Francisco, CA: Morgan Kaufmann Publishers, 1996.

Shannon, C. E. and W. Weaver, *The Mathematical Theory of Communication*, Urbana, IL: University of Illinois Press, 1949.

Welch, T. A technique for high-performance data compression, *IEEE Trans. Comput.*, 17(6), 8-19, 1984.

Ziv, J. and A. Lempel, A universal algorithm for sequential data compression, *IEEE Trans. Inf. Theory*, 23(3), 337-343, 1977.

Ziv, J. and A. Lempel, Compression of individual sequences via variable-rate coding, *IEEE Trans. Inf. Theory*, 24(5), 530-536, 1978.

Section II

Still Image Compression

7 Still Image Coding Standard: JPEG

In this chapter, the JPEG standard is introduced. This standard allows for lossy and lossless encoding of still images and four distinct modes of operation are supported: sequential DCT-based mode, progressive DCT-based mode, lossless mode and hierarchical mode.

7.1 INTRODUCTION

Still image coding is an important application of data compression. When an analog image or picture is digitized, each pixel is represented by a fixed number of bits, which correspond to a certain number of gray levels. In this uncompressed format, the digitized image requires a large number of bits to be stored or transmitted. As a result, compression become necessary due to the limited communication bandwidth or storage size. Since the mid-1980s, the ITU and ISO have been working together to develop a joint international standard for the compression of still images. Officially, JPEG [jpeg] is the ISO/IEC international standard 10918-1; digital compression and coding of continuous-tone still images, or the ITU-T Recommendation T.81. JPEG became an international standard in 1992. The JPEG standard allows for both lossy and lossless encoding of still images. The algorithm for lossy coding is a DCT-based coding scheme. This is the baseline of JPEG and is sufficient for many applications. However, to meet the needs of applications that cannot tolerate loss, e.g., compression of medical images, a lossless coding scheme is also provided and is based on a predictive coding scheme. From the algorithmic point of view, JPEG includes four distinct modes of operation, namely, sequential DCT-based mode, progressive DCT-based mode, lossless mode, and hierarchical mode. In the following sections, an overview of these modes is provided. Further technical details can be found in the books by Pennelbaker and Mitchell (1992) and Symes (1998).

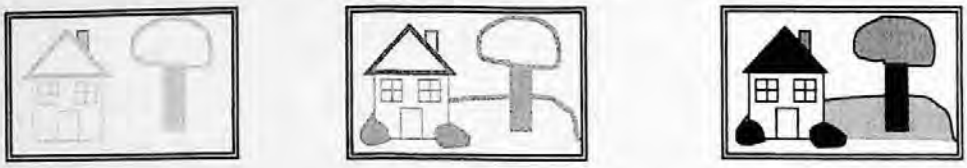
In the sequential DCT-based mode, an image is first partitioned into blocks of 8×8 pixels. The blocks are processed from left to right and top to bottom. The 8×8 two-dimensional Forward DCT is applied to each block and the 8×8 DCT coefficients are quantized. Finally, the quantized DCT coefficients are entropy encoded and output as part of the compressed image data.

In the progressive DCT-based mode, the process of block partitioning and Forward DCT transform is the same as in the sequential DCT-based mode. However, in the progressive mode, the quantized DCT coefficients are first stored in a buffer before the encoding is performed. The DCT coefficients in the buffer are then encoded by a multiple scanning process. In each scan, the quantized DCT coefficients are partially encoded by either spectral selection or successive approximation. In the method of spectral selection, the quantized DCT coefficients are divided into multiple spectral bands according to a zigzag order. In each scan, a specified band is encoded. In the method of successive approximation, a specified number of most significant bits of the quantized coefficients are first encoded and the least significant bits are then encoded in subsequent scans.

The difference between sequential coding and progressive coding is shown in Figure 7.1. In the sequential coding an image is encoded part by part according to the scanning order, while in the progressive coding the image is encoded by a multiscanning process and in each scan the full image is encoded to a certain quality level.

As mentioned earlier, lossless coding is achieved by a predictive coding scheme. In this scheme, three neighboring pixels are used to predict the current pixel to be coded. The prediction difference

(a) Sequential coding



(b) Progressive coding

FIGURE 7.1 (a) Sequential coding, (b) progressive coding.

is entropy coded using either Huffman or arithmetic coding. Since the prediction is not quantized, the coding is lossless.

Finally, in the hierarchical mode, an image is first spatially down-sampled to a multilayered pyramid, resulting in a sequence of frames as shown in Figure 7.2. This sequence of frames is encoded by a predictive coding scheme. Except for the first frame, the predictive coding process is applied to the differential frames, i.e., the differences between the frame to be coded and the predictive reference frame. It is important to note that the reference frame is equivalent to the previous frame that would be reconstructed in the decoder. The coding method for the difference frame may use the DCT-based coding method, the lossless coding method, or the DCT-based processes with a final lossless process. Down-sampling and up-sampling filters are used in the hierarchical mode. The hierarchical coding mode provides a progressive presentation similar to the progressive DCT-based mode, but is also useful in the applications that have multiresolution requirements. The hierarchical coding mode also provides the capability of progressive coding to a final lossless stage.

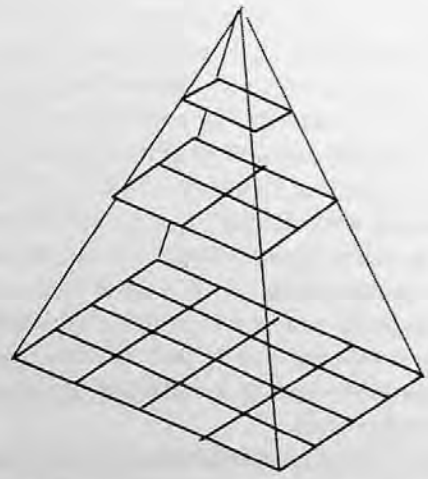


FIGURE 7.2 Hierarchical multiresolution encoding.

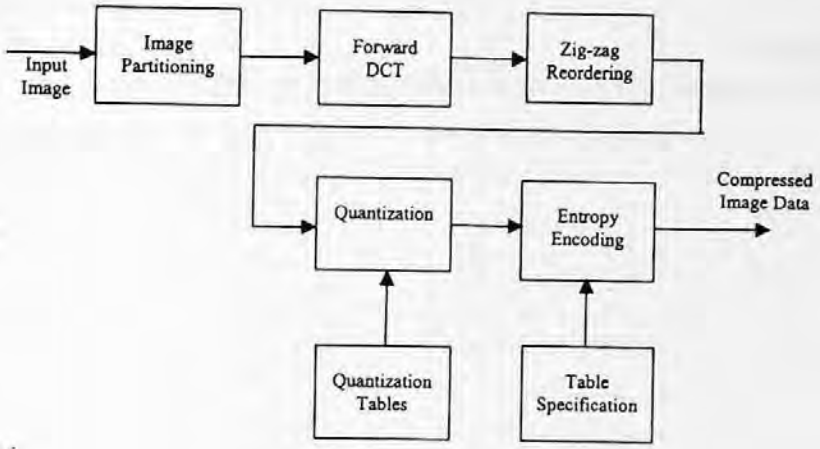


FIGURE 7.3 Block diagram of a sequential DCT-based encoding process.

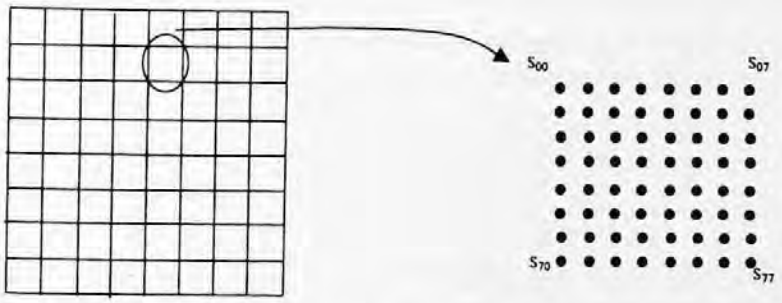


FIGURE 7.4 Partitioning to 8 × 8 blocks.

7.2 SEQUENTIAL DCT-BASED ENCODING ALGORITHM

The sequential DCT-based coding algorithm is the baseline algorithm of the JPEG coding standard. A block diagram of the encoding process is shown in Figure 7.3. As shown in Figure 7.4, the digitized image data are first partitioned into blocks of 8 × 8 pixels. The two-dimensional forward DCT is applied to each 8 × 8 block. The two-dimensional forward and inverse DCT of 8 × 8 block are defined as follows:

$$\text{FDCT:} \quad S_{uv} = \frac{1}{4} C_u C_v \sum_{i=0}^{7} \sum_{j=0}^{7} s_{ij} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16}$$

$$\text{IDCT:} \quad s_{ij} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{uv} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} \tag{7.1}$$

$$C_u C_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

where s_{ij} is the value of the pixel at position (i,j) in the block, and S_{uv} is the transformed (u,v) DCT coefficient.

TABLE 7.1
Two Examples of Quantization Tables Used by JPEG

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Luminance quantization table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Chrominance quantization table

After the forward DCT, quantization of the transformed DCT coefficients is performed. Each of the 64 DCT coefficients is quantized by a uniform quantizer:

$$S_{quv} = round\left(\frac{S_{uv}}{Q_{uv}}\right) \tag{7.2}$$

where the S_{quv} is the quantized value of the DCT coefficient, S_{uv}, and Q_{uv} is the quantization step obtained from the quantization table. There are four quantization tables that may be used by the encoder, but there is no default quantization table specified by the standard. Two particular quantization tables are shown in Table 7.1.

At the decoder, the dequantization is performed as follows:

$$R_{quv} = S_{quv} \times Q_{uv} \tag{7.3}$$

where R_{quv} is the value of the dequantized DCT coefficient. After quantization, the DC coefficient, S_{q00}, is treated separately from the other 63 AC coefficients. The DC coefficients are encoded by a predictive coding scheme. The encoded value is the difference (*DIFF*) between the quantized DC coefficient of the current block (S_{q00}) and that of the previous block of the same component (*PRED*):

$$DIFF = S_{q00} - PRED \tag{7.4}$$

The value of *DIFF* is entropy coded with Huffman tables. More specifically, the two's complement of the possible *DIFF* magnitudes are grouped into 12 categories, "SSSS". The Huffman codes for these 12 difference categories and additional bits are shown in the Table 7.2.

For each nonzero category, additional bits are added to the codeword to uniquely identify which difference within the category actually occurred. The number of additional bits is defined by "SSSS" and the additional bits are appended to the least significant bit of the Huffman code (most significant bit first) according to the following rule. If the difference value is positive, the "SSSS" low-order bits of *DIFF* are appended; if the difference value is negative, then the "SSSS" low-order bits of *DIFF-1* are appended. As an example, the Huffman tables used for coding the luminance and chrominance DC coefficients are shown in Tables 7.3 and 7.4, respectively. These two tables have been developed from the average statistics of a large set of images with 8-bit precision.

TABLE 7.2
Huffman Coding of DC Coefficients

SSSS	DIFF Values	Additional Bits
0	0	–
1	–1,1	0,1
2	–3,–2,2,3	00,01,10,11
3	–7,...,–4,4,...,7	000,...,011,100,..,111
4	–15,...,–8,8,...,15	0000,..0111,1000,...,1111
5	–31,...,–16,16,...,31	00000,...,01111,10000,...,11111
6	–63,...–32,32,...63	.....
7	–127,...,–64,64,...,127	.....
8	–255,...,–128,128,...,255	.....
9	–511,...,–256,256,...,511	.....
10	–1023,...,–512,512,...,1023	.....
11	–2047,...,–1024,1024,...,2047	.....

TABLE 7.3
Huffman Table for Luminance DC Coefficient Differences

Category	Code Length	Codeword
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

In contrast to the coding of DC coefficients, the quantized AC coefficients are arranged to a zigzag order before being entropy coded. This scan order is shown in Figure 7.5.

According to the zigzag scanning order, the quantized coefficients can be represented as:

$$ZZ(0) = S_{q00},\ ZZ(1) = S_{q01},\ ZZ(2) = S_{q10},\ \ldots.,\ ZZ(63) = S_{q77}. \tag{7.5}$$

Since many of the quantized AC coefficients become zero, they can be very efficiently encoded by exploiting the run of zeros. The run-length of zeros are identified by the nonzero coefficients. An 8-bit code 'RRRRSSSS' is used to represent the nonzero coefficient. The four least significant bits, 'SSSS', define a category for the value of the next nonzero coefficient in the zigzag sequence, which ends the zero run. The four most significant bits, 'RRRR', define the run-length of zeros in the zigzag sequence or the position of the nonzero coefficient in the zigzag sequence. The composite value, RRRRSSSS, is shown in Figure 7.6. The value 'RRRRSSSS' = '11110000' is defined as ZRL, "RRRR" = "1111" represents a run-length of 16 zeros and "SSSS" = "0000" represents a zero amplitude. Therefore, ZRL is used to represent a run-length of 16 zero coefficients followed

TABLE 7.4
Huffman table for chrominance DC coefficient differences

Category	Code Length	Codeword
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	11	11111111110

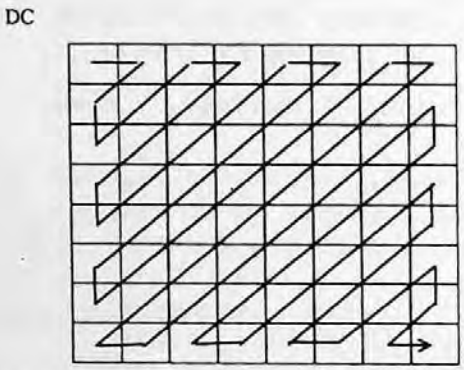


FIGURE 7.5 Zigzag scanning order of DCT coefficients.

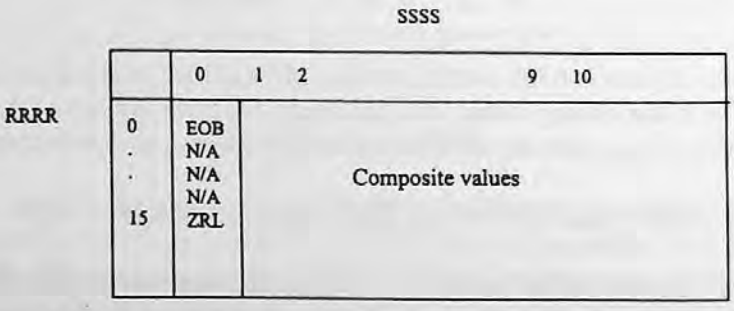


FIGURE 7.6 Two-dimensional value array for Huffman coding.

by a zero-amplitude coefficient, it is not an *abbreviation*. In the case of a run-length of zero coefficients that exceeds 15, multiple symbols will be used. A special value 'RRRRSSSS' = '00000000' is used to code the end-of-block (EOB). An EOB occurs when the remaining coefficients in the block are zeros. The entries marked "N/A" are undefined.

TABLE 7.5
Huffman Coding for AC Coefficients

Category (SSSS)	AC Coefficient Range
1	−1,1
2	−3,−2,2,3
3	−7,...,−4,4,...,7
4	−15,...,−8,8,...,15
5	−31,...,−16,16,...,31
6	−63,...,−32,32,...,63
7	−127,...,−64,.64,...,127
8	−255,...,−128,128,...,255
9	−511,...,−256,256,...,511
10	−1023,..,−512,512,...,1023
11	−2047,...,−1024,1024,...,2047

The composite value, RRRRSSSS, is then Huffman coded. SSSS is actually the number to indicate "category" in the Huffman code table. The coefficient values for each category are shown in Table 7.5.

Each Huffman code is followed by additional bits that specify the sign and exact amplitude of the coefficients. As with the DC code tables, the AC code tables have also been developed from the average statistics of a large set of images with 8-bit precision. Each composite value is represented by a Huffman code in the AC code table. The format for the additional bits is the same as in the coding of DC coefficients. The value of SSSS gives the number of additional bits required to specify the sign and precise amplitude of the coefficient. The additional bits are either the low-order SSSS bits of ZZ(k) when ZZ(k) is positive, or the low-order SSSS bits of ZZ(k)-1 when ZZ(k) is negative. Here, ZZ(k) is the *kth* coefficient in the zigzag scanning order of coefficients being coded. The Huffman tables for AC coefficients can be found in Annex K of the JPEG standard (jpeg) and are not listed here due to space limitations.

As described above, Huffman coding is used as the means of entropy coding. However, an adaptive arithmetic coding procedure can also be used. As with the Huffman coding technique, the binary arithmetic coding technique is also lossless. It is possible to transcode between two systems without either of the FDCT or IDCT processes. Since this transcoding is a lossless process, it does not affect the picture quality of the reconstructed image. The arithmetic encoder encodes a series of binary symbols, zeros or ones, where each symbol represents the possible result of a binary decision. The binary decisions include the choice between positive and negative signs, a magnitude being zero or nonzero, or a particular bit in a sequence of binary digits being zero or one. There are four steps in the arithmetic coding: initializing the statistical area, initializing the encoder, terminating the code string, and adding restart markers.

7.3 PROGRESSIVE DCT-BASED ENCODING ALGORITHM

In progressive DCT-based coding, the input image is first partitioned to blocks of 8×8 pixels. The two-dimensional 8×8 DCT is then applied to each block. The transformed DCT-coefficient data are then encoded with multiple scans. At each scan, a portion of the transformed DCT coefficient data is encoded. This partially encoded data can be reconstructed to obtain a full image size with lower picture quality. The coded data of each additional scan will enhance the reconstructed image quality until the full quality has been achieved at the completion of all scans. Two methods have been used in the JPEG standard to perform the DCT-based progressive coding. These include spectral selection and successive approximation.

In the method of spectral selection, the transformed DCT coefficients are first reordered as a zigzag sequence and then divided into several bands. A frequency band is defined in the scan header by specifying the starting and ending indexes in the zigzag sequence. The band containing the DC coefficient is encoded at the first scan. In the following scan, it is not necessary for the coding procedure to follow the zigzag ordering.

In the method of the successive approximation, the DCT coefficients are first reduced in precision by the point transform. The point transform of the DCT coefficients is an arithmetic shift right by a specified number of bits, or division by a power of 2 (near zero, there is slight difference in truncation of precision between an arithmetic shift and division by 2, see annex K10 of [jpeg]). This specified number is the successive approximation of bit position. To encode using successive approximations, the significant bits of the DCT coefficient are encoded in the first scan, and each successive scan that follows progressively improves the precision of the coefficient by one bit. This continues until full precision is reached.

The principles of spectral selection and successive approximation are shown in Figure 7.7. For both methods, the quantized coefficients are coded with either Huffman or arithmetic codes at each scan. In spectral selection and the first scan of successive approximation for an image, the AC coefficient coding model is similar to that used in the sequential DCT-based coding mode. However, the Huffman code tables are extended to include coding of runs of end-of-bands (EOBs). For distinguishing the end-of-band and end-of-block, a number, n, which is used to indicate the range of run length, is added to the end-of-band (EOBn). The EOBn code sequence is defined as follows. Each EOBn is followed by an extension field, which has the minimum number of bits required to specify the run length. The end-of-band run structure allows efficient coding of blocks which have only zero coefficients. For example, an EOB run of length 5 means that the current block and the next 4 blocks have an end-of-band with no intervening nonzero coefficients. The Huffman coding structure of the subsequent scans of successive approximation for a given image is similar to the coding structure of the first scan of that image. Each nonzero quantized coefficient is described by a composite 8-bit run length-magnitude value of the form: RRRRSSSS. The four most significant bits, RRRR, indicate the number of zero coefficients between the current coefficient and the previously coded coefficient. The four least significant bits, SSSS, give the magnitude category of the nonzero coefficient. The run length-magnitude composite value is Huffman coded. Each Huffman code is followed by additional bits: one bit is used to code the sign of the nonzero coefficient and another bit is used to code the correction, where "0" means no correction and "1" means add one to the decoded magnitude of the coefficient. Although the above technique has been described using Huffman coding, it should be noted that arithmetic encoding can also be used in its place.

7.4 LOSSLESS CODING MODE

In the lossless coding mode, the coding method is spatially based coding instead of DCT-based coding. However, the coding method is extended from the method for coding the DC coefficients in the sequential DCT-based coding mode. Each pixel is coded with a predictive coding method, where the predicted value is obtained from one of three one-dimensional or one of four two-dimensional predictors, which are shown in Figure 7.8.

In Figure 7.8, the pixel to be coded is denoted by x, and the three causal neighbors are denoted by a, b, and c. The predictive value of x, Px, is obtained from three neighbors, a, b, and c in the one of seven ways as listed in Table 7.6.

