

REFERENCE ONLY

UNIVERSITY OF LONDON THESIS

Degree PW Year 2008 Name of Author WANG, Chin Kiong

COPYRIGHT

This is a thesis accepted for a Higher Degree of the University of London. It is an unpublished typescript and the copyright is held by the author. All persons consulting this thesis must read and abide by the Copyright Declaration below.

COPYRIGHT DECLARATION

I recognise that the copyright of the above-described thesis rests with the author and that no quotation from it or information derived from it may be published without the prior written consent of the author.

LOANS

Theses may not be lent to individuals, but the Senate House Library may lend a copy to approved libraries within the United Kingdom, for consultation solely on the premises of those libraries. Application should be made to: Inter-Library Loans, Senate House Library, Senate House, Malet Street, London WC1E 7HU.

REPRODUCTION

University of London theses may not be reproduced without explicit written permission from the Senate House Library. Enquiries should be addressed to the Theses Section of the Library. Regulations concerning reproduction vary according to the date of acceptance of the thesis and are listed below as guidelines.

- A. Before 1962. Permission granted only upon the prior written consent of the author. (The Senate House Library will provide addresses where possible).
B. 1962-1974. In many cases the author has agreed to permit copying upon completion of a Copyright Declaration.
C. 1975-1988. Most theses may be copied upon completion of a Copyright Declaration.
D. 1989 onwards. Most theses may be copied.

This thesis comes within category D.

[checked box] This copy has been deposited in the Library of UCL

[empty box] This copy has been deposited in the Senate House Library, Senate House, Malet Street, London WC1E 7HU.

Investigations on Dirty Paper Trellis Codes for Watermarking

Chin Kiong Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London.

Department of Electronic & Electrical Engineering
University College London

September 2007

UMI Number: U593608

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI U593608

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.

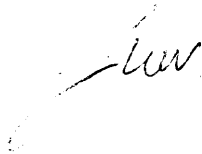


ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Declaration

I, Chin Kiong Wang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature of author:

A handwritten signature in black ink, appearing to read 'C. K. Wang', written over a faint dotted line.

Principal supervisor: Prof. Ingemar Cox (UCL)

Subsidiary supervisor: Dr. Gwenaël Doërr (UCL)

Internal examiner: Prof. Ebroul Izquierdo (Queen Mary, University of London)

External examiner: Assoc. Prof. Chang-Tsun Li (University of Warwick)

Acknowledgements

First of all, I would like to thank God for seeing me through my entire doctorate education in University College London (UCL), United Kingdom. I thank Him for His provision of all my needs and His presence during many of my ups and downs over this period of time. His care and love for me have given me much encouragement in carrying out my work.

Ph.D education is never complete without my principal supervisor, Prof. Ingemar Cox, and my subsidiary supervisor, Dr. Gwenaël Doërr, both from UCL. I would like to personally thank Prof. Cox for his dedication in guiding me through different stages of my academic life in UCL. Not only has he provided me with valuable knowledge on the subject of digital watermarking, he has ensured that I am equipped with the necessary writing and presentation skills. His experience has helped me a great deal in completing my research works and this thesis.

I am grateful to Dr. Doërr for his sincere guidance in many areas of information hiding. His wealth of knowledge on digital watermarking and other related areas provide many fruitful discussions. His professionalism and enthusiasm motivate me to seek for a high research standard. It has been a pleasure to work under him. Not forgetting Mr. Matthew Miller from NEC in the USA, he has provided several useful viewpoints on some areas of digital watermarking despite of his heavy workload.

My family has undoubtedly supported me throughout my education. I thank my parents for their constant words of encouragement and care and concern for me. Not only do they settle all of my personal issues back in Singapore, they are always eager to welcome me back to Singapore during my annual short trips. My friends back in Singapore, especially Lida Shi and Yanbin Wu, have provided endless encouragement during my four years in the UK.

My fellow brothers and sisters-in-Christ in New Life Bible-Presbyterian Church,

London, have always been around to encourage me. I would like to specially thank Mr. and Mrs. Chee Cheong Mok, Mr. and Mrs. Daniel Poh and family, Jonathan Kim, Yoong Chiang See-Toh, Constance, Joyce, Yanning, Evelyn, Joy Blundell, Grace Lim, Feng Qian Ang and Esmeralda Teo. They have made my stay in the UK an enjoyable and meaningful one.

Lastly, I would also like to thank the staffs and friends in the Adastral Park campus who are always very helpful and approachable. In particular, Richard Hutchinson and Neil Marjoram have been there when I need them to settle issues regarding network and computer equipment. Also, Brian Riley, Tracy Clarke and Chris Churcher always put in the extra effort in any of my administrative matters. Fellow students and friends include Qiao Li, Lin Lin, Shijie Zhang, Lei Zheng, Caijun Zhong, Jia Chen, Adetokunbo Bamidele, Vishwa Vinay and Oyewole Oyekoya have, in the past, provided me with different research ideas and also different viewpoints. All in all, UCL at Adastral Park has provided a conducive environment both for learning and for work.

Abstract

Recently, watermarking has been modelled as communications with side information at the transmitter. The advantage of this is that in theory the interference due to the cover Work or host signal can be eliminated, thereby improving the capacity of the watermarking system. Hence a number of different practical methods have been proposed, one of which is based on dirty paper trellis coding. These codes are a form of spherical code, and as such, have the advantage of being robust to amplitude scaling. Dirty paper trellises have a number of design parameters. There is a lack of understanding on the influence of these parameters on performance, and this thesis attempts to address this. In particular, the thesis examines the following parameters: (i) the number of states and the number of arcs per state in the trellis, (ii) the distribution of the codewords generated by the trellis, and (iii) the cost function associated with each arc. Experimental results are provided on both synthetic signals and real images that demonstrate how performance is affected and a number of suggestions and improved designs are discussed. In particular, a deeper understanding of trellis configurations is provided that serves as a foundation on which to choose the best trellis structure based on bit error rate performance and computational cost. Secondly, trellis coded modulation (TCM) is adapted for use in a dirty paper trellis. This results in an improved distribution of the codewords on the sphere which leads to improved performance. Lastly, during embedding, the embedder usually searches for the codeword that has the highest linear correlation with the cover Work. However, this codeword may be difficult to embed due to perceptual constraints. We show that searching for a codeword that maximises a cost function based on linear correlation and perceptual distance can significantly improve performance.

Contents

1	Introduction	13
1.1	Outline of the Thesis	17
1.2	Contributions	20
2	Communications and Digital Watermarking	21
2.1	A Communication System	22
2.2	Spread Spectrum Communications	28
2.2.1	Properties of Spread Spectrum Communications	29
2.2.2	Principles of Spread Spectrum Communications	30
2.3	Watermarking as a Communication System	33
2.4	Spread Spectrum Watermarking	35
2.5	Communications with Side Information at the Transmitter	36
2.6	Watermarking as Communications with Side Information	39
2.7	Improved Spread Spectrum	41
2.8	Practical Dirty Paper Coding	43
2.8.1	Lattice Codes	44
2.8.2	Syndrome Codes	47
2.8.3	Dirty Paper Spherical Codes	48
3	Dirty Paper Trellis Codes	51
3.1	Error Correcting Codes	52
3.1.1	Linear Block Codes	52
3.1.2	Convolutional Codes	56
3.2	Trellis Codes for Watermarking	62
3.3	Informed Coding	64

3.4	Parameters Affecting Dirty Paper Trellis	66
4	Trellis Structure and its Performance	69
4.1	Trellis Performance	70
4.2	Trellis Computational Cost	77
4.3	Summary	80
5	Using Trellis Coded Modulation to Improve Dirty Paper Trellis Codes	83
5.1	Trellis Coded Modulation	85
5.2	TCM Dirty Paper Trellis	93
5.3	Watermarking Using TCM DPTC	97
5.3.1	Vector Quantisation	98
5.3.2	Iterative Embedding Algorithm	99
5.3.3	Fixed Robustness Embedding	103
5.4	Summary	106
6	Using Perceptual Distance to Improve the Selection of Dirty Paper Trellis Codes for Watermarking	109
6.1	Watson’s DCT-based Visual Model	111
6.1.1	Sensitivity	111
6.1.2	Luminance Masking	112
6.1.3	Contrast Masking	112
6.1.4	Pooling	113
6.2	Fixed Fidelity Embedding	113
6.3	New Cost Function	115
6.4	Summary	119
7	Conclusion	124
7.1	Contributions and Future Work	124
A	Publications	127

List of Figures

1.1	A generic watermarking system	14
2.1	A simplified communication system.	23
2.2	Gaussian probability distribution function.	24
2.3	A extended communication system.	25
2.4	Different digital modulation schemes.	28
2.5	An example of frequency-hopping spread spectrum (FHSS).	31
2.6	An example of direct sequence spread spectrum (DSSS).	32
2.7	Watermarking system with informed detection.	33
2.8	Watermarking system with blind detection.	33
2.9	An example of a spread spectrum-based watermarking.	35
2.10	A communication model with two independent, additive noise sources.	37
2.11	Communications with side information at the transmitter.	38
2.12	Watermarking with blind embedders.	39
2.13	Watermarking as communication with side information at the transmitter.	41
2.14	Improved spread spectrum (ISS) scheme.	42
2.15	A simple example of quantisation index modulation (QIM)	45
2.16	Distributed source coding: Source coding with side information at the decoder.	47
2.17	A watermarking problem	47
2.18	Angle quantisation index modulation (AQIM)	49
3.1	A convolutional encoder with $\nu = 2$ memory blocks, $k = 1$ input bit and $n = 3$ output bits.	57
3.2	State machine representation for a four-state convolutional code.	59
3.3	Trellis representation of the convolutional encoder of Figure 3.1.	60

3.4	Viterbi decoding algorithm.	62
3.5	A traditional 8-state trellis.	64
3.6	A message selects a unique path through the trellis.	64
3.7	A dirty paper trellis with 8 states and 4 arcs per state.	65
3.8	A modified trellis that corresponds to a given message.	66
4.1	Different trellis configurations for 16 codewords: (a) 1 state, 4 arcs per state, length 2 trellis and (b) 4 states, 2 arcs per state, length 2 trellis.	71
4.2	An example of a simple additive embedding.	72
4.3	BER as a function of the number of codewords for different trellis configurations with no additive Gaussian noise.	75
4.4	Average normalised correlation (between a cover Word and the closest codeword that codes the intended message) as a function of the number of codewords for different trellis configurations.	76
4.5	A dirty paper trellis with 2 states and 4 arcs per state.	76
4.6	BER as a function of the number of codewords for different trellis configurations with -7 dB additive Gaussian noise.	77
4.7	BER vs. computation time for different trellis structures in the absence of additive Gaussian noise.	79
4.8	BER vs. actual computation time, in seconds, for different trellis structures in the absence of additive Gaussian noise.	80
4.9	BER vs. computation time for different trellis structures with additive Gaussian noise of $WNR = -7$ dB.	81
5.1	Convolutional encoder that takes 2 input bits at a time and produces 3 output bits.	85
5.2	Trellis diagram for the convolutional encoder in Figure 5.1.	86
5.3	Trellis coded modulation (TCM): A binary convolutional encoder with rate $R = k/(k + 1)$ is fed at time t with a k -bit binary word \mathbf{x}_t and a mapping function defines which symbol s_t to transmit depending on the $(k + 1)$ -bit encoded output \mathbf{y}_t	87
5.4	The Q-function, $Q(u)$, as a function of the variable u	88

5.5	Partitioning of 8PSK symbols. At each partition level, the considered set of symbols is further divided into two subsets so that the minimum distance within each resulting subset is increased ($\Delta_1 < \Delta_2 < \Delta_3$). This results in a binary tree whose branches are labelled with a 3-bit label.	90
5.6	Binary convolutional encoder with rate 2/3 and constraint length $\nu = 3$ giving the largest free distance d_{free}	91
5.7	The 8-state TCM trellis, which is generated by the convolutional encoder in Figure 5.6, uses the alphabet of 8PSK symbols whose partitioning is shown in Figure 5.5.	92
5.8	The dirty paper 8-state TCM trellis, which is generated by the convolutional encoder in Figure 5.6, uses the alphabet of 8PSK symbols whose partitioning is shown in Figure 5.5.	94
5.9	A convolutional encoder that generates a dirty paper TCM trellis having 128 states, 32 arcs per state, and 2×8 PSK symbols for every arc. . . .	96
5.10	BER performance comparison between random DPTC and TCM DPTC with synthetic signals and direction quantisation embedding. . . .	99
5.11	PER performance comparison between random DPTC and TCM DPTC with synthetic signals and direction quantisation embedding.	100
5.12	Geometric interpretation of the embedding region defined by specifying a fixed robustness σ_{max}	101
5.13	Geometric interpretation of the sub-optimal, iterative algorithm for informed embedding.	104
5.14	The 4 middle-frequency DCT coefficients (shaded in gray) used for embedding. The top-left corner DCT coefficient is the DC term.	105
5.15	BER performance between random DPTC and TCM DPTC with real images and iterative embedding.	106
5.16	MER performance between random DPTC and TCM DPTC with real images and iterative embedding.	107
6.1	Illustration of a case in which mean square error (MSE) is a poor measure of perceptual distortion.	110

6.2	An illustration of fixed fidelity embedding.	115
6.3	The 12 lowest-frequency DCT coefficients (shaded in gray) used for embedding. The top-left corner DCT coefficient is the DC term and is not used for embedding.	117
6.4	The bit error rate (BER) as a function of k for three values of fidelity. . .	119
6.5	The percentage improvement in bit error rate (BER) as a function of k for three values of fidelity.	120
6.6	The message error rate (MER) as a function of k for three values of fidelity.	121
6.7	The percentage improvement in message error rate (MER) as a function of k for three values of fidelity.	122
6.8	Images with different fidelity. (a) is an original image while (b), (c), and (d) are watermarked images with Watson distances 30, 50, and 100 respectively.	123

List of Tables

2.1	Source coding using fixed-length coding and Huffman coding	26
3.1	Standard array of a linear block code	55
3.2	Standard array and syndromes for a $(5, 2)$ linear block code.	56
5.1	Average mean square error between cover Works, c_o , and watermarked Works, c_w , with different bits of the 6-bit label output by the convolutional encoder as the <i>dirty</i> bit. If the second bit of the 6-bit label is chosen as the <i>dirty</i> bit, the distortion caused during vector quantisation embedding is the minimum.	96
5.2	Statistics of the MSE with both codes under study.	105
6.1	DCT frequency sensitivity table.	112

Chapter 1

Introduction

Rapid advancement in computer hardware and the evolution of the Web has enabled ordinary individuals to transfer perfect digital copies of multimedia content from one place to another. Large multimedia data such as music, video, computer software and pictures can be easily distributed across the internet in a short span of time at almost no cost. This raises the concern of several industrial groups, such as the RIAA (Recording Industry Association of America) and the MPAA (Motion Picture Association of America), regarding the issue of copyright protection as illegal file-sharing is claimed to have cost them billions of dollars. A number of industry technology groups, such as the CPTWG (Copy Protection Technical Working Group) and the SDMI (Strategic Digital Music Initiative), were established to tackle these concerns.

Illegal distribution of digital multimedia content, especially across the internet, has led to the development of digital watermarking in order to reduce piracy. Digital watermarking is a technique to either hide copyright information or verification messages (i.e. watermark messages) into multimedia content, or prevent unauthorised distribution. Such messages may contain information such as the name or place of a copyright holder. A digital watermark, generated by the watermark message and a watermark key, is a digital signal or pattern inserted into an original digital multimedia content, called the *cover Work*. The entire process is illustrated in Figure 1.1.

Digital watermarking is used for copyright protection of cover Works because of their three important attributes. Firstly, watermarks are imperceptible. They do not detract from the visual appearance of an image or the audio quality of a piece of music. Secondly, watermarks are inherently bound to the Works in which they are embedded. This means that they do not get removed when the Works are displayed or converted to

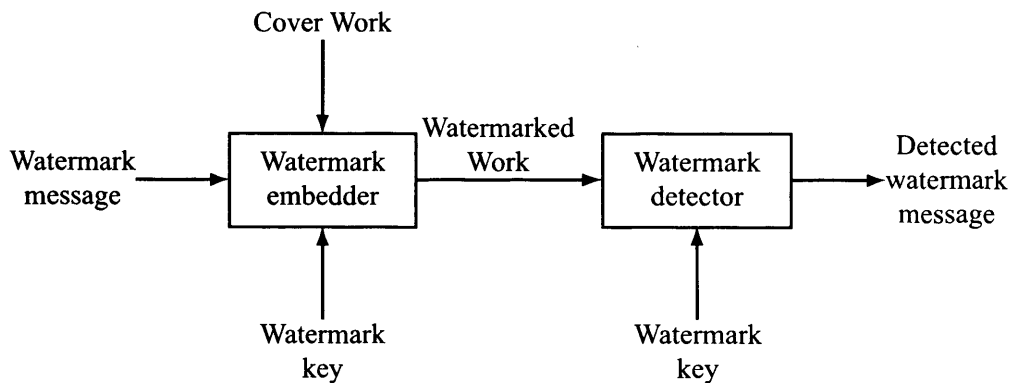


Figure 1.1: A generic watermarking system

other file formats. Lastly, watermarks undergo the same transformation as the Works. In this case, it is possible to learn something about the transformations by examining the resulting watermarks. Therefore digital watermarking can be invaluable to certain applications because of these attributes.

In general, watermarks can be classified into two categories: *semi-fragile* and *robust*. Semi-fragile watermarks are designed to lose their validity after even the slightest modification, except for some intended processes such as compression, to the watermarked content called the *watermarked Work*. These watermarks are employed to detect the presence of an intentional attack. On the other hand, robust watermarks, as the name suggests, are supposed to resist any alteration, such as channel noise or common signal processing, to the watermarked Work. Under such operating conditions, the watermark detector is still able to detect the embedded watermark. The use of either fragile or robust watermarks depends on the application.

Since digital watermarks are imperceptible, inseparable from the cover Works, and undergo the same transformation as the cover Works, digital watermarking can be useful in a variety of applications [CMB00]. For example, digital watermarking can be used for broadcast monitoring. The identification of the embedded watermark helps to monitor when advertisements were broadcasted. Another use of watermarking is owner identification whereby copyright holders can distribute their works without losing their rights. Watermarking is superior to textual copyright notices since the former is imperceptible and inseparable from the Work while the latter may be detracted from the visual appearance of the image, or easily removed by cropping. The most famous case

of copyright violation is the use of a photography of Lena Sjööblom by many image processing researchers without permission from its rightful owner, Playboy Enterprises, Inc [pla72]. Transaction tracking is another area in which an embedded watermark can be used to track the source of an illegally distributed Work. An example of this is in the distribution of Oscars-nominated preview DVDs, known as screeners, to award judges. When illegal copies of these screeners appeared on the Internet, the transaction watermark embedded in them were successfully used to determine their source.

Another application of watermarking is content authentication since the embedded watermark (or signature) is inseparable from the cover Work. Any slight modification to the cover Work will cause the watermark to be modified and thus not be detected. In this case, the watermark is termed a *semi-fragile* watermark. If the watermark is not detected, the owner will know that the Work has been modified. Since the watermark undergoes the same modification process as the Work, we can therefore learn from the watermark to determine how the Work has been tampered. Similar studies have been carried out by many researchers [WD96]. Lastly, watermarking can be used for copy control. Although cryptography is popular in copy control applications, it provides no protection after decryption, which is essential for viewing. Watermarking compliments cryptography by providing protection even after decryption. The CPTWG (Copy Protection Technical Working Group) has been interested in watermarking for copy control use in video DVDs. Readers are directed to [BCK⁺99] for more details on copy control.

Watermarking systems are often characterised by a number of properties whose importance depends on the requirements of the application. In certain situations, some properties can be more important than the others, depending on the role the watermark should play. Some of these properties are:

Robustness. Robustness refers to the ability of a watermark to survive common signal processing operations. Examples of such operations are

- analog-to-digital or digital-to-analog conversion, resampling, requantisation, lossy compression, lowpass filtering, etc.
- rotation, translation, scaling, cropping, etc.

If a watermark is to be detected at the watermark detector, it has to survive some (or all) of the transformations that are expected to be encountered during the

transmission of the watermarked Work. However, in some scenarios such as when semi-fragile watermarks are used, robustness is totally irrelevant. Therefore depending on the application, the importance of robustness varies.

Tamper resistance. Tamper resistance refers to the ability to resist hostile attacks, usually with the intention of eliminating the watermark. These kinds of attacks can involve unauthorised removal or embedding of a watermark. Two common forms of such attacks are collusion and forgery. Collusion is a process whereby the attacker gathers several watermarked copies of a given Work, each with a different watermark embedded, and combines them to produce a copy with no watermark or totally different watermark. On the other hand, forgery involves an unauthorised embedding of an illegitimate watermark so that the detector incorrectly identifies the false watermark as legitimate.

Fidelity. Fidelity is defined as the perceptual similarity between the original and the watermarked version of a cover Work. The watermark should ideally be imperceptible so that it does not affect the viewing/listening of the multimedia content. However, in certain circumstances, mild perceptibility is tolerated for a higher robustness or lower cost of implementing a watermarking system.

Embedding Effectiveness. A positive detection occurs if the correct watermark is identified at the detector. Using this definition of positive detection, the effectiveness of a watermarking system is defined as the probability of correct detection immediately after watermark embedding. Ideally, the effectiveness should be 100% but this may come at a high cost if other properties are to be met as well. Depending on the application, some effectiveness may be sacrificed for the other requirements, such as high fidelity.

Digital watermarking is different from two related technologies: *cryptography* and *steganography*. Cryptography [TW01] is another content protection technology that has been widely used in many practical applications. As compared to watermarked Works which are still perceptible to human, encrypted contents are totally pseudorandom and hence are not perceptible at all unless the encrypted contents are decrypted. However, the encrypted contents are no longer protected after decryption whereas wa-

terminals remain bound to the watermarked Works. As for steganography [KP00], which is the art of writing hidden message such that the existence of the message is only known to the intended recipient, the steganographic message will generally appear in something unrelated to itself such as a picture, an article or another message called the *covertext*. Hence there is luxury of choosing any media to conceal the steganographic message. The main priority of steganography is to conceal the very existence of the hidden message and it is usually not resistant to any distortion of the *stegotext*. In contrast, while digital watermarking is imperceptible, it is common to inform the public that content is watermarked, as this acts as a deterrent to copying. In addition, a watermark is often designed to be robust to many common signal processing operations, e.g. lossy compression, that a Work may undergo.

