DIGITAL SIGNAL COMPRESSION

Principles and Practice

William A. Pearlman and Amir Said

CAMBRIDGE

CAMBRIDGE

more information - www.cambridge.org/9780521899826

Digital Signal Compression

With clear and easy-to-understand explanations, this book covers the fundamental concepts and coding methods of signal compression, while still retaining technical depth and rigor. It contains a wealth of illustrations, step-by-step descriptions of algorithms, examples, and practice problems, which make it an ideal textbook for senior undergraduate and graduate students, as well as a useful self-study tool for researchers and professionals.

Principles of lossless compression are covered, as are various entropy coding techniques, including Huffman coding, arithmetic coding, run-length coding, and Lempel–Ziv coding. Scalar and vector quantization, and their use in various lossy compression systems, are thoroughly explained, and a full chapter is devoted to mathematical transformations, including the Karhunen–Loeve transform, discrete cosine transform (DCT), and wavelet transforms. Rate control is covered in detail, with several supporting algorithms to show how to achieve it. State-of-the-art transform and subband/wavelet image and video coding systems are explained, including JPEG2000, SPIHT, SBHP, EZBC, and H.264/AVC. Also, unique to this book is a chapter on set partition coding that sheds new light on SPIHT, SPECK, EZW, and related methods.

William A. Pearlman is a Professor Emeritus in the Electrical, Computer, and Systems Engineering Department at the Rensselear Polytechnic Institute (RPI), where he has been a faculty member since 1979. He has more than 35 years of experience in teaching and researching in the fields of information theory, data compression, digital signal processing, and digital communications theory, and he has written about 250 published works in these fields. He is a Fellow of the IEEE and the SPIE, and he is the co-inventor of two celebrated image compression algorithms: SPIHT and SPECK.

Amir Said is currently a Master Researcher at Hewlett-Packard Laboratories, where he has worked since 1998. His research interests include multimedia communications, coding and information theory, image and video compression, signal processing, and optimization, and he has more than 100 publications and 20 patents in these fields. He is co-inventor with Dr. Pearlman of the SPIHT image compression algorithm and co-recipient, also with Dr. Pearlman, of two Best Paper Awards, one from the IEEE Circuits and Systems Society and the other from the IEEE Signal Processing Society.

Digital Signal Compression

Principles and Practice

WILLIAM A. PEARLMAN
Rensselear Polytechnic Institute, New York

AMIR SAID
Hewlett-Packard Laboratories, Palo Alto, California

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Tokyo, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521899826

First published 2011

Printed in the United Kingdom at the University Press, Cambridge

A catalog record for this publication is available from the British Library

Library of Congress Cataloging in Publication data
Pearlman, William A. (William Abraham)
Digital signal compression : principles and practice / William A. Pearlman, Amir Said.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-521-89982-6 (hardback)
1. Data compression (Telecommunication) 2. Signal processing – Digital techniques.
I. Said, Amir. II. Title.
TK5102.92.P43 2011
005.74′6–dc23

2011012972

ISBN 978-0-521-89982-6 Hardback

Additional resources for this publication at www.cambridge.org/9780521899826

To Eleanor
To Cello and Ricardo

Contents

Preface

This book is an outgrowth of a graduate level course taught for several years at Rensselaer Polytechnic Institute (RPI). When the course started in the early 1990s, there were only two textbooks available that taught signal compression, Jayant and Noll[1] and Gersho and Gray.[2] Certainly these are excellent textbooks and valuable references, but they did not teach some material considered to be necessary at that time, so the textbooks were supplemented with handwritten notes where needed. Eventually, these notes grew to many pages, as the reliance on published textbooks diminished. The lecture notes remained the primary source even after the publication of the excellent book by Sayood,[3] which served as a supplement and a source of some problems. While the Sayood book was up to date, well written, and authoritative, it was written to be accessible to undergraduate students, so lacked the depth suitable for graduate students wanting to do research or practice in the field. The book at hand teaches the fundamental ideas of signal compression at a level that both graduate students and advanced undergraduate students can approach with confidence and understanding. The book is also intended to be a useful resource to the practicing engineer or computer scientist in the field. For that purpose and also to aid understanding, the 40 algorithms listed under *Algorithms* in the Index are not only fully explained in the text, but also are set out step-by-step in special algorithm format environments.

This book contains far more material than can be taught in a course of one semester. As it was being written, certain subjects came to light that begged for embellishment and others arose that were needed to keep pace with developments in the field. One example of this additional material, which does not appear in any other textbook, is Chapter 14 on "Distributed source coding," a subject which has received considerable attention lately. The intent was to present the fundamental ideas there, so that the student can understand and put into practice the various methods being proposed that use this technique.

The two longest chapters in the book are Chapters 10 and 11, entitled "Set partition coding" and "Subband/wavelet coding systems," respectively. They were actually the first chapters written and were published previously as a monograph in two parts.[4] The versions appearing here are updated with some minor errors corrected. Being the inventors of SPIHT and proponents and pioneers of set partition coding, we felt that its fundamental principles were not expounded in the technical literature. Considering the great interest in SPIHT, as evidenced by the thousands of inquiries received by us over the years since its origin in 1995 (at this writing 94,200 hits on Google), we

were eager to publish a true tutorial on the fundamental concepts of this algorithm. We believe that Chapter 10 fulfills this intent. Other books usually present only the SPIHT algorithm, almost always by working an example without revealing the underlying principles. Chapter 11 describes more image wavelet coding systems than any other book, including the JPEG2000 standard, fully scalable SPIHT, SBHP, and EZBC. The last three are set partition coders, while JPEG2000 contains auxiliary algorithms that use set partitioning. Furthermore, this chapter explains how to embed features of scalability and random access in coded bitstreams.

Besides distributed source coding, some preliminary material are also firsts in this book. They are: analysis of null-zone quantization, rate allocation algorithms, and the link between filters and wavelets. The aforementioned link is explained in Chapter 7 on "Mathematical transformations," in a way that requires only some knowledge of discrete-time Fourier transforms and linear system analysis. The treatment avoids the concepts of functional analysis and the use of polyphase transforms with little compromise of rigor. The intent was to make the book accessible to advanced undergraduates, who would likely not have exposure to these subjects. Also to serve this purpose, prior exposure to information theory is not a prerequisite, as the book teaches the relevant aspects needed to grasp the essential ideas.

One criticism that might be levied at this book is its emphasis on compression of images. Certainly, that reflects the main field of research and practice of the authors. However, image compression is possible only by understanding and applying the principles of compression that pertain to all source data. In fact, the material of the first eight chapters is generic and dimension independent. The notation is one-dimensional for the most part, and the generalization to higher dimensions is mostly obvious and hence unnecessary. Although the applications are mostly to images in the remainder of the book, except for the generic Chapter 14, the corresponding one-dimensional signal methods are mentioned when applicable and even included in some problems. The standards and state-of-the-art systems of image compression are treated in some detail, as they represent the most straightforward application of basic principles. The standard speech and audio coding systems require additional complications of perceptual and excitation models, and echo cancellation. Their inclusion would make the book too long and cumbersome and not add much to its primary objective. Nevertheless, the material on image compression systems in Chapters 9, 11, and 12 is comprehensive enough to meet the secondary objective of serving as a good tutorial and reference for image compression researchers and practitioners.

Chapter 12 treats the subject of lossless image compression. Lossless image compression is used only for archiving of image data. There seems to be no call for lossless compression of any sources for the purpose of multimedia communication, as the data transfer would be too slow or require too much bandwidth or both. For example, there is no compression of audio or speech in the WAV storage format for compact discs. MP3 is a compressed format for audio and speech transmission and recording; the compressed format of JPEG is standard in every digital camera for consumer use; all digital video is compressed by MPEG or ITU standard formats. Images seem to be the only sources

that are subject to lossless compression. The standard methods described in Chapter 12 serve as good examples of how basic principles are put into practice.

