Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DETECTION AND ROBUSTNESS OF DIGITAL IMAGE WATERMARKING SIGNALS: A COMMUNICATION THEORY APPROACH

by
George F. Elmasry

The detection and robustness of the watermark signal is studied from a communications point of view. The contributions of this dissertation are presented in two parts. The first part, which covers the detection aspect, introduces a new digital image watermarking approach that embeds meaningful information in a copyright protection watermark signal; demonstrates the need to approach the watermark signal as a power-constrained signal; studies the relationship between the watermark signal dimension and the image capacity to the signal; explains the similarities and differences between detecting the watermark signal and detecting a signal over a spread-spectrum communication channel; and analyzes the application of sequence detection techniques (MAPSD and MLSD) to the watermark signal. The second part, which covers the robustness aspect, introduces a novel multidimensional interleaving algorithm that increases the signal's robustness against burst errors; presents, analyzes, and compares two techniques for implementing the algorithm (a sliding window technique and a successive partitioning technique); and demonstrates the increase in watermark signal robustness as a result of applying this multidimensional interleaving. This increase of the signal's robustness is shown in the 2-D case by applying the 2-D version of the interleaving algorithm to watermark signals embedded in still images (where the signal layout is in 2-D), and in the 3-D case by applying the 3-D version of the interleaving algorithm to watermark signals embedded in video sequences (where the signal layout is in 3-D).

DETECTION AND ROBUSTNESS OF DIGITAL IMAGE WATERMARKING SIGNALS: A COMMUNICATION THEORY APPROACH

by
George F. Elmasry

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Electrical and Computer Engineering

August 1999

APPROVAL PAGE

DETECTION AND ROBUSTNESS OF DIGITAL IMAGE WATERMARKING SIGNALS: A COMMUNICATION THEORY APPROACH

George F. Elmasry

Prof. Yun-Qing Shi, Dissertation Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Prof. Michael Fang, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Prof. Joseph Frank, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Prof. Constantine Manikopoulos, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Zoran Siveski, Committee Member — Date
Member of Technical Staff, Bell Laboratories, Lucent Technologies, Whippany, NJ

BIOGRAPHICAL SKETCH

Author: George F. Elmasry

Degree: Doctor of Philosophy

Date: August 1999

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 1999
- Master of Science in Electrical Engineering
  New Jersey Institute of Technology, Newark, NJ, 1993
- Bachelor of Science in Electrical Engineering
  Alexandria University, Alexandria, Egypt, 1985

Major: Electrical Engineering

Presentations and Publications:

G.F. Elmasry and Y.Q. Shi,
"MAP Symbol Decoding of Arithmetic Coding with Embedded Channel Coding,"
to be presented at the IEEE Wireless Communications and Networking Conference (WCNC'99), New Orleans, LA, September 1999.

G.F. Elmasry,
"Joint Lossless-Source and Channel Coding Using Automatic Repeat Request,"
IEEE Transactions on Communications, July 1999, Vol. 47, Issue 7, pp. 953-955.

G.F. Elmasry and Y.Q. Shi,
"Interleaving Reed-Solomon Codes for 2-D Arrays: Applications in 2-D Bar Coding,"
submitted to the IEEE Transactions on Communications, June 1999.

G.F. Elmasry,
"Embedding Channel Coding in Arithmetic Coding,"
IEE Proceedings-Communications, April 1999, Vol. 146, No. 2, pp. 73-78.

G.F. Elmasry,
"Error Resilient Arithmetic Coding for Band-limited Channels With Low SNR,"
submitted to the IEEE Journal on Selected Areas on Communications - Error Resilient Image and Video Transmission, April 1999.

B. He, G.F. Elmasry, and C. Manikopoulos,
"Joint Lossless-Source and Channel Coding Using ARQ/Go-Back-(N,M) with Packet Combining,"
submitted to the IEEE Journal on Selected Areas on Communications - Error Resilient Image and Video Transmission, April 1999.

B. He, G.F. Elmasry, C. Manikopoulos, and Y.Q. Shi,
"A Go-Back-N+1 Protocol and Its Application in Wireless ATM Networks,"
Proceedings of the Thirty-Third Annual Conference on Information Sciences and Systems, The Johns Hopkins University, March 17-19, 1999, pp. 809-813.

G.F. Elmasry and Y.Q. Shi,
"Maximum Likelihood Sequence Decoding of Digital Image Watermarking,"
Proceedings of SPIE: Security and Watermarking of Multimedia Contents, San Jose, CA, January 25-27, 1999, vol. 3657, pp. 425-436.

J. Huang, G.F. Elmasry, and Y.Q. Shi,
"Power Constrained Multiple Signaling in Digital Image Watermarking,"
Proceedings of the 1998 IEEE Second Workshop on Multimedia Signal Processing, Redondo Beach, CA, December 1998, pp. 388-393.

G.F. Elmasry and Y. Q. Shi,
"An Interleaving Technique for Error Correction in 2-D Bar Coding,"
Proceedings of the Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing,
University of Illinois at Urbana-Champaign, September 23-25, 1998, pp. 49-58.

G.F. Elmasry, J. Huang, and Y.Q. Shi,
"Capacity of Multiple Signaling in Digital Image Watermarking,"
Proceedings of the Fourth International Conference on Information Systems Analysis and Synthesis (ISAS), Orlando, FL, July 1998, pp. 155-159.

G.F. Elmasry, J. Huang, and Y.Q. Shi,
"Embedding Meaningful Information in Digital Image Watermarking,"
submitted to Image Communication, June 1998.

G.F. Elmasry,
"Joint Lossless-source and Channel Codes Using Arithmetic Coding,"
Proceedings of the Thirty-Fifth Annual Allerton Conference on Communication, Control, and Computing,
University of Illinois at Urbana-Champaign, September 29, 1997, pp.126-127.

G.F. Elmasry,
"Arithmetic Coding Algorithm with Embedded Channel Coding,"
IEE Electronics Letters, vol. 33, issue 20, September 25, 1997, pp. 1687-1688.

G.F. Elmasry,
"Joint Lossless-Source and Channel Codes Over the [0-1) Vector Space,"
Proceedings of the Thirty-First Annual Conference on Information Science and Systems, Johns Hopkins University, March 1997, pp. 319-324.

G.F. Elmasry,
"An Automatic Repeat Request Scheme for Joint Lossless-Source and Channel Codes,"
Proceedings of the Thirty-Fourth Annual Allerton Conference on Communication, Control, and Computing,
University of Illinois at Urbana-Champaign, October 2-4, 1996, pp. 102-111.

G.F. Elmasry and Y. Bar-Ness,
"An Automatic Repeat Request Scheme for Error Detection and Correction of Compressed Data,"
Proceedings of the Thirty Annual Conference on Information Science and Systems, Princeton University, March 21-23, 1996, pp. 561-566.

To my wife, Lisa Fitton, and our newborn son, Andrew Solomon Elmasry,
with all my love.

ACKNOWLEDGMENT

First and foremost, I would like to thank my advisor, Professor Yun Q. Shi, for the time he put into helping me make this dissertation the comprehensive work it is today; for introducing me to the topic of digital image watermarking, which I found very interesting; for the freedom with which he allowed me to proceed; for his input in the analysis of the multidimensional interleaving technique; for his idea of applying multidimensional interleaving to the watermark signal; for his kindness; and, most of all, for his friendship.

Of course, this work would not have been possible without the input and insights from the members of my dissertation committee. Dr. Zoran Siveski has been a guiding force during my whole graduate career–from my early days in the master's program to the present. His moral support and encouragement have always made things a little easier, and his influence on my research paradigm has been profound. Professor Constantine Manikopoulos has also left a positive mark on me. His interest in my work and belief in my ability provide a sense of comfort when the going gets tough. Professor Yuguang Fang and I joined forces relatively recently, yet his encouragement and support have already proven to be valuable factors in my academic career. Finally, my committee would not have been complete without its staple: Professor Joseph Frank. I took courses with him during my first two semesters at NJIT–courses that would prove to be more useful in my current research than I would have imagined eight years ago; he taught me some of the most important communications material I would ever learn. Back then, he saw capabilities in me that I did not recognize in myself, and later he recommended me to the doctoral program. He was always there for advisement, reassurance, a letter of reference or just a friendly word.

I would like to acknowledge Dr. Jiwu Huang, with whom I started some of the early work in this dissertation, for all his effort, and for the simulations he performed in Chapter 2.

Thanks are due also to Professor Richard Haddad, Chair of the Department of Electrical and Computer Engineering and Principal Investigator of the New Jersey Center for Wireless Telecommunications. Professor Haddad has made himself available to me in more ways than I can count–from general academic counseling to providing very specific advice about my career and future. I can not thank him enough for all his help. Likewise, I must extend appreciation to Professor Ali Akansu, Director of the New Jersey Center for Multimedia Research, whose generosity knows no bounds, and whose interest in my progress in the doctoral program has been a source of strength and empowerment. Professor Sirin Tekinay has also been a special advocate. I've always admired her cheerful and friendly demeanor, as well as her energy and drive. Associate Chair for Graduate Studies, Professor Nirwan Ansari, and Doctoral Program Coordinator, Professor Stanley Reisman, similarly hold my utmost respect and goodwill.

The staff of the ECE department, Ms. Barbara Faltz, Ms. Anne McMahon, Ms. Anna Thomas, and Ms. Brenda Walker, and Engineering Computing senior system administrator, Mr. Brian White, should be commended for their part in my graduate program. They are the ones who are always there on the sideline–who actively take part but are rarely recognized. Thank you all for your constant assistance in just about everything.

No doctoral experience would be complete without the friends and labmates that have helped shape it. I'm happy to have been associated with Mr. Mahalingam Ramkumar, Mr. Bin He, Mr. I. Burak Ozer, Ms. Sebnem Zorlu, Ms. Feihong Chen, Dr. Aydin Alatan, and others. Good luck to you all, and may our paths cross again.

Finally, words can never express the gratitude I feel toward my wife, Lisa Fitton, whose vigor and dedication paved roads where there were none; whose benevolence manifested itself at every turn along the way; and whose love provided the foundation upon which this work and other achievements were built.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter Page

TABLE OF CONTENTS
(Continued)

Chapter Page

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

In recent years, digital technology has invaded almost every aspect of our lives. The idea of an information highway is not the science fiction it was once thought to be. The future promises more availability of more information and easier access with each passing day. The more technology matures, the more its capabilities spread through every layer of society. Just about anyone can gain access to the Internet–no matter how old, how rich or poor, how skilled or not.

Anyone can access huge amounts of data in the form of text, sound, and images. From an entire congressional hearing to an artistic web site design, multimedia data may be copied and transferred to someone else with the touch of a button. When we can access and transfer data so easily, we take for granted that this data was created by an individual who spent time, effort, and possibly money to put the data into a form we can so easily copy, edit, and move from one place to another.

1.1 The Need for Image Watermarking

With every advance comes a setback. It's difficult to sit through an evening news program without hearing about some form of technology misuse. As fun and as instrumental to learning as the Internet may be, there are plenty of users who would be happy to take advantage of others. Sometimes this happens in the form of the child pornography cases we hear about so often on television. Other times, it takes the form of a more subtle crime, such as illegal duplication and distribution. Not in the line of police or detective work, it is the second case with which we are in a position to deal.

Copyright protection of images is a major concern in the multimedia industry today. Image copyright protection is sought for many technologies. Digital Video Disks (DVD's), for example, can be easily mass produced and illegal copies easily distributed. On the Internet, downloading images also opens the door for false

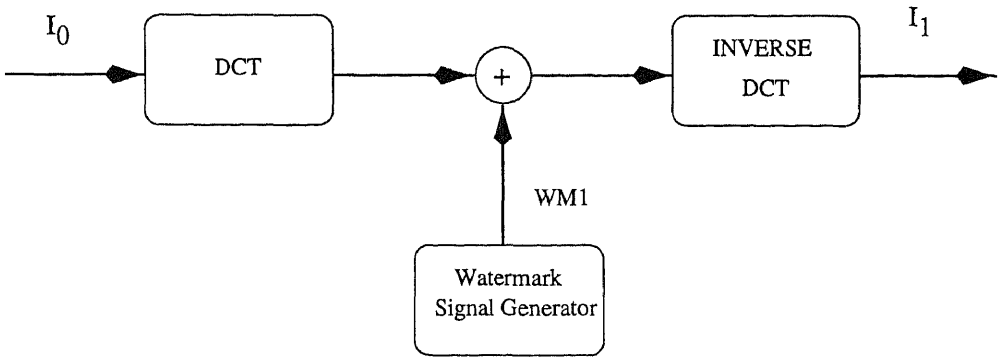
ownership claims and the production of illegal copies. Scientific images are just as likely as others to fall victim to technology misuse and, therefore, need some means of copyright protection.

As far as the entertainment industry goes, the future also holds the possibility of accessing movies over the Internet, so that viewers can watch them without having to leave the house. It is possible that someone can rent a movie (download it) and instead of watching it once, produce illegal copies of it. Data hiding will enable the owner of the movie (or the person or organization holding the copyright to the movie) to hide computer instructions that enable the renter to watch the downloaded movie once before it is deleted. A digital image watermark can carry information that not only identifies the owner, but also allows tracking–a way to identify the producer of illegal copies.

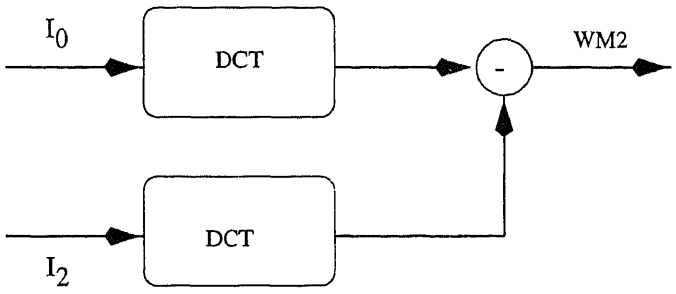
1.2 Copyright Protection and Information Hiding

Generally speaking, digital image watermarking can be divided into two main categories. The first category utilizes the watermark signal for copyright protection, while the second category utilizes the signal for other purposes, such as hiding some information in the image. Each category has its own requirements and hence its own challenges. During the course of the research that lead to this dissertation, a goal was established to develop a watermarking technique for copyright protection that simultaneously provides a means for carrying meaningful information in the signal.

In digital image watermarking, it is preferred that the watermark signal be embedded in a transform domain. When transforming the image back to the spatial domain, the change in the energy of the transform domain coefficients—where the signal is added—tends to spread among all the pixels of the transformed block, making the signal less invisible [3]. Throughout this dissertation, the watermarking encoding (embedding) and decoding (recovery) methods are based on Figure 1.1. When embedding the watermark signal, the original image I_0 is transformed to the Discrete Cosine Transform (DCT) domain. The DCT is applied to blocks of 8×8 pixels. Some l significant AC coefficients in each transformed block are selected to



(a) Embedding



(b) Recovery

Figure 1.1 Watermark Signal Embedding and Recovery Using DCT

carry the watermark signal. After adding the signal, an inverse DCT is applied and the watermarked image I_1 is obtained. Thus,

$$I_1 = I_0 + WM1, \tag{1.1}$$

where $WM1$ is the embedded watermark signal.

I_2 is a version of I_1 that went through some intentional or unintentional attacks. Intentional attacks come from someone who intends to claim the ownership of the image. Unintentional attacks occur from signal processing techniques that the image may go through (e.g., filtering and quantization). The recovery technique of the watermark signal assumes the original image I_0 (the one without the signal) is available. I_0 and I_2 are transformed to the DCT domain, and the watermark signal

is obtained by subtracting I_0 from I_2. That is,

$$WM2 = I_2 - I_0, \tag{1.2}$$

where $WM2$ is the watermark signal obtained at the decoder. $WM2$ is a noisy version of the embedded signal $WM1$.

One can see that, generally speaking, for a watermark signal designed for copyright protection, $WM2$ is expected to support the owner's right to the image by giving a clear indication of the existence of $WM1$ in I_2 even if I_2 suffers from attacks. For a watermark signal designed for hiding some information (where I_0 may not be available), $WM1$ is expected to carry as many information symbols as possible (without causing any perceptual difference between I_0 and I_1). In addition, when $WM2$ is retrieved, it is expected to reveal the embedded symbols with the least symbol decoding error possible.

1.3 Industry Requirements and Challenges

The industry requirements for image copyright, which are not straightforward, have created an area of research that is demanding and carries many problems yet to be solved. What the industry has put forth includes the following requirements [3] [4] [5]: A copyright protection digital image watermark is expected to be perceptually invisible (it should not interfere with the original image); it should be unambiguous (it should explicitly identify the owner); and it should be robust (hard to remove). The signal is expected to be robust in that it is shielded against signal processing techniques, distortion of the image itself, and intentional attacks.

There are many challenges that come with these demands. If we want to have a watermark signal that is unambiguous, and at the same time one that resists attacks, the signal itself must be hidden from someone with the intention of forging or destroying the watermark signal. In this work this challenge is addressed by asking the following question: How can we have a signal that appears to be a sequence of random variables and, at the same time, the signal can reveal (in a convincing manner) a string of meaningful information when image ownership is debated?

Another challenge is raised based on the following scenario: Suppose the original owner (A), who owns an image I_0, produced the image I_1 as explained with Equation (1.1). A fraudulent claimant (B) can take I_1, and subtract a certain signal $\overline{WM1}$ from it to produce $\overline{I_0}$. That is,

$$\overline{I_0} = I_1 - \overline{WM1}. \tag{1.3}$$

When image ownership is debated, (B) can claim that $\overline{I_0}$ is the original image before the watermark signal, I_1 (the image under debate) is the image with the watermark signal, and $\overline{WM1}$ is his watermark signal (signature). The fraudulent claimant (B) can show how his signature can be obtained from the following equation:

$$\overline{WM1} = I_1 - \overline{I_0}. \tag{1.4}$$

Note that because a watermark signal is expected to introduce no visual degradation to the image, $\overline{I_0}$ can be accepted perceptually. Research in digital image watermarking [5] showed that a solution to this attack is to make the watermark signal image dependent. Techniques that create image-dependent watermark signals attempt to make the correlation between $WM1$ and $\overline{I_0}$ higher than the correlation between $\overline{WM1}$ and I_0. This higher correlation would indicate the existence of $WM1$ in $\overline{I_0}$ and reduce the likelihood of the existence of $\overline{WM1}$ in I_0. Although this work will not cover how to generate an image-dependent watermark signal, Chapter 2 will show how to turn a watermark signal—that can reveal a string of meaningful information—into an image-dependent signal.

Another challenge proceeds as follows: We want the watermark signal to be invisible, and at the same time, the signal is expected to be robust against all possible attacks. To make the watermark signal invisible, the amount of change that can be introduced to the image to carry the watermark signal is limited. Consider transforming the image to some transform domain (e.g., DCT) to embed the watermark signal. For the DCT coefficients to carry a watermark signal that will be invisible in the spatial domain, only a limited amount of change to these coefficients is possible. This limitation of change translates to a limitation of watermark signal power; that is, the watermark signal is a power-constrained signal. The issue of achieving high

robustness is addressed by asking the following question: What is the best approach to embed the watermark signal with respect to the image, such that the signal can be detected if the image (or the signal) is partially damaged? That is, if an attacker or some signal processing technique or damage to the image itself results in a loss of part of the signal (e.g., half the signal is totally lost), we want to detect a signal that can, in an unambiguous way, reveal sufficient meaningful information when image ownership is debated.

1.4 Is It A Spread Spectrum Signal?

Although much of the available literature refers to the watermark signal as a spread-spectrum communication-channel signal [3], the watermark signal is not exactly the same as a spread-spectrum signal. In this section, some of the similarities and differences between watermark signals and spread-spectrum signals are addressed.

1.4.1 Similarities

Similar to transmitting a spread-spectrum signal over a communication channel (where the amount of information that can be transmitted is bounded by the channel capacity), the amount of information that can be embedded in the watermark signal is bounded by the image capacity. When studying the capacity of the image with respect to the watermark signal, generally speaking, the amount of information that can be embedded in an image depends on many factors. The size of the image is certainly a factor, since for large images we can embed more information than for small images with the same degradation level. The second factor is the domain in which the information is embedded. In a transform domain, we can embed the information bits in selected coefficients, making the watermarking signal more spread out and hence less noticeable in the spatial domain [3].

The work performed here embeds the watermark signal in images of size 256×256 pixels. With this method, DCT is applied to blocks of 8×8 pixels (as specified in the international image and video coding standards), and the l most significant coefficients (excluding the DC coefficient) in each transformed block are selected to

carry the watermarking signal. These components have a relatively larger capacity, to tolerate the watermark signal without perceptual degradation, than other AC coefficients [3] [4] [14]. With the approach used in this work, when we selected $l = 3$ [14], we worked with a total of 3072 coefficients ($\frac{256}{8} \times \frac{256}{8} \times 3$). Regardless of the image size, the transform domain selected for embedding the watermark signal (e.g., DCT or FT), or the number of coefficients per transformed block (l) carrying the signal, the need to make the watermark signal perceptually invisible results in a signal that is power constrained.

In this work, we show how the number of coefficients utilized by the watermark signal resembles the available bandwidth for a power constrained signal transmitted over a spread-spectrum communication channel. As with power-constrained signals, the tradeoff between increasing the signal dimension (which could mean better utilization of available capacity) and lower computational complexity is encountered.

Because the watermark signal is power constrained and it should appear as a sequence of random variables to those attempting an attack, there is a striking resemblance between detecting the watermark signal and detecting a signal over a spread spectrum communication channel [34]. The similarities between the watermark signal and a spread-spectrum signal can be summarized as follows:

- Both signals are power constrained.
- Both signals are a sequence of random variables.
- Both can have a tradeoff between the achieved capacity and the signal dimension.

1.4.2 Differences

The resemblance between the detection of the watermark signal and detecting a signal over a spread-spectrum channel, does not mean that the watermark signal is identical to a spread-spectrum signal. In this work we are interested not only in the similarities, but also in the differences between the two signals. Let us observe a fundamental and very important difference between the two.

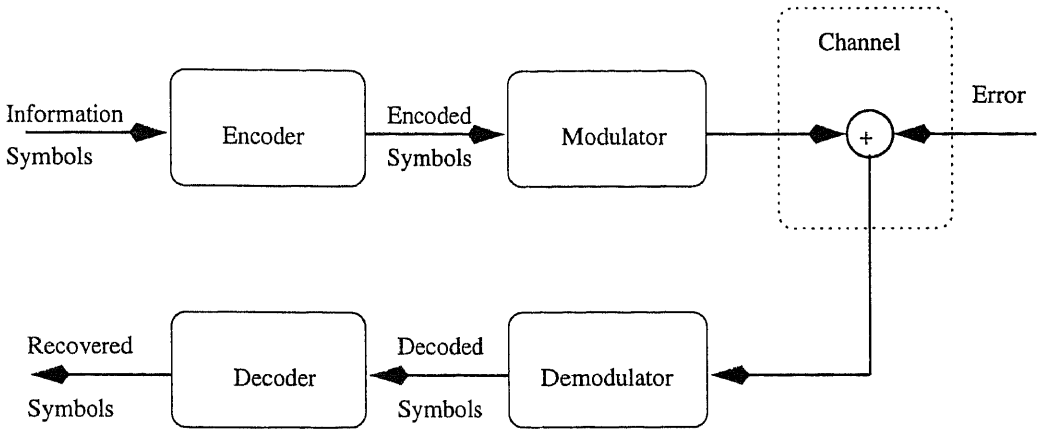


Figure 1.2 Modeling the Signaling over a Typical Communication Channel

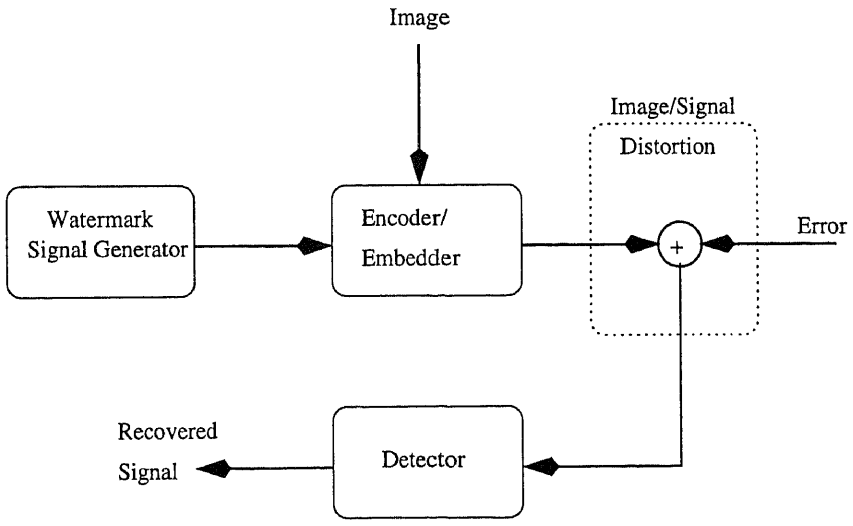


Figure 1.3 Modeling the Watermark Signaling Process

Figure 1.2 shows how a communication channel is modeled [29], while Figure 1.3 shows how the digital image watermark process is modeled. Over a communication channel, the demodulator is expected to supply the decoder with quantized values of the received signal [40]. Communications references are rich in explaining the difference between hard- and soft-decision decoding [21] [37] [39]. Researchers in the communications field are aware of the fact that if we can reach a demodulator that supplies the decoder with unquantized values of the received signal, optimum detection using sequence decoding is achievable [29]. Unfortunately, this ideal demodulator does not exist with communication channels. In this work, however, we will explain how this ideal situation is available with watermark signals. We will show how unquantized values of the detected signal are obtainable and hence show how sequence decoding is possible. The use of sequence decoding makes the increase of computational complexity manageable as the signal dimension increases. This opens the door to achieving higher capacity.

Another major difference between an image watermark signal and a communication channel signal is the signal layout. While the layout of the spread-spectrum signal normally is in 1-D (the signal as a function of time), the layout of the watermark signal is multi-dimensional. When embedding the watermark signal in a still image, the signal layout is in 2-D (x and y coordinates of each coefficient). When embedding the watermark signal in a video sequence, the layout of the signal is in 3-D (x and y coordinates decide the location in a frame, and the z coordinate decides the specific frame in the sequence).

The differences between the watermark signal and a spread-spectrum signal can be summarized as follows:

- No demodulator is used with the watermark signal.
- The watermark signal layout is multi-dimensional.

As the reader proceeds with this dissertation, it will be clear how these two major differences between a watermark signal and a communication-channel signal are utilized to improve both the detection process and the achieved robustness of watermark signaling.

1.5 Detection and Robustness

This dissertation concentrates mainly on two very important aspects of the digital image watermarking process: detection and robustness. The dissertation is divided into two parts. The first part addresses the detection aspect of the signal and the second part addresses the robustness aspect.

1.5.1 Detection

With regard to the detection process, the first part is divided as follows:

- In Chapter 2, the dilemma of keeping the watermark signal a random variable sequence, and at the same time having the detection process reveal meaningful information, is addressed. We develop a multiple signaling approach that allows us to embed meaningful information in the watermark signal while keeping the signal a sequence of random variables. We proceed in this study by examining the approach with regard to Shannon's coding theory. We will compare the following three coding approaches:

  1. Multiple embedding, where a single binary value is embedded in multiple coefficients. We will show how this is equivalent to using repetition codes in channel coding.
  2. Error correction coding, where we select the error correction code BCH (31:6) for our comparison.
  3. Correlation decoding, where we used the same redundancy as with BCH (31:6) but in the form of increasing the signal dimension.

  We will show that because the watermark signal is power constrained, correlation decoding is better than error correction coding, and that multiple embedding gives the poorest performance. We will also show how Shannon's coding theory is applied to the watermark signal: As the watermark signal to noise ratio SNR_{wm} decreases, the amount of information that could be carried in the signal decreases.

- Chapter 3 will show how the number of coefficients selected to carry the watermark signal N resembles the available bandwidth in a spread-spectrum channel. As with spread-spectrum signaling, a tradeoff between the achieved capacity and the signal dimension may be encountered. We will study the behavior of the image capacity when approaching the watermark signal as a signature (just a sequence of random variables utilized for copyright protection), and when approaching the watermark signal as a means for embedding information symbols. On this point, we show that the complexity of the detection process may increase as the signal dimension increases. This prompts the need to develop a digital image watermark detection process where a manageable increase in the computational complexity occurs when the signal dimension increases.

- In Chapter 4, the similarity between detecting the digital image watermark signal and detecting a signal over a spread spectrum communication channel with additive white Gaussian noise is explained. We study the differences between the two, and show how we can use sequence decoding, which offers a manageable increase in computational complexity as the signal dimension increases. A comparison between correlation detection and sequence detection is presented. We also present a comparison between maximum likelihood sequence decoding (MLSD), maximum a-posteriori probability sequence decoding (MAPSD), and correlation sequence decoding (CORSD). We will show how MAPSD, in addition to offering a less complicated detection technique, can also offer a good measure of the confidence level of the detected signal.

1.5.2 Robustness

The second part of this dissertation approaches the robustness aspect of the digital image watermark signal from a communications angle. Much of the literature addresses this issue from signal- and image-processing points of view. These works attempt to increase the signal robustness by trying to increase signal power through

techniques such as visual masking [14] [27] [35], or by searching for other transform techniques (which may allow increasing the signal power without causing visual degradation). Although some of these techniques showed a considerable gain in the signal power, the gain is limited. The industry requirement of having the watermark signal be invisible limits the maximum amount of signal power that can be used. In our approach, the robustness dilemma is observed from a communications point of view. The digital image watermark signal has very specific characteristics that make it fundamentally different from a signal over a communication channel. As explained in the previous section, the layout of the watermark signal normally is multi-dimensional regardless of what domain the signal is embedded in. We try to find the best approach to embed the watermark signal with respect to the image, such that the signal can be detected even if it is partially damaged. The second part of the dissertation proceeds as follows:

- Chapter 5 is a tutorial: In our research, we developed an interleaving algorithm that spreads consecutive data elements (bits or symbols) in multidimensions. This interleaving technique may be considered the n-D equivalent to block interleaving (which is a type of 1-D interleaving) commonly used in communication channels. While block interleaving spreads consecutive elements across the interleaved block, the algorithm presented in this work spreads consecutive elements across n-D. As stated above, with respect to still images, the watermark signal has a 2-D layout. With respect to a video sequence, the watermark signal has a 3-D layout.

- In Chapter 6 we present and analyze an implementation method of this interleaving algorithm that uses a sliding window technique. It will be shown how this technique is adapted from the 1-D case to the 2-D case. We will also show how the algorithm can be adapted to the 3-D case. We present simulation results that show the superiority of the algorithm over known interleaving techniques when applied to applications with nonpower-constrained signals in a 2-D layout.

- In Chapter 7 a different implementation method of the multidimensional interleaving, which uses a successive partitioning technique, is presented. Here, we discuss the 3-D interleaving case in more detail and present the general interleaving case (n-D interleaving). We compare the implementation of the sliding window interleaving technique with the implementation of the successive partitioning interleaving technique.

- In Chapter 8, the 2-D version of the above interleaving algorithm is applied to digital image watermarking signals embedded in a still image. Two approaches for overcoming burst errors in digital image watermarking will be compared. The first approach utilizes interleaving, while the second approach does not utilize interleaving. It will be shown that the interleaving approach is preferred. It will be shown how interleaving a power-constrained signal differs from interleaving a nonpower-constrained signal. The increase in watermark signaling robustness against burst errors, when the signal is interleaved, is presented. Different strategies are shown in applying interleaving such that the information carried with the watermark signal can be detected even if half, or even three quarters, of the signal is totally destroyed. The concept of erasure decoding to overcome impulse noise, and an adaptation of it to overcome quantization noise, will be explained. Cases where the signal suffers from background noise in addition to the burst error (for which we interleave) will be studied. The cases of white Gaussian background noise and background noise with a uniform distribution (quantization noise) will be discussed.

- In Chapter 9 the 3-D version of this interleaving technique is applied to watermark signals embedded in a video sequence. With this interleaving technique, if an attacker destroys the watermark signal in a group of entire frames in the video sequence or destroys the watermark signal in parts of each frame, the signal still can be detected and a meaningful string of symbols can be decoded.

1.6 A Communication Theory Approach

We would like to note the following: Because digital image watermarking is an image processing/multimedia area of research, most of the literature available was developed by experts in these fields. This dissertation, however, looks at digital image watermarking exclusively from a communication theory angle. As Figure 1.2 and Figure 1.3 show, we are building this research by asking the following questions:

- What are the similarities between the digital image watermark signal and a signal over a spread spectrum communication channel?
- What are the differences between the two cases?

The answer to the first question leads us to establish the fact that the watermark signal is power constrained, and the utilized number of coefficients to carry the signal is analogous to the available bandwidth in communication channels (Chapters 2 and 3). The answer to the second question is two-fold. The first difference is that the watermark signal detection process can obtain what is equivalent to unquantized values (from the demodulator) of signals detected over a communication channel, and hence sequence decoding is optimum (Chapter 4). The second difference is that the layout of the digital image watermark signal is in 2-D or 3-D, which lead to developing a n-D interleaving technique to be utilized for increasing signal robustness (Chapters 5-7) and applying it to the watermark signal (Chapters 8 and 9).

CHAPTER 2

EMBEDDING MEANINGFUL INFORMATION IN THE WATERMARK SIGNAL

The purposes for using digital image watermarking can be divided into two main categories. The first category includes embedding some information in the image, and the second category includes copyright protection. With the first category, the image owner intends to hide some information (a sequence of symbols) in the image. The methods used with this category do not always assume that the watermark signal will suffer from intentional attacks. With copyright protection, however, it is essential for the embedded signal (the owner's signature) to be robust against intentional attacks and certain signal processing techniques. To achieve this goal, it is preferred that the watermark signal be a unique sequence of random variables.

With copyright-protection image watermarking, the detection techniques for extracting the embedded signature measure the correlation between the uniquely selected random variable sequence originally embedded into the image, and its noisy version extracted from the image. The high correlation between the original watermark signal and the noisy version is a measure of robustness. Although these methods have achieved a perceptually invisible and robust (difficult to remove) watermarking signal, they actually trade capacity for robustness and seek a *yes* or *no* answer. In other words, we can say that a 1-bit piece of information is the result of the correlation process. In handling the copyright issue, we consider information such as producer, distributor, and user to be crucial. Obviously, including this type of information in the copyright-protection watermark signal while keeping its randomness is more advantageous than the 1-bit-piece-of-information approach. Thus, we addresses the need to include meaningful information within this signature such that a string of English characters, numbers, and punctuation is embedded within the watermark signal, while the signal remains a sequence of random variables.

With the proposed algorithm (which will be introduced in the next section), the watermark signal appears as a random variable sequence to someone attempting an attack. When image ownership is debated, the true owner can reveal convincing

meaningful information from the signature extracted from the image. This new approach is referred to as *embedding digital watermarking using multiple signaling.* It will be shown how with this approach, the digital watermark signal can be interpreted (decoded) in two different ways. One way will interpret the watermark signal as a unique random variable sequence (signature), and the other way will interpret the watermark signal as a sequence of meaningful English characters. The approach relies on the use of generation matrices [21] while dealing with this problem as power-constrained multiple signaling over an AWGN channel [40].

In the analysis and simulation presented in this chapter, we use images of size 256×256 pixels, and we apply DCT to blocks of 8×8 pixels. We select some l significant AC coefficients in each transformed block to carry the watermark signal, each coefficient carrying one bit. Thus, the dimension of the watermark signal is $N = \frac{256}{8} \times \frac{256}{8} \times l$.

This chapter proceeds as follows. Section 2.1 explains how generation matrices can be used with multiple signaling. Section 2.2 studies the relationship between the image's noise level and the watermark signal's SNR. Section 2.3 gives an estimate of the maximum number of symbols the watermark signal can carry. Section 2.4 compares different methods of implementation to find the most practical and reliable way to use multiple signaling. Section 2.5 introduces a direct sequence spread-spectrum watermarking technique that is an extension of the multiple signaling technique. Section 2.6 is a summary.

2.1 The Use of Generation Matrices

With multiple signaling, the watermark signal is first constructed as a sequence of binary random variables. As will be explained shortly, because a watermark signal that is a sequence of Gaussian random variables is preferred [3], we will show how to turn our watermark signal from a sequence of binary random variables to a sequence of Gaussian random variables. First, let us show how the sequence of binary random variables is constructed. With multiple signaling, the watermark signal is divided into multiple segments taken from a maximum separable set of signals. In this section,

we intend to show that the watermark signal, in this case, is a sequence of binary random variables. This can be explained with the following simple example.

Suppose we want the watermark signal to carry a string of symbols taken from a source alphabet with 4 symbols. That is, we have 4 symbols, a, b, c and d. Also, suppose we want the length of each segment to be $n = 4$ bits. Then, we need to construct a maximum separable signal set of 4 entries, each 4 bits long. A possible generation matrix, G, for this set is

$$G = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

This generation matrix is used as follows. The four original symbols are assigned to binary representations $a = 00, b = 01, c = 10$, and $d = 11$. Each binary representation is multiplied by G to generate the following maximum separable set of signals:

$S_1 = 0000, S_2 = 1010, S_3 = 0101$, and $S_4 = 1111$.

The reader can refer to [21] for more details about how a generation matrix is constructed from a generation polynomial.

There are many ways to construct the desired set of maximum separable signals using G. If we alternate rows or columns in the matrix, we will still be able to construct a maximum separable set of signals. Suppose we alternate the two rows of the matrix to obtain $G1$, as follows:

$$G1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

The constructed set of signals will be

$S_1 = 0000, S_2 = 0101, S_3 = 1010$, and $S_4 = 1111$.

Similarly, suppose we alternate the third and fourth columns in G. We then will obtain the following generation matrix:

$$G2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

The constructed set of signals will be

$S_1 = 0000, S_2 = 1001, S_3 = 0110$, and $S_4 = 1111$.

The larger the set of signals, the larger the generation matrix, and hence the larger the possible number of signal sets. Considering that there are many

generation matrices that can be used, and that there are many possible sets that can be generated from a generation matrix, if we consider each bit in the signal individually, it could be 0 or 1 with a probability of $\frac{1}{2}$. Thus, for an attacker with no knowledge of which generation matrix is being used, the entire watermark signal is a sequence of binary random variables. When debating the ownership of an image, the owner can reveal the generation matrix used with the random variable sequence. The generation matrix will result in decoding a sequence of meaningful information that could be used as a stronger proof of ownership than just revealing some random variable sequence.

2.2 The Image's Noise Level and the Watermark's SNR

As explained in Chapter 1, the watermarking signal is carried by some selected coefficients in the DCT domain. Each coefficient of value $f_{i,j}$ becomes $f_{i,j} + \Delta$ if it is to carry a binary 1, and becomes $f_{i,j} - \Delta$ if it is to carry a binary 0 (the number of encoded bits N in our simulation is 3072, utilizing 3072 coefficients). When selecting the value of Δ, one encounters a tradeoff. If Δ is large (that is, the watermark signal is strong), we are more likely to detect the encoded signal without error in the presence of noise. The watermark signal in this case is more robust, but the effect of the strong signal on image quality may be more noticeable in the spatial domain. On the other hand, if Δ is small (that is, the watermark signal is weak), the signal's effect on image quality may be less noticeable in the spatial domain. The watermark signal in this case is less robust since it is more likely to misdetect the signal in the presence of high noise power. Visual masking techniques consider this dilemma. In [14], experimental results showed that adaptively selecting the value of Δ to be a real number between 2-6 gives a degradation level that is unnoticeable to the human eye, and at the same time, the watermark signal (as a unique binary random variable sequence) is detectable even if the image is degraded with a PSNR as low as 19 db.

To extract the watermark signal from the image under examination, we subtract the coefficients of the original image (the one without a watermark) from the coefficients of the image under test (see Figure 1.1). If the image being tested is noise free,

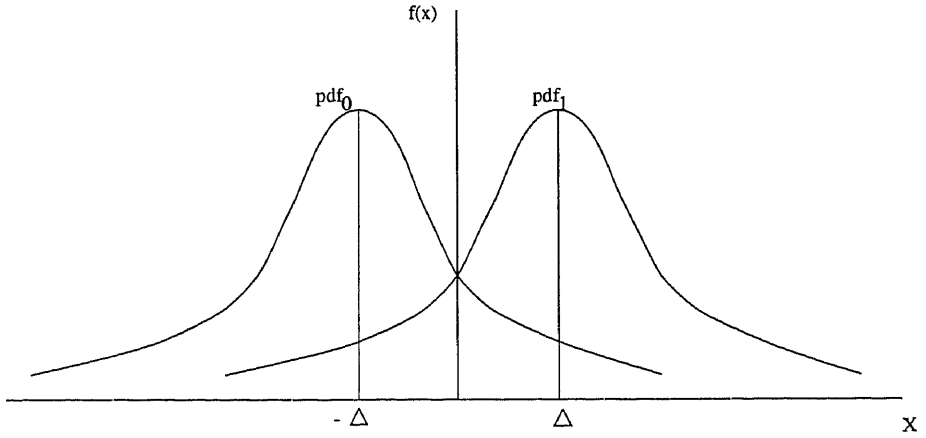


Figure 2.1 Bit-by-bit Detection of the Watermark Signal

we obtain $f_{i,j} - f_{i,j} = 0$ at the locations of the coefficients not carrying the watermark bits. At the locations of the coefficients carrying the watermark information bits, we obtain the watermark signal as follows:

$$f_{i,j} \pm \Delta - f_{i,j} = \pm\Delta. \tag{2.1}$$

If the image being tested for the watermark signal contains noise (e.g., the image has gone under some type of processing or any sort of degradation, or another watermark signal has been added by someone attempting an attack), the coefficients carrying the watermark information will have some additive noise. In this case the result of the subtraction in Equation (2.1) will be

$$f_{i,j} \pm \Delta + N_0 - f_{i,j} = \pm\Delta + N_0, \tag{2.2}$$

where N_0 is the resultant additive noise at the i, j location.

The probability distribution function (pdf) of the additive noise N_0 can be approximated as a Gaussian distribution with a zero mean and a variance of σ^2. We will refer to this additive noise as $N(0, \sigma)$. This Gaussian approximation can be validated analytically by the central limit theory [19]. The variance σ^2 is decided by the amount of noise introduced into the image.

The similarity between the detection of the watermarking signal from the image and the detection of a signal over an additive white Gaussian noise (AWGN) digital

communication channel was published in papers such as [3] and [34]. As Figure 2.1 shows, there are two hypotheses pdf_0 and pdf_1 for decoding 0 and 1 respectively. (This point will be discussed in more detail in Chapter 4). Noisy signal pdf_0 is $N(-\Delta, \sigma)$ and noisy signal pdf_1 is $N(+\Delta, \sigma)$. Since 0 and 1 occur with equal probabilities, if we decode each bit separately we can use maximum likelihood (ML) decoding [40]. This ML decoding will decide 0 if the value of $\pm\Delta + N_0 < 0$ and decide 1 if the value of $\pm\Delta + N_0 > 0$. Clearly, there is a probability of detecting a 0 as 1 and vice versa. We will refer to this probability of bit-decoding error as P_e. One can see that as PSNR decreases, σ increases and hence P_e increases. This means that if we are to look at each watermark bit individually, we can see that the noise power of the image in the spatial domain is translated into noise power in the transform domain at the locations carrying the watermark signal. This results in a ratio between the watermark signal and the noise power SNR_{wm}, which is equivalent to the SNR of *bit-by-bit* signaling over an AWGN digital communication channel. SNR_{wm} is given by the following relation:

$$SNR_{wm} = \frac{\Delta^2}{\sigma^2}. \tag{2.3}$$

P_e can be obtained from the following relation [19]:

$$P_e = \frac{1}{\sqrt{2\pi}\sigma} \int_{\Delta}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx. \tag{2.4}$$

Looking at this analysis, one may think that applying some sort of error-correction coding to lower the probability of bit error P_e is a suitable solution. The analysis in the next section and the simulation in Section 2.4 will show that error-correction codes are not the best solution for the intended multiple signaling. The fact that we are limited by a certain signal power Δ^2, and the fact that we wish to be able to include symbols in the watermark signal at low PSNR will lead us to consult Shannon's capacity theorem [1] [2] [39]. This theorem implies that the noise, the available number of dimensions, and the available signal power set a limit only on the *rate* at which reliable communication is achieved; *not* on the *accuracy*. The fact that watermark signaling is power constrained should only set a limit on the number of symbols to be included in the signal.

2.3 Image Capacity

The capacity per transmission of the AWGN channel with power constraint and a known noise variance is given by

$$C = \frac{1}{2}\log_2(1 + SNR) \qquad bits, \tag{2.5}$$

where the channel capacity C is the maximum number of information bits per transmission.

For our case, the number of locations selected to carry the watermark signal resembles the number of available transmissions. Thus, Equation (2.5) becomes

$$C_{wm} = \frac{N}{2}\log_2(1 + SNR_{wm}) \qquad bits, \tag{2.6}$$

where C_{wm} is the image capacity with respect to the watermark signal, N is the number of coefficients selected to carry the watermark signal (which is 3072 in our simulation), and SNR_{wm} is the average watermark signal-to-noise ratio from Equation (2.3).

Example: Suppose we want to extract the watermark signal accurately even if the image suffers from degradation equivalent to a PSNR of 30 db. Estimating σ for this noise power and using Equation 2.3, we can calculate SNR_{wm}. In our simulation, for the "Lena" image, SNR_{wm} for PSNR = 30 db is found to be around 0.32 (- 4.95 db). Substituting $SNR_{wm} = 0.32$ and $N = 3072$ in Equation (2.5), we get $C_{wm} = 615$ *bits*. Thus, according to Shannon's coding theorem, to be able to extract the watermark signal reliably, we can encode up to 615 information bits (as a single signal). This theoretical limit can be approached by utilizing the maximum number of dimensions (3072 in our simulation). This should be done by using a set of 2^{615} orthogonal maximum separable signals. The dimension of each signal is 3072. The signal extracted from the image is to be correlated with each signal in the set, and the entry with the highest correlation is to be selected as the decoded signal. This decoded signal is then mapped to a unique binary sequence of length 615 bits out of 2^{615} possible binary sequences. Certainly, the application of such huge sets is impractical.

One can see from this example how conventional copyright protection watermarking techniques achieve their high robustness. The selected unique random

number sequence is actually a single signal in this huge set, and the correlation between the extracted signal and (only) this signal results in a high correlation coefficient. Any other random number sequence that does not match the original signal will result in low correlation.

For multiple signaling, since the limit on the number of information bits is decided by three factors: the additive noise, the available number of dimensions, and the available signal power, and since we have a power-constrained signal and need to accommodate high noise, the factor we have the freedom to consider is the number of dimensions per signal. In the next section we will answer the question of how to implement multiple signaling using a practical approach while aiming to achieve a rate as close as possible to the theoretical capacity. The next section also compares different approaches and shows how the power-constrained approach can be applied.

2.4 Watermark Signal Detection

The problem addressed in this section is defined as follows: For a given image in which we can select no more than N coefficients to carry the watermark signal, with the watermark's bit signaling power of Δ^2, and for an additive noise of power σ^2, what is the best approach for including multiple symbols in the watermark signal?

The types of information we need to include in the watermark signal are English letters (26 characters), numeric letters (10 numbers), and some punctuation. Thus, we will need only 6 bits for each information symbol. We start by considering the part of the standard ASCII character table starting from the (space) character (with a decimal value of 32) to the (_) character (with a decimal value of 95). This part of the ASCII table covers all the capital letters, the numbers, and punctuation. We will map this practical set of 64 entries to a signal set of 64 entries. To encode the letter "A," which has an ASCII value of 65, we consider the signal with index (65-31=34) in the signal set. On the other hand, if the decoded signal is the first entry in the table, it will be translated to a (space) character. Table 2.1 shows how these 64 symbols are mapped to a set of 64 maximum separable signals of 31 bits per signal. This set is generated using the same generation matrix used for the linear block code

BCH (31:6). This generation matrix is built from a generation polynomial. The reader can refer to [21] for details about how to build a generation matrix from a generation polynomial. The generation polynomial $g(x)$ and the generation matrix G are as follows:

$$g(x) = x^{25} + x^{24} + x^{21} + x^{19} + x^{18} + x^{16} + x^{15} + x^{14} + x^{13} + x^{11} + x^9 + x^5 + x^2 + x + 1$$

$$G = \begin{pmatrix} 1110010001010111101101001100000 \\ 0111001000101011110110100110000 \\ 0011100100010101111011010011000 \\ 0001110010001010111101101001100 \\ 0000111001000101011110110100110 \\ 0000011100100010101111011010011 \end{pmatrix}. \tag{2.7}$$

Consider an example of the implementation of our watermarking method. When the "Lena" image with a PSNR of 30 db was used, by using Equation (2.6) and assuming $l = 3$, we calculated 615 as the number of information bits (capacity) that can be included in the watermark signal. Because we are using 6 bits per symbol, this translates into about 102 information symbols. Because we are using 3072 ($=\frac{256}{8} \times \frac{256}{8} \times 3$) coefficients, reaching the capacity means that we use a code rate $R = \frac{3072}{615} \simeq 5$ (encoded bits per information bit). For a code rate in this vicinity, we simulated three techniques to show how the power-constrained approach is the best method and how it can be practically implemented.

1) Multiple Embedding (introduced in [17]): With this approach, each information bit is repeated 5 times; that is, each 6-bit symbol is translated to 30 bits, and the watermark signal is composed of these 30-bit segments. When detecting the watermark signal, ML decoding is used to decode each bit individually and majority voting decides if the bit is 0 or 1. This approach is equivalent to applying the error correction repetition code (5:1) using bit-by-bit signaling over the given communication channel.

2) Error Correction Coding, where the error correction code BCH (31:6) is utilized: With this approach each 6-bit information symbol is translated into a 31-bit symbol using the BCH (31:6) encoder. The watermark signal is composed of 99 segments, each of which is 31 bits long. When detecting the watermark signal, each bit is decoded individually using ML decoding, and each resulting 31-bit segment

Table 2.1 Mapping Part of the ASCII Table to a Maximum Separable Set of 64 Signals of 31 Bits per Signal.