1.1 Outline of the Thesis

Digital watermarking has been extensively studied for multimedia content. Many different methods have been devised to deal with various watermarking scenarios with each scenario focusing on one or more specific properties of watermarking depending on the intended applications. In this thesis, we will focus our attention on robust watermarking for images, i.e. designing watermarks that are resistant to common signal processes.

In general, watermarking systems can be modelled as communication systems because of the similarities involved. The advantage of such a relationship between watermarking and communication systems is that it enables some useful communication concepts to be implemented in watermarking applications to improve watermarking performance. In Chapter 2, some ideas on communications and watermarking systems and their techniques are covered. Section 2.1 starts with a description of a general communication system. One important communications concept, which can be useful for watermarking purposes, is then briefly mentioned in Section 2.2. This communication technique, known as *spread spectrum*, enables the original information to be “spread” over a larger range of frequencies used for transmission. A watermarking system that models a communication system is then described in Section 2.3 to illustrate the similarities between both systems. These similarities enable watermarking systems to employ the concept of spread spectrum communication so as to improve its performance

through “spreading” of the original information and details can be found in Section 2.4.

Before the late 1990s, watermarking systems were modelled as simple communication systems. At the end of the 20th century, researchers recognised that watermarking systems can be better modelled as another communication system called communications with side information at the transmitter, which is described in Section 2.5. This new communication system models the transmission channel as two noise sources and the interference of one noise source on the transmitted signal is totally eliminated if this noise source is known to the transmitter. Since the original Work is normally available to the watermark embedder (i.e. transmitter), the new communication concept enables interference elimination of an original Work, which is acting as a noise source initially, on the watermarked version of the same Work. As such, the performance of the watermarking system will not be affected by the original Work. Details on watermarking systems modelled as communications with side information at the transmitter are covered in Section 2.6. Upon recognising the fact that the original Work can be considered as side information, Section 2.7 illustrates how spread spectrum watermarking can be enhanced by introducing the concept of improved spread spectrum (ISS), which simply takes into account the availability of the original Work at the watermark embedder.

In Section 2.8, three types of practical embedding strategies are described and they have been proposed by others to make use of the fact that watermarking can be regarded as communications with side information at the transmitter: *lattice codes*, *syndrome codes*, and *dirty paper spherical codes*. Lattice codes are designed with computational simplicity in mind whereas syndrome codes are proposed to make use of error-correcting codes. However, both lattice codes and syndrome codes are susceptible to a common signal processing operation, i.e. amplitude scaling, which is multiplying the Work by a scaling factor. To solve this problem, another class of codes called dirty paper spherical codes was proposed since scaling only affects the magnitude of the code and not the decoding decision if spherical codes are used. A specific subclass, called dirty paper trellis codes, has been introduced to incorporate spherical codes with coding techniques to achieve a much better performance. Dirty paper trellis codes are the focus of this thesis.

To understand the principles of dirty paper trellis codes, a brief description on error correcting codes is required. Hence Chapter 3 begins by explaining two types of error

correcting codes. One of them is trellis coding which involves a trellis structure that consists of many paths and each path corresponds to a codeword. In this case, there is one-to-one mapping between messages and codewords. An example of watermarking using this trellis is illustrated. But this type of watermarking does not make use of the fact that the original Work is available to the watermark embedder. In order to do so, dirty paper trellis codes are designed to provide a one-to-many mapping between messages and codewords, instead of a one-to-one mapping. This one-to-many mapping allows a nearer codeword to be chosen for embedding, thus incurring less distortion.

The performance of dirty paper trellis codes has not been fully examined. Without a good understanding of its performance, one may encounter problems in trying to correctly implement dirty paper trellis codes for practical watermarking purposes. In this thesis, we look at some factors that can influence the performance of dirty paper trellis codes and thus its selection for watermarking. Chapter 4 first looks at the performance in relation to different trellis structures and then takes into consideration the computation cost in the selection of a trellis structure. Hence we can have a more informed choice on the different trellis configurations having the same performance, i.e. the one with the least computation cost will be chosen. In the end, we are able to implement a watermarking system which achieves a specified performance and yet operating at a lower computation cost.

To further understand dirty paper trellis codes, it is noted that the distribution of codewords generated by a trellis can affect the performance of the trellis. In other words, a set of well-separated codewords generally brings about an enhanced performance for the trellis. Current dirty paper trellis codes may generate codewords that are not well-distributed. In view of this, Chapter 5 describes the use of a new trellis with better codeword distribution and presents experimental results to show that this new trellis performs better than the original one.

The codeword chosen for watermark embedding corresponds to a path through the trellis. The selection of this path depends on the cost associated with each path. Currently, this cost function is defined as the linear correlation between an original Work and the codewords. However, this does not guarantee that the linear correlation between the watermarked version of the Work and the codewords are maximised if a perceptual constraint is imposed that will limit the embedding strength of the chosen codeword. In

Chapter 6, a new cost function, that takes into account both linear correlation and perceptual distance, is proposed. Using this proposed cost function, experimental results have shown that significant improvement in performance can be achieved.

Finally, Chapter 7 summarises the usefulness of dirty paper trellis codes for watermarking use and then highlights the major contributions of this thesis. Some areas for further research are also provided in this chapter.

1.2 Contributions

Several design parameters can influence the performance of dirty paper trellises but the relationship between design parameters and performance has not been fully examined by other researchers [MDC02, MDC04]. This thesis attempts to improve watermarking performance by first identifying some of these parameters and then suggesting different dirty paper trellis designs upon a closer examination of these parameters. In particular, this thesis examines the following design parameters: (i) the number of states and the number of arcs per state in the trellis, (ii) the distribution of the codewords generated by the trellis, and (iii) the cost function associated with each arc.

Experiments are carried out on both synthetic signals and real images to demonstrate the influence of the design parameters on watermarking performance. In this thesis, several suggestions are discussed and trellis designs proposed to enhance watermarking performance. Specifically, a deeper understanding on how to choose an optimum trellis configuration in different noiseless and noisy channel environments is provided in Chapter 4 by taking into account both the bit error rate performance and the trellis computational cost. Secondly, trellis coded modulation (TCM) is adapted for use in a dirty paper trellis so that the codewords generated by the trellis are better distributed on the surface of a sphere. An improved codeword distribution leads to a better watermarking performance and this is shown in Chapter 5. Finally, the selection of codeword for embedding is proven to be significantly improved by maximising a cost function that is a linear combination of linear correlation and perceptual distortion, instead of just linear correlation alone as is commonly used [MDC02, MDC04]. Chapter 6 demonstrates that this improved choice of codeword leads to an enhanced watermarking performance.

Chapter 2

Communications and Digital

Watermarking

Digital watermarking can be viewed as a process which consists of several stages, each serving a specific purpose. Before going straight into the detailed, technical working principles of each of these stages, it is necessary to first describe a generic communication model which is similar to a watermarking model. With that in mind, a general communication system, with its different stages, is described in Section 2.1. This is followed by a description of one of the most important digital communications techniques - *spread spectrum* - in Section 2.2, together with its useful properties. Spread spectrum involves the “spreading” of the original information across a wider band of frequencies at the sender and gathering the spread information from those frequencies at the receiver so that the information can better resist noise and be hidden from eavesdroppers.

A general watermarking system is then described in Section 2.3, with the similarities between watermarking systems and communications systems highlighted. The advantage of highlighting such similarities is to allow watermarking systems to make full use of the advancement in communications techniques. A simple communications model of watermarking models the cover Work, i.e. the original media content, as a channel noise source. Distortions that are present after watermark embedding will be considered as a second channel noise source. Because the properties of spread spectrum are also applicable and beneficial to watermarking, spread spectrum techniques can be applied to watermarking and Section 2.4 gives an example of a simple spread spectrum watermarking scheme.

Another important communication concept is communications with side information at the transmitter which is described in details in Section 2.5. This *side information* contains knowledge on the communication link between the sender and the receiver. Specifically, Costa [Cos83] showed that for a channel with two independent noise sources, only one of which is entirely known to the transmitter whereas the other is the channel noise, but neither of which is known to the receiver, the channel capacity is equivalent to a channel in which the first noise source is absent, even when the transmitter is subjected to a power constraint. Because of similarities between watermarking and communications with side information at the transmitter, i.e. the cover Work (first noise source) is known to the watermark embedder, watermarking can be modelled as such. Section 2.6 provides details on why and how watermarking can be modelled as communications as side information at the transmitter.

Watermarking employing spread spectrum makes no use of the concept of communications with side information at the transmitter. However, a better spread spectrum-based watermarking has been developed to incorporate this new communication concept for performance enhancement. This is discussed in Section 2.7.

Finally, three practical implementations of watermarking systems, i.e. *lattice codes*, *syndrome codes*, and *dirty paper spherical codes*, using the above-mentioned communication concepts are illustrated in Section 2.8. Brief descriptions on those methods and the advantages and disadvantages of using them are discussed. Because of the advantages of dirty paper spherical codes over lattice and syndrome codes, Chapter 3 provides a detailed discussion of dirty paper spherical codes.

2.1 A Communication System

A simplified communication model, shown in Figure 2.1, consists of a transmitter (or sender), a receiver, and a communication channel (or link). An information source is fed through the transmitter for some processing to produce a signal x . Then this signal, x , is sent through the communication channel to the receiver. This communication channel is a physical medium used to send (or transmit) information between the transmitter and the receiver. When the information is sent through this channel, it is corrupted in a random manner by noise. The simplest and most commonly used channel model is additive Gaussian noise, denoted by n , which is usually characterised by

random variables having mean zero, variance σ_n^2 , and a probability distribution function (PDF)

$$P(n) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-n^2/2\sigma_n^2} \quad (2.1)$$

with

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_n} e^{-n^2/2\sigma_n^2} dn = 1 \quad (2.2)$$

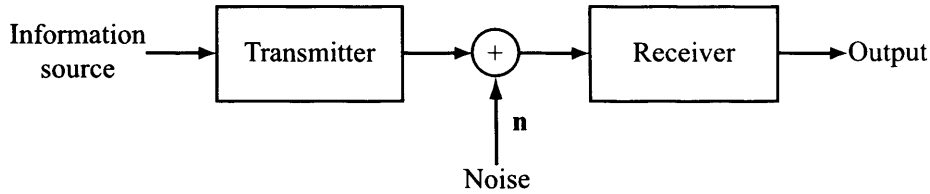


Figure 2.1: A simplified communication system.

The probability distribution function (PDF) indicates, in relative terms, how probable a random variable takes a particular value. For example, Figure 2.2 shows the probability distribution function (PDF) of a Gaussian-distributed random variable n with various standard deviations¹, σ , i.e. the PDF is represented in the form $\mathcal{N}(0, \sigma)$. When the PDF takes on the form $\mathcal{N}(0, \sigma)$, the random variable taking the zero value is more likely for a PDF with $\sigma = 1.0$ than for that with $\sigma = 2.0$. Naturally, Equation 2.2 must be satisfied because the probabilities of a random variable having all possible values must sum to one.

Once the corrupted signal, y , is captured at the receiver, an estimate of the transmitted message is produced. In order to maximise the chance of correct estimation, the receiver determines the most probable transmitted signal, x , by maximising the conditional probability of obtaining signal, y , given that a signal, x , is sent.

A fundamental question is how much information can be transmitted through this noisy communication channel from the transmitter to the receiver such that the probability of error is negligible. The capacity of such a channel per unit transmission has been defined by Shannon [Sha48] as

$$C = \max_{p(x)} I(\mathbf{X}; \mathbf{Y}) \quad (2.3)$$

¹Standard deviation is the square root of a variance

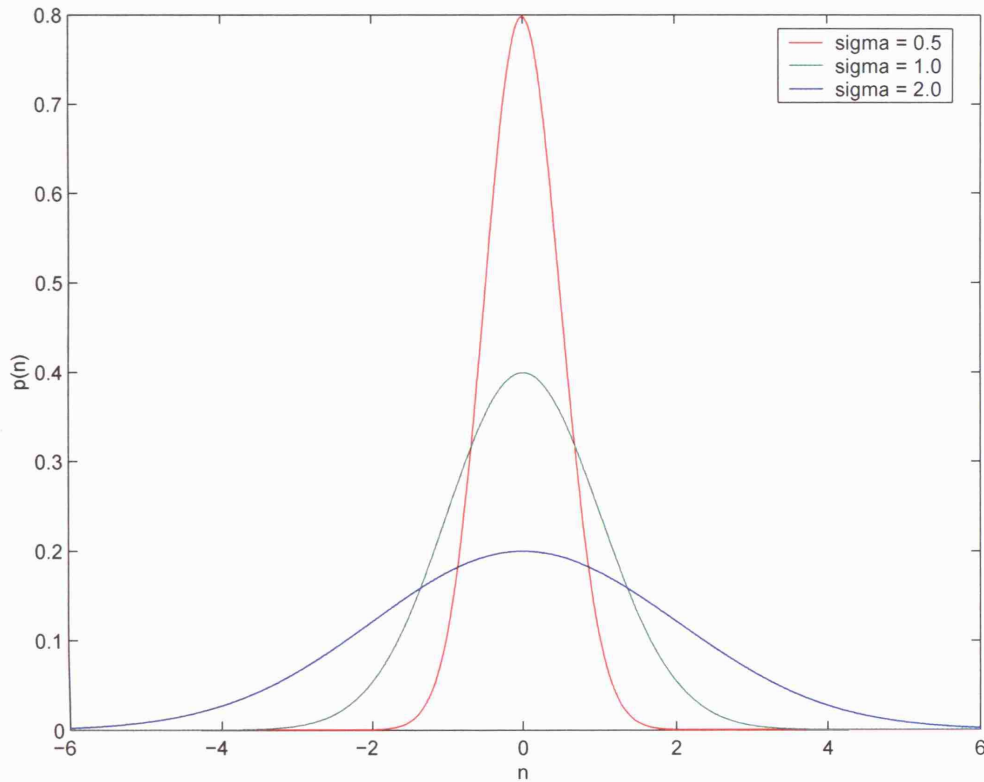


Figure 2.2: Gaussian probability distribution function.

where $p(x)$ is the probability distribution function (PDF) of the transmitted signal x , \mathbf{X} and \mathbf{Y} are the random variables of the transmitted and received signals (x and y) respectively, and $I(\mathbf{X}; \mathbf{Y})$ is the mutual information between x and y . By constraining the power of the transmitted signal, i.e. $E(X^2) \leq P_{av}$, where P_{av} is the maximum allowable transmission power, and performing maximisation of the $I(\mathbf{X}; \mathbf{Y})$ over all possible $p(x)$, the channel capacity is then given as

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P_{av}}{\sigma_n^2} \right) \quad (2.4)$$

$$= \frac{1}{2} \log_2 \left(1 + \frac{\sigma_x^2}{\sigma_n^2} \right) \text{ bits/transmission.} \quad (2.5)$$

Readers who are interested in the detailed explanation of the mutual information between \mathbf{X} and \mathbf{Y} , i.e. $I(\mathbf{X}; \mathbf{Y})$, and the derivation of the channel capacity listed in Equation 2.5 can refer to Sections 2.5 and 2.6 of [GHW92].

According to Shannon's noisy channel coding theorem, there exists channel codes and decoders (their brief descriptions follow immediately after this paragraph) to make

a reliable communication, at a very small error probability, if the data transmission rate, R , is less than the channel capacity C . However, if $R > C$, then there is no code at all that can achieve transmission at a very small error probability.

The simplified communication model, as shown in Figure 2.1, can be extended to include the following stages at the transmitter side: source encoder, channel encoder, and modulator. Source decoder, channel decoder and demodulator are present at the receiver side to perform the opposite functions. This extended communication model is shown in Figure 2.3.

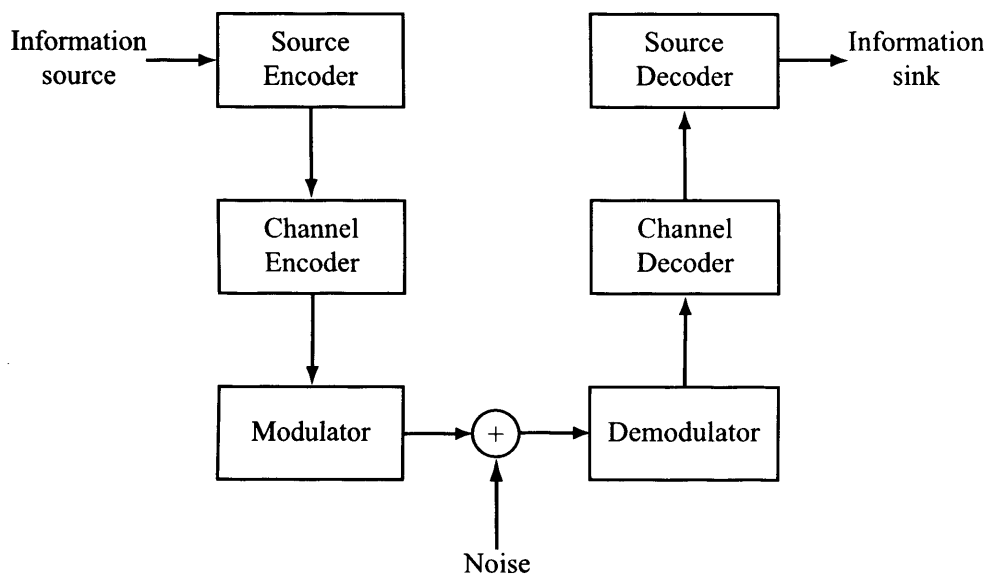


Figure 2.3: A extended communication system.

Raw information is first processed by a source encoder. The function of the source encoder is to use fewer bits (or other information-bearing units) than an uncoded information source to represent the original information source, i.e. data compression. Naturally, one would consider using as few bits as possible to present the given original data. Consider an alphabet, \mathcal{A} , consisting of 8 letters, each with a probability of occurrence as shown in Table 2.1. There are several ways in which the letters in this alphabet can be represented. One very simple way is to use a 3-bit symbol to represent each of the 8 letters, i.e. fixed-length coding. A cleverer way is to use variable length symbols, depending on the probabilities of occurrence of each of the letters. This can be done by a well-known variable-length code called a Huffman code². Refer to Table

²To know how Huffman coding works, please consult Section 3.3 of [Pro01]

2.1 to see the respective representations using fixed-length coding and Huffman coding. Although the Huffman code uses more than 3 bits to represent some letters, i.e. x_7 and x_8 , the average number of bits for Huffman code is 2.70, which is lower than that of the fixed-length code. Hence the Huffman code removes redundancy present in the data set and achieves data compression.

Letter	Probability	Fixed-Length Code	Huffman Code
x_1	0.36	000	00
x_2	0.14	001	010
x_3	0.13	010	011
x_4	0.12	011	100
x_5	0.10	100	101
x_6	0.09	101	110
x_7	0.04	110	1110
x_8	0.02	111	1111
Average symbol length:		3.00	2.70

Table 2.1: Source coding using fixed-length coding and Huffman coding

Huffman coding is a type of lossless data compression, i.e. the exact original data can be obtained from the compressed data. Since there is a one-to-one mapping between a letter and a symbol for Huffman coding, the letter, x_i , can be reconstructed given a symbol during source decoding. On the other hand, data compression is lossy if the reconstructed data from the compressed data is different from the original data, but sufficiently close for practical purposes. There are applications that require lossy compression due to bandwidth or storage constraints. There is always a trade-off between the amount of compression to be achieved and the distortion caused by the compression. Whenever lossy compression is carried out, it is not possible for the original information to be recovered from the coded symbols.

Channel coding has a different purpose from source coding. Source coding tries to remove redundancy during data compression whereas channel coding adds redundancy to reduce errors at the receiver. During signal transmission, channel noise is present and the received signal is different from the transmitted signal. Hence there is a chance

that errors can be present in the received signal. In order to detect and/or correct these errors as best as possible, redundancy is added during channel coding. An example of channel coding is by repeating each of the symbols several times, e.g. a message containing 01001 can be coded as 000 111 000 000 111 using repetitive coding having a repetition factor of three. In this case, there are two possible *codewords*, namely 000 and 111, in the *codebook* used by the channel encoder. If the received message is 100 111 000 000 111, we know that the first group of three encoded bits has errors since a group of encoded bits can only consist of 000 or 111. Whenever a group of encoded bits does not consist of all ones or all zeroes, error correction is carried out using the majority-wins rule, i.e. a '0' is decoded if most of the encoded bits within a group are '0', and '1' otherwise, since it is more likely to have a lesser number of incorrect encoded bits within a group. In this case, the decoded message after error correction is 01001, which is still the same as the transmitted message. In general, error correcting techniques can be divided into two groups - linear block codes and convolutional codes - which will be described in more details in Sections 3.1.1 and 3.1.2.

The encoded data cannot be sent through a noisy communication channel without going through a process called digital modulation, which is a method that uses a finite number of, say M , distinct signals to represent encoded symbols. In other words, every encoded symbol is mapped to a different signal for transmission. Modulation allows sharing of a common medium by different users (multiple access), and makes the signal easy to propagate in a chosen transmission medium. Common digital modulation techniques are phase-shift keying (PSK), amplitude-shift keying (ASK) and frequency-shift keying (FSK). PSK, ASK and FSK convey data by modulating the phase, amplitude and frequency of the carrier signal respectively. Figure 2.4 shows examples of these modulation schemes in the form of a signal constellation marked by the M distinct signal points. The 'M' before the terms MPSK, MASK and MFSK gives the number of distinct signals (usually a power of 2) used to represent data. If 'M' is not a number but a 'B' or a 'Q', it stands for binary (i.e. $M = 2$) and quaternary (i.e. $M = 4$) respectively.