The book did not seem complete without a chapter on how compression is applied to three-dimensional sources, such as color images, volume medical images, and video. At the time of writing, there are no other textbooks that teach three-dimensional wavelet coding methods. Therefore, we wrote Chapter 13 with the intent to show the reader how the methods of the earlier chapters are extended to these higher dimensional sources. We purposely omitted detailed descriptions of video standards. We just explained the general framework in these systems upon which compression is applied and a little about the compression methods, which are mostly covered in detail in earlier chapters.

We urge potential buyers or browsers to read Chapters 1 and 2, which discuss the motivation to learn signal compression and take a brief tour through the book. This book turned out to be different in many respects from what was taught in the RPI course. Roughly, the coverage of that course was all of Chapters 3, 4, 5, and 6, which was deemed essential material. One can be selective in Chapter 7, for example, skipping the lapped orthogonal transform and some parts of Section 7.7, "Subband transforms," especially the detailed development of the connection between wavelet theory and FIR filters. Likewise, in Chaper 8, some of the rate allocation algorithms may be skipped, as well as the detailed derivations of optimal rate allocation and coding gain. One can skim through Section 9.3 in the next chapter, and skip Section 9.4 on H.264/AVC intra coding, which did not exist when the course was last taught. Time may not allow anything in Chapter 10, other than set partitioning for SPIHT and the accompanying coding example, and in Chapter 11 only a sketch of the JPEG2000 coding system. Lossless image compression in Chapter 12 could be covered earlier in the course, perhaps after Chapter 4, "Entropy coding techniques." Certainly, there is enough material to accommodate different choices of emphasis and objective.

For students with a background in information theory and signal processing or those more interested in computer science or actual implementation, an instructor may skip some of the preliminary material in the early chapters and teach all of the rate allocation algorithms in Chapter 8 and cover Chapters 10 and 11 in their entirety. Chapter 11 contains much practical material on implementation of coding systems. In fact, we think that approach would be appropriate for a short course.

The book contains many figures and problems. The problems in many cases have to be worked using a calculation and plotting program, such as MATLAB, and sometimes by making a computer program. Some datasets and basic software in C or C++ and MATLAB m-files, will be made freely available on the course website: http://www.cambridge.org/pearlman. Also freely available on the website are Powerpoint animations of the SPIHT and SPECK algorithms. Figures and problem solutions will be made available to instructors.

Notes

1. N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice-Hall 1984.
2. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers 1992.
3. K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers 1996, 2000, 2006.
4. Look under **Books** in http://www.cipr.rpi.edu/∼pearlman/.

Acknowledgments

I wish to acknowledge my debt and gratitude over the years to all my students from whom I have learned so much. Without the experience of interacting with them, I could not have written this book. There is not enough space to list all the contributors and their contributions, as I would like. Instead, I shall mention only those who were directly involved in the content of this book. I wish to thank Sungdae Cho, who created most of the video and three-dimensional SPIHT software and who encouraged me to write this book. He also created the SPIHT animation that is available on the book's website. I am grateful to Ying Liu, who developed the higher dimensional SBHP algorithms and graciously helped me with my programming difficulties. My gratitude goes out to Asad Islam, who developed the SPECK algorithm, and Xiaoli Tang, who extended it to higher dimensions. I also wish to thank Matthias Narroschke, who created the SPECK animation while I was visiting the University of Hannover, and Emmanuel Christophe, who developed resolution-scalable SPIHT, while he was a visiting student to RPI from the University of Toulouse and TeSA (Telecom Spatiales Aeronautiques). I also wish to thank Professor James Fowler of Mississippi State University for checking my explanation of bisection in two-dimensional block partitioning. I am grateful to Alessandro Dutra, who helped me write the solutions manual for the book's problems.

A special acknowledgment goes to Amir Said, my co-author and long-time collaborator, who, while a doctoral student at RPI working under Professor John Anderson, came to me to do a special project on image compression. From this project, by virtue of Amir's intellectual brilliance, creativity, and prodigious programming skills, the SPIHT algorithm was born. There are no words that adequately express my thanks to him.

Lastly, I wish to thank my wife, Eleanor, who suffered from having an often distracted and inattentive husband during more than three years while I was writing this book. This book is dedicated to her, as I could not manage without her love and support.

William A. Pearlman

1 Motivation

1.1 The importance of compression

It is easy to recognize the importance of data compression technology by observing the way it already pervades our daily lives. For instance, we currently have more than a billion users [1] of digital cameras that employ JPEG image compression, and a comparable number of users of portable audio players that use compression formats such as MP3, AAC, and WMA. Users of video cameras, DVD players, digital cable or satellite TV, hear about MPEG-2, MPEG-4, and H.264/AVC. In each case, the acronym is used to identify the type of compression. While many people do not know what exactly compression means or how it works, they have to learn some basic facts about it in order to properly use their devices, or to make purchase decisions.

Compression's usefulness is not limited to multimedia. An increasingly important fraction of the world's economy is in the transmission, storage, and processing of all types of digital information. As Negroponte [2] succinctly put it, economic value is indeed moving "from atoms to bits." While it is true that many constraints from the physical world do not affect this "digital economy," we cannot forget that, due to the huge volumes of data, there has to be a large physical infrastructure for data transmission, processing, and storage. Thus, just as in the traditional economy it is very important to consider the efficiency of transportation, space, and material usage, the efficiency in the representation of digital information also has great economic importance.

This efficiency is the subject of this book. Data compression encompasses the theory and practical techniques that are used for representing digital information in its most efficient format, as measured (mostly) by the number of bits used for storage or telecommunication (bandwidth). Our objective is to present, in an introductory text, all the important ideas required to understand how current compression methods work, and how to design new ones.

A common misconception regarding compression is that, if the costs of storage and bandwidth fall exponentially, compression should eventually cease to be useful. To see why this is not true, one should first note that some assumptions about costs are not universal. For example, while costs of digital storage can indeed fall exponentially, wireless telecommunications costs are constrained by the fact that shared radio spectrum is a resource that is definitely limited, land line communications may need large new investments, etc. Second, it misses the fact that the value of compression is unevenly

distributed according to applications, type of user, and in time. For instance, compression is commonly essential to enable the early adoption of new technologies – like it was for digital photography – which would initially be too expensive without it. After a while, it becomes less important, but since the infrastructure to use it is already in place, there is little incentive to stop using it. Furthermore, there is the aspect of relative cost. While even cheap digital cameras may already have more memory than will ever be needed by their owners, photo storage is still a very important part of the operational costs for companies that store the photos and videos of millions of users. Finally, it also ignores the fact that the costs for generating new data, in large amounts, can also decrease exponentially, and we are just beginning to observe an explosion not only in the volume of data, but also in the number and capabilities of the devices that create new data (cf. Section 1.4 and reference [1]).

In conclusion, we expect the importance of compression to keep increasing, especially as its use moves from current types of data to new applications involving much larger amounts of information.

1.2 Data types

Before presenting the basics of the compression process, it is interesting to consider that the data to be compressed can be divided into two main classes, with different properties.

1.2.1 Symbolic information

We can use the word *text* broadly to describe data that is typically arranged in a sequence of arbitrary symbols (or *characters*), from a predefined *alphabet* or *script* (writing system) of a given size. For example, the most common system for English text is the set of 8-bit ASCII characters, which include all letters from the Latin script, plus some extra symbols. It is being replaced by 16-bit UNICODE characters, which include all the important scripts currently used, and a larger set of special symbols.