In Table 7.6, the selection value 0 is only used for differential coding in the hierarchical coding mode. Selections 1, 2, and 3 are one-dimensional predictions and 4, 5, 6, and 7 are two-dimensional predictions. Each prediction is performed with full integer precision, and without clamping of either the underflow or overflow beyond the input bounds. In order to achieve lossless coding, the prediction differences are coded with either Huffman coding or arithmetic coding. The prediction

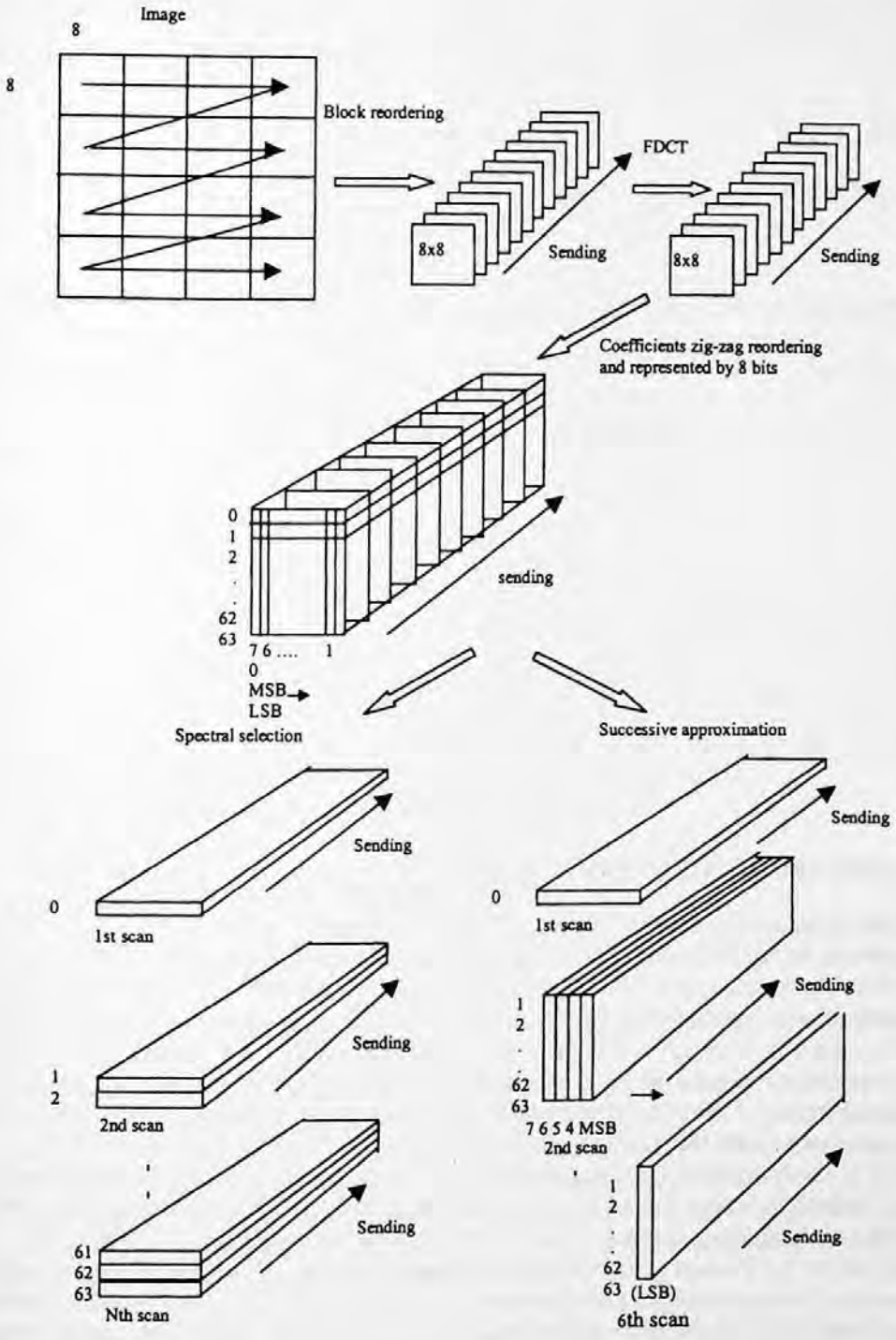


FIGURE 7.7 Progressive coding with spectral selection and successive approximation.

difference values can be from 0 to 2^{16} for 8-bit pixels. The Huffman tables developed for coding DC coefficients in the sequential DCT-based coding mode are used with one additional entry to code the prediction differences. For arithmetic coding, the statistical model defined for the DC coefficients in the sequential DCT-based coding mode is generalized to a two-dimensional form in which differences are conditioned on the pixel to the left and the line above.

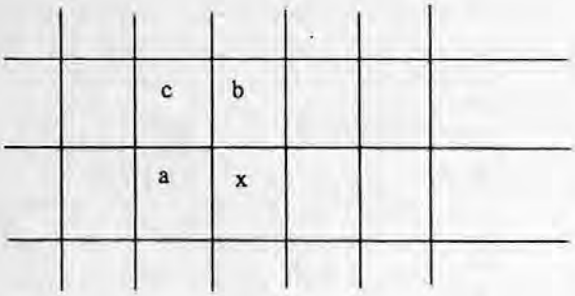


FIGURE 7.8 Spatial relationship between the pixel to be coded and three decoded neighbors.

TABLE 7.6
Predictors for Lossless Coding

Selection-Value	Prediction
0	No prediction (hierarchical mode)
1	Px = a
2	Px = b
3	Px = c
4	Px = a+b-c
5	Px = a + ((b-c)/2)[a]
6	Px = b + ((a-c)/2)[a]
7	Px = (a+b)/2

[a] Shift right arithmetic operation.

7.5 HIERARCHICAL CODING MODE

The hierarchical coding mode provides a progressive coding similar to the progressive DCT-based coding mode, but it offers more functionality. This functionality addresses applications with multi-resolution requirements. In the hierarchical coding mode, an input image frame is first decomposed to a sequence of frames, such as the pyramid shown in Figure 7.2. Each frame is obtained through a down-sampling process, i.e., low-pass filtering followed by subsampling. The first frame (the lowest resolution) is encoded as a nondifferential frame. The following frames are encoded as differential frames, where the differential is with respect to the previously coded frame. Note that an up-sampled version that would be reconstructed in the decoder is used. The first frame can be encoded by the methods of sequential DCT-based coding, spectral selection, method of progressive coding, or lossless coding with either Huffman code or arithmetic code. However, within an image, the differential frames are either coded by the DCT-based coding method, the lossless coding method, or the DCT-based process with a final lossless coding. All frames within the image must use the same entropy coding, either Huffman or arithmetic, with the exception that nondifferential frames coded with the baseline coding may occur in the same image with frames coded with arithmetic coding methods. The differential frames are coded with the same method used for the nondifferential frames except the final frame. The final differential frame for each image may use a differential lossless coding method.

In the hierarchical coding mode, resolution changes in frames may occur. These resolution changes occur if down-sampling filters are used to reduce the spatial resolution of some or all frames of an image. When the resolution of a reference frame does not match the resolution of the frame to be coded, a up-sampling filter is used to increase the resolution of the reference frame. The block diagram of coding of a differential frame is shown in Figure 7.9.

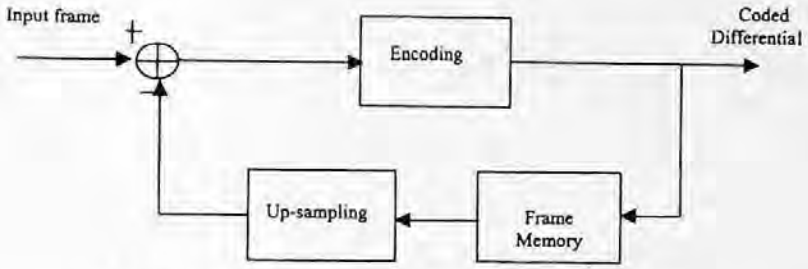


FIGURE 7.9 Hierarchical coding of a differential frame.

The up-sampling filter increases the spatial resolution by a factor of two in both horizontal and vertical directions by using bilinear interpolation of two neighboring pixels. The up-sampling with bilinear interpolation is consistent with the down-sampling filter that is used for the generation of down-sampled frames. It should be noted that the hierarchical coding mode allows one to improve the quality of the reconstructed frames at a given spatial resolution.

7.6 SUMMARY

In this chapter, the still image coding standard, JPEG, has been introduced. The JPEG coding standard includes four coding modes: sequential DCT-based coding mode, progressive DCT-based coding mode, lossless coding mode, and hierarchical coding mode. The DCT-based coding method is probably the one that most of us are familiar with; however, the lossless coding modes in JPEG which use a spatial domain predictive coding process have many interesting applications as well. For each coding mode, entropy coding can be implemented with either Huffman coding or arithmetic coding. JPEG has been widely adopted for many applications.

7.7 EXERCISES

7-1. What is the difference between sequential coding and progressive coding in JPEG? Conduct a project to encode an image with sequence coding and progressive coding, respectively.

7-2. Use the JPEG lossless mode to code several images and explain why different bit rates are obtained.

7-3. Generate a Huffman code table using a set of images with 8-bit precision (aproximately 2~3) using the method presented in Annex C of the JPEG specification. This set of images is called the training set. Use this table to code an image within the training set and an image which is not in the training set, and explain the results.

7-4. Design a three-layer progressive JPEG coder using (a) spectral selection, and (b) progressive approximation (0.3 bits per pixel at the first layer, 0.2 bits per pixel at the second layer, and 0.1 bits per pixel at the third layer).

REFERENCES

Digital compression and coding of continuous-tone still images. Requirements and Guidelines, ISO-/IEC International Standard 10918-1, CCITT T.81, September, 1992.

Pennelbaker, W. B. and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1992.

Symes, P. *Compression: Fundamental Compression Techniques and an Overview of the JPEG and MPEG Compression Systems*, McGraw-Hill, New York, 1998.

8 Wavelet Transform for Image Coding

During the last decade, a number of signal processing applications have emerged using wavelet theory. Among those applications, the most widespread developments have occurred in the area of data compression. Wavelet techniques have demonstrated the ability to provide not only high coding efficiency, but also spatial and quality scalability features. In this chapter, we focus on the utility of the wavelet transform for image data compression applications.

8.1 REVIEW OF THE WAVELET TRANSFORM

8.1.1 Definition and Comparison with Short-Time Fourier Transform

The wavelet transform, as a specialized research field, started over a decade ago (Grossman and Morlet, 1984). To better understand the theory of wavelets, we first give a very short review of the Short-Time Fourier Transform (STFT) since there are some similarities between the STFT and the wavelet transform. As we know, the STFT uses sinusoidal waves as its orthogonal basis and is defined as:

$$F(\omega,\tau)=\int_{-\infty}^{+\infty} f(t)w(t-\tau)e^{-j\omega t}dt \tag{8.1}$$

where $w(t)$ is a time-domain windowing function, the simplest of which is a rectangular window that has a unit value over a time interval and has zero elsewhere. The value τ is the starting position of the window. Thus, the STFT maps a function $f(t)$ into a two-dimensional plane (ω,τ). The STFT is also referred to as Gabor transform (Cohen, 1989). Similar to the STFT, the wavelet transform also maps a time or spatial function into a two-dimensional function in a and τ (ω and τ for STFT). The wavelet transform is defined as follows. Let $f(t)$ be any square integrable function, i.e., it satisfies:

$$\int_{-\infty}^{+\infty} |f(t)|^2 dt < \infty \tag{8.2}$$

The continuous-time wavelet transform of $f(t)$ with respect to a wavelet $\psi(t)$ is defined as:

$$W(a,\tau)=\int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{|a|}}\psi^*\left(\frac{t-\tau}{a}\right)dt \tag{8.3}$$

where a and τ are real variables and * denotes complex conjugation. The wavelet is defined as:

$$\psi_{a\tau}(t)=|a|^{-1/2}\psi\left(\frac{t-\tau}{a}\right) \tag{8.4}$$

The above equation represents a set of functions that are generated from a single function, $\psi(t)$, by dilations and translations. The variable τ represents the time shift and the variable a corresponds to the amount of time-scaling or dilation. If $a > 1$, there is an expansion of $\psi(t)$, while if $0 < a < 1$, there is a contraction of $\psi(t)$. For negative values of a, the wavelet experiences a time reversal in combination with a dilation. The function, $\psi(t)$, is referred to as the mother wavelet and it must satisfy two conditions:

1. The function integrates to zero:

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0 \tag{8.5}$$

2. The function is square integrable, or has finite energy:

$$\int_{-\infty}^{+\infty} \psi|(t)|^2 dt < \infty \tag{8. 6}$$

The continuous-time wavelet transform can now be rewritten as:

$$W(a,\tau) = \int_{-\infty}^{+\infty} f(t)\psi_{a\tau}^{*}(t)dt \tag{8.7}$$

In the following, we give two well-known examples of $\psi(t)$ and their Fourier transforms. The first example is the Morlet (modulated Gaussian) wavelet (Daubechies, 1990),

$$\Psi(\omega) = \sqrt{2\pi}\, e^{\frac{(\omega-\omega_0)^2}{2}} \tag{8.8}$$

and the second example is the Haar wavelet:

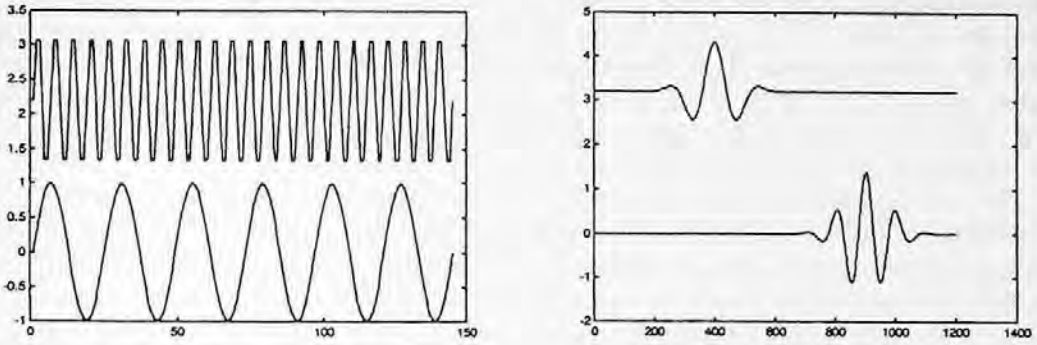
$$\psi = \begin{cases} 1 & 0 \le t \le 1/2 \\ -1 & 1/2 \le t \le 1 \\ 0 & otherwise \end{cases} \tag{8.9}$$

$$\psi(\omega) = je^{-j\frac{\omega}{2}} \frac{sm^2(\omega/4)}{\omega/4}$$

From the above definition and examples, we can find that the wavelets have zero DC value. This is clear from Equation 8.5. In order to have good time localization, the wavelets are usually bandpass signals and they decay rapidly towards zero with time. We can also find several other important properties of the wavelet transform and several differences between STFT and the wavelet transform.

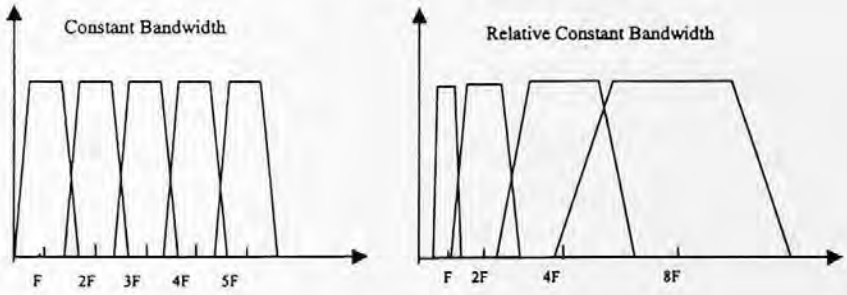
The STFT uses a sinusoidal wave as its basis function. These basis functions keep the same frequency over the entire time interval. In contrast, the wavelet transform uses a particular wavelet as its basis function. Hence, wavelets vary in both position and frequency over the time interval. Examples of two basis functions for the sinusoidal wave and wavelet are shown in Figure 8.1(a) and (b), respectively.

The STFT uses a single analysis window. In contrast, the wavelet transform uses a short time window at high frequencies and a long time window at low frequencies. This is referred to as constant Q-factor filtering or relative constant bandwidth frequency analysis. A comparison of the



(a) (b)

FIGURE 8.1 (a) Two sinusoidal waves, and (b) two wavelets.



(a) (b)

FIGURE 8.2 (a) Constant bandwidth analysis (for Fourier transform), and (b) relative constant bandwidth analysis (for wavelet transform).

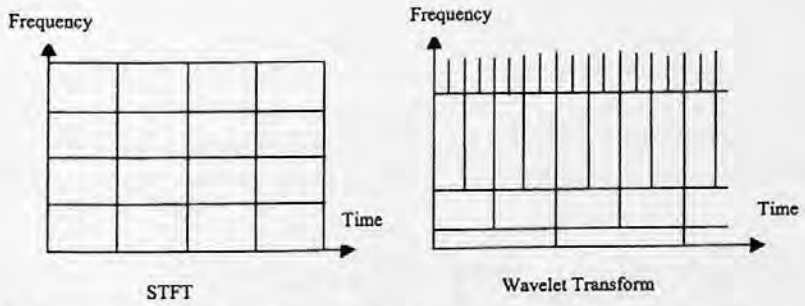


FIGURE 8.3 Comparison of the STFT and the wavelet transform in the time-frequency plane.

constant bandwidth analysis of the STFT and the relative constant bandwidth wavelet transform is shown in Figure 8.2(a) and (b), respectively.

This feature can be further explained with the concept of a time-frequency plane, which is shown in Figure 8.3.

As shown in Figure 8.3, the window size of the STFT in the time domain is always chosen to be constant. The corresponding frequency bandwidth is also constant. In the wavelet transform, the window size in the time domain varies with the frequency. A longer time window is used for

a lower frequency and a shorter time window is used for a higher frequency. This property is very important for image data compression. For image data, the concept of a time-frequency plane becomes a spatial-frequency plane. The spatial resolution of a digital image is measured with pixels, as described in Chapter 15. To overcome the limitations of DCT-based coding, the wavelet transform allows the spatial resolution and frequency bandwidth to vary in the spatial-frequency plane. With this variation, better bit allocation for active and smooth areas can be achieved.

The continuous-time wavelet transform can be considered as a correlation. For fixed a, it is clear from Equation 8.3 that $W(a,\tau)$ is the cross-correlation of functions $f(t)$ with related wavelet conjugate dilated to scale factor a at time lag τ. This is an important property of the wavelet transform for multiresolution analysis of image data. Since the convolution can be seen as a filtering operation, the integral wavelet transform can be seen as a bank of linear filters acting upon $f(t)$. This implies that the image data can be decomposed by a bank of filters defined by the wavelet transform.

The continuous-time wavelet transform can be seen as an operator. First, it has the property of linearity. If we rewrite $W(a,\tau)$ as $W_{a\tau}[f(t)]$, then we have

$$W_{a\tau}[\alpha f(t)+\beta g(t)]=\alpha W_{a\tau}[f(t)]+\beta\, W_{a\tau}[g(t)] \tag{8.10}$$

where α and β are constant scalars. Second, it has the property of translation:

$$W_{a\tau}[f(t-\lambda)]=W(a,\tau-\lambda) \tag{8.11}$$

where λ is a time lag.

Finally, it has the property of scaling

$$W_{a\tau}[f(t/\alpha)]=W(a/\alpha,\tau/\alpha) \tag{8.12}$$

8.1.2 Discrete Wavelet Transform

In the continuous-time wavelet transform, the function $f(t)$ is transformed to a function $W(a,\tau)$ using the wavelet $\psi(t)$ as a basis function. Recall that the two variables a and τ are the dilation and translation, respectively. Now let us to find a means of obtaining the inverse transform, i.e., given $W(a,b)$, find $f(t)$. If we know how to get the inverse transform, we can then represent any arbitrary function $f(t)$ as a summation of wavelets, such as in the Fourier transform and DCT that provide a set of coefficients for reconstructing the original function using sine and cosine as the basis functions. In fact, this is possible if the mother wavelet satisfies the admissibility condition:

$$C=\int_{-\infty}^{+\infty}\frac{|\Psi(\omega)|^2}{|\omega|}d\omega \tag{8.13}$$

where C is a finite constant and $\Psi(\omega)$ is the Fourier transform of the mother wavelet function $\psi(t)$. Then, the inverse wavelet transform is

$$f(t)=\frac{1}{C}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\frac{1}{|a|^2}W(a,\tau)\psi_{a\tau}(t)dad\tau \tag{8.14}$$

The above results can be extended for two-dimensional signals. If $f(x,y)$ is a two-dimensional function, its continuous-time wavelet transform is defined as:

$$W(a,\tau_x,\tau_y)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\psi^*_{a\tau_x\tau_y}(x,y)dxdy \tag{8.15}$$

where τ_x and τ_y specify the transform in two dimensions. The inverse two-dimensional continuous-time wavelet transform is then defined as:

$$f(x,y)=\frac{1}{C}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\frac{1}{|a|^3}W(a,\tau_x,\tau_y)\psi_{a\tau_x\tau_y}(x,y)dad\tau_x d\tau_y \tag{8.16}$$

where the C is defined as in Equation 8.13 and $\psi(x,y)$ is a two-dimensional wavelet

$$\psi_{a\tau_x\tau_y}(x,y)=\frac{1}{|a|}\psi\left(\frac{x-\tau_x}{a},\frac{y-\tau_y}{a}\right) \tag{8.17}$$

For image coding, the wavelet is used to decompose the image data into wavelets. As indicated in the third property of the wavelet transform, the wavelet transform can be viewed as the cross-correlation of the function $f(t)$ and the wavelets $\psi_{a\tau}(t)$. Therefore, the wavelet transform is equivalent to finding the output of a bank of bandpass filters specified by the wavelets of $\psi_{a\tau}(t)$ as shown in Figure 8.4. This process decomposes the input signal into several subbands. Since each subband can be further partitioned, the filter bank implementation of the wavelet transform can be used for multiresolution analysis (MRA). Intuitively, when the analysis is viewed as a filter bank, the time resolution must increase with the central frequency of the analysis filters. This can be exactly obtained by the scaling property of the wavelet transform, where the center frequencies of the bandpass filters increase as the bandwidth becomes wider. Again, the bandwidth becomes wider by reducing the dilation parameter a. It should be noted that such a multiresolution analysis is consistent with the constant Q-factor property of the wavelet transform. Furthermore, the resolution limitation of the STFT does not exist in the wavelet transform since the time-frequency resolutions in the wavelet transform vary, as shown in Figure 8.2(b).

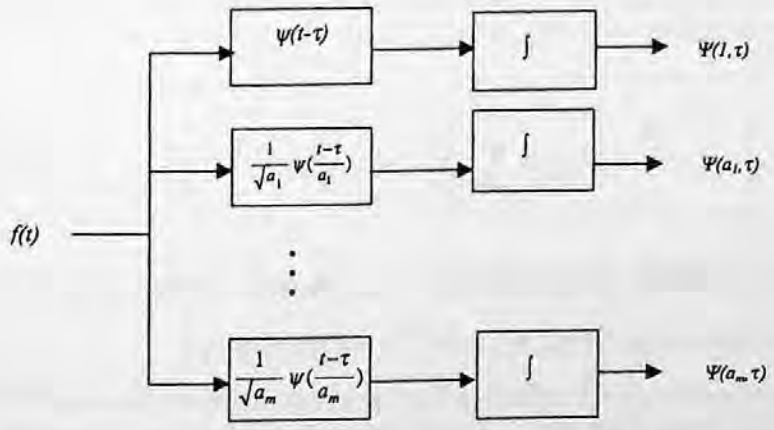


FIGURE 8.4 The wavelet transform implemented with a bank of filters.