Signal Index	ASCII Value	Character	Utilized Signal of Dimension 31
1	32		0000000000000000000000000000000
2	33	!	0000011100100010101111011010011
3	34	"	0000111001000101011110110100110
4	35	#	0000100101100111110001101110101
5	36	$	0001110010001010111101101001100
6	37	%	0001101110101000010010110011111
7	38	&	0001001011001111100011011101010
8	39	'	0001010111101101001100000111001
9	40	(	0011100100010101111011010011000
10	41	)	0011111000110111010100001001011
11	42	*	0011011101010000100101100111110
12	43	+	0011000001110010001010111101101
13	44	,	0010010110011111000110111010100
14	45	-	0010001010111101101001100000111
15	46	.	0010101111011010011000001110010
16	47	/	0010110011111000110111010100001
17	48	0	0111001000101011110110100110000
18	49	1	0111010100001001011001111100011
19	50	2	0111110001101110101000010010110
20	51	3	0111101101001100000111001000101
21	52	4	0110111010100001001011001111100
22	53	5	0110100110000011100100010101111
23	54	6	0110000011100100010101111011010
24	55	7	0110011111000110111010100001001
25	56	8	0100101100111110001101110101000
26	57	9	0100110000011100100010101111011
27	58	:	0100010101111011010011000001110
28	59	;	0100001001011001111100011011101
29	60	¡	0101011110110100110000011100100
30	61	=	0101000010010110011111000110111
31	62	¿	0101100111110001101110101000010
32	63	?	0101111011010011000001110010001
33	64	@	1110010001010111101101001100000

Table 2.1 (Continued)

Signal Index	ASCII Value	Character	Utilized Signal of Dimension 31
34	65	A	11100011011101010000010010110011
35	66	B	11101010000100101100111111000110
36	67	C	11101101001100000111001000010101
37	68	D	11111000110111010100000100101100
38	69	E	11111111111111111111111111111111
39	70	F	11110110100110000011100100010101
40	71	G	11110001101110101000010010110001
41	72	H	11011101010000100101100111111000
42	73	I	11011010011000001110010000101011
43	74	J	11010011000001110010000101011110
44	75	K	11010100001001011001111100011101
45	76	L	11000001110010001010111110110100
46	77	M	11000110111010100000100101100111
47	78	N	11001111100011011101010000010010
48	79	O	11001000101011110110100110000001
49	80	P	10010110011111000110111010100000
50	81	Q	10010001010111101101001100000011
51	82	R	10011000001110010001010111110110
52	83	S	10011111000110111010100000100101
53	84	T	10001010111101101001100000011100
54	85	U	10001101110101000010010110011111
55	86	V	10000100101100111110001101110101
56	87	W	10000011100100010101111011010001
57	88	X	10101111011010011000001110010000
58	89	Y	10101000010010110011111100011011
59	90	Z	10100001001011001111100011011110
60	91	[	10100110000011100100010101111101
61	92	\	10110011111000110111010100001000
62	93	]	10110100110000011100100010101111
63	94	^	10111101101001100000111001000010
64	95	_	10111010100001001011001111100001

(code word) is decoded using the BCH (31:6) error correction decoder. This approach is equivalent to using bit-by-bit signaling over a communication channel and using the BCH (31:6) error correction code to recover channel errors. This error correction code is capable of correcting 7 errors in each encoded block with a length of 31 bits.

3) Correlation Detection: With this approach we constructed a set of 64 signals using the same generation matrix, and mapped them to part of the ASCII table as explained above. This mapping is shown in Table 2.1. The dimension of each signal is 31 bits. The signal energy is obtained by translating 0 to $-\Delta$ and 1 to $+\Delta$ (which results in an orthogonal set of signals). This set of signals is maximum separable with a minimum Hamming distance between any two signals equal to 15 bits. That is, any two signals in the set differ by at least 15 bits. Notice that the code rate is around 1/5 which is the same as with the previous approaches. The decoding process correlates the extracted 31-bit signal with each entry in the signal set and selects the entry with the highest correlation as the encoded character. The correlation between the extracted signal and the i^{th} entry in the signal set, ρ_i, is obtained by the following relation:

$$\rho_i = \frac{\sum_{j=0}^{n-1} x_j^* . x_{i,j}}{\sqrt{\sum_{j=0}^{n-1} (x_j^*)^2}}, \tag{2.8}$$

where n is the signal dimension (31 bits in this case), $x_{i,j}$ is the base signal's magnitude level ($\pm\Delta$), and $x_{i,j}^*$ is the j^{th} bit magnitude level of the extracted signal ($\pm\Delta + N_0$).

The curve in Figure 2.2 shows the plot of the probability of symbol decoding error, P_E, in each of the three cases. Notice that in all three cases the code rate is around $\frac{1}{5}$. The first case gives the poorest performance, since it has the worst utilization of both the signal power and error correction coding. The reader may refer to [40] to see how bit-by-bit signaling is prone to channel noise in comparison with n-dimensional signaling for power-constrained signals. The reader may also refer to [21] to see how repetition codes are not recommended for channel coding and that they actually cause a negative coding gain.

Although the second case uses a powerful error correction code, the fact that bit-by-bit signaling is used explains why this case gives about 2 dB less than the

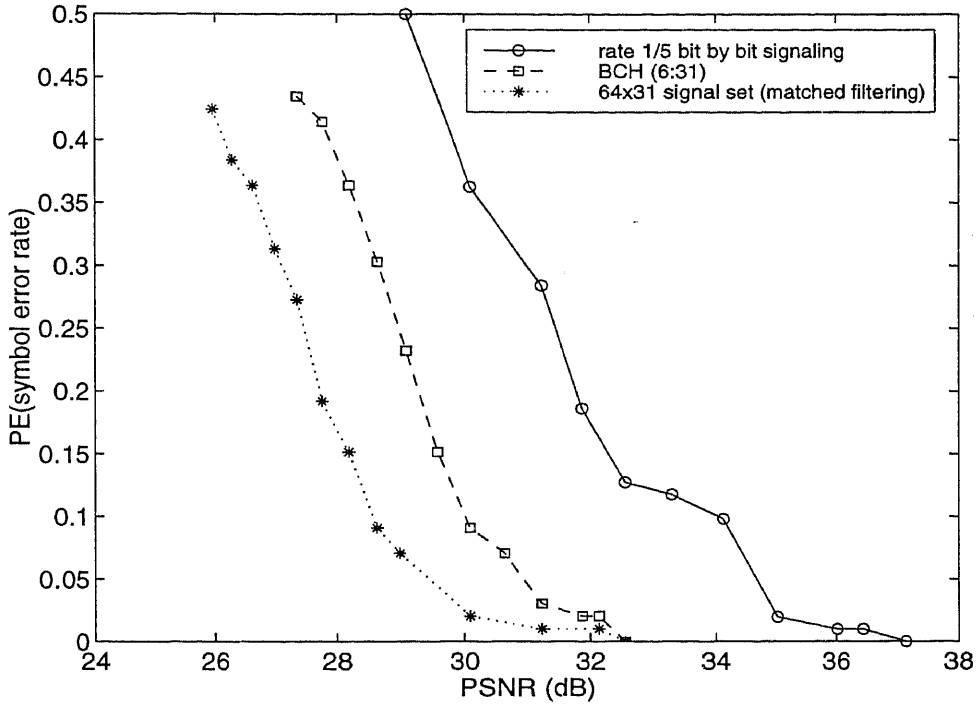


Figure 2.2 Performance of 3 Different Approaches with Rate 1/5 Signaling

third case for the same level of P_E. The fact that embedding the watermark signal in a transform domain is analogous to spread spectrum signaling does not mean that we should consider error correction codes unless the watermark signal-to-noise ratio is more than -1.6 dB [1].

One should consider the coding gain achieved from increasing the signal power by increasing its dimension versus the coding gain achieved from using error correction coding. As is apparent from these results, the constraint in watermark-signal power makes the gain achieved from increasing the signal power by increasing its dimensionality better than the gain obtained by error correction codes. Figure 2.3 shows the probability of decoding symbol error, P_E, with different lengths of the signal set. In all three cases the set size is 64 entries, only we change the number of bits (dimension) of each entry. This is equivalent to increasing the dimensionality of the signals. As the curves in Figure 2.3 illustrate, if we increase the dimensionality of the signal, we can tolerate a lower PSNR. That is, we trade the amount of

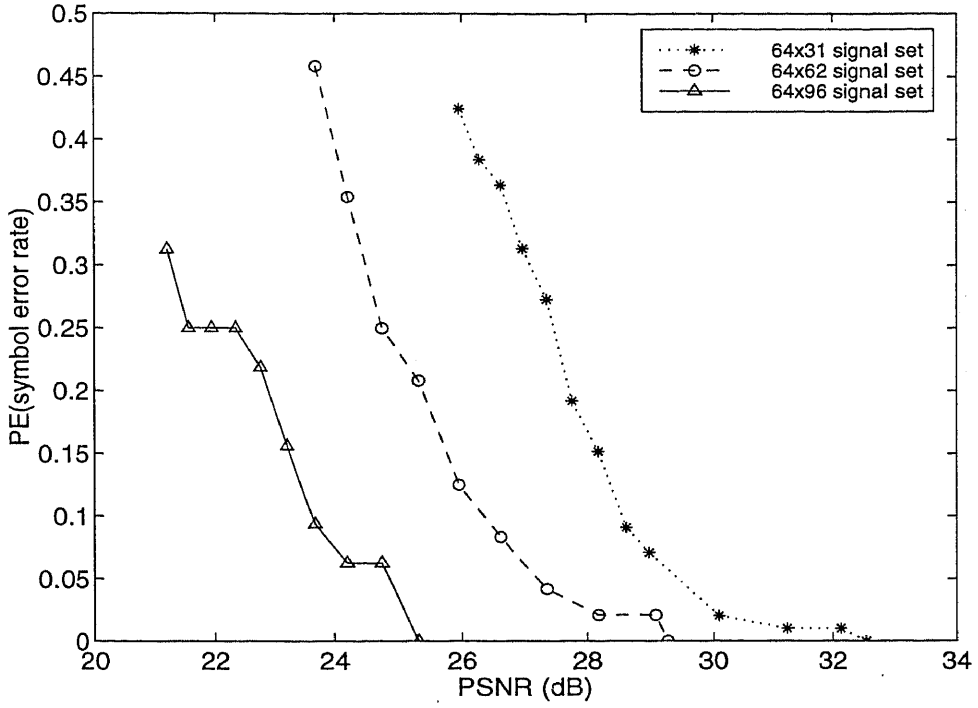


Figure 2.3 Relationship Between Error Rate and PSNR for Different Signal Sizes

information for robustness. With the 96-bit signal set, we were able to include the 32-character string (NEW JERSEY INSTIT'. OF TECH. 1998) in the watermark signal and extract it reliably at a PSNR of as low as 26 dB (see the images in Figures 2.4, 2.5, and 2.6).

So far, we studied multiple signaling when the constructed watermark signal is a sequence of binary random variables. Such a watermark signal has the advantage of utilizing all the power that can be embedded in the image. Imagine using an adaptive technique like visual masking, the maximum allowed energy decided by this technique (Δ) can be fully utilized with binary values. With a watermark signal that is constructed as a sequence of Gaussian random variables, the value decided by a generated Gaussian variable limits the amount of energy to be embedded in the corresponding coefficient. As the next section explains, the advantages of constructing a signal that is a sequence of Gaussian random variables surpass the one single advantage of constructing a signal that is a sequence of binary random variables.

Figure 2.4 Lena Image Before Adding the Watermark Signal

Figure 2.5 Lena Image After Adding the Watermark Signal

Figure 2.6 Lena Image with PSNR = 26, and the Watermark Multiple Signaling Approach Detected All Symbols Error Free

2.5 Direct-Sequence Spread-Spectrum Watermark Signaling

When the watermark signal constructed using the multiple signaling algorithm is a sequence of binary random variables, the following concerns present a main drawback of the algorithm:

1. A binary random variables signal is not as robust as a Gaussian random variables signal when it comes to a multiple-copy attack. According to reference [3], for a fraudulent claimant obtaining multiple copies from a watermarked image, it is easier to attack a binary signal than a Gaussian signal.

2. It is possible for an attacker to filter as much as half of the signal power by adding (or subtracting) $\frac{\Delta}{2}$ to all the coefficients carrying the binary signal. This can happen if the attacker knows the transform domain used, the size of the transformed blocks, and the locations of the utilized coefficients. Looking at this problem from a communications point of view, it can be interpreted as follows. A binary signal will have a spectrum that is narrow and hence it is easier to filter out. On the other hand, a Gaussian signal has a wider spectrum that is hard to filter out.

3. As mentioned in the introduction, an image-dependent watermark signal is needed to prevent an attacker from simply adding (or subtracting) his signature to the watermarked image.

The solution to the above concerns lies in a communications technique called direct-sequence spread spectrum [29]. With this technique, we will be allowed to carry a string of meaningful information in a Gaussian random variable sequence instead of a binary random variable sequence. To understand how this technique can be used with digital image watermarking, let us refer to the binary sequence generated using a generation matrix as $B(N) = (i_1, i_2, i_3, ..., i_N)$, where i is a binary variable taking the value 0 or 1, and N is the length of the sequence (number of bits in the entire watermark signal). From $B(N)$, we can obtain $B1(N) = (k_1, k_2, k_3, ..., k_N)$, where k_j is obtained from i_j $(1 \leq j \leq N)$ by the following relation:

$$k_j = \begin{cases} -1 & \text{if} \quad i_j = 0; \\ 1 & \text{if} \quad i_j = 1. \end{cases} \tag{2.9}$$

Now let us generate a Gaussian random variables sequence of length N variables, and refer to it as $G(N) = (g_1, g_2, g_3, ..., g_N)$. This sequence should be image-dependent to resist the attack mentioned in point 3 above. In this work, we are not concerned with showing how this Gaussian random variable sequence is generated: The reader can refer to [5] for details about these techniques. From $B1(N)$ and $G(N)$, we can obtain another Gaussian random variables sequence $G1(N) = (h_1, h_2, h_3, ..., h_N)$, where h_j is given by

$$h_j = g_j \times k_j. \tag{2.10}$$

$G1(N)$ is Gaussian because we will just change the sign of each element g_j in $G(N)$ that is multiplied by -1. Now, the actual sequence we embed in the image is $G1(N)$. When $G1(N)$ is extracted from the image, $B(N)$ (which carries a string of symbols) can be obtained based on $G(N)$ as follows.

$$i_j = \begin{cases} 0 & \text{if } \frac{g_j}{h_j} = -1; \\ 1 & \text{if } \frac{g_j}{h_j} = 1. \end{cases} \tag{2.11}$$

Note that the above technique also applies to embedding and extracting the watermark signal as segments where $B(N)$, $G(N)$, and $G1(N)$ are replaced by $B(n)$, $G(n)$, and $G1(n)$, where n is the segment length.

2.6 Summary

This chapter introduced a watermarking technique that constructs the watermark signal using generation matrices. This multiple signaling technique allows a copyright protection signal to carry a string of meaningful information that can be utilized to support true image ownership, and at the same time the signal is a sequence of random variables. We showed how the watermark signal can suffer from noise. We analyzed the amount of meaningful information symbols that can be included in an image watermark signal. We compared three different implementation strategies; and we showed how the power-constrained approach can be used with the multiple signaling technique; we showed that to construct the encoded sequence of symbols in the existence of more noise, the dimension of the encoded signals should be increased,

and hence a smaller number of symbols may be included in the watermark signal. We showed how we can turn the binary random variable sequence generated from a generation matrix to an image-dependent Gaussian random variable sequence using a direct-sequence spread-spectrum technique.

With multiple signaling, the process of detecting the watermark signal can be done with two different methods. The first method correlates segments of the extracted watermark signal with a set of maximum separable signals decoding a string of meaningful symbols included in the watermark. The second method correlates the entire extracted signal with the original watermark signal and uses this correlation as an indication of the existence of the watermark in the tested image. While the first detection method reveals a string of meaningful information, the second detection method gives a watermark signal that is more robust (can be detected in the presence of high noise). The next chapter studies the capacity achieved with both detecting methods.

CHAPTER 3

CAPACITY SATURATION OF THE WATERMARK SIGNAL

In the previous chapter, we showed how Shannon's capacity theorem is applied to the multiple-signaling watermarking technique by decreasing the number of symbols carried in the signal to decrease the probability of symbol decoding error. This chapter analyzes the increase in the signaling capacity as a result of increasing the dimension of (number of bits in) the watermark signal while fixing the number of symbols carried in the watermark signal. We will show that the capacity approaches saturation as the signal dimension increases. We will also show that if the watermark signal is meant to carry only the owner's signature (just a sequence of random variables or, with the multiple signaling approach, when we decode the entire signal at once without segmentation), the number of coefficients per transformed block that needs to be utilized is limited. For a 256×256 image, when we apply the DCT to 8×8 blocks, three coefficients per block are sufficient. Increasing the signal dimension by utilizing four coefficients per block instead of three, will increase the computational complexity without significantly increasing the signal's robustness. On the other hand, if the watermark signal is segmented to carry a string of symbols (with multiple signaling when decoding each segment of the watermark signal separately), addressing the capacity issue solely in light of the signal dimension would affect the probability of symbol decoding error. By consulting Shannon's capacity theorem (as we did in Chapter 2), we increase the segment power by decreasing m (the number of segments in the watermark signal) to reduce the probability of symbol decoding error. We will show how to achieve a certain percentage of the maximum capacity for a given signal-to-noise ratio of the watermark signal (SNR_{wm}), while taking the probability of symbol decoding error into consideration.

In this chapter, Section 3.1 explains how the watermark signal is a large-dimension power-constrained signal. Section 3.2 analyzes the signaling capacity with segmentation and without segmentation and gives a numeric example to clarify the relationship between the achieved capacity, (SNR_{wm}), and the segment dimension n. Section 3.3 is a summary.

3.1 Power-Constrained Watermark Signals

In digital image watermarking, a certain amount of information is embedded in the image without causing visible degradation to the image quality. As mentioned in Chapter 2, we use images of size 256×256 pixels, and we apply the DCT to blocks of 8×8 pixels. We select some l significant AC coefficients in each transformed block to carry the watermark signal, each coefficient carrying one bit. Thus, the dimension of the watermark signal is $N = \frac{256}{8} \times \frac{256}{8} \times l$. For copyright-protection watermarking techniques, a good method for detecting the signal is to correlate the signal originally embedded into an image and its noisy version extracted from the image under debate. The high correlation between the original watermark signal and the noisy version is a measure of the robustness of the watermarking technique and is used as a protection of ownership. When we decode each segment separately with multiple signaling, each segment of the watermark signal is taken from a maximum-separable set of signals. Detection techniques correlate the extracted noisy segment with all entries in the signal set and decode the symbol corresponding to the signal that gives the highest correlation.

To make the watermark signal robust (difficult to remove), watermarking techniques try to increase either the signal power (the amount of energy added to the coefficients selected to carry the watermark bits) or the signal dimensionality (the number of bits in the signal). Visual masking techniques [14] are used to increase the signal power. As mentioned above, it is possible to reach a signal dimension of a few thousand bits for images of size 256×256. Because of the unacceptable visual degradation that can occur from drastically changing the magnitude of the selected coefficients, the signal power is limited. As a result, the watermark signal is a large-dimension power-constrained signal. The watermark signal is power constrained in the sense that if we are to increase the number of coefficients carrying the signal, we have to distribute the signal energy among all coefficients. That is, to increase N we have to decrease Δ. Approaching the watermark signal as a large-dimension power-constrained signal provides the answer for the following question: Is it always desirable to increase the watermark signal dimension?

To answer the above question, we analyze the watermark signal's capacity as a function of the signal dimension ν. (ν could be the dimension of the entire signal or of a segment depending on the watermarking detection technique used). We will show that for large ν, the increase in the signaling capacity (capacity gain) due to increasing the signal dimension reaches saturation. This saturation depends on SNR_{wm}. As SNR_{wm} decreases, the capacity saturation is reached with a smaller ν. On the other hand, as SNR_{wm} increases, the capacity saturation is reached with a larger ν. With our multiple signaling technique, when the entire signal is decoded at once, the signal should be robust at a very low SNR. Thus, achieving a negligible amount of capacity would require us to increase the signal dimensions by thousands of bits. On the other hand, with multiple signaling, when we decode each segment separately, we work at a relatively higher SNR. Capacity saturation may be reached with a signal dimension that is larger than the nonsegmented case. Thus, increasing the achieved capacity by a considerable amount could be possible by increasing the watermark signal dimension (i.e., utilizing more coefficients per each transformed block).

3.2 Signaling Capacity

We established in Chapter 2 that in image watermarking, detecting the watermark signal is similar to detecting a signal over an additive white Gaussian noise (AWGN) channel. The capacity per transmission of a power-constrained AWGN channel with a known noise variance is given by

$$C = \frac{1}{2}\log_2(1 + SNR) \qquad bits, \tag{3.1}$$

where C is the channel capacity or the maximum number of information bits per transmission [1].

For the digital image watermark signal, if we are to consider each bit as a signal, the number of locations selected to carry the watermark signal resembles the number of available transmissions. Thus, Equation (3.1) becomes

$$C_{wm} = \frac{\nu}{2}\log_2(1 + SNR_{wm}) \qquad bits, \tag{3.2}$$

where C_{wm} is the image capacity with respect to the watermark signal; ν is the number of coefficients selected to carry the watermark signal (since each coefficient carries one encoded bit, ν is the signal dimension as well); SNR_{wm} is the average watermark SNR and is equal to $\frac{\Delta^2}{\sigma^2}$, where Δ is the added magnitude at the selected coefficients (the average watermark signal magnitude); and σ^2 is the variance of the additive noise. As mentioned above, ν could be the entire watermark dimension N or a segment dimension n depending on the watermarking technique used.

Because of the limitation on total signal power, increasing watermark signal power can result in unacceptable degradation of the image quality. In other words, to increase the signal dimension N (the number of coefficients carrying the watermark signal), we have to decrease the average magnitude changes at each selected coefficient. This resembles increasing the signaling bandwidth for a power-constrained signal over an AWGN channel [29]. For a signal with dimension ν, we can find the signaling capacity, C_s, as follows:

$$C_s = \frac{\nu}{2}\log_2(1+\frac{P_s}{\sigma^2}) \qquad bits, \tag{3.3}$$

where P_s is the constrained signal power, and σ^2 is the noise variance given by

$$\sigma^2 = N_o W = \frac{\nu N_o}{2}, \tag{3.4}$$

where $W = \frac{\nu}{2}$ is the equivalent signaling bandwidth, and N_o is the noise power spectrum. Thus, the signaling capacity in Equation (3.3) is given by

$$C_s = \frac{\nu}{2}\log_2(1+\frac{2P_s}{\nu N_o}) \qquad bits. \tag{3.5}$$

We will use Equation (3.5) to study the capacity saturation of the watermark signal when the entire signal is decoded at once (nonsegmented case), and when each segment of the signal is decoded separately (segmented case).

3.2.1 Capacity Saturation without Segmentation

Figure 3.1 shows the relationship between $C_s/\frac{P_s}{N_0}$ and the total signal dimension, N, for different SNR_{wm}. The negative values of the SNR's are due to the fact that we

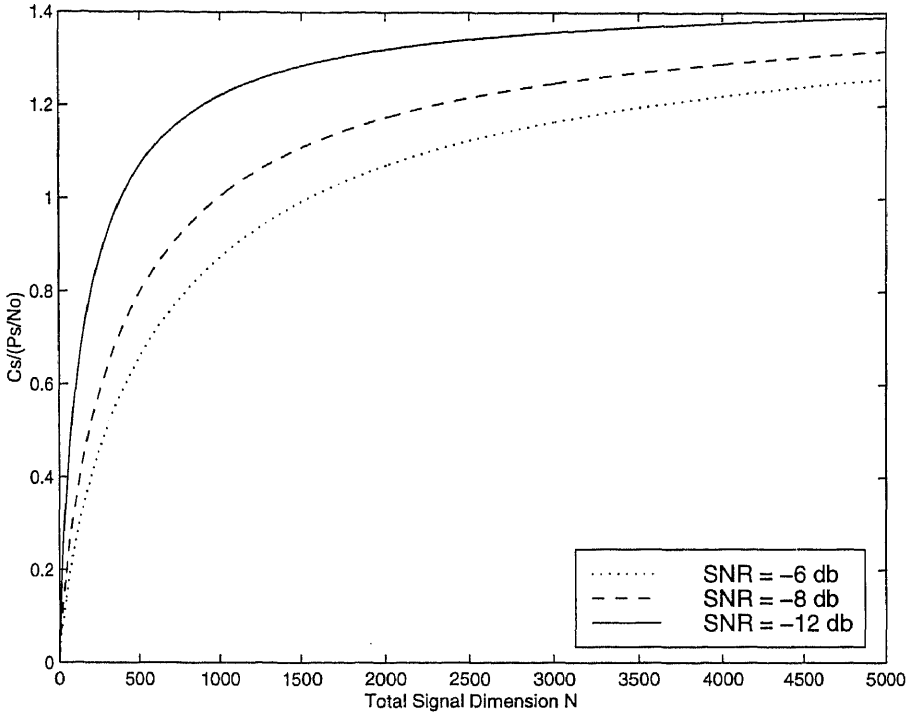


Figure 3.1 Capacity Gain Saturation as the Total Watermark Signal Dimension Increases (Nonsegmented Case)

are dealing with a power-constrained signal. Refer to the example in Section 2.3 to see how, for the "Lena" image at $PSNR = 30$, we found that $SNR_{wm} = -4.95$ db. In Section 2.4 we even dealt with lower SNR's. In Figure 3.1, we estimated the watermark signal power, which a 256×256 image may tolerate, using the "Lena" image. One can see from the curve in Figure 3.1 that the capacity, C_s, saturates as N approaches infinity. Specifically, C_s will approach $1.44\frac{P_s}{N_0}$ asymptotically. This saturation can be explained as follows: As N increases, more noise is included in the detected signal, causing the logarithm in Equation (3.5) to limit the signaling capacity [29]. Also as SNR decreases, the dimension of the signal required to reach a certain saturation level decreases.

This analysis indicates that one can achieve a larger capacity by increasing the dimensionality of the signal. This capacity increase, however, is limited. That is, the capacity increase will be negligible as the dimensionality approaches infinity. One should consider the tradeoff between the signaling capacity and the computational

complexity. On one hand, a larger signaling capacity is advantageous since more information can be embedded (with a watermark signal that is only a sequence of random variables, capacity is traded for robustness). On the other hand, a larger signal will result in more computational complexity.

Note that this analysis agrees with what is mentioned in reference [3], where the watermarking technique used a 256×256 image and applied the DCT to a watermark signal decoded as one sequence of random variables. The authors in [3] experimentally measured the robustness of the watermark signal as a function of the number of coefficients utilized in the DCT domain. These experiments measured the probability of false alarm (the probability that a random sequence generated by a fraudulent claimant, when correlated with the extracted noisy signal, gives a higher correlation coefficient than the original sequence). A conclusion was reached that for an image of size 256×256 pixels, increasing the number of utilized coefficients beyond a few thousand coefficients would not considerably increase the robustness of the watermark signal. Looking at the curve in Figure 3.1, one can see how the analysis here agrees with these experiments. Keep in mind that we are dealing with a copyright-protection watermark signal and we care about achieving more capacity specifically at a low SNR. Thus, consider the plots for an SNR of -8 and an SNR of -12 (the saturation is reached faster as the SNR decreases). Moving from 3 coefficients per block (which produces a watermark signal of dimension 3072 bits) to 4 coefficients per block (which produces a watermark signal of dimension 4096 bits) will not have any significant increase on the image capacity.

Because watermarking techniques that deal with the watermark signal as one sequence trade all the capacity for robustness, increasing the signal dimension from 3072 to 4096 bits will have no significant effect on image capacity. As a result, no significant increase of robustness is achieved. This, in turn, means that the probability of false alarm will not decrease. Keep in mind that the sole purpose of increasing the capacity of a copyright protection watermark signal is to decrease the probability of false alarm. Decreasing the probability of false alarm decreases the chance that the correlation between a sequence of random variables (generated by an attacker) and the extracted signal is higher than the correlation between the

original sequence and the extracted signal (the noisy version of the original sequence). Although Figure 3.1 indicates that at higher SNR's (e.g., in the range of -6 to -2 db) more capacity can be achieved by increasing the signal dimension, this increase may be unnecessary. The probability of false alarm at these relatively high SNR's, with such a large dimension signal, is very low anyway.

3.2.2 Capacity Saturation with Segmentation

The analysis and the curve in the previous subsection consider a watermark signal that has one segment (just a sequence of random variables). One may wonder how this applies to multiple signaling when the signal is segmented, and hence the signal power is divided among all the segments. The behavior of the capacity increase with the dimensionality increase illustrated in Figure 3.1 can equally be illustrated if we segment the watermark signal. Note that the purpose of detecting the signal with segmentation is different from detecting the signal without segmentation. Without segmentation, we specifically care about decreasing the probability of false alarm at a very low SNR, as explained above. On the other hand, with segmentation, we care about decreasing the probability of symbol decoding error. If the SNR is very low, such that most symbols are decoded in error, we can only rely on decoding the entire watermark signal as one segment to prove the existence of the signal in the image. Thus, while in the nonsegmented case we care about the capacity saturation when SNR_{wm} is relatively low (around the range of -8 db to -12 db), with the segmented case, we care about the capacity saturation when the SNR_{wm} is relatively high (around the range of -2 db to -8 db).

Consider the scenario we described earlier, that is, an image of size 256 × 256 pixels. The DCT is applied to each non-overlapped block of 8 × 8. Only some l AC coefficients in each block are selected to carry the watermark information. If we divide the total number of DCT coefficients carrying the watermark signal, N, which is $\frac{256}{8} \times \frac{256}{8} \times l$, into m equal segments, we can draw the curve in Figure 3.2. In Figure 3.2 $m = 32$, and hence the signal power is divided by 32. That is, in Equation (3.5) we use $C_s/\frac{P_s}{N_0}$, with P_s being the total signaling power when we plot the curve in Figure 3.1 and we use $C_{s_i}/\frac{P_{s_i}}{N_0}$, with P_{s_i} being the signaling power contained in one

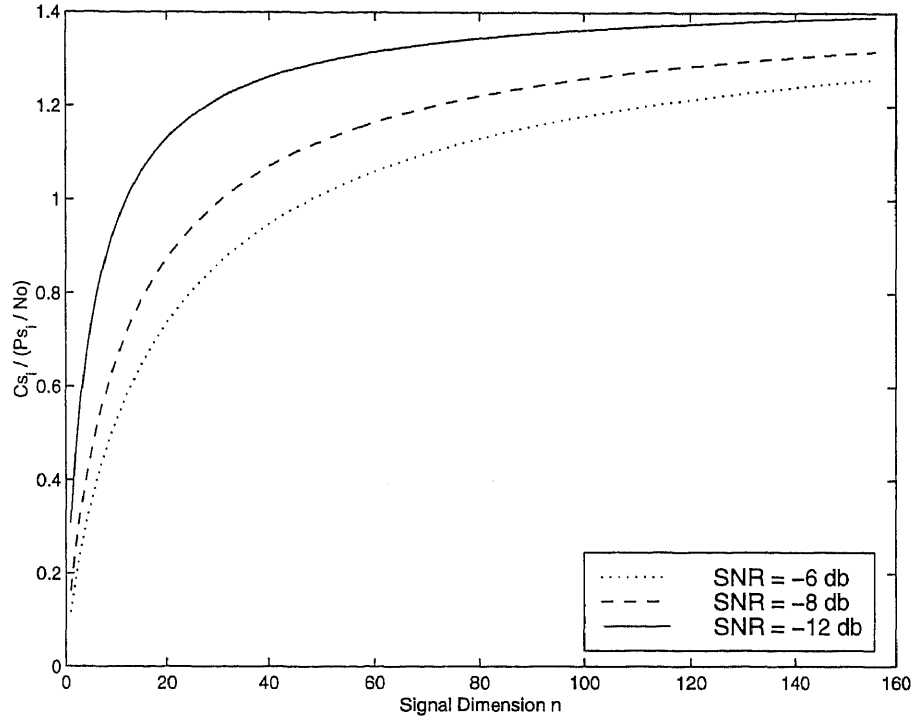


Figure 3.2 Capacity Gain Saturation as n Increases with Segmentation

segment when we plot the curve in Figure 3.2. One can see that the curve in Figure 3.2 is similar to that in Figure 3.1 if we scale the horizontal axes by 32.

When it comes to segmenting the watermark signal, we should take an additional factor into consideration. Here, we are not trading all the capacity for robustness (as is true for the nonsegmented case); we are ultimately interested in decreasing the probability of symbol decoding error. The curve in Figure 3.2 tells us how the capacity of the 32 segments saturates. This saturation tells us when we will not be able to decrease the probability of symbol decoding error by increasing the signal dimension. This prompts us to consult Shannon's capacity theorem [1] [2] [39] as we did in the previous chapter. Again, the essence of this theorem is that the noise, the available number of dimensions, and the available signal power set a limit only on the *rate* at which reliable communication is achieved; *not* on the *accuracy*. We can notice from the curve that as the SNR decreases (more noise is presented), the capacity saturation is reached with a smaller signal dimension. Considering only

the capacity versus the signal dimension may cause an increase in the probability of symbol decoding error. Consulting Shannon's capacity theorem, we would increase the signal dimension by making m smaller. This would result in more signaling power per segment (which is the way to decrease the probability of symbol decoding error with power-constrained signals). In fact, when we consider Shannon's theorem here, we trade the number encoded symbols for robustness (decreasing the probability of symbol decoding errors).

Let us differentiate between the two scenarios of increasing the segment dimension, n, here. The first scenario fixes the number of segments m (which means we carry the same number of symbols in the signal). With this scenario, we spread the signal power among more coefficients. As a result, we increase the total number of bits in the signal N. With the second scenario, based on Shannon's capacity theorem, we fix N and decrease m. Here by increasing n, we include more power per each segment, which decreases the probability of symbol decoding error.

3.2.3 Example

Starting from Equation (3.5), let us assume we want to achieve approximately 90% of the capacity saturation. That is, we want C_s to reach approximately 90% of the asymptotic value $1.44\frac{P_s}{N_0}$. Equation (3.5) will give

$$1.3\frac{P_s}{N_0} = \frac{n}{2}\log_2(1+\frac{2P_s}{nN_o}). \tag{3.6}$$

$$2.6\frac{P_s}{nN_0} = \log_2(1+\frac{2P_s}{nN_o}). \tag{3.7}$$

Substituting $\frac{P_s}{nN_0}$ by x, we get

$$2.6x = \log_2(1+2x). \tag{3.8}$$

Solving the above equation for x, we get the following approximation:

$$x = \frac{P_s}{nN_0} \approx 0.1, \tag{3.9}$$

which gives

$$n \approx \frac{10P_s}{N_0}. \tag{3.10}$$

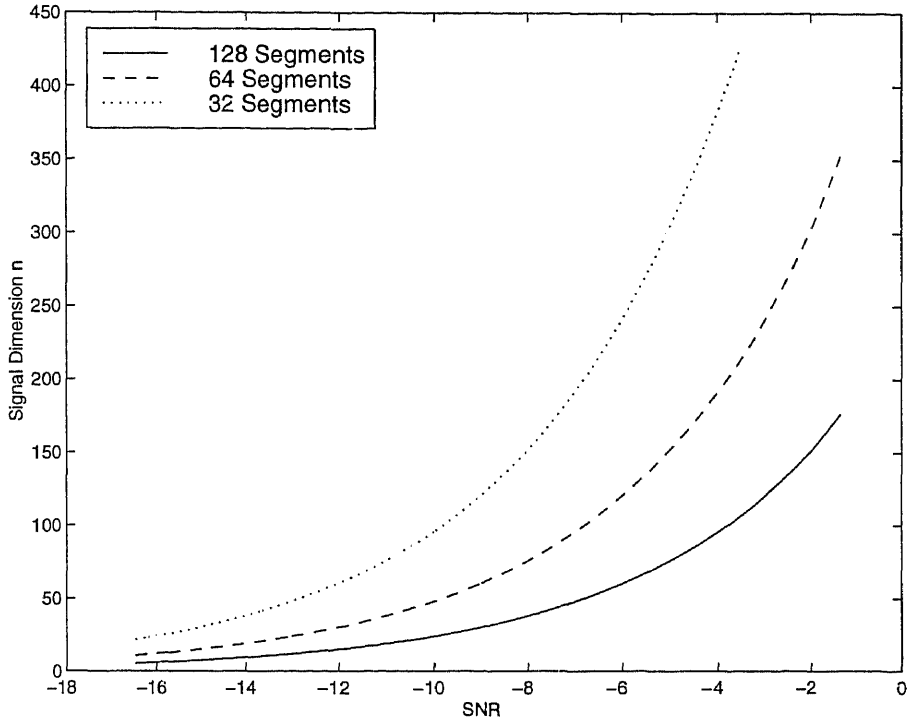


Figure 3.3 The Relation Between n and SNR for a Capacity Saturation of 90%

Using Equation (3.10), we can plot the signal dimension n versus the SNR for a different number of segments, as shown in Figure 3.3. Let us consider the case when the $SNR = -8$ db. If we segment the watermark signal to 128 segments, we get $n \approx 40$ (from the curve in Figure 3.3). This, in turn, gives a total signal dimension of about 5120. Now if the probability of symbol decoding error is high, we make m smaller. For $m = 64$, the curve gives us $n \approx 80$, which translates to $N \approx 5120$. For $m = 32$, the curve gives us $n \approx 160$, which translates to $N \approx 5120$. One can see how the need to achieve a lower probability of symbol decoding error leads to increasing the power per segment, which leads to increasing the segment dimension, while the entire watermark signal dimension, N, is fixed for the 90% achieved capacity. If we go back to Figure 3.1, for an SNR of -8 db, we find that achieving 90% of the capacity (1.3 on the vertical axis) requires $N \approx 5120$.

Now let us go back to the experimental case in Section 2.4 (where we embedded a 32-character string using 96 bits per segments), and compare it to the 32 segment

analytical case in Figure 3.3 above. In Chapter 2, we were interested in a PSNR of 30 db, which is in the range of $SNR_{wm} = -5$ db. In Figure 3.3, for an SNR of -5 db and 32 segments, we obtain $n \approx 320$ bits. This translates to $N = 320 \times 32 = 10240$ bits. This means we need to use 10 coefficients per each transformed block. In other words, to achieve 90% of the capacity using 32 segments at $SNR_{wm} = -5$ db, we need to utilize 10 bits per block. In this light, we want to know how much capacity we achieved using the 96 bits per segment (3 coefficients per block). If we plot a fourth curve in Figure 3.2 for an SNR of -5 db, and measure the capacity saturation at $n = 96$, we will find it to be about 1.14, which corresponds to almost 80% of the capacity. Thus, one can say that at $SNR_{wm} = -5$ db, to increase the achieved capacity from 80% to 90% we need to increase the entire watermark signal dimension from 3072 to 10240 bits, which requires increasing the number of coefficients per block from 3 to 10. This increase will require us to construct a signal set of size 320 bits per entry instead of 96 bits per entry. This will lead to increasing the computational complexity of the decoding process significantly.

3.3 Summary

This chapter showed that the increase in the signaling capacity of the watermark signal, as a result of increasing its dimension, is negligible for a large signal size. We showed that when dealing with a watermark signal that is one sequence of random variables (copyright protection signal or, with multiple signaling, when we decode the entire signal as one sequence), because we are interested in achieving high capacity in the presence of high noise (low SNR), capacity saturation is reached with only three coefficients per block for a 256×256 image using the DCT on 8×8 blocks.

With multiple signaling, when we decode each segment of the watermark signal separately, we have a different situation. Since we are not interested in decoding individual segments when SNR is very low (as with the nonsegmented case), increasing the signaling dimension may result in some significant increase in the achieved capacity. Also, decreasing the probability of symbol decoding error by

decreasing the number of segments m in the watermark signal results in increasing the signaling dimension as well.

Increasing the signaling dimension to increase the achieved capacity or to decrease the probability of symbol decoding error does not come without cost. Increasing the signal dimension means increasing the size of the maximum separable set of signals used in the correlation process. This, in turn, increases the computational complexity, since each extracted segment is correlated with all entries in the signal set. The next chapter addresses this tradeoff in more detail by presenting sequence decoding as a way to make the increase in computational complexity more manageable as the segment dimension, n, increases.

CHAPTER 4

SEQUENCE DETECTION TECHNIQUES FOR DIGITAL IMAGE WATERMARK SIGNALS

As explained in the previous chapter, in digital image watermarking a tradeoff between increasing the signal dimension and lower computational complexity is encountered. This chapter proceeds to explore possible methods to detect the watermark signal. We show how using sequence detection techniques, such as maximum a posteriori probability sequence decoding (MAPSD), can provide the same performance as conventionally used correlation detection while simplifying the detection process. As a result, we can increase the dimension of the watermark signal (to achieve higher capacity or to reduce the probability of symbol decoding error), and keep the increase in computational complexity manageable. We will also show that with MAPSD, we can have a measure of confidence level of the detected signal that is directly dependent on the watermark signal to noise ratio (SNR_{wm}).

This chapter proceeds as follows. In Section 4.1, we mention some characteristics of the watermark signal. In Section 4.2, we analyze optimum detection techniques. Section 4.3 covers the application of optimum detection techniques to the watermark signal. Section 4.4 presents some simulation results. Section 4.5 is a summary.

4.1 Special Characteristics of the Watermark Signal

In Chapter 2 we explained that because the watermark signal is power constrained, decoding the signal using correlation detection results in better performance than using error correction codes with bit-by-bit signaling. We also proposed in Chapter 2 an approach for embedding meaningful information in the watermark signal using multiple signaling. We showed that with the multiple signaling approach, we can detect the watermark signal in two ways. The first way divides the extracted signal into m segments and correlates each segment with all the entries in a maximum separable signal set. The entry associated with the highest correlation is selected and a corresponding symbol from the ASCII table is decoded. The result of this

segmentation is to carry a string of meaningful information symbols in the watermark signal, which can be utilized as an indication of image ownership. The second way correlates the entire extracted watermark signal with the entire original signal, the robustness of which is used to indicate the existence of the watermark signal. With this approach, the capability of making the watermark signal reveal a sequence of meaningful information symbols while keeping its randomness may protect ownership more strongly than the conventional way (which correlates the entire signal and can only provide a yes or no answer). Here, we use the same embedding technique used in Chapters 2 and 3 as follows: Apply the DCT to 8×8 blocks, utilize l coefficients per each transformed block, and when $l = 3$ the entire watermark signal dimension $N = 3072$.

In Chapter 3, we studied the bounds on the capacity of digital image watermarking. Based on this study, if the watermark signal is dealt with as one sequence of random variables (non segmented case), an increase in the signal dimension increases the achieved capacity if the signal dimension is small. As the signal dimension becomes larger, the capacity gain reaches a saturation level. On the other hand, if the watermark signal is segmented, the segment dimension may be increased either to achieve higher capacity, or to reduce the probability of symbol decoding error. In either case, increasing the segment dimension increases the computational complexity. This is because each extracted segment of the watermark signal has to be correlated with all the entries in the maximum separable signal set. Any small increase in the signal dimension results in a large increase in the computation needed to decide the symbol corresponding to the extracted segment. We provided an example where the watermark signal to noise ratio $SNR_{wm} = -8$ db. In this example, we showed that if we want to increase the achieved capacity from 80% to 90%, we have to increase the segment dimension from 96 bits to 320 bits. This causes the size of the signal set to increase from 64×96 to 64×320. As a result, the correlation process (for each decoded symbol) has to be performed 64 times on 320-bit signals instead of 96-bit signals. Thus, we need to find a watermark decoding process that allows us to increase the signal dimension, and at the same time face a manageable increase in computational complexity.

In this chapter, we study optimum detection methods in order to find the method that gives the lowest increase in computational complexity as the signal dimension increases. We base this study on the following differences between signaling over a spread spectrum communication channel and image watermark signaling. Over a communication channel, the received signal is not necessarily real; it can have imaginary (negative) frequencies, and it is a function of time. Correlation and matched filter detection via hardware are proper approaches. In digital image watermarking, the situation is different: the extracted signal consists of real numbers, it is not a function of time, and the decoding process is done using software implementation. Moreover, in digital image watermarking, we can specify the detection problem either as the need to include a string of symbols taken from a set of 64 symbols, as explained above, or as the need to detect the existence of a certain watermark signal in a given image. Thus, we need to study optimum detection methods specifically for the image watermarking case without strictly following spread-spectrum communication methods.

Although maximum a posteriori probability sequence decoding (MAPSD) and maximum likelihood sequence decoding (MLSD) are well known and are used in many coding applications [11], we would like to emphasize an important point with regard to applying them to digital image watermarking: each individual bit in the watermark signal can be treated as a signal by itself, and thus the entire watermark signal can be viewed as a sequence of binary signals. Also, unlike signaling over a communication channel, in digital image watermarking we do not have a demodulator that supplies the decoder with quantized values of the received signal. Unquantized values of the detected signal are obtainable in the watermarking case. As a result, MAPSD and MLSD offer the advantage of simplifying the detection process (since we will be detecting binary signals), and in the mean time, the estimation process bases its decision on the observation of the entire signal (sequence decoding). The result of this approach is to achieve the same performance as that with correlation detection (which performs both detection and estimation on the entire sequence simultaneously), but with a manageable increase in computational complexity as the

signal dimension increases. Also, we will show that with MAPSD, we can have a measure of confidence level of the detected signal that is proportional to the SNR_{wm}.

4.2 Optimum Detection Methods

Consider a digital image watermarking system where a part of the ASCII table (64 symbols) is mapped to a set of 64 maximum separable signals $S_j(j = 1, 2, ..., M)$, where $M = 64$ and each signal is of dimension (length) n bits. That is, the signal $S_j = (x_{j1}, x_{j2}, ..., x_{jn})$, where x_{jk} is the k^{th} bit of S_j. The entire watermark signal is divided into m equal segments such that the total number of bits in the watermark signal $N = m * n$, and each segment is taken from this set of 64 signals to form a string of meaningful information. The entire watermark signal of N bits (composed of m segments) is embedded in the image using some N coefficients in a transform domain. Binary 1 is embedded by adding Δ and binary 0 is embedded by adding $-\Delta$ to the selected coefficient. Thus, the i^{th} segment of the embedded watermark signal is $S_i = (x_{i1}, x_{i2}, ..., x_{in})$, where x_{ik} is the k^{th} component of S_i. x_{ik} is 0 or 1 if we refer to the maximum separable set of signals, and is $+\Delta$ or $-\Delta$ if we refer to the embedded signal itself. When the image encounters processing, attacks, or any sort of degradation, the resulting noise at the coefficients carrying the watermark signal may be modeled as AWGN [3] [34]. Thus, for each segment, the extracted signal R_i is given by

$$R_i = S_i + e, \tag{4.1}$$

where $R_i = (x_1^*, x_2^*, ..., x_n^*)$, and e is an AWGN vector.

We are looking for the optimum detection process that minimizes the probability of making an error given the n dimension vector R_i. This detection process will decide which of the 64 signals is embedded in this segment based on the extracted vector R_i.

We note that for a watermark signal that is a sequence of random variables, where the entire N bits of the signal are considered in the detection process (i.e., a copyright protection watermark signature or with multiple signaling when we decode the entire signal without segmentation), we face a similar problem. The optimum

detection method is expected to give a response to the original signal that is strong and robust when compared with the response to randomly generated sequences. Although the optimum detection analysis deals with detecting a signal in a set of maximum separable signals, we will show the applicability of the discussed detection methods for the detection of a non segmented copyright protection signal, as well. We also note that if the detection process has no access to the original image, as in some data hiding applications, the analysis in this chapter applies as well. The effect of considering the image itself as noise is an increase in variance (the noise power) of the modeled AWGN.

4.2.1 Correlation Detection

In this detection process, the projection between the extracted segment (with the additive noise), R_i, and the 64 signals in the maximum separable signal set is calculated using the following formula.

$$\rho_j = \frac{\sum_{k=1}^{n} x_k^* . x_{jk}}{\sqrt{\sum_{k=1}^{n} (x_k^*)^2}}, \tag{4.2}$$

where, as defined before, n is the signal dimension, x_{jk} is the k^{th} component of the S_j (which is $\pm\Delta$), and x_k^* is the k^{th} component of the extracted vector R_i.

Although this detection process is optimum in the sense that it minimizes the probability of making an error, it requires extensive computation that increases with the increase in the signal dimension n. The purpose here is to find an optimum detection process that does not require a significant increase in computational complexity as the signal dimension increases.

4.2.2 Matched Filter Detection

This detection process is similar to correlation detection, but it uses a bank of linear filters to represent the 64 maximum separable signals. The impulse response of the filter is $h_j(t) = S_j(T - t)$. The response of the filter $h_j(t)$ to the signal $S_j(t)$ is then the time-autocorrelation function. This detection process measures the time-autocorrelation function, which is an even function of t and has a peak at $t = T$. The filter associated with the signal that best matches the noisy signal will give

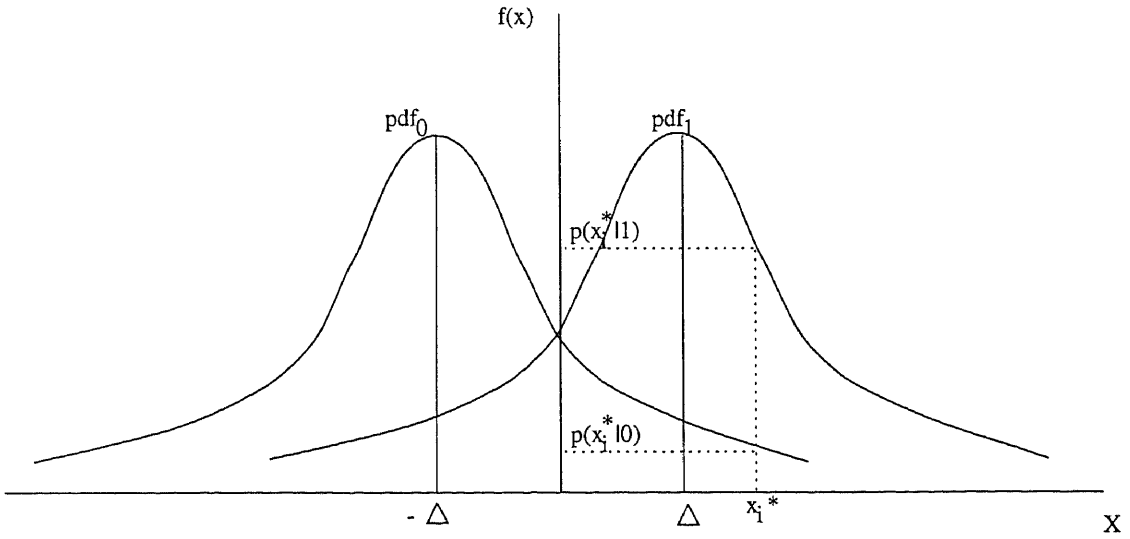


Figure 4.1 Two Hypotheses for Bit-by-bit Detection of the Watermark Signal

the highest autocorrelation at $t = T$. Clearly, this detection method does not suit our case because our digital watermark signal is not a function of time, and we are looking for detection methods best implemented via software, not by building a bank of linear filters.