The received signal is then demodulated using a demodulator which processes the corrupted transmitted signal and converts them into a sequence of data symbols. If the transmission channel is known to be additive white Gaussian noise (AWGN), the optimum demodulator is the *matched-filter demodulators*. For example, in the case of

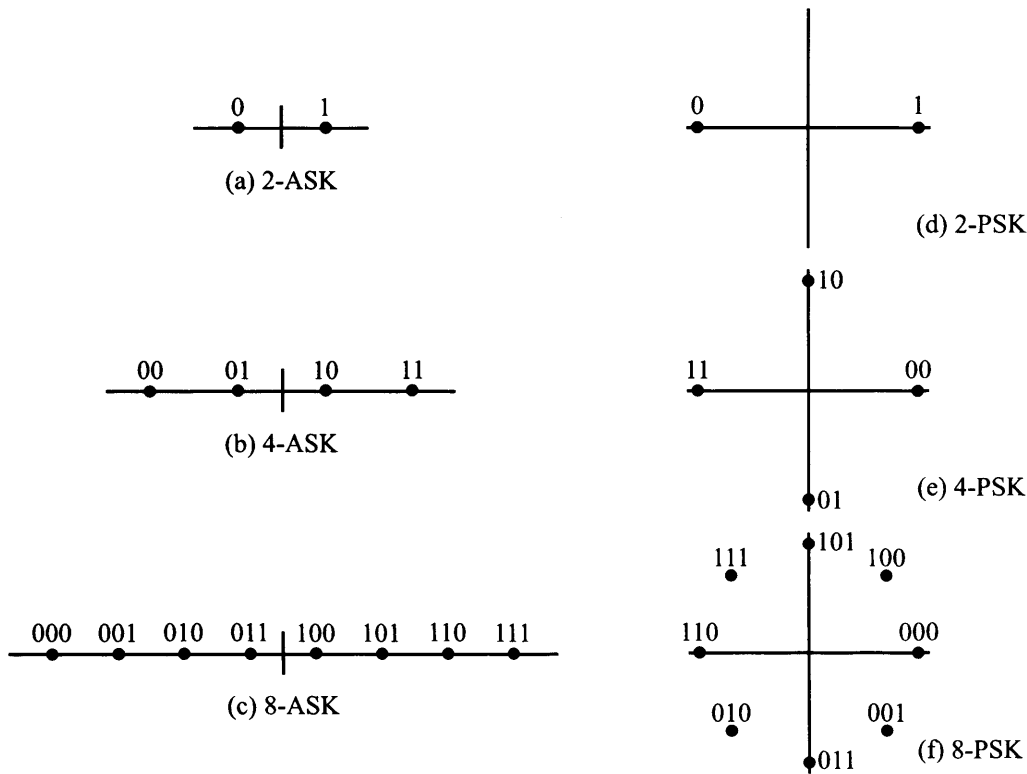


Figure 2.4: Different digital modulation schemes.

MPSK, the signal point closest, in terms of *Euclidean distance*, to the received signal will be the demodulated signal. The Euclidean distance, d , between two signal points \mathbf{x} and \mathbf{y} is defined as

$$d = |\mathbf{y} - \mathbf{x}| = \sqrt{\left(\sum_i (y_i - x_i)^2\right)} \quad (2.6)$$

where x_i and y_i are the coordinates of the signal points \mathbf{x} and \mathbf{y} respectively.

2.2 Spread Spectrum Communications

Spread spectrum, as the name suggests, is a technique to “spread” the original data over a much wider range of frequencies. The range of frequencies occupied by the transmitted signal is commonly known as the bandwidth. Spread spectrum communications was first developed for use in military communications and then several commercial applications such as CDMA (Code Division Multiple Access) mobile telephone networks adopted the technology. Because of the vast number of applications using spread spectrum, it has become one of the most important communication techniques currently

used.

The importance of spread spectrum in today's communication systems is because of its useful properties. In this section, the properties of spread spectrum communications are first discussed.

2.2.1 Properties of Spread Spectrum Communications

Spread Spectrum is a widely used technique in a lot of different communication applications because there are several favourable properties. Let us now highlight the properties of spread spectrum communications.

Anti-jamming. Jamming, in communications, means an intentional disruption of existing communication by sending another signal, usually of random nature, having substantial power. This jamming signal acts as an additional noise source on top of the channel noise that is already present in the transmission medium. The anti-jamming property of spread spectrum results from the fact that an attacker has no prior knowledge of the signal characteristics of the communication between the sender and the receiver except for the overall channel bandwidth and the type of modulation³ used. As a result, the attacker must jam the entire frequency spectrum used by the spread signal so as to be confident of preventing the communication between the sender and the receiver. However, the jammer has a limited power and is only able to jam certain frequencies, i.e. the jammer is not able to jam all frequencies all the time to prevent any communication between the sender and the receiver.

Low probability of interception. The low-probability-of-interception (LPI) property results from the very fact that the original signal intended for transmission has only a certain amount of power that is subsequently distributed over a much larger band of frequencies upon spreading such that only a very small amount of power is present at each frequency. The spread signal power at each frequency is often lower than the channel noise power such that the attacker may not even detect the presence of its transmission.

³The types of digital modulation are PSK, ASK and FSK. Refer to Section 2.1 for an elaboration on these modulation types.

Security. Spreading is carried out by superimposing a pseudo-random pattern, which is generated by a secret key known only to the sender and the intended receiver, of a transmitted message. An eavesdropper, who has no knowledge of the secret key, will not be able to decipher the communication and, as such, the privacy of the message is preserved during transmission.

2.2.2 Principles of Spread Spectrum Communications

The properties of spread spectrum suggest that communications can be carried out in a more secure manner by reducing the severity of an attack (anti-jamming), the probability of detecting a transmitted spread signal (LPI), and by increasing the security of the transmitted message. In order to fully understand these properties, we shall go into the technical details of spread spectrum.

Spread spectrum communications involves spreading a narrowband signal to a wideband signal and is carried out at the modulation stage in a communication system (See Figure 2.3). The two main types of spreading used in current communication systems are: *frequency hopping spread spectrum* (FHSS) and *direct sequence spread spectrum* (DSSS).

In frequency hopping spread spectrum (FHSS), there are many frequency slots available for transmission. Instead of transmitting the data at just one frequency slot as in the case without using FHSS, the data is transmitted at randomly selected frequency slots at various times and hence the name “frequency hopping”. For example, assume that there are 4 frequency slots (f_1, f_2, f_3, f_4) available for transmission as shown in Figure 2.5. Instead of transmitting a signal at frequency slot f_1 all the time, the transmitter chooses to transmit the signal at frequency slots f_2, f_4, f_2, f_3, f_1 at time slots t_1, t_2, t_3, t_4, t_5 respectively. In this case, an eavesdropper will not be able to catch the entire message if he does not know the range and order of frequencies used for transmission, which are only known to the transmitter and the intended receiver.

For direct sequence spread spectrum (DSSS), a large band of frequencies is available for use to transmit the original data from the sender to the intended receiver. Transmission takes place over the entire band of frequencies all the time as compared to a much smaller band of frequencies used by the data signal prior to spreading. Spreading (and despreading) is carried out at the sender (and receiver) using a pseudo-random

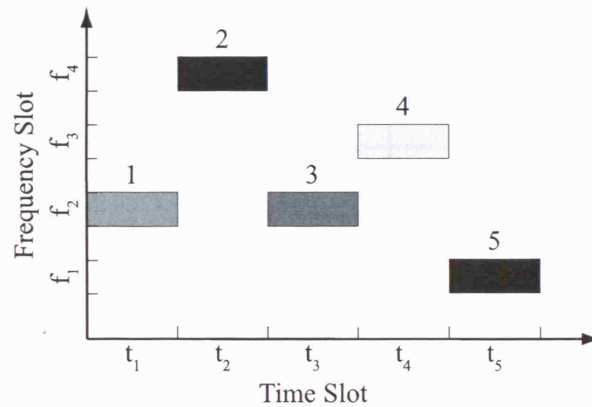


Figure 2.5: An example of frequency-hopping spread spectrum (FHSS).

(PN) signal generated by a secret key only known to the sender and receiver. To easily understand how direct sequence spread spectrum (DSSS) works, consider a BPSK modulation (i.e. bit '0' is mapped to '-1' and '+1' otherwise) for a pseudo-random (PN) signal, $s(t)$, and a data signal, $d(t)$, as shown Figure 2.6. The time duration for a pulse, called a *chip*, within the PN signal, $s(t)$, is called the *chip interval*, T_c , whereas the time duration for a bit within the data signal, $d(t)$, is the *bit interval*, T_b . In this case, 4 chip intervals make up one bit interval. The final spread signal is $c(t) = d(t) \times s(t)$. If additive white Gaussian noise (AWGN) is present during the channel, then the received signal is $r(t) = c(t) + n(t)$, where $n(t)$ is the Gaussian noise signal.

If the signal is transmitted in a wireless medium, the propagation delay from the transmitter and the receiver is generally unknown to the receiver. In order to correct this problem, the symbol timing has to be accurately estimated. Furthermore, the signal does not necessarily travel in a straight path from the transmitter to the receiver. More often than not, the signal takes various paths to reach the receiver because of the terrain along the transmission. Therefore the receiver may receive multiple copies of the signal at various times with varying power and this makes decoding especially difficult when the transmission is a stream of continuous data. This can cause serious timing and phase problems during decoding. Hence synchronisation is often required in every digital communication system which transmits information synchronously. Many different synchronisation methods have been established and interested readers are referred to Chapter 6 of [Pro01].

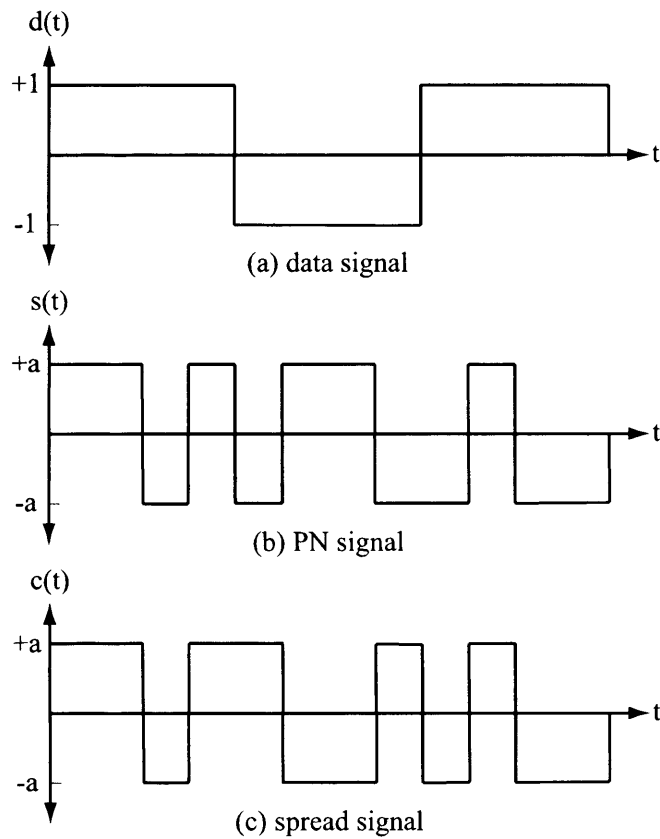


Figure 2.6: An example of direct sequence spread spectrum (DSSS).

Assuming perfect synchronisation, despreading is carried out at the receiver whereby the original information is obtained by multiplying the received signal, $r(t)$, by the same PN signal used during spreading, $s(t)$, to get a corrupted data signal

$$d'(t) = r(t) \times s(t) \quad (2.7)$$

$$= [d(t) \times s(t) + n(t)] \times s(t) \quad (2.8)$$

$$= d(t) + n(t) \times s(t) \quad (2.9)$$

where the auto-correlation of $s(t)$, i.e. $s(t) \times s(t)$, equals 1, and can be easily observed from Figure 2.6. Therefore the original data signal, $d(t)$, can be seen as corrupted by a noise source $n(t) \times s(t)$. Since the PN signal, $s(t)$, is a pseudorandom signal, $n(t)$ and $s(t)$ are likely to be orthogonal to each other, i.e. $n(t) \times s(t)$ is approximately zero. Hence by using direct sequence spread spectrum (DSSS), the effect of the noise signal, $n(t)$, is small.

2.3 Watermarking as a Communication System

Watermarking is a form of communication since a message is transmitted from the watermark embedder to the watermark detector. Therefore it is natural to fit watermarking into the general models of communication discussed in Section 2.1.

The two basic types of watermarking systems are shown in Figures 2.7 and 2.8. Further types of watermarking systems are discussed in Section 2.6. Both Figures 2.7 and 2.8 differ only at the watermark detection stage. The original cover Work, c_o , is available to the watermark detector in Figure 2.7 but not in Figure 2.8. We refer to the detector in Figure 2.7 as a *informed detector* and that in Figure 2.8 as *blind detector*. In both cases, the watermark embedder is *blind* since the cover Work, c_o , takes no part in the watermark encoding process and is simply added to the *added pattern*, w_a , which is of the same size as the cover Work, c_o .

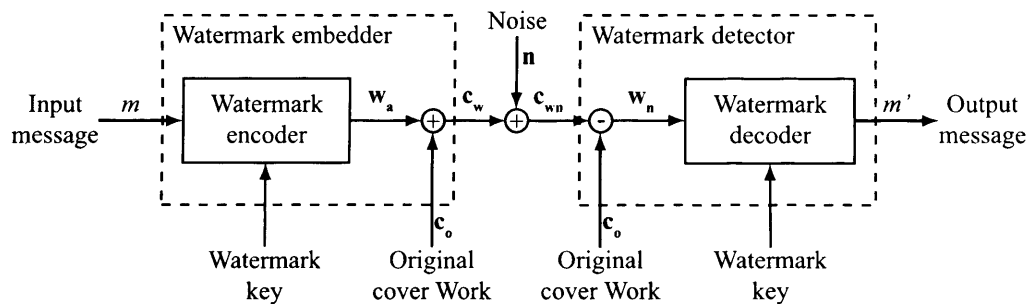


Figure 2.7: Watermarking system with informed detection.

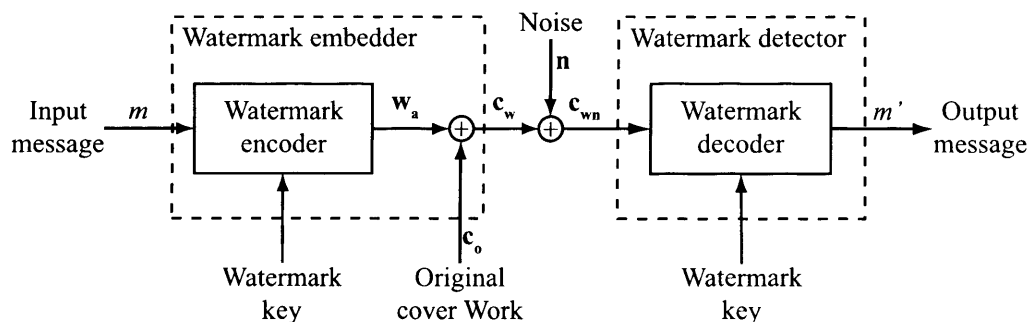


Figure 2.8: Watermarking system with blind detection.

Regardless of whether the watermarking system deploys an informed detector or a blind detector, the watermark embedding consists of two basic steps. First, the input

message, m , is mapped to an *added pattern*, w_a . This mapping is usually done with a watermark key to enhance the security of the system. There are many ways to carry out this mapping, depending on what is the criterion for the added pattern, w_a . Specific mapping rules will be described later in this chapter. Secondly, the added pattern, w_a , is added to the cover Work, c_o , to produce the watermarked Work, c_w . Note that in general, the watermark embedder produces only the watermarked Work, c_w , and the added pattern, w_a , can be obtained indirectly by subtracting the cover Work, c_o , from the watermarked Work, c_w .

The watermarked Work, c_w , is normally subjected to some degradation or distortion, either by adversarial processes or by normal signal processes, when it is transmitted through a communication channel. Examples of normal signal processing are compression and decompression, image enhancement, and broadcasting over air. Although some of these types of processes are dependent on the watermarked Work, c_w , the channel noise is usually modelled as additive noise, n .

The watermark detector now receives c_{wn} , which is the noisy version of the watermarked Work, c_w . Watermark detection can either be informed or blind as illustrated in Figures 2.7 and 2.8 respectively. In the case of informed detection (Figure 2.7), the original cover Work, c_o , is subtracted from the received Work, c_{wn} , at the watermark detector to give the noisy watermark pattern, w_n . This is then decoded by the watermark detector with the same secret key used by the watermark embedder. Since the cover Work, c_o , is added at the watermark embedder and then subtracted at the watermark detector, the cover Work, c_o , has no effect on the system and the result is that the noisy watermark pattern, w_n , is simply an addition of noise, n , to the added pattern, w_a . Therefore the watermark encoder, the noise process, and the watermark decoder form a system analogous to the communication system shown in Figure 2.1.

On the other hand, in a blind-detection watermarking system (Figure 2.8), the unwatermarked cover Work, c_o , is unknown to the watermark detector and hence is unable to be removed prior to watermark decoding. Nevertheless, this watermarking system is still analogous to the communication system in Figure 2.1, where the watermark encoder is viewed as the transmitter, the combination of the original cover Work, c_o , and the noise signal, n , is considered as a combined noise source, and the watermark detector is regarded as the receiver. In this scenario, the corrupted watermarked Work,

c_{wn} , is a noisy version of the added pattern, w_a .

2.4 Spread Spectrum Watermarking

In Section 2.2, we saw how spread spectrum communications enables a message to be spread over a larger bandwidth and also how the original message can be recovered from the spread signal with the help of a secret key known only to the sender and the receiver. In order to achieve robustness against malicious attacks, low perceptibility of the embedded watermark and security of the message, the concept of spread spectrum is applied to watermarking systems [CKLS97]. In this section, an example of watermarking based on direct sequence spread spectrum (DSSS) is illustrated.

Let us consider a simple additive spread spectrum-based (SS) watermarking system with the secret key, K , available to both the sender and the receiver, as shown in Figure 2.9. This system uses the secret key, K , to select a distinct watermark pattern, w_m , from a set of possible patterns generated by the pseudorandom number generator (PRN). The mapping between the secret key, K , and the watermark pattern, w_m , is a one-to-one, i.e. one secret key can only choose one distinct watermark pattern. This watermark pattern, w_m , is then multiplied by a BPSK-modulated message bit, b , (i.e. bit '0' is modulated to '-1', and '+1' otherwise), and the result is added to the cover Work, c_o , to form the watermarked Work,

$$c_w = c_o + bw_m. \quad (2.10)$$

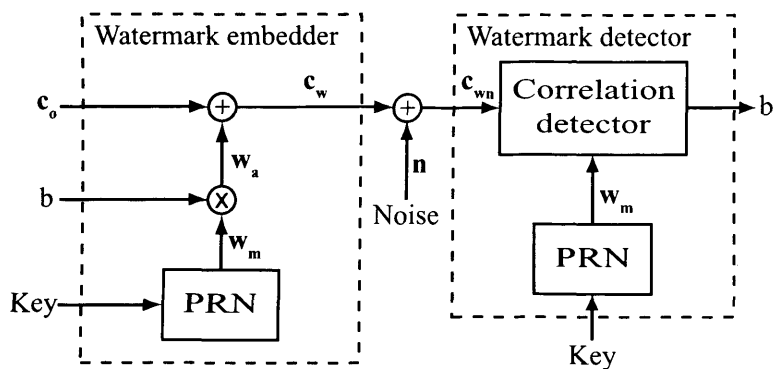


Figure 2.9: An example of a spread spectrum-based watermarking.

During the transmission process, the watermarked Work, c_w , is corrupted by channel and attack noise, n , such that the watermark detector (in the form of a correlation

detector) receives the corrupted watermarked Work, $\mathbf{c}_{wn} = \mathbf{c}_w + \mathbf{n}$. The correlation detector uses the same watermark pattern, \mathbf{w}_m , generated using the same secret key, K , to compute a correlation statistic, r , from the corrupted watermarked Work, \mathbf{c}_{wn} . This correlation statistic is defined as

$$r = \frac{\mathbf{c}_{wn} \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m}, \quad (2.11)$$

between the corrupted watermarked Work, \mathbf{c}_{wn} , and the watermark pattern, \mathbf{w}_m . The symbol “ \cdot ” is defined as the inner product (or simply correlation) operator, i.e. $\mathbf{x} \cdot \mathbf{y} = \sum_i (x_i \times y_i)$. Assume that the cover Work, \mathbf{c}_o , the watermark pattern, \mathbf{w}_m , and the noise, \mathbf{n} , consist of elements c , w , and n from Gaussian random processes such that

$$c \sim \mathcal{N}(0, \sigma_c^2), w \sim \mathcal{N}(0, \sigma_w^2), n \sim \mathcal{N}(0, \sigma_n^2). \quad (2.12)$$

By substituting Equation 2.10 into Equation 2.11 and recognising $\mathbf{w}_m \cdot \mathbf{w}_m = N\sigma_w^2$, where N is the length of each of the vectors, we get

$$r = \frac{(\mathbf{c}_w + \mathbf{n}) \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.13)$$

$$= \frac{(\mathbf{c}_o + b\mathbf{w}_m + \mathbf{n}) \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.14)$$

$$= \frac{\mathbf{c}_o \cdot \mathbf{w}_m + b\mathbf{w}_m \cdot \mathbf{w}_m + \mathbf{n} \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.15)$$

$$= b + \frac{\mathbf{c}_o \cdot \mathbf{w}_m}{N\sigma_w^2} + \frac{\mathbf{n} \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.16)$$

The correlation detector produces an estimate of the sent bit as

$$b' = \text{sign}(r) = \begin{cases} -1, & \text{if } r < 0 \\ +1, & \text{otherwise.} \end{cases} \quad (2.17)$$

The correlation statistic, r , can be easily shown to be Gaussian with mean \bar{r} and variance σ_r^2 as given below.

$$\bar{r} = b, \sigma_r^2 = \left(\frac{\sigma_c^2 + \sigma_n^2}{N\sigma_w^2} \right) \quad (2.18)$$

2.5 Communications with Side Information at the Transmitter

A simplified communication system model, shown in Figure 2.1, consists of a single additive noise source, and the channel capacity of this channel is given in Equation 2.5

as $C = \frac{1}{2} \log_2 \left(1 + \frac{\sigma_x^2}{\sigma_n^2} \right)$ bits/transmission. Let us consider a communication system having two independent, additive noise sources, \mathbf{n}_1 and \mathbf{n}_2 , with mean zero and variances $\sigma_{n_1}^2$ and $\sigma_{n_2}^2$ as shown in Figure 2.10. In this situation, the channel capacity for this communication model is given as

$$C = \frac{1}{2} \log_2 \left(1 + \frac{\sigma_x^2}{\sigma_{n_1}^2 + \sigma_{n_2}^2} \right) \text{ bits/transmission.} \quad (2.19)$$

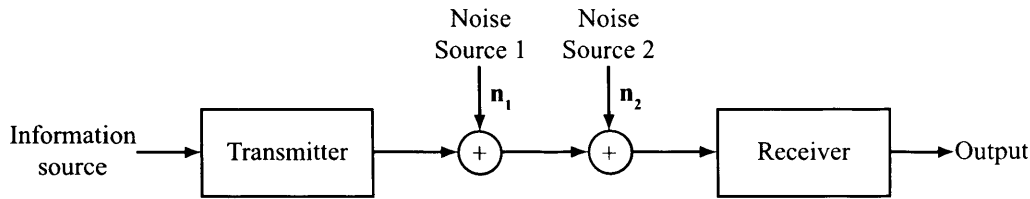


Figure 2.10: A communication model with two independent, additive noise sources.