For digital image compression, it is preferred to represent $f(t)$ as a discrete superposition sum rather than an integral. With this move to the discrete space, the dilation parameter a in Equation 8.10 takes the values $a = 2^k$ and the translation parameter τ takes the values $\tau = 2^k l$, where both k and l are integers. From Equation 8.4, the discrete version of $\psi_{a\tau}(t)$ becomes:

$$\psi_{kl}(t) = 2^{-\frac{k}{2}} \psi\left(2^{-k}t - l\right) \tag{8.18}$$

Its corresponding wavelet transform can be rewritten as:

$$W(k,l) = \int_{-\infty}^{+\infty} f(t)\psi_{kl}^{*}(t)dt \tag{8.19}$$

and the inverse transform becomes:

$$f(t) = \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} d(k,l)2^{-\frac{k}{2}}\psi\left(2^{-k}t - l\right) \tag{8.20}$$

The values of the wavelet transform at those a and τ are represented by $d(k,l)$:

$$d(k,l) = W(k,l)/C \tag{8.21}$$

The $d(k,l)$ coefficients are referred to as the discrete wavelet transform of the function $f(t)$ (Daubechies, 1992; Vetterli and Kovacevic, 1995). It is noted that the discretization so far is only applied to the parameters a and τ; $d(k,l)$ is still a continuous-time function. If the discretization is further applied to the time domain by letting $t = mT$, where m is an integer and T is the sampling interval (without loss of generality, we assume T = 1), then the discrete-time wavelet transform is defined as:

$$W_d(k,l) = \sum_{m=-\infty}^{+\infty} f(m)\psi_{kl}^{*}(m) \tag{8.22}$$

Of course, the sampling interval has to be chosen according to the Nyquist sampling theorem so that no information is lost in the process of sampling. The inverse discrete-time wavelet transform is then

$$f(m) = \sum_{m=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} d(k,l)2^{-\frac{k}{2}}\psi\left(2^{-k}m - l\right) \tag{8.23}$$

8.2 DIGITAL WAVELET TRANSFORM FOR IMAGE COMPRESSION

8.2.1 Basic Concept of Image Wavelet Transform Coding

From the previous section, we have learned that the wavelet transform has several features that are different from traditional transforms. It is noted from Figure 8.2 that each transform coefficient in the STFT represents a constant interval of time regardless of which band the coefficient belongs to, whereas for the wavelet transform, the coefficients at the course level represent a larger time

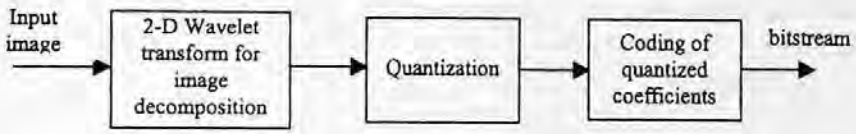


FIGURE 8.5 Block diagram of the image coding with the wavelet transform coding.

interval but a narrower band of frequencies. This feature of the wavelet transform is very important for image coding. In traditional image transform coding, which makes use of the Fourier transform or discrete cosine transform (DCT), one difficult problem is to choose the block size or window width so that statistics computed within that block provide good models of the image signal behavior. The choice of the block size has to be compromised so that it can handle both active and smooth areas. In the active areas, the image data are more localized in the spatial domain, while in the smooth areas the image data are more localized in the frequency domain. With traditional transform coding, it is very hard to reach a good compromise. The main contribution of wavelet transform theory is that it provides an elegant framework in which both statistical behaviors of image data can be analyzed with equal importance. This is because that wavelets can provide a signal representation in which some of the coefficients represent long data lags corresponding to a narrow band or low frequency range, and some of the coefficients represent short data lags corresponding to a wide band or high frequency range. Therefore, it is possible to obtain a good trade-off between spatial and frequency domain with the wavelet representation of image data.

To use the wavelet transform for image coding applications, an encoding process is needed which includes three major steps: image data decomposition, quantization of the transformed coefficients, and coding of the quantized transformed coefficients. A simplified block diagram of this process is shown in Figure 8.5. The image decomposition is usually a lossless process which converts the image data from the spatial domain to frequency domain, where the transformed coefficients are decorrelated. The information loss happens in the quantization step and the compression is achieved in the coding step. To begin the decomposition, the image data are first partitioned into four subbands labeled as LL_1, HL_1, LH_1, and HH_1, as shown in Figure 8.6(a). Each coefficient represents a spatial area corresponding to one-quarter of the original image size. The low frequencies represent a bandwidth corresponding to $0 < |\omega| < \pi/2$, while the high frequencies represent the band $\pi/2 < |\omega| < \pi$. To obtain the next level of decomposition, the LL_1 subband is further decomposed into the next level of four subbands, as shown in Figure 8.6(b). The low frequencies of the second level decomposition correspond to $0 < |\omega| < \pi/4$, while the high frequencies at the second level correspond to $\pi/4 < |\omega| < \pi/2$. This decomposition can be continued

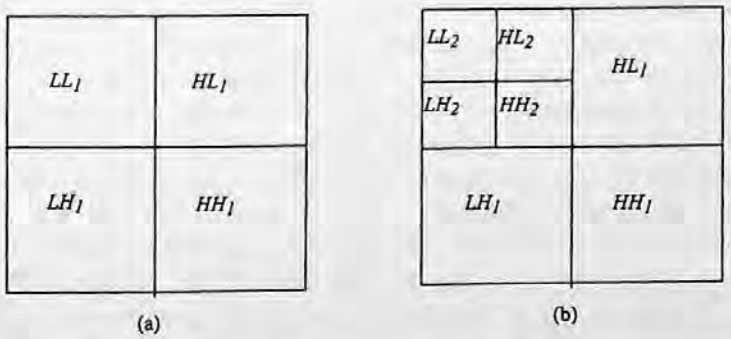


FIGURE 8.6 Two-dimensional wavelet transform. (a) First-level decomposition, and (b) second-level decomposition. (L denotes a low band, H denotes a high band, and the subscript denotes the number of the level. For example, LL_1 denotes the low-low band at level 1.)

to as many levels as needed. The filters used to compute the discrete wavelet transform are generally the symmetric quadrature mirror filters (QMF), as described by Woods (1991). A QMF-pyramid subband decomposition is illustrated in Figure 8.6(b).

During quantization, each subband is quantized differently depending on its importance, which is usually based on its energy or variance (Jayant and Noll, 1984). To reach the predetermined bit rate or compression ratio, coarse quantizers or large quantization steps would be used to quantize the low-energy subbands while the finer quantizers or small quantization steps would be used to quantize the large-energy subbands. This results in fewer bits allocated to those low-energy subbands and more bits for large-energy subbands.

8.2.2 Embedded Image Wavelet Transform Coding Algorithms

As with other transform coding schemes, most wavelet coefficients in the high-frequency bands have very low energy. After quantization, many of these high-frequency wavelet coefficients are quantized to zero. Based on the statistical property of the quantized wavelet coefficients, Huffman coding tables can be designed. Generally, most of the energy in an image is contained in the low-frequency bands. The data structure of the wavelet-transformed coefficients is suitable to exploit this statistical property.

Consider a multilevel decomposition of an image with the discrete wavelet transform, where the lowest levels of decomposition would correspond to the highest-frequency subbands and the finest spatial resolution, and the highest level of decomposition would correspond to the lowest-frequency subband and the coarsest spatial resolution. Arranging the subbands from lowest to highest frequency, we expect a decrease in energy. Also, we expect that if the wavelet-transformed coefficients at a particular level have lower energy, then coefficients at the lower levels or high-frequency subbands, which correspond to the same spatial location, would have smaller energy.

Another feature of the wavelet coefficient data structure is spatial self-similarity across subbands. Several algorithms that have been developed to exploit this and the above-mentioned properties for image coding. Among them, one of the first was proposed by Shapiro (1993) and used an embedded zerotree technique referred to as EZW. Another algorithm is the so-called set partitioning in hierarchical trees (SPIHT) developed by Said and Pearlman (1996). This algorithm also produces an embedded bitstream. The advantage of the embedded coding schemes allows an encoding process to terminate at any point so that a target bit rate or distortion metric can be met exactly. Intuitively, for a given bit rate or distortion requirement a nonembedded code should be more efficient than an embedded code since it has no constraints imposed by embedding requirements. However, embedded wavelet transform coding algorithms are currently the best. The additional constraints do not seem to have deleterious effect. In the following, we introduce the two embedded coding algorithms: the zerotree coding and the set partitioning in hierarchical tree coding.

As with DCT-based coding, an important aspect of wavelet-based coding is to code the positions of those coefficients that will be transmitted as nonzero values. After quantization the probability of the zero symbol must be extremely high for the very low bit rate case. A large portion of the bit budget will then be spent on encoding the significance map, or the binary decision map that indicates whether a transformed coefficient has a zero or nonzero quantized value. Therefore, the ability to efficiently encode the significance map becomes a key issue for coding images at very low bit rates. A new data structure, the zerotree, has been proposed for this purpose (Shapiro, 1993). To describe zerotree, we first must define insignificance. A wavelet coefficient is insignificant with respect to a given threshold value if the absolute value of this coefficient is smaller than this threshold. From the nature of the wavelet transform we can assume that every wavelet transformed at a given scale can be strongly related to a set of coefficients at the next finer scale of similar orientation. More specially, we can further assume that if a wavelet coefficient at a coarse scale is insignificant with respect to the preset threshold, then all wavelet coefficients at finer scales are likely to be insignificant with respect to this threshold. Therefore, we can build a tree with these

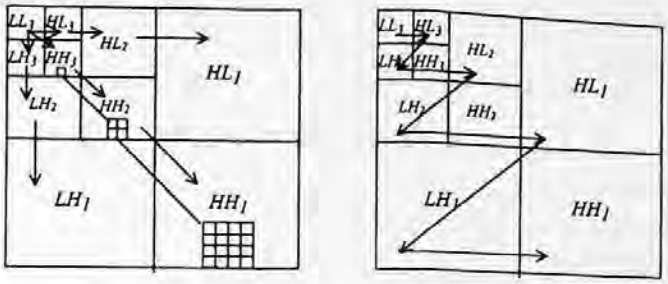


FIGURE 8.7 (Left) Parent-children dependencies of subbands; the arrow points from the subband of the parents to the subband of the children. At top left is the lowest-frequency band. (Right) The scanning order of the subbands for encoding a significance map.

parent-child relationships, such that coefficients at a coarse scale are called parents, and all coefficients corresponding to the same spatial location at the next finer scale of similar orientation are called children. Furthermore, for a parent, the set of all coefficients at all finer scales of similar orientation corresponding to the same spatial location are called descendants. For a QMF-pyramid decomposition the parent-children dependencies are shown in Figure 8.7(a). For a multiscale wavelet transform, the scan of the coefficients begins at the lowest frequency subband and then takes the order of *LL*, *HL*, *LH*, and *HH* from the lower scale to the next higher scale, as shown in Figure 8.7(b).

The zerotree is defined such that if a coefficient itself and all of its descendants are insignificant with respect to a threshold, then this coefficient is considered an element of a zerotree. An element of a zerotree is considered as a zerotree root if this element is not the descendant of a previous zerotree root with respect to the same threshold value. The significance map can then be efficiently represented by a string with three symbols: zerotree root, isolated zero, and significant. The isolated zero means that the coefficient is insignificant, but it has some significant descendant. At the finest scale, only two symbols are needed since all coefficients have no children, thus the symbol for zerotree root is not used. The symbol string is then entropy encoded. Zerotree coding efficiently reduces the cost for encoding the significance map by using self-similarity of the coefficients at different scales. Additionally, it is different from the traditional run-length coding that is used in DCT-based coding schemes. Each symbol in a zerotree is a single terminating symbol, which can be applied to all depths of the zerotree, similar to the end-of-block (EOB) symbol in the JPEG and MPEG video coding standards. The difference between the zerotree and EOB is that the zerotree represents the insignificance information at a given orientation across different scale layers. Therefore, the zerotree can efficiently exploit the self-similarity of the coefficients at the different scales corresponding to the same spatial location. The EOB only represents the insignificance information over the spatial area at the same scale.

In summary, the zerotree-coding scheme tries to reduce the number of bits to encode the significance map, which is used to encode the insignificant coefficients. Therefore, more bits can be allocated to encode the important significant coefficients. It should be emphasized that this zerotree coding scheme of wavelet coefficients is an embedded coder, which means that an encoder can terminate the encoding at any point according to a given target bit rate or target distortion metric. Similarly, a decoder which receives this embedded stream can terminate at any point to reconstruct an image that has been scaled in quality.

Another embedded wavelet coding method is the SPIHT-based algorithm (Said and Pearlman, 1996). This algorithm includes two major core techniques: the set partitioning sorting algorithm and the spatial orientation tree. The set partitioning sorting algorithm is the algorithm that hierarchically divides coefficients into significant and insignificant, from the most significant bit to the least significant bit, by decreasing the threshold value at each hierarchical step for constructing a significance map. At each threshold value, the coding process consists of two passes: the sorting

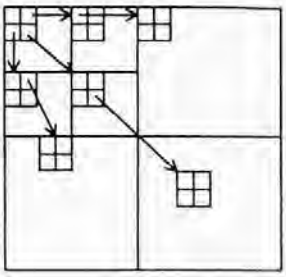


FIGURE 8.8 Relationship between pixels in the spatial orientation tree.

pass and the refinement pass — except for the first threshold that has only the sorting pass. Let $c(i,j)$ represent the wavelet-transformed coefficients and m is an integer. The sorting pass involves selecting the coefficients such that $2^m \le |c(i,j)| \le 2^{m+1}$, with m being decreased at each pass. This process divides the coefficients into subsets and then tests each of these subsets for significant coefficients. The significance map constructed in the procedure is tree-encoded. The significant information is store in three ordered lists: list of insignificant pixels (LIP), list of significant pixels (LSP), and list of insignificant sets (LIS). At the end of each sorting pass, the LSP contains the coordinates of all significant coefficients with respect to the threshold at that step. The entries in the LIS can be one of two types: type A represents all its descendants, type B represents all its descendants from its grandchildren onward. The refinement pass involves transmitting the mth-most significant bit of all the coefficients with respect to the threshold, 2^{m+1}.

The idea of a spatial orientation tree is based on the following observation. Normally, among the transformed coefficients most of the energy is concentrated in the low frequencies. For the wavelet transform, when we move from the highest to the lowest levels of the subband pyramid the energy usually decreases. It is also observed that there exists strong spatial self-similarity between subbands in the same spatial location such as in the zerotree case. Therefore, a spatial orientation tree structure has been proposed for the SPIHT algorithm. The spatial orientation tree naturally defines the spatial relationship on the hierarchical pyramid as shown in Figure 8.8.

During the coding, the wavelet-transformed coefficients are first organized into spatial orientation trees as in Figure 8.8. In the spatial orientation tree, each pixel (i,j) from the former set of subbands is seen as a root for the pixels $(2i, 2j)$, $(2i+1, 2j)$, $(2i,2j+1)$, and $(2i+1, 2j+1)$ in the subbands of the current level. For a given n-level decomposition, this structure is used to link pixels of the adjacent subbands from level n until to level 1. In the highest-level n, the pixels in the low-pass subband are linked to the pixels in the three high-pass subbands at the same level. In the subsequent levels, all the pixels of a subband are involved in the tree-forming process. Each pixel is linked to the pixels of the adjacent subband at the next lower level. The tree stops at the lowest level.

The implementation of the SPIHT algorithm consists of four steps: initialization, sorting pass, refinement pass, and quantization scale update. In the initialization step, we find an integer $m = \lfloor \log_2(\max_{(i,j)}\{|c(i,j)|\}) \rfloor$. Here $\lfloor\ \rfloor$ represent an operation of obtaining the largest integer less than $|c(i,j)|$. The value of m is used for testing the significance of coefficients and constructing the significance map. The LIP is set as an empty list. The LIS is initialized to contain all the coefficients in the low-pass subbands that have descendants. These coefficients can be used as roots of spatial trees. All these coefficients are assigned to be of type A. The LIP is initialized to contain all the coefficients in the low-pass subbands.

In the sorting pass, each entry of the LIP is tested for significance with respect to the threshold value 2^m. The significance map is transmitted in the following way. If it is significant, a "1" is transmitted, a sign bit of the coefficient is transmitted, and the coefficient coordinates are moved to the LSP. Otherwise, a "0" is transmitted. Then, each entry of the LIS is tested for finding the significant descendants. If there are none, a "0" is transmitted. If the entry has at least one significant descendant, then a "1" is transmitted and each of the immediate descendants are tested for significance. The significance map for the immediate descendants is transmitted in such a way that if it

is significant, a "1" plus a sign bit are transmitted and the coefficient coordinates are appended to the LSP. If it is not significant, a "0" is transmitted and the coefficient coordinates are appended to the LIP. If the coefficient has more descendants, then it is moved to the end of the LIS as an entry of type B. If an entry in the LIS is of type B, then its descendants are tested for significance. If at least one of them is significant, then this entry is removed from the list, and its immediate descendants are appended to the end of the list of type A. For the refinement pass, the mth-most significant bit of the magnitude of each entry of the LSP is transmitted except those in the current sorting pass. For the quantization scale update step, m is decreased by 1 and the procedure is repeated from the sorting pass.

8.3 WAVELET TRANSFORM FOR JPEG-2000

8.3.1 Introduction to JPEG-2000

Most image coding standards so far have exploited the DCT as their core technique for image decomposition. However, recently there has been a noticeable change. The wavelet transform has been adopted by MPEG-4 for still image coding (mpeg4). Also, JPEG-2000 is considering using the wavelet transform as its core technique for the next generation of the still image coding standard (jpeg2000 vm). This is because the wavelet transform can provide not only excellent coding efficiency, but also good spatial and quality scalable functionality. JPEG-2000 is a new type of image compression system under development by Joint Photographic Experts Group for still image coding. This standard is intended to meet a need for image compression with great flexibility and efficient interchangeability. JPEG-2000 is also intended to offer unprecedented access into the image while still in compressed domain. Thus, images can be accessed, manipulated, edited, transmitted, and stored in a compressed form. As a new coding standard, the detailed requirements of JPEG-2000 include:

Low bit-rate compression performance: JPEG-2000 is required to offer excellent coding performance at bit rates lower than 0.25 bits per pixel for highly detailed gray-bits per level images since the current JPEG (10918-1) cannot provide satisfactory results at this range of bit rates. This is the primary feature of JPEG-2000.

Lossless and lossy compression: it is desired to provide lossless compression naturally in the course of progressive decoding. This feature is especially important for medical image coding where the loss is not always allowed. Also, other applications such as high-quality image archival systems and network applications desire to have the functionality of lossless reconstruction.

Large images: currently, the JPEG image compression algorithm does not allow for images greater than 64K by 64K without tiling.

Single decomposition architecture: the current JPEG standard has 44 modes; many of these modes are for specific applications and not used by the majority of JPEG decoders. It is desired to have a single decomposition architecture that can encompass the interchange between applications.

Transmission in noisy environments: it is desirable to consider error robustness while designing the coding algorithm. This is important for the application of wireless communication. The current JPEG has provision for restart intervals, but image quality suffers dramatically when bit errors are encountered.

Computer-generated imagery: the current JPEG is optimized for natural imagery and does not perform well on computer-generated imagery or computer graphics.

Compound documents: the new coding standard is desired to be capable of compressing both continuous-tone and bilevel images. The coding scheme can compress and decompress images from 1 bit to 16 bits for each color component. The current JPEG standard does not work well for bilevel images.

Progressive transmission by pixel accuracy and resolution: progressive transmission that allows images to be transmitted with increasing pixel accuracy or spatial resolution is important for many applications. The image can be reconstructed with different resolutions and pixel accuracy as needed for different target devices such as in World Wide Web applications and image archiving.

Real-time encoding and decoding: for real-time applications, the coding scheme should be capable of compressing and decompressing with a single sequential pass. Of course, optimal performance cannot be guaranteed in this case.

Fixed rate, fixed size, and limited workspace memory: the requirement of fixed bit rate allows the decoder to run in real time through channels with limited bandwidth. The limited memory space is required by the hardware implementation of decoding.

There are also some other requirements such as backwards compatibility with JPEG, open architecture for optimizing the system for different image types and applications, interface with MPEG-4, and so on. All these requirements are seriously being considered during the development of JPEG-2000. However, it is still too early to comment whether all targets can be reached at this moment. There is no doubt, though, that the basic requirement on the coding performance at very low bit rate for still image coding will be achieved by using wavelet-based coding as the core technique instead of DCT-based coding.

8.3.2 Verification Model of JPEG-2000

Since JPEG-2000 is still awaiting finalization, we introduce the techniques that are very likely to be adopted by the new standard. As in other standards such as MPEG-2 and MPEG-4, the verification model (VM) plays an important role during the development of standards. This is because the VM or TM (test model for MPEG-2) is a platform for verifying and testing the new techniques before they are adopted as standards. The VM is updated by completing a set of core experiments from one meeting to another. Experience has shown that the decoding part of the final version of VM is very close to the final standard. Therefore, in order to give an overview of the related wavelet transform parts of the JPEG-2000, we start to introduce the newest version of JPEG-2000 VM (jpeg2000 vm). The VM of JPEG-2000 describes the encoding process, decoding process, and the bitstream syntax, which eventually completely defines the functionality of the existing JPEG-2000 compression system.

The newest version of the JPEG-2000 verification model, currently VM 4.0, was revised on April 22, 1999. In this VM, the final convergence has not been reached, but several candidates have been introduced. These techniques include a DCT-based coding mode, which is currently the baseline JPEG, and a wavelet-based coding mode. In the wavelet-based coding mode, several algorithms have been proposed: overlapped spatial segmented wavelet transform (SSWT), non-overlapped SSWT, and the embedded block-based coding with optimized truncation (EBCOT). Among these techniques, and according to current consensus, EBCOT is a very likely candidate for adoption into the final JPEG-2000 standard.

The basic idea of EBCOT is the combination of block coding with wavelet transform. First, the image is decomposed into subbands using the wavelet transform. The wavelet transform is not restricted to any particular decomposition. However, the Mallat wavelet provides the best compression performance, on average, for natural images; therefore, the current bitstream syntax is restricted to the standard Mallat wavelet transform in VM 4.0. After decomposition, each subband is divided into 64×64 blocks, except at image boundaries where some blocks may have smaller sizes. Every block is then coded independently. For each block, a separate bitstream is generated without utilizing any information from other blocks. The key techniques used for coding include an embedded quad-tree algorithm and fractional bit-plane coding.