Normally we cannot exploit the numerical value of the digital representation of symbolic information, since the identification of the symbols, and ultimately their meaning, depend on the convention being used, and the character's context. This means that the compression of symbolic data normally has to be *lossless*, i.e., the information that is recovered after decompression has to be identical to the original. This is usually what is referred to as *text compression* or *data compression.*

Even when all information must be preserved, savings can be obtained by removing a form of *redundancy* from the representation. Normally, we do not refer to compression as the simplest choices of more economical representations, such as converting text that is known to be English from 16-bit UNICODE to 8-bit ASCII. Instead, we refer to compression as the techniques that exploit the fact that some symbols, or sequences

of symbols, are much more commonly used than others. As we explain later, it is more efficient to use a smaller number of bits to represent the most common characters (which necessarily requires using a larger number for the less common). So, in its simplest definition, lossless compression is equivalent simply to reducing "wasted space." Of course, for creating effective compression systems, we need a rigorous mathematical framework to define what exactly "wasted" means. This is the subject of *information theory* that is covered from Chapter 2.

1.2.2 Numerical information

The second important type of data corresponds to information obtained by measuring some physical quantity. For instance, audio data is created by measuring sound intensity in time; images are formed by measuring light intensity in a rectangular area; etc. For convenience, physical quantities are commonly considered to be real numbers, i.e., with infinite precision. However, since any actual measurement has limited precision and accuracy, it is natural to consider how much of the measurement data needs to be preserved. In this case, compression savings are obtained not only by eliminating redundancy, but also by removing data that we know are *irrelevant* to our application. For instance, if we want to record a person's body temperature, it is clear that we only need to save data in a quite limited range, and up to a certain precision. This type of compression is called *lossy* because some information—the part deemed irrelevant—is discarded, and afterwards the redundancy of the remaining data is reduced.

Even though just the process of discarding information is not by itself compression, it is such a fundamental component that we traditionally call this combination *lossy compression.* Methods that integrate the two steps are much more efficient in keeping the essential information than those that do not. For instance, the reader may already have observed that popular lossy media compression methods such as the JPEG image compression, or MP3 audio compression, can achieve one or two orders of magnitude in size reduction, with very little loss in perceptual quality.

It is also interesting to note that measurement data commonly produce relatively much larger amounts of data than text. For instance, hundreds of text words can be represented with the number of bits required to record speech with the single word "hello." This happens because text is a very economical representation of spoken words, but it excludes many other types of information. From recorded speech, on the other hand, we can identify not only the spoken words, but possibly a great deal more, such as the person's sex, age, mood, accents, and even very reliably identify the person speaking (e.g., someone can say "this is definitely my husband's voice!").

Similarly, a single X-ray image can use more data than that required to store the name, address, medical history, and other textual information of hundreds of patients. Thus, even though lossless compression of text is also important, in this book the emphasis is on compression of numerical information, since it commonly needs to be represented with a much larger number of bits.

1.3 Basic compression process

In practical applications we frequently have a mixture of data types that have to be compressed together. For example, in video compression we have to process a sequence of images together with their corresponding (synchronized) multichannel audio components. However, when starting the study of compression, it is much better to consider separately each component, in order to properly understand and exploit its particular properties. This approach is also used in practice (video standards do compress image and audio independently), and it is commonly easy to extend the basic approach and models to more complex situations.

A convenient model for beginning the study of compression is shown in Figure 1.1: we have a *data source* that generates a sequence of data elements $\{x_1, x_2, \ldots\}$, where all x_i are of the same type and each belongs to a set $\mathcal{A}$. For example, for compressing ASCII text, we can have $\mathcal{A} = \{a, b, c, \ldots, A, B, C, \ldots, 0, 1, 2, \ldots\}$. However, this representation is not always convenient, and even though this is a symbolic data type (instead of numerical), it is frequently better to use for x_i the numerical value of the byte representing the character, and have $\mathcal{A} = \{0, 1, 2, \ldots, 255\}$. For numerical data we can use as $\mathcal{A}$ intervals in the set of integer or real numbers.

The compression or *encoding* process corresponds to mapping the sequence of source symbols into the sequence $\{c_1, c_2, \ldots\}$, where each c_i belongs to a set $\mathcal{C}$ of compressed data symbols. The most common data symbols are bits ($\mathcal{C} = \{0, 1\}$) or, less frequently, bytes ($\mathcal{C} = \{0, 1, \ldots, 255\}$). The decompression or *decoding* process maps the compressed data sequence back to a sequence $\{\tilde{x}_1, \tilde{x}_2, \ldots\}$ with elements from $\mathcal{A}$. With lossless compression we always have $x_i = \tilde{x}_i$ for all i, but not necessarily with lossy compression.

There are many ways data can be organized before being encoded. Figure 1.2 shows some examples. In the case of Figure 1.2(a), groups with a fixed number of source symbols are mapped to groups with a fixed number of compressed data symbols. This approach is employed in some mathematical proofs, but is not very common. Better compression is achieved by allowing groups of different sizes. For instance, we can create a simple text compression method by using 16 bits as indexes to all text characters plus about 64,000 frequently used words (a predefined set). This corresponds to the variable-to-fixed scheme of Figure 1.2(b), where a variable number of source symbols are parsed into characters or words, and each is coded with the same number of bits. Methods for coding numerical data commonly use the method of Figure 1.2(c) where groups with a fixed number of data symbols are coded with a variable number of bits. While these fixed schemes are useful for introducing coding concepts, in practical applications it is interesting to have the maximum degree of freedom in organizing both the

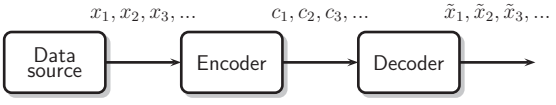

Figure 1.1 Basic data encoding and decoding model.

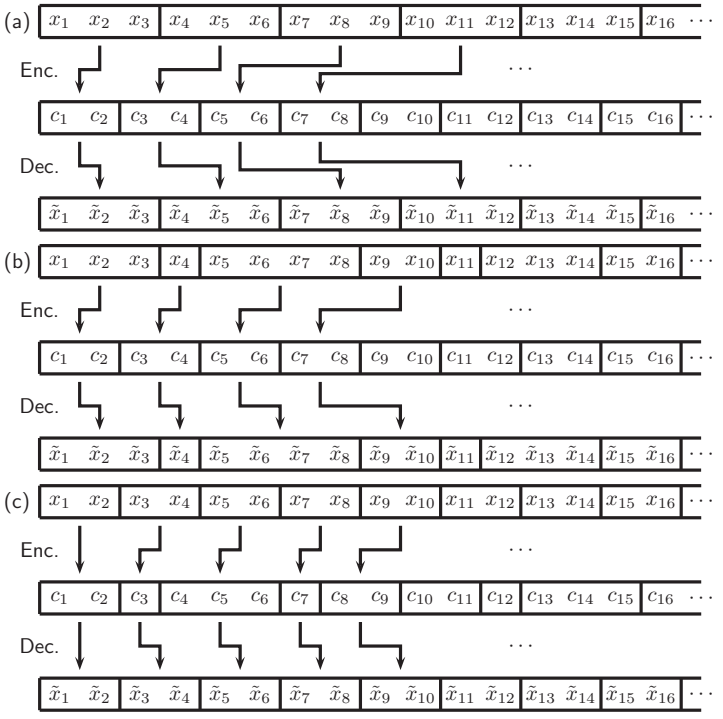

Figure 1.2 Examples of some forms in which the source and compressed data symbols can be organized for compression.

source and the compressed data symbols, so a variable-to-variable approach is more common.

1.4 Compression applications

The first popular compression applications appeared with the transition from analog to digital media formats. A great deal of research activity, and the development of the first compression standards, happened because there was a lot of analog content, such as movies, that could be converted. This is still happening, as we observe that only now is television broadcast in the USA transitioning to digital.

Another wave of applications is being created by the improvement, cost reduction, and proliferation of data-generating devices. For instance, much more digital voice traffic, photos, and even video, are being created by cell phones than with previous devices, simply because many more people carry them everywhere they go, and thus have many more opportunities to use them. Other ubiquitous personal communications and collaboration systems, such as videoconference, also have compression as a fundamental component.