In 1958, Shannon [Sha58] considered the scenario where the first noise source, \mathbf{n}_1 , is known to the transmitter. In this case, the first noise source, \mathbf{n}_1 , is considered to be *side information*, which contains the state of the transmission channel that can be used to aid the coding and transmission of information. Hence a communication system that is based on this scenario is called communications with side information.

Costa [Cos83] further investigated this concept and considered a specific case illustrated in Figure 2.11. The side information, \mathbf{n}_1 , in this case is a sequence of independent and identically distributed (i.i.d.)⁴ $\mathcal{N}(0, \sigma_{n_1}^2)$ Gaussian random variables while the noise source \mathbf{n}_2 is another sequence of i.i.d. $\mathcal{N}(0, \sigma_{n_2}^2)$ Gaussian random variables, where $\sigma_{n_1}^2$ and $\sigma_{n_2}^2$ are their signal powers respectively. The transmitted signal, \mathbf{x} , is a function of the input message, m , and the side information, \mathbf{n}_1 , i.e. the input message, m , and the side information, \mathbf{n}_1 , determines what will be transmitted through the channel. Assume that there is a power constraint on the transmitted signal, \mathbf{x} , such that

$$\frac{1}{N} \sum_{i=1}^N x_i^2 \leq P_{av}, \quad (2.20)$$

where N is the length of the signal \mathbf{x} .

⁴In probability theory, a sequence of random variables are said to be independent and identically distributed (i.i.d.) if every random variable has the same probability distribution and each of them is independent of one another.

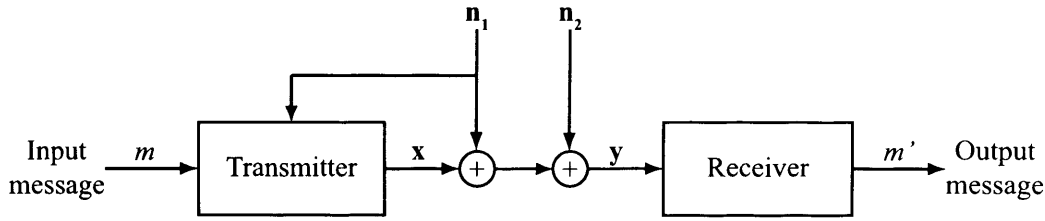


Figure 2.11: Communications with side information at the transmitter.

If the transmitter (and not the receiver) has the first noise source, \mathbf{n}_1 , as side information, the channel capacity of such a communication system, as investigated by Costa, is shown to be $C = \frac{1}{2} \log_2(1 + \frac{\sigma_x^2}{\sigma_{n_2}^2})$ bits/transmission. This means that the first noise source, \mathbf{n}_1 , has no effect on the channel capacity, even though \mathbf{n}_1 is not known to the receiver. Note that the channel capacity cannot exceed that given in Equation 2.5 since the second noise source, \mathbf{n}_2 , is still present to limit the maximum allowable capacity as in the case of Figure 2.1.

In order to achieve this result, Costa assumed that the encoder contains a codebook of N_u distinct sequences \mathbf{u} and that there are only N_m different messages m that can be generated, where $N_u \gg N_m$. The N_u sequences are distributed uniformly over N_m messages such that a message can be encoded by N_u/N_m different sequences. By considering $\mathbf{u} = \mathbf{x} + \alpha \mathbf{n}_1$, and maximising the capacity of this channel, Costa obtained $\alpha^* = \sigma_x^2 / (\sigma_x^2 + \sigma_{n_2}^2)$ as the optimum α . According to Costa [Cos83], the encoder (at the transmitter) looks for a sequence \mathbf{u} such that

$$|(\mathbf{u} - \alpha^* \mathbf{n}_1)^T \mathbf{n}_1| \leq \delta \quad (2.21)$$

for some approximately small δ , where the superscript T indicates a matrix transpose. In other words, the encoder searches for a sequence $(\mathbf{u} - \alpha^* \mathbf{n}_1)$, or simply \mathbf{x} , which is nearly orthogonal to \mathbf{n}_1 . However, there is no guarantee that a sequence \mathbf{u} can be found to satisfy Equation 2.21 unless N_u and N_m are very large. Although Costa provided a solution to build a random codebook of N_u distinct sequences \mathbf{u} , there was no proposed method to efficiently search for the sequence \mathbf{u} that satisfies Equation 2.21, especially if the N_u , number of distinct sequences \mathbf{u} , and N_m , the number of different messages m , are not very large.

2.6 Watermarking as Communications with Side Information

As mentioned earlier, watermarking can be viewed as a communication system (see Figures 2.7 and 2.8). Let us consider a very simple, practical watermarking system that employs a simple scaling of the watermark pattern, w_m , shown in Figure 2.12. The secret message, m , to be hidden is first encoded (at the message encoding block) as a watermark pattern, w_m , without the knowledge of the cover Work, c_o . In this case, the message encoding block provides a one-to-one mapping between a message, m , and a watermark pattern, w_m , i.e. one message will only be mapped to a specific watermark pattern. This watermark pattern, w_m , is modified by simple scaling (at the scaling block) to obtain an added watermark

$$w_a = \alpha w_m, \quad (2.22)$$

which is then added to the cover Work, c_o , to produce a watermarked Work

$$c_w = c_o + w_a, \quad (2.23)$$

where α (α is always positive) is the scaling factor used during watermark embedding. In this scenario, the watermark embedder is *blind* since no knowledge on the cover Work, c_o , is available during the message coding stage (*blind coding*) and the scaling stage (*blind embedding*).

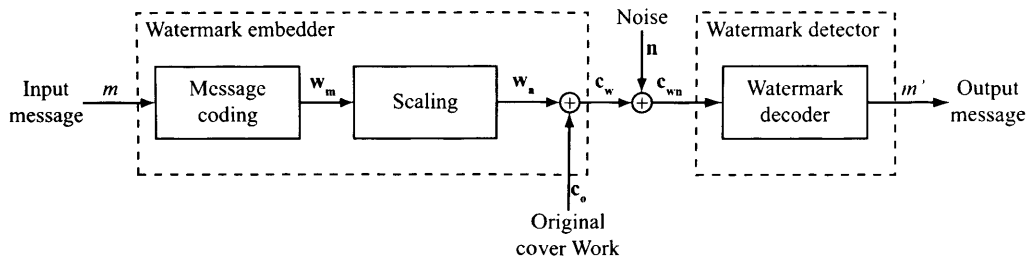


Figure 2.12: Watermarking with blind embedders.

Although simply scaling the watermark is easy to implement, this method will result in a much reduced capacity since the cover Work, c_o , acts as a noise source (see Equation 2.19). To increase the capacity of a watermarking system, note that Costa's dirty paper scheme, mentioned in Section 2.5, is almost identical to watermarking with

blind detection as shown in Figure 2.8. The side information plays the role of the cover Work, c_o , whereas the second noise source plays the role of distortions that occur during normal processing or malicious attack. The power constraint, P_{av} , expresses the fidelity requirement of a watermarking application. The only alteration to Figure 2.8 is to make sure that the watermark encoder has access to the cover Work, c_o so that the resulting watermarking system is shown in Figure 2.13, with the watermark encoder in Figure 2.12 replaced by a message coding block and a modification block in Figure 2.13. Several researchers have recognised that watermarking can be modelled as communications as side information at the transmitter [CW98, CMM99].

If a watermarking system behaves exactly as the system modelled by Costa, then the cover Work, c_o , will not act as a noise source and affect the capacity of the watermarking system. However, there are substantial differences between Costa's communications model and a real watermarking system. In watermarking, the distributions of the two noise sources — the cover Work, c_o , and the distortion n — are rarely Gaussian. When watermarking in the Discrete Cosine Transform (DCT) domain, the DCT coefficients may be approximately Gaussian [Pra78] although they resemble more towards Laplacian distributions [RG83]. Furthermore, the power constraint, measured in terms of the mean squared error metric, is known to be a poor indication of perceptual distortions. In an attempt to relax the restriction on the distributions of the two noise sources, recent research by Cohen and Lapidath [CL02] and Erez *et al* [ESZ00] proved that Costa's dirty paper scheme can be extended to specific cases where the second noise source, n , (in Figure 2.11) comes from other arbitrary distributions. Although watermarking may not be exactly the same as Costa's scheme, it can still greatly benefit from Costa's result by identifying the cover Work, c_o , as side information available at the watermark embedder.

Referring to Figure 2.13, the secret message, m , is first mapped to several different watermark patterns during the message coding stage (*informed coding*), i.e. a one-to-many mapping. Out of the several watermark patterns, the final watermark pattern, w_m , is chosen using the cover Work, c_o . The watermark pattern, w_m , is then modified during the modification stage according to some embedding criteria such as a specified perceptual distance or robustness, also with knowledge of the cover Work, c_o , to produce the added watermark, w_a . In this case, the transformation between the water-

mark pattern, w_m , to the added watermark, w_a , is known as *informed embedding*. The watermarked Work, c_w , is produced according to Equation 2.23 and is subsequently corrupted by additive noise, n . The hidden message is then determined by the watermark detector from the corrupted watermark Work, c_{wn} .

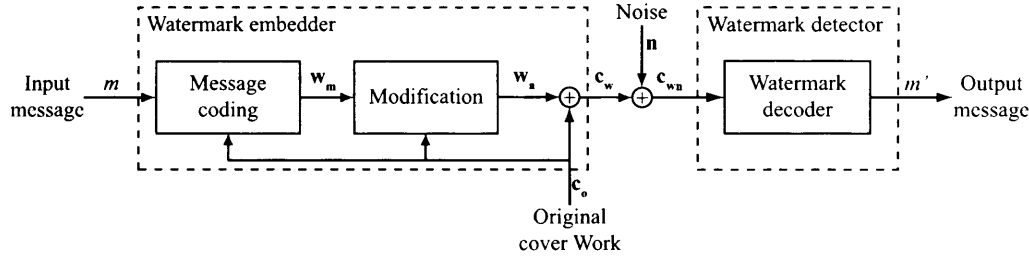


Figure 2.13: Watermarking as communication with side information at the transmitter.

2.7 Improved Spread Spectrum

From Section 2.6, we have seen that by modelling watermarking as communications with side information at the transmitter, the effect of the cover Work, c_o , can be significantly reduced. Since simple spread spectrum (SS) watermarking, illustrated in Figure 2.9, does not utilise knowledge of the cover Work, c_o , at the embedder, its channel capacity is expected to be much lower.

To improve the performance of simple SS watermarking, Malvar and Florencio [MF03] proposed a new scheme called *Improved Spread Spectrum* (ISS). Instead of using Equation 2.10 during watermark embedding, the new embedding approach is slightly modified such that the knowledge of the cover Work, c_o , is taken into account at the watermark embedder, i.e.

$$c_w = c_o + \mu(b, c_o)w_m, \quad (2.24)$$

where $\mu(b, c_o)$ is a general function that has input variables b and c_o . Similarly in this case, the elements of those vectors (i.e. c_o , w_m , n) are from Gaussian random processes according to Equation 2.12. Note that the simple spread spectrum watermarking scheme (of Figure 2.9) is a special case of ISS if the function $\mu(b, c_o)$ is made independent of the cover Work, c_o .

In order to illustrate how ISS works, Malvar and Florencio [MF03] gave an example, illustrated in Figure 2.14, which uses a linear function for $\mu(b, c_o)$ such that the

watermarked Work is

$$\mathbf{c}_w = \mathbf{c}_o + \left(\alpha b - \frac{\lambda \mathbf{c}_o \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m} \right) \mathbf{w}_m. \quad (2.25)$$

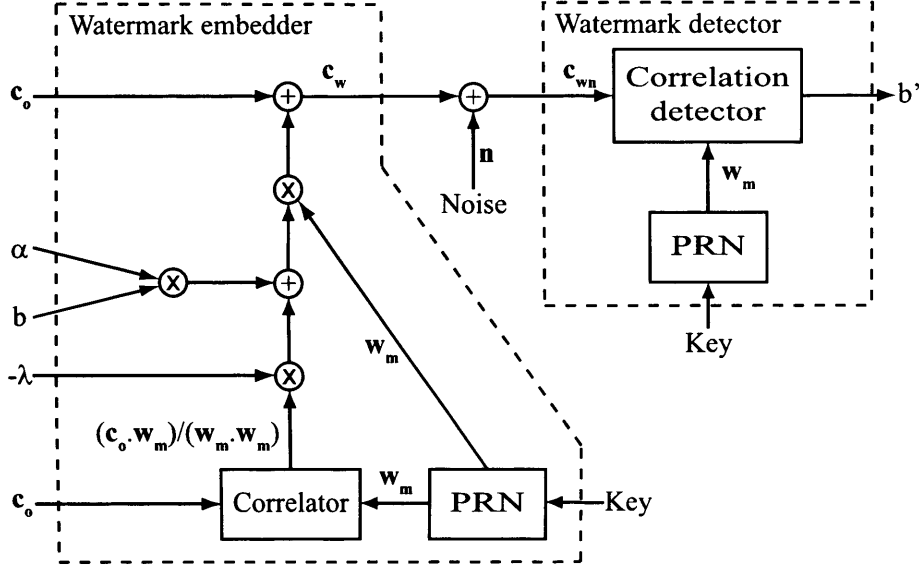


Figure 2.14: Improved spread spectrum (ISS) scheme.

The parameters α and λ determine the strength of embedding and the degree of feedback by the cover Work, \mathbf{c}_o , respectively. The simple spread spectrum watermarking system is obtained by setting $\alpha = 1$ and $\lambda = 0$.

Using Equation 2.25, the correlation statistic, r , computed by the correlation detector in this case is given as

$$r = \frac{(\mathbf{c}_w + \mathbf{n}) \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m} \quad (2.26)$$

$$= \frac{\left(\mathbf{c}_o + \left(\alpha b - \frac{\lambda \mathbf{c}_o \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m} \right) \mathbf{w}_m + \mathbf{n} \right) \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.27)$$

$$= \frac{\mathbf{c}_o \cdot \mathbf{w}_m + \alpha b \mathbf{w}_m \cdot \mathbf{w}_m - \frac{\lambda \mathbf{c}_o \cdot \mathbf{w}_m}{\mathbf{w}_m \cdot \mathbf{w}_m} \mathbf{w}_m \cdot \mathbf{w}_m + \mathbf{n} \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.28)$$

$$= \frac{\alpha b N\sigma_w^2 + (1 - \lambda) \mathbf{c}_o \cdot \mathbf{w}_m + \mathbf{n} \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.29)$$

$$= \alpha b + \frac{(1 - \lambda) \mathbf{c}_o \cdot \mathbf{w}_m}{N\sigma_w^2} + \frac{\mathbf{n} \cdot \mathbf{w}_m}{N\sigma_w^2} \quad (2.30)$$

The detector makes the same decoding decision as that of the simple SS watermarking, i.e. using Equation 2.17. From Equation 2.30, the influence of $\frac{\mathbf{c}_o \cdot \mathbf{w}_m}{\sigma_w^2}$ on

the correlation statistic, r , is reduced if the value of λ is close to 1. In this case, the probability of error is minimised when λ takes a value close to, but not equal to, 1.

Similar to the case of SS watermarking, this correlation statistic, r , can be shown to be Gaussian with mean \bar{r} and variance σ_r^2 where N is the length of all the vectors.

$$\bar{r} = \alpha b, \sigma_r^2 = \frac{(1 - \lambda)^2 \sigma_c^2 + \sigma_n^2}{N \sigma_w^2} \quad (2.31)$$

The probability of error is defined as $P(r < 0 | b = +1)$, i.e. probability of receiving $r < 0$ given that the sent bit is '1', or $P(r > 0 | b = -1)$. By comparing Equations 2.18 and 2.31 and noting that both means \bar{r} are Gaussian, it is obvious that ISS can achieve a lower probability of error compared to the SS watermarking scheme (under the condition that σ_c^2 , σ_n^2 and σ_w^2 are the same) since the former is able to raise \bar{r} and reduce σ_r^2 by increasing α and decrease λ respectively.

However, the value of α cannot be too large since the distortion (in terms of mean square error), which is defined as

$$D_{mse}(\mathbf{c}_o, \mathbf{c}_w) = |\mathbf{c}_w - \mathbf{c}_o| = \sqrt{(\mathbf{c}_w - \mathbf{c}_o) \cdot (\mathbf{c}_w - \mathbf{c}_o)}, \quad (2.32)$$

to the cover Work, \mathbf{c}_o , will be too large and therefore unacceptable.

2.8 Practical Dirty Paper Coding

So far, we have seen how watermarking is modelled as a general communication system, with special focus on two important communications concepts: spread spectrum communications and communications with side information at the transmitter. In particular, because of the potential benefits the latter can bring, several researchers [CW99, CPR99, CMM99] have proposed watermarking systems based on Costa's dirty-paper scheme as depicted in Figure 2.13. Watermarking methods that are based on Costa's dirty-paper scheme are also known as dirty-paper codes.

In this section, three practical dirty-paper codes - lattice codes, syndrome codes, and dirty paper spherical codes - that use Costa's dirty-paper scheme are described and some comparisons are made among them. Firstly, lattice codes, as the name suggests, use a lattice structure (in a multi-dimensional space) such that each point in the lattice represents a unique watermarked Work, \mathbf{c}_w . The most popular method is Quantisation Index Modulation (QIM), by Chen and Wornell [CW99], whereby the point in the

lattice closest to the cover Work, c_o , is assigned to be the watermarked Work, c_w . More details of lattice codes are found in Section 2.8.1.

Another type of dirty-paper code proposed by Chou and Ramchandran [CPR99] is syndrome coding where the watermark embedder incorporates a source encoder having the knowledge of the cover Work, c_o , as side information. Many examples of syndrome codes are based on trellises constructed using channel codes such as convolutional codes. A more detailed description of syndrome codes is given in Section 2.8.2.

As an alternative to lattice codes, dirty paper spherical codes are designed such that all the watermark patterns lie on the surface of a multi-dimensional sphere. As such, every watermark pattern has the same energy. An example of dirty paper spherical codes is dirty paper trellis codes (DPTC) proposed by Miller *et al* [MDC02]. Dirty paper trellis codes (DPTC) use a trellis structure and a unique watermark pattern corresponds to a specific path through the trellis. Dirty paper spherical codes are described in more detail in Section 2.8.3.

2.8.1 Lattice Codes

A simple design of a dirty-paper code is a regular lattice structure that spans a real vector space. Each point in the lattice structure corresponds to a watermarked Work and rules are designed to map a message to a watermarked Work using the knowledge of the cover Work. Watermarking techniques that are based on such a technique are collectively called *lattice codes*.

One class of lattice codes, quantisation index modulation (QIM), proposed by Chen and Wornell [CW98] is a simple and low complexity method to watermark multimedia content. QIM is basically a quantisation process that maps the input (i.e. cover Work), that can be anywhere in the vector space, to the output (i.e. watermarked Work) that can only be from the set of points present in the lattice structure. In this case, we say that the cover Work is *quantised*.

For QIM, the set of points in the lattice structure are divided into $|\mathcal{M}|$ subsets, where \mathcal{M} is the set of possible messages. In the simplest case, let us assume that there are only 2 messages, '0' and '1' (i.e. $|\mathcal{M}| = 2$), and the lattice structure is just a one-dimensional line as shown in Figure 2.15. The set of points in this particular

lattice is divided into two subsets, one marked with crosses and another marked with circles. The points having circles correspond to message '0' and the points showing crosses correspond to message '1'. The distance between a pair of adjacent points within the same subset is called the *step size* Δ . Just to note that this is a form of uniform quantisation since the step size is uniform throughout. Assume that the cover Work, c_o , is denoted by a black square, the embedding function is defined as

$$c_w = q(c_o; m, \Delta) \quad (2.33)$$

where $q(c_o; m, \Delta)$ is the m -th quantiser, i.e. the cover Work, c_o , is quantised to the closest point belonging to the subset that corresponds to the message. Referring back to Figure 2.15, the cover Work, c_o , is quantised to the closest point in the chosen subset, i.e. the right circle if a message '0' is to be embedded or to the left cross if a message '1' is to be embedded. Note that QIM is a form of dirty-paper code since a message can be mapped to any of the alternative points in the corresponding subset.