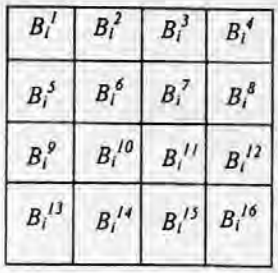


FIGURE 8.9 Example of sub-block partitioning for a block of 64 × 64.

The idea of an embedded quad-tree algorithm is that it uses a single bit to represent whether or not each leading bit-plane contains any significant samples. The quad-tree is formed in the following way. The subband is partitioned into a basic block. The basic block size is 64 × 64. Each basic block is further partitioned into 16 × 16 sub-blocks, as shown in Figure 8.9. Let $\sigma^j(B_i^k)$ denote the significance of sub-block, B_i^k (k is the kth sub-block as shown in Figure 8.9), in jth bit plane of ith block. If one or more samples in the sub-block have the magnitude greater than 2^j, then $\sigma^j(B_i^k) = 1$; otherwise, $\sigma^j(B_i^k) = 0$. For each bit-plane, the information concerning the significant sub-blocks is first encoded. All other sub-blocks can then be bypassed in the remaining coding procedure for that bit-plane. To specify the exact coding sequence, we define a two-level quad-tree for the block size of 64 × 64 and sub-block size of 16 × 16. The level-1 quads, $Q_i^1[k]$, consist of four sub-blocks, B_i^1, B_i^2, B_i^3, B_i^4 from Figure 8.9. In the same way, we define level-2 quads, $Q_i^2[k]$, to be 2 × 2 groupings of level-1 quads. Let $\sigma^j(Q_i^1[k])$ denote the significance of the level-1 quad, $Q_i^1[k]$, in jth bit-plane. If at least one member sub-block is significant in the jth bit-plane, then $\sigma^j(Q_i^1[k]) = 1$; otherwise, $\sigma^j(Q_i^1[k]) = 0$. At each bit-plane, the quad-tree coder visits the level-2 quad first, followed by level-1 quads. When visiting a particular quad, $Q_i^L[k]$(L = 1 or 2, it is the number of the level), the coder sends the significance of each of the four child quads, $\sigma^j(Q_i^L[k])$, or sub-blocks, $\sigma^j(B_i^k)$, as appropriate, except if the significance value can be deduced from the decoder. Under the following three cases, the significance may be deduced by the decoder: (1) the relevant quad or sub-block was significant in the previous bit-plane; (2) the entire sub-block is insignificant; or (3) this is the last child or sub-block visited in $Q_i^L[k]$ and all previous quads or sub-blocks are insignificant.

The idea of bit-plane coding is to entropy code the most significant bit first for all samples in the sub-blocks and to send the resulting bits. Then, the next most significant bit will be coded and sent, this process will be continued until all bit-planes have been coded and sent. This kind of bitstream structure can be used for robust transmission. If the bitstream is truncated due to a transmission error or some other reason, then some or all the samples in the block may lose one or more least significant bits. This will be equivalent to having used a coarser quantizer for the relevant samples and we can still obtain a reduced-quality reconstructed image. The idea of fractional bit-plane coding is to code each bit-plane with four passes: a forward significance propagation pass, a backward significance propagation pass, a magnitude refinement pass, and a normalization pass. For the technical details of fractional bit-plane coding, the interested readers can refer to the VM of JPEG-2000 (jpeg2000 vm).

Finally, we briefly describe the optimization issue of EBCOT. The encoding optimization algorithm is not a part of the standard, since the decoder does not need to know how the encoder generates the bitstream. From the viewpoint of the standard, the only requirement from the decoder to the encoder is that the bitstream must be compliant with the syntax of the standard. However, from the other side, the bitstream syntax could always be defined to favor certain coding algorithms for generating optimized bitstreams. The optimization algorithm described here is justified only if the distortion measure adopted for the code blocks is additive. That is, the final distortion, D, of the whole reconstructed image should satisfy

$$D = \sum D_i^{T_i} \tag{8.24}$$

where D_i is the distortion for block B_i and T_i is the truncation point for B_i. Let R be the total number of bits for coding all blocks of the image for a set of truncation point T_i, then

$$R = \sum R_i^{T_i} \tag{8.25}$$

where $R_i^{T_i}$ are the bits for coding block B_i. The optimization process wishes to find the suitable set of T_i values, which minimizes D subject to the constraint $R \le R_{max}$. R_{max} is the maximum number of bits assigned for coding the image. The solution is obtained by the method of Lagrange multipliers:

$$L = \sum \left(R_i^{T_i} - \lambda D_i^{T_i} \right) \tag{8.26}$$

where the value λ must be adjusted until the rate obtained by the truncation points, which minimize the value of L, satisfies $R = R_{max}$. From Equation 8.26, we have a separate trivial optimization problem for each individual block. Specially, for each block, B_i, we find the truncation point, T_i, which minimizes the value $(R_i^{Ri} - \lambda D_i^{Ti})$. This can be achieved by finding the slope turning points of rate distortion curves. In the VM, the set of truncation points and the slopes of rate distortion curves are computed immediately after each block is coded, and we only store enough information to later determine the truncation points which correspond to the slope turning points of rate distortion curves. This information is generally much smaller than the bitstream which is stored for the block itself. Also, the search for the optimal λ is extremely fast and occupies a negligible portion of the overall computation time.

8.4 SUMMARY

In this chapter, image coding using the wavelet transform has been introduced. First, an overview of wavelet theory was given, and second, the principles of image coding using wavelet transform have been presented. Additionally, two particular embedded image coding algorithms have been explained, namely, the embedded zerotree and set partitioning in hierarchical trees. Finally, the new standard for still image coding, JPEG-2000, which may adopt the wavelet as its core technique, has been described.

8.5 EXERCISES

8-1. For a given function, the Mexican hat wavelet,

$$f(t) = \begin{cases} 1, & for\ |t| \le 1, \\ 0, & otherwise \end{cases}$$

Use Equations 8.3 and 8.4 to derive a closed-form expression for the continuous wavelet transform, $\psi_{ab}(t)$.

8-2. Consider the dilation equation

$$\varphi(t) = \sqrt{2} \sum_k h(k) \varphi(2t - k)$$

How does $\varphi(t)$ change if $h(k)$ is shifted? Specifically, let $g(k) = h(n-l)$

$$u(t) = \sqrt{2}\sum_k g(k)u(2t-k)$$

How does $u(t)$ relate to $\varphi(t)$?

8-3. Let $\varphi_a(t)$ and $\varphi_b(t)$ be two scaling functions generated by the two scaling filters $h_a(k)$ and $h_b(k)$. Show that the convolution $j_a(t)^* j_b(t)$ satisfies a dilation equation with $h_a(k)^* h_b(k)/\sqrt{2}$.

8-4. In the applications of denoising and image enhancement, how can the wavelet transform improve the results?

8-5. For a given function

$$f(t) = \begin{cases} 0 & t < 0 \\ t & 0 \le t < 1 \\ 1 & t \ge 1 \end{cases}$$

show that the wavelet transform of $f(t)$ will be

$$W(a,b) = \text{sgn}\frac{2f\left(b+\frac{a}{2}\right) - f(b) - f(b+a)}{\sqrt{|a|}}$$

where sgn(x) is the signum function defined as

$$\text{sgn}(x) = \begin{cases} -1 & t < 0 \\ 1 & t > 0 \\ 0 & t = 0 \end{cases}$$

REFERENCES

Cohen, L. Time-Frequency Distributions — A Review, *Proc. IEEE,* Vol. 77, No. 7, July 1989, pp. 941-981.

Daubechies, I. *Ten Lectures on Wavelets,* CBMS Series, Philadelphia, SIAM, 1992.

Grossman, A. and J. Morlet, Decompositions of hard functions into square integrable wavelets of constant shape, *SIAM J. Math. Anal.,* 15(4), 723-736, 1984.

Jayant, N. S. and P. Noll, *Digital Coding of Waveforms,* Englewood Cliffs, NJ: Prentice-Hall, 1984.

jpeg2000 vm, JPEG-2000 Verification Model 4.0 (Tech. description), sc29wg01 N1282, April 22, 1999.

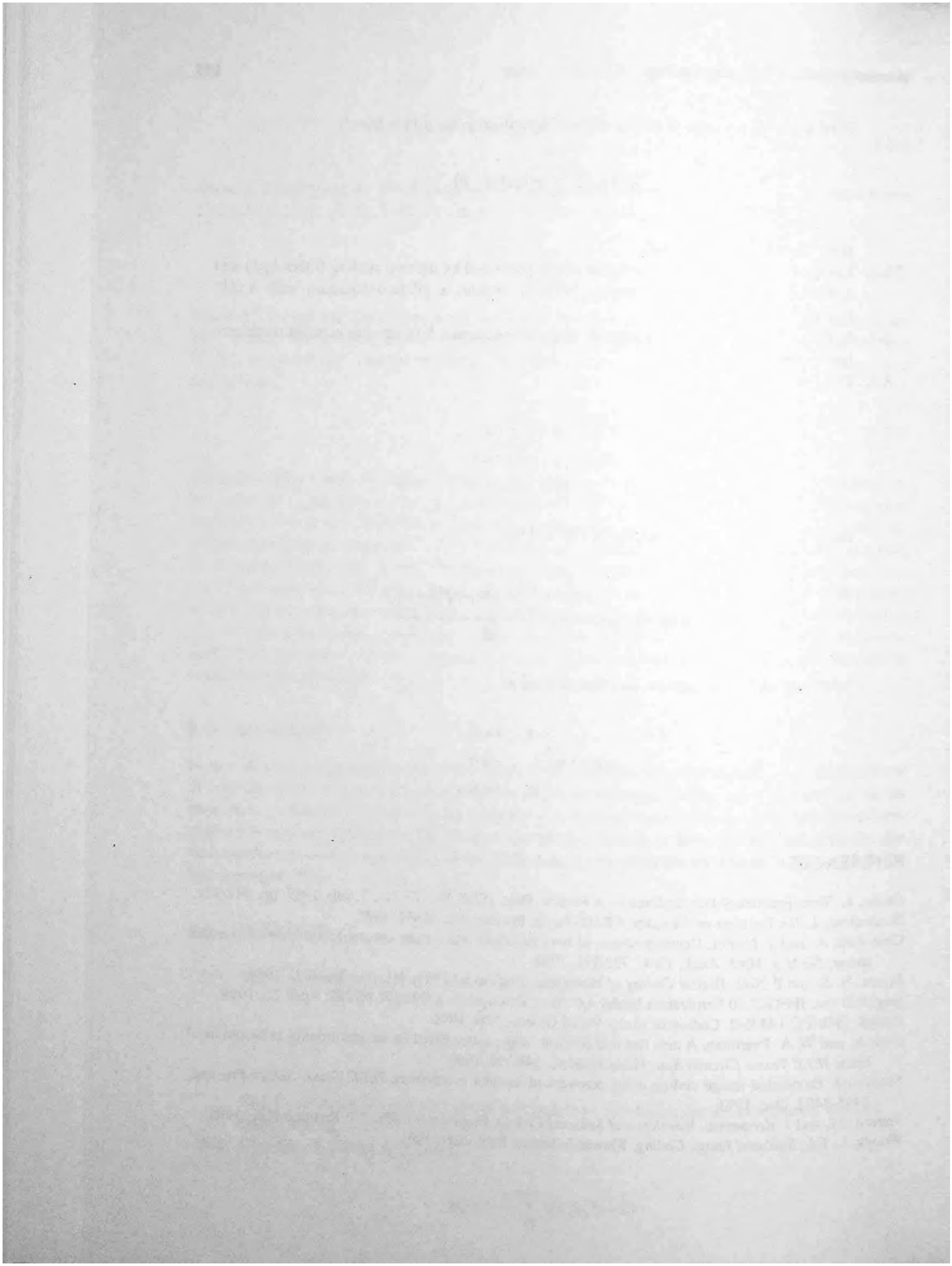
mpeg4, ISO/IEC 14496-2, Coding of Audio-Visual Objects, Nov. 1998.

Said, A. and W. A. Pearlman, A new fast and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. Circuits Syst. Video Technol.,* 243-250, 1996.

Shapiro, J. Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. Signal Process.,* 3445-3462, Dec. 1993.

Vetterli, M. and J. Kovacevic, *Wavelets and Subband Coding,* Englewood Cliffs, NJ: Prentice-Hall, 1995.

Woods, J., Ed., *Subband Image Coding,* Kluwer Academic Publishers, 1991.



9 Nonstandard Image Coding

In this chapter, we introduce three nonstandard image coding techniques: vector quantization (VQ) (Nasrabadi and King, 1988), fractal coding (Barnsley and Hurd, 1993; Fisher, 1994; Jacquin, 1993), and model-based coding (Li et al., 1994).

9.1 INTRODUCTION

The VQ, fractal coding, and model-based coding techniques have not yet been adopted as an image coding standard. However, due to their unique features these techniques may find some special applications. Vector quantization is an effective technique for performing data compression. Theoretically, vector quantization is always better than scalar quantization because it fully exploits the correlation between components within the vector. The optimal coding performance will be obtained when the dimension of the vector approaches infinity, and then the correlation between all components is exploited for compression. Another very attractive feature of image vector quantization is that its decoding procedure is very simple since it only consists of table look-ups. However, there are two major problems with image VQ techniques. The first is that the complexity of vector quantization exponentially increases with the increasing dimensionality of vectors. Therefore, for vector quantization it is important to solve the problem of how to design a practical coding system which can provide a reasonable performance under a given complexity constraint. The second major problem of image VQ is the need for a codebook, which causes several problems in practical application such as generating a universal codebook for a large number of images, scaling the codebook to fit the bit rate requirement, and so on. Recently, the lattice VQ schemes have been proposed to address these problems (Li, 1997).

Fractal theory has a long history. Fractal-based techniques have been used in several areas of digital image processing such as image segmentation, image synthesis, and computer graphics, but only in recent years have they been extended to the applications of image compression (Jacquin, 1993). A fractal is a geometric form which has the unique feature of having extremely high visual self-similar irregular details while containing very low information content. Several methods for image compression have been developed based on different characteristics of fractals. One method is based on Iterated Function Systems (*IFS*) proposed by Barnsley (1988). This method uses the self-similar and self-affine property of fractals. Such a system consists of sets of transformations including translation, rotation, and scaling. On the encoder side of a fractal image coding system, a set of fractals is generated from the input image. These fractals can be used to reconstruct the image at the decoder side. Since these fractals are represented by very compact fractal transformations, they require very small amounts of data to be expressed and stored as formulas. Therefore, the information needed to be transmitted is very small. The second fractal image coding method is based on the fractal dimension (Lu, 1993; Jang and Rajala, 1990). Fractal dimension is a good representation of the roughness of image surfaces. In this method, the image is first segmented using the fractal dimension and then the resultant uniform segments can be efficiently coded using the properties of the human visual system. Another fractal image coding scheme is based on fractal geometry, which is used to measure the length of a curve with a yardstick (Walach, 1989). The details of these coding methods will be discussed in Section 9.3.

The basic idea of model-based coding is to reconstruct an image with a set of model parameters. The model parameters are then encoded and transmitted to the decoder. At the decoder the decoded

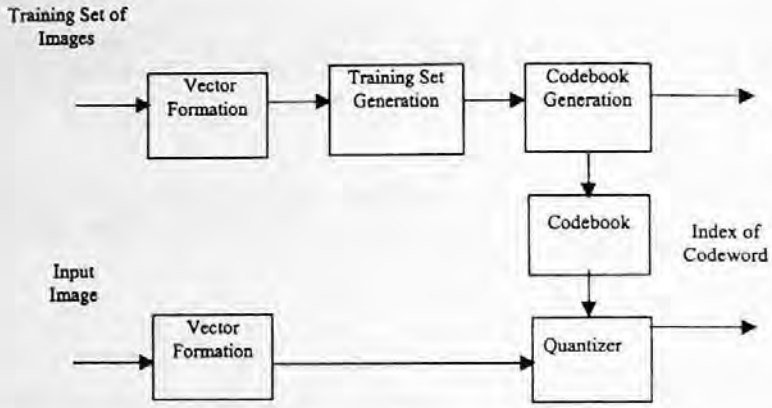


FIGURE 9.1 Principle of image vector quantization. The dashed lines correspond to training set generation, codebook generation, and transmission (if it is necessary).

model parameters are used to reconstruct the image with the same model used at the encoder. Therefore, the key techniques in the model-based coding are image modeling, image analysis, and image synthesis.

9.2 VECTOR QUANTIZATION

9.2.1 Basic Principle of Vector Quantization

An N-level vector quantizer, Q, is mapping from a K-dimensional vector set $\{V\}$, into a finite codebook, $W = \{w_1, w_2, \ldots, w_N\}$:

$$Q: V \rightarrow W \tag{9.1}$$

In other words, it assigns an input vector, v, to a representative vector (codeword), w from a codebook, W. The vector quantizer, Q, is completely described by the codebook, $W = \{w_1, w_2, \ldots, w_N\}$, together with the disjoint partition, $R = \{r_1, r_2, \ldots, r_N\}$, where

$$r_i = \{v: Q(v) = w_i\} \tag{9.2}$$

and w and v are K-dimensional vectors. The partition should identically minimize the quantization error (Gersho, 1982). A block diagram of the various steps involved in image vector quantization is depicted in Figure 9.1.

The first step in image vector quantization is the image formation. The image data are first partitioned into a set of vectors. A large number of vectors from various images are then used to form a training set. The training set is used to generate a codebook, normally using an iterative clustering algorithm. The quantization or coding step involves searching each input vector for the closest codeword in the codebook. Then the corresponding index of the selected codeword is coded and transmitted to the decoder. At the decoder, the index is decoded and converted to the corresponding vector with the same codebook as at the encoder by look-up table. Thus, the design decisions in implementing image vector quantization include (1) vector formation; (2) training set generation; (3) codebook generation; and (4) quantization.

9.2.1.1 Vector Formation

The first step of vector quantization is vector formation; that is, the decomposition of the images into a set of vectors. Many different decompositions have been proposed; examples include the

intensity values of a spatially contiguous block of pixels (Gersho and Ramamuthi, 1982; Baker and Gray, 1983); these same intensity values, but now normalized by the mean and variance of the block (Murakami et al., 1982); the transformed coefficients of the block pixels (Li and Zhang, 1995); and the adaptive linear predictive coding coefficients for a block of pixels (Sun, 1984). Basically, the approaches of vector formation can be classified into two categories: direct spatial or temporal, and feature extraction. Direct spatial or temporal is a simple approach to forming vectors from the intensity values of a spatial or temporal contiguous block of pixels in an image or an image sequence. A number of image vector quantizaton schemes have been investigated with this method. The other method is feature extraction. An image feature is a distinguishing primitive characteristic. Some features are natural in the sense that they are defined by the visual appearance of an image, while the other so-called artificial features result from specific manipulations or measurements of images or image sequences. In vector formation, it is well known that the image data in a spatial domain can be converted to a different domain so that subsequent quantization and joint entropy encoding can be more efficient. For this purpose, some features of image data, such as transformed coefficients and block means can be extracted and vector quantized. The practical significance of feature extraction is that it can result in the reduction of vector size, consequently reducing the complexity of coding procedure.

9.2.1.2 Training Set Generation

An optimal vector quantizer should ideally match the statistics of the input vector source. However, if the statistics of an input vector source are unknown, a training set representative of the expected input vector source can be used to design the vector quantizer. If the expected vector source has a large variance, then a large training set is needed. To alleviate the implementation complexity caused by a large training set, the input vector source can be divided into subsets. For example, in (Gersho, 1982) the single input source is divided into "edge" and "shade" vectors, and then the separate training sets are used to generate the separate codebooks. Those separate codebooks are then concatenated into a final codebook. In other methods, small local input sources corresponding to portions of the image are used as the training sets, thus the codebook can better match the local statistics. However, the codebook needs to be updated to track the changes in local statistics of the input sources. This may increase the complexity and reduce the coding efficiency. Practically, in most coding systems a set of typical images is selected as the training set and used to generate the codebook. The coding performance can then be insured for the images with the training set, or for those not in the training set but with statistics similar to those in the training set.

9.2.1.3 Codebook Generation

The key step in conventional image vector quantization is the development of a good codebook. The optimal codebook, using the mean squared error (MSE) criterion, must satisfy two necessary conditions (Gersho, 1982). First, the input vector source is partitioned into a predecided number of regions with the minimum distance rule. The number of regions is decided by the requirement of the bit rate, or compression ratio and coding performance. Second, the codeword or the representative vector of this region is the mean value, or the statistical center, of the vectors within the region. Under these two conditions, a generalized Lloyd clustering algorithm proposed by Linde, Buzo, and Gray (1980) — the so-called LBG algorithm — has been extensively used to generate the codebook. The clustering algorithm is an iterative process, minimizing a performance index calculated from the distances between the sample vectors and their cluster centers. The LBG clustering algorithm can only generate a codebook with a local optimum, which depends on the initial cluster seeds. Two basic procedures have been used to obtain the initial codebook or cluster seeds. In the first approach, the starting point involves finding a small codebook with only two codewords, and then recursively splitting the codebook until the required number of codewords is

obtained. This approach is referred to as binary splitting. The second procedure starts with initial seeds for the required number of codewords, these seeds being generated by preprocessing the training sets. To address the problem of a local optimum, Equitz (1989) proposed a new clustering algorithm, the pairwise nearest neighbor (PNN) algorithm. The PNN algorithm begins with a separate cluster for each vector in the training set and merges together two clusters at a time until the desired codebook size is obtained. At the beginning of the clustering process, each cluster contains only one vector. In the following process the two closest vectors in the training set are merged to their statistical mean value, in such a way the error incurred by replacing these two vectors with a single codeword is minimized. The PNN algorithm significantly reduces computational complexity without sacrificing performance. This algorithm can also be used as an initial codebook generator for the LBG algorithm.

9.2.1.4 Quantization

Quantization in the context of a vector quantization involves selecting a codeword in the codebook for each input vector. The optimal quantization, in turn, implies that for each input vector, v, the closest codeword, w_i, is found as shown in Figure 9.2. The measurement criterion could be mean squared error, absolute error, or other distortion measures.

A full-search quantization is an exhaustive search process over the entire codebook for finding the closest codeword, as shown in Figure 9.3(a). It is optimal for the given codebook, but the computation is more expensive. An alternative approach is a tree-search quantization, where the search is carried out based on a hierarchical partition. A binary tree search is shown in Figure 9.3(b). A tree search is much faster than a full search, but it is clear that the tree search is suboptimal for the given codebook and requires more memory for the codebook.