4.2.3 Other Optimum Detection Methods for Bit-by-bit Signaling

Consider how the watermark signal is embedded in the image utilizing some selected coefficients in a transform domain. At the selected coefficients, we add $+\Delta$ for binary 1 and $-\Delta$ for binary 0. When decoding the watermark signal, one needs to overcome the effect of additive noise at the selected coefficients. That is, the detected signal is composed of $+\Delta + N_0$ for binary 1 and $-\Delta + N_0$ for binary 0, where N_0 is a Gaussian noise with a zero mean and a variance of σ^2. As Figure 4.1 demonstrates, when we decode a single bit from the extracted watermark signal, we encounter two hypotheses, h_0 and h_1, as a result of the additive noise. h_0 is based on the pdf of $-\Delta + N_0$, which we will refer to as pdf_0. h_1 is based on the pdf of $+\Delta + N_0$, which we will refer to as pdf_1. For the extracted value x_k^*, the conditional pdf's are given by

$$f(x_k^*|0) = pdf_0 = \frac{1}{\sqrt{2\pi}\sigma}.\exp\{-\frac{(x_k^* - m_0)^2}{2\sigma^2}\}, \tag{4.3}$$

and

$$f(x_k^*|1) = pdf_1 = \frac{1}{\sqrt{2\pi}\sigma}.\exp\{-\frac{(x_k^* - m_1)^2}{2\sigma^2}\}, \tag{4.4}$$

where m_0 is the mean of the Gaussian distribution of pdf_0 and is equal to $-\Delta$, m_1 is the mean of the Gaussian distribution of pdf_1 and is equal to $+\Delta$, and x_k^* is the extracted value of the watermark signal at the location carrying the k^{th} bit of the signal.

Now, if we use Bayes' theorem, we obtain the a posteriori probabilities $p(0|x_k^*)$ and $p(1|x_k^*)$. $p(0|x_k^*)$ is the probability that 0 is the encoded bit given x_k^*, which is given by

$$p(0|x_k^*) = \frac{f(x_k^*|0)p(0)}{f(x_k^*)}, \tag{4.5}$$

and $p(1|x_k^*)$ is the probability that 1 is the encoded bit given x_k^* is given by

$$p(1|x_k^*) = \frac{f(x_k^*|1)p(1)}{f(x_k^*)}, \tag{4.6}$$

where $f(x_k^*)$ is given by

$$f(x_k^*) = f(x_k^*|0)p(0) + f(x_k^*|1)p(1). \tag{4.7}$$

Substituting Equation (4.7) in Equations (4.5) and (4.6), and taking into consideration that $p(0) = p(1) = 1/2$, we get

$$p(0|x_k^*) = \frac{f(x_k^*|0)}{f(x_k^*|0) + f(x_k^*|1)}, \tag{4.8}$$

and

$$p(1|x_k^*) = \frac{f(x_k^*|1)}{f(x_k^*|0) + f(x_k^*|1)}. \tag{4.9}$$

Using Equations (4.3) and (4.4), we get

$$p(0|x_k^*) = \frac{1}{1 + \exp\{-\frac{(x_k^* - m_1)^2}{2\sigma^2} + \frac{(x_k^* - m_0)^2}{2\sigma^2}\}}, \tag{4.10}$$

and

$$p(1|x_k^*) = \frac{1}{1 + \exp\{-\frac{(x_k^* - m_0)^2}{2\sigma^2} + \frac{(x_k^* - m_1)^2}{2\sigma^2}\}}. \tag{4.11}$$

Substituting for $m_0 = -\Delta$ and $m_1 = +\Delta$, we get

$$p(0|x_k^*) = \frac{1}{1 + \exp\{2\Delta\frac{x_k^*}{\sigma^2}\}}, \tag{4.12}$$

and

$$p(1|x_k^*) = \frac{1}{1 + \exp\{-2\Delta\frac{x_k^*}{\sigma^2}\}}. \tag{4.13}$$

In Equations (4.12) and (4.13), σ^2 (the noise variance) and Δ (the added value of the watermark signal at the specified location) are known. x_k^* (the extracted value from the image) is the only variable, which is $\pm\Delta + N_0$.

If we divide Equation (4.8) by Equation (4.9), we obtain the following relation:

$$\frac{p(0|x_k^*)}{p(1|x_k^*)} = \frac{f(x_k^*|0)}{f(x_k^*|1)}. \tag{4.14}$$

Equation (4.14) implies that if $p(0|x_k^*) > p(1|x_k^*)$, then $f(x_k^*|0) > f(x_k^*|1)$ and vice versa. In other words, because $p(0) = p(1) = 1/2$, maximum a posteriori probability (MAP) decoding is equivalent to maximum likelihood (ML) decoding. That is, if we are to decode bit-by-bit, we decide 0 if $p(0|x_k^*) > p(1|x_k^*)$ and we decide 1 if $p(1|x_k^*) > p(0|x_k^*)$ or, equivalently, we decide 0 if $f(x_k^*|0) > f(x_k^*|1)$ and we decide 1 if $f(x_k^*|1) > f(x_k^*|0)$. As can be seen from Figure 4.1, it is also equivalent to decide 0 if the distance between x_k^* and $-\Delta$ (the mean of pdf_0) is less than the distance between x_k^* and $+\Delta$ (the mean of pdf_1), and decide 1 if the distance between x_k^* and $+\Delta$ is less than the distance between x_k^* and $-\Delta$. Thus, the optimum detector of bit-by-bit signaling can calculate the Euclidean distance between x_k^* and the mean of pdf_0 and pdf_1 and decode the bit that corresponds to the minimum distance or, equivalently, it can calculate the a posteriori probabilities $p(1|x_k^*)$ and $p(0|x_k^*)$ and decode the bit that corresponds to the MAP or, equivalently, it can calculate the conditional pdf's $f(x_k^*|1)$ and $f(x_k^*|0)$ and decode the bit that corresponds to the ML.

4.2.4 Sequence Detection

In Section 4.2.3, we demonstrated how optimum detection can be done in the case of bit-by-bit signaling. MAP, ML or minimum distance decisions minimize the probability of bit decoding error. Although these detection methods are known in digital

communication for any set of signals, we specifically showed them for the case of binary signals. We derived Equations (4.12) and (4.13), since we will use them for MAPSD.

Now, consider the case of detecting a watermark signal of length n bits. The optimum detector in this case is a sequence detector that bases its decision on the observation of the entire sequence (n extracted bits). Considering that we are observing a memoryless sequence of n bits, and taking into consideration that the noise is white and independent, the reader can refer to communications references such as [29], [39], and [40] for the derivation of sequence detection techniques. Keeping in mind that our watermark signal is a sequence of binary signals, the sequence detector may equivalently apply any of the following three detection methods:

1. Sum the a posteriori probabilities (of individual bits) $p(0|x_k^*)$ and $p(1|x_k^*)$ for each signal in the set given the extracted sequence of n bits. We will refer to this sum of the a posteriori probabilities as the MAP weight of the signal, and will use the notation $MAPW_S$, where S is the signal to be weighted. Equations (4.12) and (4.13) will be used. This sequence detection will decode the signal with the highest MAP weight. This is what we refer to as MAPSD.

2. Sum the conditional pdf's (of individual bits) $f(x_k^*|0)$ and $f(x_k^*|1)$ for each signal in the set given the extracted sequence of n bits. We will refer to this sum of the conditional pdf's as the ML weight of the signal, and will use the notation MLW_S, where S is the signal to be weighted. Equations (4.3) and (4.4) will be used. This sequence detection will decode the signal with the highest ML weight. This is what we refer to as MLSD.

3. Apply correlation detection as explained in Section 4.2.1. At the end of Section 4.3, we will show how the sequence detection approach can also be adopted for correlation detection.

In the remainder of this chapter, we compare the application of these three sequence detection techniques in digital image watermarking.

4.3 Applying Sequence Detection

This section explains how each of the three detection methods above can be implemented for digital image watermarking, and it demonstrates the amount of computation needed for each technique with a numerical example.

4.3.1 Sequence Detection Based on MAP Weight (MAPSD)

This decoding technique applies the following steps:

1. For the extracted signal of length n $(x_1^*, x_2^*, ...x_n^*)$, find the a posteriori probabilities of 0 and 1 of each value x_k^* using Equations (4.12) and (4.13), and construct an a posteriori probabilities table of size $n * 2$.

2. For each entry in the signal set, find the weight (the sum of the a posteriori probabilities) associated with this entry using the a posteriori probabilities table.

3. Decode the signal with the highest weight.

With this method, Equations (4.12) and (4.13) are applied n times in step 1. We perform n summations for each of the 64 signals in step 2.

Example: Suppose we have a signal set of 4 signals, each having 4 bits. A possible set of maximum separable signals is $S_1 = 0000, S_2 = 0101, S_3 = 1010$, and $S_4 = 1111$ (see Section 2.1). If the extracted values from the image are $x_1^* = -1.257, x_2^* = 0.333, x_3^* = -0.756$, and $x_4^* = 0.9567$, and if $\Delta = 6$ and $\sigma^2 = 4$, by applying Equations (4.12) and (4.13) four times for the four values x_k^*, we can construct the a posteriori probabilities shown in Table 4.1. These a posteriori probabilities are used to calculate the MAP weight of each signal, as follows:

The weight of $S_1 = 0.977 + 0.27 + 0.9 + 0.05 = 2.197$

The weight of $S_2 = 0.977 + 0.73 + 0.9 + 0.95 = 3.557$

The weight of $S_3 = 0.023 + 0.27 + 0.1 + 0.05 = 0.443$

The weight of $S_4 = 0.023 + 0.73 + 0.1 + 0.95 = 1.803$

Since S_2 has the highest weight, it is selected as the signal to be decoded.

Table 4.1 A Posteriori Probabilities when $\Delta = 6$ and $\sigma = 2$

x_k^*	-1.257	0.333	-0.756	0.9567
a posteriori probabilities of 0	0.977	0.27	0.9	0.05
a posteriori probabilities of 1	0.023	0.73	0.1	0.95

Using MAP weight, the bulk of computation occurs in applying Equations (4.12) and (4.13) n times, where n is the signal dimension. The actual implementation can be simplified by applying either of the equations and taking the complement. For example, we can apply Equation (4.12) to find $p(0|x_k^*)$, and we can find $p(1|x_k^*)$ from the following equation:

$$p(0|x_k^*) + p(1|x_k^*) = 1. \tag{4.15}$$

When the signal length is very large, and the size of the signal set is also large (which is needed for achieving a higher capacity), we can further decrease the computational complexity if we apply this approach vertically instead of horizontally, as follows:

1. Extract a value of one bit and find the a posteriori probability of 0 and 1 for this bit.

2. Update the weight of each signal based on the calculated probability.

That is, instead of calculating the a posteriori probability table first and calculating the weight of each signal (moving horizontally), we extract the value x_k^*, find $p(0|x_k^*)$ and $p(1|x_k^*)$, and scan each signal for its k bit (vertically) to update the weight of the signal. The advantages of this vertical scanning are

1. The elimination of the need for storing the a posteriori probabilities table.

2. the possibility of eliminating signals with low weight early in the calculation process.

Consider the example given above: a signal such as S_3 (having the lowest weight) could be eliminated from the calculation process before extracting the fourth bit. This is because an a posteriori probability is less than 1, and because the weight

of S_3 after the third bit is $0.023 + 0.27 + 0.1 = 0.393$, while the weight of S_2 is $0.977 + 0.73 + 0.9 = 2.607$. Even if the a posteriori probability of the fourth bit is the maximum for S_3, the weight of S_3, at the end, will be 1.393, which does not reach the weight of S_2 after decoding the third bit. This elimination process can decrease the computational complexity further.

We would like to note that for a watermark that is a sequence of random variables, where we detect the entire embedded signal without segmentation, this probabilistic measure is a good indication of confidence level. The highest MAP weight an n-bit signal can have is n. In the previous example, S_2 had a weight of 3.557 out of a maximum of 4. For someone attempting an attack by using randomly generated sequences of length n bits, the MAP weight approaches $n/2$ as n increases. This can be shown as follows. Let us refer to the randomly generated sequence (by the attacker) as $S_{rv} = (z_1, z_2, ..., z_n)$, which is a sequence of random variables with a zero mean and takes the magnitude $+\Delta$ and $-\Delta$ with equal probabilities. The MAP weight $MAPW_{rv}$ of this sequence is given by

$$MAPW_{rv} = \sum_{k=1}^{n} p(z_k|x_k^*), \tag{4.16}$$

where x_k^*'s are the extracted values.

Because the attacker has no knowledge of the original sequence, z_k is independent of x_k^*. That is,

$$MAPW_{rv} = \sum_{k=1}^{n} p(z_k) = \sum_{k=1}^{n} \frac{1}{2} = \frac{n}{2}. \tag{4.17}$$

For the original sequence $S = (x_1, x_2, ..., x_n)$, we can find two important limits for the MAP weight. The first limit is reached when the SNR of the watermark signal goes to infinity and the second limit is reached when the SNR goes to zero. The MAP weight of the original signal $MAPW_S$ is given by

$$MAPW_S = \sum_{k=1}^{n} p(x_k|x_k^*), \tag{4.18}$$

where $p(x_k|x_k^*)$ is given from Equations (4.12) and (4.13), which can be written as follows:

$$p(0|x_k^*) = \frac{1}{1 + \exp\{2\Delta\frac{x_k^*}{\sigma^2}\}} = \frac{1}{1 + \exp\{\frac{2\Delta(-\Delta+N_o)}{\sigma^2}\}}, \tag{4.19}$$

and

$$p(1|x_k^*) = \frac{1}{1+\exp\{-2\Delta\frac{x_k^*}{\sigma^2}\}} = \frac{1}{1+\exp\{\frac{-2\Delta(\Delta+N_o)}{\sigma^2}\}}. \tag{4.20}$$

When SNR goes to infinity (that is N_o goes to zero), and σ^2 goes to zero, the a posteriori probabilities in Equations (4.19) and (4.20) will go to 1, and the summation in Equation (4.18) will go to n. On the other hand, when SNR goes to zero, the a posteriori probabilities in Equations (4.19) and (4.20) will then go to 1/2, and the summation in Equation (4.18) will go to $n/2$. In other words, as the noise variance approaches infinity, the MAP weight of the original signal approaches $\frac{n}{2}$, which is that of a randomly generated sequence, and as the noise variance approaches zero the MAP weight approaches n, which is the highest MAP weight possible. Thus, we can measure the robustness of some embedded signal based on where its MAP weight falls on the $(\frac{n}{2} \rightarrow n)$ interval.

4.3.2 Sequence Detection Based on ML Weight (MLSD)

Although this detection process is equivalent to sequence detection based on the MAP weight, there are some factors that make it less desirable. These factors are listed below:

- Equations (4.12) and (4.13) are simpler than Equations (4.3) and (4.4).
- The values of the conditional pdf's $f(x_k^*|0)$ and $f(x_k^*|1)$ obtained from using Equations (4.3) and (4.4) are very small compared with the values of the a posteriori probabilities obtained from Equations (4.12) and (4.13). ML weight may require a higher precision calculation. The simulation at the end of this chapter provides an indication of the required precision.
- $f(x_k^*|0)$ and $f(x_k^*|1)$ do not sum to 1 and the computational complexity can not be simplified as with MAP weight using Equation (4.15).

Although some communication references consider MAPSD and MLSD to be the same because $p(0) = p(1) = 1/2$, we prefer to distinguish between the two in the detection of the watermark signal. In MAPSD, the decision is based on the sum of

the a posteriori probabilities; whereas in MLSD, the decision is based on the sum of the conditional probabilities.

4.3.3 Sequence Detection Based on Correlation Weight

For the same example, with $\Delta = 6$, when using conventional correlation detection, we apply the following steps:

1. Obtain the representation of the given binary signal set by substituting 0 with $-\Delta$ and 1 with Δ, upon which we obtain the following set: $S_1 = (-6, -6, -6, -6), S_2 = (-6, 6, -6, 6), S_3 = (6, -6, 6, -6)$, and $S_4 = (6, 6, 6, 6)$.

2. For the extracted values, x_k^*, and the representation signal values, x_{jk}, we apply Equation (4.2) four times to obtain the correlation coefficient (projection) of the extracted noisy signal with each signal in the maximum separable set.

3. Select the entry with the highest correlation coefficient.

We would like to note some advantages when choosing MAPSD over the correlation detection method explained so far. Consider embedding the digital image watermark signal using an adaptive technique, where the value of Δ is selected depending on the characteristics of the coefficient carrying the bit. When applying correlation detection, we need to obtain the representation of the binary signals. This requires calculating the values of $\pm\Delta$ for the entire signal, and constructing the representation accordingly. Also, correlation has to be calculated using the entire signal. This is because Δ can take different values; so correlation has to be decided based on the entire signal. With MAPSD, the situation differs: the detection process is performed for bit-by-bit signaling, and hence we can calculate Δ for one bit at a time, extract one bit at a time, find the a posteriori probability for one bit at a time, and update the a posteriori probabilities weight based on one bit at a time. Moreover, regardless of the value Δ takes, the a posteriori probability is bounded by 1, which allows us to eliminate some signals without having to wait until we extract the entire signal. Also, with MAPSD, we only need to keep the maximum separable signal set in binary format (which does not require much memory space). The representation of

the binary signals (in floating points), on the other hand, requires higher precision for the values of Δ and occupies a considerable memory space that increases as the signal dimension n increases.

From this comparison, we can say that with MAPSD, the correlation process is replaced by applying either Equation (4.12) or Equation (4.13), one at a time, n times, and Equation (4.2) is replaced by a simple summation of the a posteriori probabilities, which is also done one at a time.

At this point, we come to a question: Knowing how MAPSD is implemented, can we apply sequence detection using correlation? In other words, can we come up with correlation sequence detection (CORSD) that implements Equation (4.2) while performing bit-by-bit detection? The answer is yes. In Equation (4.2), the denominator can be ignored since it is a measure of the extracted signal power [40]. The term $\sum_{k=1}^{n} x_k^*.x_{jk}$ can be considered a correlation weight, and correlation then can be implemented for sequence detection as follows:

1. Extract a value of one bit x_k^* and find $x_k^*.x_{jk}$.

2. Update the correlation weight of each signal based on the calculated $x_k^*.x_{jk}$.

Notice that because the term $x_k^*.x_{jk}$ can take either positive or negative values, the capability of eliminating signals before completing the extraction of the entire signal (as can be done for MAPSD) can not be applied here. The simulation in Section 4.4 shows how correlation can take negative values, while MAP and ML weights are always positive.

Now, for CORSD, let us consider the limits for a random variable sequence (generated by an attacker) and for the original embedded signal as we did for MAPSD. For a generated random variable sequence, the correlation weight $CORW_{rv}$, is given by the following summation:

$$CORW_{rv} = \sum_{k=1}^{n} z_k.x_k^*, \tag{4.21}$$

where z_k is the k^{th} component of the generated random sequence, and x_k^* is the given k^{th} component of the extracted signal. We will calculate the limit here by showing

the expected value of $CORW_{rv}$, which can be found from Equation (4.21) as follows:

$$\begin{aligned} E[CORW_{rv}] &= E[\sum_{k=1}^{n} z_k.x_k^*] = \sum_{k=1}^{n} E[z_k.x_k^*] \\ &= \sum_{k=1}^{n} \Delta.p(z_k = \Delta).x_k^* - \Delta.p(z_k = -\Delta).x_k^*. \\ &= \sum_{k=1}^{n} \Delta.x_k^*\{p(z_k = \Delta) - p(z_k = -\Delta)\}. \end{aligned} \tag{4.22}$$

Since $p(z_k = \Delta) = p(z_k = -\Delta) = \frac{1}{2}$, $E[CORW_{rv}] = 0$. This can also be shown in the simulations obtained in the next section. See the correlation curves in Figure 4.2 and Figure 4.3.

For the original signal, calculating the expected values of the limits of $CORW_S$ (as SNR_{wm} goes to zero or infinity) is not possible. Consider finding the expected value of $CORW_S$ as follows:

$$\begin{aligned} E[CORW_S] &= E[\sum_{k=1}^{n} x_k.x_k^*] \\ &= \sum_{k=1}^{n} E[x_k.(x_k + N_0)] \\ &= \sum_{k=1}^{n} E[x_k^2] + E[x_k.N_o]. \end{aligned} \tag{4.23}$$

Since the noise is independent from the signal, and the expected value of the noise is zero, the term $E[x_k.N_o]$ is zero. The expected value of the correlation weight is then

$$E[CORW_S] = \sum_{k=1}^{n} E[x_k^2] = n\Delta^2, \tag{4.24}$$

which is not a function of the SNR.

The difference between the MAPW and CORW can be be explained physically as follows. For MAPW, as SNR_{wm} decreases, the weight of the original signal decreases to approach that of randomly generated sequences. On the other hand, for CORW, as SNR_{wm} decreases, the weight of the original signal is fixed, and the variance of the weight of the randomly generated sequences increases, making the CORW approach that of the original signal. The fact that the MAPW of the original signal is dependent on the SNR_{wm} makes this weight a good measure of the confidence level of the signal.

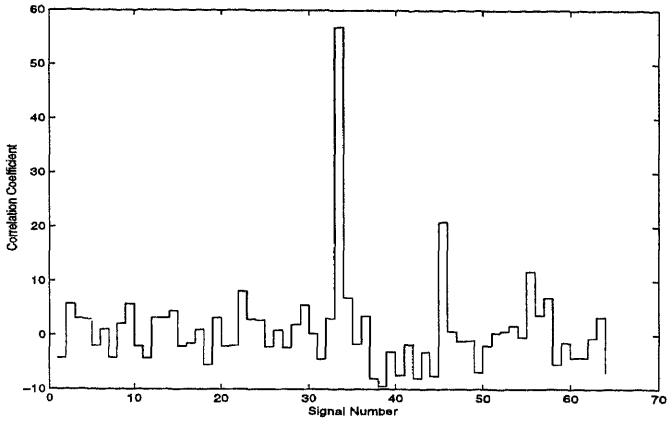
4.4 Simulation

In this section, we demonstrate by simulation how all three optimum detection techniques give the same performance for both non segmented and segmented cases. Keep in mind that with the non segmented case, we compare randomly generated sequences of length N bits with the entire original sequence (also of length N bits). With the segmented case, we use a maximum separable set of signals to decode the symbol corresponding to each embedded segment. We simulated correlation decoding, sequence decoding based on MAP weight, and sequence decoding based on ML weight, with the detection process facing a noise variance $\sigma = 8$, and the watermark added value $\Delta = 6$. This gives a watermark SNR of about -2.5 db. The simulation for the first case shows the behavior of optimum detection when comparing 64 randomly generated binary sequences (each 512 bits long). The simulation for the second case shows the behavior of optimum detection when the decoding process extracts an embedded segment of the signal and selects one of 64 symbols in a maximum separable signal set of length 31 bits per signal.

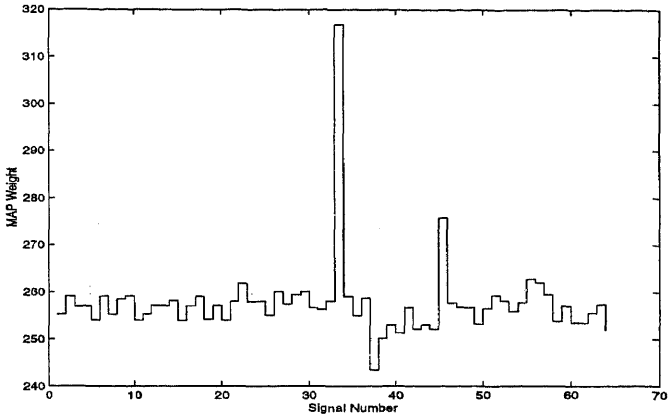
Figure 4.2 shows the first case. Figure 4.2(a) shows correlation decoding using Equation (4.2), Figure 4.2(b) shows sequence decoding based on MAP weight, and Figure 4.2(c) shows sequence decoding based on ML weight. In all three curves, the 33rd sequence is the original one.

Figure 4.3 shows the second case of 64 maximum separable signals, each 31 bits long. Figure 4.3(a) shows correlation decoding using Equation (4.2), Figure 4.3(b) shows sequence decoding based on MAP weight, and Figure 4.3(c) shows sequence decoding based on ML weight. In all three curves, the 33rd signal corresponds to the embedded symbol. The original signal is decoded correctly with all three methods. It can be seen from the curve that the 33rd signal has the highest weight in all three cases. Figure 4.4 shows the probability of signal decoding error as a function of the noise variance σ for MLSD and correlation detection. In both cases a set of 64 maximum separable signals of 31 bits per signal is used. Naturally, as σ increases, the probability of symbol decoding error increases.

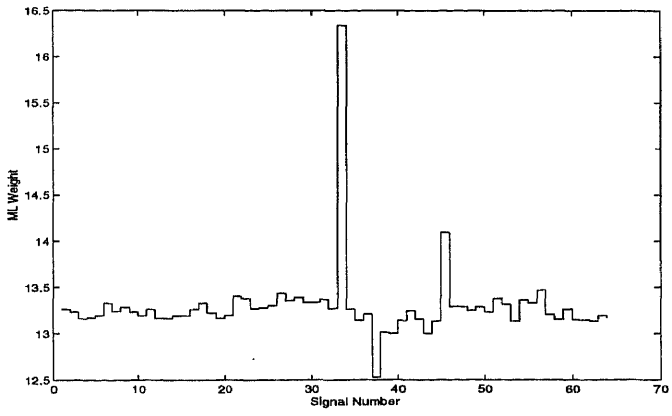
One can see from the two curves the equivalence of the two decoding methods. The reader can link this to Chapter 2, where we explained how the signal dimension



(a) Correlation

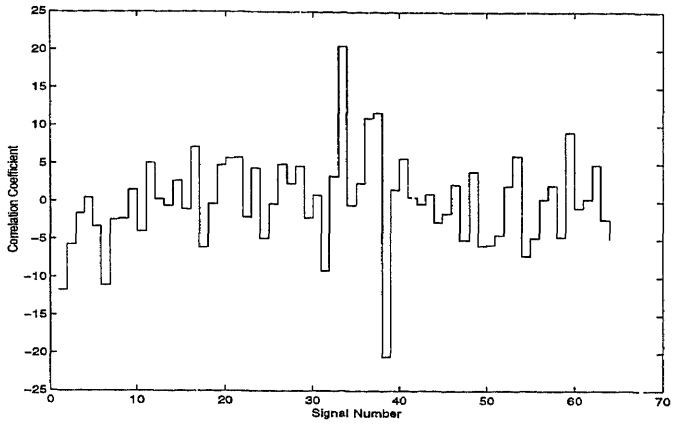


(b) MAPSD

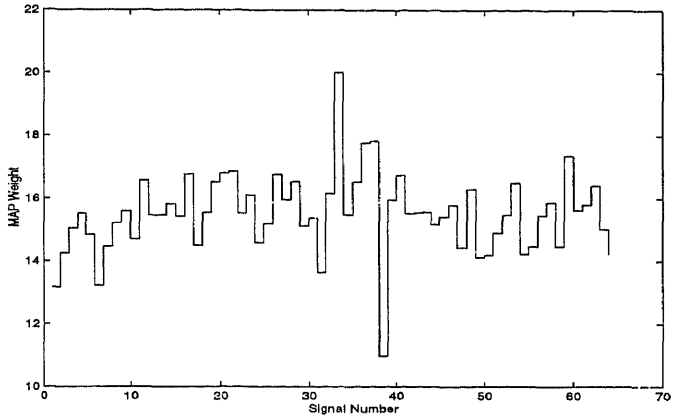


(c) MLSD

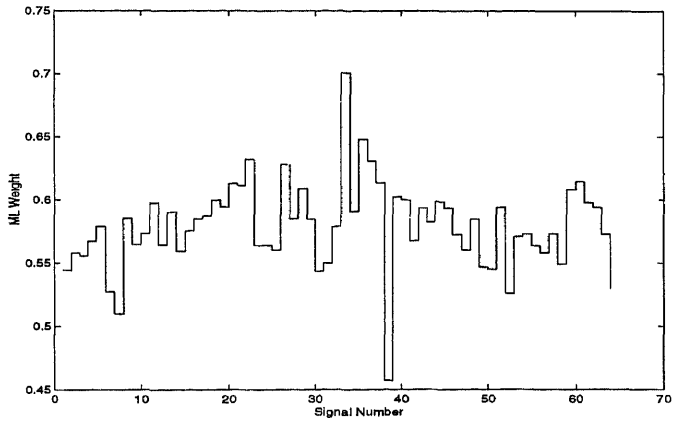
Figure 4.2 Detection of 64 Random Sequences of 512 Bits Per Signal. The 33rd Sequence is the Embedded One.



(a) Correlation

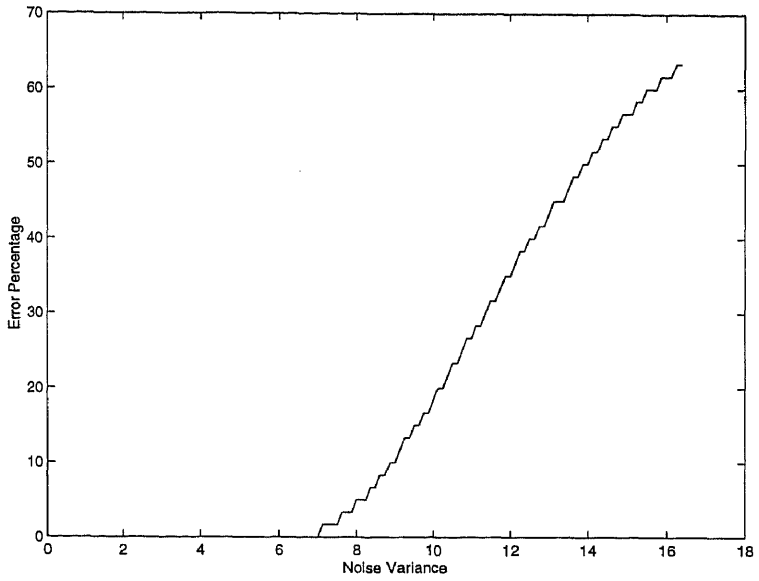


(b) MAPSD

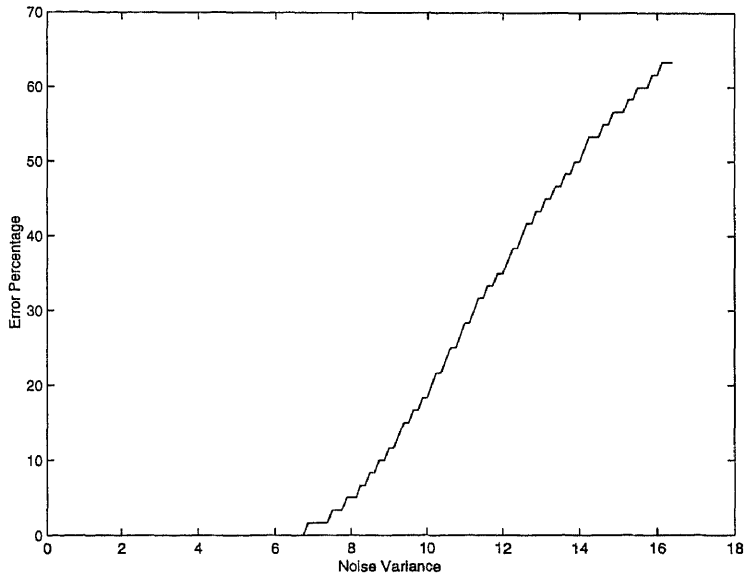


(c) MLSD

Figure 4.3 Detection of 64 Maximum Separable Signals of 31 Bits Per Signal. The 33rd Signal Corresponds to the Embedded Symbol.



(a) Correlation



(b) MAPSD

Figure 4.4 Probability of Signal Decoding Error as a Function of the Noise Variance. A set of 64 Maximum Separable Signals of 31 Bits Per Signal is Used.

n is selected according to Shannon's coding theorem. n should be increased when SNR_{wm} decreases (higher noise variance). As n (the number of bits per segment) increases, m (the number of segments in the watermark signal) decreases. This leads to including fewer symbols in the watermark signal or, in other words, a trade-off between the amount of information and robustness.

4.5 Summary

In this chapter we studied possible optimum detection techniques that can be applied for the detection of the digital image watermark signal. We showed how MAPSD, MLSD, and CORSD can be applied to detecting the watermark signal in both the segmented and the non segmented cases. These optimum detection techniques can be used with copyright protection and with information-symbol embedding applications. These sequence detection techniques offer the advantage of simplifying the detection process (bit-by-bit detection) and, at the same time, the decision is based on the estimation of the entire sequence. This makes the increase in computational complexity in the decoding process (which is due to the increase in the signal dimension) more manageable, which ultimately can enable us to achieve higher capacity and/or decrease the probability of symbol decoding error when the watermark signal is segmented. We showed the advantages of using MAPSD. We also showed that using the a posteriori probability weight has the following advantages over CORSD:

1. The possibility of eliminating signals with low weight early in the detection process, and

2. Having an upper and lower limit of the MAPW of a given signal, which is directly related to the SNR_{wm}. This offers the capability of measuring the confidence level based on where the signal's MAPW falls on the $(n \rightarrow \frac{n}{2})$ interval.

CHAPTER 5

2-D INTERLEAVING FOR 2-D SIGNALS

This chapter starts the second part of this dissertation, which addresses the robustness aspect of the watermark signal. As mentioned in Chapter 1, we base this work on the fact that the watermark signal differs from a communication-channel signal in that the layout of the watermark signal with respect to a still image is in 2-D. With communication channels, interleaving techniques are often used to increase the robustness of the transmitted signal against burst errors. With digital image watermarking, interleaving the watermark signal should increase the signal's robustness against intentional or unintentional distortions that cause part of the signal to be entirely destroyed. Interleaving techniques used with communication-channel signals (e.g. block interleaving), deal with 1-D signals and with 1-D bursts of error. For example, in a fading-channel application, if the channel fades for a period of time, the received signal can suffer from a burst error. As functions of time, both the signal and the burst of error are considered 1-D. On the other hand, the layout of our digital image watermark signal in a still image or a single frame in a video sequence is in 2-D. With respect to an entire video sequence, the signal layout is in 3-D. Intentional attacks are likely to destroy the signal in certain areas of the image and have little effect on other areas, which can result in a 2-D burst of error. Also, quantization noise can be high in certain blocks of the image and minimal in other blocks. In a video sequence, it is possible to lose the watermark signal in parts of some frames, or lose the watermark signal in a group of entire frames in the sequence. This raises the need to have an interleaving technique that can interleave a signal with a layout in 2-D or 3-D, to increase its robustness against 2-D or 3-D error bursts. This part of the dissertation presents a new multidimensional interleaving technique and shows its application to the watermark signal.

Throughout this chapter and Chapters 6 and 7, we will present multi-dimensional interleaving as a technique that interleaves a signal composed of a group of symbols. This presentation will explain a general case—not defining the symbol. In Chapters 8 and 9, where we apply this interleaving technique to

the watermark signal, we will emphasize what we mentioned in Chapter 4, that the watermark signal detection process is different from detecting a signal over a communication channel. The fact that with the watermark signal the decoder has access to unquantized values of individual bits is utilized to specify how this interleaving technique should be applied to the watermark signal. We will see that when we apply this interleaving technique to the watermark signal, three rules have to be followed that are different from those followed with nonpower-constrained signals. With nonpower-constrained signals, we adhere to the following:

1. Cluster a group of bits corresponding to one symbol in one area.
2. Use error correction codes. (With most burst error cases, RS codes are used).
3. Perform interleaving on a symbol-by-symbol basis.

Because the watermark signal is power constrained, however, and because we can have access to unquantized values of each bit, the above rules are reversed as follows:

1. Deal with each bit in the signal separately.
2. Add the redundancy needed to overcome the noise introduced to the watermark signal in the form of increasing the signaling dimension rather than using error correction codes.
3. Perform interleaving on a bit-by-bit basis.

The work in the second part of the dissertation proceeds as follows. In this chapter we give a tutorial for the new interleaving technique. In Chapter 6 we provide some theoretical analysis of the technique and explain an implementation method based on a sliding-window technique. In Chapter 7 we explain a different implementation method based on a successive partitioning approach and we give special emphasis to the 3-D case. In Chapter 8 we apply the 2-D version of this technique to the 2-D watermark signal embedded in a still image, and show how the robustness of the signal is increased. In Chapter 9 we apply the 3-D version of this technique to the 3-D watermark signal embedded in a video sequence.

This chapter will proceed as follows. In Section 5.1, we explain some characteristics of signals that have a 2-D layout. In Section 5.2, we present the well known block interleaving used in communications applications as a 1-D technique. In Section 5.3 we introduce interleaving for 2-D arrays. In Section 5.4 we give a tutorial comparison between applying 1-D interleaving techniques to 2-D applications, and applying the proposed 2-D algorithm. Section 5.5 is a summary.

5.1 Characteristics of Signals in a 2-D Layout

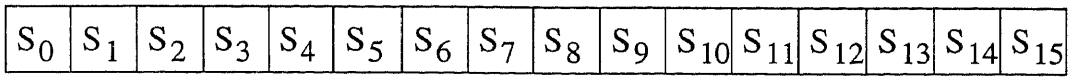
Imagine an application (like our watermark signal), where the signal detector has access to all the encoded data at once, and the signal layout is in 2-D. Thus, we can say that the signal layout forms a 2-D array of symbols. In this 2-D array, errors occur in bursts that stretch across the horizontal and vertical axes, corrupting a cluster of symbols in the array. This can occur in the watermark signal when an attacker tries to remove certain parts of the current watermark signal to force his own signature instead. Because it is easier to remove the signal from some transformed blocks than others, certain areas (clusters of 1 or more blocks) of the watermark signal can be corrupted. Our intension here is to introduce an interleaving technique that fully interleaves the encoded sequence of symbols for this type of error. For communication channel applications, and using linear codes, a well known technique (block interleaving) is used to overcome 1-D bursts of error. In this tutorial chapter, we will use simple examples to compare the performance of this newly developed 2-D interleaving technique with the performance of block interleaving when applied to this 2-D application.

This technique addresses the following question: For a given signal (block of symbols), knowing that the decoding device has access to all the data at once, and that this data is going to be distributed across a 2-D array, and knowing that the error bursts stretch across both vertical and horizontal axes, what is the best interleaving strategy for the encoding process to follow? In other words, starting from well known block interleaving, which is a 1-D interleaving technique that spreads *consecutive* errors across the interleaved block, can we come up with a 2-D inter-

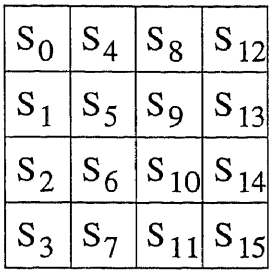
leaving technique that spreads *neighboring* errors across the interleaved sequence? "Neighboring" means that errors are adjacent across both vertical and horizontal axes (that is, a subset of symbols in the array is damaged). In the next section we will show how block interleaving can be achieved by visualizing a 1-D interleaving array and utilizing less memory space than that needed by the $n \times n$ interleaving array conventionally used in block interleaving. In Section 5.3 we will use this analysis to develop a 2-D interleaving technique for 2-D arrays that is equivalent to block interleaving for 1-D arrays.

5.2 Block Interleaving as a 1-D Interleaving Technique

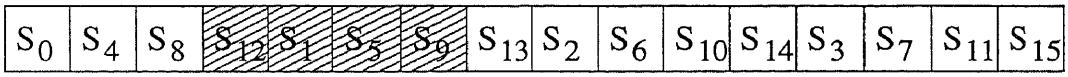
Suppose we have a block of length l^2 symbols, which is a 1-D array, and we want to apply block interleaving to it. Interleaving can be obtained by using a 2-D memory array of size $l \times l$ symbols [21] [37] [39]. Symbols are written to the memory array column-by-column and read row-by-row (or vice versa). In this case, the *interleaving degree*, known as λ, is equal to l. If the linear code used is capable of correcting a single burst error of length t symbols or less in a block of length l, the interleaved code will correct any single burst of length λl or less. Figure 5.1 demonstrates this technique when $l = 4$ and $t = 1$; that is, the sequence to be interleaved is a 1-D array of 16 symbols. Figure 5.1(a) shows the original sequence, Figure 5.1(b) shows the 4×4 interleaving array used, and Figure 5.1(c) shows the resulting interleaved sequence. If four consecutive symbols are in error, as shown by the shaded areas in Figure 5.1(c), the de-interleaved array will have these errors spread out, as shown in Figure 5.1(d), where each block of four symbols contains only one symbol in error. If we use a linear code capable of correcting one symbol in a four-symbol block, we will succeed to recover the original data. Without interleaving, the same code will fail to recover the original data. Notice that if we divide the de-interleaved array into four equal segments, any four consecutive symbols in the interleaved array are spread in the de-interleaved array such that one symbol falls in each segment, which ensures the desired correction capability.



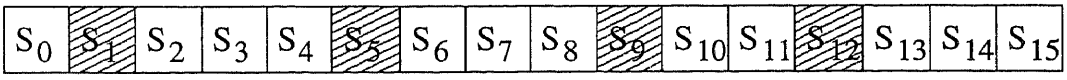
(a) Original Sequence



(b) Interleaving Array



(c) Interleaved Sequence



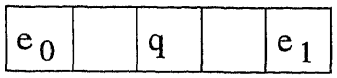
(d) De-interleaved Sequnce

Figure 5.1 1-D Block Interleaving Using a 2-D Interleaving Array

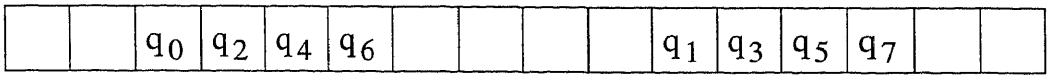
To apply the same concept to a 2-D array of symbols, one may think of utilizing a 3-D or 4-D interleaving array (which is hard to visualize) and find an algorithm to read from and write to this interleaving array such that maximum interleaving is achieved. This algorithm may require a large memory space and extensive computation. Another approach is to interleave 1-D arrays using 1-D interleaving arrays and utilize as little memory space and computation power as possible, and adapt this method for 2-D arrays. Figure 5.2 shows how this 1-D interleaving is done for the case of $l = 4$ (16 symbols), explained above. Figure 5.2(a) shows a 1-D interleaving array of five symbols. We will name this interleaving array I. In I, we will call the location at the center q and the end locations e_0 and e_1. Figure 5.2(b) shows an empty array of 16 symbols with eight locations, q_0 to q_7, marked at the positions shown. We will name this array R. Imagine that we write the first two symbols of the original sequence (shown in Figure 5.1(a)) to e_0 and e_1 in I, locate I such that the center q is at location q_0 in array R, and write the two symbols from e_0 and e_1 to their corresponding locations in array R. These will be the first and fifth locations in R. Afterwards, we write the next two symbols from the original array to e_0 and e_1, move I such that the center q is at location q_1 in array R, and write from e_0 and e_1 to their corresponding locations in R. These will be the ninth and thirteenth locations. If we repeat these steps until we cover the eight marked locations in R, we will end up with symbols written to R in the same sequence as in Figure 5.1(c). One can see that, in general, the length of I is $1 + \lambda$, where λ is the interleaving degree. Also, the number of locations that need to be marked for q to follow is $l^2/2$.

To apply this method practically (in real-time), aside from the original array and the interleaved array, we will need an array of size $l^2/2$ memory locations to contain the indexing of the marked locations, q_i's, with respect to R. We will name this array the q array. For the example of $l = 4$ explained above, the q array will contain the indexes 2, 10, 3, 11, 4, 12, 5, and 13 respectively.

Now comes the question of how to obtain the locations q_0 to q_7. The next section will show how a simple iterative algorithm can decide the locations, q_i's, for different 2-D array sizes. We will show the convenience of applying this concept for



(a) 1-D Interleaving Array I



(b) Positions of q in the Interleaved Array R

Figure 5.2 1-D Block Interleaving Using a 1-D Interleaving Array

2-D interleaving, since it does not require utilizing a complicated interleaving array or a large memory space, and it can be implemented in real time.

5.3 Interleaving for 2-D Arrays

In this section, interleaving for 2-D arrays is explained for $l \times l$ arrays, where l is a power of 2. The interleaving technique utilizes a 2-D interleaving array, which we will refer to as I. The dimensionality of I is $p \times p$ symbols, with the four corner symbols e_0, e_1, e_2, and e_3 arranged as shown in Figure 5.3. p is decided by l through the following relation:

$$p = l/2 + 1, \tag{5.1}$$

and the center of I is **in** from the four corners by q' symbols, where q' is given by

$$q' = p - median(p). \tag{5.2}$$

The locations of q, denoted as q_i's, form another 2-D array, which is referred to as the q array. The q array is a subset of the 2-D array referred to as R (which will carry the interleaved symbols). The q array has a dimension of $\frac{l}{2} \times \frac{l}{2}$ and is located q' symbols **in** from the edges of the R array. The application of this algorithm is iterative; that is, to interleave an $l \times l$ array, we obtain the interleaved $\frac{l}{2} \times \frac{l}{2}$ array first and use its sequencing as the q array sequencing. The following cases explain this interleaving technique. The reader can verify from these cases that the

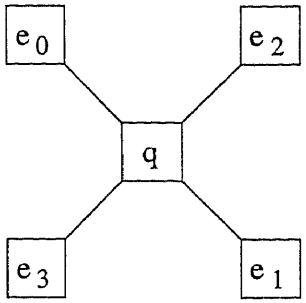
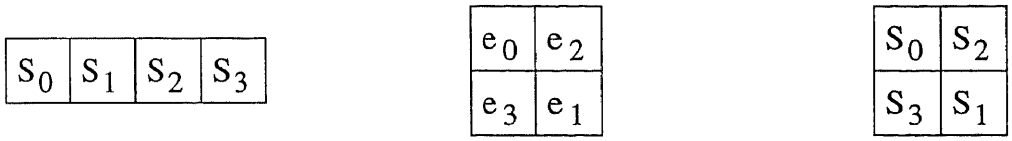


Figure 5.3 The Interleaving 2-D Array I



(a) Original Sequence (b) Interleaving Array (c) Interleaved Array

Figure 5.4 2-D Interleaving for 4 Symbols (l=2)

algorithm interleaves across the horizontal axis with a degree $\lambda_x \geq l/4$ and across the vertical axis with a degree $\lambda_y \geq l/4$ such that the total interleaving $\lambda = \lambda_x + \lambda_y$ is lower-bounded by $l/2$; that is, $\lambda \geq l/2$.

For $l = 2$: In this case Equation (5.1) gives $p = 2$, and the R array is a copy of the I array itself, as shown in Figure 5.4. Note how the interleaving degree $\lambda \geq 1$; between symbols 1 and 2, $\lambda_x = 1$ and $\lambda_y = 1$; between symbols 2 and 3, $\lambda_x = 0$ and $\lambda_y = 1$; and between symbols 3 and 4, $\lambda_x = 1$ and $\lambda_y = 1$.

For $l = 4$: In this case Equation (5.1) gives $p = 3$ and Equation (5.2) gives $q' = 1$. The original sequence is shown in Figure 5.5(a). The interleaving array I is shown in Figure 5.5(b), and the q array is located one symbol inside the R array, as shown in Figure 5.5(c). The sequencing of the q array is the same as the interleaved array obtained previously for $l = 2$ (compare Figure 5.5(c) with Figure 5.4(c)). Applying the following three steps four times for $i = 0, 1, 2$, and 3, the interleaved array shown in Figure 5.5(d) is obtained:

S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}

(a) Original Sequence

e_0		e_2
	q	
e_3		e_1

(b) Interleaving Array

	q_0	q_2	
	q_3	q_1	

(c) Positions of q

S_0	S_8	S_2	S_{10}
S_{12}	S_4	S_{14}	S_6
S_3	S_{11}	S_1	S_9
S_{15}	S_7	S_{13}	S_5

(d) Interleaved Array

Figure 5.5 2-D Interleaving for 16 Symbols (l=4)

1. e_0, e_1, e_2, and e_3 read from the original sequence in Figure 5.5(a).
2. The I array center q is located at q_i.
3. e_0, e_1, e_2, and e_3 write to the corresponding locations in the R array.

One can see that $\lambda_x \geq 1$, $\lambda_y \geq 1$, and $\lambda \geq 2$. Notice the resemblance to block interleaving in the sense that if we divide the de-interleaved sequence (which is the same sequence in Figure 5.5(a)) into four equal segments (code blocks), any cluster of 2×2 symbols in the interleaved array is spread in the de-interleaved sequence such that one symbol falls in each segment, which ensures the desired correction capability. Similarly, if we divide the de-interleaved sequence into eight equal segments, any cluster of 2×4 symbols (1/2 of the signal layout) in the interleaved array is spread in the de-interleaved sequence such that one symbol falls in each segment. To apply this algorithm in real time (as explained for the 1-D case in the previous section) using the above three steps, we will need two memory locations for each entry in the q array for the x and y coordinates. That is, we will need an array of size $l^2/2 = 8$, as in the 1-D case. This array will carry the coordinates (1,1; 2,2; 2,1; and 1,2).

For $l = 8$: In this case $p = 5$ and $q' = 2$. The interleaving array I is shown in Figure 5.6(a), and the q array is located 2 symbols inside the R array, as shown in Figure 5.6(b). The sequencing of the q array is the same as the interleaved array obtained previously for $n = 4$ (compare Figure 5.6(b) with Figure 5.5(d)). Applying the same three steps above 16 times for $i = 0, 1, ...$ and 15, the interleaved 2-D array shown in Figure 5.6(c) is obtained. In this case $\lambda_x \geq 2$, $\lambda_y \geq 2$, and $\lambda \geq 4$. $l^2/2 = 32$ memory locations are needed for interleaving. The resemblance to block interleaving can be observed if we divide the de-interleaved sequence into 16 equal segments. Any cluster of 4×4 symbols in the interleaved array is spread in the de-interleaved sequence such that one symbol falls into each segment, which ensures the desired correction capability. Also, if we divide the de-interleaved sequence into four equal segments, any cluster of 2×2 symbols in the interleaved array is spread in the de-interleaved sequence such that one symbol falls in each segment.

5.4 Comparison

Consider the case when $l = 4$, and four neighboring symbols are in error in the interleaved array, as shown in Figure 5.7(a). When the de-interleaved sequence is obtained, the error will be spread across it, as shown in Figure 5.7(b). If the code used is capable of correcting one error in a four-symbol block, it will succeed to recover the original data. Without interleaving, the same code would fail. As explained in the previous section, this correction capability applies for any single cluster of 2×2 symbols in error. When considering the case of $l = 8$, we showed that, for the same correction capability of one error in four symbols, this interleaving technique will succeed to correct any single cluster of 4×4 symbols in error. In the $l = 8$ case, if the code used is capable of correcting one error in 16 symbols, the original information will be recovered if any single cluster of 2×2 symbols is in error.