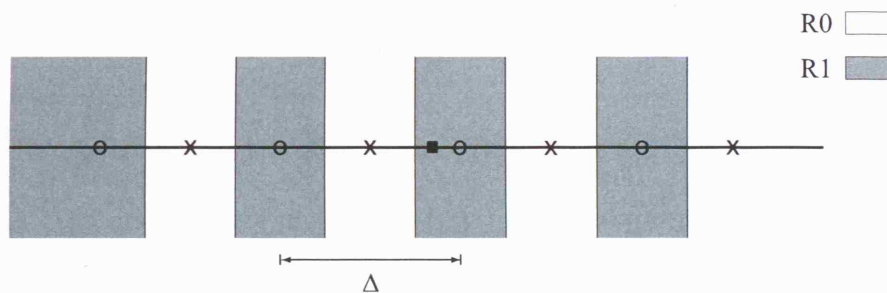


Figure 2.15: A simple example of quantisation index modulation (QIM)

Quantisation will inevitably cause distortion to the original content, i.e. the cover Work c_o . This distortion is sometimes known as quantisation noise (in this case, it is $c_o - c_w$) and it depends solely on the step size, Δ , i.e. a smaller step size will lead to a smaller distortion. However, a smaller step size can also lead to a higher chance of an error, i.e. the decoded point lies in a subset that does not correspond to the message, if the watermarked Work, c_w , is subjected to noise.

Although standard QIM is a dirty paper code, its performance is still far from Costa's results. In order to approach Costa's performance, Chen and Wornell [CW01] proposed distortion compensation as a type of post-quantisation process. This can be

achieved by using the embedding function

$$\mathbf{c}_w = q(\mathbf{c}_o; m, \Delta/\alpha) + (1 - \alpha)[\mathbf{c}_o - q(\mathbf{c}_o; m, \Delta/\alpha)] \quad (2.34)$$

where α is a scaling factor and $q(\mathbf{c}_o; m, \Delta/\alpha)$ is the m -th quantiser whose step size has been scaled from Δ by a factor of α so that the distance between a pair of adjacent points within the same subset is now Δ/α . If $\alpha < 1$, the distortion will be increased, thereby increasing the robustness at the same time. However, $(1 - \alpha)$ of the quantisation noise, $[\mathbf{c}_o - q(\mathbf{c}_o; m, \Delta/\alpha)]$, is added back to the quantised value, $q(\mathbf{c}_o; m, \Delta/\alpha)$, to compensate for the increase in distortion. The first term in Equation 2.34 is the normal QIM embedding whereas the second term is called the distortion-compensation term. This type of embedding scheme is called distortion-compensation QIM (DC-QIM).

There are other variants of QIM such as dither modulation (DM) and spread transform dither modulation (ST-DM). For dither modulation (DM), the quantisers used are called *dithered quantisers*, which are shifted versions of the original quantisers. The shift arises from the use of the *dither vectors*, $\mathbf{d}(m)$, which are pseudorandom vectors mapped uniquely from a given message, m . The embedding function for DM is thus defined as

$$\mathbf{c}_w = q(\mathbf{c}_o + \mathbf{d}(m); m, \Delta/\alpha) - \mathbf{d}(m) \quad (2.35)$$

An extension to dither modulation (DM) is spread transform dither modulation (ST-DM), which uses an additional random spreading vector, \mathbf{v} , during its quantisation process, instead of just the usual quantisation of the cover Work, \mathbf{c}_o . The ST-DM scheme quantises the projection of the cover Work, \mathbf{c}_o , onto the random spreading vector, \mathbf{v} , using the dithered quantisers.

Although QIM and all its variants mentioned so far (DM, DC-QIM and ST-DM) are computationally simple to implement, they are very sensitive to amplitude scaling, i.e. a multiplication of the volume or brightness of a multimedia content by a factor. Even small changes in volume or brightness can lead to a complete loss of the embedded message. Several methods have been proposed to deal with this sensitivity. These methods can be categorised as either: (i) introducing a pilot signal to determine the scaling [EBG02], (ii) directly estimating the scaling [LKKM03], (iii) adaptive quantisation in which the quantisation step size is automatically adjusted based on the scaling [OKS04, LC07].

2.8.2 Syndrome Codes

Syndrome codes, proposed by Chou *et al* [CPR99], are related to distributed source coding [PR99], as illustrated in Figure 2.16. In distributed source coding, the source X is encoded and then transmitted to the decoder which has access to side information Y . Even though, in this case, the encoder has no access to the side information Y , the availability of the joint statistics of X and Y enables such a system performs as well as the case when both the encoder and decoder have side information Y . This result is known as the Slepian-Wolf theorem [SW73].

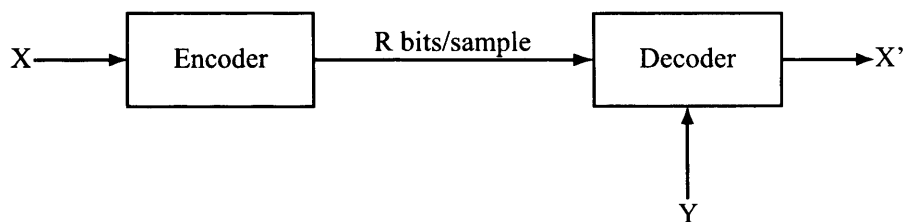


Figure 2.16: Distributed source coding: Source coding with side information at the decoder.

In the watermarking context, the side information c_o is available to the encoder but not the decoder as shown in Figure 2.17. The encoder encodes the message m , to be embedded, with the knowledge of the side information, i.e. the cover Work, c_o , and the output of the encoder is a watermarked Work c_w . An adversary will try to corrupt the watermarked Work, c_w , so that the decoder will fail to detect the watermark. However, the adversary has a limit on the amount of distortion he can inflict on the watermarked Work c_w . Hence the encoder is to be designed such that it can withstand the allowable degradation on the watermarked Work c_w .

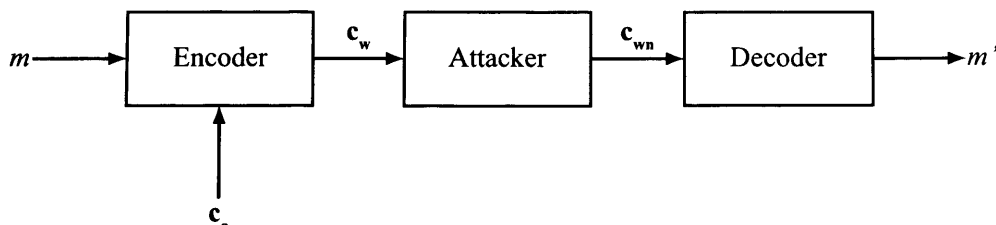


Figure 2.17: A watermarking problem

The watermarking problem in Figure 2.17 can be viewed as the problem of chan-

nel coding with side information. Because of the similarities between distributed source coding (source coding with side information at the decoder (SCSI)) and watermarking (channel coding with side information at the encoder (CCSI)), the two cases are considered duals of each other. Pradhan *et al* [PCR03] explored the duality between the former and the latter and provided a mathematical characterisation of the conditions under which their functions can be exchangeable.

Using the duality between distributed source coding and watermarking, Chou *et al* [CPR99, CPGR00] proposed a framework to use the concept of distributed source coding on watermarking problems. This framework can be carried out in 3 main steps: (1) build a channel code over the space of U , which is a set of all possible codewords (i.e. codebook) used by the encoder, (2) partition this channel codeword space into cosets of source codes, and (3) choose a codeword, w_m , to represent the side information, c_o , from the coset which corresponds to the message m . In [CPR99], Chou *et al* used trellis-coded modulation (TCM) [Ung82] as channel codes and source codes such as trellis-coded quantisation (TCQ) [MF90] to partition the channel codes. Refer to the respective references for more information on TCM and TCQ.

Although syndrome codes are robust to additive white Gaussian noise (AWGN), they are still susceptible to amplitude scaling since quantisation, in the form of trellis-coded quantisation (TCQ), is used in one of the watermark encoding steps.

2.8.3 Dirty Paper Spherical Codes

Because of the vulnerability of lattice codes and syndrome codes to amplitude scaling, researchers have considered spherical codes as an alternative solution. Spherical codes are a class of codes with codewords distributed, usually uniformly, on the surface of a high-dimensional sphere. Even if a scaling factor alters the codewords, the altered codewords will still be distributed on the surface of another sphere (of the same dimension) with a larger (or smaller) radius. Hence scaling will not affect the correct detection of codewords if watermark detection relies on angle-based detection statistic.

One method that is based on spherical codes is Angle QIM (AQIM) proposed by Ourique *et al* [OLJPG05]. Instead of quantising a cover Work along the linear axes (with each axis perpendicular to one another) of a lattice used by QIM, AQIM performs quantisation along the angular axes of a *hyperspherical coordinate system*.

In a L -dimensional hyperspherical coordinate system, the coordinates of a point in that space are given by a radius r and an angular vector $\theta = (\theta_1, \theta_2, \dots, \theta_{L-1})$.

An example of AQIM is given in Figure 2.18, which shows a 2-dimensional vector space with 8 quantisation axes uniformly separated from one another and radially out from the origin (centre). These 8 axes are divided equally into 2 subsets corresponding to each of the message bit '0' (denoted by \circ) and '1' (denoted by \square) as shown in Figure 2.18. The detection region, i.e. an area whereby any point within it is decoded to the assigned message bit, is clearly indicated, with the shaded region corresponding

mentioned in Section 2.1. In order to fully understand dirty paper trellis codes (DPTC), the concepts of channel coding have to be first explained in details. Hence in the next chapter, we will cover the concepts of channel coding and then go on to describe the principles of dirty paper trellis codes (DPTC).

Chapter 3

Dirty Paper Trellis Codes

The basic idea of dirty paper trellis codes (DPTC) [MDC02] is the use of a trellis code to distribute the codewords on the surface of a high-dimensional sphere. The selection of the chosen codeword (i.e. watermark patterns) for embedding, and the detection of the embedded codeword, is based on correlation values, hence the problem of amplitude scaling encountered by lattices codes and syndrome codes (refer to Section 2.8) does not affect dirty paper trellis codes.

In order to understand dirty paper trellis codes, we have to first understand the concept of trellis codes (i.e. convolutional codes), which are a type of channel code, or error correcting code. Convolutional codes are generated by passing a data stream through a *linear finite-state shift register*, which consists of a finite number of memory blocks. Because of these memory blocks, the output bits of a convolutional encoder are related to one another. Another type of channel code, linear block code, are composed of fixed-length codewords. In general, an (n, k) block code means that the codewords are n binary bits long and contain k binary bits of information. A description of both the block codes and convolutional codes are provided in Section 3.1.

With a deeper understanding of error correcting codes, we see how trellis codes can be used for watermarking purposes in Section 3.2. However, the traditional trellis codes cannot provide dirty paper coding (mentioned in Section 2.6), which can eliminate the interference from the cover Work and improve the performance of watermarking systems. Therefore, Miller *et al.* [MDC02] proposed dirty paper trellis codes, which is a modification to traditional trellis codes. Dirty paper trellis codes provide a one-to-many mapping between the input message bits and the codewords (i.e. watermark patterns). Section 3.3 provides a more detailed description of dirty paper trellis

codes.

To implement a dirty paper trellis for watermarking, the selection of a suitable trellis depends on various parameters. A brief discussion on the parameters is covered in Section 3.4.

3.1 Error Correcting Codes

Modern error correcting methods started with Shannon [Sha48], Hamming [Ham50], and Golay [Gol49]. Since then, intensive research has been carried out on new error correcting methods to try to approach Shannon's channel capacity limit [Sha48]. As we have seen in Section 2.1, channel codes achieve error correcting capabilities by introducing redundancies, i.e. adding additional bits to the original information bits. There are two fundamentally different approaches to add this redundancy: linear block codes and convolutional codes. This section will begin the discussion of linear block codes, which is an easy way to implement redundancies by appending *parity-check* bits to the original information bits. The parity-check bits not only detect any errors that may be present, but also correct errors up to a certain limit.

The second part of this section deals with convolutional codes, which are obtained by passing a data stream through a linear finite-state shift register. By doing so, the output bits of the convolutional encoder will be related to one another. Since noise can be present in a communication channel, errors may be found in the output stream from the convolutional encoder and errors within this output stream may be corrected using the output bits around the error. However, there is a limit to the number of errors that can be corrected. A very efficient way to decode this output stream is the *Viterbi algorithm* [Vit67].

3.1.1 Linear Block Codes

A block code consists of fixed-length vectors called codewords. The length of the codeword is the number of elements in the vector and is denoted by n . The elements of a codeword are chosen from an alphabet \mathcal{A} of q symbols. Since we will be dealing mostly with binary symbols, the alphabet \mathcal{A} consists of two symbols '0' and '1', i.e. $q = 2$. In this case, there are 2^n possible codewords in this binary code of length n . Out of these 2^n possible codewords, a *codebook* \mathcal{C} consisting of $M = 2^k$ codewords

are selected to form a code and these 2^k codewords are called valid codewords. Thus k information bits are encoded into a codeword of length n , i.e. there is a redundancy of $(n - k)$ bits. The resulting block code is referred to as an (n, k) code.

The number of different elements between any two codewords c_i and c_j within the codebook \mathcal{C} having M valid codewords is called the *Hamming distance* between the two codewords, denoted as $d[i, j]$, where $0 \leq d[i, j] \leq n$. The smallest value within the set $d[i, j]$ for the M codewords is called the *minimum distance*, denoted by d_{min} , of the code. Generally, a code with a larger d_{min} indicates a larger separation between the codewords and hence has the ability to detect and/or correct more errors.

A linear block code can be expressed using linear algebra. Suppose $\mathbf{x} = [x_1, x_2, \dots, x_k]$ are the information bits encoded into its codeword $\mathbf{c} = [c_1, c_2, \dots, c_n]$ such that

$$\mathbf{c} = \mathbf{xG} \quad (3.1)$$

where \mathbf{G} is called the generator matrix of the code, and is

$$\mathbf{G} = \begin{bmatrix} g_{1,1} & g_{1,2} & \dots & g_{1,n} \\ g_{2,1} & g_{2,2} & \dots & g_{2,n} \\ \vdots & \vdots & & \vdots \\ g_{k,1} & g_{k,2} & \dots & g_{k,n} \end{bmatrix} \quad (3.2)$$

By elementary row operations, \mathbf{G} can be expressed in its *systematic form* as shown below.

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}] = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & \dots & 0 & p_{1,1} & p_{1,2} & \dots & p_{1,n-k} \\ 0 & 1 & 0 & \dots & 0 & p_{2,1} & p_{2,2} & \dots & p_{2,n-k} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & p_{k,1} & p_{k,2} & \dots & p_{k,n-k} \end{array} \right] \quad (3.3)$$

where \mathbf{I}_k is the $k \times k$ identity matrix and \mathbf{P} is a $k \times (n - k)$ matrix containing the $(n - k)$ redundant bits or parity-check bits. Note that the rank of \mathbf{G} , i.e. the number of linearly independent rows in \mathbf{G} , must be equal to k . If the rank of \mathbf{G} is less than k , then the code is not considered a valid linear block code.

When Equation 3.3 is substituted into Equation 3.1, the resulting codeword is

$$\mathbf{c} = \mathbf{xG} = [\mathbf{xI}_k | \mathbf{xP}] = [\mathbf{x} | \mathbf{xP}] \quad (3.4)$$

It can be seen from Equation 3.4 that the final code consists of the k original information bits \mathbf{x} and the $(n - k)$ parity bits \mathbf{xP} .

For every generator matrix \mathbf{G} , there is a corresponding $(n - k) \times n$ parity check matrix \mathbf{H} which can be expressed as

$$\mathbf{H} = [\mathbf{P}^T | \mathbf{I}_{n-k}] \quad (3.5)$$

where \mathbf{P}^T is the transpose of the matrix \mathbf{P} and \mathbf{I}_{n-k} is the $(n - k) \times (n - k)$ identity matrix.

Let $\mathbf{y} = [y_1, y_2, \dots, y_n]$ be the corrupted received codeword, i.e. $\mathbf{y} = \mathbf{c} + \mathbf{e}$, where $\mathbf{e} = [e_1, e_2, \dots, e_n]$ is the error vector. During decoding, the product \mathbf{yH}^T yields

$$\begin{aligned} \mathbf{yH}^T &= (\mathbf{c} + \mathbf{e})\mathbf{H}^T \\ &= \mathbf{cH}^T + \mathbf{eH}^T \\ &= \mathbf{eH}^T = \mathbf{s} \end{aligned}$$

where \mathbf{s} is a $1 \times (n - k)$ vector called the *syndrome of the error pattern*. Note that since \mathbf{c} is a valid codeword, multiplying with the parity check matrix \mathbf{H}^T of the code will yield a zero vector, $\mathbf{0}$. Note that $\mathbf{GH}^T = \mathbf{0}$.

The syndrome, \mathbf{s} , can be used in error detection and/or error correction. Note that there are 2^n error patterns and only 2^{n-k} syndromes since the error patterns are of length n whereas the syndromes are of length $(n - k)$. Therefore k different error patterns can result in the same syndrome.

To carry out decoding, a decoding table is first constructed by listing the first row with all the 2^k possible codewords, beginning with the all-zero codeword, \mathbf{c}_1 , and ending with the codeword, \mathbf{c}_{2^k} , as shown in Table 3.1. Next, the first column is filled in with all $(n - 1)$ error patterns of weight 1. If $n < 2^{n-k}$, all double error patterns, triple error patterns, etc., may be filled into the table until there are 2^{n-k} entries in the first column. Hence there are a total of 2^{n-k} rows and it equals the number of syndromes. Next, each error pattern in the first column is added to the corresponding codewords to form the complete table shown in Table 3.1. Note that the error pattern \mathbf{e}_1 is an all-zero vector.

This decoding table is called a standard array with each row consisting of the possible received codewords that would result from a corresponding error pattern from

Error Patterns	Codewords				
e_1	c_1	c_2	c_3	...	c_{2^k}
e_2	$c_1 + e_2$	$c_2 + e_2$	$c_3 + e_2$...	$c_{2^k} + e_2$
\vdots	\vdots	\vdots	\vdots		\vdots
$e_{2^{n-k}}$	$c_1 + e_{2^{n-k}}$	$c_2 + e_{2^{n-k}}$	$c_3 + e_{2^{n-k}}$...	$c_{2^k} + e_{2^{n-k}}$

Table 3.1: Standard array of a linear block code

the first column. Given a received codeword, its entry is looked up in the standard array, and the corresponding error pattern, \hat{e} , can be determined. This error pattern is then added to the received codeword y to give the decoded word

$$\hat{c} = y \oplus \hat{e}$$

Having gone through the decoding method for an (n, k) linear block code, it is interesting to know how many errors can be detected and corrected. The following gives an answer to this question. When the syndrome consists of all zeroes, the received codeword is one of the possible 2^k transmitted codewords. It is possible for an error pattern of weight d_{min} to transform one of these 2^k codewords to another codeword since the minimum separation between a pair of codewords is d_{min} , therefore creating an undetected error. However, if the actual number of errors is less than d_{min} , the syndrome will have a nonzero weight. In this case, one or more errors can be detected. Therefore an (n, k) linear block code can detect up to $d_{min} - 1$ errors.

To determine the number of errors that can be corrected by an (n, k) code, the 2^k codewords are viewed as points in an n -dimensional space where each codeword is the centre of a sphere of radius (Hamming distance) t . The largest value of t such that there is no intersection between any pair of the 2^k spheres is $t = \lfloor \frac{1}{2}(d_{min} - 1) \rfloor$, where $\lfloor x \rfloor$ denotes the closest integer smaller than x . All possible codewords of distance less than or equal to t from a valid codeword lie within this sphere. Any received codeword that is within a valid sphere is decoded to the valid codeword at the centre of the sphere. Therefore an (n, k) linear block code with minimum distance d_{min} can correct up to $t = \lfloor \frac{1}{2}(d_{min} - 1) \rfloor$ errors.

Example. As an illustration to the coding and decoding of linear block codes, consider

an (5, 2) linear block code with generator and parity-check matrices given by

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The standard array and the corresponding syndromes are given in Table 3.2. From the valid codewords (those in bold), the code has a minimum distance $d_{min} = 3$. Note that all five error patterns of weight 1 are included and there is only room for two error patterns of weight 2, i.e. this code can correct all single errors and only two double errors, namely [1 1 0 0 0] and [1 0 0 1 0]. If the received codeword $\mathbf{y} = [1 0 1 0 0]$, by looking up Table 3.2, the syndrome for the error is $\mathbf{s} = [0 0 1]$ and the corresponding error pattern determined is $\hat{\mathbf{e}} = [0 0 0 0 1]$. The most likely transmitted codeword is then computed by adding (modulo-2) $\hat{\mathbf{e}}$ to \mathbf{y} , i.e. $\hat{\mathbf{c}} = \mathbf{y} + \hat{\mathbf{e}} = [1 0 1 0 1]$.

Syndromes	Error Patterns	Codewords			
0 0 0	0 0 0 0 0	0 0 0 0 0	0 1 0 1 1	1 0 1 0 1	1 1 1 1 0
0 0 1	0 0 0 0 1	0 0 0 0 1	0 1 0 1 0	1 0 1 0 0	1 1 1 1 1
0 1 0	0 0 0 1 0	0 0 0 1 0	0 1 0 0 1	1 0 1 1 1	1 1 1 0 0
1 0 0	0 0 1 0 0	0 0 1 0 0	0 1 1 1 1	1 0 0 0 1	1 1 0 1 0
0 1 1	0 1 0 0 0	0 1 0 0 0	0 0 0 1 1	1 1 1 0 1	1 0 1 1 0
1 0 1	1 0 0 0 0	1 0 0 0 0	1 1 0 1 1	0 0 1 0 1	0 1 1 1 0
1 1 0	1 1 0 0 0	1 1 0 0 0	1 0 0 1 1	0 1 1 0 1	0 0 1 1 0
1 1 1	1 0 0 1 0	1 0 0 1 0	1 1 0 0 1	0 0 1 1 1	0 1 1 0 0

Table 3.2: Standard array and syndromes for a (5, 2) linear block code.

Examples of block codes are repetition codes, Hamming codes [Ham50], Golay codes [Gol49], Bose-Chaudhuri-Hocquenghem (BCH) codes [Hoc59, BRC60b, BRC60a], and Reed-Solomon (RS) codes [RS60].