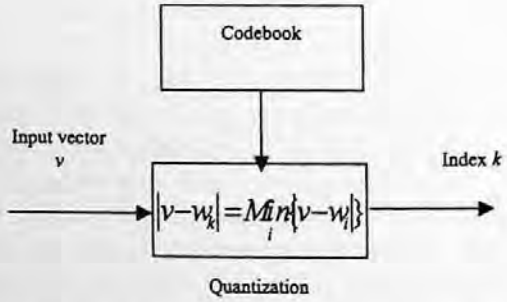


FIGURE 9.2 Principle of vector quantization.

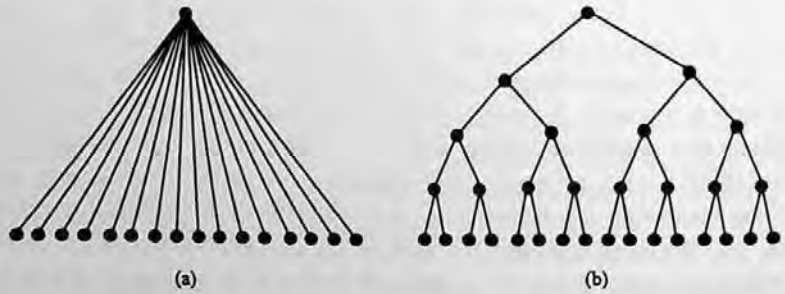


FIGURE 9.3 (a) Full search quantization; (b) binary tree search quantization.

9.2.2 Several Image Coding Schemes with Vector Quantization

In this section, we are going to present several image coding schemes using vector quantization which include residual vector quantization, classified vector quantization, transform domain vector quantization, predictive vector quantization, and block truncation coding (BTC) which can be seen as a binary vector quantization.

9.2.2.1 Residual VQ

In the conventional image vector quantization, the vectors are formed by spatially partitioning the image data into blocks of 8 × 8 or 4 × 4 pixels. In the original spatial domain the statistics of vectors may be widely spread in the multidimensional vector space. This causes difficulty in generating the codebook with a finite size and limits the coding performance. Residual VQ is proposed to alleviate this problem. In residual VQ, the mean of the block is extracted and coded separately. The vectors are formed by subtracting the block mean from the original pixel values. This scheme can be further modified by considering the variance of the blocks. The original blocks are converted to the vectors with zero mean and unit standard deviation with the following conversion formula (Murakami et al., 1982):

$$m_i = \frac{1}{K}\sum_{j=0}^{k-1} s_j \tag{9.3}$$

$$x_j = \frac{\left(s_j - m_i\right)}{\sigma_i} \tag{9.4}$$

$$\sigma_i = \left[\frac{1}{K}\sum_{j=0}^{K-1}\left(s_j - m_i\right)^2\right]^{\frac{1}{2}} \tag{9.5}$$

where m_i is the mean value of ith block, σ_i is the variance of ith block, s_j is the pixel value of pixel j ($j = 0, \ldots, K\text{-}1$) in the ith block, K is the total number of pixels in the block, and x_j is the normalized value of pixel j. The new vector X_i is now formed by x_j ($j = 0, 1, \ldots, k\text{-}1$):

$$X_i = [x_0, x_1, \ldots, x_K]_i \tag{9.6}$$

With the above normalization the probability function $P(X)$ of input vector X is approximately similar for image data from different scenes. Therefore, it is easy to generate a codebook for the new vector set. The problem with this method is that the mean and variance values of blocks have to be coded separately. This increases the overhead and limits the coding efficiency. Several methods have been proposed to improve the coding efficiency. One of these methods is to use predictive coding to code the block mean values. The mean value of the current block can be predicted by one of the previously coded neighbors. In such a way, the coding efficiency increases as the use of interblock correlation.

9.2.2.2 Classified VQ

In image vector quantization, the codebook is usually generated using training set under constraint of minimizing the mean squared error. This implies that the codeword is the statistical mean of the

region. During quantization, each input vector is replaced by its closest codeword. Therefore, the coded images usually suffer from edge distortion at very low bit rates, since edges are smoothed by the operation of averaging with the small-sized codebook. To overcome this problem, we can classify the training vector set into edge vectors and shade vectors (Gersho, 1982). Two separate codebooks can then be generated with the two types of training sets. Each input vector can be coded by the appropriate codeword in the codebook. However, the edge vectors can be further classified into many types according to their location and angular orientation. The classified VQ can be extended into a system which contains many sub-codebooks, each representing a type of edge. However, this would increase the complexity of the system and would be hard to implement in practical applications.

9.2.2.3 Transform Domain VQ

Vector quantization can be performed in the transform domain. A spatial block of 4×4 or 8×8 pixels is first transformed to the 4×4 or 8×8 transformed coefficients. There are several ways to form vectors with transformed coefficients. In the first method, a number of high-order coefficients can be discarded since most of the energy is usually contained in the low-order coefficients for most blocks. This reduces the VQ computational complexity at the expense of a small increase in distortion. However, for some active blocks, the edge information is contained in the high frequencies, or high-order coefficients. Serious subjective distortion will be caused by discarding high frequencies. In the second method, the transformed coefficients are divided into several bands and each band is used to form its corresponding vector set. This method is equivalent to the classified VQ in spatial domain. An adaptive scheme is then developed by using two kinds of vector formation methods. The first method is used for the blocks containing the moderate intensity variation and the second method is used for the blocks with high spatial activities. However, the complexity increases as more codebooks are needed in this kind of adaptive coding system.

9.2.2.4 Predictive VQ

The vectors are usually formed by the spatially consecutive blocks. The consecutive vectors are then highly statistically dependent. Therefore, better coding performance can be achieved if the correlation between vectors is exploited. Several predictive VQ schemes have been proposed to address this problem. One kind of predictive VQ is finite state VQ (Foster et al., 1985). The finite-state VQ is similar to a trellis coder. In the finite state VQ, the codebook consists of a set of sub-codebooks. A state variable is then used to specify which sub-codebook should be selected for coding the input vector. The information about the state variable must be inferred from the received sequence of state symbols and initial state such as in a trellis coder. Therefore, no side information or no overhead need be transmitted to the decoder. The new encoder state is a function of the previous encoder state and the selected sub-codebook. This permits the decoder to track the encoder state if the initial condition is known. The finite-state VQ needs additional memory to store the previous state, but it takes advantage of correlation between successive input vectors by choosing the appropriate codebook for the given past history. It should be noted that the minimum distortion selection rule of conventional VQ is not necessary optimum for finite-state VQ for a given decoder since a low-distortion codeword may lead to a bad state and hence to poor long-term behavior. Therefore, the key design issue of finite-state VQ is to find a good next-state function.

Another predictive VQ was proposed by Hang and Woods (1985). In this system, the input vector is formed in such a way that the current pixel is as the first element of the vector and the previous inputs as the remaining elements in the vector. The system is like a mapping or a recursive filter which is used to predict the next pixel. The mapping is implemented by a vector quantizer look-up table and provides the predictive errors.

9.2.2.5 Block Truncation Coding

In the block truncation code (BTC) (Delp and Mitchell, 1979), an image is first divided into 4×4 blocks. Each block is then coded individually. The pixels in each block are first converted into two-level signals by using the first two moments of the block:

$$a = m + \sigma\sqrt{\frac{q}{N-q}}$$
$$b = m - \sigma\sqrt{\frac{N-1}{q}} \tag{9.7}$$

where m is the mean value of the block, σ is the standard deviation of the block, N is the number of total pixels in the block, and q is the number of pixels which are greater in value than m. Therefore, each block can be described by the values of block mean, variance, and a binary-bit plane which indicates whether the pixels have values above or below the block mean. The binary-bit plane can be seen as a binary vector quantizer. If the mean and variance of the block are quantized to 8 bits, then 2 bits per pixel is achieved for blocks of 4×4 pixels. The conventional BTC scheme can be modified to increase the coding efficiency. For example, the block mean can be coded by a DPCM coder which exploits the interblock correlation. The bit plane can be coded with an entropy coder on the patterns (Udpikar and Raina, 1987).

9.2.3 Lattice VQ for Image Coding

In conventional image vector quantization schemes, there are several issues, which cause some difficulties for the practical application of image vector quantization. The first problem is the limitation of vector dimension. It has been indicated that the coding performance of vector quantization increases as the vector dimension while the coding complexity exponentially increases at the same time as the increasing vector dimension. Therefore, in practice only a small vector dimension is possible under the complexity constraint. Another important issue in VQ is the need for a codebook. Much research effort has gone into finding how to generate a codebook. However, in practical applications there is another problem of how to scale the codebook for various rate-distortion requirements. The codebook generated by LBG-like algorithms with a training set is usually only suitable for a specified bit rate and does not have the flexibility of codebook scalability. For example, a codebook generated for an image with small resolution may not be suitable for images with high resolution. Even for the same spatial resolution, different bit rates would require different codebooks. Additionally, the VQ needs a table to specify the codebook and, consequently, the complexity of storing and searching is too high to have a very large table. This further limits the coding performance of image VQ.

These problems become major obstacles for implementing image VQ. Recently, an algorithm of lattice VQ has been proposed to address these problems (Li et al., 1997). Lattice VQ does not have the above problems. The codebook for lattice VQ is simply a collection of lattice points uniformly distributed over the vector space. Scalability can be achieved by scaling the cell size associated with every lattice point just like in the scalar quantizer by scaling the quantization step. The basic concept of the lattice can be found in (Conway and Slone, 1991). A typical lattice VQ scheme is shown in Figure 9.4. There are two steps involved in the image lattice VQ. The first step is to find the closest lattice point for the input vector. The second step is to label the lattice point, i.e., mapping a lattice point to an index. Since lattice VQ does need a codebook, the index assignment is based on a lattice labeling algorithm instead of a look-up table such as in conventional VQ. Therefore, the key issue of lattice VQ is to develop an efficient lattice-labeling algorithm. With this

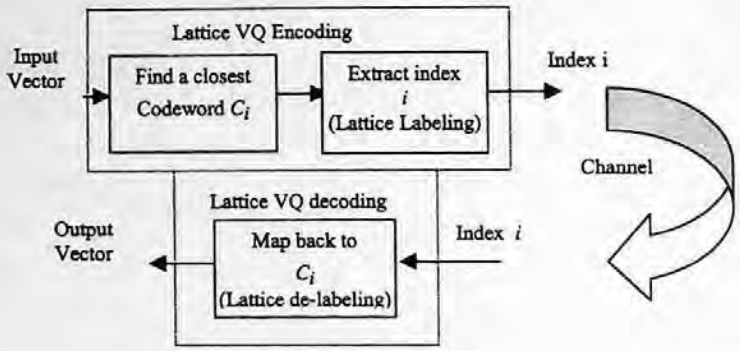


FIGURE 9.4 Block diagram of lattice VQ.

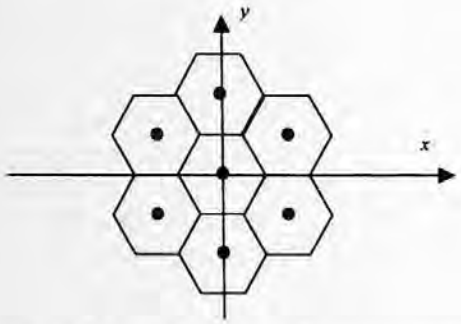


FIGURE 9.5 Labeling a two-dimensional lattice.

algorithm the closest lattice point and its corresponding index within a finite boundary can be obtained by a calculation at the encoder for each input vector.

At the decoder, the index is converted to the lattice point by the same labeling algorithm. The vector is then reconstructed with the lattice point. The efficiency of a labeling algorithm for lattice VQ is measured by how many bits are needed to represent the indices of the lattice points within a finite boundary. We use a two-dimensional lattice to explain the lattice labeling efficiency. A two-dimensional lattice is shown in Figure 9.5.

In Figure 9.5, there are seven lattice points. One method used to label these seven 2-D lattice points is to use their coordinates (x,y) to label each point. If we label x and y separately, we need two bits to label three values of x and three bits to label a possible five values of y, and need a total of five bits. It is clear that three bits are sufficient to label seven lattice points. Therefore, different labeling algorithms may have different labeling efficiency. Several algorithms have been developed for multidimensional lattice labeling. In (Conway, 1983), the labeling method assigns an index to every lattice point within a Voronoi boundary where the shape of the boundary is the same as the shape of Voronoi cells. Apparently, for different dimension, the boundaries have different shapes. In the algorithm proposed in (Laroia, 1993), the same method is used to assign an index to each lattice point. Since the boundaries are defined by the labeling algorithm, this algorithm might not achieve a 100% labeling efficiency for a prespecified boundary such as a pyramid boundary. The algorithm proposed by Fischer (1986) can assign an index to every lattice point within a prespecified pyramid boundary and achieves a 100% labeling efficiency, but this algorithm can only be used for the Z^n lattice. In a recently proposed algorithm (Wang et al., 1998), the technical breakthrough was obtained. In this algorithm a labeling method was developed for Construction-A and Construction-B lattices (Conway, 1983), which is very useful for VQ with proper vector dimensions, such as 16, and achieves 100% efficiency. Additionally, these algorithms are used for labeling lattice

points with 16 dimensions and provide minimum distortion. These algorithms were developed based on the relationship between lattices and linear block codes. Construction-A and Construction-B are the two simplest ways to construct a lattice from a binary linear block code C = (n, k, d), where n, k, and d are the length, the dimension, and the minimum distance of the code, respectively.

A Construction-A lattice is defined as:

$$\Lambda_n = C + 2Z^n \tag{9.8}$$

where Z^n is the n-dimensional cubic lattice and C is a binary linear block code. There are two steps involved for labeling a Construction-A lattice. The first is to order the lattice points according to the binary linear block code C, and then to order the lattice points associated with a particular nonzero binary codeword. For the lattice points associated with a nonzero binary codeword, two sub-lattices are considered separately. One sub-lattice consists of all the dimensions that have a "0" component in the binary codeword and the other consists of all the dimensions that have a "1" component in the binary codeword. The first sub-lattice is considered as a 2Z lattice while the second is considered as a translated 2Z lattice. Therefore, the labeling problem is reduced to labeling the Z lattice at the final stage.

A Construction-B lattice is defined as:

$$\Lambda_n = C + 2D_n \tag{9.9}$$

where D_n is an n-dimensional Construction-A lattice with the definition as:

$$D_n = (n, n-1, 2) + 2Z^n \tag{9.10}$$

and C is a binary doubly even linear block code. When n is equal to 16, the binary even linear block code associated with Λ_{16} is $C = (16, 5, 8)$. The method for labeling a Construction-B lattice is similar to the method for labeling a Construction-A lattice with two minor differences. The first difference is that for any vector $y = c + 2x$, $x \in Z^n$, if y is a Construction-A lattice point; and $x \in D_n$, if y is a Construction-B lattice point. The second difference is that C is a binary doubly even linear block code for Construction-B lattices while it is not necessarily doubly even for Construction-A lattices. In the implementation of these lattice point labeling algorithms, the encoding and decoding functions for lattice VQ have been developed in (Li et al., 1997). For a given input vector, an index representing the closest lattice point will be found by the encoding function, and for an input index the reconstructed vector will be generated by the decoding function. In summary, the idea of lattice VQ for image coding is an important achievement in eliminating the need for a codebook for image VQ. The development of efficient algorithms for lattice point labeling makes lattice VQ feasible for image coding.

9.3 FRACTAL IMAGE CODING

9.3.1 Mathematical Foundation

A fractal is a geometric form whose irregular details can be represented by some objects with different scale and angle, which can be described by a set of transformations such as affine transformations. Additionally, the objects used to represent the image's irregular details have some form of self-similarity and these objects can be used to represent an image in a simple recursive way. An example of fractals is the Von Koch curve as shown in Figure 9.6. The fractals can be used to generate an image. The fractal image coding that is based on iterated function systems

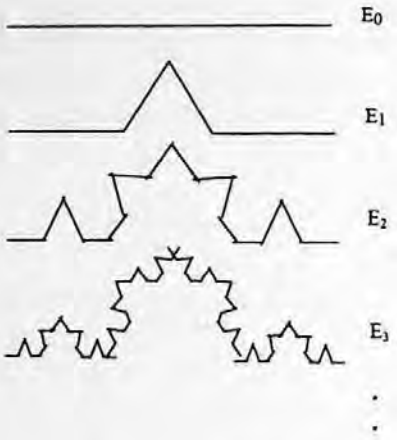


FIGURE 9.6 Construction of the Von Koch curve.

(*IFS*) is the inverse process of image generation with fractals. Therefore, the key technology of fractal image coding is the generation of fractals with an *IFS*.

To explain *IFS*, we start from the contractive affine transformation. A two-dimensional affine transformation A is defined as follows:

$$A\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{9.11}$$

This is a transformation which consists of a linear transformation followed by a shift or translation, and maps points in the Euclidean plane into new points in the another Euclidean plane. We define that a transformation is contractive if the distance between two points P_1 and P_2 in the new plane is smaller than their distance in the original plane, i.e.,

$$d\left(A(P_1), A(P_2)\right) < s\, d(P_1, P_2) \tag{9.12}$$

where s is a constant and $0 < s < 1$. The contractive transformations have the property that when the contractive transformations are repeatedly applied to the points in a plane, these points will converge to a fixed point. *An iterated function system (IFS) is defined as a collection of contractive affine transformations*. A well-known example of *IFS* contains four following transformations:

$$A_i\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad i = 1,2,3,4. \tag{9.13}$$

This is the *IFS* of a fern leaf, whose parameters are shown in Table 9.1.

The transformation A_1 is used to generate the stalk, the transformation A_2 is used to generate the right leaf, the transformation A_3 is used to generate the left leaf, and the transformation A_4 is used to generate main fern. A fundamental theorem of fractal geometry is that each *IFS* defines a unique fractal image. This image is referred to as the attractor of the *IFS*. In other words, an image corresponds to the attractor of an *IFS*. Now let us explain how to generate the image using the *IFS*. Let us suppose that an *IFS* contains N affine transformations, $A_1, A_2, \ldots A_N$, and each transformation has an associated probability, $p_1, p_2, \ldots, p_N$, respectively. Suppose that this is a complete set and the sum of the probability equals to 1, i.e.,

TABLE 9.1
The Parameters of the *IFS* of a Fern Leaf

	a	b	c	d	e	f
A_1	0	0	0	0.16	0	0.2
A_2	0.2	−0.26	0.23	0.22	0	0.2
A_3	−0.15	0.28	0.26	0.24	0	0.2
A_4	0.85	0.04	−0.04	0.85	0	0.2

$$p_1 + p_2 + \ldots + p_N = 1 \text{ and } p_i > 0 \text{ for } i = 0, 1, \ldots, N. \tag{9.14}$$

The procedure for generating an attractor is as follows. For any given point (x_0, y_0) in a Euclidean plane, one transformation in the *IFS* according to its probability is selected and applied to this point to generate a new point (x_1, y_1). Then another transformation is selected according to its probability and applied to the point (x_1,y_1) to obtain a new point (x_2,y_2). This process is repeated over and over again to obtain a long sequence of points: (x_0,y_0), (x_1,y_1), ..., (x_n,y_n), ... According to the theory of iterated function systems, these points will converge to an image that is the attractor of the given *IFS*. The above-described procedure is shown in the flowchart of Figure 9.7. With the above algorithm and the parameters in Table 9.1, initially the point can be anywhere within the large square, but after several iterations it will converge onto the fern. The 2-D affine transformations are extended to 3-D transformations, which can be used to create fractal surfaces with the iterated function systems. This fractal surface can be considered as the gray level or brightness of a 2-D image.

9.3.2 *IFS*-Based Fractal Image Coding

As described in the last section, an *IFS* can be used to generate a unique image, which is referred to as an attractor of the *IFS*. In other words, this image can be simply represented by the parameters of the *IFS*. Therefore, if we can use an inverse procedure to generate a set of transformations, i.e.,

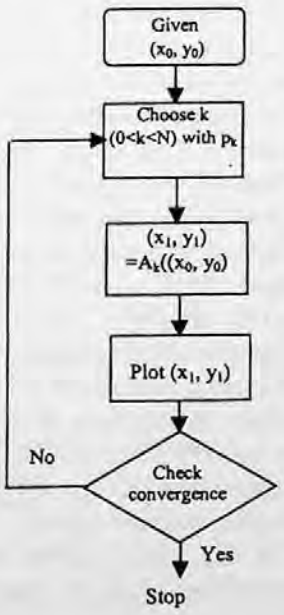


FIGURE 9.7 Flowchart of image generation with an *IFS*.

an *IFS* from an image, then these transformations or the *IFS* can be used to represent the approximation of the image. The image coding system can use the parameters of the transformations in the *IFS* instead of the original image data for storage or transmission. Since the *IFS* contains only very limited data such as transformation parameters, this image coding method may result in a very high compression ratio. For example, the fern image is represented by 24 integers or 192 bits (if each integer is represented by 8 bits). This number is much smaller than the number needed to represent the fern image pixel by pixel. Now the key issue of the *IFS*-based fractal image coding is to generate the *IFS* for the given input image. Three methods have been proposed to obtain the *IFS* (Lu, 1993). One is the direct method, that directly finds a set of contractive affine transformations from the image based on the self-similarity of the image. The second method is to partition an image into the smaller objects whose *IFS*s are known. These *IFS*s are used to form a library. The encoding procedure is to look for an *IFS* from the library for each small object. The third method is called partitioned *IFS* (*PIFS*). In this method, the image is first divided into smaller blocks and then the *IFS* for each block is found by mapping a larger block into a small block.