The development of new sensor technologies, with the increase in resolution, precision, diversity, etc., also enables the creation of vast amounts of new digital information.

For example, in medical imaging, three-dimensional scans, which produce much more data than two-dimensional images, are becoming much more common. Similarly, better performance can be obtained by using large numbers of sensors, such as microphone or antenna arrays.

Another important trend is the explosive growth in information exchanged between devices only, instead of between devices and people. For instance, much of surveillance data is commonly stored without ever being observed by a person. In the future an even larger volume of data will be gathered to be automatically analyzed, mostly without human intervention. In fact, the deployment of sensor networks can produce previously unseen amounts of data, which may use communications resources for local processing, without necessarily being stored.

1.5 Design of compression methods

When first learning about compression, one may ask questions like:

- Why are there so many different types of compression?
- What makes a method superior in terms of performance and features?
- What compromises (if any) should be taken into account when designing a new compression method?

From the practical point of view, the subject of data compression is similar to many engineering areas, i.e., most of the basic theory had been established for many years, but research in the field is quite active because the practice is a combination of both art and science. This happens because while information theory results clearly specify optimal coding, and what are the coding performance limits, it commonly assumes the availability of reasonably accurate statistical models for the data, including model parameter values. Unfortunately, the most interesting data sources, such as text, audio, video, etc., are quite difficult to model precisely.[1]

In addition, some information theory results are asymptotic, i.e., they are obtained only when some parameter, which may be related to computational complexity or size of the data to be compressed, goes to infinity. In conclusion, while information theory is essential in providing the guidance for creating better compression algorithms, it frequently needs to be complemented with practical schemes to control computational complexity, and to create good statistical models.

It has been found that there can be great practical advantages in devising schemes that implicitly exploit our knowledge about the data being compressed. Thus, we have a large number of compression methods that were created specifically for text, executable code, speech, music, photos, tomography scans, video, etc. In all cases, assumptions were made about some typical properties of the data, i.e., the way to compress the data is in itself an implicit form of modeling the data.

Data statistical modeling is a very important stage in the design of new compression methods, since it defines what is obviously a prime design objective: reducing

the number of bits required to represent the data. However, it is certainly not the only objective. Typical additional design objectives include

- low computational complexity;
- fast search or random access to compressed data;
- reproductions scalable by quality, resolution, or bandwidth;
- error resiliency.

The need to control computational complexity is a main factor in making practical compression differ from more abstract and general methods derived from information theory. It pervades essential aspects of all the compression standards widely used, even though it is not explicitly mentioned. This also occurs throughout this book: several techniques will be presented that were motivated by the need to optimize the use of computational resources.

Depending on the type of data and compression method, it can be quite difficult to identify in compressed data some types of information that are very easily obtained in the uncompressed format. For instance, there are very efficient algorithms for searching for a given word in an uncompressed text file. These algorithms can be easily adapted if the compression method always uses the same sequence of bits to represent the letters of that word. However, advanced compression methods are not based on the simple replacement of each symbol by a fixed set of bits. We have many other techniques, such as

- data is completely rearranged for better compression;
- numerical data goes through some mathematical transformation before encoding;
- the bits assigned for each character may change depending on context and on how the encoder automatically learns about the data characteristics;
- data symbols can be represented by a fractional number of bits;
- data may be represented by pointers to other occurrences of the same (or similar) data in the data sequence itself, or in some dynamic data structures (lists, trees, stacks, etc.) created from it.

The consequence is that, in general, fast access to compressed information cannot be done in an ad hoc manner, so these capabilities can only be supported if they are planned when designing the compression method. Depending on the case, the inclusion of these data access features may degrade the compression performance or complexity, and this presents another reason for increasing the number of compression methods (or modes in a given standard).

Resiliency to errors in the compressed data can be quite important because they can have dramatic consequences, producing what is called *catastrophic error propagation.* For example, modifying a single bit in a compressed data file may cause all the subsequent bits to be decoded incorrectly. Compression methods can be designed to include techniques that facilitate error detection, and that constrains error propagation to well-defined boundaries.

1.6 Multi-disciplinary aspect

As explained above, developing and implementing compression methods involves taking many practical factors, some belonging to different disciplines, into account. In fact, one interesting aspect of practical compression is that it tends to be truly multidisciplinary. It includes concepts from coding and information theory, signal processing, computer science, and, depending on the material, specific knowledge about the data being compressed. For example, for coding image and video it is advantageous to have at least some basic knowledge about some features of the human vision, color perception, etc. The best audio coding methods exploit psychophysical properties of human hearing.

Learning about compression also covers a variety of topics, but as we show in this book, it is possible to start from basic material that is easy to understand, and progressively advance to more advanced topics, until reaching the state-of-the-art. As we show, while some topics can lead to a quite complex and advanced theory, commonly only some basic facts are needed to be used in compression.

In the next chapter we present a quick overview of the topics included in this book.

Note

1. Note that we are referring to statistical properties only, and excluding the much more complex semantic analysis.

References

1. J. F. Gantz, C. Chite, A. Manfrediz, S. Minton, D. Reinsel, W. Schliditing, and A. Torcheva, "The diverse and exploding digital universe," International Data Corporation (IDC), Framingham, MA, White paper, Mar. 2008, (http://www.emc.com/digital_universe).
2. N. Negroponte, *Being Digital*. New York, NY: Alfred A. Knopf, Inc., 1995.

2 Book overview

2.1 Entropy and lossless coding

Compression of a digital signal source is just its representation with fewer information bits than its original representation. We are excluding from compression cases when the source is trivially over-represented, such as an image with gray levels 0 to 255 written with 16 bits each when 8 bits are sufficient. The mathematical foundation of the discipline of *signal compression,* or what is more formally called *source coding,* began with the seminal paper of Claude Shannon [1, 2], entitled "A mathematical theory of communication," that established what is now called *Information Theory.* This theory sets the ultimate limits on achievable compression performance. Compression is theoretically and practically realizable even when the reconstruction of the source from the compressed representation is identical to the original. We call this kind of compression *lossless coding*. When the reconstruction is not identical to the source, we call it *lossy coding*. Shannon also introduced the discipline of *Rate-distortion Theory* [1–3], where he derived the fundamental limits in performance of lossy coding and proved that they were achievable. Lossy coding results in loss of information and hence distortion, but this distortion can be made tolerable for the given application and the loss is often necessary and unavoidable in order to satisfy transmission bandwidth and storage constraints. The payoff is that the degree of compression is often far greater than that achievable by lossless coding.

In this book, we attempt to present the principles of compression, the methods motivated by these principles, and various compression (source coding) systems that utilize these methods. We start with the theoretical foundations as laid out by Information Theory. This theory sets the framework and the language, motivates the methods of coding, provides the means to analyze these methods, and establishes ultimate bounds in their performance. Many of the theorems are presented without proof, since this book is not a primer on Information Theory, but in all cases, the consequences for compression are explained thoroughly. As befits any useful theory, the source models are simplifications of practical ones, but they point the way toward the treatment of compression of more realistic sources.