3.1.2 Convolutional Codes

The second type of channel code are convolutional codes, which were proposed by Elias [Eli54] because he believed that linear block codes do not exploit the channel

to its fullest. A convolutional binary encoder is implemented using a finite-state shift register and an example is shown in Figure 3.1. The shift register consists of ν memory blocks (delay elements) and n linear algebraic function generators, which are modulo-2 adders. At any time instance, assuming that the input data to the encoder is binary, k bits are drawn into the finite-state shift-register and shifted along the shift register before producing n output bits. Hence, the code rate is defined as k/n . The shift register contributes to the interconnectivity between the output bits, and thus the error correcting capability of the convolutional code. The parameter ν , also known as the constraint length of the convolutional code, determines the total number of possible states, $S = 2^\nu$. The current *state* of the convolutional code is determined by the bits stored inside all the memory blocks at that particular time instance. Note that the current state is dependent on the previous state and the input bits. In Figure 3.1, the convolutional encoder has $\nu = 2$ memory blocks and a code rate of $1/3$ since $k = 1$ and $n = 3$.

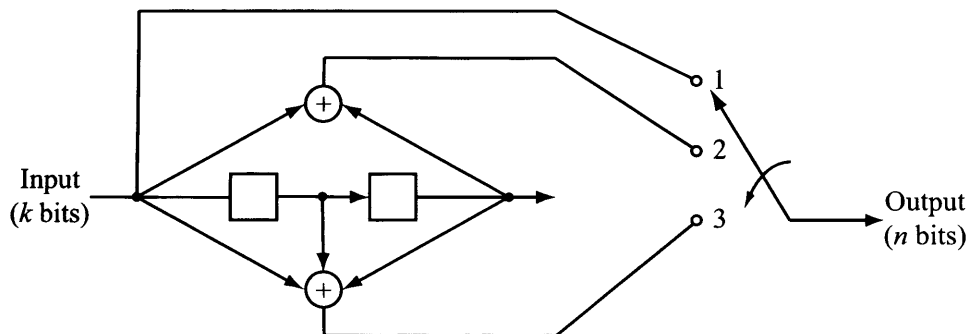


Figure 3.1: A convolutional encoder with $\nu = 2$ memory blocks, $k = 1$ input bit and $n = 3$ output bits.

The output of the convolutional encoder can be determined using its generator polynomials, one for each output bit. The generator polynomials depend on the connection to the memory blocks (delay elements). In the case of Figure 3.1, the first generator polynomial is

$$g_1 = [100]$$

since the output is connected only to the first stage, i.e. before the first memory block.

The second generator polynomial is

$$g_2 = [101]$$

since the output is connected to the first and third stage, i.e. before the first memory block and after the second memory block respectively.

And finally the third generator polynomial is

$$\mathbf{g}_3 = [111]$$

Note that just as for the block codes in Equations 3.2 and 3.3, the above generator polynomials can be expressed in the generator matrix form.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1^T & \mathbf{g}_2^T & \mathbf{g}_3^T \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad (3.6)$$

A convolutional encoder can also be described using a *state machine* diagram or a *trellis* diagram. A state machine is simply a graph of all the possible *states* of the convolutional encoder and the possible transitions from one state to another. The state of an encoder is determined by the bits stored in the memory blocks (see Figure 3.1). The state machine corresponding to the convolutional encoder used in Figure 3.1 is illustrated in Figure 3.2. In this case, the state machine has four states (because there are two memory blocks, i.e. $\nu = 2$) with an initial state A. For every input bit processed by the convolutional encoder, a codeword of three bits is produced and subsequently assigned to the respective arcs in Figure 3.2. If the input bit is a '0', the non-bold arc is traversed and the output is the three bits corresponding to the arc. Similarly, if the input bit is a '1', a bold arc is traversed and the three bits corresponding to the arc are the output. The state will change accordingly depending on the previous state and the input bit.

Let us consider the state machine shown in Figure 3.2 and a 4-bit message sequence 1010. Starting from an initial state A, the first bit, a '1', will select the bold arc from state A to state C and the output is 111. Now at state C, the second bit, a '0', will select the non-bold arc from state C to state B and the output is 001. This carries on until the last bit and the final output sequence is 111 001 100 001. Note that each input bit affects not only the three current output bits, but also several subsequent output bits. Therefore the subsequent output bits contain redundant information about the earlier bits.

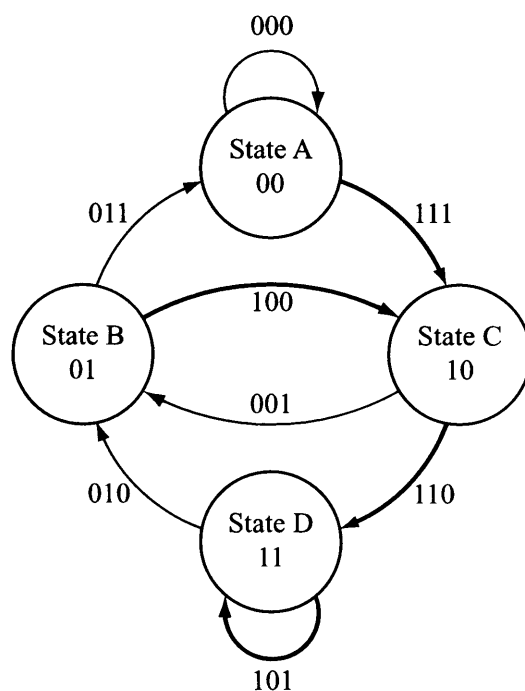


Figure 3.2: State machine representation for a four-state convolutional code.

Another representation of the convolutional code is a trellis, where an example of the convolutional encoder of Figure 3.1 is shown in Figure 3.3. A trellis consists of many nodes and each of them corresponds to a specific state at different times. In this case, there are four nodes at each stage and each node has two incoming arcs and two outgoing arcs since the input of the convolutional encoder at any time instance is one bit, i.e. only two possible values. In other situations, there may be more than one input bit per time instance and hence there can be more than two incoming and outgoing arcs per node. Using the same convention as in Figure 3.2, a non-bold arc and a bold arc represent bits '0' and '1' respectively. Figure 3.3 illustrates an example of a trellis with an initial state A and the input message bits are 1010. The highlighted path corresponds to the encoding of the message bits 1010.

The same trellis is used during decoding and the decoder computes the most likely transmitted path traversed through the trellis. Thus, the bit sequence corresponding to this most likely path will be the estimated message. However, if the original message is long, the trellis will have a lot of stages and computation during decoding will be too intensive if the likelihoods of all possible paths through the trellis are calculated. Fortunately, Viterbi [Vit67] proposed a very efficient way to obtain the most likely

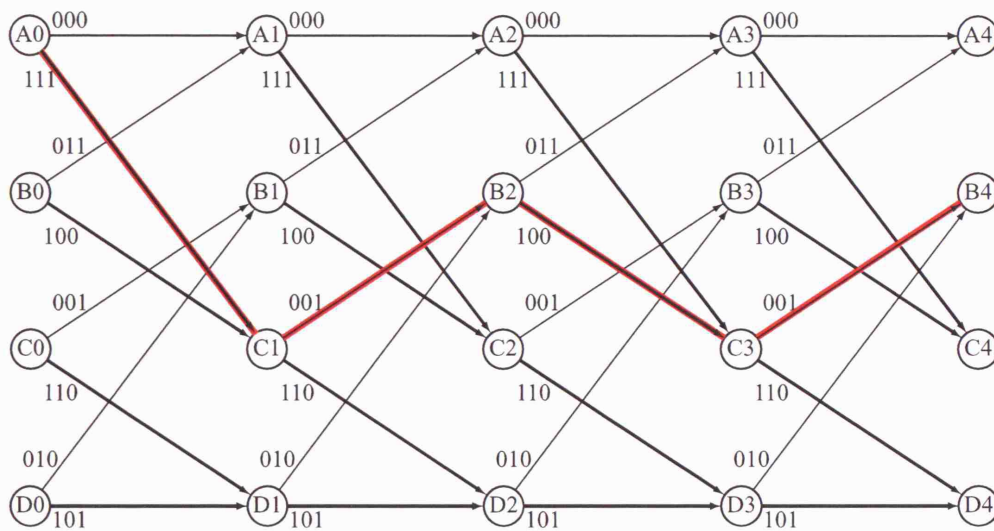


Figure 3.3: Trellis representation of the convolutional encoder of Figure 3.1.

path through the trellis by performing *maximum-likelihood* (ML) decoding. This ML decoder chooses the codeword that minimises the Hamming distance (or *Euclidean distance* if received codeword consists of soft values, i.e. not just ‘0’ and ‘1’) between the received codeword and all possible codewords, i.e. minimum distance decoding. A value, called a *branch metric*, is associated to each arc and is the Hamming distance between the received codeword and the arc at that time instance. This branch metric can be seen as the cost associated with traversing that arc. Another value, called the *path metric*, associated with each node is the sum of the path metric of the predecessor and the branch metric of the connecting arc. The following gives a pseudocode for the Viterbi decoding algorithm.

1. Assuming that the convolutional encoder is at the zero state initially, assign the path metric zero to the initial node; set $t = 0$.
2. For each node at stage $(t + 1)$, find for each of the predecessors at stage t the sum of the path metric of the predecessor and the branch metric of the connecting arc (**ADD**). Determine the minimum of these sums (**COMPARE**) and assign it to this node; label the node with the shortest path (also known as *survivor path*) to it (**SELECT**).
3. If we have reached the end of the trellis, then stop and choose as the decoded

codeword the path to the terminating node with the smallest path metric; otherwise increment t by 1 and go to *Step 2*.

An example of Viterbi decoding for the received sequence 101 001 101 001, i.e. a corrupted version of the encoded bits from Figure 3.3, is given in Figure 3.4. In order to find the codeword closest (in terms of Hamming distance) to the received sequence, the trellis is traversed from left to right, eliminating all paths that would not be the prefix of the most likely path through the trellis. Each stage represents an iteration of the encoding. Stage 0 is before the start of encoding and stage 1 is after the first bit is encoded, and so on. The transition between one stage to next stage is defined as a *step*, which corresponds to one decoded bit in this case. When stage 2 is reached, there are four paths - one for each node. However at the next stage, i.e. stage 3, there are eight paths — two per node. For each node at this stage, only one path of the two incoming paths will remain - the one closest (in terms of Hamming distance) to the corresponding prefix of the received sequence. The other path into each node will be eliminated since it could not be the prefix of the most likely path through the trellis. The reason is that, since the Viterbi algorithm maximises the probability of a correct decision, a path with a higher path metric (i.e. Hamming distance) results in a lower probability of correct decision. This process continues until the whole trellis is traversed, i.e. until stage 4. The most likely path through the trellis is the one with the smallest Hamming distance. In this case, the highlighted path in Figure 3.4 is the most likely path that corresponds to the decoded sequence 1010. In Figure 3.4, the Hamming distance between the prefix of received sequence and the most likely path leading to each node is shown above the nodes. The eliminated path to each node is marked with the symbol 'x'.

As mentioned earlier, the Viterbi algorithm is an efficient way to find the most likely path through the trellis and the explanation for its efficiency is as follows. Assume there are S number of states and L steps in the trellis. If an exhaustive search is used to find the most likely path through the trellis, all 2^L possible paths through the trellis are to be taken into consideration and comparison at the end of the trellis. The number of possible paths through the trellis grows exponentially as L increases. If the Viterbi decoding algorithm is used, there are only S possible paths at the end of the trellis to be taken into consideration for comparison, regardless of L . The reason is that after $\log_2(S)$ steps and for the remaining $(L - \log_2(S))$ steps, i.e. when all the states, at

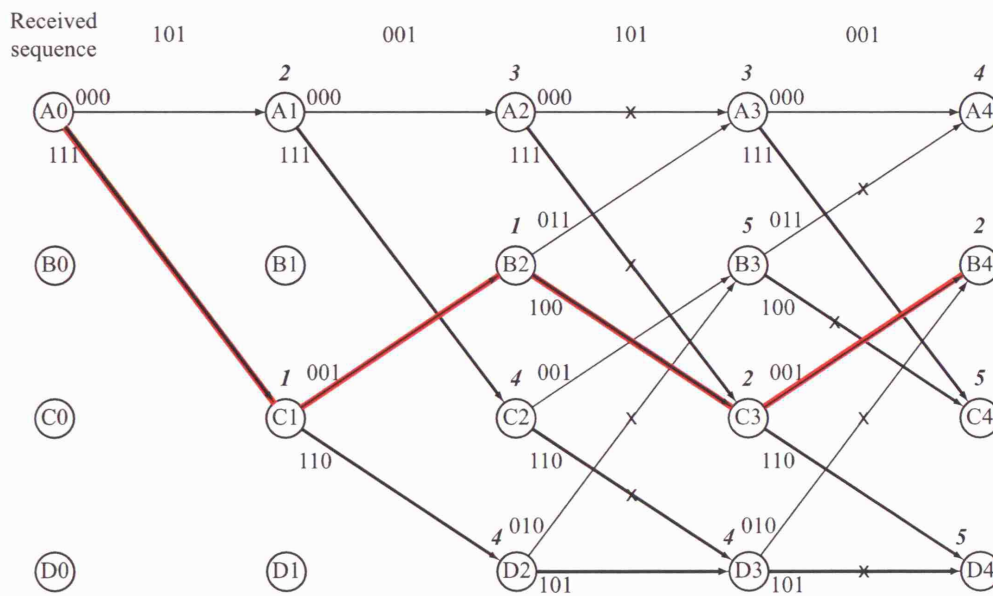


Figure 3.4: Viterbi decoding algorithm.

one time instance, have been connected by the arcs, the number of paths double at the start of each step and half the number of paths are eliminated at the end of each step. This eliminates those paths that have larger path metrics at each node and are definitely not going to be the most likely path even before they reach the end of the trellis. In this case, this minimises the probability of error or, in another words, maximises the probability of correct decision. Note that the Viterbi algorithm is optimum when the channel is additive white Gaussian noise (AWGN).

3.2 Trellis Codes for Watermarking

Practical dirty paper codes, described in Section 2.8, are designed in such a way where codewords are grouped into subsets and each subset is associated with a given message. A practical dirty paper watermarking scheme has to be designed to allow efficient search of codewords during watermarking embedding (at the transmitter) and watermark detector (at the receiver). At the watermark embedder, the codeword, which is within the subset belonging to a given message, *closest* to the cover Work, c_o , has to be found quickly. During watermark detection, the subset within which the codeword that is *closest* to the watermarked Work, c_w , is to be determined efficiently as well. It should be noted that the cover Work, c_o , can be obtained from many different ways.

One way is to get some pixel values of a multimedia content to be the elements of the cover Work [vSTO94]. Another is to allow the elements of the cover Work to be from some transformed domain, such as Discrete Cosine Transform (DCT) [TD97], Discrete Fourier Transform (DFT) [RDB96], Discrete Wavelet Transform (DWT) [BBC⁺99], of a multimedia content, selecting only a portion to form the cover Work.

This watermarking case is similar to that of the traditional error correcting codes. In this section, we see how a traditional trellis, as depicted in Figure 3.5, can be used for watermarking. Each node has two arcs emanating from it to two different nodes in the next column of nodes. A *step* is defined to be the transition from one column of nodes to the next column of nodes, moving from left to right. Each step corresponds to one message bit and each arc is labelled with a reference pattern of length N , i.e. a N -samples pseudo-random sequence. Assuming that the starting node is $A0$, the trellis is traversed from left to right by choosing a bold arc if the message bit is ‘1’ or a non-bold arc if the message bit is ‘0’. Thus, each L -bit message is mapped to a unique L -step path through the trellis, i.e. a one-to-one mapping, and the length $(N \times L)$ output watermark pattern, w_m , is obtained by concatenating the labels (i.e. reference patterns) associated with the arcs of the path. Figure 3.6 shows an example of the chosen path if the first three bits of a L -bit message are ‘1 0 0’ and the last bit is ‘1’. It should be noted that the cover Work, c_o , is not involved in this process, and hence, this process is called *blind coding*.

The resulting watermark, w_m , is subsequently embedding into the cover Work using a simple blind additive approach:

$$c_w = c_o + \alpha w_m \quad (3.7)$$

where α is the embedding strength. During watermark detection, the *most likely* path through the trellis is determined by using the Viterbi decoding algorithm mentioned in Section 3.1.2. To do so, each step of the trellis is associated with a portion of the cover Work, c_o , e.g. N samples or coefficients. The cost of traversing an arc is then defined as the *linear correlation* between the reference pattern associated with the arc and the relevant portion of the watermarked Work c_w . As a result, the Viterbi decoding algorithm finds the path which exhibits the *highest* linear correlation with the watermarked Work c_w .

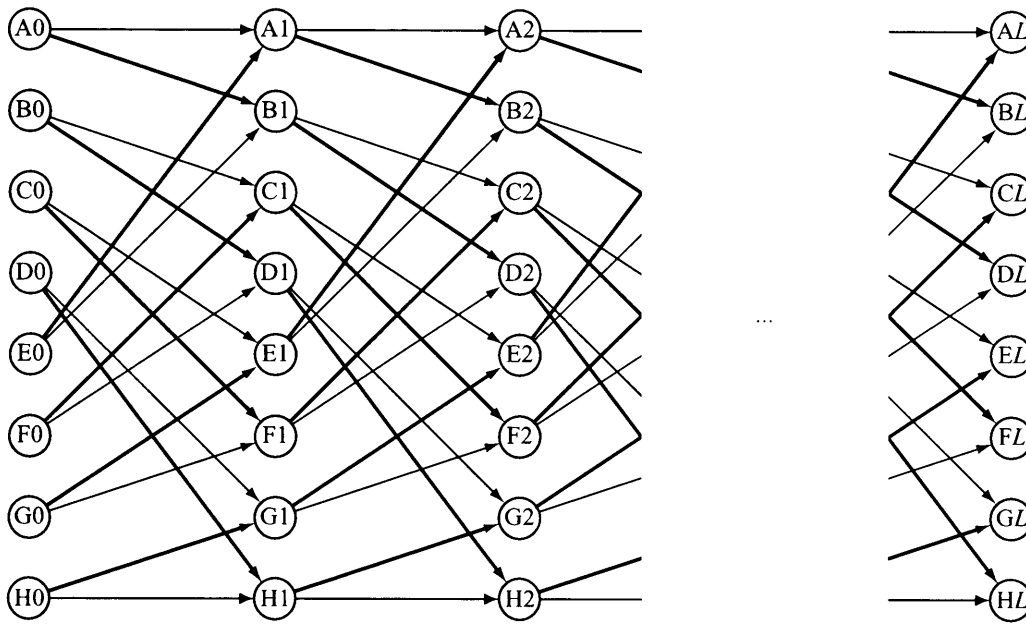


Figure 3.5: A traditional 8-state trellis.

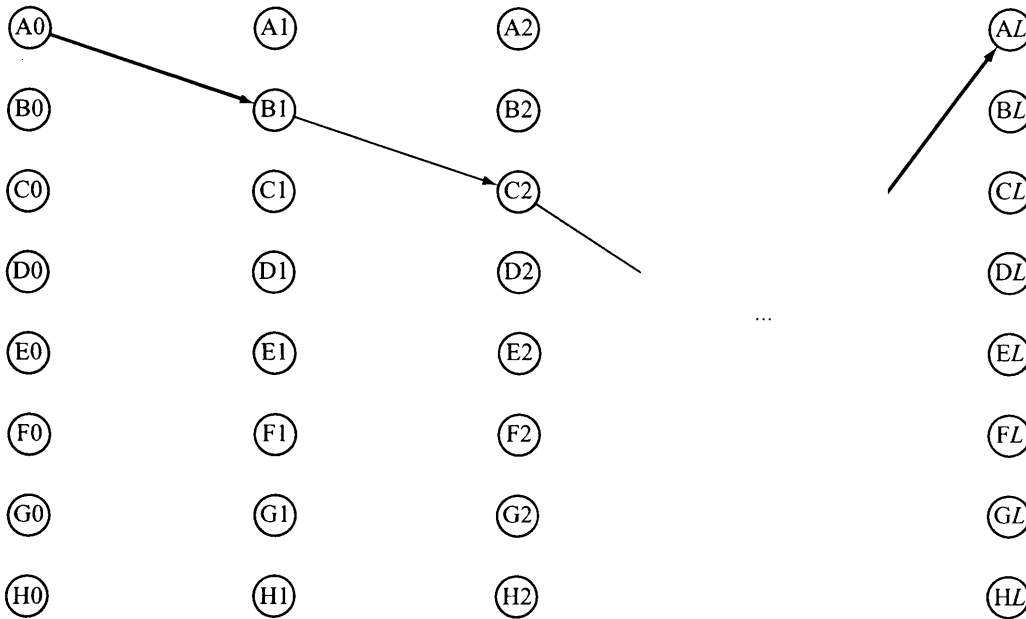


Figure 3.6: A message selects a unique path through the trellis.

3.3 Informed Coding

A given message, m , may be represented by several alternative codewords. In order to provide a one-to-many mapping between messages and codewords, a simple modifica-

tion is made to the traditional trellis. A computationally efficient way to map a message to a desired codeword, consists of modifying a traditional trellis so that more than two arcs leave and enter a node. Such a trellis is shown in Figure 3.7 and is referred to as dirty paper trellis [MDC02]. Once again, a bold arc is traversed if the corresponding bit of the message is a '1', and a non-bold arc is traversed if the corresponding bit is a '0'. A dirty paper trellis has the property that several paths through the trellis encode the same message m . It is consequently necessary to tailor a procedure which decides which path, and by extension which watermark signal, will be determined for embedding. This is where the original cover Work, c_o , plays a role, and hence, this process is called *informed coding*.