In the direct approach, the image is first partitioned into nonoverlapped blocks in such a way that each block is similar to the whole image and a transformation can map the whole image to the block. The transformation for each individual block may be different. The combination of these transformations can be taken as the *IFS* of the given image. Then much fewer data are required to represent the *IFS* or the transformations than to transmit or store the given image in the pixel by pixel way. For the second approach, the key issue is how to partition the given image into objects whose *IFS*s are known. The image processing techniques such as color separation, edge detection, spectrum analysis, and texture variation analysis can be used for image partitioning. However, for natural images or arbitrary images, it may be impossible or very difficult to find an *IFS* whose attractor perfectly covers the original image. Therefore, for most natural images the partitioned *IFS* method has been proposed (Lu, 1993). In this method, the transformations do not map the whole image into small block. For encoding an image, the whole image is first partitioned into a number of larger blocks that are referred to as domain blocks. The domain blocks can be overlapped. Then the image is partitioned into a number of smaller blocks that are called as range blocks. The range blocks do not overlap and the sum total of the range blocks covers the whole image. In the third step, a set of contractive transformations is chosen. Each range block is mapped into a domain block with a searching method and a matching criterion. The combination of the transformations is used to form a partitioned *IFS* (*PIFS*). The parameters of *PIFS* are transmitted to the decoder. It is noted that no domain blocks are transmitted. The decoding starts with a flat background. The iterated process is then applied with the set of transformations. The reconstructed image is then obtained after the process converges. From the above discussion, it is found that there are three main design issues involved in the block fractal image coding system. First are partitioning techniques which include range block partitioning and domain block partitioning. As mentioned earlier, the domain block is larger than the range block. Dividing the image into square blocks is the simplest partitioning approach. The second issue is the choice of distortion measurement and a searching method. The common distortion measurement in the block fractal image coding is the root mean square (*RMS*) error. The closest match between the range block and transformed domain block is found by the *RMS* distortion measurement. The third method is the selection of a set of contractive transformations defined consistently with a partition.

It is noted that the partitioned *IFS* (*PIFS*)-based fractal image coding has several similar features with image vector quantization. Both coding schemes are block-based coding schemes and need a codebook for encoding. For *PIFS*-based fractal image coding the domain blocks can be seen as forming a virtual codebook. One difference is that the fractal image coding does not need to transmit the codebook data (domain blocks) to the decoder while VQ does. The second difference is the block size. For VQ, block size for the code vector and input vector is the same while in *PIFS* fractal coding the size of the domain block is different from the size of the range blocks. Another

difference is that in fractal image coding the image itself serves as the codebook, while this is not true for VQ image coding.

9.3.3 Other Fractal Image Coding Methods

Besides the *IFS*-based fractal image coding, there are several other fractal image coding methods. One is the segmentation-based coding scheme using fractal dimensions. In this method, the image is segmented into regions based on the properties of the human visual system (*HVS*). The image is segmented into the regions, each of these regions is homogeneous in the sense of having similar features by visual perception. This is different from the traditional image segmentation techniques that try to segment an image into regions of constant intensity. For a complicated image, good representation of an image needs a large number of small segmentations. However, in order to obtain a high compression ratio, the number of segmentations is limited. The trade-off between image quality and bit rate has to be considered. A parameter, fractal dimension, is used as a measure to control the trade-off. Fractal dimension is a characteristic of a fractal. It is related to a metric property such as the length of a curve and the area of a surface. The fractal dimension can provide a good measurement of the perceptual roughness of the curve and surface. For example, if we use many segments of straight lines to approximate a curve, by increasing the length of the straight lines perceptually rougher curves are represented.

9.4 MODEL-BASED CODING

9.4.1 Basic Concept

In the model-based coding, an image model that can be a 2-D model for still images or a 3-D model for video sequence is first constructed. At the encoder, the model is used to analyze the input image. The model parameters are then transmitted to the decoder. At the decoder the reconstructed image is synthesized by the model parameters, with the same image model used at the encoder. This basic idea of model-based coding is shown in the Figure 9.8. Therefore, the basic techniques in the model-based coding are the image modeling, image analysis, and image synthesis techniques. Both image analysis and synthesis are based on the image model. The image modeling techniques used for image coding can normally be divided into two classes: structure modeling and motion modeling. Motion modeling is usually used for video sequences and moving pictures, while structure modeling is usually used for still image coding. The structure model is used for reconstruction of a 2-D or 3-D scene model.

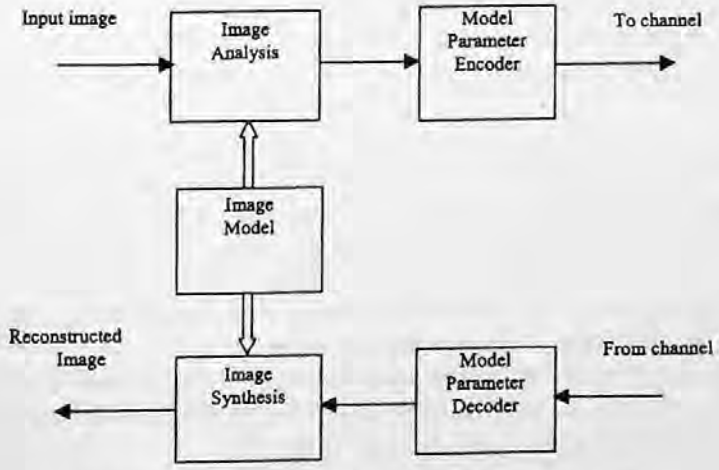


FIGURE 9.8 Basic principle of model-based coding.

9.4.2 Image Modeling

The geometric model is usually used for image structure description. The geometric model can be classified into a surface-based description and volume-based description. The major advantage of surface description is that such description is easily converted into a surface representation that can be encoded and transmitted. In these models the surface is approximated by planar polygonal patches such as triangle patches. The surface shape is represented by a set of points that represent the vertices of these triangle meshes. The size of these triangle patches can be adjusted according to the surface complexity. In other words, for more complicated areas, more triangle meshes are needed to approximate the surface while for smoothing areas, the mesh sizes can be larger or less vertices of the triangle meshes are needed to represent the surface. The volume-based description is a natural approach for modeling most solid world objects. Most existing research work on volume-based description focuses on the parametric volume description. The volume-based description is used for 3-D objects or video sequences.

However, model-based coding is successfully applicable only to certain kinds of images since it is very hard to find general image models suitable for most natural scenes. The few successful examples of image models include the human face, head, and body. These models are developed for the analysis and synthesis of moving images. The face animation has been adopted for the MPEG-4 visual coding. The body animation is under consideration for version 2 of MPEG-4 visual coding.

9.5 SUMMARY

In this chapter three kinds of image coding techniques, vector quantization, fractal image coding, and model-based coding, which are not used in the current standards, have been presented. All three techniques have several important features such as very high compression ratios for certain kinds of images and very simple decoding procedures (especially for VQ). However, due to some limitations these techniques have not been adopted by industry standards. It should be noted that recently the facial model face animation technique has been adopted for the MPEG-4 visual standard (mpeg4 visual).

9.6 EXERCISES

9-1. In the modified residual VQ described in Equation 9.5, with a 4×4 block and 8 bits for each pixel of original image, we use 8 bits for coding block mean and block variance. We want to obtain the final bit rate of 2 bits per pixel. What codebook size do we have to use for the coding residual, assuming that we use fixed-length coding to code vector indices?

9-2. In the block truncation coding described in Equation 9.7, what is the bit rate for a block size of 4×4 if the mean and variance are both encoded with 8 bits? Do you have any suggestions for reducing the bit rate without seriously affecting the reconstruction quality?

9-3. Is the codebook generated with the LBG algorithm local optimum? List the several important factors that will affect the quality of codebook generation.

9-4. In image coding using VQ, what kind of problems will be caused by using the codebook in practical applications (hint: changing bit rate).

9-5. What is the most important improvement of the lattice VQ over traditional VQ in practical application. What is the key issue for lattice VQ for image coding application?

9-6. Write a subroutine to generate a fern leaf (using C).

REFERENCES

Baker, R. L. and R. M. Gray, Image compression using nonadaptive spatial vector quantization, ISCAS'83, 1983, 55-61.

Barnsley, M.F. and A.E. Jacquin, Application of recurrent iterated function systems, SPIE, vol. 1001, *Visual Communications and Image Processing*, 1988, 122-131.

Barnsley, M. and L. P. Hurd, *Fractal Image Compression*, A.K. Peters, Wellesley, MA, 1993.

Conway, J. H. and N. J. A. Slone, A fast encoding method for lattice codes and quantizers, *IEEE Trans. Inform. Theory*, vol. IT-29, 820-824, 1983.

Conway, J. H. and N. J. A. Slone, *Sphere Packings, Lattices and Groups*, New York: Springer-Verlag, 1991.

Delp, E. J. and D. R. Mitchell, Image compression using block truncation coding, *IEEE Trans. Commun.*, COM-27, 1979.

Dunham, M. and R. Gray, An algorithm for the design of labelled-transition finite-state vector quantizer, *IEEE Trans. Commun.*, COM-33, 83-89, 1985.

Equits, W. H. A new vector quantization clustering algorithm, *IEEE Trans. Acoust. Speech Signal Process.*, 37, 1568-1575, 1989.

Fischer, T. R., A paramid vector quantization, *IEEE Trans. Inform. Theory*, vol. IT-32, 568-583, 1986.

Fisher, Y. *Fractal Image Compression — Theory and Application*, New York: Springer-Verlag, 1994.

Foster, J., R. M. Gray, and M. O. Dunham, Finite-state vector quantization for waveform coding, *IEEE Trans. Inf. Theory*, IT-31, 348-359, 1985.

Gersho, A. and B. Ramamurthi, Image coding using vector quantization, ICASSP'82, Paris, May 1982, 428-431.

Gersho, A. On the structure of vector quantizer, *IEEE Trans. Inf. Theory*, IT-28, 157-166, 1982.

Hang, H. M. and J. W. Woods, Predictive vector quantization of images, *IEEE Trans. Commun.*, COM-33, 1208-1219, 1985.

ISO/IEC 14496-2, Coding of Audio-Visual Objects, Part 2, Dec. 18, 1998.

Jacquin, A. E. Fractal Image Coding: A Review, *Proc. IEEE*, 81(10), 1451-1465, 1993.

Jang, J. and S. A. Rajala, Segmentation-based image coding using fractals and the human visual system, IEEE Int. Conf. Acoust. Speech Signal Processing, 1990, pp. 1957-1960.

Laroia, R. and N. Farvardin, A structured fixed rate vector quantizer derived from a variable length scaler quantizer: I & II, *IEEE Trans. Inform. Theory*, vol. 39, 851-876, 1993.

Li, H., A. Lundmark, and R. Forchheimer, Image Sequence Coding at Very Low Bitrates: A Review, *IEEE Trans. Image Process.*, 3(5), 1994.

Li, W. and Y. Zhang, Vector-based signal processing and quantization for image and video compression, *Proc. IEEE*, Volume 83(2), 317-335, 1995.

Li, W. et al., A video coding algorithm using vector-based tehnique, *IEEE Trans. Circuits Syst. Video Technol.*, 7(1), 146-157, 1997.

Linde, Y. A. Buzo, and R. M. Gray, An algorithm for vector quantizer design, *IEEE Trans. Commun.*, 28, 84-95, 1980.

Lu, G. Fractal image compression, *Signal Process. Image Commun.*, 5, 327-343, 1993.

Murakami, T., K. Asai, and E. Yamazaki, Vector quantization of video signals, *Electron. Lett.*, 7, 1005-1006, 1982.

Nasrabadi, N. M. and R. A. King, Image Coding using Vector Quantization: A Review, *IEEE Trans. Commun.*, COM-36(8), 957-971, 1988.

Stewart, L. C., R. M. Gray and Y. Linde, The design of trellis waveform coders, *IEEE Trans. Commun.*, COM-30, 702-710, 1982.

Sun, H. and M. Goldberg, Image coding using LPC with vector quantization, *IEEE Proc. Int. Conf. Digital Signal Processing*, Florence, Italy, Sept. 1984, 508-512.

Udpikar, V. R. and J. P. Raina, BTC image coding using vector quantization, *IEEE Trans. Commun.*, COM-35, 352-356, 1987.

Walach, E. and E. Karnin, A fractal-based approach to image compression, *ICASSP 1986*, 529-532.

Wang, C., H. Q. Cao, W. Li, and K. K. Tzeng, Lattice Labeling Algorithm for Vector Quantization, *IEEE Trans. Circuits Syst. Video Technol.*, 8(2), 206-220, 1998.

Section III

Motion Estimation and Compression

10 Motion Analysis and Motion Compensation

Up to this point, what we have discussed in the previous chapters were basic techniques in image coding, specifically, techniques utilized in still image coding. From here on, we are going to address the issue of video sequence compression. To fulfill the task, we will first define the concepts of image and video sequences. Then we address the issue of interframe correlation between successive frames. Two techniques in exploitation of interframe correlation, frame replenishment and motion-compensated coding, will then be discussed. The rest of the chapter covers the concepts of motion analysis and motion compensation in general.

10.1 IMAGE SEQUENCES

In this section the concept of various image sequences is defined in a theoretical and systematic manner. The relationship between image sequences and video sequences is also discussed.

It is well known that in the 1960s the advent of the semiconductor computer and the space program swiftly brought the field of digital image processing into public focus. Since then the field has experienced rapid growth and has entered every aspect of modern technology. Since the early 1980s, digital image sequence processing has been an attractive research area (Huang, 1981a, 1983). This is not surprising, because an image sequence, as a collection of images, may provide more information than a single image frame. The increased computational complexity and memory space associated with image sequence processing are becoming more affordable due to more advanced, achievable computational capability. With the tremendous advancements continuously made in VLSI computer and information processing, image and video sequences are evermore indispensable elements of modern life. While the pace and the future of this development cannot be predicted, one thing is certain: this process is going to drastically change all aspects of our world in the next several decades.

As far as image sequence processing is concerned, it is noted that in addition to temporal image sequences, stereo image pair and stereo image sequences also received attention in the middle of the 1980s (Waxman and Duncan, 1986). The concepts of temporal and spatial image sequences, and the imaging space (which may be considered as a next-higher-level unification of temporal and spatial image sequences) may be illustrated as follows.

Consider a sensor located in a specific position in the three-dimensional (3-D) world space. It generates images about the scene, one after another. As time goes by, the images form a sequence. The set of these images can be represented with a brightness function g(x,y,t), where x and y are coordinates on the image planes. This is referred to as a *temporal image sequence*. This is the basic outline about the brightness function g(x,y,t) dealt with by researchers in both computer vision, e.g., Horn and Schunck (1980) and signal processing fields, e.g., Pratt (1979).

Now consider a generalization of the above basic outline. A sensor, as a solid article, can be translated (in three free dimensions) and rotated (in two free dimensions). It is noted that here the rotation of a sensor about its optical axis is not counted, since the images generated will remain unchanged when this type of rotation takes place. So, we can obtain a variety of images when a sensor is translated to different coordinates and rotated to different angles in the 3-D world space. Equivalently, we can imagine that there is an infinite number of sensors in the 3-D world space

that occupies all possible spatial coordinates and assumes all possible orientations at each coordinate; i.e., they are located on all possible positions. At one specific moment, all of these images form a set, which can be referred to as a *spatial image sequence*. When time varies, these sets of images form a much larger set of images, called an *imaging space*.

Clearly, it is impossible to describe such a set of images by using the above-mentioned g(x,y,t). Instead, it should be described by a more general brightness function,

$$g(x, y, t, \bar{s}), \tag{10.1}$$

where $\bar{s}$ indicates the sensor's position in the 3-D world space; i.e., the coordinates of the sensor center and the orientation of the optical axis of the sensor. Hence $\bar{s}$ is a 5-D vector. That is,

$$\bar{s} = (\bar{x}, \bar{y}, \bar{z}, \beta, \gamma), \tag{10.2}$$

where $\bar{x}$, $\bar{y}$, and $\bar{z}$ represent the coordinates of the optical center of the sensor in the 3-D world space; and β and γ represent the orientation of the optical axis of the sensor in the 3-D world space. More specifically, each sensor in the 3-D world space may be considered associated with a 3-D Cartesian coordinate system such that its optical center is located on the origin and its optical axis is aligned with the *OZ* axis. In the 3-D world space we choose a 3-D Cartesian coordinate system as the reference coordinate system. Hence, a sensor with its Cartesian coordinate system coincident with the reference coordinate system has its position in the 3-D world space denoted by $\bar{s}$ = (0,0,0,0,0). An arbitrary sensor position denoted by $\bar{s}$ = ($\bar{x}$, $\bar{y}$, $\bar{z}$, $\hat{a}$, $\bar{a}$) can be described as follows. The sensor's associated Cartesian coordinate system is first shifted from the reference coordinate system in the 3-D world space with its origin settled at ($\bar{x}$, $\bar{y}$, $\bar{z}$) in the reference coordinate system. Then it is rotated with the rotation angles β and γ being the same as Euler angles (Shu and Shi, 1991; Shi et al., 1994). Figure 10.1 shows the reference coordinate system and an arbitrary Cartesian coordinate system (indicating an arbitrary sensor position). There, *oxy* and $o'x'y'$ represent, respectively, the related image planes.

Assume now a world point *P* in the 3-D space that is projected onto the image plane as a pixel with the coordinates x_p and y_p. Then, x_p and y_p are also dependent on t and $\bar{s}$. That is, the coordinates of the pixel can be denoted by $x_p = x_p(t, \bar{s})$ and $y_p = y_p(t, \bar{s})$. So generally speaking, we have

$$g = g\left(x_p(t, \bar{s}), y_p(t, \bar{s}), t, \bar{s}\right). \tag{10.3}$$

As far as temporal image sequences are concerned, let us take a look at the framework of Pratt (1979), and Horn and Schunck (1980). There, $g = g(x_p(t), y_p(t), t)$ is actually a special case of Equation 10.3, i.e., $g = g(x_p(t, \bar{s} = \text{constant vector}), y_p(t, \bar{s} = \text{constant vector}), (t, \bar{s} = \text{constant vector})$. In other words, the variation of $\bar{s}$ is restricted to be zero, i.e., $\Delta\bar{s} = 0$. This means the sensor is fixed in a certain position in the 3-D world space.

Obviously, an alternative is to define the imaging space as a set of all temporal image sequences; i.e., those taken by sensors located at all possible positions in the 3-D world space. Stereo image sequences can thus be viewed as a proper subset of the imaging space, just like a stereo pair of images can be considered as a proper subset of a spatial image sequence.

In summary, the imaging space is a collection of all possible forms assumed by the general brightness function g (x, y, t, $\bar{s}$). Each picture taken by a sensor located on a particular position at a specific moment is merely a special cross section of this imaging space. Both temporal and spatial image sequences are special proper subsets of the imaging space. They are in the middle level, between the imaging space and the individual images. This hierarchical structure is depicted in Figure 10.2.

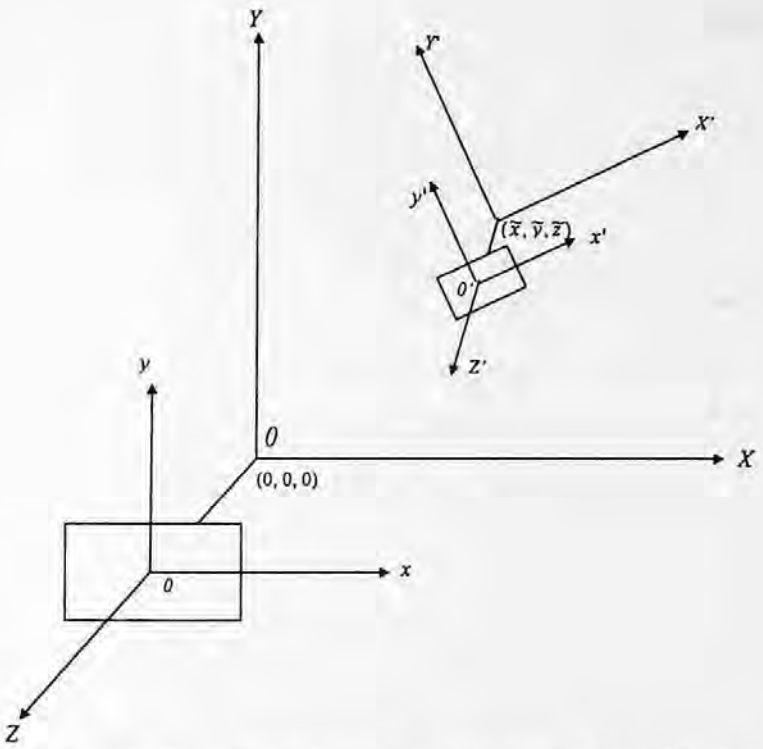


FIGURE 10.1 Two sensor positions $\bar{s} = (0,0,0,0,0)$ and $\bar{s} = (\bar{x}, \bar{y}, \bar{z}, \hat{a}, \bar{a})$.

Before we conclude this section, we should discuss the relationship between image sequences and video sequences. It is noted that the term *video* is used very often nowadays in addition to the terms *image frames* and *image sequence*. It is necessary to pause for a while to discuss the relationship between these terms. Image frames and image sequence have been defined clearly above with the introduction of the concept of the imaging space. Video can mean an individual video frame or video sequences. It refers, however, to those frames and sequences that are associated with the visible frequency band in the electromagnetic spectrum. For image frames and image sequences, there is no such restriction. For instance, infrared image frames and sequences correspond to a band outside the visible band in the spectrum. From this point of view, the scope of image frames and sequences is wider than that of video frames and sequences. When the visible band is concerned, the terms *image frame and sequence* are interchangeable with that *video frame and sequence*.

Another point we would like to bring to the reader's attention is as follows. Though video is referred to as visual information, which includes both a single frame and frame sequences, in practice it is often used to mean sequences exclusively. Such an example can be found in *Digital Video Processing* (Tekalp, 1995).

In this book, we use *image compression* to indicate still image compression, and *video compression* to indicate video sequence compression. Readers should keep in mind, however, that (1) video can mean a single frame or sequences of frames; and (2) the scope of image is wider than that of video, and video is more pertinent to multimedia engineering.