The main focus of the theoretical presentation is the definition of information entropy and its role as the smallest size in bits achievable in lossless coding of a source. The data source emits a sequence of random variables, $X_1, X_2, \ldots$, with respective probability mass functions, $q_{X_1}(x_1), q_{X_2}(x_2), \ldots$ These variables are called *letters* or *symbols* and

their range of values is called the *alphabet.* This range is assumed to be discrete and finite. Such a source is said to be *discrete.* We start with the simplest kind of discrete data source, one that emits its letters independently, each with the same probability distribution, a so-called *i.i.d.* (independently and identically distributed) source. Therefore, we consider just a single random variable or letter X that takes values $a_1, a_2, \ldots, a_K$ with probabilities $P(a_1), P(a_2), \ldots, P(a_K)$, respectively. The entropy of X is defined to be

$$H(X) = \sum_{k=1}^{K} P(a_k) \log \frac{1}{P(a_k)}, \tag{2.1}$$

where the base of the logarithm is 2 unless otherwise specified. Note that $H(X)$ depends on the probability distribution, not on the letters themselves. An important property of $H(X)$ is that

$$0 \leq H(X) \leq \log K. \tag{2.2}$$

The entropy is non-negative and is upper bounded by the base-2 logarithm of the alphabet size K. This upper bound is achieved if and only if the letters are equally probable and is the least number of bits required in a natural binary representation of the indices $1, 2, \ldots, K$. Especially important is that the entropy $H(X)$ is the fewest bits per source letter or lowest rate to which an i.i.d. source can be encoded without loss of information and can be decoded perfectly.

We then present several methods of lossless coding, most of which are provably capable of reaching the entropy limit in their compression. Therefore, such methods are referred to as *entropy coding* methods. Examples of these methods are *Huffman coding* and *arithmetic coding.* However, these methods can only achieve the lower rate limit of entropy as the length N of the data sequence tends to infinity. Their average codeword length per source symbol can reach within a tolerance band of $1/N$ or $2/N$ bits anchored at $H(X)$, so sometimes a small source length N is good enough for the application. Lossless or entropy coding is extremely important in all coding systems, as it is a component of almost every lossy coding system, as we shall see.

2.2 Quantization

Almost all physical sources are adequately modeled with a range of continuous values having a continuous probability distribution or density function. Such sources are said to be *continuous.* Discrete sources are often *quantized* samples of a continuous source. *Quantization* is the approximation of values of a source with a given smaller set of values, known at the source and destination. The values in this smaller set are indexed, so that only the indices have to be encoded. For example, rounding values to the nearest integer is quantization. Analog to digital (A/D) conversion is quantization followed by binary representation or encoding of the quantized value, as depicted in Figure 2.1. Quantization is absolutely necessary, because the number of bits needed to represent

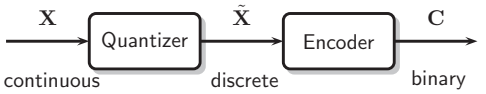

Figure 2.1 Source letter quantization followed by binary coding.

a continuous variable is infinite. That is, infinite precision requires an infinite number of bits. Clearly, infinite precision is beyond the capability of any physical device, but we wish to capture this finite precision, whatever it is, using fewer bits than its normal representation. We can think of the source as emitting either scalar (single) or vector (blocks of) random variables at each time. Whether scalar or vector, we seek to minimize the number of bits needed to encode the quantized values for a given distortion measure between the original and quantized values. The number of bits is determined either by its raw representation or its entropy when entropy coding is used. The idea is to choose the set of quantization values and the ranges of the random variables that are mapped to these values, so that the required minimum is attained. Quantizing vectors rather than scalars leads to smaller numbers of bits per source letter for a given distortion, but at the cost of larger memory to hold the quantization values and more computation to find the quantization value closest to the source value.

2.3 Source transformations

2.3.1 Prediction

As mentioned above, encoding blocks of source letters is more efficient than encoding the letters individually. This is true whether or not the letters are statistically independent. However, when the source letters are dependent, i.e., they have memory, one tries to remove the memory or transform the sequence mathematically to produce an independent sequence for encoding. Clearly such memory removal or transformation must be completely reversible. The motivation is that one can prove that memory removal achieves a reduction in distortion for a given bit rate for lossy coding and a reduction in bit rate in lossless coding. The ratio of distortions in lossy coding and ratio of rates in lossless coding between similar coding with and without memory removal is called *coding gain.* If the mathematical operation of memory removal or transformation is linear, then the output sequence is statistically independent only if the source is Gaussian and stationary. If non-Gaussian, the sequence is uncorrelated, which is a weaker property than independence. Nonetheless, these same linear operations are used, because coding gains are obtained even when the outputs of the memory removal are uncorrelated or approximately uncorrelated.

Various kinds of memory removal operation are used. The simplest is prediction of the current source value from past values to produce the error value, called the residual. A linear minimum mean squared error (MMSE) prediction of each member of the source sequence produces an uncorrelated residual sequence. Each member of the residual sequence is then encoded independently. Such coding is called *predictive*

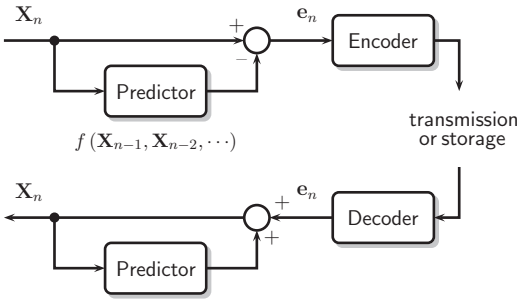

Figure 2.2 Lossless predictive coding system: top is encoder; bottom is decoder.

coding. When the encoding of the residuals is lossless, the source sequence can be reconstructed perfectly. A block diagram of a lossless predictive coding system appears in Figure 2.2. The decoder receives the current residual and is able to predict the current source value from perfect reconstructions of past source values. The current residual added to the prediction gives the exact current source value. Lossy predictive coding systems are slightly more complicated, having a quantizer directly following the residual in the encoder, and will be treated in detail in a later chapter. There, we shall study various means of prediction, and quantization and encoding of the residual sequence.

2.3.2 Transforms

2.3.2.1 Principal components and approximations

Mathematical transformations decorrelate the source sequence directly, whereas the prediction operation in predictive coding systems produces an uncorrelated residual sequence. The output of the transformation is a sequence of coefficients that represents the source sequence in a new basis. One such transformation is the Karhunen–Loève or *principal components* transformation. This transform depends on the statistical characteristics of the source, which is often unknown beforehand, so most often is used as an approximately decorrelating transform, such as the discrete cosine transform (DCT), that is not source dependent. A block diagram of such a transform coding system is shown in Figure 2.3. After transformation of the source sequence, the transform coefficients are individually quantized and encoded. In the decoder side, the coefficient codes are decoded to produce the quantized coefficients, which are then inverted by the inverse transform to yield a lossy reconstruction of the source sequence. Another property of these transforms besides uncorrelatedness is that the transform coefficients with the largest variances (energy) always appear at the smallest indices. Therefore, retaining only a few of these low index coefficients normally yields a fairly accurate reconstruction upon inverse transformation. When encoding such a transform with a target bit budget, bits are allocated unevenly, so that the highest energy coefficients get the most bits and the lowest very few or zero bits. Various bit allocation procedures have been formulated to allocate the bits optimally, so that the distortion is smallest for the given bit budget. When the bit allocation is done properly, it leads to coding gains, as shall be proved by analysis and practice.

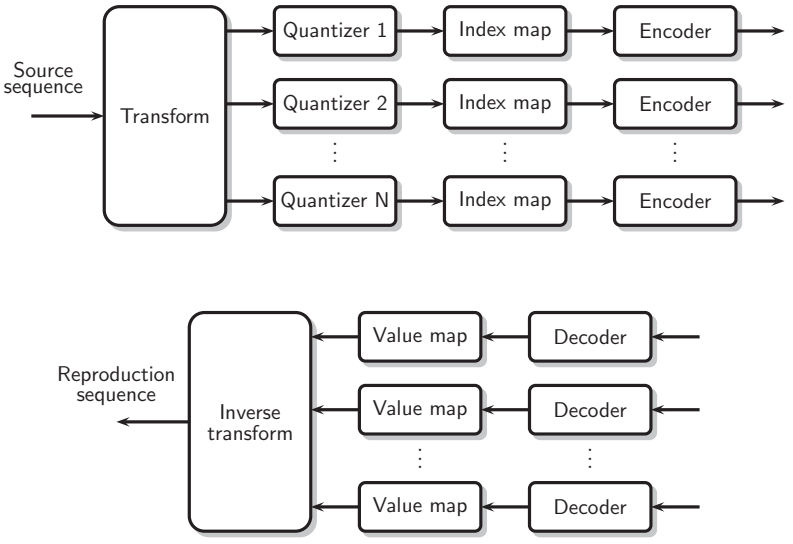

Figure 2.3 Transform coding system: encoder on top; decoder at bottom.