Let us compare the performance of linear codes when the error correction code applies the proposed 2-D interleaving technique versus block interleaving. In [41], block interleaving is used with two symbol writing strategies. With the first strategy, called the "boustrophedonic" pattern, shown in Figure 5.8(a), symbols are written

e_0				e_2
		q		
e_3				e_1

(a) Interleaving Array

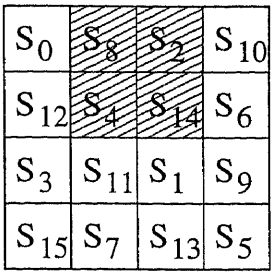
		q_0	q_8	q_2	q_{10}		
		q_{12}	q_4	q_{14}	q_6		
		q_3	q_{11}	q_1	q_9		
		q_{15}	q_7	q_{13}	q_5		

(b) Positions of q

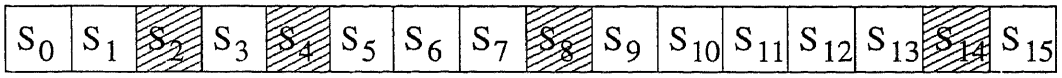
S_0	S_{32}	S_8	S_{40}	S_2	S_{34}	S_{10}	S_{42}
S_{48}	S_{16}	S_{56}	S_{24}	S_{50}	S_{18}	S_{58}	S_{26}
S_{12}	S_{44}	S_4	S_{36}	S_{14}	S_{46}	S_6	S_{38}
S_{60}	S_{28}	S_{52}	S_{20}	S_{62}	S_{30}	S_{54}	S_{22}
S_3	S_{35}	S_{11}	S_{43}	S_1	S_{33}	S_9	S_{41}
S_{51}	S_{19}	S_{59}	S_{27}	S_{49}	S_{17}	S_{57}	S_{25}
S_{15}	S_{47}	S_7	S_{39}	S_{13}	S_{45}	S_5	S_{37}
S_{63}	S_{31}	S_{55}	S_{23}	S_{61}	S_{29}	S_{53}	S_{21}

(c) Interleaved 2-D Array

Figure 5.6 2-D Interleaving for 64 Symbols (l=8)



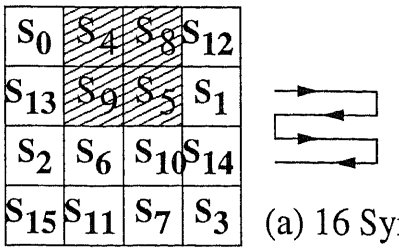
(a) Error Pattern in the Interleaved Array



(b) Error Pattern in the De-interleaved Sequence

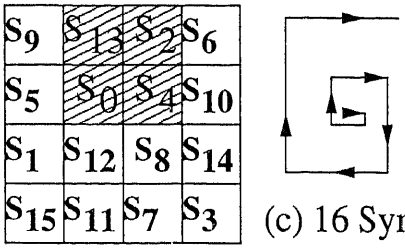
Figure 5.7 The Error Correction Capability Obtained with Interleaving

alternately in opposite directions, from left to right and right to left after applying block interleaving. With the second strategy, called the spiral data pattern, shown in Figure 5.8(c), symbols are written in a rough spiral moving outward (much like a CD recording [16]) after applying block interleaving. Figures 5.8(b) and 5.8(d) show the resulting de-interleaved sequence for the respective strategies. With the same error correction code and the same error pattern of a 2×2 cluster of symbols, one can see that the error correction code will fail to recover the data using either strategy. One may think that for large arrays, the spiral data pattern with block interleaving can result in optimum interleaving as the spiral moves outward. As can be verified from Figure 5.8(e), where block interleaving is applied with the spiral pattern for the $l = 8$ case, this is not so. Consider any round of the spiral. There is no guarantee that symbols in the next or previous round (which can fall in the same damaged area) will not share the same segment (code block) in the de-interleaved sequence, which would result in a block coding error.

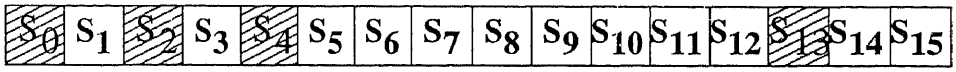


(a) 16 Symbol, Boustrophedonic Writing

(b) The De-Interleaved Sequence



(c) 16 Symbol, Spiral-Pattern Writing



(d) The De-Interleaved Sequence

S_{21}	S_{29}	S_{37}	S_{45}	S_{53}	S_{61}	S_6	S_{14}
S_{13}	S_{34}	S_{42}	S_{50}	S_{58}	S_3	S_{11}	S_{22}
S_5	S_{26}	S_{48}	S_{56}	S_1	S_9	S_{19}	S_{30}
S_{60}	S_{18}	S_{40}	S_0	S_8	S_{17}	S_{27}	S_{38}
S_{52}	S_{10}	S_{32}	S_{24}	S_{16}	S_{25}	S_{35}	S_{46}
S_{44}	S_2	S_{57}	S_{49}	S_{41}	S_{33}	S_{43}	S_{54}
S_{36}	S_{28}	S_{20}	S_{12}	S_4	S_{59}	S_{51}	S_{62}
S_{63}	S_{55}	S_{47}	S_{39}	S_{31}	S_{23}	S_{15}	S_7

(e) 64 Symbol, Spiral-Pattern Writing

Figure 5.8 Applying Block Interleaving for 2-D Applications

5.5 Summary

This chapter is a tutorial on a 2-D interleaving technique developed in the course of this research. We showed with some error patterns that 2-D interleaving gives better performance than 1-D interleaving when applied to 2-D applications. In the next chapter, a detailed analysis of this technique will be given. We will show how this technique is related to the well known set partitioning technique, which is used to partition signal constellations [37].

CHAPTER 6

SLIDING WINDOW INTERLEAVING FOR MULTIDIMENSIONAL SIGNALS

This chapter provides a sliding window implementation of the multidimensional interleaving algorithm introduced in Chapter 5. We will explain the 1-D and 2-D cases of the sliding window technique starting from the assumption that the encoded stream is of length l^2 symbols. In the 2-D case, these l^2 symbols are translated to a layout of dimension $l \times l$. While Chapter 5 introduced the algorithm in a tutorial way, this chapter will explain it in more depth and analysis.

This chapter proceeds as follows: In Section 6.1, we explain what is expected from a 2-D interleaving algorithm based on what is expected from 1-D block interleaving. In Section 6.2 we introduce the concept of interleaving using a sliding window. In Section 6.3, we apply sliding window interleaving to the 2-D case. In Section 6.4, we show how the introduced algorithm can be adopted to a non square signal layout and to a higher dimension (e.g., 3-D). Section 6.5 gives some simulation results, and Section 6.6 is a summary.

6.1 From 1-D to 2-D Interleaving

In a block of length N codewords, 1-D interleaving (block interleaving) allows the error correction decoder to correct a single burst of length $T_0 = \lambda t$ symbols using a *linear* code that is only capable of correcting a burst of length t errors [21] . λ is known as the interleaving degree and is bounded by N; that is, $T_0 \leq Nt$. Interleaving techniques considered to be optimum assure that any consecutive sequence of length Nt in the interleaved block is distributed in the de-interleaved block such that each codeword contains only t symbols. They are optimum in the sense that no other method that can assure correcting more than an Nt burst using a t correcting *linear* code. The main idea behind 1-D interleaving is shown in Figure 6.1, where any consecutive symbols of length Nt in the interleaved block are distributed in the de-interleaved block between the N codewords such that each codeword contains t symbols. As a result, when a single burst of length $\lambda t \leq Nt$ occurs in the interleaved

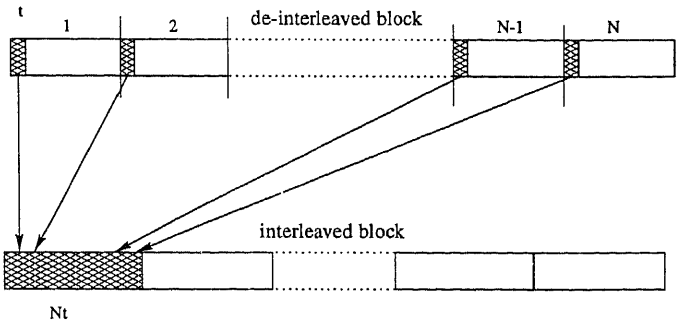


Figure 6.1 Distributing a Burst Error Nt Between the N Code Words of the 1-D De-interleaved Block

block, the error correction decoder will succeed in recovering the original data. The burst error correction capability of 1-D block interleaving can be expressed as follows:

$$T_0 = \lambda t \leq Nt. \tag{6.1}$$

For applications where the signal layout is in 2-D, we introduce an interleaving technique that uses a code capable of correcting a burst of length t symbols to correct a single burst of size (area) $T_0 \leq Nt$ in a block of length N codewords. This correction capability is based on the following definition.

Definition 1: For a signal layout of size $l \times l$ symbols, a single burst is said to be of size (area) $T_0 \leq Nt$ if all symbols in error are contained in an area $X_0 Y_0$ such that

$$X_0 \leq \frac{l}{2} \ , \ Y_0 \leq \frac{l}{2}, \tag{6.2}$$

and

$$X_0 \leq \sqrt{N}\sqrt{t} \ , \ Y_0 \leq \sqrt{N}\sqrt{t}. \tag{6.3}$$

Notice that because X_0, Y_0, N, and t are positive integers, $X_0 \leq \sqrt{N}\sqrt{t}$ and $Y_0 \leq \sqrt{N}\sqrt{t}$ implies that $X_0 Y_0 \leq Nt$. The reasons these two conditions are needed will be clarified shortly.

Figure 6.2 is the 2-D analogy to Figure 6.1 based on the above definition. A 2-D interleaving technique is expected to assure that any area of size $T_0 \leq Nt$ in the signal layout is distributed in the de-interleaved block such that each codeword contains only t symbols.

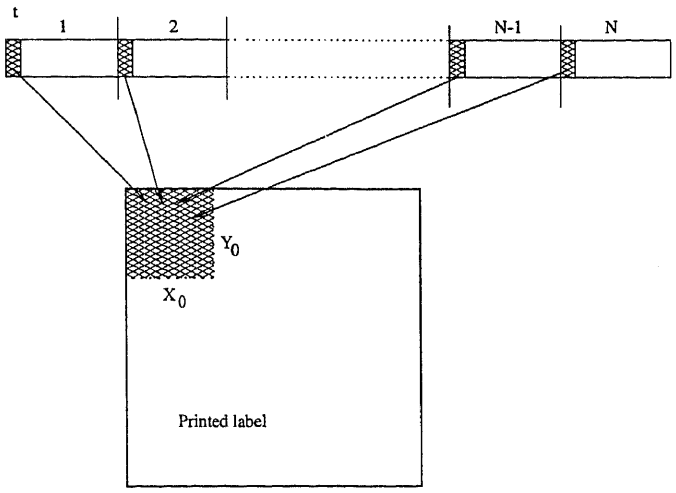


Figure 6.2 Distributing a Burst Error $Nt = X_0Y_0$ Between the N Code Words Using 2-D Interleaving

6.2 1-D Interleaving Using a Sliding Window

In this section, we introduce the concept of interleaving using a sliding window. Similar to what we did in the tutorial in the previous chapter, as this chapter proceeds, it will be shown that this technique is easy to adapt from the 1-D case to the 2-D case. First we will explain the 1-D case and show how it is equivalent to *set partitioning*, which is used to partition a signal constellation.

6.2.1 Interleaving Using Set Partitioning

With signal constellations, proper set partitioning should produce subconstellations having an increased minimum distance with each partition [37]. Figure 6.3 shows how set partitioning is used to interleave a block of 8 symbols. In the parent block (labeled A_0) we can consider that each two consecutive symbols have a distance of 1. That is, $d_{uninterleaved} = 1$. In the first partitioned blocks (I_0 and I_1), any two consecutive symbols have a distance of 2; that is, $d_{interleaved} = 2$. In the second partition (I_{00}, I_{01}, I_{10}, and I_{11}), for the two symbols within the partitioned block $d_{interleaved} = 4$. The interleaved block (labeled A_2) is obtained from the last partition. In general, if we define the interleaving gain (obtained from partitioning) as $\gamma =$

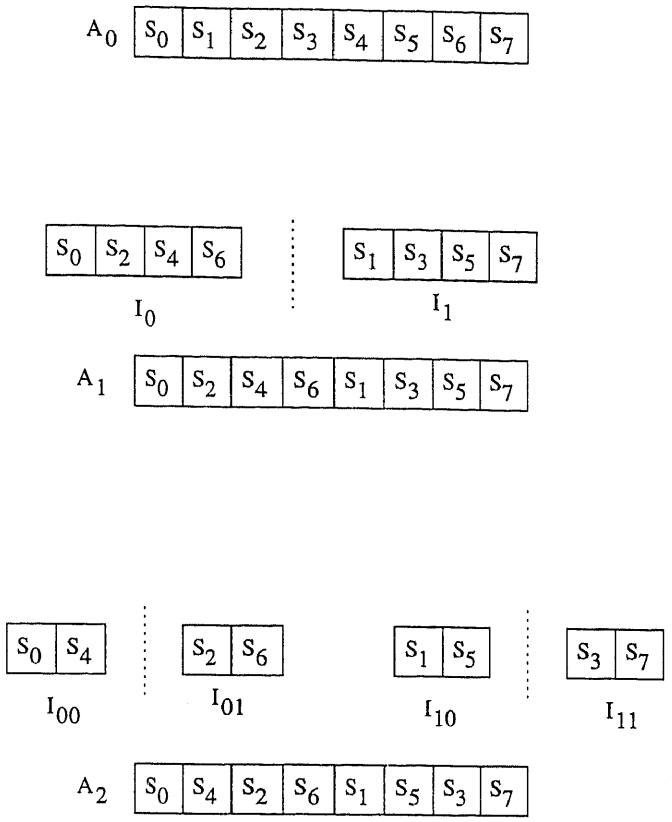


Figure 6.3 1-D Interleaving Using Set Partitioning

$\frac{d_{partition}}{d_{unpartition}}$, the maximum interleaving gain is

$$\gamma_{max} = 2^{p-1}. \tag{6.4}$$

Interleaving using this set partitioning (if applied to a block of length $l = 2^p$ symbols, with p being a positive integer, and if the partitioning tree grows $p - 1$ layers) has the following property for all power-of-2 values of N:

- Regardless of how the block is divided (that is, for any number of code words N in the block), and for any given error correction capability (t), the condition that any consecutive Nt symbols in the interleaved block are equally spread among the N codewords of the de-interleaved block is satisfied.

This can be verified from Figure 6.3 by comparing the interleaved block A_2 and the de-interleaved block (which is the same as A_0), considering the following cases:

$N = 4$; i.e., $n = 2$ (n is the number of symbols per codeword) and $t = 1$,

$N = 2$; i.e., $n = 4$ and $t = 2$, and

$N = 2$ and $t = 1$.

Notice that the $N = 4$, $n = 2$ case is satisfied from the first partition (which produces block A_1), while the $N = 2$, $n = 4$ case with $t = 1$ can only be satisfied from the second partition (block A_2). In general, with each consecutive partitioning (added layer), γ increases by a factor of 2, which increases n (which makes $T_0 = Nt$ for all t) by a factor of 2.

6.2.2 Sliding Window Interleaving Technique

The sliding window utilized is a window with a center q and two ends e_0 and e_1, as shown in Figure 6.4(a). The window's task is to read from the original block and write to the interleaved block following a given index with respect to the interleaved block. Here, the technique is explained using an *iterative* approach. Figure 6.4(b) shows how set-partition interleaving of the given example (block A_2 in Figure 6.3) can be equally obtained using a sliding window. The two-symbol block C_0 is the root block and it consists of S_0 and S_1. Similar to the partitioning tree, interleaving is performed in two layers.

- LAYER 1: Block B_0 (of length 4 symbols) is interleaved using the indexing positions in Q_{B1} where the inner-most two locations (marked q_0 and q_1) carry the indexing position of the sliding window. (In Q_{B1}, the subscripts in q_0q_1 follow the same order as the subscripts in S_0S_1 in C_0.) The window span $w = l/2 = 2$ symbols. The block is interleaved as follows:

  1-1) From the original block (B_0), read symbols S_0 and S_1 to e_0 and e_1.

  1-2) Position q over q_0.

  1-3) Write from e_0 and e_1 to the interleaved block.

  2-1) From the original block, read symbols S_2 and S_3 to e_0 and e_1.

  2-2) Position q over q_1.

  2-3) Write from e_0 and e_1 to the interleaved block.

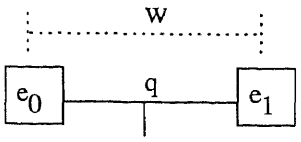


Figure 6.4(a)

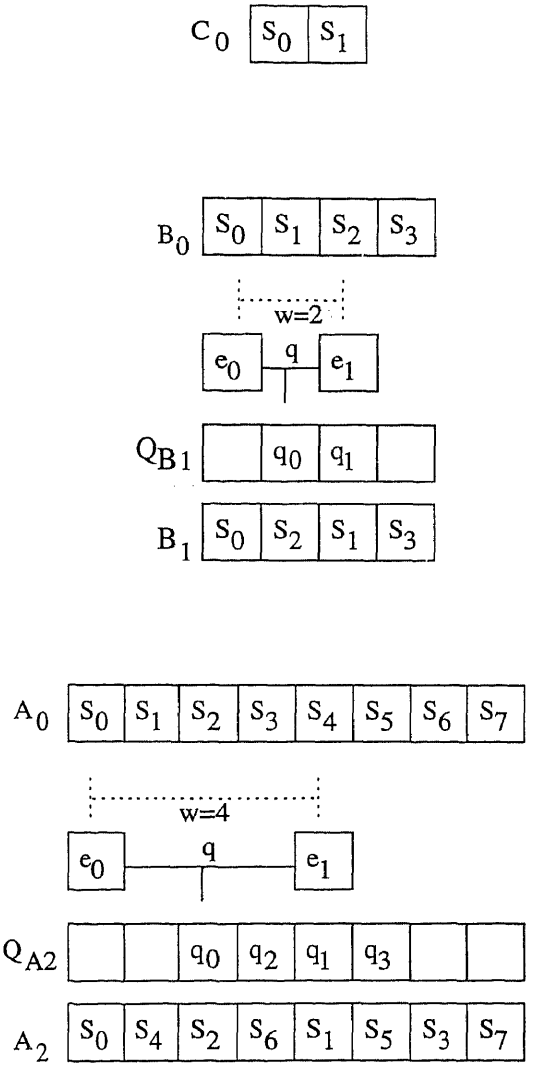


Figure 6.4(b)

Figure 6.4 1-D Interleaving Using the Sliding Window Technique
(a)Sliding Window (b)The Iterative Process

- LAYER 2: For block A_0 of size 8 symbols, block B_1 is used to obtain Q_{A2}, which has the sequence of the indexes of the sliding window. (In Q_{A2}, the subscripts in $q_0q_2q_1q_3$ follow the same order as the subscripts in $S_0S_2S_1S_3$ in B_1.) The span of the window $w = l/2 = 4$, and the process of reading from the original block and writing to the interleaved block is repeated for the four indexing positions.

6.2.3 Alternative Description of the Sliding Window Technique

To clarify the similarity between this iterative indexing process and the partitioning tree, the algorithm can be explained in an alternative way by fixing the sliding window span to $w = \frac{l}{2}$ and obtaining the indexing positions iteratively. This is shown in Figure 6.5 when $l = 8$, where the indexing positions Q_{A1} and Q_{A2} correspond to the first and second layer of the partitioning tree.

- LAYER 1: In Q_{A1}, the indexing positions $q_0q_1q_2q_3$ are not interleaved. The interleaved block A_1 is equivalent to the one obtained from the first layer of the tree (block A_1 in Figure 6.3).

- LAYER 2: Q_{A2} is obtained from a four-symbol interleaved block, as illustrated in Figure 6.4. The interleaved block A_2 is equivalent to the one obtained from the second layer of the tree (block A_2 in Figure 6.3).

As with the set partitioning tree, the iterative indexing of the sliding window assures that as the size of the block increases, the number of iterations performed to obtain the indexing positions increases, and hence γ_{max} increases.

The replacement of the binary partitioning tree with the sliding window makes the implementation of this interleaving easy to apply, as well as adaptable to the 2-D case. Although interleaving may be done in real time, the actual implementation need not go through the iterative process. To interleave a block of size l, the $l/2$ indexing positions can be obtained and stored in advance, making the actual implementation a simple memory assignment, as will be shown in Section 7.5.

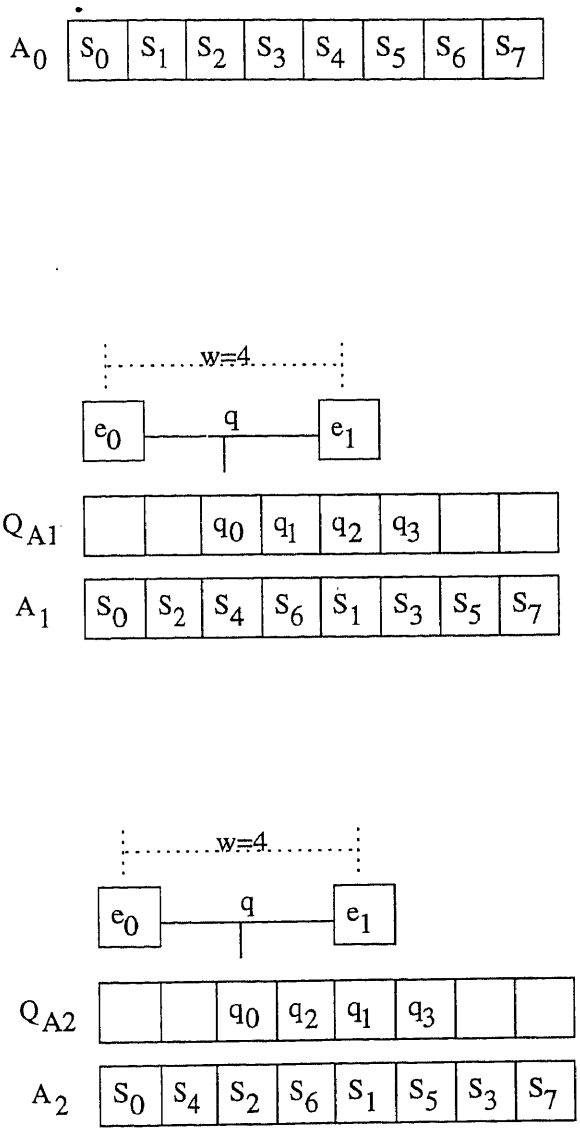


Figure 6.5 The Interleaved Block in Relation to the Iteration Steps

6.2.4 Sliding Window Interleaving Algorithm

The iterative sliding window interleaving algorithm can be summarized as follows:

For a block of length $l = 2^p$ symbols, the original block A_0 is referred to as a single-row array $A0$, and the interleaved block at the m^{th} iteration as a single-row array Am. Note that $A0$ is a sequence of l symbols $S_0, S_1, ..., S_{(l-1)}$. Am consists of 2^m single-row subarrays and is obtained from $A0$ through m iterations. The interleaving gain obtained with Am is $\gamma_m = 2^m$, and m is bounded by $1 \leq m \leq p-1$. Am is obtained from $A0$ through the following three steps.

- STEP 1: Write Am in an i-representation.

  a) Each subarray in Am is represented by $I_{i_1 i_2 i_3 ... i_m}$, where i_j is a binary variable assuming either 0 or 1 and $1 \leq j \leq m$.

  b) The 2^m subarrays are arranged in Am such that $i_1 i_2 i_3 ... i_m$ follow the natural binary code (e.g., if $m = 3$, $A3 = [I_{000} I_{001} I_{010} I_{011} I_{100} I_{101} I_{110} I_{111}]$).

- STEP 2: Transfer Am from the i-representation to a k-representation.

  a) $I_{k_1 k_2 k_3 ... k_m}$ is obtained from $I_{i_1 i_2 i_3 ... i_m}$ where k_j is obtained from i_j by

$$k_j = i_j . \gamma_{(j-1)}, \tag{6.5}$$

  where $\gamma_{(j-1)}$ is the interleaving gain at the $(j-1)^{th}$ iteration and $\gamma_0 = 2^0 = 1$.

  b) Keep the ordering of the subarrays in the k-representation the same as in the i-representation.

- STEP 3: Transfer Am from the k-representation to an s-representation.

  a) Determine the $\frac{l}{2^m}$ symbols of each subarray from the following equation:

$$\begin{aligned} I_{k_1 k_2 k_3 ... k_m} = [&S_{k_1+k_2+k_3+...+k_m} \quad S_{k_1+k_2+k_3+...+k_m+\gamma_m} \\ &S_{k_1+k_2+k_3+...+k_m+2\gamma_m} \quad \cdots \\ &S_{k_1+k_2+k_3+...+k_m+(2^{p-m}-1)\gamma_m}], \end{aligned} \tag{6.6}$$

  where $S_{k_1+k_2+k_3+...+k_m+r.\gamma_m}$ represents a symbol with an index equal to $k_1+k_2+k_3+...+k_m+r.\gamma_m$, where r is an integer in the range $0 \leq r \leq 2^{p-m}-1$. Note that for the last symbol in the subarray (the $(\frac{l}{2^m})^{th}$ symbol) $r = \frac{l}{2^m} - 1 = 2^{p-m} - 1$.

b) Keep the ordering of the subarrays in the s-representation the same as in the k-representation.

6.2.5 Example

$l = 8$, $p = 3$.

For the first iteration, $m = 1$, we have two subarrays, and

$p - m = 3 - 1 = 2 \qquad 0 \leq r \leq 2^2 - 1 = 3 \qquad \gamma_1 = 2^1 = 2$

$$\begin{aligned} A1 &\stackrel{i}{=} [I_0 \quad I_1] \\ A1 &\stackrel{k}{=} [I_0 \quad I_1] \\ A1 &\stackrel{s}{=} [S_0 S_2 S_4 S_6 \mid S_1 S_3 S_5 S_7], \end{aligned} \tag{6.7}$$

where i, k, and s above the equal sign mean in the i-, k-, and s-representations respectively.

For the second iteration, $m = 2$, we have four subarrays, and

$p - m = 3 - 2 = 1 \qquad 0 \leq r \leq 2^1 - 1 = 1 \qquad \gamma_2 = 2^2 = 4$

$$\begin{aligned} A2 &\stackrel{i}{=} [I_{00} \quad I_{01} \quad I_{10} \quad I_{11}] \\ A2 &\stackrel{k}{=} [I_{00} \quad I_{02} \quad I_{10} \quad I_{12}] \\ A2 &\stackrel{s}{=} [S_0 S_4 \mid S_2 S_6 \mid S_1 S_5 \mid S_3 S_7]. \end{aligned} \tag{6.8}$$

Clearly, $A1$ and $A2$ in the s-representations in Equations (6.7) and (6.8) match blocks A_1 and A_2 in Figure 6.5.

6.3 2-D Interleaving Using a Sliding Window

To adapt sliding-window interleaving to the 2-D case, we consider a 2-D array of size $l \times l$ symbols with l being a power of 2 (in Section 6.4.1, we will consider the non-square case). As with the 1-D case, to interleave a 2-D array of size $l \times l$, we need to obtain an interleaved 2-D array of size $\frac{l}{2} \times \frac{l}{2}$.

6.3.1 2-D Sliding Window Interleaving Technique

Here, the sliding window has four ends, as shown in Figure 6.6(a). The dimension of the sliding window is $\frac{l}{2} \times \frac{l}{2}$, which brings us to the first condition in definition

1 (see Equation (6.2)). This condition is forced, since neighboring symbols are not guaranteed to be separated by more than $l/2$ symbols across any axis. The second condition (see Equation (6.3)) is needed because the sliding window is expected to interleave equally across vertical and horizontal axes. A burst of size t in the 1-D de-interleaved block corresponds to a burst of size $\sqrt{t} \times \sqrt{t}$ in the 2-D signal layout, where $\sqrt{t}$ is an integer representing the number of symbols affected by this error across each axis. Similarly, if $t = 1$, a burst of length N in the 1-D de-interleaved block corresponds to an $\sqrt{N} \times \sqrt{N}$ burst in the signal layout, where $\sqrt{N}$ is an integer representing the number of symbols affected by this error across each axis.

Figure 6.6(b) shows the iterative steps starting from the root array D_0. If we have a four-symbol 1-D block, we can write it to a 2×2 array without interleaving (row-by-row) and obtain D_0. Here interleaving is done in three layers.

- LAYER 1: Interleaving D_0 using Q_{C1} results in the interleaved array C_1, which has the symbols $\begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$.
- LAYER 2: The 4×4 interleaved array B_2 is obtained from the indexes in Q_{B2} and using a sliding window with span $w = \frac{l}{2} = 2$. In Q_{B2}, the inner-most four elements carry the indexing positions of the sliding window and are marked $\begin{bmatrix} q_0 & q_2 \\ q_3 & q_1 \end{bmatrix}$, following the same subscripts as those in C_1. The process of reading from the original array to the sliding window (four symbols at a time) and writing to the interleaved array is repeated four times (for the four indexing locations in Q_{B2}) to obtain the interleaved array B_2.
- LAYER 3: Array B_2 is used to create the indexing positions in Q_{A3}. Q_{A3} is used with a window of span $w = 4$ to obtain the 8×8 interleaved array A_3.

6.3.2 Alternative Description of the 2-D Sliding Window Technique

As with the 1-D case, the iterative indexing of the sliding window assures that as the size of the array increases, the number of iterations increases, which increases γ_{max}. To clarify the analogy with the 1-D case, with the conditions forced in Equations (6.2) and (6.3), consider the case of an 8×8 array, shown in Figure 6.7. Here, we fix $w = \frac{l}{2} = 4$ and obtain the indexing positions iteratively. If we do not use

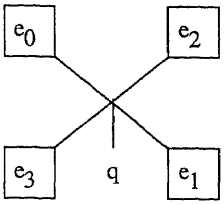


Figure 6.6(a)

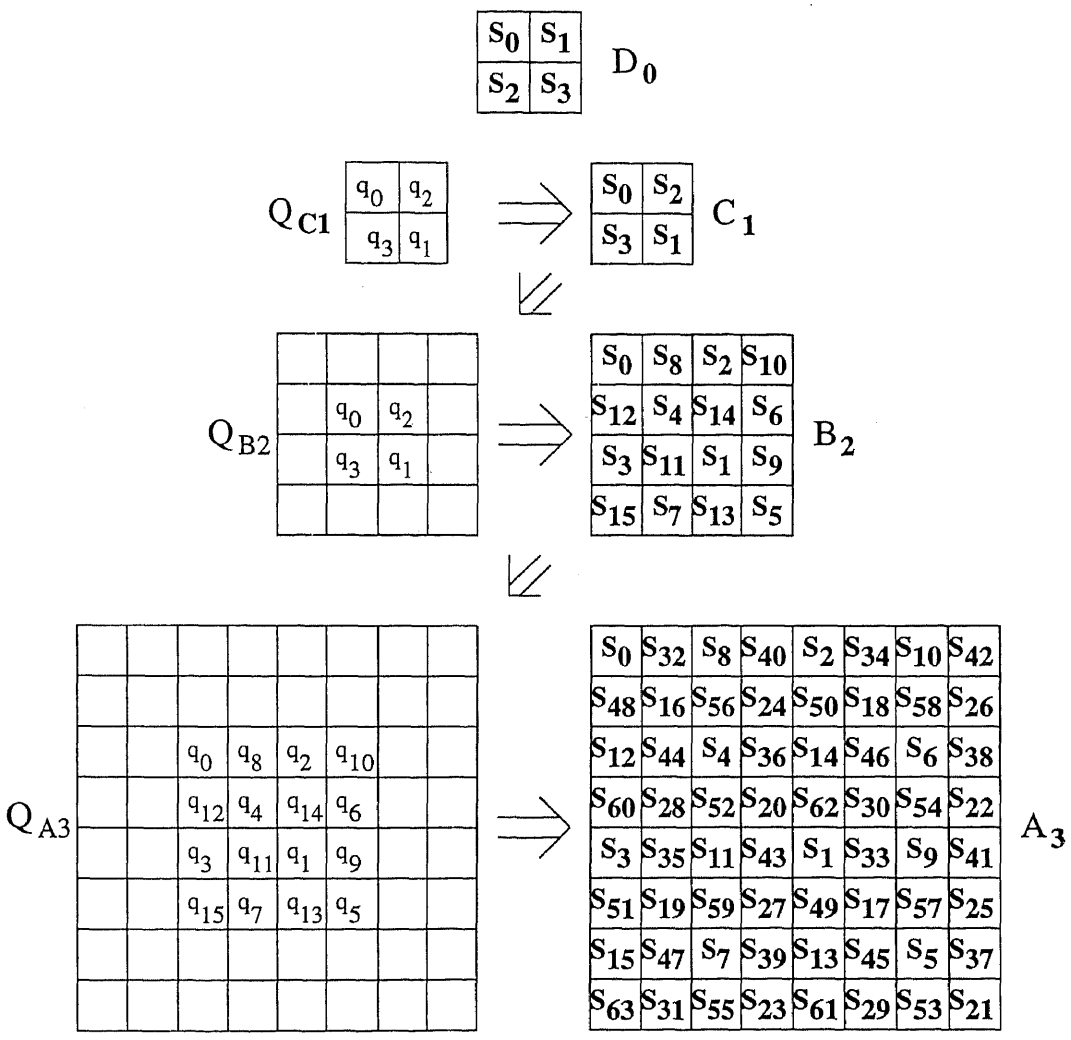


Figure 6.6(b)

Figure 6.6 2-D Interleaving Using the Sliding Window Technique
(a)2-D Sliding Window (b)The Iterative Process

any indexing (no interleaving), the 64-symbol block will be written to the 2-D array row-by-row as shown in A_0 (where $\gamma_0 = 2^0 = 1$). The three layers are as follows.

- LAYER 1: Use the indexing positions in Q_{A1} where the indexes $q_0 ... q_{15}$ are not interleaved. The interleaved array obtained is A_1. In this case, the sliding window only assures that each four consecutive symbols are located in four different quarters of the signal layout ($\gamma_1 = 2^2 = 4$). The following cases are satisfied:

  1. $N = 32, n = 2, t = 1$. Because of Equation (6.2), $T_0 = 4 \times 4$. Any 4×4 symbols are distributed among 16 different codewords.
  2. $N = 16, n = 4, t = 1$. Because of Equation (6.3), $T_0 = 4 \times 4$. Any 4×4 symbols are distributed among the 16 codewords.
  3. $N = 16, n = 4, t = 2$. Because of Equation (6.3), the nearest integer less than or equal to $\sqrt{t}$ is 1, and $T_0 = 4 \times 4 = 16$. Any 4×4 symbols are distributed among the 16 codewords.

  The case of $N = 8, n = 8, t = 1$ is not satisfied for A_1. According to Equation (6.3), the nearest integer less than or equal to $\sqrt{8}$ is 2. Thus, $T_0 = 2 \times 2 = 4$. Any 2×2 symbols are not guaranteed to be distributed among 4 different codewords (e.g., in the area containing symbols S_0, S_4, S_{16}, and S_{20}, symbols S_0 and S_4 fall in the same codeword).

- LAYER 2: The indexing positions are interleaved from one layer above, as in Q_{A2} in Figure 6.7. (Q_{A2} is obtained from Q_{B1}, where Q_{B1} is not interleaved.) The interleaved array obtained is A_2 with $\gamma_2 = 2^4 = 16$. In this case, any 2×2 symbols are distributed among four different quarters of the de-interleaved block. In addition to the above scenarios, A_2 satisfies all cases of $N = 8, n = 8$, and $N = 4, n = 16$.

- LAYER 3: The indexing positions are interleaved from two layers above, as in Q_{A3} in Figure 6.6(b). (Q_{A3} is obtained from Q_{B2}, and Q_{B2} is obtained from Q_{C1}.) The interleaved array obtained is A_3 with $\gamma_3 = 32$. In this case, in

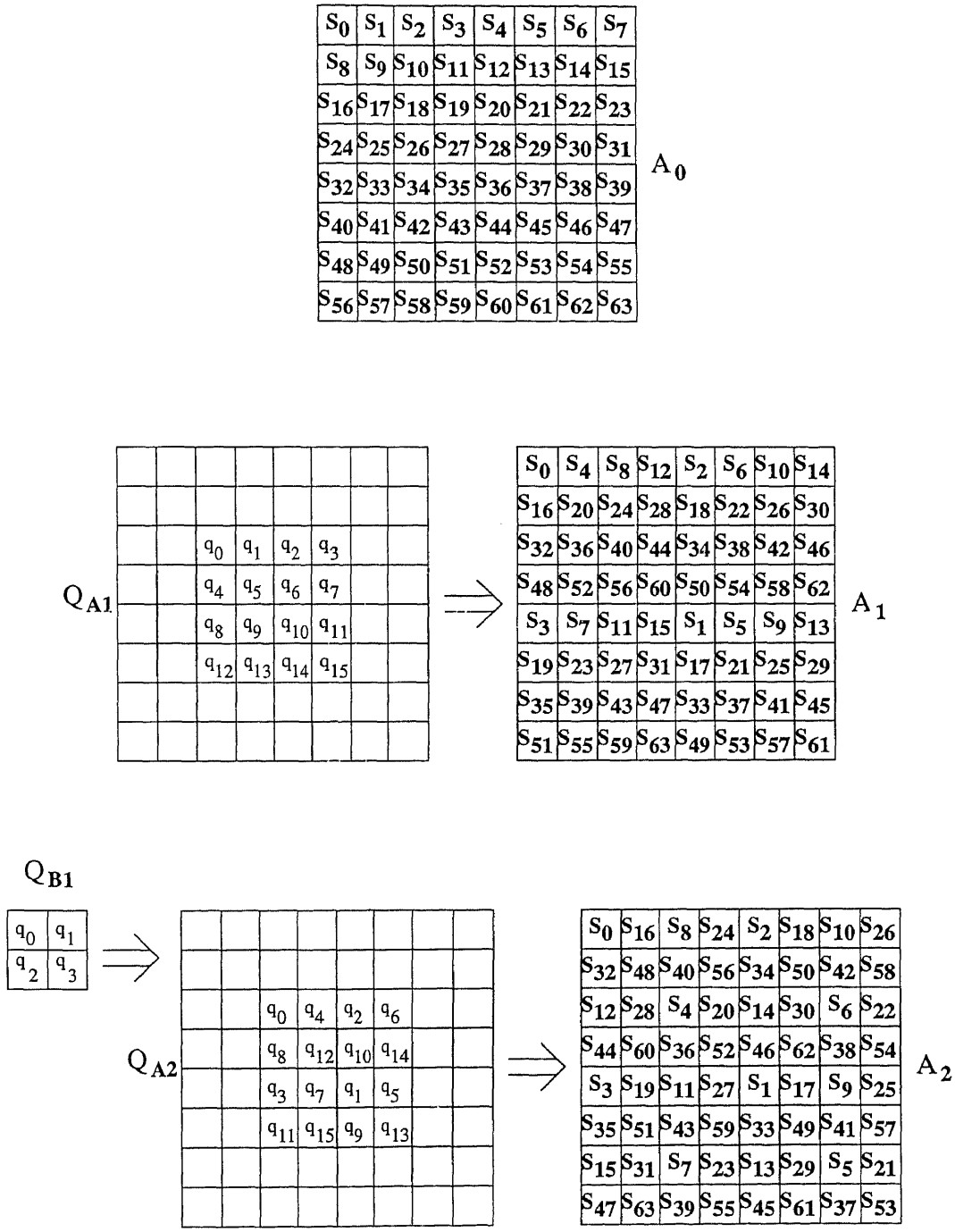


Figure 6.7 The Interleaved Array in Relation to the Iteration Steps

addition to the above scenarios, A_3 assures that any two neighboring symbols (e.g., S_0 and S_{32}; or S_0 and S_{48}) fall in different halves of the de-interleaved block, which satisfies the $N = 2, n = 32$ case.

6.3.3 2-D Sliding Window Interleaving Algorithm

The iterative sliding window interleaving algorithm for the 2-D case can be summarized as follows.

For a block of length $l^2 = 2^p.2^p = 2^{2p}$ symbols, if we do not use any indexing (no interleaving), the 1-D block is written to the $2-D$ array row-by-row. We refer to this un-interleaved array as a 2-D array $A0$, and to the interleaved 2-D array as Am. Am consists of 2^{2m} 2-D subarrays and is obtained from $A0$ through m iterations. The interleaving gain obtained with Am is $\gamma_m = 2^{2m}$. m is bounded by $1 \leq m \leq p$, and γ_m is bounded by $\gamma_m \leq \frac{l^2}{2}$. Am is obtained from $A0$ through the following steps.

- STEP 1: Write Am in an i-representation.

  a) Each subarray in the 2-D array, Am, is represented by $I_{i_1 i_2 i_3 \ldots i_m}$, where i_j is a quaternary variable ($i_j \in \{0, 1, 2, 3\}$), and $1 \leq j \leq m$.

  b) The 2^{2m} (i.e., 4^m) subarrays $I_{i_1 i_2 i_3 \ldots i_m}$ are arranged in Am as follows: a subarray $I_{i_1 i_2 i_3 \ldots i_m}$ is located in one of the four quadrants of A_m according to the value i_1 takes, which follows the pattern

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}. \tag{6.9}$$

  Within the quadrant decided above, $I_{i_1 i_2 i_3 \ldots i_m}$ is located in one of the four subquadrants according to the value i_2 takes, which also follows the pattern in Equation (6.9). Continue this process until i_m is examined, which completely specifies the location of $I_{i_1 i_2 i_3 \ldots i_m}$ in A_m. For example, if $m = 2$,

$$A1 = \begin{bmatrix} I_0 & I_2 \\ I_3 & I_1 \end{bmatrix}, \quad \text{and} \quad A2 = \begin{bmatrix} I_{00} & I_{02} & I_{20} & I_{22} \\ I_{03} & I_{01} & I_{23} & I_{21} \\ I_{30} & I_{32} & I_{10} & I_{12} \\ I_{33} & I_{31} & I_{13} & I_{11} \end{bmatrix}.$$

- STEP 2: Transfer Am from the i-representation to a k-representation.

  a) $I_{k_1 k_2 k_3 \ldots k_m}$ is obtained from $I_{i_1 i_2 i_3 \ldots i_m}$, where k_j is obtained from i_j by

$$k_j = i_j . \gamma_{(j-1)}, \tag{6.10}$$

where $\gamma_{(j-1)}$ is the interleaving gain at the $(i-1)^{th}$ iteration, and $\gamma_0 = 2^0 = 1$.

b) Keep the ordering of the subarrays in the k-representation the same as in the i-representation.

- STEP 3: Transfer Am from the k-representation to an s-representation.

  a) Determine the $\frac{l^2}{2^{2m}} = 2^{(2p-2m)}$ symbols of each subarray $I_{k_1 k_2 \ldots k_m}$, where each symbol is referred to as $S_{k_1+k_2+\ldots+k_m+r.\gamma_m}$. $k_1 + k_2 + \ldots + k_m + r.\gamma_m$ represent the symbol index, and r is an integer in the range $0 \leq r \leq 2^{(2p-2m)} - 1$. The following 2-D array R shows the values r takes in the corresponding s-representation (e.g., in the symbol at the first row and first column, $r = 0$; and in the symbol at the first row and second column, $r = 1$).

$$R = \begin{bmatrix} 0 & 1 & \ldots & 2^{(p-m)} - 1 \\ 2^{(p-m)} & 2^{(p-m)} + 1 & \ldots & 2^{(p-m+1)} - 1 \\ \ldots & \ldots & \ldots & \ldots \\ 2^{(2p-2m)} - 2^{(p-m)} & 2^{(2p-2m)} - 2^{(p-m)} + 1 & \ldots & 2^{(2p-2m)} - 1 \end{bmatrix}. \quad (6.11)$$

  b) Keep the ordering of the subarrays in the s-representation the same as in the k-representation.

6.3.4 Example

$l = 8$, $p = 3$.

For the first iteration, $m = 1$, we have four subarrays, and

$2p - 2m = 6 - 2 = 4$ $\quad 0 \leq r \leq 2^4 - 1 = 15$ $\quad \gamma_1 = 2^2 = 4$

$$A1 \stackrel{i}{=} \begin{bmatrix} I_0 & I_2 \\ I_3 & I_1 \end{bmatrix}.$$

$$A1 \stackrel{k}{=} \begin{bmatrix} I_0 & I_{2\gamma_0} \\ I_{3\gamma_0} & I_{\gamma_0} \end{bmatrix} = \begin{bmatrix} I_0 & I_2 \\ I_3 & I_1 \end{bmatrix}.$$

$$A1 \stackrel{s}{=} \left[\begin{array}{cccc|cccc} S_0 & S_4 & S_8 & S_{12} & S_2 & S_6 & S_{10} & S_{14} \\ S_{16} & S_{20} & S_{24} & S_{28} & S_{18} & S_{22} & S_{26} & S_{30} \\ S_{32} & S_{36} & S_{40} & S_{44} & S_{34} & S_{38} & S_{42} & S_{46} \\ S_{48} & S_{52} & S_{56} & S_{60} & S_{50} & S_{54} & S_{58} & S_{62} \\ -- & -- & -- & -- & -- & -- & -- & -- \\ S_3 & S_7 & S_{11} & S_{15} & S_1 & S_5 & S_9 & S_{13} \\ S_{19} & S_{23} & S_{27} & S_{31} & S_{17} & S_{21} & S_{25} & S_{29} \\ S_{35} & S_{39} & S_{43} & S_{47} & S_{33} & S_{37} & S_{41} & S_{45} \\ S_{51} & S_{55} & S_{59} & S_{63} & S_{49} & S_{53} & S_{57} & S_{61} \end{array}\right]. \quad (6.12)$$

For the second iteration, $m = 2$, we have sixteen subarrays, and

$2p - 2m = 6 - 4 = 2 \qquad 0 \le r \le 2^2 - 1 = 3 \qquad \gamma_2 = 2^4 = 16$

$$A2 \stackrel{i}{=} \begin{bmatrix} I_{0,0} & I_{0,2} & I_{2,0} & I_{2,2} \\ I_{0,3} & I_{0,1} & I_{2,3} & I_{2,1} \\ I_{3,0} & I_{3,2} & I_{1,0} & I_{1,2} \\ I_{3,3} & I_{3,1} & I_{1,3} & I_{1,1} \end{bmatrix}.$$

$$A2 \stackrel{k}{=} \begin{bmatrix} I_{0,0} & I_{0,2\gamma_1} & I_{2\gamma_0,0} & I_{2\gamma_0,2\gamma_1} \\ I_{0,3\gamma_1} & I_{0,\gamma_1} & I_{2\gamma_0,3\gamma_1} & I_{2\gamma_0,\gamma_1} \\ I_{3\gamma_0,0} & I_{3\gamma_0,2\gamma_1} & I_{\gamma_0,0} & I_{\gamma_0,2\gamma_1} \\ I_{3\gamma_0,3\gamma_1} & I_{3\gamma_0,\gamma_1} & I_{\gamma_0,3\gamma_1} & I_{\gamma_0,\gamma_1} \end{bmatrix} = \begin{bmatrix} I_{0,0} & I_{0,8} & I_{2,0} & I_{2,8} \\ I_{0,12} & I_{0,4} & I_{2,12} & I_{2,4} \\ I_{3,0} & I_{3,8} & I_{1,0} & I_{1,8} \\ I_{3,12} & I_{3,4} & I_{1,12} & I_{1,4} \end{bmatrix}.$$

$$A2 \stackrel{s}{=} \left[\begin{array}{cc|cc|cc|cc} S_0 & S_{16} & S_8 & S_{24} & S_2 & S_{18} & S_{10} & S_{26} \\ S_{32} & S_{48} & S_{40} & S_{56} & S_{34} & S_{50} & S_{42} & S_{58} \\ -- & -- & -- & -- & -- & -- & -- & -- \\ S_{12} & S_{28} & S_4 & S_{20} & S_{14} & S_{30} & S_6 & S_{22} \\ S_{44} & S_{60} & S_{36} & S_{52} & S_{46} & S_{62} & S_{38} & S_{54} \\ -- & -- & -- & -- & -- & -- & -- & -- \\ S_3 & S_{19} & S_{11} & S_{27} & S_1 & S_{17} & S_9 & S_{25} \\ S_{35} & S_{51} & S_{43} & S_{59} & S_{33} & S_{49} & S_{41} & S_{57} \\ -- & -- & -- & -- & -- & -- & -- & -- \\ S_{15} & S_{31} & S_7 & S_{23} & S_{13} & S_{29} & S_5 & S_{21} \\ S_{47} & S_{63} & S_{39} & S_{55} & S_{45} & S_{61} & S_{37} & S_{53} \end{array}\right], \tag{6.13}$$

which matches A_1 and A_2 in Figure 6.7.