10.2 INTERFRAME CORRELATION

As far as video compression is concerned, all the techniques discussed in the previous chapters are applicable. By this we mean two classes of techniques. The first class, which is also the most straightforward way to handle video compression, is to code each frame separately. That is,

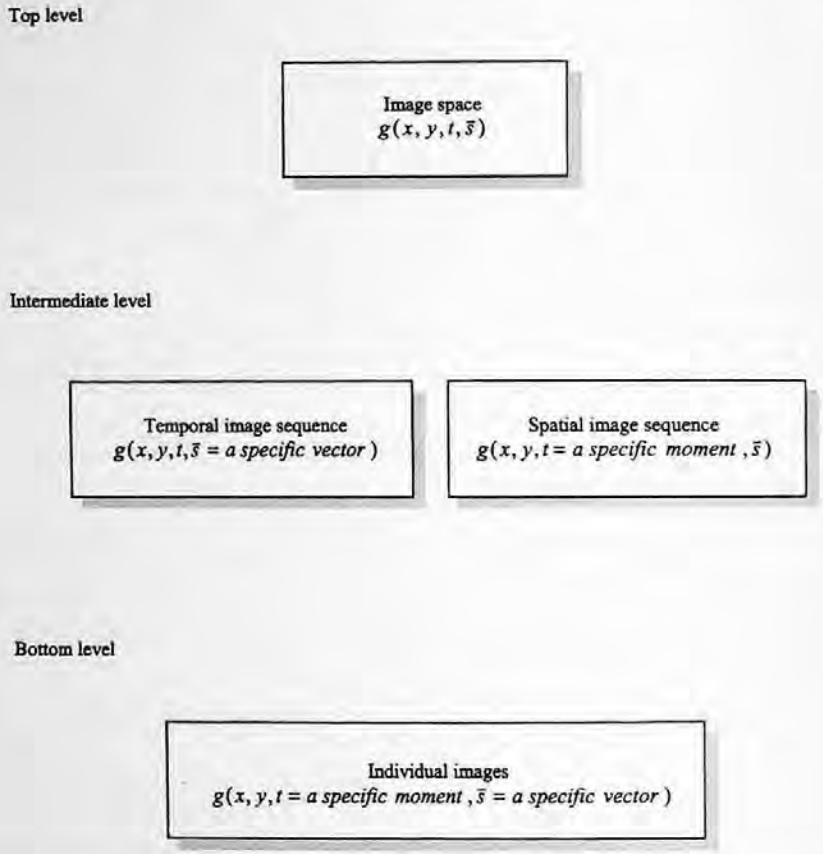


FIGURE 10.2 A hierarchical structure.

individual frames are coded independently on each other. For instance, using a JPEG compression algorithm to code each frame in a video sequence results in *motion JPEG* (Westwater and Furht, 1997). In the second class, methods utilized for still image coding can be generalized for video compression. For instance, (DCT) transform coding can be generalized and applied to video coding by extending 2-D DCT to 3-D DCT. That is, instead of 2-D DCT, say, 8×8, applied to a single image frame, we can apply 3-D DCT, say, $8 \times 8 \times 8$, to a video sequence. Refer to Figure 10.3. That is, 8 blocks of 8×8 each located, respectively, at the same position in one of the 8 successive frames from a video sequence are coded together with the 3-D DCT. It was reported that this 3-D DCT technique is quite efficient (Lim, 1990; Westwater and Furht, 1997). In addition, the DPCM technique and the hybrid technique can be generalized and applied to video compression in a similar fashion (Jain, 1989; Lim, 1990). It is noted that in the second class of techniques several successive frames are grouped and coded together, while in the first class each frame is coded independently.

Video compression has its own characteristics, however, that make it quite different from still image compression. The major difference lies in the exploitation of interframe correlation that exists between successive frames in video sequences, in addition to the intraframe correlation that exists within each frame. As mentioned in Chapter 1, the interframe correlation is also referred to as *temporal redundancy*, while the intraframe correlation is referred to as *spatial redundancy*. In order to achieve coding efficiency, we need to remove these redundancies for video compression. To do so we must first understand these redundancies.

Consider a video sequence taken in a videophone service. There, the camera is static most of the time. A typical scene is a head-and-shoulder view of a person imposed on a background. In this type of video sequence the background is usually static. Only the speaker is experiencing motion, which is not severe. Therefore, there is a strong similarity between successive frames, that

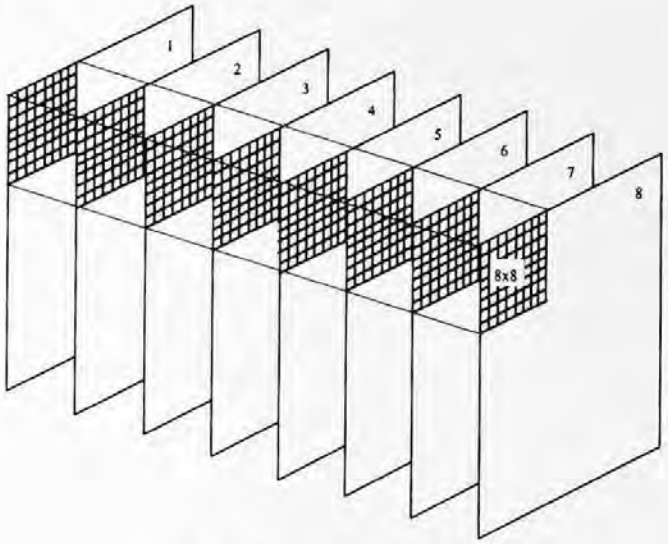


FIGURE 10.3 3-D DCT of $8 \times 8 \times 8$.

is, a strong adjacent-frame correlation. In other words, there is a strong interframe correlation. It was reported by Mounts (1969) that when using videophone-like signals with moderate motion in the scene, on average, less than one-tenth of the elements change between frames by an amount which exceeds 1% of the peak signal. Here, a 1% change is regarded as significant. Our experiment on the first 40 frames of the Miss America sequence supports this observation. Two successive frames of the sequence, frames 24 and 25, are shown in Figure 10.4.

Now, consider a video sequence generated in a television broadcast. It is well known that television signals are generated with a scene scanned in a particular manner in order to maintain a steady picture for a human being to view, regardless of whether there is a scenery change or not. That is, even if there is no change from one frame to the next, the scene is still scanned constantly. Hence there is a great deal of frame-to-frame correlation (Haskell et al., 1972b; Netravali and Robbins, 1979). In TV broadcasts, the camera is most likely not static, and it may be panned, tilted, and zoomed. Furthermore, more movement is involved in the scene. As long as the TV frames are taken

FIGURE 10.4 Two frames of the Miss America sequence: (a) frame 24, (b) frame 25.

densely enough, then most of the time we think the changes between successive frames are due mainly to the apparent motion of the objects in the scene that takes place during the frame intervals. This implies that there is also a high correlation between sequential frames. In other words, there is an interframe redundancy (interpixel redundancy between pixels in successive frames). There is more correlation between television picture elements along the frame-to-frame temporal dimension than there is between adjacent elements in a single frame along the spatial dimension. That is, there is generally more interframe correlation than intraframe correlation. Taking advantage of the interframe correlation, i.e., eliminating or decreasing the uncertainty of successive frames, leads to video data compression. This is analogous to the case of still image coding with the DPCM technique, where we can predict part of an image by knowing the other part. Now the knowledge of the previous frames can remove the uncertainty of the next frame. In both cases, knowledge of the past removes the uncertainty of the future, leaving less actual information to be transmitted (Kretzmer, 1952). In Chapter 16, we will see that the words "past" and "future" used here are not necessary. They can be changed, respectively, to "some frames" and "some other frames" in advanced video coding techniques such as MPEG. There, a frame might be predicted from both its previous frames and its future frames.

At this point, it becomes clear that the second class of techniques (mentioned at the beginning of this section), which generalizes techniques originally developed for still image coding and applies them to video coding, exploits interframe correlation. For instance, in the case of the 3-D DCT technique, a strong temporal correlation causes an energy compaction within the low temporal frequency region. The 3-D DCT technique drops transform coefficients associated with high temporal frequency, thus achieving data compression.

The two techniques specifically developed to exploit interframe redundancy, i.e., frame replenishment and motion-compensated coding, are introduced below. The former is the early work, while the latter is the more popular recent work.

10.3 FRAME REPLENISHMENT

As mentioned in Chapter 3, frame-to-frame redundancy has long been recognized in TV signal compression. The first few experiments of a frame sequence coder exploiting interframe redundancy may be traced back to the 1960s (Seyler, 1962, 1965; Mounts, 1969). In (Mounts, 1969) the first real demonstration was presented and was termed *conditional replenishment*. This frame replenishment technique can be briefly described as follows. Each pixel in a frame is classified into *changing* or *unchanging* areas depending on whether or not the intensity difference between its present value and its previous one (the intensity value at the same position on the previous frame) exceeds a threshold. If the difference does exceed the threshold, i.e., a *significant* change has been identified, the address and intensity of this pixel are coded and stored in a buffer and then transmitted to the receiver to replenish intensity. For those unchanging pixels, nothing is coded and transmitted. Their previous intensities are repeated in the receiver. It is noted that the buffer is utilized to make the information presented to the transmission channel occur at a smooth bit rate. The threshold is to make the average replenishment rate match the channel capacity.

Since the replenishment technique only encodes those pixels whose intensity value has changed significantly between successive frames, its coding efficiency is much higher than the coding techniques which encode every pixel of every frame, say, the DPCM technique applied to each single frame. In other words, utilizing interframe correlation, the replenishment technique achieves a lower bit rate, while keeping the equivalent reconstructed image quality.

Much effort had been made to further improve this type of simple replenishment algorithm. As mentioned in the discussion of 3-D DPCM in Chapter 3, for instance, it was soon realized that intensity values of pixels in a changing area need not be transmitted independently of one another. Instead, using both spatial and temporal neighbors' intensity values to predict the intensity value

FIGURE 10.5 Dirty window effect.

of a changing pixel leads to a *frame-difference* predictive coding technique. There, the differential signal is coded instead of the original intensity values, thus achieving a lower bit rate. For more detail, readers are referred to Section 3.5.2. Another example of the improvements is that measures have been taken to distinguish the intensity difference caused by noise from those associated with changing to avoid the dirty window effect, whose meaning is given in the next paragraph. For more detailed information on these improvements over the simple frame replenishment technique, readers are referred to two excellent reviews by Haskell et al. (1972b, 1979).

The main drawback associated with the frame replenishment technique is that it is difficult to handle frame sequences containing more rapid changes. When there are more rapid changes, the number of pixels whose intensity values need to be updated increases. In order to maintain the transmission bit-rate at a steady and proper level the threshold has to be raised, thus causing many slow changes that cannot show up in the receiver. This poorer reconstruction in the receiver is somewhat analogous to viewing a scene through a dirty window. This is referred to as the dirty window effect. The result of one experiment on the dirty window effect is displayed in Figure 10.5. From frame 22 to frame 25 of the Miss America sequence, there are 2166 pixels (less than 10% of the total pixels) that change their gray level values by more than 1% of the peak signal. When we only update the gray level values for 25% (randomly chosen) of these changing pixels, we can clearly see the dirty window effect. When rapid scene changes exceed a certain level, buffer saturation will result, causing picture breakup (Mounts, 1969). Motion-compensated coding, which is discussed below, has been proved to be able to provide better performance than the replenishment technique in situations with rapid changes.

10.4 MOTION-COMPENSATED CODING

In addition to the frame-difference predictive coding technique (a variant of the frame replenishment technique discussed above), another technique: displacement-based predictive coding, was developed at almost the same time (Rocca, 1969; Haskell and Limb, 1972a). In this technique, a motion model is assumed. That is, the changes between successive frames are considered due to the translation of moving objects in the image planes. Displacement vectors of objects are first estimated. Differential signals between the intensity value of the picture elements in the moving areas and those of their counterparts in the previous frame, which are translated by the estimated

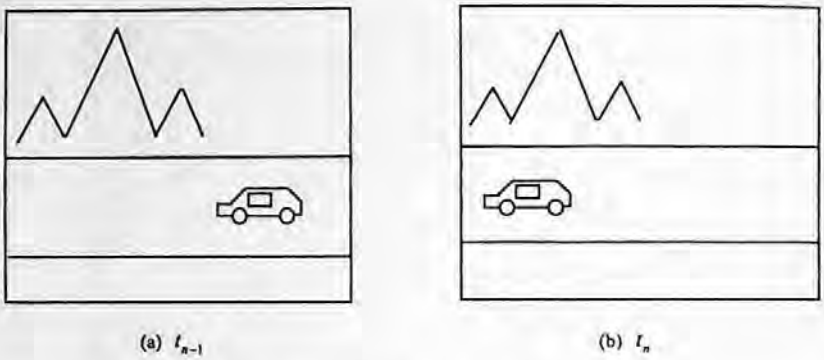


FIGURE 10.6 Two consecutive frames of a video sequence.

displacement, are encoded. This approach, which takes motion into account to compress video sequences, is referred to as motion-compensated predictive coding. It has been found to be much more efficient than the frame-difference prediction technique.

To understand the above statement, let us look at the diagram shown in Figure 10.6. Assume a car translating from the right side to the left side in the image planes at a uniform speed during the time interval between the two consecutive image frames. Other than this, there are no movements or changes in the frames. Under this circumstance, if we know the displacement vector of the car on the image planes during the time interval between two consecutive frames, we can then predict the position of the car in the latter frame from its position in the former frame. One may think that if the translation vector is estimated well, then so is the prediction of the car position. This is true. In reality, however, estimation errors occurring in determination of the motion vector, which may be caused by various noises existing in the frames, may cause the predicted position of the car in the latter frame to differ from the actual position of the car in the latter frame.

The above translational model is a very simple one; it cannot accommodate motions other than translation, say, rotation, and camera zooming. Occlusion and disocclusion of objects make the situation even more complicated since in the occlusion case some portions of the images may disappear, while in the disocclusion case some newly exposed areas may appear. Therefore, the prediction error is almost inevitable. In order to have good-quality frames in the receiver, we can find the prediction error by subtracting the predicted version of the latter frame from the actual version of latter frame. If we encode both the displacement vectors and the prediction error, and transmit these data to the receiver, we may be able to obtain high-quality reconstructed images in the receiver. This is because in the receiving end, using the displacement vectors transmitted from the transmitter and the reconstructed former frame, we can predict the latter frame. Adding the transmitted prediction error to the predicted frame, we may reconstruct the latter frame with satisfactory quality. Furthermore, if manipulating the procedure properly, we are able to achieve data compression.

The displacement vectors are referred to as side or overhead information to indicate their auxiliary nature. It is noted that motion estimation drastically increases the computational complexity of the coding algorithm. In other words, the higher coding efficiency is obtained in motion-compensated coding, but with a higher computational burden. As we pointed out in Section 10.1, this is both technically feasible and economically desired since the cost of digital signal processing decreases much faster than that of transmission (Dubois et al., 1981).

Motion-compensated video compression has become a major development in coding. For more information, readers should refer to several excellent survey papers (Musmann et al., 1985; Zhang et al., 1995; Kunt, 1995).

The common practice of motion-compensated coding in video compression can be split into the following three stages. First, the *motion analysis* stage; that is, displacement vectors for either

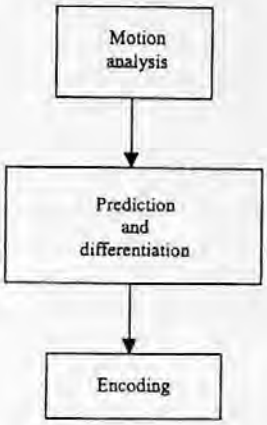


FIGURE 10.7 Block diagram of motion-compensated coding.

every pixel or a set of pixels in image planes from sequential images are estimated. Second, the present frame is predicted by using estimated motion vectors and the previous frame. The prediction error is then calculated. This stage is called *prediction and differentiation*. The third stage is *encoding*. The prediction error (difference between the present and the predicted present frames) and the motion vectors are encoded. Through an appropriate manipulation, the total amount of data for both the motion vectors and prediction error is expected to be much less than the raw data existing in the image frames, thus resulting in data compression. A block diagram of motion-compensated coding is shown in Figure 10.7.

Before leaving this section, we compare the frame replenishment technique with the motion-compensated coding technique. Qualitatively speaking, we see from the above discussion that the replenishment technique is also a kind of predictive coding in nature. This is particularly true if we consider the frame-difference predictive technique used in frame replenishment. There, it uses a pixel's intensity value in the previous frame as an estimator of its intensity value in the present frame.

Now let's look at motion-compensated coding. Consider a pixel on the present frame. Through motion analysis, the motion-compensated technique finds its counterpart in the previous frame. That is, a pixel in the previous frame is identified such that it is supposed to translate to the position on the present frame of the pixel under consideration during the time interval between successive frames. This counterpart's intensity value is used as an estimator of that of the pixel under consideration. We can see that the model used for motion-compensated coding is much more advanced than that used for frame replenishment, therefore, it achieves a much higher coding efficiency. A motion-compensated coding technique that utilized the first pel-recursive algorithm for motion estimation (Netravali and Robbins, 1979) was reported to achieve a bit rate 22 to 50% lower than that obtained by simple frame-difference prediction, a version of frame replenishment.

The more advanced model utilized in motion-compensated coding, on the other hand, leads to higher computational complexity. Consequently, both the coding efficiency and the computational complexity in motion-compensated coding are higher than that in frame replenishment.

10.5 MOTION ANALYSIS

As discussed above, we usually conduct motion analysis in video sequence compression. There, 2-D displacement vectors of a pixel or a group of pixels on image planes are estimated from given image frames. Motion analysis can be viewed from a much broader point of view. It is well known that the vision systems of both humans and animals observe the outside world to ascertain motion and to navigate themselves in the 3-D world space. Two groups of scientists study vision. Scientists

in the first group, including psychophysicists, physicians, and neurophysiologists study human and animal vision. Their goal is to understand biological vision systems — their operation, features, and limitations. Computer scientists and electrical engineers form the second group. As pointed out by Aggarwal and Nandhakumar (1988), their ultimate goal is to develop computer vision systems with the ability to navigate, recognize, and track objects, and estimate their speed and direction. Each group benefits from the research results of the other group. The knowledge and results of research in psychophysics, physiology, and neurophysiology have influenced the design of computer vision systems. Simultaneously, the research results achieved in computer vision have provided a framework in modeling biological vision systems and have helped in remedying faults in biological vision systems. This process will continue to advance research in both groups, hence benefiting society.

10.5.1 Biological Vision Perspective

In the field of biological vision, most scientists consider motion perception as a two-step process, even though there is no ample biological evidence to support this view (Singh, 1991). The two steps are measurement and interpretation. The first step measures the 2-D motion projected on the imaging surfaces. The second step interprets the 2-D motion to induce the 3-D motion and structure on the scene.

10.5.2 Computer Vision Perspective

In the field of computer vision, motion analysis from image sequences is traditionally split into two steps. In the first step, intermediate variables are derived. By *intermediate variables*, we mean 2-D motion parameters in image planes. In the second step, 3-D motion variables, say, speed, displacement, position, and direction, are determined.

Depending on the different intermediate results, all approaches to motion analysis can be basically classified into two categories: feature correspondence and optical flow. In the former category, a few distinct features are first extracted from image frames. For instance, consider an image sequence containing an aircraft. Two consecutive frames are shown in Figure 10.8. The head and tail of the aircraft, and the tips of its wings may be chosen as features. The correspondence of these features on successive image frames needs to be established. In the second step, 3-D motion can then be analyzed from the extracted features and their correspondence in successive frames. In the latter category of approaches, the intermediate variables are optical flow. An optical flow vector is defined as a velocity vector of a pixel on an image frame. An optical flow field is referred to as the collection of the velocity vectors of all the pixels on the frame. In the first step, optical flow vectors are determined from image sequences as the intermediate variables. In the second step, 3-D motion is estimated from optical flow. It is noted that optical flow vectors are closely

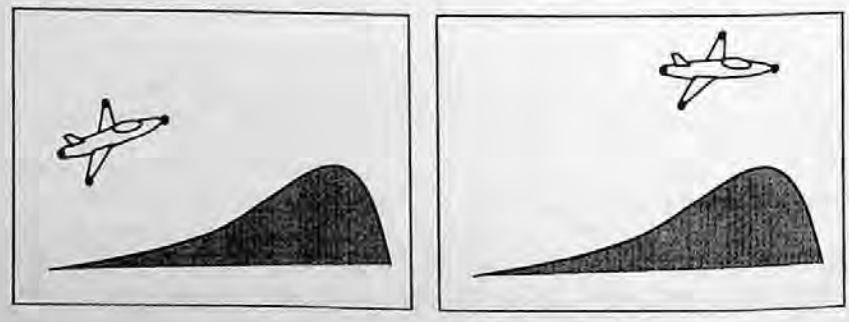


FIGURE 10.8 Feature extraction and correspondence from two consecutive frames in a temporal image sequence.

related to displacement vectors in that a velocity vector multiplying by the time interval between two consecutive frames results in the corresponding displacement vector. Optical flow and its determination will be discussed in detail in Chapter 13.

It is noted that there is a so-called direct method in motion analysis. Contrary to the above optical flow approach, instead of determining 2-D motion variables, (i.e., the intermediate variables), prior to 3-D motion estimation, the direct method attempts to estimate 3-D motion without explicitly solving for the intermediate variables. In (Huang and Tsai, 1981b) the equation characterizing displacement vectors in the 2-D image plane and the equation characterizing motion parameters in 3-D world space are combined so that the motion parameters in 3-D world space can be directly derived. This method has been utilized to recover structure (object surfaces) in 3-D world space as well (Negahdaripour and Horn, 1987; Horn and Weldon, 1988; Shu and Shi, 1993). The direct method has certain limitations. That is, if the geometry of object surfaces is not known in advance, then the method fails.

The feature correspondence approach is sometimes referred to as the discrete approach, while the optical flow approach is sometimes referred to as the continuous approach. This is because the correspondence approach concerns only a set of relatively sparse but highly discriminatory 2-D features on image planes. The optical flow approach is concerned with a dense field of motion vectors.

It has been found that both feature extraction and correspondence establishment are not trivial tasks. Occlusion and disocclusion which, respectively, cause some features to disappear and some features to reappear, make feature correspondence even more difficult. The development of robust techniques to solve the correspondence problem is an active research area and is still in its infancy. So far, only partial solutions suitable for simplistic situations have been developed (Aggarwal and Nandhakumar, 1988). Hence the feature correspondence approach is rarely used in video compression. Because of this, we will not discuss this approach any further.

Motion analysis (sometimes referred to as motion estimation or motion interpretation) from image sequences is necessary in automated navigation. It has played a central role in the field of computer vision since the late 1970s and early 1980s. A great deal of the papers presented at the International Conference on Computer Vision cover this and related topics. Many workshops, symposiums, and special sessions are organized around this subject (Thompson, 1989).

10.5.3 Signal Processing Perspective

In the field of signal processing, motion analysis is mainly considered in the context of bandwidth reduction and/or data compression in the transmission of visual signals. Therefore, instead of the motion in 3-D world space, only the 2-D motion in the image plane is concerned.