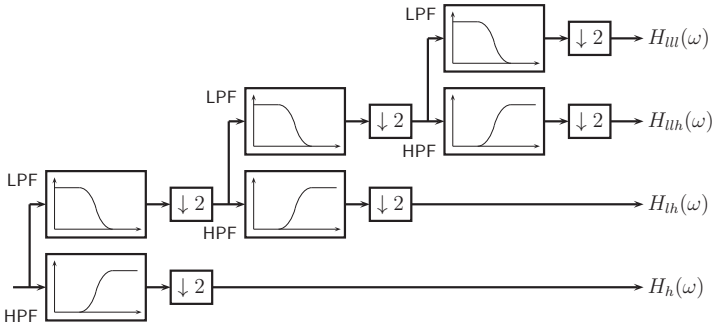

Figure 2.4 Example of how lowpass (LPF) and highpass (HPF) filters can be used for computing the subband transform.

2.3.2.2 Subband transforms

Certain mathematical transformations, called *subband transforms,* can be implemented by a series of lowpass and highpass filtering plus decimation. For example, for time sequences we can implement the subband transform with temporal filtering using a dynamic system as shown in Figure 2.4. The different outputs correspond to different responses for different frequency intervals or *subbands,* as shown in the example of Figure 2.5 (which corresponds to the filters of Figure 2.4). In images the subband transforms are done in a similar manner, but with two-dimensional spatial filtering. Commonly the filters are defined so that filtering can be done for each dimension separately, and they produce two-dimensional subbands, which are logically classified as shown in the top-left diagram of Figure 2.6, but that are actually obtained from two-dimensional filter responses as shown in the other graphs of Figure 2.6. Note that in all

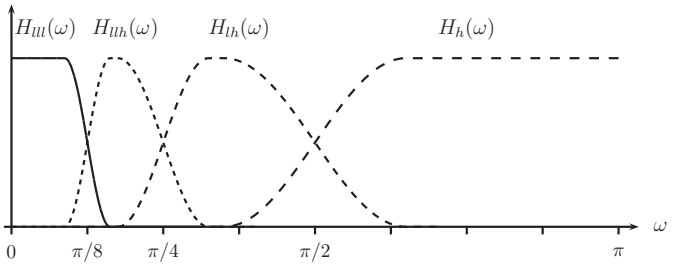

Figure 2.5 Example of a type of frequency response produced by filters used in the subband transform.

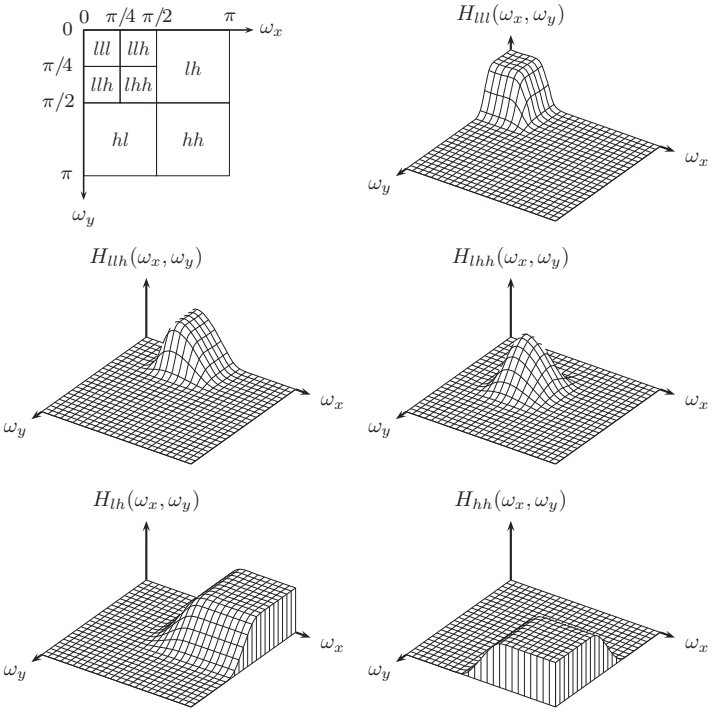

Figure 2.6 Example logical division of subbands (top left) and types of frequency response produced by some of the filters used in the two-dimensional subband transform.

cases the filter responses do not correspond to ideal filters, but nevertheless it is possible to use special sets of easy-to-implement filters that make the subband transform perfectly reversible.

When the filters are derived from a wavelet kernel, they are called *wavelet filters* and their subbands are called *wavelet subbands.* There are various kinds of wavelet filter and ways to implement them very efficiently, among them being a method called lifting, which will be explained in this book. There are filters that are reversible only if we assume floating-point computation, but there are forms of computation that are also completely reversible while maintaining a fixed, finite precision for all operations, so that there are no roundoff errors of any kind. Most of the high-magnitude or

high-energy coefficients appear in the lowest frequency subbands and the coefficients in all subbands, except the lowest one, tend to be nearly uncorrelated. The subbands are often encoded independently, since they are independent for a linearly filtered, stationary Gaussian source. In all cases, the subbands have different variances and are treated as having the same probability distribution function, except perhaps the lowest frequency subband. Therefore, as with the non-subband transforms above,[1] the target bit budget has to be unevenly allocated among the subbands to minimize the distortion in lossy coding. This kind of bit allocation results in coding gain. Lossless coding of the subbands requires no bit allocation and gives perfect reconstruction of the source when reversible, fixed-precision filters are used for the subband transform. There is even a theoretical justification for coding gains in lossless coding, as it has been proved that a subband transform reduces the entropy of the source and therefore the rate for lossless coding [4].

2.4 Set partition coding

One of the unique aspects of this book is a coherent exposition of the principles and methods of *set partition coding* in Chapter 10. This kind of coding is fundamentally lossless, but is not entropy coding in the same category as other general purpose coding methods that can be used for any type of data. It generally works best upon transforms of a source because it can more efficiently exploit some properties that are difficult to model. As explained in Section 1.3, coding methods divide the data to be coded in various ways to obtain better compression. For example, the fixed-to-variable scheme of Figure 1.2(c) is commonly used for coding quantized transform coefficients. Grouping source symbols for coding can lead to better results, but it tends to produce exponential growth in complexity with the number of symbols. Set partition coding employs a variable-to-variable approach, but one in which it adaptively subdivides the source symbols, in order to obtain simultaneously better compression and maintain low computational complexity. More specifically, set partition coding refers to methods that divide the source (or transform) sequence into sets of different amplitude ranges.