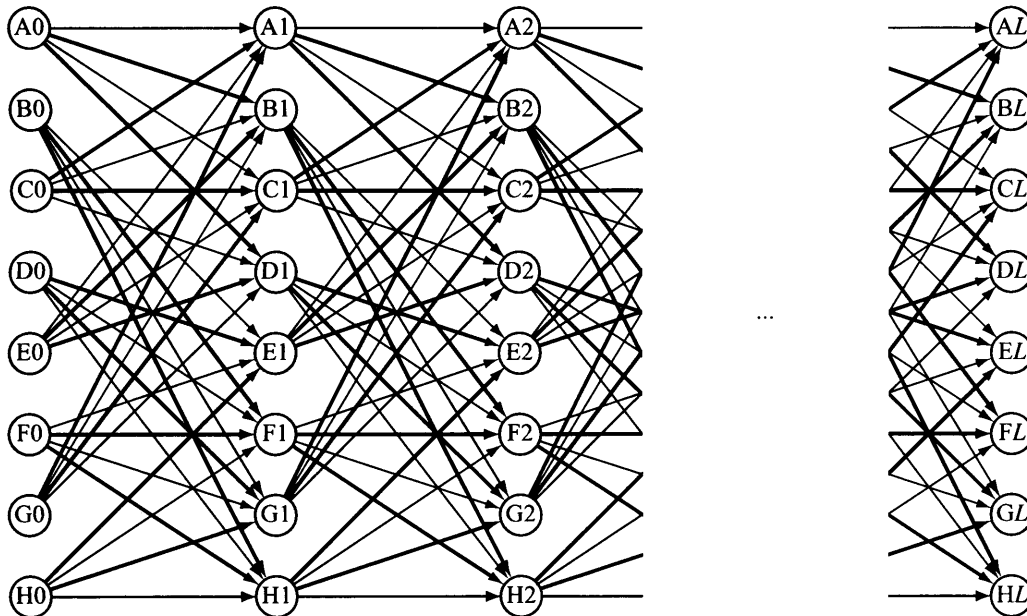


Figure 3.7: A dirty paper trellis with 8 states and 4 arcs per state.

During the embedding process, the choice of which codeword to embed is determined by first modifying the dirty paper trellis so that all paths through the trellis encode the same message m . This is accomplished by removing all the arcs which do not encode the desired message. For example, if the first message bit is a '1', the non-bold arcs are removed in the first step (nodes $A0 \cdots H0$ to nodes $A1 \cdots H1$). Figure 3.8 shows a *modified trellis* that corresponds to a message with '1 0 0' as the first three bits and '1' as the last bit. The Viterbi decoder is then run to find the path through this

modified trellis which has the highest linear correlation with the input cover Work c_o . Once again, the watermark, w_m , is obtained by concatenating all the labels of the arcs along the identified best path. In contrast to blind coding, both the message, m , and the cover Work, c_o , influence the encoding process.

The resulting watermark, w_m , is then embedded blindly according to Equation 3.7. At the detector, the decoder applies the Viterbi algorithm to the entire dirty paper trellis, as depicted in Figure 3.7. This identifies the path through the trellis which has the highest linear correlation with the watermarked Work c_w . The hidden message can then be determined by examining each arc in the optimum path to determine whether it encodes a '1' or '0'.

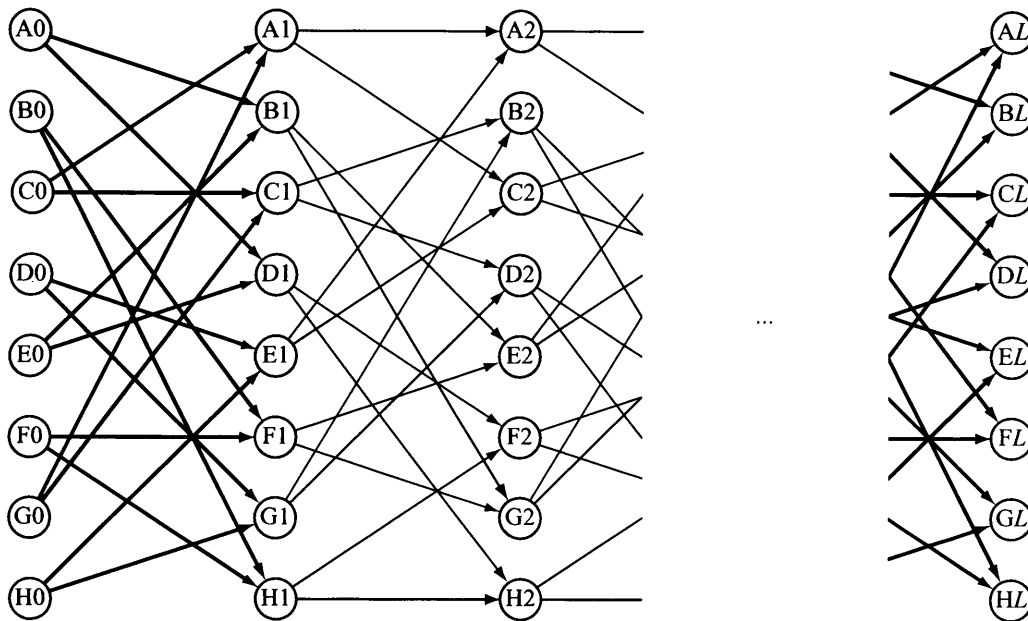


Figure 3.8: A modified trellis that corresponds to a given message.

3.4 Parameters Affecting Dirty Paper Trellis

A dirty paper trellis code is a form of spherical code since all of its codewords have equal energy, i.e. same distance from the origin, and therefore are distributed on the surface of a high dimensional sphere. A spherical code is designed such that its codewords are well-distributed across the surface of the sphere, i.e. the minimum distance between any pair of codewords is maximised. This set of codewords is usually divided

in cosets (also known as subsets) and these cosets are well-distributed over the surface of the sphere. Any point on the sphere should be close to a codeword in a coset so that the distortion is minimised. Generally, a well-distributed set of codewords would generate a better performance. In view of this, the design of dirty paper trellis codes has to take into account the codeword distribution to ensure it has a good performance.

The performance of a system or scheme can be measured in several ways. One of them is to determine the *received signal-to-noise ratio*, i.e. a ratio of the transmitted signal power at the receiver to the noise present in the received signal. Because the transmitter and the receiver can be a distance away from each other, the received signal power can be considerably lower than the transmitted signal power. In this case, if the received signal-to-noise ratio (SNR) is lower than the expected level, then the system performance is said to be unsatisfactory. Another method to determine the performance of a system is to measure the frequency of errors during transmission. This can be done using an error ratio, which is the ratio of the number of bits, symbols, or blocks incorrectly received to the total number of bits, symbols, or blocks sent during a specified time interval. The most commonly encountered ratio is the bit error rate (BER).

Upon closer examination of the dirty paper trellis, we can determine several parameters that can affect the distribution of its codewords. They are:

- the number of states, S , in the trellis,
- the number of arcs, A , entering/leaving each node,
- the connectivity between the nodes, i.e. how the arcs join from one node to another,
- the set of reference patterns used to label the arcs,
- the mapping between the reference patterns and the arcs, and
- the mapping between the *dirty* bits (bold vs. non-bold) and the arcs

The performance of convolutional codes improves as the number of states, S , increases since the encoder has more memory blocks and the coded bits have higher interdependence, thus higher error correcting capability (refer to Section 3.1.2 for more details). Since a dirty paper trellis is based on convolutional codes, the number of states,

S , together with the number of arcs per state, A , also affect the trellis performance. The effect of these two parameters on the performance of a dirty paper trellis is discussed in Chapter 4.

The connectivity between the nodes, the set of reference patterns for arcs labelling, the mapping between the reference patterns and the arcs, and the mapping between the *dirty* bit and the arcs can also affect the distribution of codewords on the surface of the sphere. Out of these four factors, the first three are also applicable to traditional trellises if the reference patterns are replaced by coded symbols. These three factors will affect a property of a trellis, called the *free distance*, i.e. the minimum distance between any pair of codewords having the same start state and end state, which has a direct link to the trellis performance. The discussion of all these four factors on dirty paper trellis is covered in Chapter 5.

While traversing a trellis, arcs are chosen based on the cost function, i.e. there is an associated cost of selecting each arc. Miller *et al.* [MDC02] proposed the linear correlation between a Work and the reference pattern labelled on each arc to be the cost function. The final path through the trellis is one with the highest linear correlation between a Work and the reference patterns associated with the arcs along this path. This may be desirable to some watermarking schemes, but it may not be the optimum solution depending on the embedding strategy. Chapter 6 explores different cost functions that will bring about better watermarking performance according to the embedding scheme.

Chapter 4

Trellis Structure and its Performance

In Chapter 3, we have seen how to generate a dirty paper trellis and a corresponding modified trellis (given a message) for watermark purposes. In this chapter, we examine how the structure of a dirty paper trellis, i.e. the number of states, S , and the number of arcs per state, A , can affect its performance. Performance of a dirty paper trellis, which is commonly measured in terms of bit error rate (BER), is an important issue for practical watermarking applications. Bit error rate (BER) is defined as a ratio of incorrectly received bits to total number of bits sent.

Several preliminary studies have been done on dirty paper trellis using simple additive embedding [MDC02] and informed coding with informed embedding [MDC04]. In those studies, a simple evaluation of dirty paper trellis reveals the relationship between its performance and its structure in a noiseless channel environment. Generally, the cover Work, c_o , lies in the detection region of a codeword that does not encode the message. Since there is no channel noise present, the errors come from the fact that the embedding of the watermark pattern, w_m , is not strong enough to “move” the cover Work, c_o , to the detection region of the correct codeword. Hence errors are present even if there is no channel noise. The results in [MDC02, MDC04] indicated that the greater the number of states, S , and the number of arcs per state, A , the better the performance due to the fact that the number of codewords increases as the number of states, S , and the number of arcs per state, A , increase. Hence a codeword close to the cover Work can be found. However, with an increased number of codewords, the minimum distance between any pair of codewords is reduced, thereby causing the code to be less resistant to noise. It was suggested in [MDC02, MDC04] that the optimum trellis configuration (using a noiseless channel) is $S = 64$ and $A = 64$ since a larger number of

states, S , and number of arcs per state, A , will only marginally improve the performance of dirty paper trellis.

The performance evaluation of dirty paper trellis from [MDC02, MDC04] is too simplistic since it only considers the noiseless channel. Although it is desirable to employ a dirty paper trellis with a larger number of states, S , and a larger number of arcs per state, A , to improve its performance in this case, there is a need to take into account the performance using a noisy channel as well as the computation time (or complexity). Therefore the aim of this chapter is to examine the relationship between the trellis structure and its error rate performance, using computational complexity as an additional factor in selecting a suitable trellis structure.

Firstly, the bit error rate (BER) performance of a dirty paper trellis is investigated using simple additive embedding described in Section 4.1. Having obtained the performance of a dirty paper trellis, the trellis complexity is then analysed and the selection of a suitable trellis configuration, using both the BER performance and computation time, are covered in Section 4.2.

4.1 Trellis Performance

Performance is one of the most important considerations in determining a trellis configuration for use in a watermarking application. In order to compare the performance of different dirty paper trellises, one has to keep one or more parameters fixed. To ensure a fair comparison between different dirty paper trellises with different number of states, S , and number of arcs per state, A , the total number of distinct paths through a dirty paper trellis (i.e. the number of different codewords) should be kept constant. The total number of codewords is given by

$$N_c = S.A^L \quad (4.1)$$

where L is the length of the trellis (i.e. the number of steps in the trellis). To illustrate this, Figure 4.1 depicts two alternative trellis configurations which share the same total number of codewords, $N_c = 16$. Configuration (b) is basically a traditional 4-state trellis whereas configuration (a) is a degenerate trellis with only a single state and 4 *parallel* arcs. Keeping the number of codewords constant means that the measured BER variations are only due to the changes in the trellis configuration, i.e. the minimum dis-

tance between any pair of codewords changes. In the proposed example, configuration (a) is memoryless, i.e. an error that occurs at one step is independently from decisions in the previous steps. Therefore, the BER is likely to be higher than with configuration (b). In the latter case, errors are more costly since an error at the first step inevitably induces another error at the second step.

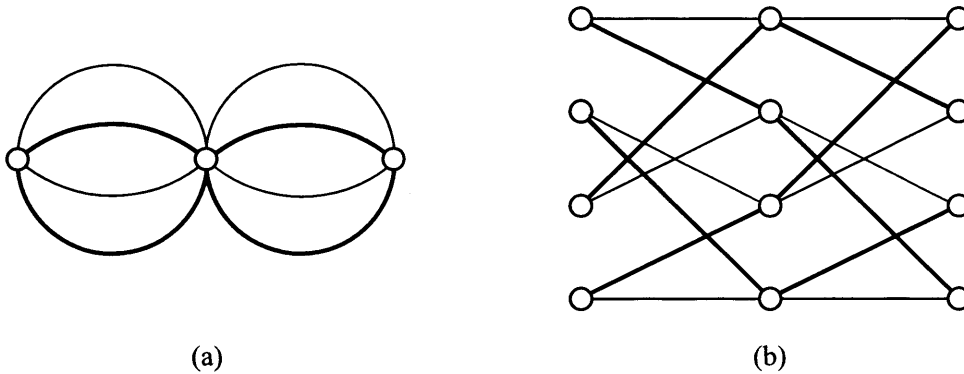


Figure 4.1: Different trellis configurations for 16 codewords: (a) 1 state, 4 arcs per state, length 2 trellis and (b) 4 states, 2 arcs per state, length 2 trellis.

In practice, maintaining a fixed number of codewords is not always feasible. Referring back to Equation (4.1), if the number of arcs per state, A , is divided by 2, then the number of states, S , has to be multiplied by 2^L , which grows exponentially and rapidly since L is large in practice. Therefore the decoding performances of all possible trellis configurations have to be evaluated.

To analyse the performance of a dirty paper trellis, several performance metrics such as Bit Error Rate (BER), Message Error Rate (MER), and Path Error Rate (PER) can be used. The BER is the probability a message bit is incorrectly decoded whereas the MER is the probability a message is decoded incorrectly, i.e. one or more bit errors indicates a message error. The PER is the probability a path (i.e. codeword) output from the Viterbi decoder differs between the path traversed during message coding at the embedder and the path traversed during watermark detection. Note that a path error may still occur even if the two different paths encode the same message, i.e. the BER and MER are zero.

For simplicity, experiments have been carried out with synthetic signals as cover

Works using simple additive embedding as shown in the following equation.

$$\mathbf{c}_w = \mathbf{c}_o + \alpha \mathbf{w}_m \quad (4.2)$$

where α is the embedding strength. This simple additive embedding is illustrated in Figure 4.2. Note that the power of the watermarked Work, \mathbf{c}_w , can be larger than that of the cover Work, \mathbf{c}_o , by a significant amount if a large embedding strength is allowed. This will undeniably cause a huge distortion to the cover Work, \mathbf{c}_o . The *Document-to-Watermark Ratio* (DWR), which is a ratio of the document (i.e. cover Work) power to the watermark power, is the distortion measure used in this set of experiments to control the embedding strength. A higher DWR indicates a lower distortion to the cover Work \mathbf{c}_o . The Document-to-Watermark Ratio is defined as

$$DWR = 10 \log_{10} \left(\frac{|\mathbf{c}_o|^2}{|\mathbf{c}_w - \mathbf{c}_o|^2} \right) = 10 \log_{10} \left(\frac{|\mathbf{c}_o|^2}{|\alpha \mathbf{w}_m|^2} \right) \quad (4.3)$$

where $|\mathbf{c}_o|^2$ is the power of the cover Work, \mathbf{c}_o , and $|\mathbf{w}_m|^2$ is the power of the watermark pattern, \mathbf{w}_m . It is obvious that the embedding strength, α , controls the Document-to-Watermark Ratio (DWR). For a desired, fixed DWR, α is given by

$$\alpha = \frac{|\mathbf{c}_o|}{|\mathbf{w}_m|} \times 10^{-0.1 \times DWR} \quad (4.4)$$

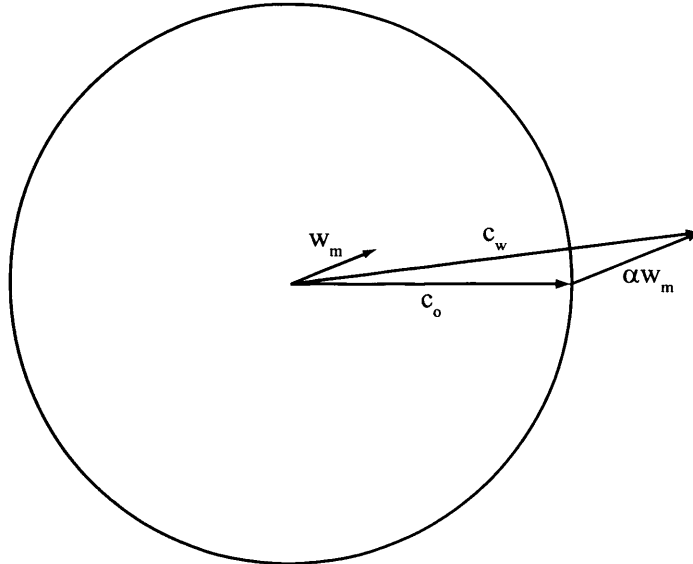


Figure 4.2: An example of a simple additive embedding.

To reduce computational complexity, the number of steps in the trellis has been set to 10 ($L = 10$), i.e. the hidden message consists of 10 bits. The length of the arc

labels, i.e. the number of elements in the reference patterns associated with the arcs, has been set to 64 ($N = 64$) and each element of those patterns is drawn from a Gaussian distribution with zero mean and unit variance. Since we are interested in the effect of the trellis structure, the number of states, S , and the number of arcs per state, A are parameters. Therefore for a selected set of couples (S, A) , the following experiment is run 10^6 times.

1. Generate a random cover Work $c_o \sim \mathcal{N}(0, 1)$ of length $N.L$.
2. Create a dirty paper trellis with S states and A arcs per state.
3. Generate a random L -bit message m .
4. Identify the path p_1 in the dirty paper trellis which encodes the message m and has the highest linear correlation with the cover Work, c_o , using the Viterbi decoder.
5. Use Equation 4.2 to embed the resulting watermark pattern, w_m , with an embedding strength $\alpha = \sqrt{0.1}$ so that the Document-to-Watermark Ratio (DWR) is equal to 10 dB¹.
6. Corrupt the watermarked Work c_w with additive zero mean Gaussian noise whose variance is adjusted to obtain a certain *watermark-to-noise ratio* $WNR = 10 \log_{10} \left(\frac{|\alpha w_m|^2}{|n|^2} \right)$, where $|n|^2$ is the power of the noise, n , added to the watermarked Work, c_w , before watermark detection.
7. Identify the path p_2 in the *entire* trellis which has the highest linear correlation with the corrupted watermarked Work using the Viterbi decoder.
8. Compute the number of different bits between p_1 and p_2 .

The BER can then be computed by dividing the total number of reported bit errors by the number of iterations times the length L of the path.

Figure 4.3 shows the measured BER for different dirty paper trellis configurations with no channel noise (i.e. additive Gaussian noise), n , with respect to the total number

¹This value has been chosen to observe enough bit errors to estimate the BER without running a huge number of iterations.

of codewords N_c . As shown in Figure 4.3, the BER first decreases as the number of codewords increases because the larger the codebook, the more likely that a codeword exists that is similar (i.e. highly correlated) to the cover Work, c_o . This similarity can be measured by the *normalised correlation*² between the cover Work, c_o , and the closest codeword, w_m , that codes the intended message, i.e. $\frac{c_o \cdot w_m}{|c_o||w_m|}$. Figure 4.4 shows average values of such normalised correlation for different trellis configurations computed during the experiment described above. From Figure 4.4, the average normalised correlation increases as the number of arcs per state, A , increases. This is quite obvious since the number of codewords, N_c , increases, thereby allowing the cover Work, c_o , to be closer to any of the codeword. Hence Figure 4.4 proves that a codeword is more likely to be similar (in terms of normalised correlation) to the cover Work, c_o , as the number of codewords, N_c , increases. Note that for a given average normalised correlation (from Figure 4.4), some trellises require more codewords than others to achieve the same correlation value. Since more codewords are needed to ensure that the distance between a randomly chosen cover Work, c_o , and the nearest codeword remains constant, there is a higher tendency for clusters (or non-uniform distribution) to form among codewords for those trellises requiring more codewords to achieve the same normalised correlation.

As the number of codewords increases, it is surprising to see that the BER increases at some trellis configurations before decreasing again, thus creating a “bump”. Upon closer examination of Figure 4.3, it can be noted that the bumps occur when parallel arcs start to occur within the trellis (denoted by the black circles \bigcirc in the figure). Parallel arcs happen when arcs are linked to the same nodes, e.g. from node $A0$ to node $A1$, as shown in Figure 4.5. In this case, single errors can occur without inducing additional errors. In other words, making errors is easier and thus errors happen more often. This is an important difference, with trellis configurations on the left side of these bumps where an error necessarily induces other ones, i.e. making an error is more costly and thus happens more rarely. This suggests that configurations with $A > S$ should be avoided. Finally, it can be observed that, for a given number of codewords (roughly, A constant), very different BER can be obtained depending on the

²Since the cover Work, c_o , can have varying power, normalised correlation is used instead of linear correlation in order for fair comparison.