Because of the real-time nature in visual transmission, the motion model cannot be very complicated. So far, the 2-D translational model is most frequently assumed in the field. In the 2-D translational model it is assumed that the change between a frame and its previous one is due to the motion of objects in the frame plane during the time interval between two consecutive frames. In many cases, as long as frames are taken densely enough, this assumption is valid. By *motion analysis* we mean the estimation of translational motion — either the displacement vectors or velocity vectors. With this kind of motion analysis, one can apply the motion-compensated coding discussed above, making coding more efficient.

Basically there are three techniques in 2-D motion analysis: correlation, and recursive and differential techniques. Philosophically speaking, the first two techniques belong to the same group: region matching.

Refer to Figure 10.6, where the moving car is the object under investigation. By *motion analysis* we mean finding the displacement vector, i.e., a vector representing the relative positions of the car in the two consecutive frames. With region matching, one may consider the car (or a portion of the car) as a region of interest, and seek the best match between the two regions in the two

frames: specifically, the region in the present frame and the region in the previous frame. For identifying the best match, two techniques, the correlation and the recursive methods, differ in methodology. The correlation technique finds the best match by searching the maximum correlation between the two regions in a predefined search range, while the recursive technique estimates the best match by recursively minimizing a nonlinear measurement of the dissimilarities between the two regions.

A couple of comments are in order. First, it is noted that the most frequently used technique in motion analysis is called block matching, which is a type of the correlation technique. There, a video frame is divided into nonoverlapped rectangular blocks with each block having the same size, usually 16 × 16. Each block thus generated is assumed to move as one, i.e., all pixels in a block share the same displacement vector. For each block, we find its best match in the previous frame with correlation. That is, the block in the previous frame, which gives the maximum correlation, is identified. The relative position of these two best matched blocks produces a displacement vector. This block matching technique is simple and very efficient, and will be discussed in detail in Chapter 11. Second, as multimedia finds more and more applications, the regions occupied by arbitrarily-shaped objects (no longer always rectangular blocks) become increasingly important in content-based video retrieval and manipulation. Motion analysis in this case is discussed in Chapter 18. Third, although the recursive technique is categorized as a region matching technique, it may be used for finding displacement vectors for individual pixels. In fact the recursive technique was originally developed for determining displacement vectors of pixels and, hence, it is called pel-recursive. This technique is discussed in Chapter 12. Fourth, both correlation and recursive techniques can be utilized for determining optical flow vectors. Optical flow is discussed in Chapter 13.

The third technique in 2-D motion analysis is the differential technique. This is one of the main techniques utilized in determining optical flow vectors. It is named after the term of differentials because it uses partial differentiation of an intensity function with respect to the spatial coordinates x and y, as well as the temporal coordinate t. This technique is also discussed in Chapter 13.

10.6 MOTION COMPENSATION FOR IMAGE SEQUENCE PROCESSING

Motion analysis has long been considered a key issue in image sequence processing (Huang, 1981a; Shi, 1997). Obviously, in an area like automated navigation, motion analysis plays a central role. From the discussion in this chapter, we see that motion analysis also plays a key role in video data compression. Specifically, we have discussed the concept of motion-compensated video coding in Section 10.4. In this section we would like to consider motion compensation for image sequence processing, in general. Let us first consider motion-compensated interpolation. Then, we will discuss motion-compensated enhancement, restoration, and down-conversion.

10.6.1 Motion-Compensated Interpolation

Interpolation is a simple yet efficient and important method in image and video compression. In image compression, we may only transmit, say, every other row. We then try to interpolate these missing rows from the other half of the transmitted rows in the receiver. In this way, we compress the data to half. Since the interpolation is carried out within a frame, it is referred to as *spatial* interpolation. In video compression, for instance, in videophone service, instead of transmitting 30 frames per second, we may choose a lower frame rate, say, 10 frames per second. In the receiver, we may try to interpolate the dropped frames from the transmitted frames. This strategy immediately drops the transmitted data to one third. Another example is the conversion of a motion picture into an NTSC (National Television System Commission) TV signal. There, every first frame in the

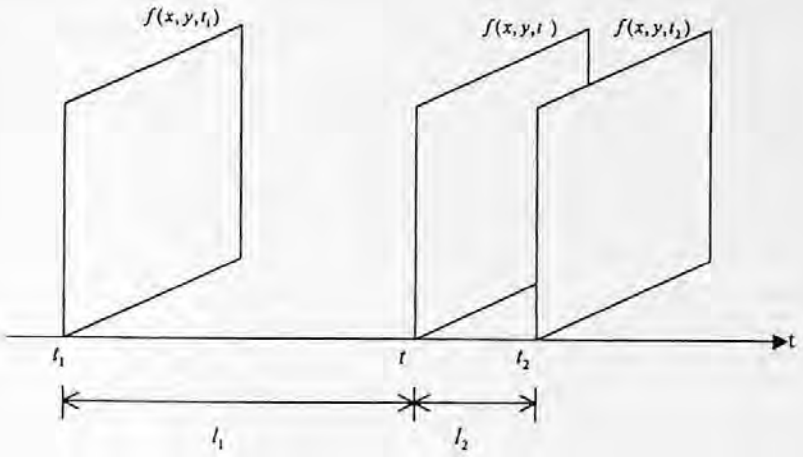


FIGURE 10.9 Weighted linear interpolation.

motion picture is repeated three times and the next frame twice, thus converting a 24-frame-per-second motion picture to a 60-field-per-second NTSC signal. This is commonly referred to as 3:2 pulldown. In these two examples concerning video, interpolation is along the temporal dimension, which is referred to as *temporal* interpolation.

For basic concepts of zero-order interpolation, bilinear interpolation, and polynomial interpolation, readers are referred to signal processing texts, for instance (Lim, 1990). In temporal interpolation, the zero-order interpolation means creation of a frame by copying its nearest frame along the time dimension. The conversion of a 24-frame-per-second motion picture to a 60-field-per-second NTSC signal can be classified into this type of interpolation. Weighted linear interpolation can be illustrated with Figure 10.9.

There, the weights are determined according to the lengths of time intervals, which is similar to the bilinear interpolation widely used in spatial interpolation, except that here only one index (along the time axes) is used, while two indexes (along two spatial axes) are used in spatial bilinear interpolation. That is,

$$f(x,y,t)=\frac{l_2}{l_1+l_2}f(x,y,t_1)+\frac{l_1}{l_1+l_2}f(x,y,t_2) \tag{10.4}$$

If there are one or multiple moving objects existing in successive frames, however, the weighted linear interpolation will blur the interpolated frames. Taking motion into account in the interpolation results in motion-compensated interpolation. In Figure 10.10, we still use the three frames shown in Figure 10.9 to illustrate the concept of motion-compensated interpolation. First, motion between two given frames is estimated. That is, the displacement vectors for each pixel are determined. Second, we choose a frame that is nearer to the frame we want to interpolate. Third, the displacement vectors determined in the first step are proportionally converted to the frame to be created. Each pixel in this frame is projected via the determined motion trajectory to the frame chosen in step 2. In the process of motion-compensated interpolation, spatial interpolation in the frame chosen in step 2 usually is needed.

10.6.2 Motion-Compensated Enhancement

It is well known that when an image is corrupted by additive white Gaussian noise (AWGN) or burst noise, linear low-pass filtering, such as simple averaging or nonlinear low-pass filtering, such as a median filter, performs well in removing the noise. When an image sequence is concerned,

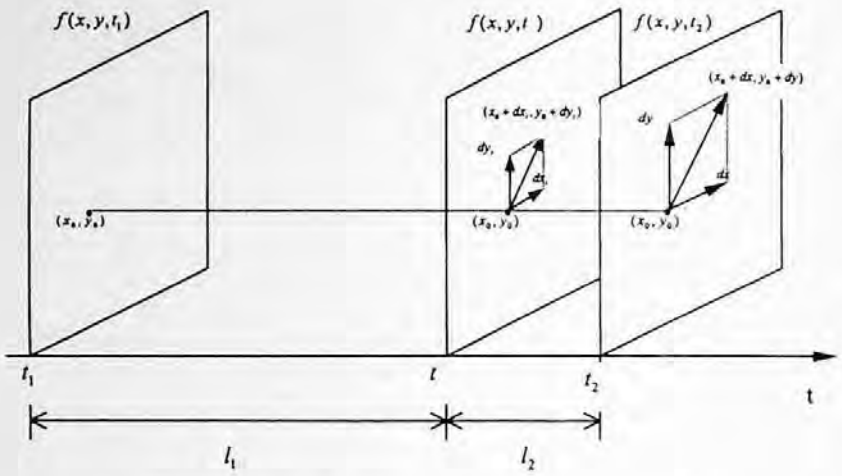


FIGURE 10.10 Motion-compensated interpolation.

we may apply such types of filtering along the temporal dimension to remove noise. This is called temporal filtering. These types of low-pass filtering may blur images, an effect that may become quite serious when motion exists in image planes. The enhancement, which takes motion into account, is referred to as motion-compensated enhancement, and has been found very efficient in temporal filtering (Huang and Hsu, 1981c).

To facilitate the discussion, we consider simple averaging as a means for noise filtering in what follows. It is understood that other filtering techniques are possible, and that everything discussed here is applicable there. Instead of simply averaging n successive image frames in a video sequence, motion-compensated temporal filtering will first analyze the motion existing in these frames. That is, we estimate the motion of pixels in successive frames first. Then averaging will be conducted only on those pixels along the same motion trajectory. In Figure 10.11, three successive frames are shown and denoted by f (x, y, t_1), f (x, y, t_2), and f (x, y, t_3), respectively. Assume that three pixels, denoted by (x_1, y_1), (x_2, y_2), and (x_3, y_3), respectively, are identified to be perspective projections of the same object point in the 3-D world space on the three frames. The averaging is then applied to these three pixels. It is noted that the number of successive frames, n, may not necessarily have to be three. Motion analysis can use any one of the several techniques discussed in Section 10.5.

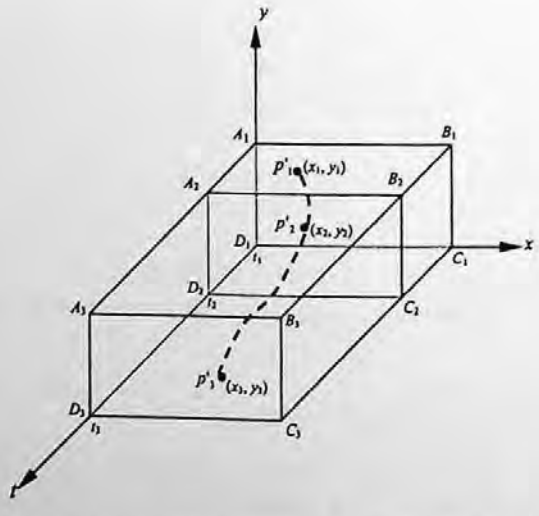


FIGURE 10.11 Motion-compensated temporal filtering.

Motion-compensated temporal filtering is not necessarily implemented pixelwise; it can also be used objectwise, or regionwise.

10.6.3 Motion-Compensated Restoration

Extensive attention has been paid to the restoration of full-length feature films. There, typical artifacts are due to dirt and sparkle. Early work in the detection of these artifacts ignored motion information completely. Late motion estimation has been utilized to detect these artifacts based on the assumption that the artifacts occur occasionally along the temporal dimension. Once the artifacts have been found, motion-compensated temporal filtering and/or interpolation will be used to remove the artifacts. One successful algorithm for the detection and removal of anomalies in digitized animation film can be found in (Tom et al., 1998).

10.6.4 Motion-Compensated Down-Conversion

Here we present one more example in which motion compensation finds application in digital video processing.

It is believed that there will be a need to down-convert a high definition television (HDTV) image sequence for display onto an NTSC monitor during the upcoming transition to digital television broadcast. The most straightforward approach is to fully decode the image sequence first, then apply a prefiltering and subsampling process to each field of the interlaced sequence. This is referred to as a full-resolution decoder (FRD). The merit of this approach is the high quality achieved, while the drawback is a high cost in terms of the large amount of memory required to store the reference frames. To reduce the required memory space, another approach is considered. In this approach, the down-conversion is conducted within the decoding loop and is referred to as a low-resolution decoder (LRD). It can significantly reduce the required memory and still achieve a reasonably good picture quality.

The prediction drift is a major type of artifact existing in the down-conversion. It is defined as the successive blurring of forward-predicted frames with a group of pictures. It is caused mainly by non-ideal interpolation of sub-pixel intensities and the loss of high-frequency data within the block. An optimal set of filters to perform low-resolution motion compensation has been derived to effectively minimize the drift. For details on an algorithm in the down-conversion utilizing an optimal motion compensation scheme, readers are referred to Vetro and Sun (1998).

10.7 SUMMARY

After Section II, still image compression, we shift our attention to video compression. Prior to Section IV, where we discuss various video compression algorithms and standards, however, we first address the issue of motion analysis and motion compensation in this chapter that starts Section III, motion estimation and compensation. This is because video compression has its own characteristics, which are different from those of still image compression. The main difference lies in interframe correlation.

In this chapter, the concept of various image sequences is discussed in a broad scope. In doing so, a single image, temporal image sequences, and spatial image sequences are all unified under the concept of imaging space. The redundancy between pixels in successive frames is analyzed for both videoconferencing and TV broadcast cases. In these applications, there is more interframe correlation than intraframe correlation, in general. Therefore, the utilization of interframe correlation becomes a key issue in video compression.

There are two major techniques in exploitation of interframe correlation: frame replenishment and motion compensation. In the conditional replenishment technique, only those pixel gray level values, whose variation from their counterparts in the previous frame exceeds a threshold, are

encoded and transmitted to the receiver. These pixels are called changing pixels. For pixels other than the changing pixels, their gray values are just repeated in the receiver. This simplest frame replenishment technique achieves higher coding efficiency than coding each pixel in each frame due to the utilization of interframe redundancy. In the more advanced frame replenishment techniques, say, the frame-difference predictive coding technique, both temporal and spatial neighboring gray values of the pixels are used to predict that of a changing pixel. Instead of the intensity values of the changing pixels, the prediction error is encoded and transmitted. Because the variance of the prediction error is smaller than that of the intensity values, this more advanced frame replenishment technique is more efficient than the conditional replenishment technique.

The main drawback of frame replenishment techniques is associated with rapid motion and/or intensity variation occurring on the image planes. Under these circumstances, frame replenishment will suffer from the dirty window effect, and even buffer saturation.

In motion-compensated coding, the motion of pixels is first analyzed. Based on the previous frame and the estimated motion, the current frame is predicted. The prediction error together with motion vectors are encoded and transmitted to the receiver. Due to more accurate prediction based on a motion model, motion-compensated coding achieves higher coding efficiency compared with frame replenishment. This is conceivable because frame replenishment basically uses the intensity value of a pixel in the previous frame to predict that of the pixel in the same location in the present frame, while the prediction in motion-compensated coding uses motion trajectory. This implies that higher coding efficiency is obtained in motion compensation at the cost of higher computational complexity. This is technically feasible and economically desired since the cost of digital signal processing decreases much faster than that of transmission.

Because of the real-time requirement in video coding, only a simple 2-D translational model is used. There are mainly three types of motion analysis techniques used in motion-compensated coding. They are block matching, pel-recursion, and optical flow. By far, block matching is used most frequently. These three techniques are discussed in detail in the following three chapters.

Motion compensation is also widely utilized in other tasks of digital video sequence processing. Examples include motion-compensated interpolation, motion-compensated enhancement, motion-compensated restoration, and motion-compensated down-conversion.

10.8 EXERCISES

10-1. Explain the analogy between a stereo image sequence vs. the imaging space, and a stereo image pair vs. the spatial image sequence to which the stereo image pair belongs.

10-2. Explain why the imaging space can be considered as a unification of image frames, spatial image sequences, and temporal image sequences.

10-3. Give the definitions of the following several concepts: image, image sequence, and video. Discuss the relationship between them.

10-4. What feature causes video compression to be quite different from still image compression?

10-5. Describe the conditional replenishment technique. Why can it achieve higher coding efficiency in video coding than those techniques encoding each pixel in each frame?

10-6. Describe the frame-difference predictive coding technique. You may want to refer to Section 3.5.2.

10-7. What is the main drawback of frame replenishment?

10-8. Both the frame-difference predictive coding and motion-compensated coding are predictive codings in nature.

(a) What is the main difference between the two?

(b) Explain why motion-compensated coding is usually more efficient.

(c) What is the price paid for higher coding efficiency with motion-compensated coding?

10-9. Motion analysis is an important task encountered in both computer vision and video coding. What is the major different requirement for motion analysis in these two fields?

10-10. Work on the first 40 frames of a video sequence other than the Miss America sequence. Determine, on an average basis, what percentage of the total pixels change their gray-level values by more than 1% of the peak signal between two consecutive frames.

10-11. Similar to the experiment associated with Figure 10.5, do your own experiment to observe the dirty window effect. That is, work on two successive frames of a video sequence chosen by yourself, and only update a part of those changing pixels.

10-12. Take two frames from the Miss America sequence or from another sequence of your own choice in which a relatively large amount of motion is involved.

(a) Using the weighted linear interpolation defined in Equation 10.4, create an interpolated frame, which is located in the 1/3 of the time interval from the second frame (i.e., $I_2 = \frac{1}{3}(I_1 + I_2)$ according to Figure 10.9).

(b) Using motion-compensated interpolation, create an interpolated frame at the same position along the temporal dimension.

(c) Compare the two interpolated frames and make your comments.

REFERENCES

Aggarwal, J. K. and N. Nandhakumar, On the computation of motion from sequences of images — a review, *Proc. IEEE*, 76(8), 917-935, 1988.

Dubois, E., B. Prasada and M. S. Sabri, Image Sequence Coding, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981, chap. 3.

Haskell, B. G. and J. O. Limb, Predictive Video Encoding Using Measured Subject Velocity, U.S. Patent 3,632, 865, January 1972.

Haskell, B. G., F. W. Mounts and J. C. Candy, Interframe coding of videotelephone pictures, *Proc. IEEE*, 60(7), 792-800, July 1972.

Haskell, B. G. Frame Replenishment Coding of Television, in *Image Transmission Techniques*, W. K. Pratt, Ed., Academic Press, New York, 1979, chap. 6.

Horn, B. K. P. and B. G. Schunck, Determining optical flow, *Artificial Intelligence*, 17, 185-203, 1981.

Horn, B. K. P. and E. J. Weldon, Jr., Direct methods for recovering motion, *Int. J. Comput. Vision*, 2, 51-76, 1988.

Huang, T. S. Ed., *Image Sequence Analysis*, Springer-Verlag, New York, 1981.

Huang, T. S. and R. Y. Tsai, Image Sequence Analysis: Motion Estimation, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981.

Huang, T. S. and Y. P. Hsu, Image Sequence Enhancement, in *Image Sequence Analysis*, T. S. Huang, Ed., Springer-Verlag, New York, 1981.

Huang, T. S., Ed., *Image Sequence Processing and Dynamic Scene Analysis*, Springer-Verlag, New York, 1983.

Jain, A. K. *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

Kretzmer, E. R. Statistics of television signal, *Bell Syst. Tech. J.*, 31(4), 751-763, 1952.

Kunt, M., Ed., Special Issue on Digital Television. Part I: Technologies, *Proc. IEEE*, 83(6), 1995.

Lim, J. S. *Two-Dimensional Signal and Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Mounts, F. W. A video encoding system with conditional picture-element replenishment, *Bell Syst. Tech. J.*, 48(7), 2545-1554, 1969.

Musmann, H. G., P. Pirsch, and H. J. Grallert, Advances in picture coding, *Proc. IEEE*, 73(4), 523-548, 1985.

Negahdaripour, S. and B. K. P. Horn, Direct passive navigation, *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-9(1), 168-176, 1987.

Netravali, A. N. and J. D. Robbins, Motion-compensated television coding: Part I, *Bell Syst. Tech. J.*, 58(3), 631-670, 1979.

Pratt, W. K., Ed., *Image Transmission Techniques*, Academic Press, New York, 1979.

Rocca, F. Television bandwidth compression utilizing frame-to-frame correlation and movement compensation, *Symposium on Picture Bandwidth Compression*, Gordon and Breach, Newark, NJ, 1972.

Seyler, A. J. The coding of visual signals to reduce channel-capacity requirements, *IEEE Monogr.* 533E, July 1962.

Seyler, A. J. Probability distributions of television frame difference, *Proc. IREE (Australia)*, 26, 335, 1965.
Singh, A. *Optical Flow Computation: A Unified Perspective*, IEEE Press, New York, 1991.
Shi, Y. Q., C. Q. Shu, and J. N. Pan, Unified optical flow field approach to motion analysis from a sequence of stereo images, *Pattern Recognition*, 27(12), 1577-1590, 1994.
Shi, Y. Q. Editorial introduction to special issue on image sequence processing, *Int. J. Imaging Syst. Technol.*, 9(4), 189-191, 1998.
Shu, C. Q. and Y. Q. Shi, On unified optical flow field, *Pattern Recognition*, 24(6), 579-586, 1991.
Shu, C. Q. and Y. Q. Shi, Direct recovering of Nth order surface structure using unified optical flow field, *Pattern Recognition*, 26(8), 1137-1148, 1993.
Tekalp, A. M. *Digital Video Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
Thompson, W. B. Introduction to special issue on visual motion, *IEEE Trans. Pattern Anal. Machine Intell.*, 11(5), 449-450, 1989.
Tom, B. C., M. G. Kang, M. C. Hong and A. K. Katsaggelos, Detection and removal of anomalies in digitized animation film, *Int. J. Imaging Syst. Technol.*, 9(4), 283-293, 1998.
Vetro, A. and H. Sun, Frequency domain down-conversion of HDTV using an optimal motion compensation scheme, *Int. J. Imaging Syst. Technol.*, 9(4), 274-282, 1998.
Waxman, A. M. and J. H. Duncan, Binocular image flow: steps towards stereo-motion fusion, *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8(6), 715-729, 1986.
Westwater, R. and B. Furht, *Real-Time Video Compression*, Kluwer Academic, Norwall, MA, 1997.
Zhang, Y.-Q., W. Li, and M. L. Liou, Ed., Special Issue on Advances in Image and Video Compression, *Proc. IEEE*, 83(2), 1995.