For the third iteration, $m = 3$, we have 64 subarrays, and

$2p - 2m = 6 - 6 = 0 \qquad 0 \le r \le 2^0 - 1 = 0 \qquad \gamma_3 = \frac{l^2}{2} = 32$

Note that $r = 0$, which is expected since, in this last iteration, each subarray has only one symbol.

$$A3 \stackrel{i}{=} \begin{bmatrix} I_{0,0,0} & I_{0,0,2} & I_{0,2,0} & I_{0,2,2} & I_{2,0,0} & I_{2,0,2} & I_{2,2,0} & I_{2,2,2} \\ I_{0,0,3} & I_{0,0,1} & I_{0,2,3} & I_{0,2,1} & I_{2,0,3} & I_{2,0,1} & I_{2,2,3} & I_{2,2,1} \\ I_{0,3,0} & I_{0,3,2} & I_{0,1,0} & I_{0,1,2} & I_{2,3,0} & I_{2,3,2} & I_{2,1,0} & I_{2,1,2} \\ I_{0,3,3} & I_{0,3,1} & I_{0,1,3} & I_{0,1,1} & I_{2,3,3} & I_{2,3,1} & I_{2,1,3} & I_{2,1,1} \\ I_{3,0,0} & I_{3,0,2} & I_{3,2,0} & I_{3,2,2} & I_{1,0,0} & I_{1,0,2} & I_{1,2,0} & I_{1,2,2} \\ I_{3,0,3} & I_{3,0,1} & I_{3,2,3} & I_{3,2,1} & I_{1,0,3} & I_{1,0,1} & I_{1,2,3} & I_{1,2,1} \\ I_{3,3,0} & I_{3,3,2} & I_{3,1,0} & I_{3,1,2} & I_{1,3,0} & I_{1,3,2} & I_{1,1,0} & I_{1,1,2} \\ I_{3,3,3} & I_{3,3,1} & I_{3,1,3} & I_{3,1,1} & I_{1,3,3} & I_{1,3,1} & I_{1,1,3} & I_{1,1,1} \end{bmatrix}.$$

$$A3 \stackrel{k}{=} \begin{bmatrix} I_{0,0,0} & I_{0,0,2\gamma_2} & I_{0,2\gamma_1,0} & I_{0,2\gamma_1,2\gamma_2} & I_{2\gamma_0,0,0} & I_{2\gamma_0,0,2\gamma_2} & I_{2\gamma_0,2\gamma_1,0} & I_{2\gamma_0,2\gamma_1,2\gamma_2} \\ I_{0,0,3\gamma_2} & I_{0,0,\gamma_2} & I_{0,2\gamma_1,3\gamma_2} & I_{0,2\gamma_1,\gamma_2} & I_{2\gamma_0,0,3\gamma_2} & I_{2\gamma_0,0,\gamma_2} & I_{2\gamma_0,2\gamma_1,3\gamma_2} & I_{2\gamma_0,2\gamma_1,\gamma_2} \\ I_{0,3\gamma_1,0} & I_{0,3\gamma_1,2\gamma_2} & I_{0,\gamma_1,0} & I_{0,\gamma_1,2\gamma_2} & I_{2\gamma_0,3\gamma_1,0} & I_{2\gamma_0,3\gamma_1,2\gamma_2} & I_{2\gamma_0,\gamma_1,0} & I_{2\gamma_0,\gamma_1,2\gamma_2} \\ I_{0,3\gamma_1,3\gamma_2} & I_{0,3\gamma_1,\gamma_2} & I_{0,\gamma_1,3\gamma_2} & I_{0,\gamma_1,\gamma_2} & I_{2\gamma_0,3\gamma_1,3\gamma_2} & I_{2\gamma_0,3\gamma_1,\gamma_2} & I_{2\gamma_0,\gamma_1,3\gamma_2} & I_{2\gamma_0,\gamma_1,\gamma_2} \\ I_{3\gamma_0,0,0} & I_{3\gamma_0,0,2\gamma_2} & I_{3\gamma_0,2\gamma_1,0} & I_{3\gamma_0,2\gamma_1,2\gamma_2} & I_{\gamma_0,0,0} & I_{\gamma_0,0,2\gamma_2} & I_{\gamma_0,2\gamma_1,0} & I_{\gamma_0,2\gamma_1,2\gamma_2} \\ I_{3\gamma_0,0,3\gamma_2} & I_{3\gamma_0,0,1\gamma_2} & I_{3\gamma_0,2\gamma_1,3\gamma_2} & I_{3\gamma_0,2\gamma_1,\gamma_2} & I_{\gamma_0,0,3\gamma_2} & I_{\gamma_0,0,\gamma_2} & I_{\gamma_0,2\gamma_1,3\gamma_2} & I_{\gamma_0,2\gamma_1,\gamma_2} \\ I_{3\gamma_0,3\gamma_1,0} & I_{3\gamma_0,3\gamma_1,2\gamma_2} & I_{3\gamma_0,\gamma_1,0} & I_{3\gamma_0,\gamma_1,2\gamma_2} & I_{\gamma_0,3\gamma_1,0} & I_{\gamma_0,3\gamma_1,2\gamma_2} & I_{\gamma_0,\gamma_1,0} & I_{\gamma_0,\gamma_1,2\gamma_2} \\ I_{3\gamma_0,3\gamma_1,3\gamma_2} & I_{3\gamma_0,3\gamma_1,\gamma_2} & I_{3\gamma_0,\gamma_1,3\gamma_2} & I_{3\gamma_0,\gamma_1,\gamma_2} & I_{\gamma_0,3\gamma_1,3\gamma_2} & I_{\gamma_0,3\gamma_1,\gamma_2} & I_{\gamma_0,\gamma_1,3\gamma_2} & I_{\gamma_0,\gamma_1,\gamma_2} \end{bmatrix}.$$

$$= \begin{bmatrix} I_{0,0,0} & I_{0,0,32} & I_{0,8,0} & I_{0,8,32} & I_{2,0,0} & I_{2,0,32} & I_{2,8,0} & I_{2,8,32} \\ I_{0,0,48} & I_{0,0,16} & I_{0,8,48} & I_{0,8,16} & I_{2,0,48} & I_{2,0,16} & I_{2,8,48} & I_{2,8,16} \\ I_{0,12,0} & I_{0,12,32} & I_{0,4,0} & I_{0,4,32} & I_{2,12,0} & I_{2,12,32} & I_{2,4,0} & I_{2,4,32} \\ I_{0,12,48} & I_{0,12,16} & I_{0,4,48} & I_{0,4,16} & I_{2,12,48} & I_{2,12,16} & I_{2,4,48} & I_{2,4,16} \\ I_{3,0,0} & I_{3,0,32} & I_{3,8,0} & I_{3,8,32} & I_{1,0,0} & I_{1,0,32} & I_{1,8,0} & I_{1,8,32} \\ I_{3,0,48} & I_{3,0,16} & I_{3,8,48} & I_{3,8,16} & I_{1,0,48} & I_{1,0,16} & I_{1,8,48} & I_{1,8,16} \\ I_{3,12,0} & I_{3,12,32} & I_{3,4,0} & I_{3,4,32} & I_{1,12,0} & I_{1,12,32} & I_{1,4,0} & I_{1,4,32} \\ I_{3,12,48} & I_{3,12,16} & I_{3,4,48} & I_{3,4,16} & I_{1,12,48} & I_{1,12,16} & I_{1,4,48} & I_{1,4,16} \end{bmatrix}.$$

$$A3 \stackrel{s}{=} \left[\begin{array}{c|c|c|c|c|c|c|c} S_0 & S_{32} & S_8 & S_{40} & S_2 & S_{34} & S_{10} & S_{42} \\ \hline S_{48} & S_{16} & S_{56} & S_{24} & S_{50} & S_{18} & S_{58} & S_{26} \\ \hline S_{12} & S_{44} & S_4 & S_{36} & S_{14} & S_{46} & S_6 & S_{38} \\ \hline S_{60} & S_{28} & S_{52} & S_{20} & S_{62} & S_{30} & S_{54} & S_{22} \\ \hline S_3 & S_{35} & S_{11} & S_{43} & S_1 & S_{33} & S_9 & S_{41} \\ \hline S_{51} & S_{19} & S_{59} & S_{27} & S_{49} & S_{17} & S_{57} & S_{25} \\ \hline S_{15} & S_{47} & S_7 & S_{39} & S_{13} & S_{45} & S_5 & S_{37} \\ \hline S_{63} & S_{31} & S_{55} & S_{23} & S_{61} & S_{29} & S_{53} & S_{21} \end{array}\right], \tag{6.14}$$

which matches A_3 in Figure 6.6.

Notice that this interleaving process can be done in a successive manner similar to that used with the Fast Fourier Transform (FFT) [12], in which case the algorithm is applied in layers. In the first layer the first iteration is applied to obtain the four quadrants $A1 \stackrel{i}{=} \begin{bmatrix} I_0 & I_2 \\ I_3 & I_1 \end{bmatrix}$. The s-representation subarrays are obtained for each quadrant. In the second layer, each of the four subarrays is dealt with independently. Each goes through the first iteration as the original array did. This process continues successively until the quadrants of an interleaved array produce a subarray of size 2×2 (the last layer). At the last layer, subarrays are interleaved following the

pattern $\begin{bmatrix} S_0 & S_2 \\ S_3 & S_1 \end{bmatrix}$. This successive partitioning approach will be the way we present the multidimensional interleaving algorithm in the next chapter, where we will also compare the advantages and disadvantages of both sliding window and successive partitioning approaches.

6.4 Adapting the Algorithm

In this section, we adapt the algorithm to the case when the signal layout is not square and for the 3-D case. The former case can be applied to images that are not square (e.g., 192×128). The latter case will be explained in more detail in Section 7.3, and will be used to interleave the watermark signal in a video sequence in Chapter 9.

6.4.1 For a Non-Square Layout

2-D sliding-window interleaving can be applied to a non-square signal layout of size $l \times m$, where l and m are multiples of two. If we start from an $l \times l$ signal layout, we can append two columns (or two rows) at a time, to obtain $l \times (l+2)$, $l \times (l+4)$ layouts. Figure 6.8 shows how a 4×4 array is appended to 4×6 and to 4×8. In this case, the sliding window dimension is $\frac{m}{2} \times \frac{l}{2}$.

6.4.2 To Higher-Dimension Interleaving

In the same way that sliding-window interleaving is adapted from the 1-D case to the 2-D case, it can be adapted to higher-dimension interleaving. For a cube of dimension $l \times l \times l$, where $l = 2^p$, the sliding window is a cube of dimension $l/2$ (with the 8 ends as the 8 corners of the cube). Figure 6.9 shows the base cube of dimension $2 \times 2 \times 2$. The interleaved cube can be expressed in terms of the XYZ coordinates shown in Table 6.1. Table 6.2 shows the coordinates of each interleaved symbol for an interleaved cube of size $4 \times 4 \times 4$ that is obtained using the coordinates of the interleaved $2 \times 2 \times 2$ cube in Table 6.1. Notice that this is not the only way to interleave a $2 \times 2 \times 2$ or a $4 \times 4 \times 4$ cube. In Section 7.3, we will cover 3-D interleaving in detail.

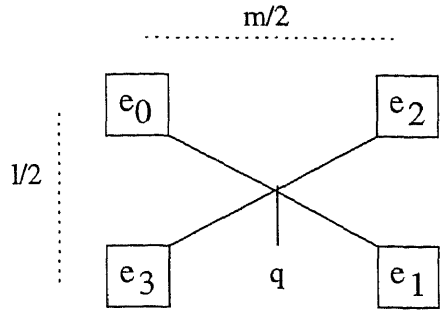


Figure 6.8(a)

	q_0	q_4	q_2		
	q_3	q_5	q_1		

S_0	S_{16}	S_8	S_2	S_{18}	S_{10}
S_{12}	S_{20}	S_4	S_{14}	S_{22}	S_6
S_3	S_{19}	S_{11}	S_1	S_{17}	S_9
S_{15}	S_{23}	S_7	S_{13}	S_{21}	S_5

Figure 6.8(b)

		q_0	q_4	q_2	q_6		
		q_3	q_7	q_1	q_5		

S_0	S_{16}	S_8	S_{24}	S_2	S_{18}	S_{10}	S_{26}
S_{12}	S_{28}	S_4	S_{20}	S_{14}	S_{30}	S_6	S_{22}
S_3	S_{19}	S_{11}	S_{27}	S_1	S_{17}	S_9	S_{25}
S_{15}	S_{31}	S_7	S_{23}	S_{13}	S_{29}	S_5	S_{21}

Figure 6.8(c)

Figure 6.8 Appending an $l \times l$ Array to Obtain an $l \times m$ Array.
(a)Interleaving Window; (b)$l = 4, m = 6$; (c)$l = 4, m = 8$

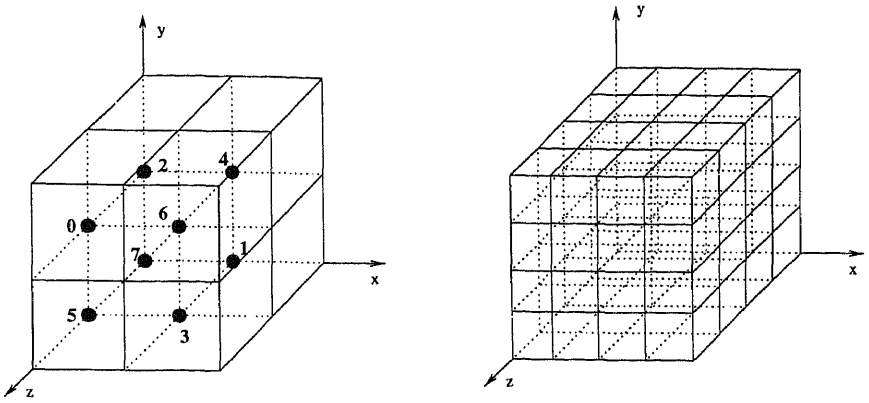


Figure 6.9 3-D Interleaving, $2 \times 2 \times 2$ and $4 \times 4 \times 4$

Table 6.1 3-D Coordinates for an Interleaved $2 \times 2 \times 2$ Cube

Symbol	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
Coordinates	011	100	010	101	110	001	111	000

Table 6.2 3-D Coordinates for an Interleaved $4 \times 4 \times 4$ Cube

Symbol	Coordin.	Symbol	Coordin.	Symbol	Coordin.	Symbol	Coordin.
S_0	033	S_{16}	032	S_{32}	132	S_{48}	133
S_1	211	S_{17}	210	S_{33}	310	S_{49}	311
S_2	031	S_{18}	030	S_{34}	130	S_{50}	131
S_3	213	S_{19}	212	S_{35}	312	S_{51}	313
S_4	231	S_{20}	230	S_{36}	330	S_{52}	331
S_5	013	S_{21}	012	S_{37}	112	S_{53}	113
S_6	233	S_{22}	232	S_{38}	332	S_{54}	333
S_7	011	S_{23}	010	S_{39}	110	S_{55}	111
S_8	122	S_{24}	123	S_{40}	023	S_{56}	022
S_9	300	S_{25}	301	S_{41}	201	S_{57}	200
S_{10}	120	S_{26}	121	S_{42}	021	S_{58}	020
S_{11}	302	S_{27}	303	S_{43}	203	S_{59}	202
S_{12}	320	S_{28}	321	S_{44}	221	S_{60}	220
S_{13}	102	S_{29}	103	S_{45}	003	S_{61}	002
S_{14}	322	S_{30}	323	S_{46}	223	S_{62}	222
S_{15}	100	S_{31}	101	S_{47}	001	S_{63}	000

6.5 Simulation

At the end of Chapter 5 we showed, with a tutorial example, that applying the proposed 2-D interleaving is better than applying block interleaving to signals with 2-D layout. In this section we will show the same conclusion using simulation. 2-D barcoding [9] [23] [26] [41] [42] is one of the applications where the encoded data layout is in 2-D. In 2-D barcoding, block interleaving is used with the two symbol writing strategies, boustrophedonic and spiral, explained in Section 5.4. In [41], it is reported that experimental results showed the first strategy to give closer performance to the second strategy while allowing the writing process (which decides the signal layout) to be simple. We present simulation results here which show that the presented 2-D interleaving technique outperforms both boustrophedonic and spiral strategies. We present our simulation based on the assumption that we are not dealing with a power-constrained signal. We used Reed-Solomon (RS) codes from $GF(2^6)$ and we used six bits per symbol. (In Chapter 8, where we deal with a power-constrained watermark signal, we will apply this interleaving technique with some modifications.)

For an assumed 2-D barcode of size 16×16 symbols, the following three strategies were tested:

1) 1-D interleaving with spiral writing.

2) 1-D interleaving with boustrophedonic writing.

3) 2-D interleaving with conventional writing (i.e., symbols are always written from left to right).

First, an RS code of rate 1/2 with erasure (i.e., k=1 and n=2) was tested. The simulation intended to calculate the probability of symbol decoding error with each strategy as the size of a square burst error increases. All possible positions of the error burst with respect to the signal layout were considered equally. The curve in Figure 6.10 shows the obtained results, where the horizontal axis is the dimension of the error ($X_0 = Y_0$), and the vertical axis is the calculated probability of symbol decoding error. One can see that with 2-D interleaving, symbol decoding error started to occur only when the size of the error burst exceeded $\frac{l}{2} \times \frac{l}{2} = 64$ (see Equation (6.2)). One

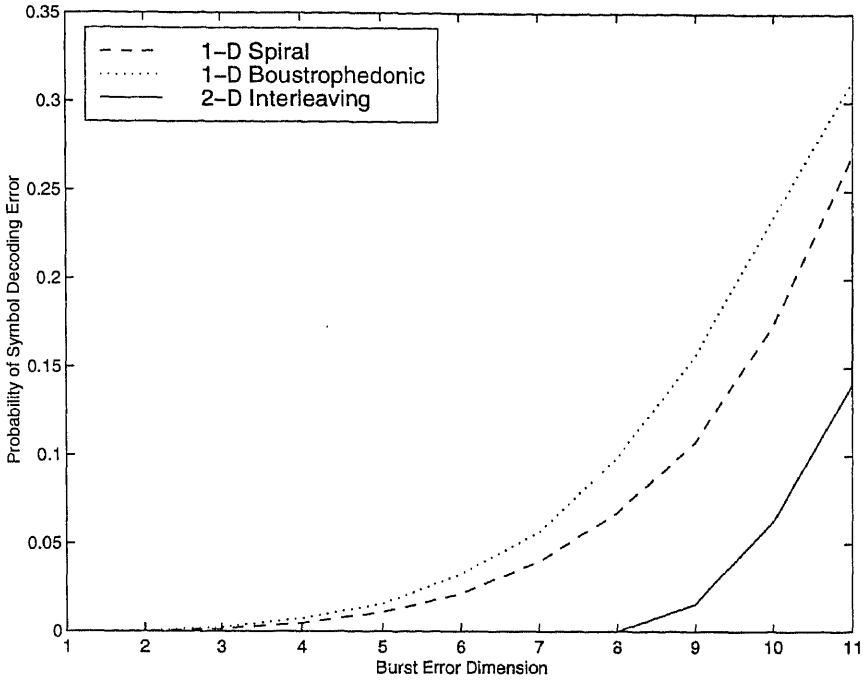


Figure 6.10 Three Coding Approaches Using a $k = 1, n = 2$ RS Code with Erasure

can also see from the 1-D interleaving curves how, in order to simplify the writing process, some performance was sacrificed with the boustrophedonic approach.

Figure 6.11 shows how the performance of 1-D interleaving can be improved by increasing the dimension n of the used RS code (for the same code rate), which leads to a more complex error correction encoder/decoder. An $n = 8, k = 4$ rate 1/2 code with 1-D interleaving and spiral writing still gave lower performance than an $n = 2$ rate 1/2 code with 2-D interleaving. We would like to note that for higher-order codes (e.g., $n = 16$) the probability of symbol decoding error of the three tested strategies decreased to around the same level. For an application with a larger signal layout (e.g., image watermarking), the difference between 1-D and 2-D interleaving, shown in Figure 6.10, is expected to increase. We would also like to note that with 2-D interleaving, the same performance is obtained regardless of whether the writing process is from left to right or is boustrophedonic.

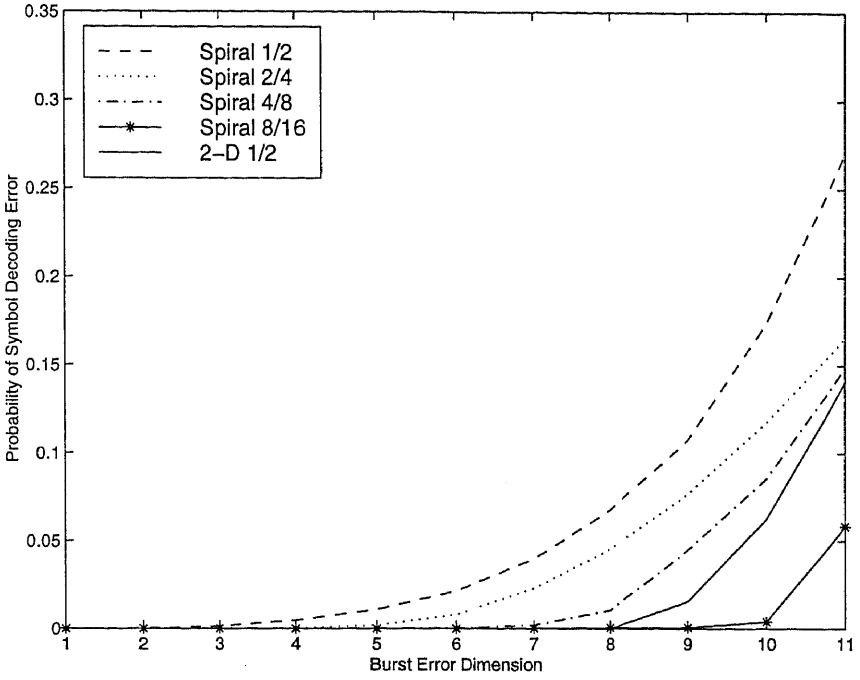


Figure 6.11 Rate $\frac{1}{2}$ 1-D Interleaving with Spiral Writing as n Increases

6.6 Summary

This chapter gives an analytical explanation of the presented multidimensional interleaving algorithm based on a sliding window approach. Based on this approach, we presented the 1-D case and adapted it to higher-dimension cases (e.g., 2-D interleaving and 3-D interleaving). We showed that 2-D interleaving, when used in 2-D applications where the signal in not power constrained, has the following two advantages:

- It can simplify the error correction code used since it can allow the use of low-dimension codes, and
- It can simplify the writing process, since no spiral writing is required.

In Chapter 7, we will present this multidimensional interleaving technique using a successive partitioning approach instead of the sliding window approach and we will emphasize the 3-D case. In Chapter 8, we will apply the 2-D version of this technique to interleaving the watermark signal within a still image or a single frame in a video

sequence. We will show how interleaving power-constrained signals is approached differently from interleaving nonpower-constrained signals. In Chapter 9, we will apply the 3-D version of this technique to interleaving the watermark signal in a video sequence.

CHAPTER 7

MULTIDIMENSIONAL INTERLEAVING USING SUCCESSIVE PARTITIONING TECHNIQUES

In Chapter 6, our multidimensional interleaving algorithm is explained using a sliding window technique. In this chapter, the algorithm is explained using a successive partitioning technique and generalized for the n-D case. The implementation of each technique is shown with a programming example.

Interleaving using successive partitioning interleaves in n-D using a partitioning tree with 2^n branches. That is, for the 1-D case a binary partitioning tree is used, for the 2-D case a quaternary partitioning tree is used, and for the 3-D case an octonary partitioning tree is used. Note that implementing this multidimensional interleaving algorithm can be done using the sliding window technique or the successive partitioning technique. The simulation in Chapter 8, where the watermark signal is interleaved in 2-D, implements the sliding window technique, while the simulation in Chapter 9, where the watermark signal is interleaved in 3-D, implements the successive partitioning technique.

This chapter proceeds by explaining the 1-D, 2-D, and 3-D cases in Sections 7.1, 7.2, and 7.3 respectively, and presenting the n-D case in Section 7.4. Section 7.5 compares the implementation of the algorithm using the sliding window technique and the successive partitioning technique. Section 7.6 is a summary

7.1 1-D Interleaving Using A Binary Partitioning Tree

In this section the successive 1-D interleaving using a binary partitioning tree is explained. A tutorial of the technique is presented, followed by the implementation algorithm.

7.1.1 Binary Partitioning Tree Technique

This section shows how the 1-D interleaving case (explained using set partitioning in Section 6.2) can be explained using a successive binary partitioning approach.

Figure 7.1 shows how this is done when the block to be interleaved, A_0, consists of 16 symbols. Successive partition interleaving is done in layers as follows.

- LAYER 1: The block is divided into two parts, the left side and the right side. The left side contains symbols with even indexes ($S_0S_2S_4S_6S_8S_{10}S_{12}S_{14}$), while the right side contains symbols with odd indexes ($S_1S_3S_5S_7S_9S_{11}S_{13}S_{15}$). The interleaved block from the first partitioning layer is A_1 (which is obtained by concatenating the two parts) and the interleaving gain obtained is $\gamma_1 = 2$.

- LAYER 2: Each of the two parts (obtained at the first layer) is dealt with separately. The left side itself is considered a block that needs to be interleaved and the right side itself is considered a block that needs to be interleaved. By partitioning the left block, we obtain the two parts $S_0S_4S_8S_{12}$ and $S_2S_6S_{10}S_{14}$, and by partitioning the right block, we obtain the two parts $S_1S_5S_9S_{13}$ and $S_3S_7S_{11}S_{15}$. The interleaved block from the second partitioning layer is A_2 (which is obtained by concatenating the now four parts) and the interleaving gain obtained is $\gamma_2 = 4$.

- LAYER 3: Each of the four parts obtained at the second layer is dealt with separately. Each part itself is considered a block that needs to be interleaved. The block containing symbols $S_0S_4S_8S_{12}$ is partitioned to S_0S_8, and S_4S_{12}, the block containing symbols $S_2S_6S_{10}S_{14}$ is partitioned to S_2S_{10} and S_6S_{14}, the block containing symbols $S_1S_5S_9S_{13}$ is partitioned to S_1S_9 and S_5S_{13}, and the block containing symbols $S_3S_7S_{11}S_{15}$ is partitioned to S_3S_{11} and S_7S_{15}. The interleaved block from the third partitioning layer is A_3 (which is obtained by concatenating the now eight parts) and the (maximum) interleaving gain $\gamma_{max} = \gamma_3 = 8$ is obtained from the third (last) layer.

One can see that with this technique the first layer turns the problem from partitioning a 16-symbol block into two problems of partitioning two 8-symbol blocks. The second layer turns the problem from partitioning two 8-symbol blocks into four problems of partitioning four 4-symbol blocks. This approach is similar to the way

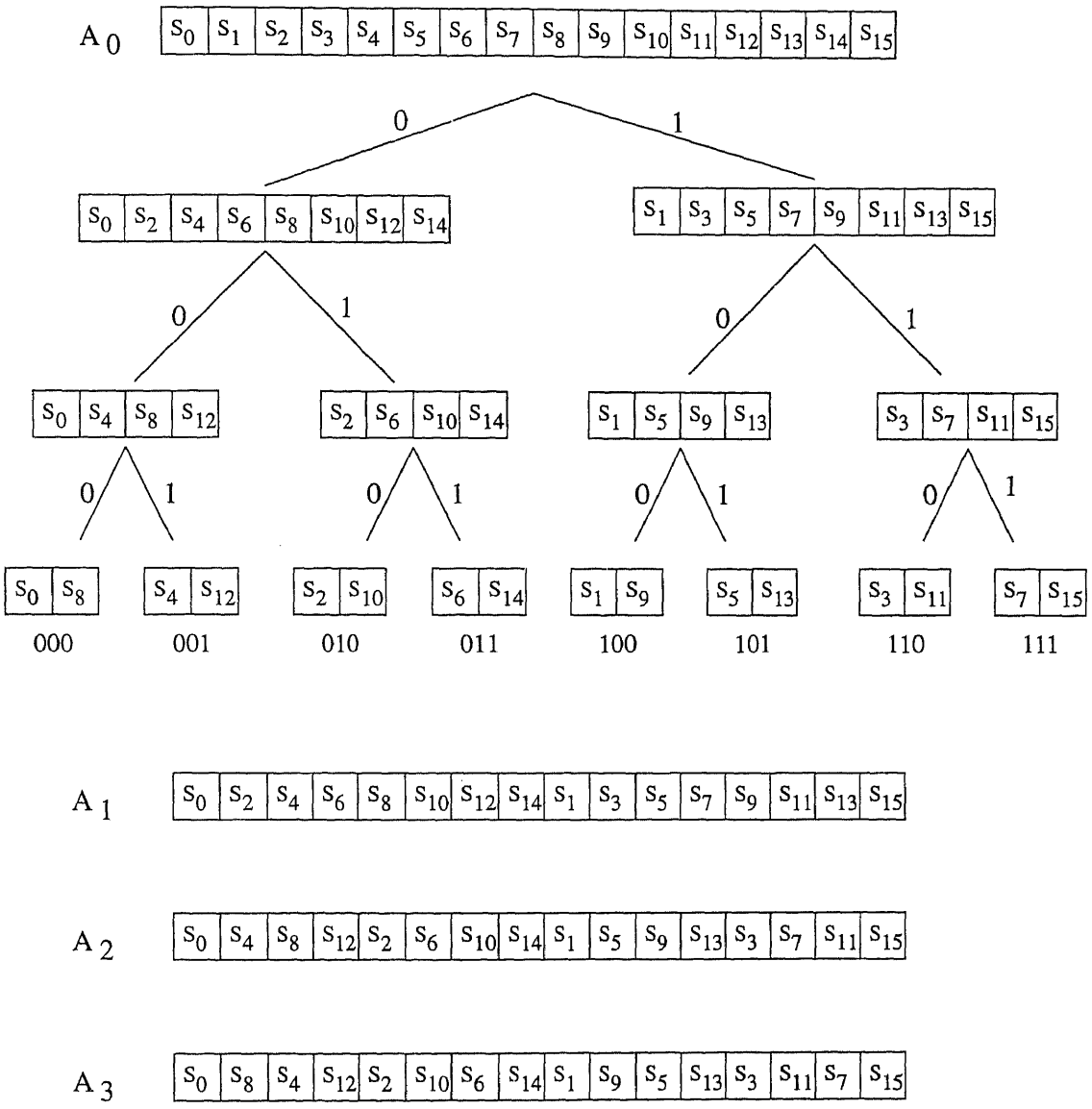


Figure 7.1 1-D Interleaving Using a Binary Partitioning Tree for a 16-Symbol Block

successive doubling is applied for the Fast Fourier Transform (FFT) [19] and to partitioning signal constellations for Trellis Coded Modulations (TCM) [37].

Similar to what is explained in Chapter 6 for the sliding window technique, for a block of length $l = 2^p$ symbols, where p is a positive integer, the maximum number of layers the tree grows increases as the size of the block increases, making γ_{max} increase. The maximum interleaving gain is given by

$$\gamma_{max} = 2^{p-1}. \tag{7.1}$$

Section 7.5 shows that if we obtain the order in which indexes are interleaved, interleaving becomes a simple memory assignment. For real-time implementation, the interleaved indexes can be obtained in advance and stored for use by the interleaving program, which is a simple memory assignment. Thus, throughout the rest of this chapter, the algorithm is presented as a way to obtain the order in which the original sequence symbol indexes are arranged in the interleaved sequence.

7.1.2 Successive Binary Partition Interleaving Algorithm

The successive binary partition interleaving algorithm can be summarized as follows:

From a block A_0 of length $l = 2^p$ symbols, the interleaved block, A_m, which consists of 2^m subblocks (parts), is obtained by utilizing a binary tree with m layers, which produces 2^m end-leaves (see Figure 7.2). The interleaving gain obtained with A_m is $\gamma_m = 2^m$, and m is bounded by $1 \leq m \leq p-1$. Note that A_0 is a sequence of l symbols, $S_0, S_1, ..., S_{(l-1)}$, where the indexes of these symbols are $0, 1, 2, ..., l-1$. The order of these indexes in A_m is obtained as follows.

- STEP 1: Find the label $(i_1 i_2 i_3 ... i_m)$ of each end leaf in the tree.

  a) With each partition, a node in the tree is split into two branches, the left branch is labeled 0 and the right branch is labeled 1.

  b) At the m^{th} layer, follow the labels at the partitioning path branches to find the end-leaf labels, (e.g., if $m = 3$, the eight end leaves will have the following labels: $000, 001, 010, 011, 100, 101, 110$, and 111, as shown in Figure 7.2).

- STEP 2: Transfer end-leaf labels to leading-index labels.

  A leading-index label, $k_1 k_2 k_3 ... k_m$, is obtained from an end-leaf label, $i_1 i_2 i_3 ... i_m$,

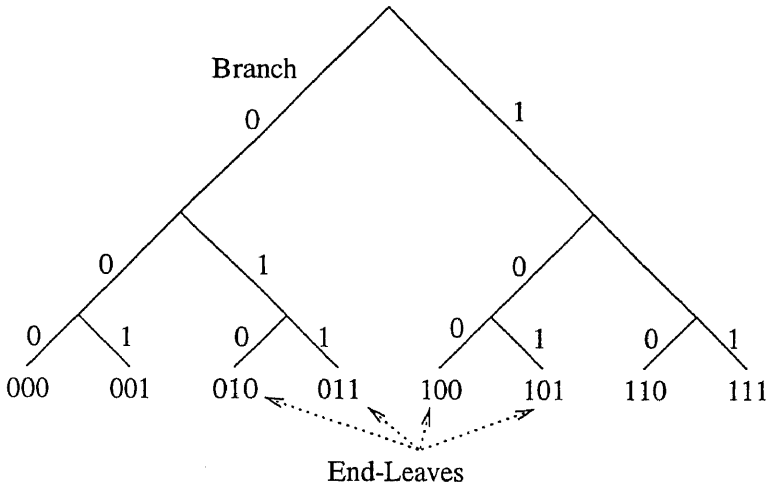


Figure 7.2 Labeling the Binary Tree Branches to Obtain the End-leaf Labels

where k_j is obtained from i_j by

$$k_j = i_j . \gamma_{(j-1)}, \tag{7.2}$$

where $\gamma_{(j-1)}$ is the interleaving gain at the $(j-1)^{th}$ layer, $0 \leq j \leq m$, and $\gamma_0 = 2^0 = 1$.

- STEP 3: Find the location of each of the 2^m subblocks.
  Based on the end-leaf labels obtained in step 1, find the location of each subblock with respect to A_m. The first digit in the end-leaf label decides which half of A_m the subarray falls in (the right half or the left half). The second digit decides which quarter (within the already decided half) the subblock falls in. Continue this until the m^{th} digit is checked, which specifies the exact location of the subblock.

- STEP 4: Find the indexes of the interleaved array symbols.
  For a leading-index label $k_1 k_2 k_3 ... k_m$, find the indexes of the subblock (group of $\frac{l}{2^m}$ symbols) decided by this index. The indexes of the $\frac{l}{2^m}$ symbols are given by

$$k_1 + k_2 + ... + k_m + r\gamma_m, \tag{7.3}$$

where r is an integer in the range $0 \leq r \leq 2^{p-m} - 1$. For the first symbol in the subblock $r = 0$ and for the last symbol in the subblock (the $(\frac{l}{2^m})^{th}$ symbol), $r = \frac{l}{2^m} - 1 = 2^{p-m} - 1$.

7.1.3 Example

$l = 16$, $p = 4$.

For the first layer, $m = 1$, there are two subblocks, and

$p - m = 4 - 1 = 3$ $\quad 0 \leq r \leq 2^3 - 1 = 7$ $\quad \gamma_1 = 2^1 = 2.$

The labels of the two end-leaves are (0, 1).

The leading-index labels are (0, 1).

The indexes of the interleaved array symbols are

$[0, 2, 4, 6, 8, 10, 12, 14 \mid 1, 3, 5, 7, 9, 11, 13, 15]$,

which matches the indexes in A_1 in Figure 7.1.

For the second layer, $m = 2$, there are four subblocks, and

$p - m = 4 - 2 = 2$ $\quad 0 \leq r \leq 2^2 - 1 = 3$ $\quad \gamma_2 = 2^2 = 4.$

The labels of the four end leaves are (00, 01, 10, 11).

The leading-index labels are (00, 02, 10, 12).

The indexes of the interleaved array symbols are

$[0, 4, 8, 12 \mid 2, 6, 10, 14 \mid 1, 5, 9, 13 \mid 3, 7, 11, 15]$,

which matches the indexes in A_2 in Figure 7.1.

For the third layer, $m = 3$, there are eight subblocks, and

$p - m = 4 - 3 = 1$ $\quad 0 \leq r \leq 2^1 - 1 = 1$ $\quad \gamma_3 = 2^3 = 8.$

The labels of the four end leaves are (000, 001, 010, 011, 100, 101, 110, 111).

The leading-index labels are (000, 004, 020, 024, 100, 104, 120, 124).

The indexes of the interleaved array symbols are

$[0, 8, \mid 4, 12 \mid 2, 10 \mid 6, 14 \mid 1, 9 \mid 5, 13 \mid 3, 11 \mid 7, 15]$,

which matches the indexes in A_3 in Figure 7.1

7.2 2-D Interleaving Using A Quaternary Partitioning Tree

In this section, successive 2-D interleaving using a quaternary partitioning tree is explained. As with the 1-D case, a tutorial of the technique is provided followed by the implementation algorithm.

7.2.1 Quaternary Partitioning Tree Technique

This section shows how the 2-D interleaving case explained using a sliding window in Section 6.3 can be explained using a successive partitioning approach. Figure 7.3 shows how this is done when the block to be interleaved, $A0$, consists of 64 symbols, and the signal layout is in 2-D (8×8 symbols). Figure 7.4 shows the obtained interleaved signal layout at each layer. In Figure 7.4, A_0 is the obtained 2-D layout if no interleaving is done; that is, if the 64 symbol of $A0$ are written directly to the 2-D layout row-by-row. Successive partition interleaving is done in layers as follows.

- LAYER 1: The block is divided into four parts. The first part (which will be referred to as the far-left part) contains symbols with indexes (0,4,8,12, ..., 60); the second part (the mid-left part) contains symbols with indexes (1,5,9,13, ..., 61); the third part (the mid-right part) contains symbols with indexes (2,6,10,14, ..., 62); and the fourth part (the far-right part) contains symbols with indexes (3,7,11,15, ..., 63). These four parts are assigned to the four branches labeled 0, 1, 2, and 3 at the first layer of the tree as shown in Figure 7.3.

  The interleaved 2-D array obtained from the first partitioning layer is A_1 (shown in Figure 7.4) and the interleaving gain obtained is $\gamma_1 = 4$. This is done by writing the four parts to A_1 such that one part is written to one quarter with the branch labels following the pattern

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}. \tag{7.4}$$

  That is, the far-left part (with branch label 0 in Figure 7.3) is written to the upper-left quarter of A_1, the mid-left part (with branch label 1 in Figure 7.3) is written to the lower-right quarter of A_1, the mid-right part (with branch label

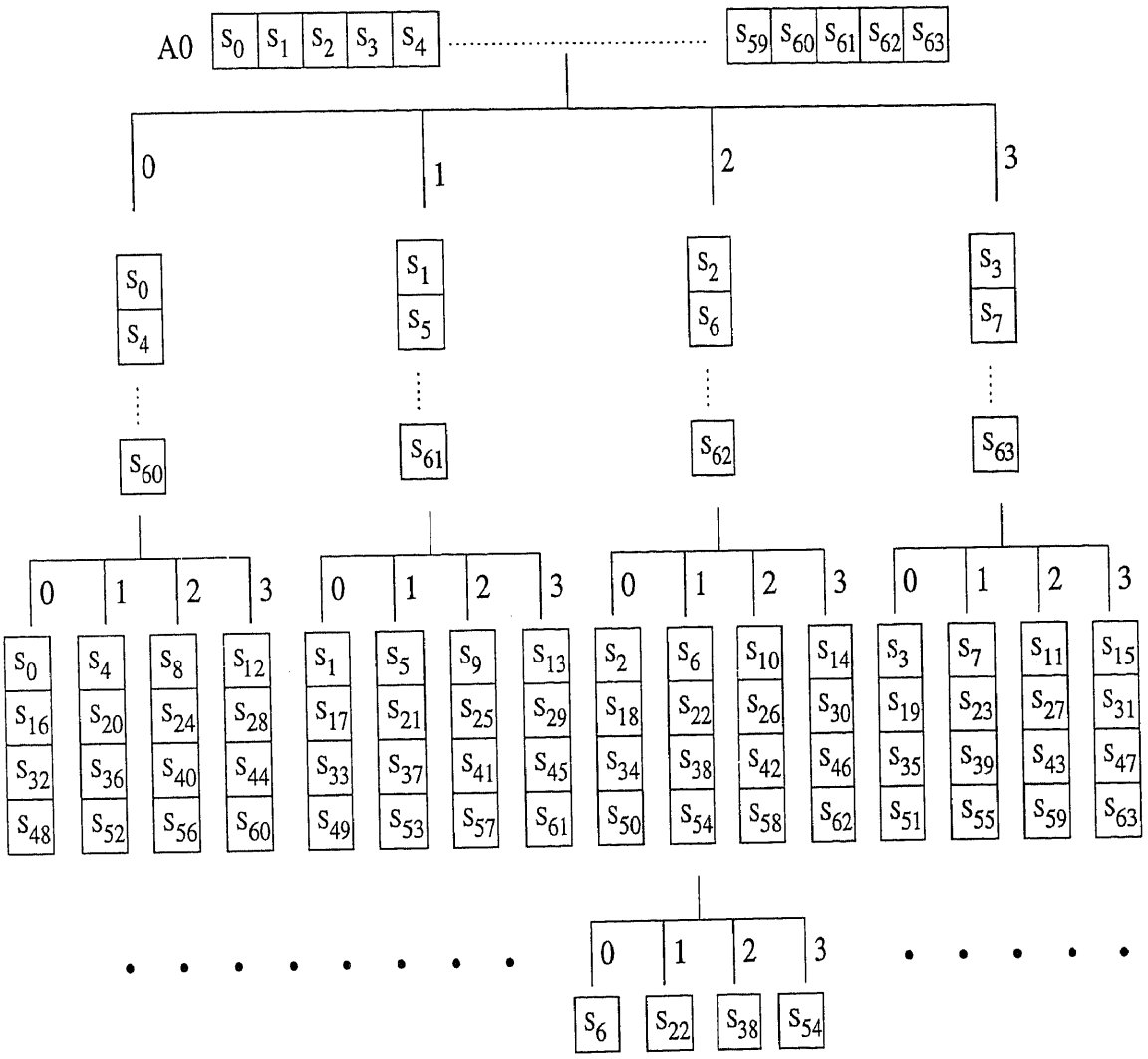


Figure 7.3 2-D Interleaving Using a Quaternary Partitioning Tree for a 64-Symbol Block (the third layer shows only one part)

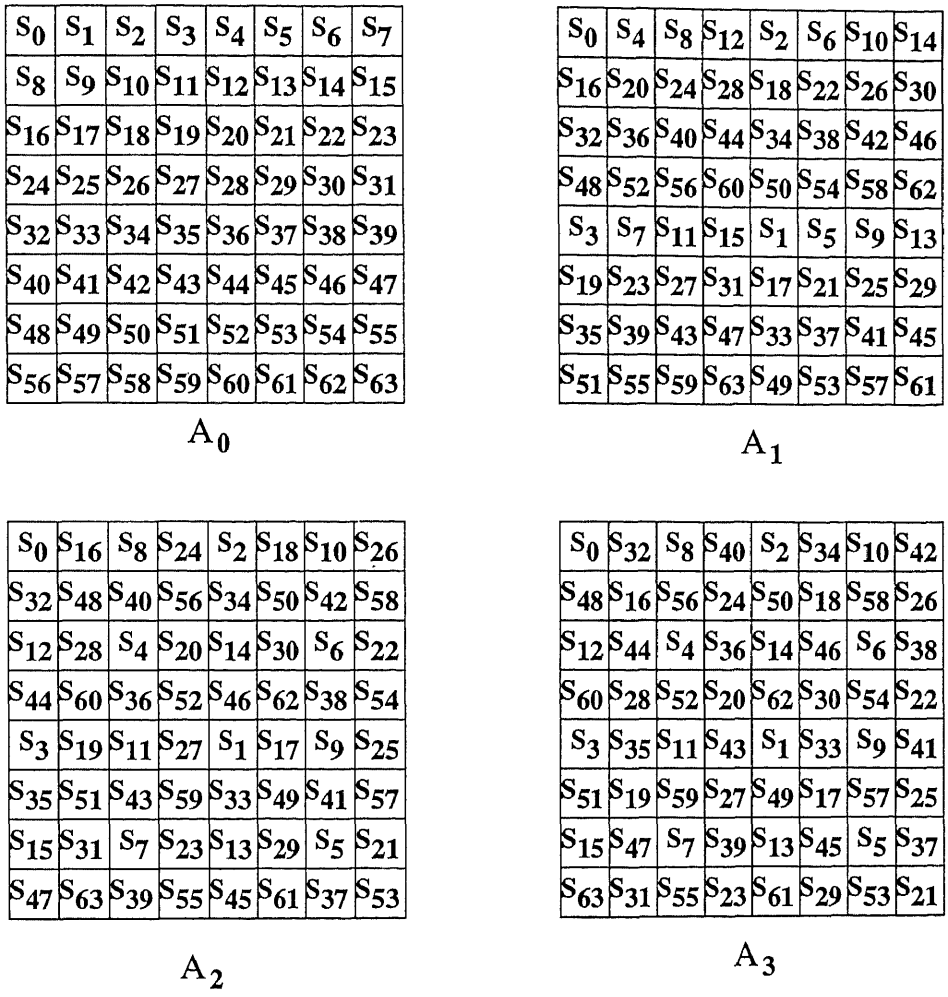


Figure 7.4 Interleaved 2-D Arrays Obtained at Each Partitioning Layer

2 in Figure 7.3) is written to the upper-right quarter of A_1, and the far-right part (with branch label 3 in Figure 7.3) is written to the lower-left quarter of A_1.

- LAYER 2: Each of the four parts is dealt with separately. That is, each of the four quadrants in A_1 is by itself an array that needs to be interleaved. Let us concentrate on one of the four parts and explain how it is done. (The same will apply to the other three parts.) Consider the far-left part with branch label 0 at the first layer. Partition this part into the four 4-symbol parts, with the following symbol indexes (0, 16, 32, 48), (4, 20, 36, 52), (8, 24, 40, 56), and (12, 28, 44, 60) with branch labels 0, 1, 2, and 3 respectively; divide the upper-left quarter of the interleaved array into four subquarters; and write the four 4-symbol parts to the four subquarters with the branch labels following the pattern in Equation (7.4). If we repeat the same steps for the remaining three parts obtained at layer 1, we obtain the interleaved layout, A_2, shown in Figure 7.4.

- LAYER 3: Each of the sixteen parts obtained at layer 2 is dealt with separately. Let us concentrate on one of these parts. (The same will apply to the other fifteen parts.) Consider the part with branch label 2 at the first layer and branch label 1 at the second layer. In A_2, this part occupies the 2×2 subarray in the 3^{rd} and 4^{th} row and the 7^{th} and 8^{th} columns. This part is partitioned to the four symbols S_6, S_{22}, S_{38}, and S_{54}, and these four symbols are written to their 2×2 segment of the interleaved array with the branch labels following the pattern in Equation (7.4).

  If we repeat the same steps for the remaining fifteen parts obtained at layer 2, we obtain the interleaved layout, A_3, shown in Figure 7.4. Note that this last layer is needed for the interleaving degree to reach γ_{max}. For the interleaved layout A_1, the interleaving degree obtained is $\gamma_2 = 2^2 = 4$. For the interleaved layout A_2, the interleaving degree obtained is $\gamma_2 = 2^4 = 16$. For the interleaved layout A_3, the interleaving degree obtained is $\gamma_3 = \frac{l^2}{2} = 32$, which is γ_{max}. (Refer to Section 6.3.)

7.2.2 Successive Quaternary Partition Interleaving Algorithm

The successive quaternary partition interleaving algorithm can be summarized as follows:

From a block $A0$ of length $l \times l = 2^p \times 2^p$ symbols, if no interleaving is done, the 2-D layout of the symbols is referred to as a 2-D array, A_0. The interleaved 2-D array, A_m (which consists of 2^{2m} 2-D subarrays), is obtained by utilizing a quaternary tree with m layers (which produce 2^{2m} end leaves). The interleaving gain obtained with A_m is $\gamma_m = 2^{2m}$, where $\gamma_{max} \leq \frac{l^2}{2}$ and $1 \leq m \leq p$. Note that the original block $A0$ is a sequence of l^2 symbols $S_0, S_1, ..., S_{(l^2-1)}$, where the indexes of these symbols are $0, 1, 2, ..., l^2 - 1$. The order of these indexes in A_m is obtained as follows.

- STEP 1: Find the label $(i_1 i_2 i_3 ... i_m)$ of each end leaf in the tree.
  a) With each partition, a node in the tree is split into four branches, the far-left branch is labeled 0, the mid-left branch is labeled 1, the mid-right branch is labeled 2, and the far-right branch is labeled 3.
  b) At the m^{th} layer, follow the labels at the partitioning path to find the end-leaf labels (e.g., if $m = 2$, the sixteen end leaves will have the following labels: $00, 01, 02, 03, 10, 11, 12, 13, 20, 21, 22, 23, 30, 31, 32, 33$.)

- STEP 2: Transfer end-leaf labels to leading-index labels.
  A leading-index label $k_1 k_2 k_3 ... k_m$ is obtained from the end-leaf label $i_1 i_2 i_3 ... i_m$, where k_j is obtained from i_j by

$$k_j = i_j . \gamma_{(j-1)}, \tag{7.5}$$

  where $\gamma_{(j-1)}$ is the interleaving gain at the $(j-1)^{th}$ layer, $0 \leq j \leq m$, and $\gamma_0 = 2^0 = 1$.

- STEP 3: Find the location of each of the 2^{2m} subarrays.
  Based on the end-leaf labels obtained in step 1, find the location of each subarray with respect to A_m. The first digit in the end-leaf label decides which quarter of A_m the subarray falls in (which follows the pattern in Equation (7.4)). The second digit decides which subquarter the subarray falls in (which

is also decided according to the pattern in Equation (7.4)). Continue this until the m^{th} digit is checked, which specifies the exact location of the subarray.

- STEP 4: Find the indexes of the interleaved array symbols.
  For a leading-index label $k_1k_2k_3...k_m$, find the indexes of the $\frac{l^2}{2^{2m}}$ symbols in its subarray. A symbol index is given by

$$k_1 + k_2 + ... + k_m + r\gamma_m, \tag{7.6}$$

  where r is an integer in the range $0 \leq r \leq 2^{(2p-2m)} - 1$. For the first symbol in the subarray, $r = 0$ and for the last symbol in the subarray (the $(\frac{l}{2^m})^{th}$ symbol) $r = \frac{l}{2^{2m}} - 1 = 2^{2p-2m} - 1$. The following 2-D array, R, shows the values r takes in the corresponding interleaved 2-D array (e.g., in the symbol at the first row and first column, $r = 0$; and in the symbol at the first row and second column, $r = 1$).

$$R = \begin{bmatrix} 0 & 1 & ... & 2^{(p-m)} - 1 \\ 2^{(p-m)} & 2^{(p-m)} + 1 & ... & 2^{(p-m+1)} - 1 \\ ... & ... & ... & ... \\ 2^{(2p-2m)} - 2^{(p-m)} & 2^{(2p-2m)} - 2^{(p-m)} + 1 & ... & 2^{(2p-2m)} - 1 \end{bmatrix}. \tag{7.7}$$

7.2.3 Example

$l = 8$, $p = 3$, and there are 64 symbols.

For the zero layer, $m = 0$, and

$2p - 2m = 6 - 0 = 6$ $\qquad$ $0 \leq r \leq 2^6 - 1 = 63$ $\qquad$ $\gamma_0 = 2^0 = 1$.

There is no interleaving, and the 64-symbol block is written to the 8×8 2-D layout row-by-row. The indexes of the 2-D layout of the symbols are

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 \\ 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 \\ 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix}. \tag{7.8}$$

For the first layer, $m = 1$, there are four subarrays, and

$2p - 2m = 6 - 2 = 4 \qquad 0 \le r \le 2^4 - 1 = 15 \qquad \gamma_1 = 2^2 = 4.$

The labels of the four end-leaves are (0, 1, 2, 3).

The leading-index labels are (0, 1, 2, 3).