Figure 2.7 illustrates sequential partitioning of a two-dimensional block of data samples into sets of decreasing maximum value. The "dark" samples exceed in magnitude

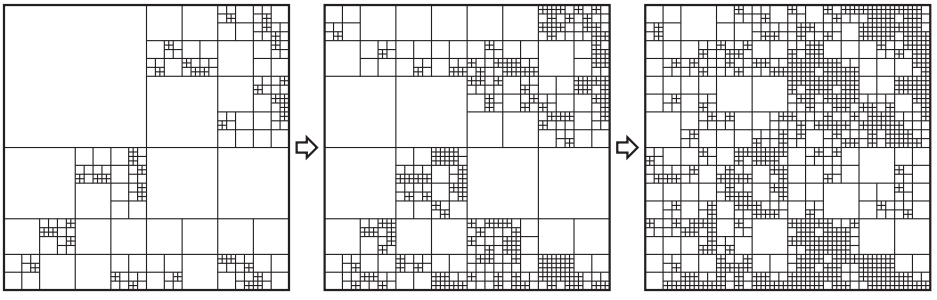

Figure 2.7 Example of how a set of image pixels is sequentially subdivided for more efficient compression using set partition coding.

a given threshold while samples in the "white" areas do not. When the threshold is lowered, every white area containing one or more above-threshold samples is recursively quadri-sected until all these above-threshold samples are located and hence darkened. The threshold is again lowered and the process repeats. Once a source or transform sample is located within a set, the most bits needed to represent it are the base-2 logarithm of the associated range. In transforms, the sets with a small range tend to be large, so their coefficients can be encoded with very few bits. All-zero sets require zero bits once they are located. The combination of the set location bits and the range bits often provides excellent compression.

Methods of set partitioning are fundamentally both simple and powerful. They require only simple mathematical and computer operations, and produce efficient compression, even without subsequent entropy coding of the range and location bits. Furthermore, these methods are natural companions for *progressive coding.* One usually defines the sets as non-overlapping with given upper and lower limits. When one searches them in order of decreasing upper limit, then the elements in ranges of larger magnitudes are encoded before those in any range of smaller magnitude. Therefore, in the decoder, the fidelity of the reconstruction increases progressively and as quickly as possible as more bits are decoded.

2.5 Coding systems

2.5.1 Performance criteria

At first glance, the performance criterion for a coding system seems obvious. One would like the system to produce the fewest bits possible to reconstruct the source either perfectly or within a given distortion. If perfect reproduction is unnecessary or unattainable, then one needs to define the measure of distortion. The most common measure is mean squared error (MSE) between the reproduction and the source. The reason is that it is easy to measure and is analytically tractable in that it allows derivation of formulas for compression performance, optimal bit allocation, and other aspects of a coding system. There are various arguments against this distortion criterion, but the chief one is that it is inappropriate for particular kinds of exploitation of the data. For example, if the source is audio, MSE is only indirectly related to the aural perception. Similarly for images, in that visual reproduction artifacts may not be adequately measured by MSE, especially if the artifacts are only locally contained. Also, one may only be interested in detection of objects or classification of vegetation in images, so the accuracy of these tasks cannot be related directly to MSE. Nonetheless, when a source is compressed for general use, MSE seems to be the best criterion. Practically all coding systems are created and proven for their performance in MSE or the related peak signal-to-noise ratio (PSNR), which is a translation of the logarithm of MSE.

Beyond ultimate performance, there are other attributes sought for a coding system. We have already mentioned progressive coding. There, decoding can stop when the user is satisfied with the fidelity of the reconstruction. A related attribute is *embedded*

coding, when the system's compressed file of any lower rate resides within its larger compressed file of larger rate. Therefore, one compressed bitstream can serve the needs of various users with different terminal capabilities. Another attribute is resolution scalability, when source reconstructions of reduced resolution can be decoded directly from the compressed bitstream of the full resolution source.

An attribute often more important even than minimum possible rate (bits per source letter) is low complexity. Low complexity manifests itself in low numbers of mathematical or computational operations, high speed in encoding and decoding, and small, compact hardware with low power requirements. One most often must sacrifice compression performance to meet the demands of lower complexity or high speed. For example, according to industry professionals, physicians will not wait more than one or two seconds to view a decompressed X-ray image. So as long as the compression is adequate, the method with the fastest decoding speed is the winner for physicians. Anyway, in creating a coding system for a particular application, the various attributes and requirements of the compression must be carefully considered.

2.5.2 Transform coding systems

2.5.2.1 JPEG image compression

The most ubiquitous coding system, the JPEG image compression standard, is a transform coding system. The source image is divided into contiguous 8×8 blocks, which are then independently transformed with the DCT. The DC or (0,0) indexed coefficients are encoded by a lossless predictive scheme and the non-DC (AC) are quantized with uniform range intervals depending on their coordinates within their block. A linear sequence of indices of the quantization intervals is formed through a zigzag scan of the block. This sequence is then encoded losslessly by a combination of run-length symbols and Huffman coding. The details may be found in Chapter 11. The JPEG baseline mode, which seems to be the only one in use, has no attributes of progressiveness, embeddedness, or scalability in resolution, but the complexity is quite low.

2.5.2.2 H.264/AVC intraframe standard

The standard video compression systems all have an intraframe or single image frame mode for the beginning frame of a group. The H.264/AVC Video Standard's intraframe mode is a transform coding method. The frame is divided into either 4×4, 4×8, or 8×8 blocks, depending on a measure of activity. For example, 8×8 blocks have the lowest activity and 4×4 the highest. The blocks are individually transformed by a simple integer approximation to the DCT. The resulting DCT blocks are encoded differently than JPEG, as intra-block prediction and context-based, adaptive arithmetic bitplane coding are employed. More details will be revealed in Chapter 9. The method has no resolution scalability, but it does have an optional limited fidelity progressive mode attained by encoding two residual layers.

2.5.3 Subband coding systems

Subband coding of images has received much attention since 1986, when Woods and O'Neil [5] published the first journal paper on the subject. However, the seminal paper in subband coding of speech by Crochiere *et al.* [6] appeared 10 years earlier. Really surprising was that the image coding results were far better than what standard analysis predicted. Specifically, standard analysis revealed that optimal predictive coding in subbands was no better than optimal predictive coding within the original image. In other words, there was supposedly no coding gain when predictive coding was used in subbands versus the same being used on the original source. A later article resolved the contradiction between theory and practice [7, 8]. It proved that coding in subbands using non-optimal finite order prediction, as what must be done in practice, does indeed show gains over coding using the same order prediction within the source. This fact is true regardless of the dimensionality of the source.

Systems employing coding of subbands seem to fall into two categories: block-based and tree-based. We shall be exploring several of them in both categories. Block-based systems include the classical systems that encode the subbands independently. The bit rates for the subbands are assigned with a bit allocation procedure based on variances. Currently popular systems assign bit rates based on the actual amplitude of coefficients in the subbands. Such block-based systems are the JPEG2000 still image standard with its embedded block coding with optimized truncation (EBCOT) coding engine, set partitioning embedded block (SPECK) and its variants subband block hierarchical partitioning (SBHP) and embedded zero block coder (EZBC), and amplitude and group partitioning (AGP). Tree-based systems encode spatial orientation trees (SOTs), or what are called erroneously *zerotrees,* in the wavelet transform.[2] These trees are rooted in the lowest frequency subband and branch successively to coefficients having the same spatial orientation in the higher subbands. A SOT of an image (two-dimensional) wavelet transform is shown in Figure 2.8. Embedded zerotree wavelet (EZW) and set partitioning in hierarchical trees (SPIHT) are the two best-known tree-based coders. Of the coders mentioned, all except JPEG2000 (and EBCOT) and EZW are set partitioning

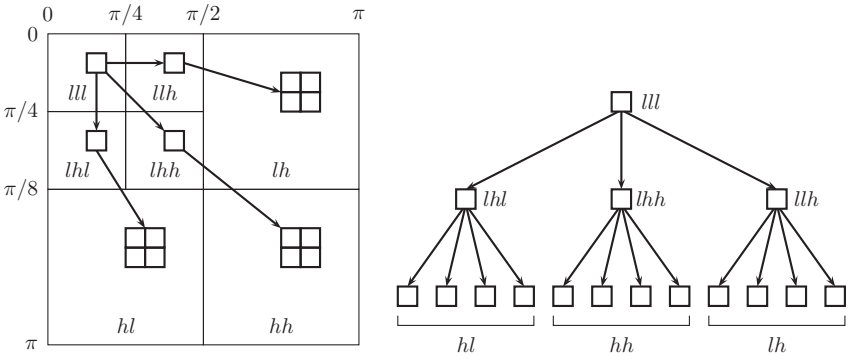

Figure 2.8 Transform coefficients in spatial frequency wavelet subbands (left) are organized to form spatial orientation trees (SOTs, right).

coders. Other set partitioning coders are group testing of wavelets (GTW), which will be be described later, and significance linked connected component analysis (SLCCA) [9]. The interested reader may consult the given reference for information about the latter method.

The coding methods just mentioned were originally developed for compressing images. Many of these methods can be extended in an obvious way to higher dimensional data sources, such as hyperspectral and tomographic data, video, and three-dimensional motion images, such as fMRI. They also can be utilized for compression of one-dimensional signals, such as audio, speech, and biological signals. Audio and speech require integration and synthesis of a perceptual model to obtain efficient compression without annoying artifacts.

2.6 Distributed source coding

Distributed source coding is a subject that has come into prominence recently. At the time of writing, there are no textbooks that treat this subject. A system employing distributed source coding encodes correlated sources independently and decodes them jointly and attains performance nearly or exactly as good as if they were encoded jointly at the source. Such a system is illustrated in Figure 2.9 for two sources.

Handheld devices, such as mobile phones, that capture and transmit video, require coding that does not consume much energy and does not cost too much to purchase. Most of the cost and complexity of video encoding are linked to motion estimation, so through distributed coding, the decoding in the base station absorbs the major brunt of the complexity, with little compromise in performance. A network of sensors, each of which transmits images or video of the same scene, cannot cooperate easily in source coding, so a processing node or base station can reconstruct the transmitted data with nearly the same accuracy as if the sensors cooperated in encoding.

For the example of a two-source system, the sum of the rate for the two sources need only be the joint entropy $H(X, Y)$ to achieve lossless recovery, as long as certain lower limits on the bit rates of the two sources are obeyed. The way the system might work is roughly as follows. One of the sources X is encoded losslessly with rate $H(X)$. The other source is encoded with rate $H(Y/X)$ ($H(X) + H(Y/X) = H(X, Y)$). The input

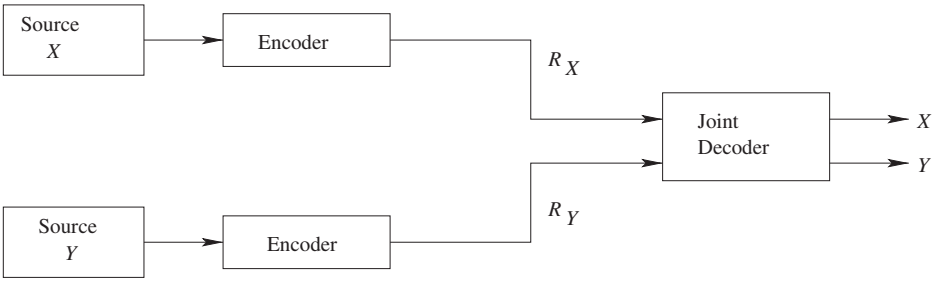

Figure 2.9 Distributed source coding: independent encoding and joint decoding of correlated sources.

space is partitioned into $H(Y/X)$ subsets (called *cosets* of a channel code) of dispersed elements. Instead of encoding Y itself, the source conveys the coset into which Y falls using $H(Y/X)$ bits. Since X is correlated with Y, it occurs with high probability close to Y. So the decoder declares Y to be the member of that coset that is closest to X. The details and justification are explained in Chapter 14.

The intention of the treatment of distributed source coding in this textbook is to explain the principles both for lossless and lossy source coding. In a practical system, one cannot avoid a small probability of recovery failure, so these probabilities are derived for some source correlation models. There is no attempt to describe the numerous application scenarios, as they lie outside the purview and objective of this book, which is to concentrate on the teaching of fundamental principles.

Notes

1. One can show that these mathematical transforms are special cases of subband transforms, but we make this distinction here, because they are usually not treated like subband transforms for encoding purposes.
2. The term *zerotree* means that all nodes in the SOT are tagged with zero to signify that they are insignificant. The term *SOT* just designates the structure of the tree without assigning values to the nodes.

References

1. C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Technol. J.*, vol. 27, pp. 379–423 and 632–656, July and Oct. 1948.
2. C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press, 1949.
3. C. E. Shannon, "Coding theorems of a discrete source with a fidelity criterion," in *Information and Decision Processes*, ed. R. E. Machol New York, NY: McGraw-Hill Publishing Co., 1960, pp. 93–126.
4. R. P. Rao and W. A. Pearlman, "On entropy of pyramid structures," *IEEE Trans. Inf. Theory*, vol. 37, no. 2, pp. 407–413, Mar. 1991.
5. J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 34, no. 5, pp. 1278–1288, Oct. 1986.
6. R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in sub-bands," in *Proceedings of the IEEE International Conference on Acoustics Speech Signal Processing*, Philadelphia, PA, Apr. 1976, pp. 233–236.
7. W. A. Pearlman, "Performance bounds for subband coding," in *Subband Image Coding*, ed. J. W. Woods Norwell, MA: Kluwer Academic Publishers, 1991, ch. 1.
8. S. Rao and W. A. Pearlman, "Analysis of linear prediction, coding, and spectral estimation from subbands," *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1160–1178, July 1996.
9. B.-B. Chai, J. Vass, and X. Zhuang, "Significance-linked connected component analysis for wavelet image coding," *IEEE Trans. Image Process.*, vol. 8, no. 6, pp. 774–784, June 1999.

Further reading

B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. IEEE*, vol. 91, no. 1, pp. 71–83, Jan. 2005.

E. S. Hong and R. E. Ladner, "Group testing for image compression," *IEEE Trans. Image Process.*, vol. 11, no. 8, pp. 901–911, Aug. 2002.

W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 11, pp. 1219–1235, Nov. 2004.

S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Process. Mag.*, pp. 51–60, Mar. 2002.

A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.

J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

D. S. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, July 2000.

3 Principles of lossless compression

3.1 Introduction

Source coding began with the initial development of information theory by Shannon in 1948 [1] and continues to this day to be influenced and stimulated by advances in this theory. Information theory sets the framework and the language, motivates the methods of coding, provides the means to analyze the methods, and establishes the ultimate bounds in performance for all methods. No study of image coding is complete without a basic knowledge and understanding of the underlying concepts in information theory.

In this chapter, we shall present several methods of lossless coding of data sources, beginning with the motivating principles and bounds on performance based on information theory. This chapter is not meant to be a primer on information theory, so theorems and propositions will be presented without proof. The reader is referred to one of the many excellent textbooks on information theory, such as Gallager [2] and Cover and Thomas [3], for a deeper treatment with proofs. The purpose here is to set the foundation and present lossless coding methods and assess their performance with respect to the theoretical optimum when possible. Hopefully, the reader will derive from this chapter both a knowledge of coding methods and an appreciation and understanding of the underlying information heory.

The notation in this chapter will indicate a scalar source on a one-dimensional field, i.e., the source values are scalars and their locations are on a one-dimensional grid, such as a regular time or space sequence. Extensions to multi-dimensional fields, such as images or video, and even to vector values, such as measurements of weather data (temperature, pressure, wind speed) at points in the atmosphere, are often obvious once the scalar, one-dimensional field case is mastered.

3.2 Lossless source coding and entropy

Values of data are not perfectly predictable and are only known once they are emitted by a source. However, we have to know something about the source statistics to characterize the compressibility of the source. Therefore, the values are modeled as random variables with a given probability distribution. In that vein, consider that an information source emits a sequence of N random variables $(X_1, X_2, \ldots, X_N)$. We group these random variables to form the random vector $\mathbf{X} = (X_1, X_2, \ldots, X_N)$. Each random variable