The indexes of the interleaved array symbols are

$$\left[\begin{array}{cccc|cccc} 0 & 4 & 8 & 12 & 2 & 6 & 10 & 14 \\ 16 & 20 & 24 & 28 & 18 & 22 & 26 & 30 \\ 32 & 36 & 40 & 44 & 34 & 38 & 42 & 46 \\ 48 & 52 & 56 & 60 & 50 & 54 & 58 & 62 \\ - & - & - & - & - & - & - & - \\ 3 & 7 & 11 & 15 & 1 & 5 & 9 & 13 \\ 19 & 23 & 27 & 31 & 17 & 21 & 25 & 29 \\ 35 & 39 & 43 & 47 & 33 & 37 & 41 & 45 \\ 51 & 55 & 59 & 63 & 49 & 53 & 57 & 61 \end{array}\right]. \tag{7.9}$$

For the second layer, $m = 2$, there are sixteen subarrays, and

$2p - 2m = 6 - 4 = 2 \qquad 0 \le r \le 2^2 - 1 = 3 \qquad \gamma_2 = 2^4 = 16.$

The labels of the sixteen end-leaves are

$$\begin{array}{l} (\ 0\ 0,\ \ 0\ 1,\ \ 0\ 2,\ \ 0\ 3,\ \ 1\ 0,\ \ 1\ 1,\ \ 1\ 2,\ \ 1\ 3, \\ \quad 2\ 0,\ \ 2\ 1,\ \ 2\ 2,\ \ 2\ 3,\ \ 3\ 0,\ \ 3\ 1,\ \ 3\ 2,\ \ 3\ 3\ ). \end{array} \tag{7.10}$$

The leading-index labels are

$$\begin{array}{l} (\ 0\ 0,\ \ 0\ 4,\ \ 0\ 8,\ \ 0\ 12,\ \ 1\ 0,\ \ 1\ 4,\ \ 1\ 8,\ \ 1\ 12, \\ \quad 2\ 0,\ \ 2\ 4,\ \ 2\ 8,\ \ 2\ 12,\ \ 3\ 0,\ \ 3\ 4,\ \ 3\ 8,\ \ 3\ 12\ ). \end{array} \tag{7.11}$$

The indexes of the interleaved array symbols are

$$\left[\begin{array}{cc|cc|cc|cc} 0 & 16 & 8 & 24 & 2 & 18 & 10 & 26 \\ 32 & 48 & 40 & 56 & 34 & 50 & 42 & 58 \\ - & - & - & - & - & - & - & - \\ 12 & 28 & 4 & 20 & 14 & 30 & 6 & 22 \\ 44 & 60 & 36 & 52 & 46 & 62 & 38 & 54 \\ - & - & - & - & - & - & - & - \\ 3 & 19 & 11 & 27 & 1 & 17 & 9 & 25 \\ 35 & 51 & 43 & 59 & 33 & 49 & 41 & 57 \\ - & - & - & - & - & - & - & - \\ 15 & 31 & 7 & 23 & 13 & 29 & 5 & 21 \\ 47 & 63 & 39 & 55 & 45 & 61 & 37 & 53 \end{array}\right]. \tag{7.12}$$

For the third layer, $m = 3$, there are 64 subarrays, and

$2p - 2m = 6 - 6 = 0 \qquad 0 \le r \le 2^0 - 1 = 0 \qquad \gamma_3 = \frac{l^2}{2} = 32.$

Note that $r = 0$, which is expected since, in this last layer, each subarray has only one symbol. The labels of the sixty four end-leaves are

$$\begin{array}{cccccccccccccccc}
0\,0\,0, & 0\,0\,1, & 0\,0\,2, & 0\,0\,3, & 0\,1\,0, & 0\,1\,1, & 0\,1\,2, & 0\,1\,3, \\
0\,2\,0, & 0\,2\,1, & 0\,2\,2, & 0\,2\,3, & 0\,3\,0, & 0\,3\,1, & 0\,3\,2, & 0\,3\,3, \\
1\,0\,0, & 1\,0\,1, & 1\,0\,2, & 1\,0\,3, & 1\,1\,0, & 1\,1\,1, & 1\,1\,2, & 1\,1\,3, \\
1\,2\,0, & 1\,2\,1, & 1\,2\,2, & 1\,2\,3, & 1\,3\,0, & 1\,3\,1, & 1\,3\,2, & 1\,3\,3, \\
2\,0\,0, & 2\,0\,1, & 2\,0\,2, & 2\,0\,3, & 2\,1\,0, & 2\,1\,1, & 2\,1\,2, & 2\,1\,3, \\
2\,2\,0, & 2\,2\,1, & 2\,2\,2, & 2\,2\,3, & 2\,3\,0, & 2\,3\,1, & 2\,3\,2, & 2\,3\,3, \\
3\,0\,0, & 3\,0\,1, & 3\,0\,2, & 3\,0\,3, & 3\,1\,0, & 3\,1\,1, & 3\,1\,2, & 3\,1\,3, \\
3\,2\,0, & 3\,2\,1, & 3\,2\,2, & 3\,2\,3, & 3\,3\,0, & 3\,3\,1, & 3\,3\,2, & 3\,3\,3.
\end{array} \tag{7.13}$$

The leading-index labels are

$$\begin{array}{cccccccccccccccc}
0\,0\,0, & 0\,0\,16, & 0\,0\,32, & 0\,0\,48, & 0\,4\,0, & 0\,4\,16, & 0\,4\,32, & 0\,4\,48, \\
0\,8\,0, & 0\,8\,16, & 0\,8\,32, & 0\,8\,48, & 0\,12\,0, & 0\,12\,16, & 0\,12\,32, & 0\,12\,48, \\
1\,0\,0, & 1\,0\,16, & 1\,0\,32, & 1\,0\,48, & 1\,4\,0, & 1\,4\,16, & 1\,4\,32, & 1\,4\,48, \\
1\,8\,0, & 1\,8\,16, & 1\,8\,32, & 1\,8\,48, & 1\,12\,0, & 1\,12\,16, & 1\,12\,32, & 1\,12\,48, \\
2\,0\,0, & 2\,0\,16, & 2\,0\,32, & 2\,0\,48, & 2\,4\,0, & 2\,4\,16, & 2\,4\,32, & 2\,4\,48, \\
2\,8\,0, & 2\,8\,16, & 2\,8\,32, & 2\,8\,48, & 2\,12\,0, & 2\,12\,16, & 2\,12\,32, & 2\,12\,48, \\
3\,0\,0, & 3\,0\,16, & 3\,0\,32, & 3\,0\,48, & 3\,4\,0, & 3\,4\,16, & 3\,4\,32, & 3\,4\,48, \\
3\,8\,0, & 3\,8\,16, & 3\,8\,32, & 3\,8\,48, & 3\,12\,0, & 3\,12\,16, & 3\,12\,32, & 3\,12\,48.
\end{array} \tag{7.14}$$

The indexes of the interleaved array symbols are

$$\left[\begin{array}{c|c|c|c|c|c|c|c}
0 & 32 & 8 & 40 & 2 & 34 & 10 & 42 \\ \hline
48 & 16 & 56 & 24 & 50 & 18 & 58 & 26 \\ \hline
12 & 44 & 4 & 36 & 14 & 46 & 6 & 38 \\ \hline
60 & 28 & 52 & 20 & 62 & 30 & 54 & 22 \\ \hline
3 & 35 & 11 & 43 & 1 & 33 & 9 & 41 \\ \hline
51 & 19 & 59 & 27 & 49 & 17 & 57 & 25 \\ \hline
15 & 47 & 7 & 39 & 13 & 45 & 5 & 37 \\ \hline
63 & 31 & 55 & 23 & 61 & 29 & 53 & 21
\end{array}\right]. \tag{7.15}$$

On can see that the layouts in Equations (7.8), (7.9), (7.12), and (7.15) match the indexes in A_0, A_1, A_2, and A_3 in Figure 7.4.

7.3 3-D Interleaving Using an Octonary Partitioning Tree

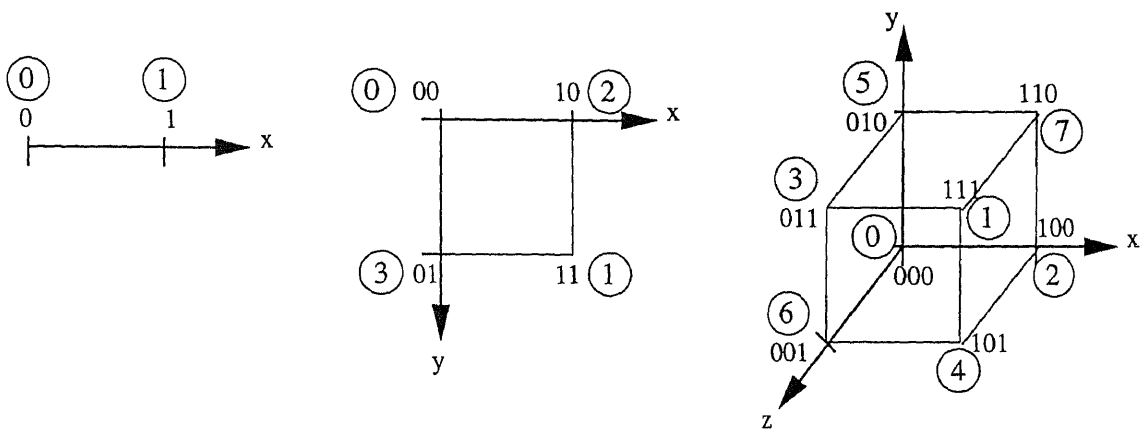
This section explains successive 3-D interleaving using an octonary partitioning tree. As with the 1-D and 2-D cases, a tutorial of the technique is provided, followed by the implementation algorithm.

7.3.1 Octonary Partitioning Tree Technique

This section shows how the 3-D interleaving case introduced in Section 6.4 can be explained using a successive partitioning approach. First, the way in which the interleaving principle is adapted from the 1-D to the 2-D and 3-D cases should be explained. As shown in Figure 7.5, in the 1-D case the binary partitioning pattern can be expressed as two points over the x axis. The first point has the coordinate 0 and the second point has the coordinate 1. (In Figure 7.5, the small font indicates the coordinate, while the large font in the circle indicates the symbol index). Moving to the 2-D case, the quaternary partitioning pattern can be expressed as the four points 0, 1, 2, and 3, with the coordinates 00, 11, 10, and 01, respectively. This pattern is expressed in the 2-D array in Equation (7.4), The essence of this pattern is to maximize the distance between each pair of indexes (0, 1 and 2, 3). In the 3-D case, the octonary partitioning pattern is expressed with the 8 points 0, 1, 2, 3, 4, 5, 6, and 7, with the coordinates 000, 111, 100, 011, 101, 010, 001, and 110, respectively. As an analogy to Equation (7.4), this pattern can be expressed in the following pair of 2-D arrays:

$$\begin{bmatrix} 3 & 1 \\ 6 & 4 \end{bmatrix} \& \begin{bmatrix} 5 & 7 \\ 0 & 2 \end{bmatrix}, \tag{7.16}$$

where the first 2-D array is the front layer (face) of the cube and the second 2-D array is the back layer of the cube. See Figure 7.5(c). Similar to the 2-D case, the essence of this pattern is to maximize the distance between each pair (0,1; 2,3; 4,5; and 6,7).



(a) Binary (b) Quaternary (c) Octanry

Figure 7.5 Maximizing the Distance Between Each Pair of Symbols in 1-D, 2-D, and 3-D Cases (small font indicates the coordinate; large font in the circle indicates the symbol index).

For a block $A0$ of length 64 symbols, if no interleaving is done, the block will be written to a $4 \times 4 \times 4$ cube row-by-row, and layer-by-layer, as shown in A_0 in Figure 7.6. The bold font in the figure indicates the layer number. Figure 7.7(a) shows two layers of the octonary partitioning tree, which will be used to interleave $A0$ in 3-D utilizing the pattern in Equation (7.16). The octonary successive partition interleaving is done in layers as follows:

- LAYER 1: The block is divided into eight parts, and each part is assigned to a branch. Branches are labeled 0, 1, 2, 3, 4, 5, 6, and 7 at the first layer of the tree, as shown in Figure 7.7(b).

  The interleaved 3-D array obtained from the first partitioning layer is A_1 (shown in Figure 7.8) and the interleaving gain obtained is $\gamma_1 = 2^3 = 8$. A_1 is obtained by writing the eight parts resulting from the first partition of the interleaving tree to A_1 such that one part is written to one corner with the branch labels following the pattern in Equation (7.16). For example, the far-left part (with branch label 0 in Figure 7.7) is written to the back-bottom-left corner of the cube. The resulting interleaved cube is shown in Figure 7.8. The bottom part of the figure shows how each of the eight corners of the cube is laid out.

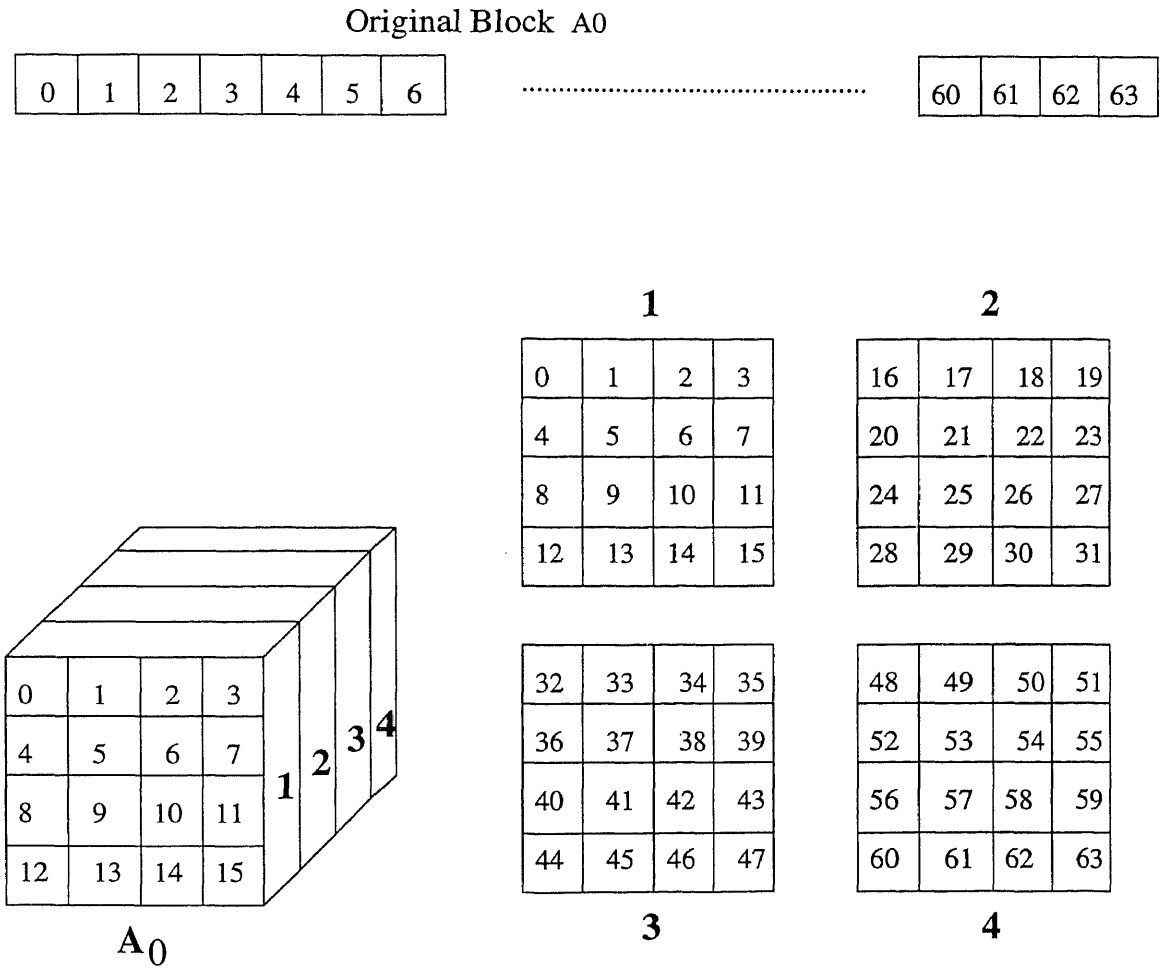
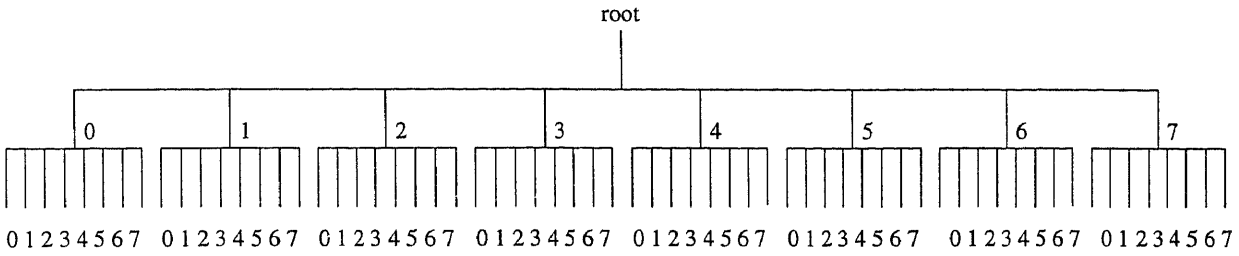
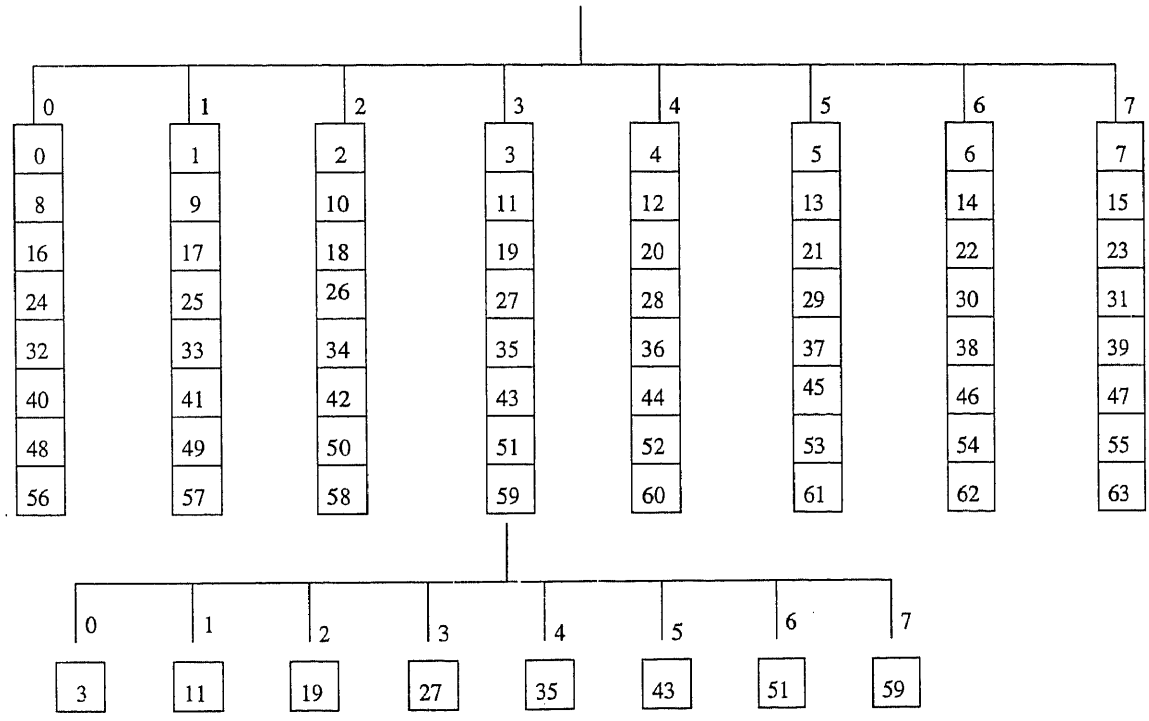


Figure 7.6 The Layout of 64 Symbols in a $4 \times 4 \times 4$ Cube without Interleaving



(a) Two-Layer Octonary Tree



(b) Octonary Tree Partitioning

Figure 7.7 Octonary Partitioning Tree for a 64-Symbol Block

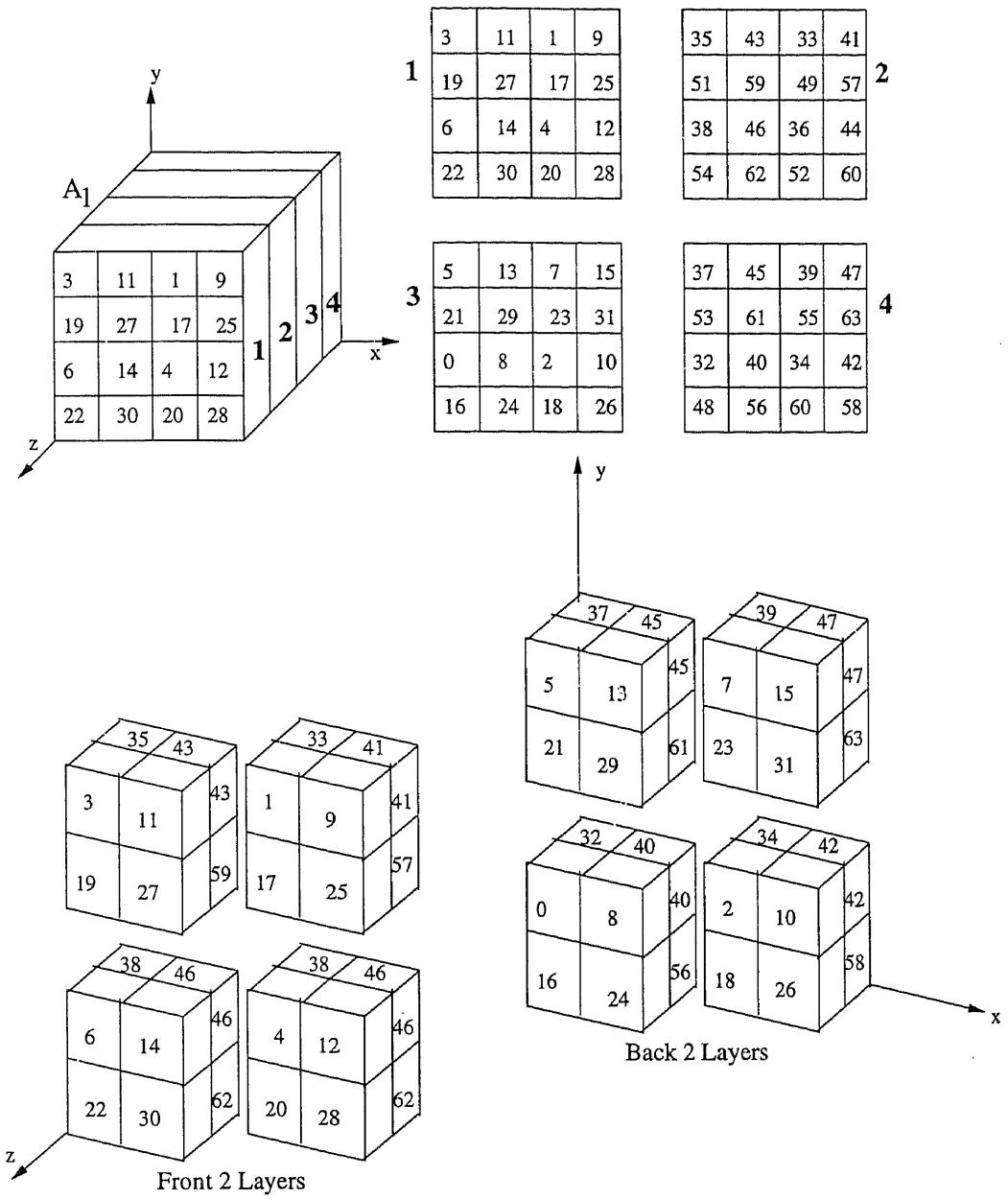


Figure 7.8 The Layout of 64 Symbols in a $4 \times 4 \times 4$ Cube at the First Layer

- LAYER 2: Each of the eight parts obtained at layer 1 is dealt with separately. Let us concentrate on one of these parts. (The same will apply to the other seven parts.) Consider the part with the branch label 3 at the first layer (which has symbols with indexes 3, 11, 19, 27, 35, 43, 51, and 59. This part occupies the $2 \times 2 \times 2$ subarray in the front-upper-left corner of the 3-D array. This part is partitioned to eight symbols, and these eight symbols are written to their $2 \times 2 \times 2$ subarray following the pattern in Equation (7.16). If we repeat the same step for the remaining seven parts, we obtain the interleaved layout, A_2, as shown in Figure 7.9.

  Note that this last layer is needed for the interleaving degree to reach γ_{max}. For the interleaved layout A_1, the interleaving degree obtained is $\gamma_1 = 2^3 = 8$. For the interleaved layout A_2, the interleaving degree obtained is $\gamma_{max} = \gamma_2 = \frac{l^3}{2} = \frac{64}{2} = 32$.

7.3.2 Successive Octonary Partition Interleaving Algorithm

The successive octonary partition interleaving algorithm can be summarized as follows:

For a block $A0$ of length $l \times l \times l = 2^p \times 2^p \times 2^p$ symbols, if no interleaving is done the 3-D layout of symbols is referred to as a 3-D array (cube), A_0. The interleaved 3-D array, A_m, (which consists of 2^{3m} 3-D subarrays), is obtained by utilizing an octonary tree with m layers (which produces 2^{3m} end leaves). The interleaving gain obtained with A_m is $\gamma_m = 2^{3m}$, where $\gamma_{max} \leq \frac{l^3}{2}$ and $1 \leq m \leq p$. Note that the original block $A0$ is a sequence of l^3 symbols $S_0, S_1, ..., S_{(l^3-1)}$, where the indexes of these symbols are $0, 1, 2, ..., l^3 - 1$. The order of these indexes in A_m is obtained as follows.

- STEP 1: Find the label $(i_1 i_2 i_3 ... i_m)$ of each end leaf in the tree.

  a) With each partition, a node in the tree is split into eight branches, which are labeled 0-7 from left to right.

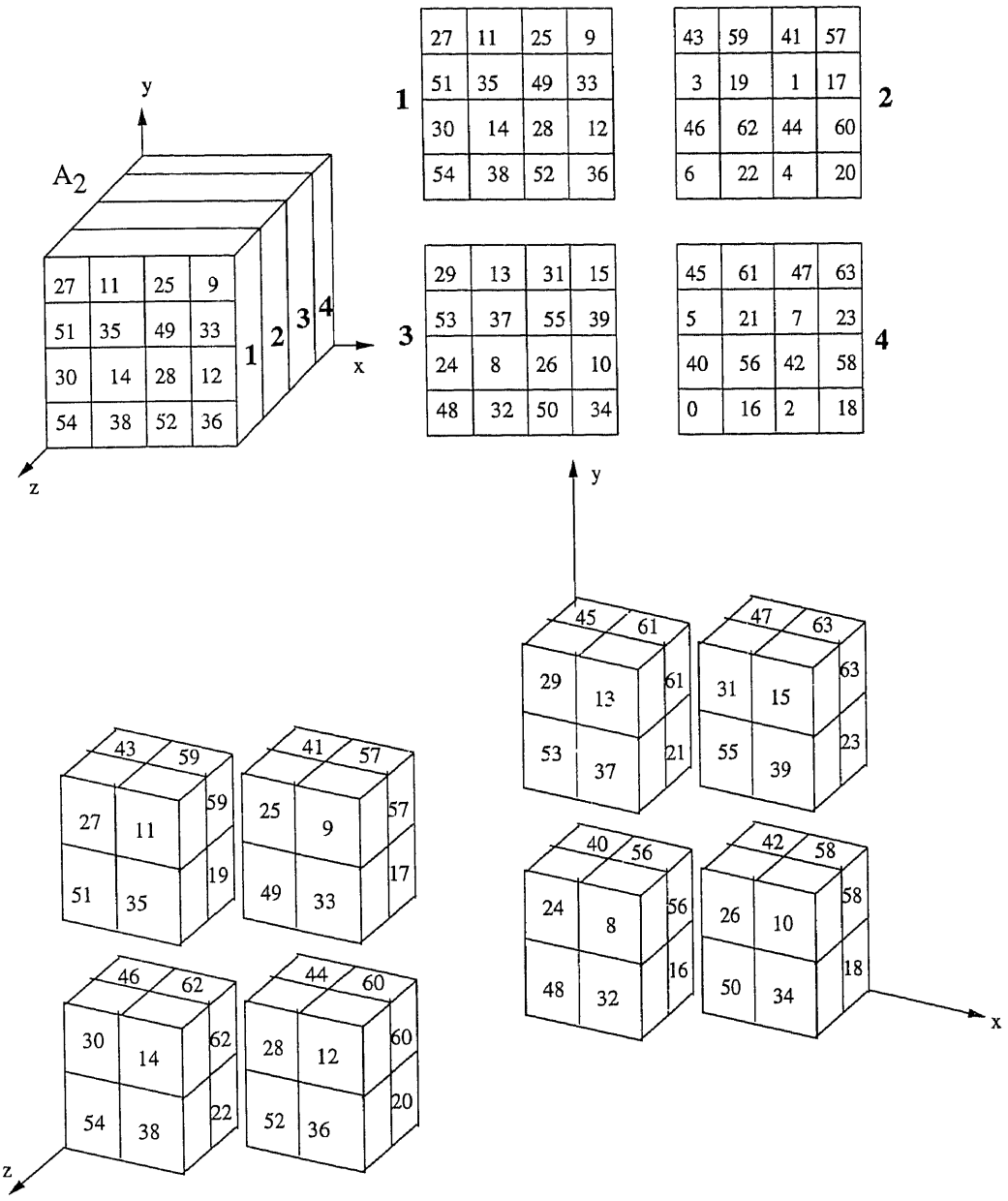


Figure 7.9 The Layout of 64 Symbols in a $4 \times 4 \times 4$ Cube at the Second Layer

b) At the m^{th} layer, follow the labels at the partitioning path to find the end-leaf labels (e.g., if $m = 2$, the sixty four end leaves will be as shown in Figure 7.7(a).

- STEP 2: Transfer the end-leaf label to a leading-index label.

  A leading-index label $k_1k_2k_3...k_m$ is obtained from the end leaf label $i_1i_2i_3...i_m$, where k_j is obtained from i_j by

$$k_j = i_j.\gamma_{(j-1)}, \tag{7.17}$$

  where $\gamma_{(j-1)}$ is the interleaving gain at the $(j-1)^{th}$ layer, $0 \leq j \leq m$, and $\gamma_0 = 2^0 = 1$.

- STEP 3: Find the location of each of the 2^{3m} subcubes.

  Based on the end-leaf labels obtained in step 1, find the location of each subcube with respect to A_m. The first digit in the end-leaf label decides which corner of A_m the subcube falls in (which follows the pattern in Equation (7.16)). The second digit decides which subcorner the subcube falls in (which is also decided according to the pattern in Equation (7.16)). Continue this until the m^{th} digit is checked, which specifies the exact location of the examined subcube.

- STEP 4: Find the indexes of the interleaved cube symbols.

  For a leading-index label $k_1k_2k_3...k_m$, find the indexes of the $\frac{l^3}{2^{3m}}$ symbols in its subarray. A symbol index is given by

$$k_1 + k_2 + ... + k_m + r\gamma_m, \tag{7.18}$$

  where r is an integer in the range $0 \leq r \leq 2^{(3p-3m)} - 1$. The following group of 2-D arrays, R, shows the values r takes in the corresponding interleaved 3-D cube. We use the same layout as explained with Figure 7.6.

$$R = \begin{bmatrix} 0 & 1 & ... & 2^{(p-m)} - 1 \\ 2^{(p-m)} & 2^{(p-m)} + 1 & ... & 2^{(p-m)+1} - 1 \\ ... & ... & ... & ... \\ 2^{(2p-2m)} - 2^{(p-m)} & 2^{(2p-2m)} - 2^{(p-m)} + 1 & ... & 2^{(2p-2m)} - 1 \end{bmatrix}$$

$$\& \begin{bmatrix} 2^{(2p-2m)} & 2^{(2p-2m)} + 1 & ... & 2^{(2p-2m)} + 2^{(p-m)} - 1 \\ 2^{(2p-2m)} + 2^{(p-m)} & 2^{(2p-2m)} + 2^{(p-m)} + 1 & ... & 2^{(2p-2m)} + 2^{(p-m+1)} - 1 \\ ... & ... & ... & ... \\ 2^{(2p-2m)} + 2^{(p-m+1)} & 2^{(2p-2m)} + 2^{(p-m+1)} + 1 & ... & 2^{(2p-2m+1)} - 1 \end{bmatrix}$$

$$
\&\begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \& \quad \cdots
$$

$$
\&\begin{bmatrix} 2^{(3p-3m)} - 2^{(2p-2m)} & \cdots & \cdots & 2^{(3p-3m)} - 2^{(p-m)} - 1 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ 2^{(3p-3m)} - 2^{(p-m)} & 2^{(3p-3m)} - 2^{(p-m)} + 1 & \cdots & 2^{(3p-3m)} - 1 \end{bmatrix} \tag{7.19}
$$

7.3.3 Example

$l = 4$, $p = 2$, and there are 64 symbols.

For the zero layer, $m = 0$, and

$3p - 3m = 6 - 0 = 6 \qquad 0 \le r \le 2^6 - 1 = 63 \qquad \gamma_0 = 2^0 = 1.$

There is no interleaving, and the 64-symbol block is written to the $4 \times 4 \times 4$ 3-D layout row-by-row, and layer-by-layer. The indexes of 3-D layout of the symbols are

$$
\begin{aligned} A_0 \;=\; & \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix} \& \begin{bmatrix} 16 & 17 & 18 & 19 \\ 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 \\ 28 & 29 & 30 & 31 \end{bmatrix} \\ & \& \begin{bmatrix} 32 & 33 & 34 & 35 \\ 36 & 37 & 38 & 39 \\ 40 & 41 & 42 & 43 \\ 44 & 45 & 46 & 47 \end{bmatrix} \& \begin{bmatrix} 48 & 49 & 50 & 51 \\ 52 & 53 & 54 & 55 \\ 56 & 57 & 58 & 59 \\ 60 & 61 & 62 & 63 \end{bmatrix}, \end{aligned} \tag{7.20}
$$

where these four 2-D arrays express the layout in the four layers of the cube. This layout agrees with Figure 7.6

For the first layer, $m = 1$, there are eight subcubes, and

$3p - 3m = 6 - 3 = 3 \qquad 0 \le r \le 2^3 - 1 = 7 \qquad \gamma_1 = 2^3 = 8.$

The labels of the eight end-leaves are (0, 1, 2, 3, 4, 5, 6, 7).

The leading-index labels are (0, 1, 2, 3, 4, 5, 6, 7).

The indexes of the interleaved cube symbols are

$$
A_1 \;=\; \left[\begin{array}{cc|cc} 3 & 11 & 1 & 9 \\ 19 & 27 & 17 & 25 \\ \hline 6 & 14 & 4 & 12 \\ 22 & 30 & 20 & 28 \end{array}\right] \& \left[\begin{array}{cc|cc} 35 & 43 & 33 & 41 \\ 51 & 59 & 49 & 57 \\ \hline 38 & 46 & 36 & 44 \\ 54 & 62 & 52 & 60 \end{array}\right]
$$

$$\&\left[\begin{array}{cc|cc} 5 & 13 & 7 & 15 \\ 21 & 29 & 23 & 31 \\ \hline 0 & 8 & 2 & 10 \\ 16 & 24 & 18 & 26 \end{array}\right]\&\left[\begin{array}{cc|cc} 37 & 45 & 39 & 47 \\ 53 & 61 & 55 & 63 \\ \hline 32 & 40 & 34 & 42 \\ 48 & 56 & 60 & 58 \end{array}\right], \tag{7.21}$$

which agrees with the layout of A_1 in Figure 7.8.

For the second layer, $m = 2$, there are 64 subcubes, and

$3p - 3m = 6 - 6 = 0 \qquad 0 \le r \le 2^0 - 1 = 0 \qquad \gamma_2 = \frac{l^3}{2} = 32.$

The labels of the 64 end leaves are

$$\begin{array}{llllllll} 0\ 0, & 0\ 1, & 0\ 2, & 0\ 3, & 0\ 4, & 0\ 5, & 0\ 6, & 0\ 7, \\ 1\ 0, & 1\ 1, & 1\ 2, & 1\ 3, & 1\ 4, & 1\ 5, & 1\ 6, & 1\ 7, \\ 2\ 0, & 2\ 1, & 2\ 2, & 2\ 3, & 2\ 4, & 2\ 5, & 2\ 6, & 2\ 7, \\ 3\ 0, & 3\ 1, & 3\ 2, & 3\ 3, & 3\ 4, & 3\ 5, & 3\ 6, & 3\ 7, \\ 4\ 0, & 4\ 1, & 4\ 2, & 4\ 3, & 4\ 4, & 4\ 5, & 4\ 6, & 4\ 7, \\ 5\ 0, & 5\ 1, & 5\ 2, & 5\ 3, & 5\ 4, & 5\ 5, & 5\ 6, & 5\ 7, \\ 6\ 0, & 6\ 1, & 6\ 2, & 6\ 3, & 6\ 4, & 6\ 5, & 6\ 6, & 6\ 7, \\ 7\ 0, & 7\ 1, & 7\ 2, & 7\ 3, & 7\ 4, & 7\ 5, & 7\ 6, & 7\ 7. \end{array} \tag{7.22}$$

The leading-index labels are

$$\begin{array}{llllllll} 0\ 0, & 0\ 8, & 0\ 16, & 0\ 24, & 0\ 32, & 0\ 40, & 0\ 48, & 0\ 56, \\ 1\ 0, & 1\ 8, & 1\ 16, & 1\ 24, & 1\ 32, & 1\ 40, & 1\ 48, & 1\ 56, \\ 2\ 0, & 2\ 8, & 2\ 16, & 2\ 24, & 2\ 32, & 2\ 40, & 2\ 48, & 2\ 56, \\ 3\ 0, & 3\ 8, & 3\ 16, & 3\ 24, & 3\ 32, & 3\ 40, & 3\ 48, & 3\ 56, \\ 4\ 0, & 4\ 8, & 4\ 16, & 4\ 24, & 4\ 32, & 4\ 40, & 4\ 48, & 4\ 56, \\ 5\ 0, & 5\ 8, & 5\ 16, & 5\ 24, & 5\ 32, & 5\ 40, & 5\ 48, & 5\ 56, \\ 6\ 0, & 6\ 8, & 6\ 16, & 6\ 24, & 6\ 32, & 6\ 40, & 6\ 48, & 6\ 56, \\ 7\ 0, & 7\ 8, & 7\ 16, & 7\ 24, & 7\ 32, & 7\ 40, & 7\ 48, & 7\ 56. \end{array} \tag{7.23}$$

The indexes of the interleaved cube are

$$A_2 = \left[\begin{array}{c|c|c|c} 27 & 11 & 25 & 9 \\ \hline 51 & 35 & 49 & 33 \\ \hline 30 & 14 & 28 & 12 \\ \hline 54 & 38 & 52 & 36 \end{array}\right] \& \left[\begin{array}{c|c|c|c} 43 & 59 & 41 & 57 \\ \hline 3 & 19 & 1 & 17 \\ \hline 46 & 62 & 44 & 60 \\ \hline 6 & 22 & 4 & 20 \end{array}\right]$$
$$\& \left[\begin{array}{c|c|c|c} 29 & 13 & 31 & 15 \\ \hline 53 & 37 & 55 & 39 \\ \hline 24 & 8 & 26 & 10 \\ \hline 48 & 32 & 50 & 34 \end{array}\right] \& \left[\begin{array}{c|c|c|c} 45 & 61 & 47 & 63 \\ \hline 5 & 21 & 7 & 23 \\ \hline 40 & 56 & 42 & 58 \\ \hline 0 & 16 & 2 & 18 \end{array}\right], \quad (7.24)$$

which agrees with the layout of A_2 in Figure 7.9.

7.4 n-D Interleaving Using 2^n Partitioning Tree

This section shows how the interleaving algorithm can be generalized to any n dimension. Although no known specific application requires more than 3-D interleaving, it is appropriate to mention a generalization of the algorithm in this context. From the 1-D, 2-D, and 3-D cases discussed in Sections 7.1 through Section 7.3, one can see that to interleave for any dimension, the following are needed:

1. The partitioning tree (binary, quaternary, octonary, ...).
2. The interleaving pattern (which is expressed in Equations (7.4) and (7.16) for the 2-D case and the 3-D case, respectively).

According to the algorithms in Sections 7.1.2, 7.2.2, and 7.3.2, the partitioning tree will give us the labels of the end leaves (step 1), which gives us the leading-index labels (step 2). The interleaving pattern will decide the location of each part (step 3). From the interleaving gain (which is $\gamma_m = 2^{nm}$ and $\gamma_{max} \leq \frac{l^n}{2}$) and the leading-index labels, the indexes of the interleaved symbols are found (step 4). Thus, for this general case (n-D interleaving), it is sufficient to show how to establish the partitioning tree and how to find the interleaving pattern.

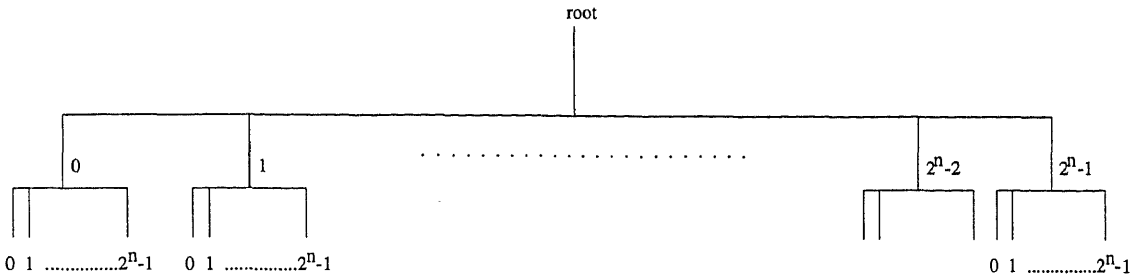


Figure 7.10 2^n Partitioning Tree

7.4.1 Partitioning Tree for n-D Interleaving

For the 1-D case, a binary tree was used; for the 2-D case, a quaternary tree was used; and for the 3-D case, an octonary tree was used. One can see that, in general, to interleave in n-D, we need to utilize a 2^n partitioning tree. This generalization can be explained as follows: for each axis (dimension) there are two possibilities (see Figure 7.5), a zero coordinate with respect to this axis, or a positive coordinate. For n-D interleaving (n-axes), all possible combinations of the coordinates are 2^n. Since each partitioned part from the tree follows one of these 2^n coordinates, we would need the tree to partition a given block into 2^n parts. Figure 7.10 shows a 2^n partitioning tree.

7.4.2 Interleaving Pattern for n-D Interleaving

Recall the beginning of Section 7.3, where the interleaving principle was adapted from the 1-D case to the 2-D and 3-D cases. This principle attempts to maximize the distance between each pair of symbols. At higher dimensions, the same principle should be applied. In the 3-D case, note the following:

1. The distance between the symbol with index 0 and the symbol with index 1 is maximized by assigning the coordinate 000 to the symbol with index 0, and assigning the coordinate 111 to the symbol with index 1. This can be generalized to any dimension.

2. The coordinate assigned to the symbol with index 2 differs by $(3-1=2)$ bits from the coordinate of the symbol with index 1. This principle will also be generalized to the n-D case.

3. Step 2 applies to the rest of the eight coordinates: each pair of symbols have coordinates that differ by 3 bits; while moving from one pair to the next, the coordinates differ by 2 bits (if possible). We will generalize this to the n-D case, as well.

Shown here is a search algorithm that generates these coordinates. The algorithm can be implemented via a computer program to generate the pattern for the n-D case. Here are the steps of this algorithm.

For the 2^n branches of the partitioning pattern, we have $\frac{2^n}{2} = 2^{(n-1)}$ pairs. Let us refer to these $2^{(2n-1)}$ pairs as $P0$, $P1$, ..., Pi, ... Pa where $a = 2^{(2n-1)} - 1$. The two coordinates of pair Pi are referred to as Pi_0 and Pi_1. The algorithm proceeds as follows:

a) Assign the all-zero and the all-one coordinates to the first pair. That is, $P0_0 =$ 00..0 and $P0_1 = 11..1$.

b) Starting from $i = 1$ to $i = 2^{(2n-1)} - 1$, apply the following:

- From the first coordinate in the $i-1$ pair, which is $P(i-1)_0$, toggle (change) a combination of T bits of this coordinate to obtain Pi_0. T is obtained from n through the following relation:

$$T = \lfloor \frac{n}{2} \rfloor, \tag{7.25}$$

  where $\lfloor x \rfloor$ means the nearest integer less than or equal to x.

- If the obtained Pi_0 was not used before, Pi_1 is obtained as the complement of Pi_0, and the search algorithm moves to the next pair. If the obtained Pi_0 was used in any of the previous $i-1$ pairs, reverse the toggled bits back, and toggle another combination of T bits. If all possible combinations of T bits have been tried, and the algorithm could not obtain a pair, the algorithm toggles $T-1$ bits instead of T bits. (In this search, the algorithm toggles $T-1$ bits only if

toggling T bits did not succeed). Continue this search until all of the $2^{(2n-1)}$ pairs of the pattern are decided.

7.4.3 Example

$n = 4$ (4-D pattern).

The partitioning tree will have $2^4 = 16$ branches.

$T = \lfloor \frac{4}{2} \rfloor = 2$, $T - 1 = 1$.

The above algorithm generates 16 coordinates (8 pairs) as follows:

- $P0_0 = 0000$, $P0_1 = 1111$
- $P1_0 = 1100$, $P1_1 = 0011$
- $P2_0 = 1010$, $P2_1 = 0101$
- $P3_0 = 0110$, $P3_1 = 1001$
- $P4_0 = 1110$, $P4_1 = 0001$
- $P5_0 = 0010$, $P5_1 = 1101$
- $P6_0 = 0111$, $P6_1 = 1000$
- $P7_0 = 1011$, $P7_1 = 0100$

Thus, the resulting coordinates for the 4-D interleaving pattern are 0000,1111; 1100,0011; 1010,0101; 0110,1001; 1110,0001; 0010,1101; 0111,1000; 1011,0100.

7.5 Sliding-Window and Successive-Partitioning Implementations

This section compares the implementation of the successive partitioning technique with the implementation of the sliding window technique (discussed in Chapter 6). This comparison is shown using the 2-D interleaving example of 16 symbols in a 4×4 layout. The case of $m = 2$, which gives an interleaving degree $\gamma_2 = \gamma_{max} = 16/2 = 8$, is shown.

With the sliding window technique, the sliding window at the m^{th} layer requires finding the q-array (see Section 6.3.2). The number of elements in the q-array decides

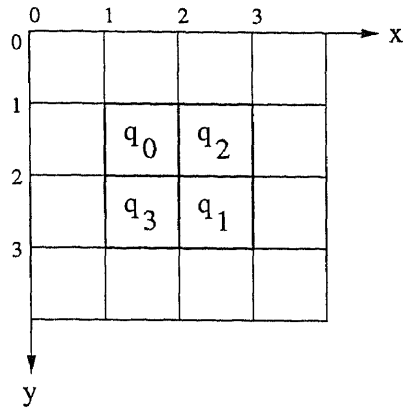


Figure 7.11 The Coordinates of the Sliding Window Indexes: 1,1; 2,2; 2,1; and 1,2 for q_0, q_1, q_2, and q_3, Respectively

the amount of memory needed by the interleaver. For the case of interleaving for a 4×4 layout, the q-array (the indexing array) is of size 2×2 (see Figure 6.6(b)). Each of these four indexes in the q-array requires knowing the x-coordinate and the y-coordinate, as shown in Figure 7.11. Thus, the number of memory elements needed for indexing is eight (each index needs two memory elements for the x-coordinate and the y-coordinate). For the q-array in Figure 7.11, q_0 has the coordinates (1,1), q_1 has the coordinates (2,2), q_2 has the coordinates (2,1), and q_3 has the coordinates (1,2). These coordinates are considered with respect to the upper-left corner of the area corresponding to each index. The sliding window follows the four positions q_0, q_1, q_2, and q_3 and interleaves four symbols at each position, as explained in Section 6.3.2. The following program (written in C) shows how to implement this case. Note that in this program, the 16-element 1-D array, Org[16], contains the original sequence to be interleaved. The 2-D array, R[4][4], should contain the 2-D layout (the interleaved block in its 2-D format). The 2-D array, q[2][4], contains the four indexes, where each index requires the x-coordinate and the y-coordinate.

```
/* 2-D sliding window interleaving for l = 4 */

#include <stdio.h>
```

```
main(){

int i, j = 0;


/* Original Array */
float Org[16] = { 0.00, 1.00, 2.00, 3.00, 4.00, 5.00, 6.00, 7.00,
                  8.00, 9.00, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0 };

/* Resulting Array */
float R[4][4];

/* Indexing Array which carries the coordinates of each index */
int q[2][4] = { 1, 2, 2, 1,
                1, 2, 1, 2 };

/* Sliding Window Interleaving happens here */
   for(i = 0; i < 4; i++){ /* four positions of the sliding window */
      R[q[0][i]-1][q[1][i]-1] = Org[i*4 + j++];
      R[q[0][i]+1][q[1][i]+1] = Org[i*4 + j++];
      R[q[0][i]+1][q[1][i]-1] = Org[i*4 + j++];
      R[q[0][i]-1][q[1][i]+1] = Org[i*4 + j];
      j = j - 3;
   }

/* Print the 2-D layout */
   for(i = 0; i < 4; i++){
      for(j = 0; j < 4; j++){
         printf("%f\t", R[j][i]);
      }
      printf("\n");
   }
}
```

With the successive partitioning technique, the algorithm in Section 7.2.2 produces the 16 indexes that map the original block to the interleaved block. The interleaved block is written to the 2-D layout row-by-row and column-by-column. This method of implementation is shown with the following program. Note that in this program, the 16-element array Org[16] contains the original sequence to interleaved. The array R[16] should contain the interleaved block in its 1-D format (it will be written to the 2-D layout row-by-row). The array q[16] contains the 16 indexes

(the order in which the indexes of the original array are mapped to the interleaved array and are obtained by the algorithm in Section 7.2.2.

```
/* 2-D successive partition interleaving for l = 4 */

#include <stdio.h>

main(){

int i;

/* Original Array */
float Org[16] = { 0.00, 1.00, 2.00, 3.00, 4.00, 5.00, 6.00, 7.00,
                  8.00, 9.00, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0 };

/* Resulting Array */
float R[16];

/* Indexing Array */
int q[16] = { 0, 8,  2, 10, 12, 4, 14, 6,
              3, 11, 1, 9,  15, 7, 13, 5 };

/* Interleaving happens here, a simple memory assignment */
   for(i = 0; i < 16; i++){
      R[i] = Org[q[i]];
   }

/* 2-D layout, write the 16 symbols to the 2-D layout row-by-row */
   for(i = 0; i < 16; i++){
      if(i%4 == 0) printf("\n");
         printf("%f\t", R[i]);
   }
   printf("\n");
}
```

Notice that with both cases, the indexes are obtained in advance and the interleaving process uses them to make the implementation as simple as a direct-memory assignment. One can see a tradeoff between memory and speed with these two methods of implementation. Although the sliding window approach uses eight memory elements to store the q-array (which is half of that of the successive approach), this space savings results in a slow (longer) run-time. Where interleaving

occurs, the program has to calculate the resulting array index from the window index. This is not the case with the successive approach, where all the indexes are obtained in advance. In applications where the arrays to be interleaved are very large, this tradeoff between memory and speed can be an important factor in choosing the implementation method.

7.6 Summary

This chapter presented n-D interleaving using a successive partitioning technique. For the 1-D case, a binary partitioning tree was used; for the 2-D case, a quaternary partitioning tree was used; for the 3-D case, an octonary partitioning tree was used. The general case, where interleaving in n-D uses a partitioning tree with 2^n branches, was also shown. A comparison between sliding-window implementation and successive-partitioning implementation was given. At this point we are ready to move to applying 2-D interleaving and 3-D interleaving to the watermark signal.

CHAPTER 8

2-D INTERLEAVING OF THE WATERMARK SIGNAL FOR STILL IMAGES

In this chapter, the 2-D case of the interleaving technique discussed in Chapters 5, 6, and 7 is applied to the watermark signal. It should be noted that when we deal with the watermark signal, some factors should be taken into consideration. Although when we divide the signal into segments in multiple signaling we embed a block of symbols in a similar fashion as the work discussed in Chapters 5, 6, and 7, digital image watermarking has some specific characteristics. Here, the decoder deals with individual bits; not with symbols. That is, the detection stage of the watermark signal has access to each individual bit, and a group of bits forms a symbol. Also, as explained in Chapter 2, because the watermark signal is power constrained, decoding bit-by-bit using repetition codes or any other error correction code gives lower performance than optimum detection techniques (e.g., correlation detection). Because of these two differences, to enhance watermark signal robustness through interleaving, the decoding technique should

- apply an interleaving strategy that suits a power-constrained signal, and
- apply one of the optimum detection techniques discussed in Chapter 4 to the de-interleaved signal.

First, let us define the exact scenario which makes interleaving the watermark signal an advantage. Suppose the image owner "A", who has access to the original image without the watermark signal, I_0, produced the watermarked image, I_1. "A" wants to extract the watermark signal from the image, I_2. I_2 is a version of I_1 that underwent some attacks. The attacker's success in removing the original signal may depend on the texture of the image itself. The attacker may succeed in destroying the signal in certain blocks and have little effect on other blocks. As a result, certain parts of the watermark signal will be entirely destroyed. Interleaving should allow the decoding process of the watermark signal to minimize the probability of symbol decoding error despite these attacks. That is, the watermark signal gives as much

meaningful information as possible despite the fact that part of the signal is entirely destroyed. This chapter illustrates cases where part of the watermark signal is entirely damaged and the rest of the signal suffers more from some background noise, yet the decoding process reveals the embedded sequence of symbols without error. Note that a similar situation (where a burst of errors may occur in the watermark signal) happens if the image undergoes compression. Due to compression, certain blocks of the image may suffer from higher quantization noise than others. Interleaving enhances the robustness of the signal by decreasing the probability of symbol decoding error. In this chapter we will decode a watermark signal that suffers from a combination of burst errors and quantization noise.

Note that if the watermark signal does not carry a sequence of meaningful information (it is just one sequence of random variables), interleaving is irrelevant. For a watermark signal composed of all the embedded bits, it does not matter if the error is burst or random. The robustness of this signal is decided simply by the signal dimension and the SNR. This interleaving of the watermark signal is intended for enhancing the robustness of information-symbol embedding techniques and when we decode each segment separately in multiple signaling (which is the core of this dissertation).

This chapter proceeds as follows. In Section 8.1, two approaches that can be used for overcoming burst errors are compared. The first approach does not use interleaving, while the second approach uses interleaving. In Section 8.2, the effect of interleaving on bit-by-bit decoding is demonstrated. In Section 8.3, some specifics about interleaving the power-constrained watermark signal are discussed. In Section 8.4, interleaving the watermark signal with the existence of background noise, in addition to the burst error for which we interleave, is discussed. In Section 8.5, interleaving the watermark signal in the presence of quantization noise is studied. The case when only quantization noise exists, and the case when quantization noise and burst errors exist are covered. In Section 8.6, power-constrained versus nonpower-constrained interleaved codes are compared. Section 8.7 is a summary.

8.1 Two Watermarking Approaches

Decoding the partially damaged watermark signal can be approached from two different angles. The first angle does not use interleaving, while the second uses interleaving. In this section, the two approaches are compared, and it is shown that the interleaving approach can achieve the same performance as the noninterleaving approach while avoiding a significant increase in the computational complexity of the decoding process.

8.1.1 Noninterleaving Approach

For this approach, the watermarking process can perform the following steps:

1. In the image under debate, I_2 (which contains the damaged watermark signal), substitute the damaged part of the signal (which is assumed to be known) by the equivalent part in the original image, I_0 (this is done in the spatial domain). This will make this part of the image contain no watermark signal at all.

2. Instead of applying the DCT to 8×8 blocks, apply the DCT to the entire image as one block (i.e., a 256×256 block).

3. Subtract I_0 from I_2 (in the DCT domain), and extract the watermark signal.

The end result of this approach can be expressed as follows. If the image with the watermark signal was not damaged at all, correlation detection applies Equation (2.8). If we exclude the denominator as explained in Chapter 4, Equation (2.8) becomes

$$\rho_i = \sum_{j=0}^{n-1} x_{i,j}.x_j^*, \tag{8.1}$$

where n is the length of the decoded segment, $x_{i,j}$ is the base signal's magnitude ($\pm\Delta$), and x_j^* is the j^{th} bit magnitude of the extracted signal ($\pm\Delta + N_0$).

Now suppose the attacker destroyed half of the watermark signal in I_2. After substituting this part of I_2 from I_0, according to the above steps, we do not obtain half a watermark signal. Because the DCT is applied to the entire image, the energy of the remaining part of the signal is spread among all the coefficients of the watermark signal. Thus, on average, Equation (8.1) becomes

$$\rho_i = \sum_{j=0}^{n-1} \frac{1}{2} x_{i,j}.x_j^* = \frac{1}{2} \sum_{j=0}^{n-1} x_{i,j}.x_j^*. \quad (8.2)$$

8.1.2 Interleaving Approach

For this approach, the above three steps are performed with the DCT applied to 8×8 blocks instead of to the entire image (as was done in previous chapters, and as specified in the international video coding standards). After substituting the part of I_2 containing the damaged part of the signal by its equivalent part in I_0, any coefficient carrying the watermark signal that falls in the destroyed part is excluded from the correlation estimation. As a result, if half of the image—and thus half of the watermark signal—is lost, and the signal is *fully interleaved*, Equation (8.1) becomes

$$\rho_i = \sum_{j=0}^{(n-1)/2} x_{i,j}.x_j^*. \quad (8.3)$$

For the correlation detector, Equations (8.2) and (8.3) should yield the same performance. On average, both cases give vectors with half of the energy obtained if the watermark signal is fully recovered. It is apparent from this comparison that if we can interleave the watermark signal, the interleaving approach can avoid the mathematical complexity of applying the DCT to the entire image as one block. Note here that the computation needed for interleaving is just simple memory assignments, as explained in Section 7.5.

8.2 Effect of Interleaving on Bit-by-bit Decoding

This section demonstrates how interleaving can increase the robustness of the watermark signal with the simplest coding approach possible (bit-by-bit decoding using first-order repetition code). Although Chapter 2 explained that, for the power-constrained watermark signal, bit-by-bit decoding gives the poorest performance, bit-by-bit decoding is the most straightforward approach, and we will use it here just to demonstrate how the 2-D interleaving technique studied in Chapters 5, 6,

and 7 outperforms known 1-D communications interleaving techniques when applied to the watermark signal. At the end of this chapter, the specifics of interleaving the power-constrained watermark signal are established.

Recall the two strategies used in writing a 1-D interleaved signal to a 2-D layout, discussed in Section 5.4: the spiral pattern and the boustrophedonic pattern. In Chapter 6, using simulation, we compared the performance of these two strategies with the performance of our interleaving technique when applied to a 16×16 symbol layout using RS codes. In this section, we compare 2-D interleaving with these two strategies when applied to the watermark signal using bit-by-bit decoding. Here, the 256×256 "Lena" image was used, the DCT was applied for 8×8 blocks, and the number of coefficients per block was selected to be 3, as was done in Chapters 2, 3, and 4. Thus, as before, we are working with an embedded signal of 3072 bits. To apply first-order repetition code, these 3072 encoded bits are made to correspond to 1536 source bits. That is, every source bit is repeated and the 1536 source bits are translated into 3072 encoded bits. Three strategies (1-D spiral writing, 1-D boustrophedonic writing, and 2-D interleaving) were simulated. With all three scenarios, the watermark embedding process scans the image three times and embeds one bit in each block with each scan. Scanning the entire image is done according to the specifics of each strategy (spiral, boustrophedonic, or line-by-line for 2-D interleaving). Each scan embeds $1024 = 32 \times 32$ encoded bits.

For the image with the embedded watermark signal, we started to destroy part of the image (group of blocks), and as a result destroyed the corresponding part (group of blocks) in the watermark signal. The size of the destroyed part is expressed in the number of blocks affected by the error across one axis (X_0 or Y_0 in Equation (6.3)). The curve in Figure 8.1 shows the obtained results. The horizontal axis is the size of the destroyed part of the signal ($X_0 = Y_0$ is expressed as the number of blocks) and the vertical axis is the probability of bit decoding error. A bit decoding error occurs when two embedded bits, which correspond to one of the 1536 source bits, fall in the destroyed part of the image. In other words, each of the 1536 source bits are repeated, and the 3072 encoded bits are embedded at the encoder side. At the decoder side, the two encoded bits corresponding to a source bit are checked.

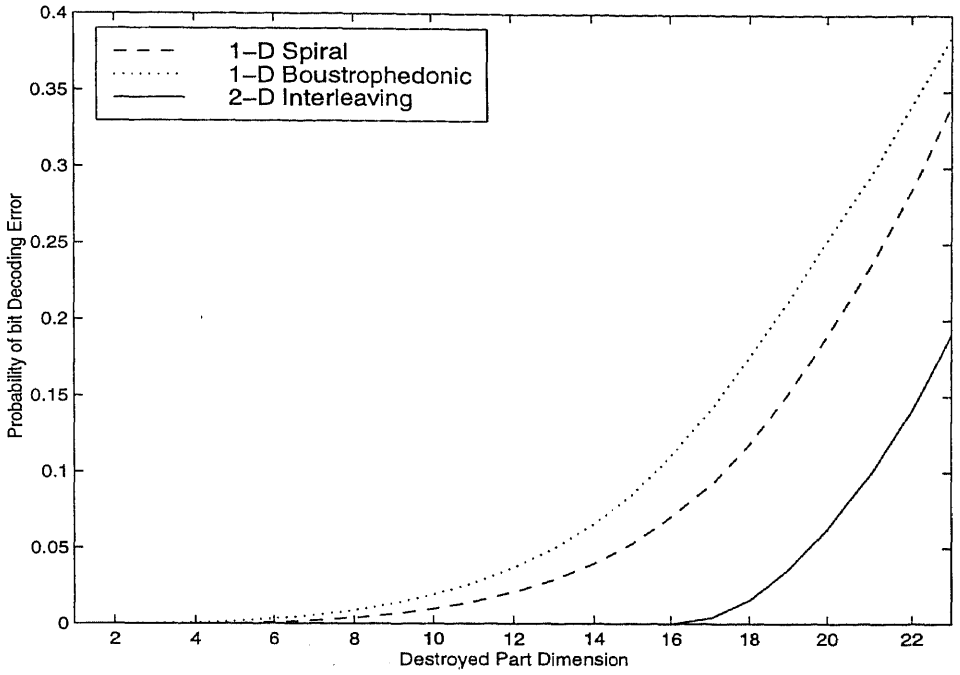


Figure 8.1 Probability of Bit Decoding Error for Bit-by-bit Interleaving

If at least one bit remains intact, no decoding error occurs since the original bit is recoverable. It is only if both encoded bits are destroyed that a decoding error occurs. All possible positions of the destroyed part with respect to the image are considered equally. That is, these statistics are collected by moving the cluster of the destroyed blocks to all possible positions the cluster can fall in with respect to the image.

8.3 How to Interleave the Watermark Signal

At this point, it is established that 2-D interleaving is more advantageous than interleaving techniques used in communication channels when it comes to signals with 2-D layouts. Throughout the rest of this chapter, what we did in the previous chapters is continued: divide the watermark signal into segments and make each segment decode a symbol. We are interested in finding the 2-D *interleaving strategy* that minimizes the probability of symbol decoding error. *Interleaving strategy* refers to the best 2-D interleaving specifics for the power-constrained watermark signal.

Burst error correction-coding strategies that use interleaved codes assume a nonpower-constrained signal. The reader can refer to some of the 1-D applications that use interleaved codes, such as deep-space communications [22], automatic repeat request protocols [38], spread-spectrum communications [32], and frequency-hop communications [30]. The reader can also find some applications for a signal with a 2-D layout. These applications include compact disk (CD) recording [16] and 2-D barcoding [9] [23] [26] [41] [42]. The closest of these applications to the watermark signal is 2-D barcoding because

1. like the watermark signal, the encoded data layout is in 2-D.

2. like the watermark signal, the decoder can have access to all the encoded data at once (this is not the case with the CD, where the encoded data are broken into segments and each segment is interleaved separately).

The major difference between 2-D barcoding and watermarking is that the watermark signal is a power-constrained signal, while in 2-D barcoding it is a nonpower-constrained signal.

8.3.1 Interleaving for Nonpower-constrained Signals

The most commonly interleaved codes for burst error correction are Reed-Solomon (RS) codes. Interleaving the data can allow using simple (low-order) RS codes (e.g., rate $\frac{1}{2}$) instead of higher-order codes, as explained in Section 6.5. RS codes (which are used to correct symbols; not bits) have the following property: If one bit in the encoded symbol is recovered in error, the entire symbol is decoded in error. Thus, when interleaving RS codes for these applications, all the bits corresponding to one encoded symbol are clustered together in one area. When a burst error occurs, this error affects the least number of encoded symbols. The redundancy added with the interleaved code allows for recovering the original data as long as the number of symbols decoded in error does not exceed the correction capability of the interleaved code, λt (see Equation (6.1)). With 2-D barcoding, the bits associated with an encoded symbol are clustered together in one area (see [41] for some clustering

strategies). When a burst error occurs (which damages a certain area of the 2-D layout), it affects fewer encoded symbols than if no clustering is applied.

Now comes the question: Do we have to cluster the bits associated with each segment of the watermark signal the way nonpower-constrained applications do? The answer to this question is no. With a power-constrained signal, however, the situation is the opposite. We should actually try to avoid any clustering and interleave the signal on a bit-by-bit basis. The next subsection supplies an explanation for why a power-constrained signal is interleaved on a bit-by-bit basis, as well as simulation results to support this point.

8.3.2 Interleaving for Power-Constrained Signals

Let n be the number of bits per segment (which should decode a symbol, a_i), and let m be the number of segments in the watermark signal such that $N = n \times m$, where N is the total number of bits. Let μ be the total number of bits excluded from the estimation process of the entire signal due to the burst error. Let μ_i be the number of bits excluded from the estimation process of the symbol a_i due to the burst error. (These bits are excluded from the detection process, as explained in the three steps in Section 8.1). With power-constrained signals, a symbol decoding error occurs when the number of bits included in the estimation process, $n - \mu_i$, are not enough for the decoder to decode a_i. That is, a_i is decoded correctly if

$$n - \mu_i \geq \alpha, \tag{8.4}$$

where α is the minimum number of bits needed to decode a_i. α depends on the utilized maximum separable signal set, as will be explained shortly.

Equation (8.4) suggests that unlike interleaved RS codes, with the power-constrained signal, some of the bits associated with a certain symbol can be in error but the symbol is still decoded correctly. (This situation is the opposite of nonpower-constrained signals.) With power-constrained signals, interleaving should attempt to force the burst of errors to affect the least number of bits per each symbol (i.e., to spread the error among all symbols). On the other hand, interleaving for nonpower-constrained signals attempts to force the burst error to affect the least

number of symbols; that is, contain the error to as few symbols as possible. If one bit falls in the destroyed part, every other bit in this symbol should be made to fall in the destroyed part, as well, which gives a better chance for other symbols to be decoded error free.

To show how interleaving for power-constrained signals should be done on a bit-by-bit basis, three different clustering strategies were simulated—all using our 2-D interleaving technique. As was done previously, the simulation here uses the 256×256 "Lena" image, and a DCT of 8×8 blocks. Here, the number of coefficients per block is eight (this is needed to utilize the interleaving technique efficiently). The energy per embedded bit, Δ, is selected to be 2.5 instead of 6. When three coefficients per block were used, we utilized 3072 bits with $\Delta = 6$. Here, eight bits per block are used and 8192 bits are utilized with $\Delta = 2.5$. The signal set in Table 2.1 was adapted for a signal set of 64 bits per entry by duplicating the 31 bits in each entry to 62 bits and adding two dummy bits at the end (the two added bits are not utilized). The minimum Hamming distance in this signal set is 30 (twice that of the 31 bits/signal case). With the 64 bits/signal case, since each block carries eight bits, each signal occupies eight blocks and 128 symbols are embedded in the image.

Based on the above parameters, the following three strategies were studied, and they showed the necessity of avoiding clustering.

- **Bit-by-bit Interleaving:** With this strategy, interleaving is done based on each bit individually (no clustering). The image is scanned eight times. The first scan embeds 1024 bits in the same AC coefficient in each block. The second scan embeds another 1024 bits in another coefficient, and so on. Within each scan, 16 symbols are embedded while interleaving the 1024 bits corresponding to the 16 symbols using our 2-D interleaving technique.

- **Eight-bit Interleaving:** With this strategy, interleaving is done based on each block (eight-bit clustering). The 64 bits corresponding to one symbol are embedded in eight blocks. Each cluster of eight bits is interleaved in the image layout using our 2-D interleaving. Here, the image is scanned once, and the

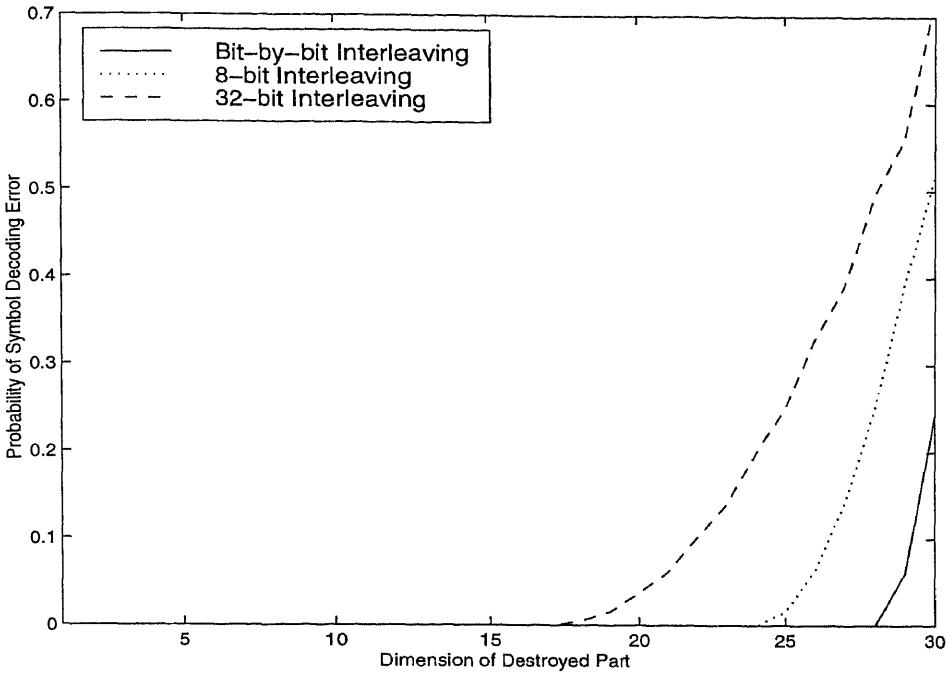


Figure 8.2 Probability of Symbol Decoding Error for Three Different Clustering Strategies

embedding mechanism fills each block with eight bits before it moves to the next block according to our 2-D interleaving technique.

- **32-bit Interleaving:** With this strategy, interleaving is done by splitting the 64 bits corresponding to each symbol into two clusters (32 bits per cluster). The 64 bits corresponding to one symbol are embedded in two groups, each of four blocks. Each cluster of 32 bits is interleaved in the image layout using our 2-D interleaving technique. Here, the image is scanned once, and the embedding mechanism fills a group of 2×2 blocks with 32 bits before moving to the next group according to our 2-D interleaving technique.

Figure 8.2 shows the obtained results of the above three interleaving strategies. One can see that the bit-by-bit strategy gives the best performance (the lowest error rate). One should not, however, underestimate the powerful performance obtained from 2-D interleaving with all three strategies. With 32-bit interleaving, symbol decoding error starts to occur when the size of the destroyed blocks exceeds $18 \times 18 =$

324 blocks. This is almost $\frac{1}{3}$ of the entire image, $32 \times 32 = 1024$ blocks. With eight-bit interleaving, symbol decoding error starts to occur when the size of the destroyed blocks exceeds $24 \times 24 = 576$ blocks. This is more than $\frac{1}{2}$ of the entire image. With bit-by-bit interleaving, symbol decoding error starts to occur when the size of the destroyed blocks exceeds $28 \times 28 = 784$ blocks. This is more than $\frac{3}{4}$ of the entire image. Notice that, because of the structure of the interleaving technique, this destroyed area can fall anywhere in the image.

Figure 8.2 reveals very important criteria with regard to bit-by-bit interleaving. This interleaving technique seems to be optimum if we look at the following parameters (considering that symbol decoding error started to occur after $X_0 = Y_0 = 28$):

- Total number of blocks = 1024.
- Number of blocks in error = 784.
- Number of blocks error free = 240.
- Number of bits error free = 1920.
- Average number of error free bits per each embedded symbol = $\frac{1920}{128} = 15$ bits.

The above parameters suggest that as long as there are 15 or more error-free bits per symbol (on the average), all symbols can be detected correctly. This interleaving technique will be considered optimum if, for the utilized signal set, the minimum number of bits needed to decode a symbol is $\alpha = 15$. See Equation (8.4). The optimality can be considered since the average case is the lower bound. Now, if we consider the structure of the utilized 64-bits per symbol signal set, we find that the minimum Hamming distance between any two entries is 30, which is the diameter of the Hamming sphere of each symbol. The radius of the Hamming sphere is then 15. This suggests that the Hamming sphere of neighboring symbols will start to intersect, resulting in a symbol decoding error if there are less than 15 error-free bits [21].

8.4 Burst Errors and Background Noise

Chapters 5-7 established that 2-D interleaving should be used with applications where the signal layout is in 2-D. Section 8.3 established that with the power-constrained watermark signal, bit-by-bit interleaving gives better performance than other clustering strategies. The rest of this chapter covers practical scenarios with the watermark signal. One practical scenario is having some part of the watermark signal destroyed (due to some attack) and, at the same time, some background noise (within the entire image) exists. An argument can be raised from the following question: Can the existence of background noise change the performance of the above three clustering strategies? That is, can the existence of background noise make an interleaving strategy give better performance than bit-by-bit interleaving? This argument can be clarified as follows. Imagine that we have a destroyed area that affected some $\mu \leq n$ bits, where n is the number of bits per symbol. There are two extreme scenarios here. Bit-by-bit interleaving makes these μ bits affect some μ symbols. That is, each group of bits that decodes a symbol will have one of its bits damaged. The other extreme scenario is the maximum amount of clustering possible, which makes these μ bits affect as few symbols as possible (because $m \leq n$, we end up with only one symbol affected by this error). This gives the other symbols a chance to overcome the existence of background noise.

The above argument might proceed to state that either of these extreme cases may not be the best scenario that gives the lowest probability of symbol decoding error. With the first scenario, although only one bit in each symbol is destroyed, it can have a bad effect if other background noise (e.g., filtering noise) exists. On the other hand, the second scenario may affect only one symbol, assuring a one-symbol decoding error. This may not be the best case either, because it might be better to spread this error among many symbols, and have no decoding error at all.

We rely on simulation to prove that the above argument is not completely valid. As the coming simulation result shows, the background noise will affect all three interleaving strategies. It does affect bit-by-bit interleaving more than the other two strategies, but not to the extent of making it give less performance than the other two strategies.

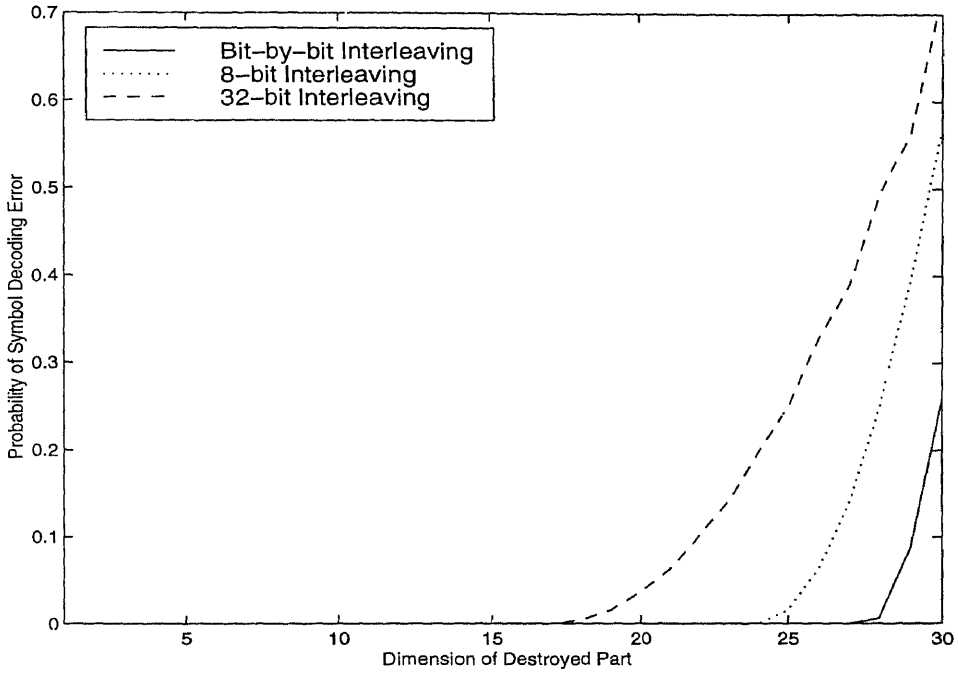


Figure 8.3 Probability of Symbol Decoding Error with $SNR_{WM} = 6$ db

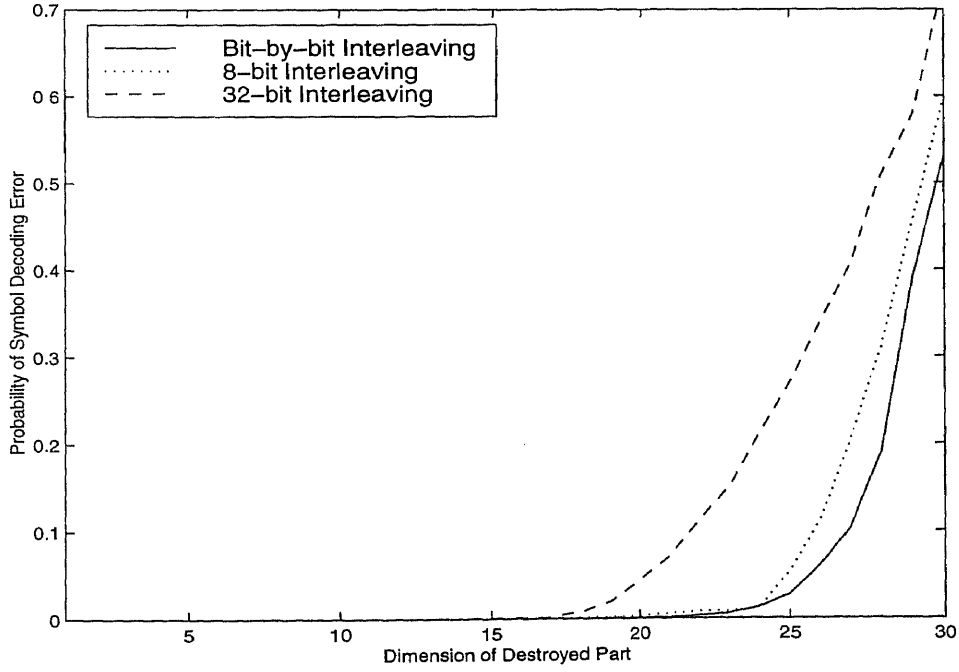


Figure 8.4 Probability of Symbol Decoding Error with $SNR_{WM} = 0$ db

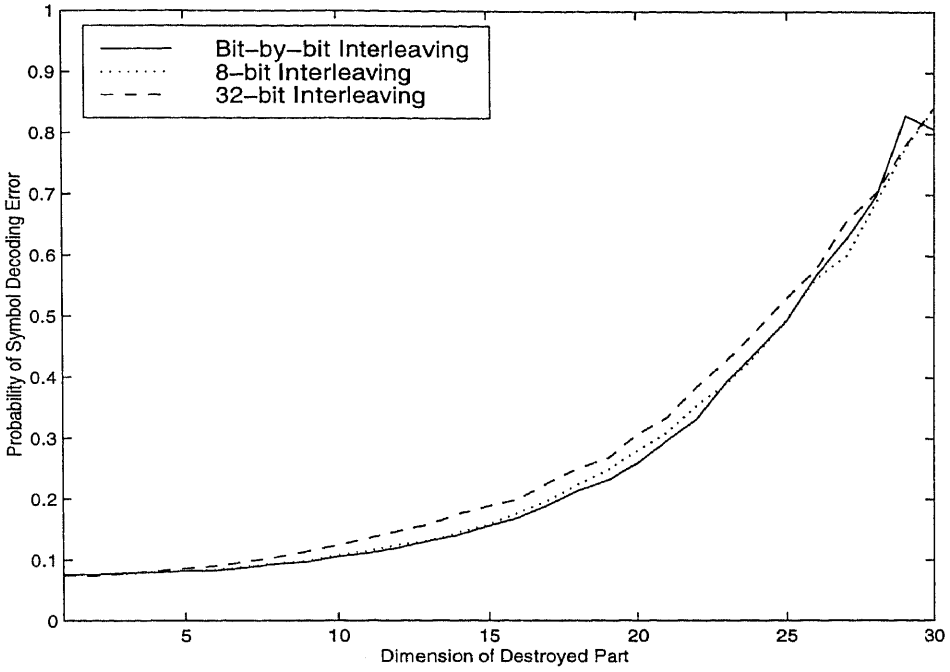


Figure 8.5 Probability of Symbol Decoding Error with $SNR_{WM} = -6$ db

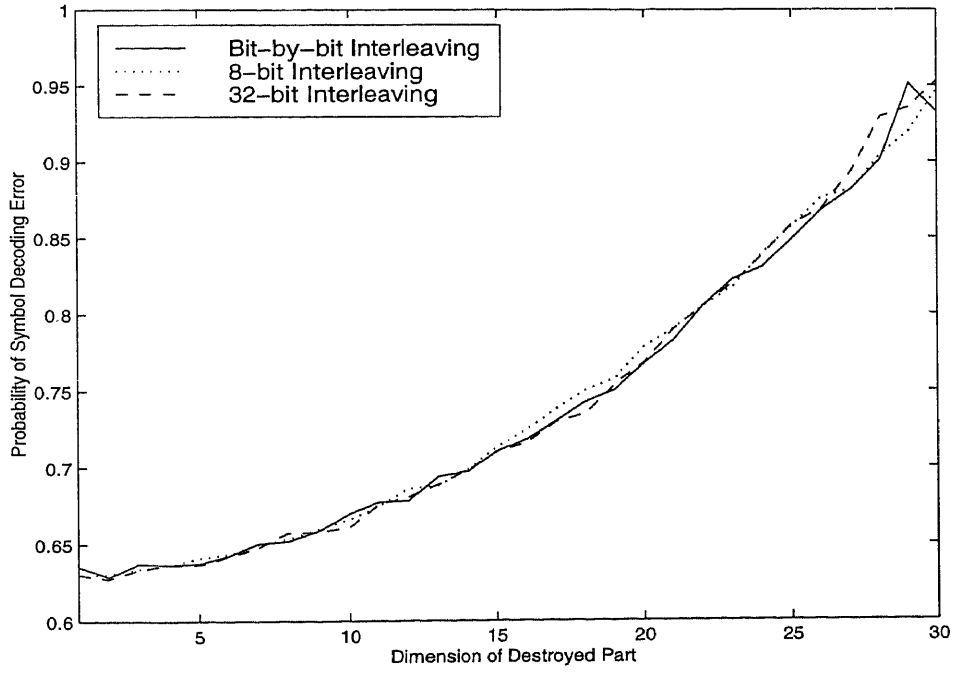


Figure 8.6 Probability of Symbol Decoding Error with $SNR_{WM} = -12$ db

The curves in Figures 8.3 through 8.6 demonstrate the effect of the background noise. With this simulation, in addition to destroying part of the signal, we introduced some background noise to the entire image and increased this noise gradually. In Figure 8.3 the background noise magnitude is 1.25 ($\sigma = 1.25$), which resulted in an SNR_{WM} of 6 db (see Equation 2.3). This did not have any significant affect on the performance obtained with no background noise. The difference between Figure 8.3 and Figure 8.2 is minimal. In Figure 8.4 the background noise magnitude is increased from 1.25 to 2.5, which is the same magnitude as the signal Δ. This resulted in an SNR_{WM} of 0 db. One can see that, although bit-by-bit interleaving is affected more than the rest, it still gives the best performance. In Figure 8.5 the noise magnitude is increased to twice the magnitude of the signal, which resulted in an SNR_{WM} of -6 db. One can see that the three strategies approach the same performance level. This degraded performance is due mostly to the background noise. In Figure 8.6 the noise magnitude is increased to four times as much as the signal magnitude, which resulted in an SNR_{WM} of -12 db. At this level the lowest probability of symbol decoding error (where there is no burst error at all) is more than 0.6, which is totally unacceptable. There is no indication, however, that the other two clustering strategies give better performance than bit-by-bit interleaving.

At this point it is established that 2-D interleaving following a bit-by-bit strategy is the proper way to interleave the watermark signal, even in the presence of both burst errors and background AWGN. We now move to the other case where interleaving is useful; that is, in the presence of quantization errors.

8.5 Interleaving the Watermark Signal in the Presence of Quantization Noise Using Erasure Decoding

In the previous section, interleaving the watermark signal in the presence of background noise, in addition to the burst error for which we interleave the signal, was discussed. The background noise used in Section 8.4 is white Gaussian. In this section, interleaving the watermark signal in the presence of quantization noise is discussed. First, only quantization noise was considered, then the existence of both burst and quantization errors were considered. Note that white Gaussian noise and

quantization noise (which has a uniform distribution) are the worst types of noise the watermark signal can face. The effect of many types of noise can be reduced by using a threshold, τ, in the decoding process as follows. If a coefficient is affected by noise magnitude that exceeds the level of τ, this coefficient can be eliminated from the decoding process. That is, we eliminate a coefficient if

$$|\pm\Delta + r| \geq \tau, \tag{8.5}$$

where r is the additive noise (not Gaussian).

Using this threshold decoding approach (which also can be called erasure decoding) can be seen best in the presence of high-level impulse noise. To see this practically, background AWGN was introduced and some impulse noise was then added. That is, the extracted signal magnitude, x_k^*, can be modeled at each coefficient affected by the impulse noise as

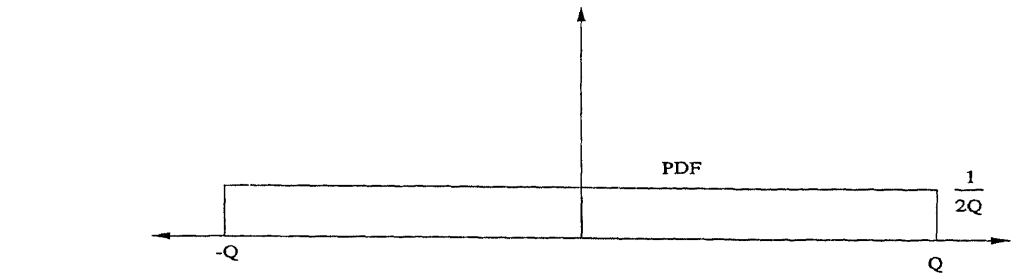
$$x_k^* = \pm\Delta + N_o + \delta, \tag{8.6}$$

where δ is the impulse noise magnitude. At the coefficients not affected by the impulse noise,

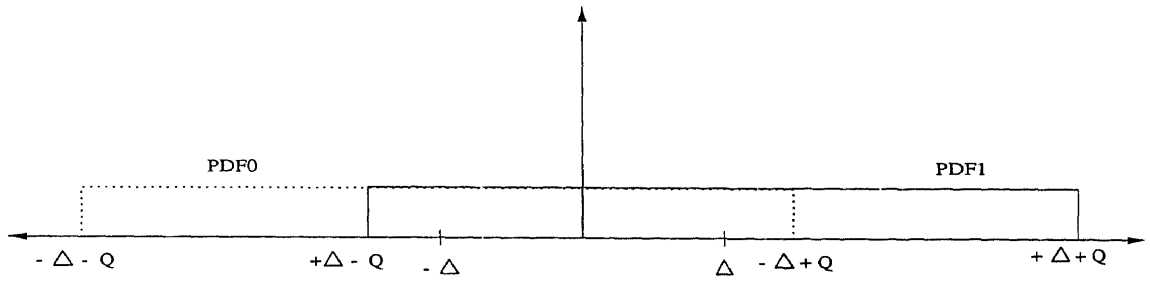
$$x_k^* = \pm\Delta + N_o. \tag{8.7}$$

In the decoding process, τ is set to be 3Δ, and the effect of the impulse noise is found to be minimal. Actually, with $\tau = 3\Delta$ and if the impulse noise magnitude is $\delta \geq 4\Delta$, the decoding process erases (excludes) this coefficient from the correlation process with a probability close to 1. If the number of coefficients affected by the impulse noise is small, the effect of the impulse noise is minimal.

Practically, a watermark signal may suffer from a combination of some attacks that cause a part of the signal to be lost (burst error), in addition to quantization noise that results from compressing the image. Here, how to deal with the quantization noise by itself will be shown first (using erasure decoding), then the combination of burst and quantization noise will be studied. At this point, we are applying 2-D interleaving on a bit-by-bit basis.



(a) PDF of Quantization Noise



(b) Two Hypotheses When Decoding the Watermark Signal

Figure 8.7 Two Hypotheses of Decoding the Watermark Signal in the Presence of Quantization Noise

8.5.1 With Quantization Noise Only

When the watermarked image goes through compression, the added watermark signal magnitude suffers from quantization noise. The pdf of this noise has a uniform distribution between the values of $-Q$ and $+Q$. The value of Q depends on the amount of compression the image goes through. With a high compression ratio, Q is high and vice versa [20]. Thus, similar to Figure 2.1, we can draw Figure 8.7 where the two hypotheses are pdf_0 and pdf_1.

Looking at Figure 8.7, where the pdf of the extracted values are not concentrated around $+\Delta$ and $-\Delta$ as with the Gaussian distribution, one may wonder if any other detection technique can be applied that may perform better than correlation detection, or if correlation detection can be adapted from the case of Gaussian noise to the case of uniform noise. Here, correlation detection with erasure is used. Recall what is mentioned in Section 8.3.2: that we can detect a segment correctly as long as the number of error-free bits exceeds a certain level, α. See Equation (8.4). Now,

using Figure 8.7, and considering the threshold decoding approach mentioned above for high magnitude impulse noise, a threshold, τ, could be established that allows only the consideration of the bits that can be decoded correctly. Consider the range between $+\Delta - Q$ and $-\Delta + Q$. If the extracted value falls in this range, it can correspond to an encoded bit 0 or an encoded bit 1 with the same likelihood. If the extracted value falls in the range of $-\Delta - Q$ to $+\Delta - Q$, it certainly corresponds to an encoded 0. On the other hand, if the extracted value falls in the range of $-\Delta + Q$ to $+\Delta + Q$, it certainly corresponds to an encoded bit 1. Thus, if the decoding threshold is used as follows:

$$\tau = |\Delta - Q|, \tag{8.8}$$

only the bits that can be decoded error free are considered. As mentioned before, this type of decoding is referred to as either threshold decoding or erasure decoding.

Now, one may ask the following questions with regard to quantization noise: Is it better to use erasure decoding all the time, or is it better to avoid it all together, or when is it better to consider erasure decoding? To answer the above questions we rely on Figure 8.7, with the threshold given in Equation (8.8). For the extracted value x_k^*, and for the k^{th} bit in the signal, the pdf of 0 can be expressed as follows:

$$f(x_k^*|0) = pdf_0 = \begin{cases} \frac{1}{2Q} & \text{if} \quad -\Delta - Q \leq x_k^* \leq -\Delta + Q \\ 0 & \text{Otherwise} \end{cases}. \tag{8.9}$$

The pdf of 1 can be expressed as follows:

$$f(x_k^*|1) = pdf_1 = \begin{cases} \frac{1}{2Q} & \text{if} \quad +\Delta - Q \leq x_k^* \leq +\Delta + Q \\ 0 & \text{Otherwise} \end{cases}. \tag{8.10}$$

Using erasure decoding (where only correct bits are considered), the probability of decoding 0, given that the encoded bit is 0, can be calculated as follows:

$$\begin{aligned} p(0|0) &= \frac{1}{2Q} \times (+\Delta - Q - (-\Delta - Q)) \\ &= \frac{1}{2Q} \times (+\Delta - Q + \Delta + Q) \\ &= \frac{2\Delta}{2Q} = \frac{\Delta}{Q}. \end{aligned} \tag{8.11}$$

Similarly, the probability of decoding 1, given that the encoded bit is 1, can be calculated as follows:

$$\begin{aligned} p(1|1) &= \frac{1}{2Q} \times (+\Delta + Q - (-\Delta + Q)) \\ &= \frac{1}{2Q} \times (+\Delta + Q + \Delta - Q) \\ &= \frac{2\Delta}{2Q} = \frac{\Delta}{Q}. \end{aligned} \tag{8.12}$$

The total probability of decoding the correct bit, P_c, is then given by

$$P_c = p(1|1)p(1) + p(0|0)p(0), \tag{8.13}$$

where $p(0)$ and $p(1)$ are the probability of encoding 0 and 1, respectively. Since $p(0) = p(1) = \frac{1}{2}$,

$$P_c = \frac{1}{2}\frac{\Delta}{Q} + \frac{1}{2}\frac{\Delta}{Q} = \frac{\Delta}{Q}. \tag{8.14}$$

From Equation (8.14), the average number of bits included in the decoding process (this number is $n - \mu_i$ in Equation (8.4)) can be estimated as follows:

$$n - \mu_i = n.P_c = n\frac{\Delta}{Q} = \frac{n\Delta}{Q}. \tag{8.15}$$

From Equation (8.4), Equation (8.8), and Equation (8.15), it can be stated that for a given quantization noise with a magnitude Q, a watermark signal magnitude of Δ, a segment length n, and a threshold τ, the expected number of error-free bits can be evaluated from Equation (8.15).

It is clear from Equation (8.15) that as Δ decreases (assuming a fixed Q), the number of bits included in the decoding process decreases, and hence more symbol decoding error occurs. As a result, there may be better performance with erasure than without erasure when Δ is high, and lower performance when Δ is low. Recall from Chapter 2 that for our power-constrained watermark signal, we can trade n (the segment dimension) for Δ (the signal magnitude). Thus, one needs to know for a given Δ if it is appropriate to use erasure decoding or not. In other words, we want to know specifically when to use erasure decoding and when to avoid it.

Using the 256×256 "Lena" image, the watermarked image was compressed using JPEG compression (using SPIHT) from a size of 64k to a size of 5k. This

Table 8.1 The Threshold τ for Erasure Decoding Using $Q = 12$

Watermark Signal Magnitude Δ	8	7	6	5	4	3	2
Threshold Level τ	4	5	6	7	8	9	10

introduces a relatively high quantization noise. The compressed image is shown in Figure 8.8 (the original image can be seen in Figure 2.4). The maximum quantization magnitude, Q, was first estimated to be close to 12. Equation (8.8) was used to estimate τ. The values of τ are given in Table 8.1 for different levels of the watermark signal magnitude Δ.

Using simulation for different levels of Δ, the probability of symbol decoding error with erasure and without erasure was calculated. With the erasure approach, using the thresholds in Table 8.1, only the error-free bits, $n - \mu_i$, were included in the correlation process. Without erasure, all the bits were included in the correlation process. The obtained results are plotted in Figure 8.9. One can see that as long as the watermark signal magnitude is larger than about 3, erasure decoding gives a better performance. The change in performance can be explained as follows: As Δ decreases, the number of bits included in the estimation process, $n - \mu_i$ (given from Equation (8.15)), decreases, which increases the likelihood of symbol decoding error. Notice that when Δ is less than 3, the probability of decoding error is relatively high with both approaches (more than 0.25). Thus, at this point one can conclude that erasure decoding is better than nonerasure decoding when it comes to decoding a watermark signal that suffers solely from quantization noise.

Recall from Chapters 2 and 3 that when the probability of symbol decoding error is high, the number of bits per segment, n, should be increased. This increase in n will increase $n - \nu$ and keep the probability of symbol decoding error low. With erasure decoding the same concept can also be applied.

8.5.2 With Burst Error and Quantization Noise

To consider the existence of both types of error, the combination of burst errors and quantization noise as the size of the burst error increases were simulated. The same

Figure 8.8 Lena Image After Compression

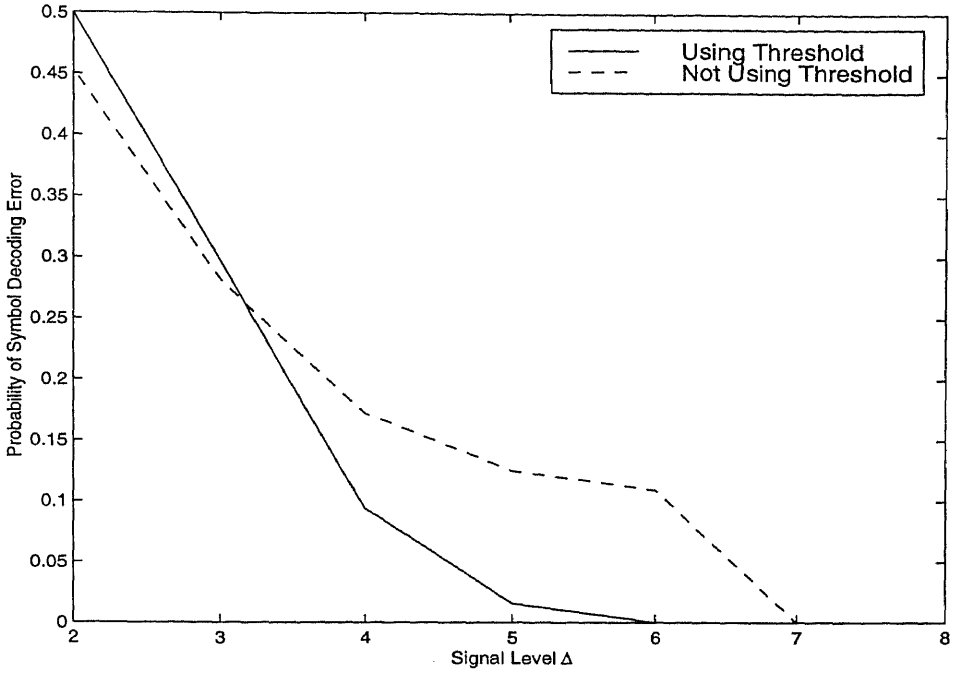


Figure 8.9 Decoding with Erasure Versus Decoding without Erasure

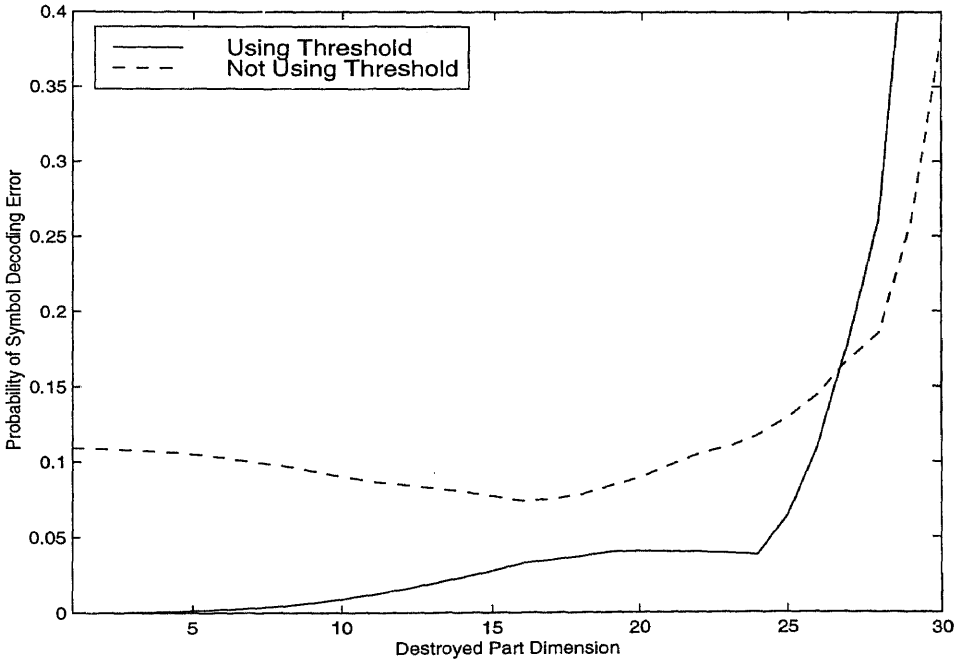


Figure 8.10 Interleaving the Watermark Signal in the Presence of Burst Errors and Quantization Noise When $\Delta = 6$, with and without Threshold Decoding

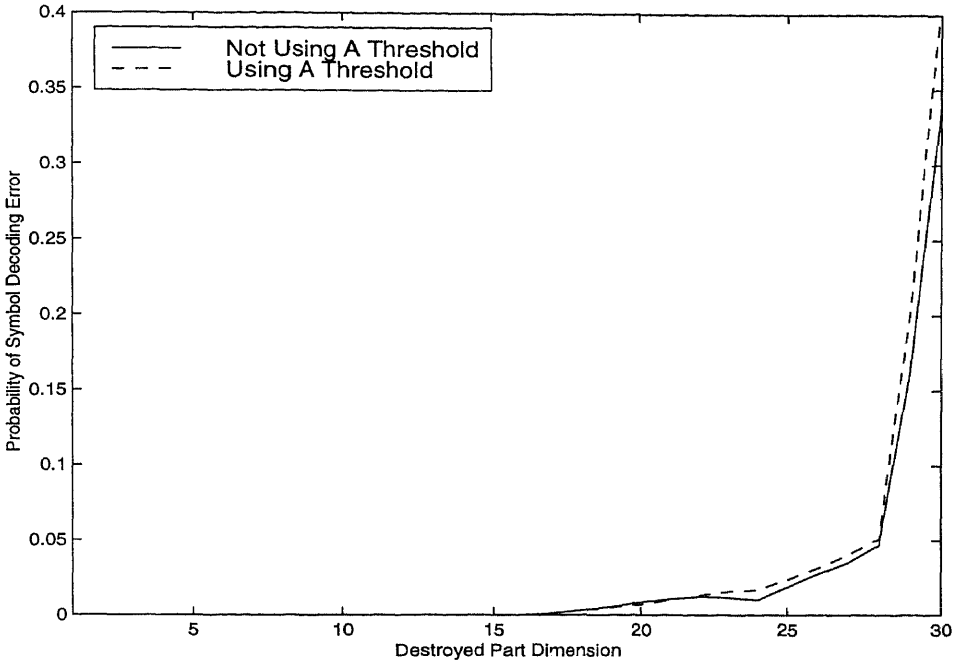


Figure 8.11 Interleaving the Watermark Signal in the Presence of Quantization Noise When $\Delta = 8$, with and without Threshold Decoding

"Lena" image of size 256×256 was used and the DCT was applied to 8×8 blocks. The number of coefficients carrying the watermark signal was selected to be 3. The burst-error size was gradually increased as was done before. In the presence of both the burst error and quantization noise, decoding with erasure and without erasure was considered. The obtained results are shown in Figure 8.10. One can see that erasure decoding gives better performance until the size of the burst error exceeds $27 \times 27 = 729$ blocks, which is close to $\frac{3}{4}$ of the entire image. At this level of error, the number of segments, m, should be decreased to increase n or Δ, as will be explained in the next paragraph.

One may wonder about the performance of erasure decoding versus nonerasure decoding when the signal magnitude, Δ, reaches 8. According to Figure 8.9, with no burst error, both approaches give a probability of symbol decoding error equal to zero. On this issue, the simulation was repeated with different parameters: the number of coefficients carrying the watermark signal was selected to be 2 and Δ was selected to be 8. This scenario will be explained shortly. The results obtained from

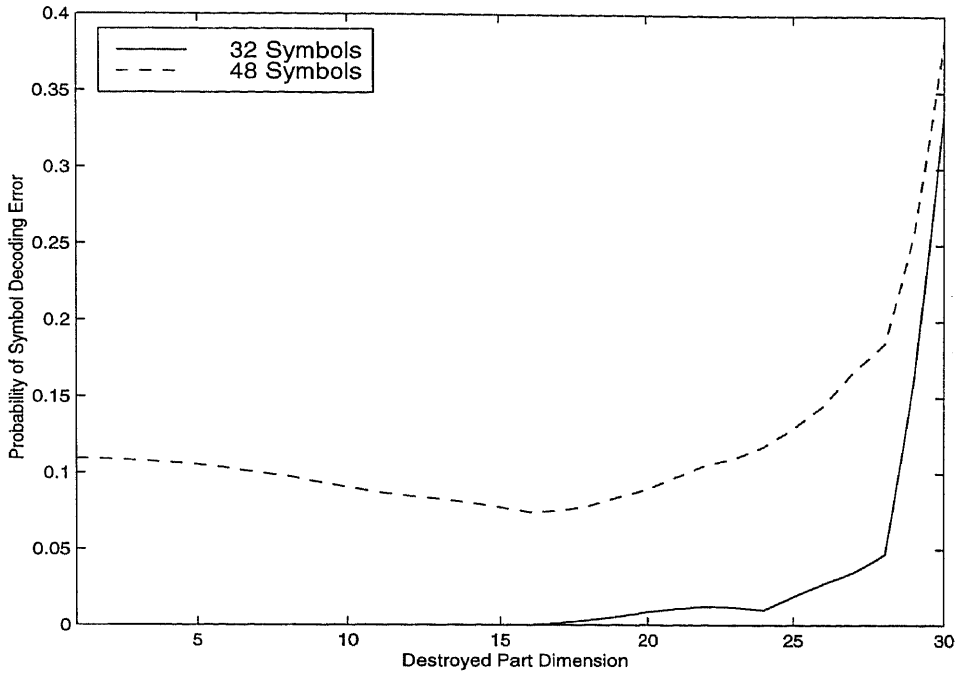


Figure 8.12 Interleaving the Watermark Signal in the Presence of Quantization Noise, Trading the Number of Segments for the Signal Magnitude.

this simulation are plotted in Figure 8.11. One can see that as the size of the burst error increases, the performance of both approaches degrades more or less by the same amount.

The curve in Figure 8.12 shows how the presence of both quantization noise and burst errors may be approached if erasure decoding is not to be considered. Because the probability of symbol decoding error is high, even with a small size burst of errors, the capacity analysis in Chapter 3 can be referenced, which indicates that decreasing the watermark signal dimension (use less coefficients) and increasing the magnitude per each coefficient is one way to lower the probability of symbol decoding error. That is, an attempt is made to overcome the presence of more noise by keeping the segment dimension, n, the same (64 bits per segment) and reducing the total watermark signal dimension, N, from 3072 to 2048, while increasing the signal magnitude, Δ, from 6 to 8. The end result is to reduce the number of symbols carried in the signal from 48 to 32 (which agrees with Shannon's capacity theorem). Notice that with this approach, to overcome the error rate with $\Delta = 6$, a penalty was paid by reducing the

number of segments from 48 to 32. Concentrating the watermark signal power in two coefficients instead of three may not be appropriate. On the other hand, considering Figure 8.10, one can see that erasure decoding achieves better performance without penalty. The signal dimension, the number of segments, and the signal magnitude remain the same, and only a threshold is considered in the decoding process.

8.6 Power-Constrained Versus Nonpower-Constrained Interleaved Codes

Section 8.4 concludes that 2-D bit-by-bit interleaving is the most suitable interleaving approach for the watermark signal. This section compares the power-constrained bit-by-bit interleaving approach with a nonpower-constrained bit-by-bit interleaving approach using the same coding rate. One can look at this section as the analogy to Section 2.4 when evaluating interleaved codes. In Section 2.4, we measured the performance of error correction codes (which are supposed to be applied to nonpower-constrained signals) when applied to our power-constrained watermark signal versus the performance of correlation detection (which is an optimum detection technique for power-constrained signals). In this section, a comparison is made to measure the performance of interleaved error correction codes (which are suppose to be applied to nonpower-constrained signals) when applied to our power-constrained watermark signal versus the performance of the power-constrained interleaving approach (which adds redundancy by increasing the signal dimension).

Since we are using a powerful interleaving approach, we can use a simple error correction code to produce an interleaved code. For the nonpower-constrained approach, an RS code of rate $\frac{1}{2}$ was simulated while the same code rate as was used throughout this chapter (64 bits per symbol) was utilized, as follows. A set of maximum separable signals of 32-bits per signal was built from the signal set in Table 2.1 (instead of the 64-bits per signal set used throughout this chapter). This was done by adding a dummy bit at the end of each signal in the set in Table 2.1. Now, if the watermark signal is divided into 32-bit segments, and the signal associated with each encoded symbol is repeated, each symbol will use $2 \times 32 = 64$ bits in two different segments, which is the same rate as 64 bits/signal. In the

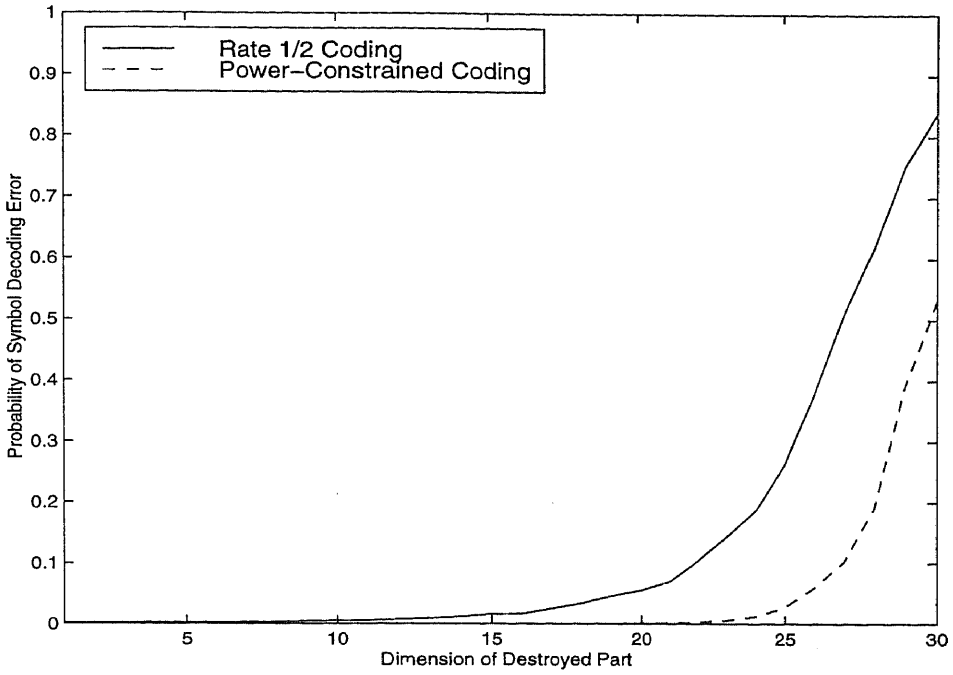


Figure 8.13 Power Constrained Versus Nonpower Constrained Coding for Bit-by-bit Interleaving, $SNR_{WM} = 0$ db

presence of burst errors and white Gaussian noise, the decoder correlates each 32-bit segment with all the entries in the signal set and decodes the signal with the highest correlation. The simulation program considers a symbol decoding error only if both of the 32-bit segments that correspond to a symbol decode the wrong symbol. The results of this comparison are shown in Figures 8.13 and 8.14. Figure 8.13 shows this comparison for bit-by-bit interleaving, where both the power-constrained and the nonpower-constrained approaches use bit-by-bit interleaving. Figure 8.14 shows this comparison for 32-bit interleaving, where both the power-constrained and the nonpower-constrained approaches use 32-bit interleaving. Clearly, for the same code rate, the power-constrained approach (which adds error-recovery redundancy by increasing the signal dimension) performs better than the nonpower-constrained approach (which adds error-recovery redundancy using error correction codes).

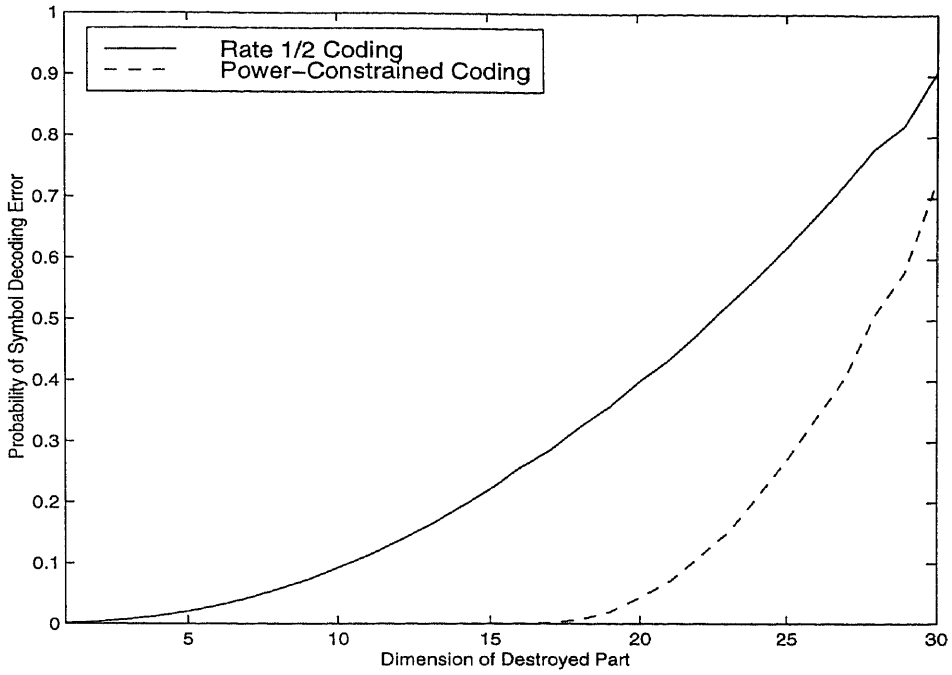


Figure 8.14 Power Constrained Versus Nonpower Constrained Coding for 32-bit Interleaving, $SNR_{WM} = 0$ db

8.7 Summary

This chapter studied 2-D interleaving of the watermark signal with respect to a still image. It was shown that for the interleaving process to obtain the best performance, it needs to consider three important factors:

1. Since the watermark signal layout is in 2-D, 2-D interleaving should be applied.
2. The signal should be interleaved on a bit-by-bit basis.
3. The watermark signal should be treated as a power-constrained signal. That is, the redundancy needed for error recovery should be added by increasing the signal dimension; not by using error correction codes.

Three interleaving strategies were compared, and it was established that because the watermark signal is power-constrained and because each bit is dealt with separately, the embedded signal should be interleaved on a bit-by-bit basis. It was also shown that this bit-by-bit interleaving is the most suitable approach, even

if the signal suffers from some background Gaussian noise in addition to the burst error for which we interleave. Erasure decoding was introduced for impulse noise and applied to quantization noise. It was shown how erasure decoding can reduce the probability of symbol decoding error without decreasing the number of symbols carried within the watermark signal. Interleaving was applied to the case with quantization noise and burst errors. It was also shown that with the bit-by-bit interleaving strategy, the power-constrained approach (which adds error-recovery redundancy by increasing the signal dimension), outperforms the nonpower-constrained approach (which adds error-recovery redundancy using error correction codes). In the next chapter, 3-D interleaving of a watermark signal embedded in a video sequence is presented.

CHAPTER 9

3-D INTERLEAVING OF THE WATERMARK SIGNAL FOR VIDEO SEQUENCES

As mentioned in Chapter 1, the layout of the watermark signal, with respect to a video sequence, is in 3-D. In this 3-D layout, the x and y coordinates are considered with respect to a single frame, and the z coordinate expresses the frame number within the video sequence. Thus, to increase the robustness of a watermark signal embedded in a video sequence, the signal needs to be interleaved in 3-D. In this chapter, the 3-D case of our interleaving technique (which was introduced in Section 6.4.2 and was discussed in detail in Section 7.3) is applied to the watermark signal.

In Chapter 8, with interleaving the watermark signal in 2-D, the following points are concluded:

- The best interleaving strategy is bit-by-bit.
- The decoding process of the interleaved signal should apply an optimum detection method (e.g., correlation detection).
- Error recovery redundancy should be added by increasing the signal dimension; not by using nonpower-constrained error-correction codes.

In this chapter, the results of the 2-D interleaving case provide some basis for the 3-D interleaving case. That is, 3-D interleaving is applied on a bit-by-bit basis, and the de-interleaved signal is decoded using correlation detection. Thus, the 2-D results will not be discussed for the 3-D case. Rather, this chapter concentrates on the type of error that can occur to a watermark signal embedded in a video sequence. As mentioned in Chapter 8 for 2-D interleaving—which applies to 3-D interleaving, as well—if the watermark signal does not carry a sequence of meaningful information (it is just one sequence of random variables), interleaving is irrelevant. Interleaving the watermark signal is intended for enhancing the robustness of information-symbol embedding techniques, and for the multiple signaling technique (which is the core of this dissertation) when we decode each segment separately.

Interleaving the watermark signal for a video sequence can increase the signal's robustness (decrease the probability of symbol decoding error) for two possible error scenarios.

1. 3-D error cluster:

   To understand how this scenario occurs, imagine a single frame in a video sequence. In this frame, a 2-D error cluster can occur, as explained in Chapter 8. That is, an attacker succeeds in destroying the watermark signal in a group of neighboring blocks (which may depend on the image texture). In a video sequence, when a block carries a certain texture, it is likely that a similar texture occurs in the same location in the proceeding frames. Thus, if a watermark signal has a 2-D error cluster in a single frame, the signal is likely to have a 3-D error cluster in a video sequence. 3-D interleaving should distribute this error cluster among all the symbols embedded in the entire sequence of frames, and hence reduce the probability of symbol decoding error.

2. Frame error:

   This error scenario can occur when the attacker focuses on destroying the signal in some selected frames within the video sequence. That is, the attack technique may have more knowledge of the watermark signal in some frames than others. For example, in a sequence of 32 frames, the attacker may be capable of destroying the signal within 16 randomly selected frames. Interleaving should allow the detection process of the watermark signal to retrieve the embedded symbols from the 16 frames with the least probability of symbol decoding error. Notice that a similar scenario can occur if some frames are lost (which can occur if the watermarked video sequence is transmitted over a noisy channel), and as a result the watermark signal decoding technique has to exclude the lost frames from the detection process. Three-dimensional interleaving should distribute a frame error among all the symbols embedded in the entire sequence, and hence increase the signal's robustness (decrease the probability of symbol decoding error).

This chapter proceeds as follows: Section 9.1 discusses how 3-D interleaving is applied to the watermark signal. Section 9.2 illustrates 3-D interleaving for a 3-D error cluster (the first error scenario). Section 9.3 illustrates 3-D interleaving for frame errors (the second error scenario). Section 9.4 is a summary.

9.1 How 3-D Interleaving is Applied

This section explains how 3-D interleaving is applied to a watermark signal embedded in a video sequence. The successive octonary partitioning algorithm explained in Section 7.3 is applied for a 32-frame video sequence. Recall how the 2-D interleaving case is implemented in Chapter 8. A 256×256 image is used with the DCT applied to 8×8 blocks. In the 2-D case, we dealt with 32×32 blocks per image. Here, we also deal with 32×32 blocks per image (frame). If we select a video sequence of 32 frames, we end up with $32 \times 32 \times 32 = 32768$ blocks within the entire sequence. Thus, if the algorithm discussed in Section 7.3 is used, the video sequence can be considered a 3-D case of size $2^p \times 2^p \times 2^p$ blocks, where $p = 5$. In this case, based on the algorithm discussed in Section 7.3, $\gamma_{max} = \frac{(2^p)^3}{2} = 16384$ blocks. To maximize the interleaving degree, the number of layers in the octonary partitioning tree, m, was selected to be 5. As a result, the achieved interleaving degree is $\gamma_5 = \gamma_{max}$. If the number of coefficients per block is selected to be 3, as was done throughout most of this work, we will use a total of $N = 32 \times 32 \times 32 \times 3 = 98304$ bits. If a maximum separable signal set is built with the number of bits per symbol $n = 64$ (as was done in Chapter 8), $98304/64 = 1536$ symbols will be embedded in the entire sequence. Note that with this utilized maximum separable signal set, the minimum Hamming distance between any two entries is 30 bits. As explained in Chapter 8, for such a set, symbol decoding error starts to occur for a symbol, a_i, when

$$n - \mu_i \geq \alpha, \tag{9.1}$$

where μ_i is the number of bits affected by the burst error and α is the minimum number of bits needed to decode a_i. For this signal set, $\alpha = 15$, as explained in Chapter 8.

In the simulation performed in this chapter, correlation detection is applied and bit-by-bit interleaving is carried out, as was done in the 2-D case. To do this, the video sequence is scanned three times. In each scan, the watermark signal encoder embeds 32768 bits. The location of each bit (the particular frame and the location within the frame) is decided by the 3-D interleaving algorithm, as explained in Section 7.3.2. The watermark signal decoder de-interleaves the extracted signal before applying correlation decoding.

9.2 3-D Interleaving for 3-D Error Clusters

This burst-error scenario is the 3-D analogy of the 2-D burst error discussed in Chapter 8. One can visualize the video sequence as a cube of size $32 \times 32 \times 32$ blocks and the error cluster as a cube of size $X_0 \times Y_0 \times Z_0$, where $X_0 = Y_0 = Z_0 < 32$. In Chapter 8 (with the 2-D case), we increased the 2-D error cluster size, and with each size we considered all possible positions the 2-D error cluster may take within the image (which is a 2-D array of 32×32 blocks). In this chapter (with the 3-D case), we also increase the volume of the 3-D error cluster, and with each volume we consider all possible positions the 3-D error cluster may take within the video sequence (which is a 3-D cube of $32 \times 32 \times 32$ blocks). All possible positions are considered equally. For example, if the error cluster is of volume $24 \times 24 \times 24$, there are 512 different locations the error cluster may take within the $32 \times 32 \times 32$ cube. All of these 512 locations are considered equally when collecting the statistics used to plot the experimental probability of symbol decoding error curves.

In Chapter 8, the 2-D interleaving technique was compared with two strategies used in writing a 1-D interleaved signal to a 2-D layout: the spiral pattern and the boustrophedonic pattern. Since there is no known interleaving pattern used in a known 3-D application, the simulation in this chapter starts by comparing 3-D interleaving with the noninterleaved case. (With the noninterleaved case, bits are written to the video sequence row-by-row and frame-by-frame). That is, we will show the achieved reduction in the probability of symbol decoding error obtained from applying interleaving. Figure 9.1 shows this comparison. The solid line shows

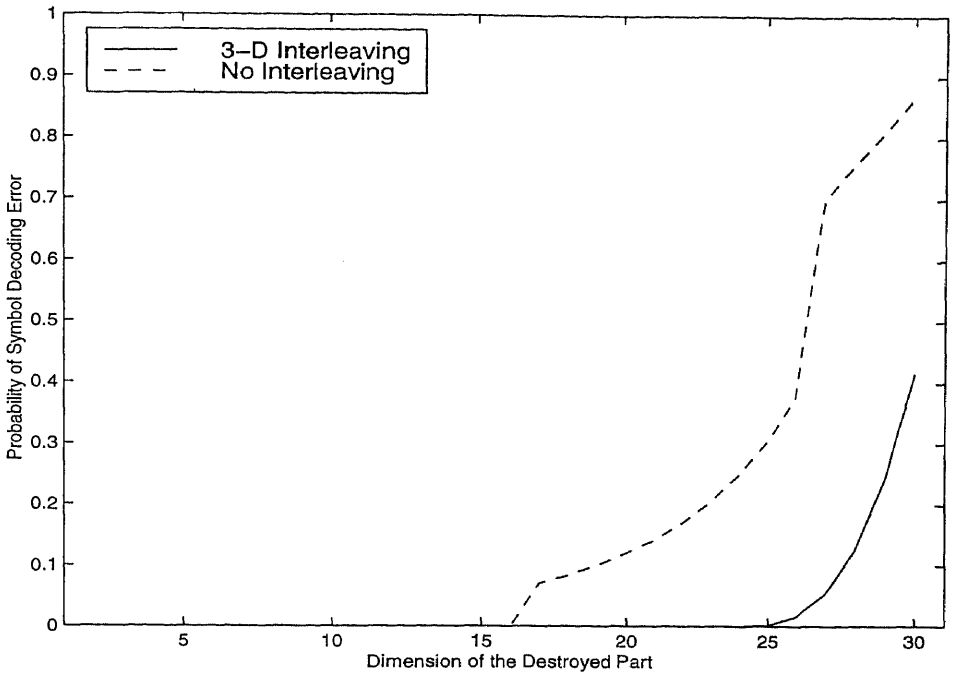


Figure 9.1 Probability of Symbol Decoding Error for 3-D Error-Clusters with Interleaving and without Interleaving

the interleaved case while the dashed line shows the noninterleaved case. With the interleaved case, one can see some parameters that resemble those carried out in Section 8.3 for the 2-D case. From Figure 9.1, it is apparent that symbol decoding error started to occur when the error cluster volume increased from $X_0 = Y_0 = Z_0 = 25$ to $X_0 = Y_0 = Z_0 = 26$. When the dimension of the error cluster is 25, the following parameters can be considered:

- Total number of blocks = 32768,
- Number of blocks in error = 15625,
- Number of blocks error free = 17143.

When the dimension of the error cluster is 26, the following parameters are obtained:

- Number of blocks in error = 17576,
- Number of blocks error free = 15192.

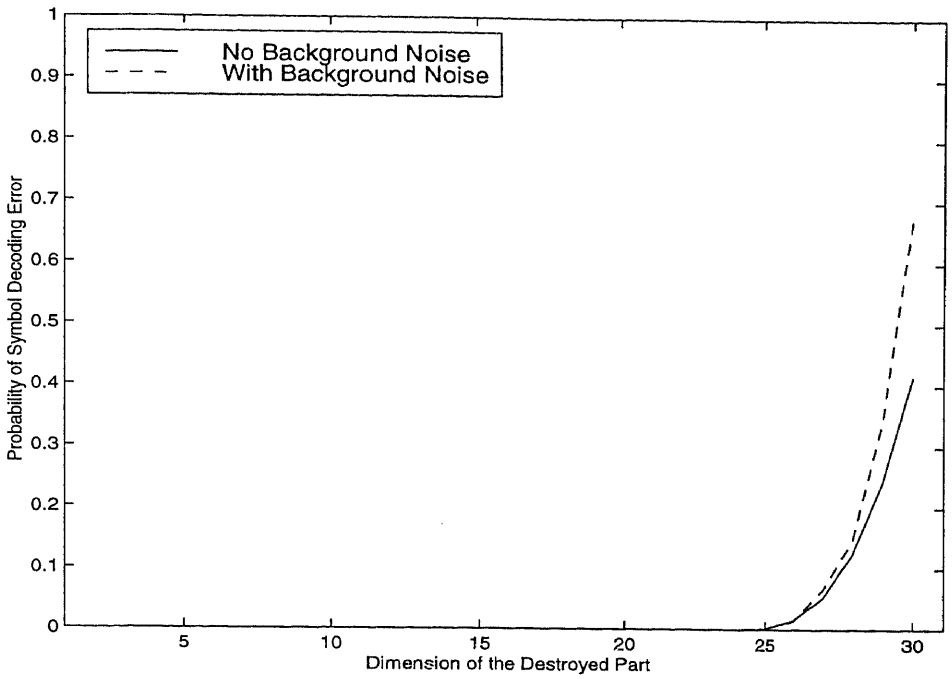


Figure 9.2 Probability of Symbol Decoding Error for 3-D Error-Clusters

That is, almost a zero probability of symbol decoding error can be achieved even though the error cluster destroyed about half of the encoded sequence.

With the 2-D interleaving in Chapter 8, the case when the watermark signal suffers from WGN, in addition to the 2-D error burst we interleave for, is simulated. For the 3-D case in this chapter, the case when the watermark signal suffers from WGN, in addition to the 3-D error burst we interleave for, is simulated. The obtained results are plotted in Figure 9.2. The solid-line curve is for the no-background-noise case (the same as Figure 9.1), while the dashed-line curve shows the background-noise case. In this simulation, the background noise resulted in an SNR_{wm} of 0 db.

Figure 9.3 shows the obtained probability of symbol decoding error as the variance (power) of the background noise increases. Naturally, as the noise power increases (lower SNR_{wm}), the probability of symbol decoding error increases.

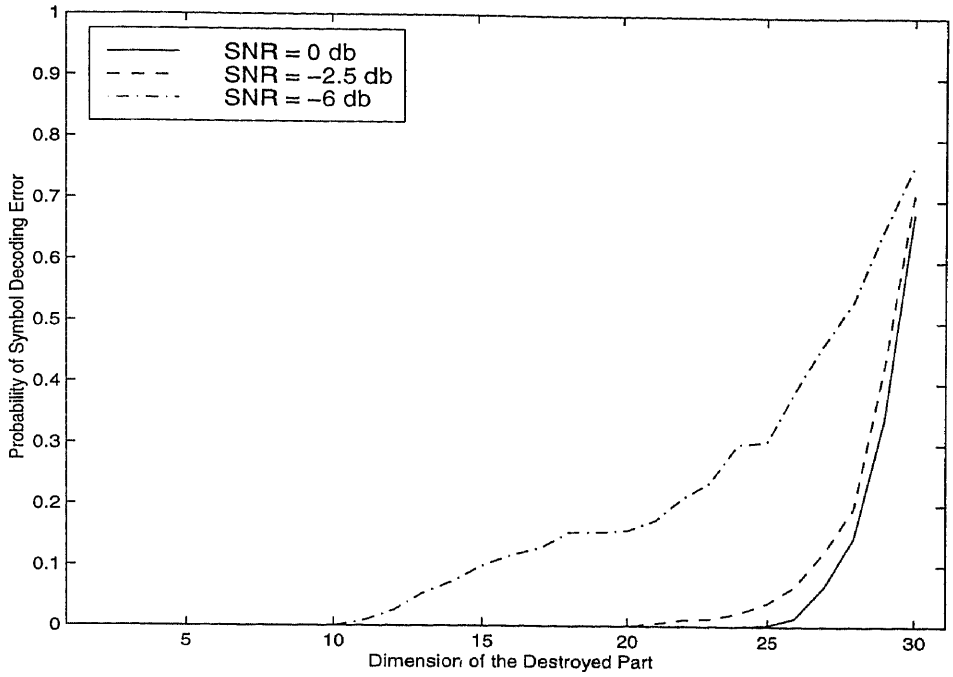


Figure 9.3 Probability of Symbol Decoding Error for Different Noise Levels in the Presence of 3-D Error-Clusters and Background WGN

9.3 3-D Interleaving for Frame Errors

This section covers the second possible error scenario, where a group of entire frames is excluded from the decoding process. As mentioned above, this can occur when some frames of the sequence are lost, or when an attacker succeeds in destroying the signal in some frames of the sequence, while other frames remain unaffected.

To simulate this error scenario, the error cluster size is selected such that $X_0 = 32$ and $Y_0 = 32$. The 3-D cluster volume is increased by increasing the number of lost frames (which is expressed in the Z coordinate). The obtained results are plotted in Figures 9.4, 9.5, and 9.6, where the horizontal axis is the number of lost (destroyed) frames, and the vertical axis is the resulting probability of symbol decoding error. As with the error cluster discussed in the previous section, all possible locations a cluster of frame errors can take are considered equally. For example, if the number of frames excluded from the decoding process is 16 frames, in a 32-frame video sequence there are 16 possible locations where this error cluster can be.

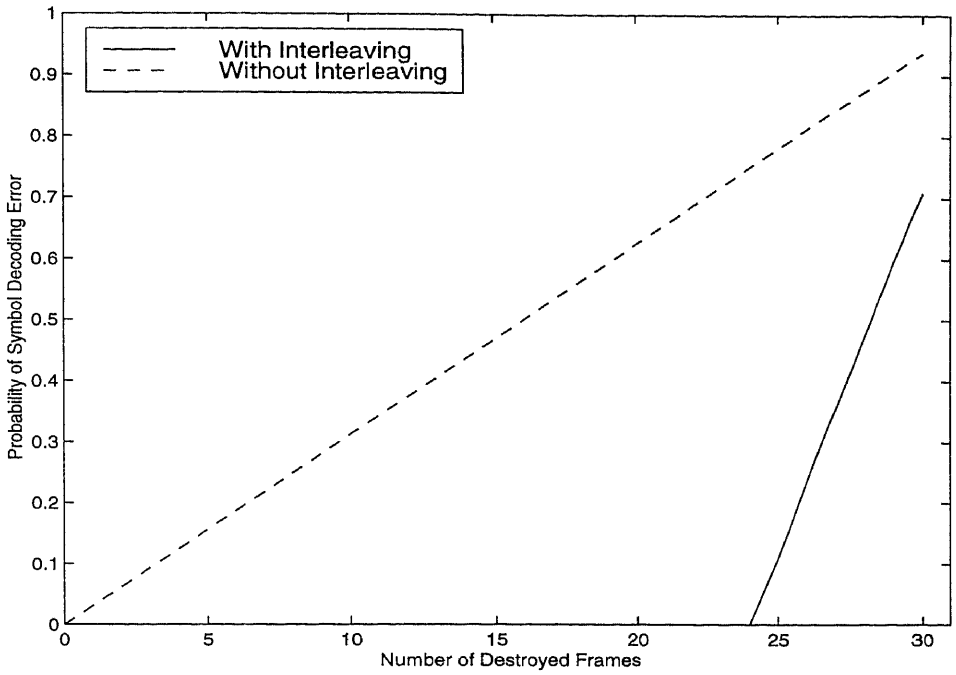


Figure 9.4 Probability of Symbol Decoding Error for Frame Errors with Interleaving and without Interleaving

Figure 9.4 compares the probability of symbol decoding error, when the signal is interleaved with that of the noninterleaved signal. With interleaving, symbol decoding error starts to occur when the number of destroyed frames exceeds 24, while without interleaving symbol decoding error occurs with the first destroyed frame.

Figure 9.5 compares the case of no background noise (just the burst error for which we interleave) and a case of additive Gaussian noise that results in an SNR_{wm} of 0 db.

Figure 9.6 shows the obtained results as the WGN variance increases. As one expects, when noise power increases (lower SNR_{wm}), the probability of symbol decoding error increases.

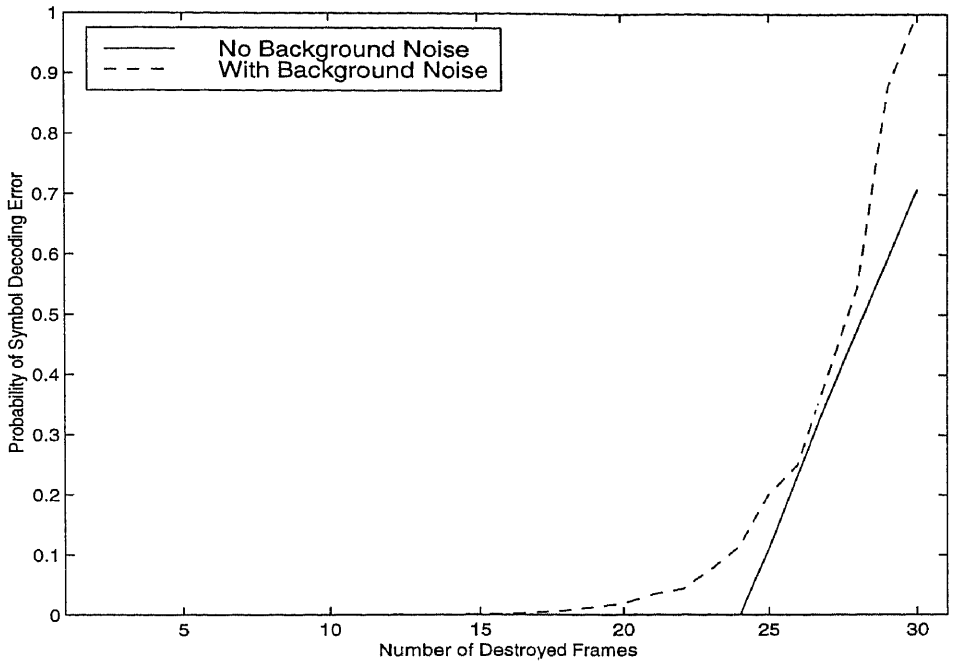


Figure 9.5 Probability of Symbol Decoding Error for Frame Errors without Background Noise and with Background Noise

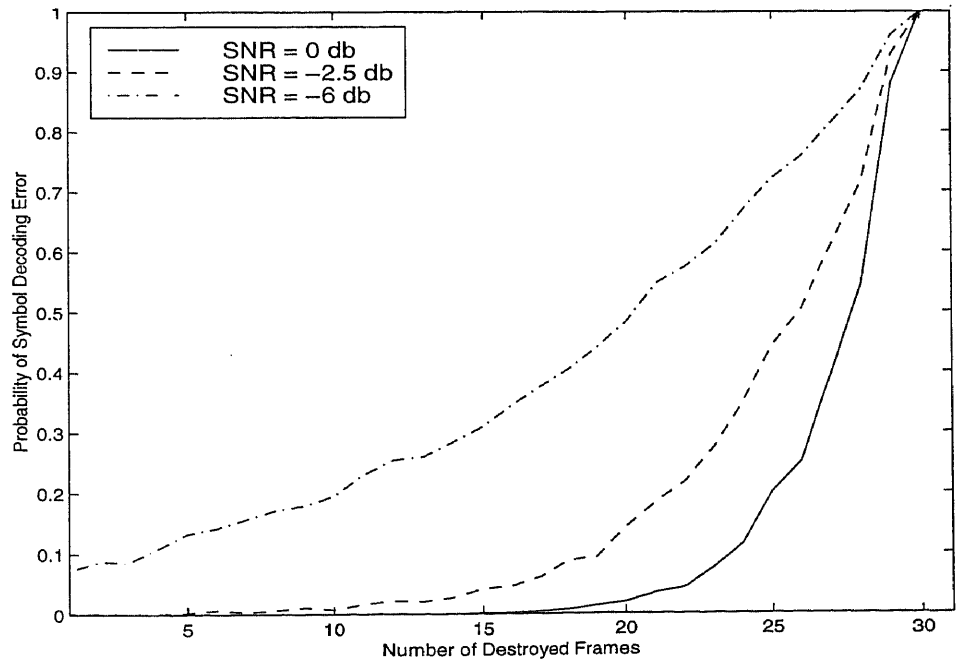


Figure 9.6 Probability of Symbol Decoding Error for Different Noise Levels in the Presence of Frame Errors and Background WGN

9.4 Summary

This chapter presented simulation results for the 3-D interleaving of a digital image watermark signal embedded in a video sequence. We presented results for two different error scenarios. With the first scenario, the watermark signal suffers from an error that destroys part of each frame (3-D error cluster). With the second scenario the signal is entirely lost in a group of frames (frame error). With both error scenarios, the interleaved case is compared with the noninterleaved case. Also, with both scenarios, the simulation is performed with the existence of background white Gaussian noise in addition to the error cluster for which we interleave, and with the existence solely of the error cluster.

CHAPTER 10

CONCLUSION

This dissertation focused on two aspects of digital image watermarking: detection and robustness. Following the introductory chapter, the dissertation is divided into two main parts. The first part is dedicated to the detection aspect of the signal, and the second part is dedicated to the robustness aspect. In this conclusion, we summarize the main contributions of this work, give some recommendations, and provide the scope of further research.

10.1 Contributions

The contributions of the first part of this work can be summarized as follows.

- In Chapter 2, a new multiple signaling approach was presented. This approach allows us to embed meaningful information in the watermark signal while keeping the signal a sequence of random variables. This approach was examined with regard to Shannon's coding theory by comparing the following three coding methods:

  1. Multiple embedding, where a single binary value is embedded in multiple coefficients. We showed how this is equivalent to using repetition codes in channel coding.
  2. Error correction coding, where we selected the error correction code BCH (31:6) for our comparison.
  3. Correlation decoding, where we used the same redundancy as with BCH (31:6) but in the form of increasing the signal dimension.

  We also showed that because the watermark signal is power constrained, correlation decoding gives the best performance. Further, we showed how Shannon's coding theory is applied to the watermark signal: As the watermark signal-to-noise ratio SNR_{wm} decreases, the amount of information that could be carried in the signal decreases.

- Chapter 3 presented a study on the capacity saturation of the watermark signal. We showed how the number of coefficients selected to carry the watermark signal, N, resembles the available bandwidth in a communication channel. We showed a tradeoff between the achieved capacity and the signal dimension. We studied the behavior of the image capacity when approaching the watermark signal as one sequence of random variables and when segmenting the watermark signal to carry meaningful information. We showed that the complexity of the detection process may increase as the signal dimension increases.

- In Chapter 4, the similarity between detecting the digital image watermark signal and detecting a signal over a spread spectrum communication channel with additive white Gaussian noise was explained. We studied the differences between the two, and showed how we can use sequence decoding, which offers a manageable increase in computational complexity as the signal dimension increases. A comparison between correlation detection and sequence detection was presented. We also presented a comparison between maximum likelihood sequence decoding (MLSD), maximum a-posteriori probability sequence decoding (MAPSD), and correlation sequence decoding (CORSD). We showed how MAPSD, in addition to offering a less complicated detection technique, can also offer a good measure of the confidence level of the detected signal.

The contributions of the second part of this work revolve around a novel multidimensional interleaving technique developed during the course of the research that lead to this dissertation. These contributions can be summarized as follows:

- Chapter 5 provided a tutorial of the new multidimensional interleaving. The technique was presented as the n-D equivalent to block interleaving (which is a type of 1-D interleaving) commonly used in communication channels. While block interleaving spreads consecutive elements across the interleaved block, the new algorithm spreads consecutive elements across n-D.

- Chapter 6 presented and analyzed an implementation method of this interleaving algorithm using a sliding window technique. This technique was

adapted from the 1-D case to the 2-D case. This chapter also presented simulation results that show the superiority of the algorithm over known interleaving techniques when applied to applications with nonpower-constrained signals in a 2-D layout.

- Chapter 7 presented a different implementation method for the multidimensional interleaving technique using a successive partitioning approach. This chapter provided discussion of the 3-D interleaving case in detail and presented the general interleaving case (n-D interleaving). This chapter also provided a comparison between the implementation of the sliding window interleaving approach and the implementation of the successive partitioning interleaving approach.

- Chapter 8 applied the 2-D version of the above interleaving algorithm to digital image watermark signals embedded in a still image. Two approaches for overcoming burst errors in digital image watermarking were compared. The first approach utilized interleaving, while the second approach did not utilize interleaving. It was shown that the interleaving approach is preferred. It was also shown how interleaving a power-constrained signal differs from interleaving a nonpower-constrained signal. The increase in watermark signaling robustness against burst errors, when the signal is interleaved, was presented. This chapter also examined the concept of applying erasure decoding to the detection process of the watermark signal. Cases where the signal suffers from background noise, in addition to the burst error (for which we interleave), were studied.

- Chapter 9 applied the 3-D version of the new interleaving technique to watermark signals embedded in a video sequence. The increase in the signal's robustness when applying 3-D interleaving was demonstrated with two error scenarios. It was shown that with this interleaving technique, the probability of symbol decoding error is very small even if an attacker destroys part of the watermark signal in a group of frames in the video sequence or in parts of each frame.

10.2 Recommendations

In this conclusion chapter, we would like to make some recommendations to the watermark research community. These recommendations are based on some of the contributions mentioned above, and they are summarized as follows:

- Avoid using error correction codes. The redundancy needed to overcome the noise introduced to the watermark signal should be applied in the form of increasing the signaling dimension rather than using error correction codes.

- Avoid increasing the signal dimension close to the saturation level. This can only increase computational complexity without significantly increasing achieved capacity. According to Shannon's capacity theorem, to accommodate more noise, the amount of information embedded in the watermark signal must be reduced.

- Use MAPSD instead of correlation detection if it is necessary to simplify the detection process. Most of the work done in watermark signal detection use correlation detection. Some publications even mention the use of certain error correction codes. The research in this dissertation offers MAPSD as an optimum detection technique, with which the increase in the signal dimension can be done without facing unmanageable computational complexity.

- For a watermark signal embedded in a still image, if the signal is segmented, consider using 2-D interleaving. Remember that the increase in signal robustness achieved with interleaving does not require sacrificing any of the information symbols carried within the signal. Interleaving could be useful in the case of 2-D error bursts and in cases with quantization noise.

- For a watermark signal embedded in a video sequence, consider using 3-D interleaving. Similar to the 2-D case, the increase in signal robustness achieved with interleaving does not require sacrificing any embedded information. Interleaving is useful in cases of 3-D error bursts and in frame errors or frame loss.

10.3 Scope of Further Research

The nature of the research in the first part of this dissertation differs from that in the second part. While the first part studied watermark-signal detection techniques in light of known communication principles, the second part opened an entirely new area of research with regard to multidimensional interleaving. Although there is always more work that can be done by applying more communication theory principles to the watermark signal (which is a relatively new area of research), multidimensional interleaving based on the techniques introduced in this work seems more interesting, since it can lead to new research with regard to both theory and applications. In this section we outline the research directions that seem most worthwhile.

10.3.1 Theory

More theoretical work on the proposed multidimensional interleaving technique can be done as follows:

1. Proving (theoretically) that the n-D case (see Section 7.4) holds for any n. This can be done by induction, that is, proving that the case holds if we move from n-D to (n+1)-D.

2. If possible, come up with an algebraic description of the n-D interleaving pattern using the theory of groups, finite fields, and vector spaces. One can consider that each element in the interleaving pattern (see Figure 7.5) is a vector. If all the elements of the interleaving pattern form a group, each pair of two elements forms a distinct coset. A principle similar to that of generation and parity-check matrices can be developed where an interleaving matrix and a de-interleaving matrix can be obtained for each interleaving case.

10.3.2 Applications

With regard to the applications of the new multidimensional interleaving technique, the following may be done:

1. The algorithm can be applied to applications other than watermarking, where the signal layout is multidimensional (e.g., 2-D bar coding [9]).

2. With regard to the 2-D case of the watermark signal, the performance of the algorithm can be tested in the presence of multiple error clusters.

3. The performance of the algorithm may be tested when the watermark signal suffers from a Stirmark attack [18] [24] [25].

4. With regard to the 3-D case of the watermark signal, the performance of the algorithm can be tested in the presence of random frame errors. (Chapter 9 only covered the case of a cluster of frame errors.) Similar to the 2-D case, multiple 3-D error clusters can be tested, as well.

REFERENCES

1. R.E. Blahut, *Principles and Practice of Information Theory.* Reading, MA: Addison-Wesley Publishing Co., 1990.

2. T.M. Cover and J.A. Thomas, *Elements of Information Theory.* New York, NY: John Wiley & Sons, Inc., 1991.

3. I.J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia," *IEEE Transactions on Image Processing,* vol. 6, no. 12, pp. 1673-1687, December 1997.

4. I.J. Cox and J.M.G. Linnartz, "Some General Methods for Tampering with Watermarks," *IEEE Journal on Selected Areas in Communications,* vol. 16, no. 4, pp. 587-593, May 1998.

5. S. Craver, N. Memon, B. Yeo, and M. M. Yeung, "Resolving Rightful Ownership with Invisible Watermarking Techniques: Limitations, Attacks, and Implications," *IEEE Journal on Selected Areas in Communications,* vol. 16, no. 4, pp. 573-586, May 1998.

6. G.F. Elmasry and Y.Q. Shi, "Interleaving Reed-Solomon Codes for 2-D Arrays: Applications in 2-D Bar Coding," submitted to the *IEEE Transactions on Communications,* June 1999.

7. G.F. Elmasry, J. Huang, and Y.Q. Shi, "Embedding Meaningful Information in Digital Image Watermarking," submitted to *Image Communication,* June 1998.

8. G.F. Elmasry and Y.Q. Shi, "Maximum Likelihood Sequence Decoding of Digital Image Watermarking," *Proceedings of SPIE: Security and Watermarking of Multimedia Contents,* vol. 3657, San Jose, CA, pp. 425-436, January 25-27, 1999.

9. G.F. Elmasry and Y. Q. Shi, "An Interleaving Technique for Error Correction in 2-D Bar Coding," *Proceedings of the Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing,* University of Illinois at Urbana-Champaign, pp. 49-58, September 23-25, 1998.

10. G.F. Elmasry, J. Huang, and Y.Q. Shi, "Capacity of Multiple Signaling in Digital Image Watermarking," *Proceedings of the Fourth International Conference on Information Systems Analysis and Synthesis* (ISAS), Orlando, FL, pp. 155-159, July 1998.

11. G.F. Elmasry and Y.Q. Shi, "MAP Symbol Decoding of Arithmetic Coding with Embedded Channel Coding," to be presented at the IEEE Wireless Communications and Networking Conference (WCNC'99), New Orleans, LA, September 1999.

12. R.C. Gonzalez and R.E. Woods, *Digital Image Processing.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1992.

13. J.R. Hernandez, F. Perez-Gonzalez, J.M. Rodrigues, and G. Nieto, "Performance Analysis of 2-D Multiple Amplitude Modulation Scheme for Data Hiding and Watermarking of Still Images," *IEEE Journal on Selected Areas in Communications*, vol. 16, No. 4, pp. 510-524, May 1998.

14. J. Huang and Y.Q. Shi, "An Adaptive Image Watermarking Scheme Based on Visual Masking," *IEE Electronics Letters*, vol. 34, no. 8, pp. 748-750, April 1998.

15. J. Huang, G.F. Elmasry, and Y.Q. Shi, "Power Constrained Multiple Signaling in Digital Image Watermarking," *Proceedings of the 1998 IEEE Second Workshop on Multimedia Signal Processing,* Redondo Beach, CA, pp. 388-393, December 1998.

16. K.A.S. Immink, "Reed-Solomon Codes and the Compact Disc," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 41-59, 1994.

17. M. Kutter, F. Jordan, and F. Bossen, "Digital Signature of Color Images Using Amplitude Modulation," *Proceedings of SPIE-EI 97,* pp. 578-589, February 13-14, 1997.

18. M. Kutter and F.A.P. Petitcolas, "A Fair Benchmark for Image Watermarking," *Proceedings of SPIE: Security and Watermarking of Multimedia Contents,* vol. 3657, San Jose, CA, pp. 226-239, January 25-27, 1999.

19. A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Second Ed. Reading, MA: Addison-Wesley Publishing Company, Inc., 1994.

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

21. S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

22. R.J. McEliece and L. Swanson, "Reed-Solomon Codes and the Exploration of the Solar System," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 25-40, 1994.

23. T. Pavlidis, J. Swartz, and Y.P. Wang, "Information Encoding with Two-Dimensional Bar-Codes," IEEE *Computer* magazine, pp. 18-28, June 1992.

24. F.A.P. Petitcolas, "Weakness of Existing Watermarking Schemes," http://www.cl.cam.ac.uk/~fapp2/watermarking/image_watermarking/≻, October 1997.

25. F.A.P. Petitcolas and M.G. Kuhn, "StirMark2," http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark/≻, November 1997.

26. J. Piatek, "Automotive Industry Recommends 2-D Standards," *Automatic I.D. News*, pp. 54-56, August 1994.

27. C.I. Podilchuk and W. Zeng, "Image-Adaptive Watermarking Using Visual Models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 525-539, May 1998.

28. A. Popplewell and J.J. O'Reilly, "Simple Technique for Constructing 2-Dimensional Maximum-run-length-limited Error-Control Codes," *IEE Electronics Letters*, vol. 28, no. 22, pp. 2060-2061, October 22, 1992.

29. J.G. Proakis and M. Salehi, *Communication Systems Engineering.* Upper Saddle River, NJ: Prentice-Hall, Inc., 1994.

30. M.B. Pursley, "Reed-Solomon Codes in Frequency-Hop Communications," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 150-174, 1994.

31. Y. Saitoh and H. Imai, "RS-Based Unidirectional Byte Error Control Codes Perform Better than RS Codes," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 272-291, 1994.

32. D. Sarwate, "Reed-Solomon Codes and the Design of Sequences for Spread-Sprectrum Multiple-Access," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 175-204, 1994.

33. M.K. Simon, S.M. Hinedi, and W.C Lindsey, *Digital Communication Techniques: Signal Design and Detection.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

34. J.R. Smith and B.O. Comiskey, "Modulation and Information Hiding in Images," *Proceedings of the Workshop on Information Hiding in Images*, University of Cambridge, UK, May 30-June 1, 1996.

35. M.D. Swanson, B. Zhu, and A.H. Tewfik, "Multiresolution Scene-Based Video Watermarking Using Perceptual Models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 540-550, May 1998.

36. A.B. Watson, "DCT Quantization Matrices Visually Optimized for Individual Images," *SPIE Human Vision, Visual Processing on Digital Display IV*, vol. 1913, pp. 202-216, 1993.

37. S.B. Wicker, *Error Control Systems for Digital Communication and Storage.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1995.

38. S.B. Wicker and M. Bartz, "Reed-Solomon Codes in Hybrid Automatic Repeat-Request Protocols," in *Reed-Solomon Codes and Their Applications,* S.B. Wicker and V.K. Bhargava (Eds.), New York: IEEE Press, pp. 125-149, 1994.

39. S.G. Wilson, *Digital Modulation and Coding.* Englewood Cliffs, NJ: Prentice-Hall Inc., 1996.

40. J.M. Wozencraft and I. M. Jacobs, *Principles of Communications Engineering.* Prospect Heights, IL: Waveland Press, Inc., 1990.

41. MaxiCode Briefing Document, UPS Inc., http://www.maxicode.com, 1996.

42. Uniform Symbology Specification Code One, AIM Technology Group, TSC040, 1994.