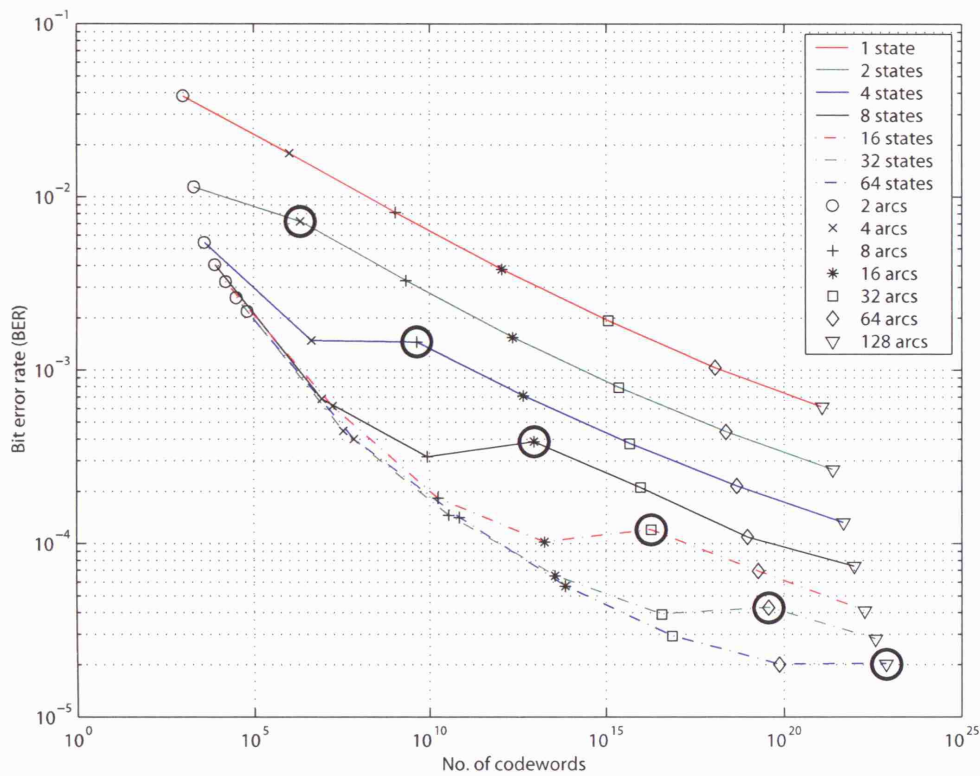


Figure 4.3: BER as a function of the number of codewords for different trellis configurations with no additive Gaussian noise.

trellis configuration. This implies that the trellis structure has a great influence on how uniformly distributed the codewords are within each message coset (set of codewords encoding the same message).

So far, we have only considered the performance of a dirty paper trellis without any channel noise present. When additive Gaussian noise corrupts the watermarked Work, c_w , before it reaches the watermark detector, the performance of a dirty paper trellis will be worse than that reported in Figure 4.3. The effect of the channel noise, \mathbf{n} , on the performance depends on the power of the channel noise. For example, for a Watermark-to-Noise Ratio (WNR) of -7 dB, the performance of the dirty paper trellis is shown in Figure 4.6. As one can expect, the BER first decreases as the number of codewords, N_c , increases. This is intuitive, since the larger the codebook (i.e. N_c) is, the more likely it is that a codeword exists which is similar to the cover Work, c_o . Therefore this codeword can be embedded more strongly, i.e. a larger correlation value

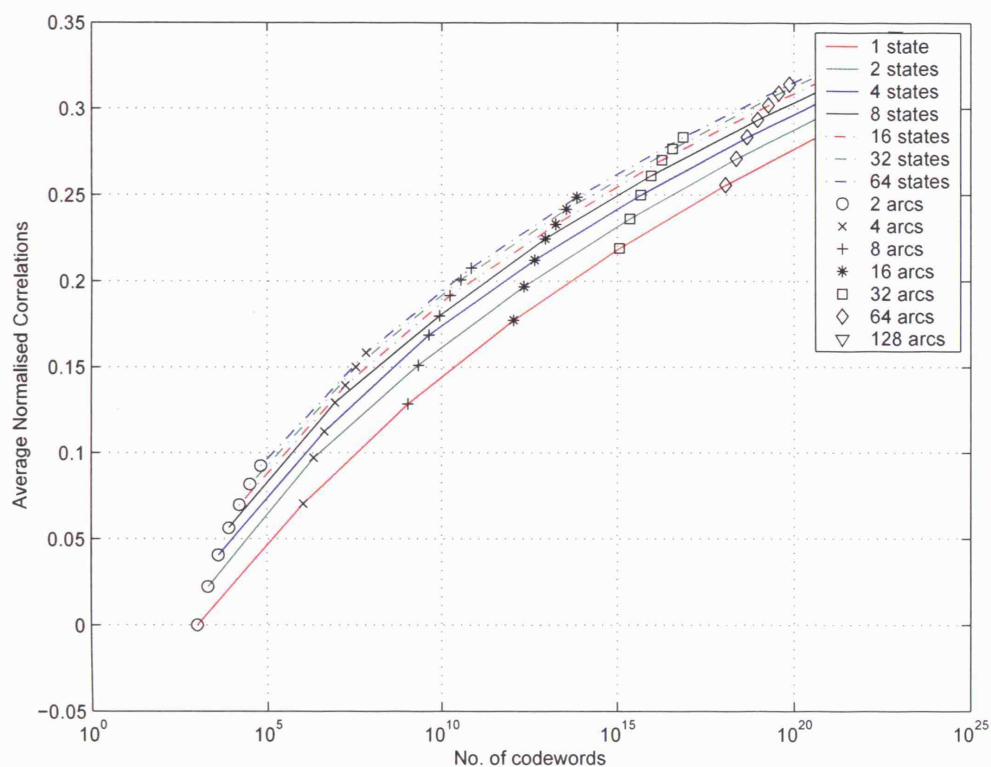


Figure 4.4: Average normalised correlation (between a cover Work and the closest codeword that codes the intended message) as a function of the number of codewords for different trellis configurations.

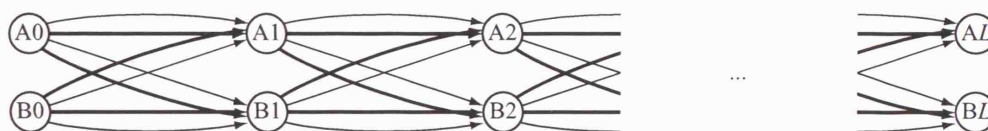


Figure 4.5: A dirty paper trellis with 2 states and 4 arcs per state.

will result, for a fixed Document-to-Watermark Ratio (DWR). However, because of noise, the BER increases after some point. This is because increasing the number of codewords, N_c , reduces the minimum distance between any pair of codewords, thereby shrinking the detection region of each codeword. As a result of a smaller minimum distance between a pair of codewords, the watermarked Work, c_w , starts to be more sensitive to channel noise.

One other thing to note is that different trellis structures have to be chosen to

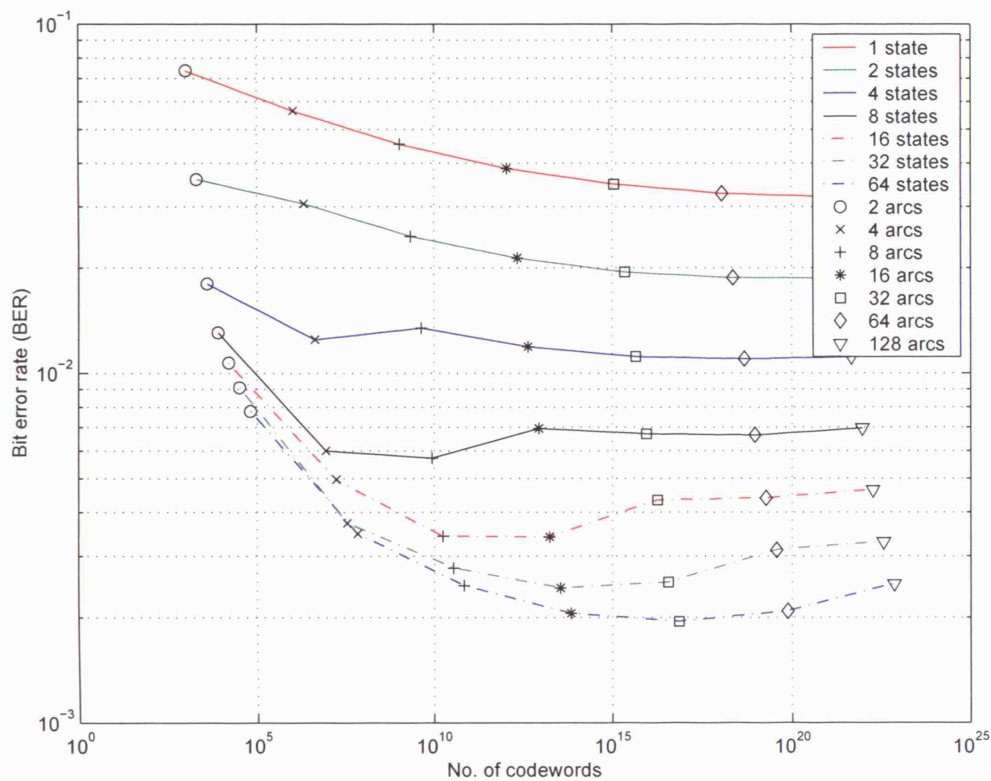


Figure 4.6: BER as a function of the number of codewords for different trellis configurations with -7 dB additive Gaussian noise.

operate under different noise conditions. Let us consider the noiseless situation and a noisy situation with $\text{WNR} = -7$ dB (see Figures 4.3 and 4.6 respectively). From Figure 4.3, trellis configurations $(S, A) = (16, 2), (2, 8), (1, 16)$ are chosen if the watermarking system is operating at a $\text{BER} = 3.3 \times 10^{-3}$. Under the same BER, trellis configurations $(S, A) = (64, 4), (32, 4), (16, 8)$ will be chosen if an additive Gaussian noise of $\text{WNR} = -7$ dB is present. Hence the optimum trellis structure depends on the operating conditions.

4.2 Trellis Computational Cost

In the previous example, we observed that for a particular operating condition, e.g. $\text{WNR} = -7$ dB, there were several trellis configurations that would provide a desired BER of 3.3×10^{-3} . Even if a trellis configuration is found to give good decoding performance, its computational complexity should also be taken into account to determine whether it can be used in practice. In the case of dirty paper trellis watermarking, the most costly operation is the Viterbi decoding of the whole trellis, i.e. during the water-

mark detection process. To carry out this operation, it is necessary to compute the cost function for every arc. Since linear correlation is used as the cost function in the set of experiments in Section 4.1, the Viterbi decoding operation requires N multiplications, $(N - 1)$ additions and *one* division for each arc. Since there are $A.S.L$ arcs in the trellis, the computational time is given by:

$$\tau = A.S.L(N\tau_{\times} + (N - 1)\tau_{+} + \tau_{/}) \quad (4.5)$$

where τ_{\times} (resp. τ_{+} and $\tau_{/}$) is the computational time of a multiplication (resp. addition and division). Assuming that these three values are roughly the same³, then the computation time is simply given by

$$\tau = 2.A.S.L.N. \quad (4.6)$$

Referring back to Figure 4.1, it means that configuration (a) has half the computational cost of configuration (b). There may be several alternative trellis configurations operating at a particular BER and the idea is to choose the one which is computationally the least. The relationship between BER and computation time should therefore be investigated.

Figure 4.7 shows the performance of different dirty paper trellises, in the absence of additive Gaussian noise, in terms of BER with respect to the computation time which is given by Equation 4.5. This shifts the previous curves (in Figure 4.3) horizontally so that configurations sharing the same total number $A.S.L$ of arcs are aligned vertically. The main point to note is that, whereas most configurations have very similar bit error rates for a given computational cost, a few specific trellis configurations appear to offer a significantly better compromise. In fact, when we look closely at Figure 4.7, it is clear that configurations with $A = S$ (fully connected trellis) or $A = S/2$ (half connected trellis) should be preferred. Indeed, for all the other configurations, an alternative structure can be found to offer a better trade-off between BER and computational complexity. It is worth noting that the trellis configuration retained in [MDC02, MDC04] (i.e. $S = A = 64$) is one of those few efficient trellis structures.

³In actual fact, the computation time for a multiplication, an addition, or a division differs, with the division operation much more time consuming than the multiplication operation, which in turn consumes slightly more time than the addition operation.

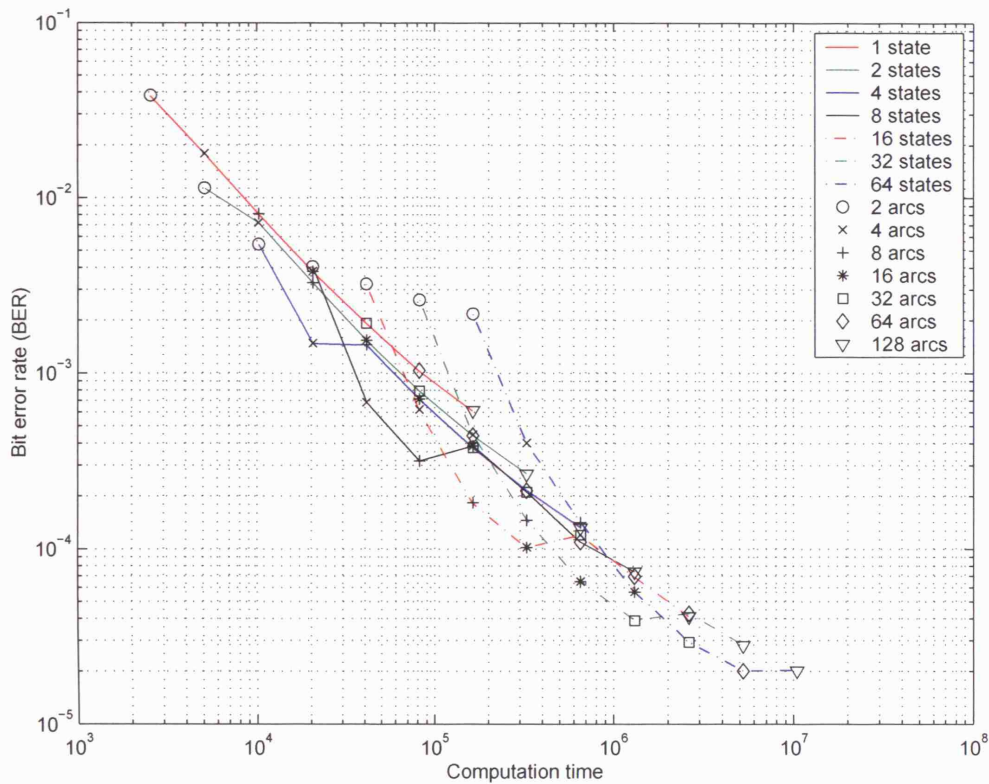


Figure 4.7: BER vs. computation time for different trellis structures in the absence of additive Gaussian noise.

Figure 4.8 shows another performance-computation time plot similar to Figure 4.7, except that, in this case, the computation time recorded is the actual average time taken during the watermark detection process executed using Intel Pentium Xeon Dual-Core 2.8 GHz. The similarity between these two plots enables us to use Equation 4.6 instead of recording the actual time taken during the watermark detection process.

Similar to Figure 4.7, Figure 4.9 shows the performance of different trellis structures, when additive Gaussian noise of $WNR = -7$ dB is added to the watermarked Work, c_w , before watermark detection. Under close observation of Figure 4.9, it is noted that certain trellis configurations are preferred to others. For instance, assuming that we want to operate a watermarking system at a bit error rate (BER) of about 3.3×10^{-3} , Figure 4.9 indicates that three alternative configurations can be used, namely $(S, A) = (16, 8)$, $(32, 4)$ or $(64, 4)$. Even though these configurations have similar performance in terms of BER, Figure 4.9 clearly shows that configuration $(64, 4)$ should

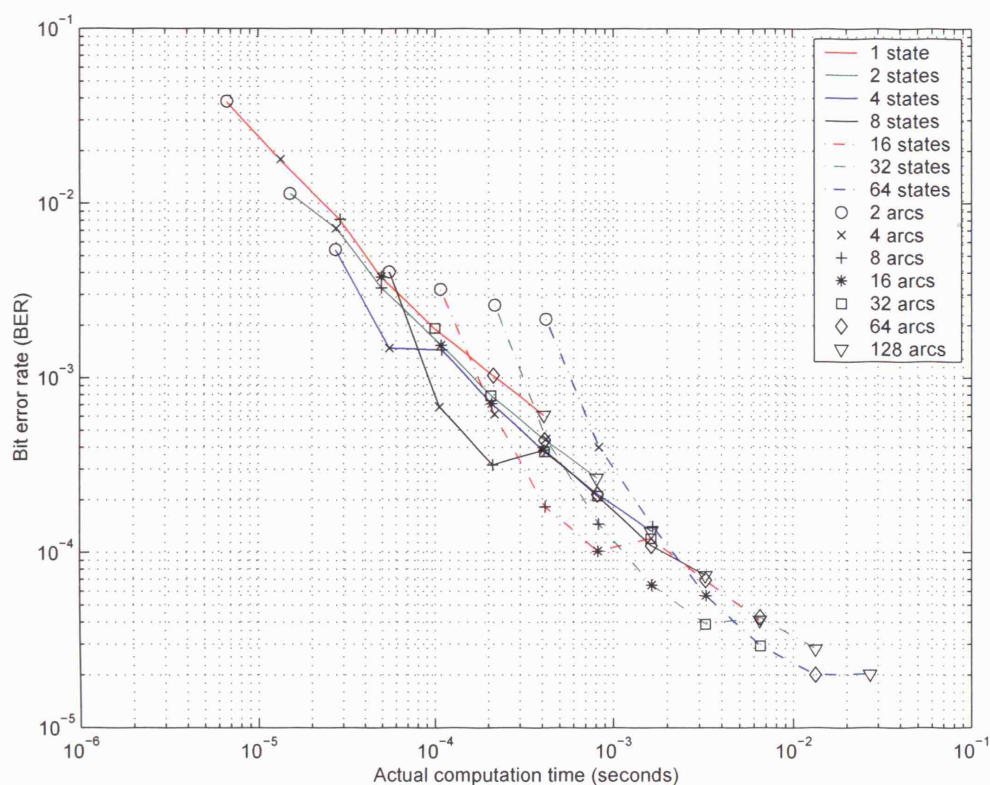


Figure 4.8: BER vs. actual computation time, in seconds, for different trellis structures in the absence of additive Gaussian noise.

be discarded because of its higher computational cost. In fact, looking closely at this figure, one can easily isolate the set of configurations which operates under the same BER and WNR.

4.3 Summary

The influence of the trellis structure on performance has been investigated. In this chapter, we have measured the BER and the computation time for many different configurations of a dirty paper trellis. The reported experiments have provided a better understanding of the impact of different trellis configurations. The following conclusions can be drawn:

- parallel arcs should be avoided in the trellis structure since they introduce single errors
- fully and half connected trellis structures (i.e. $A = S$ and $A = S/2$ respec-

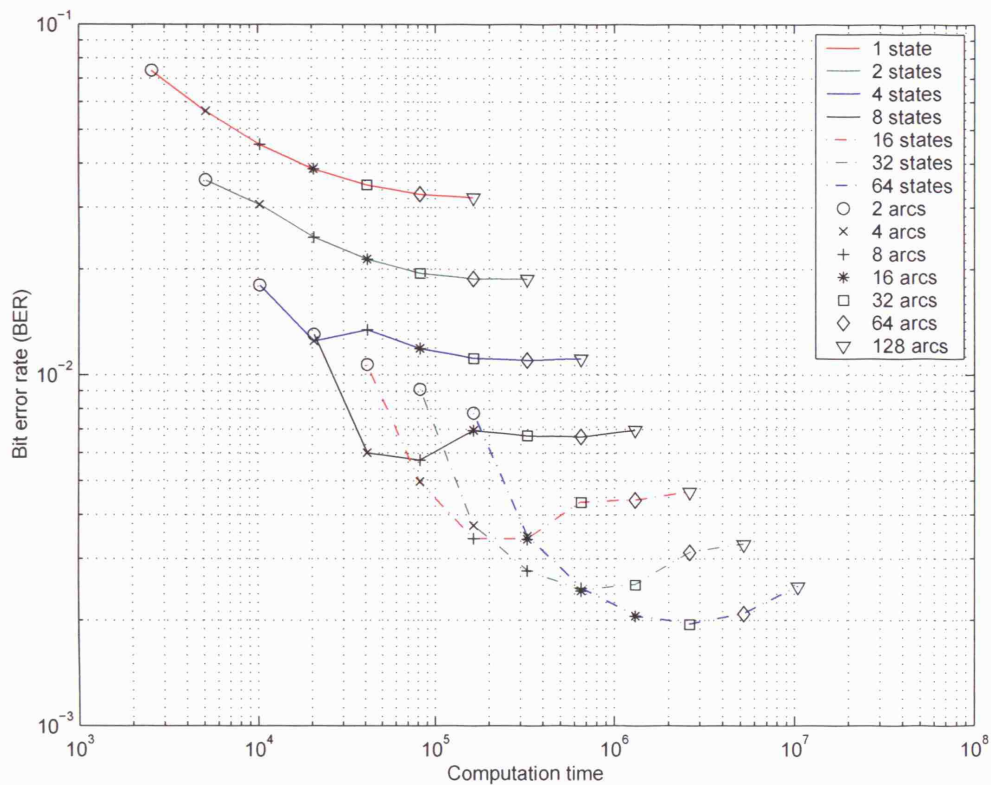


Figure 4.9: BER vs. computation time for different trellis structures with additive Gaussian noise of $WNR = -7$ dB.

tively) are preferred since they offer improved performance (BER) with respect to both the number of codewords and computational cost in a noiseless channel environment

- fully and half connected trellis structures do not guarantee an improved performance (BER) with respect to both the number of codewords and computational cost in a noisy channel environment
- optimum trellis configurations should be chosen, from a group of trellises which are operating at the same BER in a noisy channel environment, to be one having the least computational cost
- alternative trellis configurations having the same number of codewords, N_c , lead to very different BER
- alternative trellis configurations having the same BER may have very different

computational cost

Chapter 5

Using Trellis Coded Modulation to Improve Dirty Paper Trellis Codes

The performance of dirty paper trellis watermarking is strongly related to the distribution of the codewords over the surface of a hyper-dimensional sphere. Generally, a well-distributed set of codewords would ensure that (i) the codewords are robust to noise and that (ii) the distortion to the content (or cover Work) is minimised. In Chapter 4, we have seen how the number of states, S , and the number of arcs, A , influence the performance of dirty paper trellis and therefore affect the selection of a trellis configuration. In this chapter, we examine the following factors that affect the distribution of codewords and thus the trellis performance.

- the set of reference patterns used to label the arcs,
- the connectivity between the nodes, i.e. how the arcs join from one node to another,
- the mapping between the reference patterns and the arcs, and
- the mapping between the dirty bits (bold vs. non-bold) and the arcs

Currently, for dirty paper trellis codes, the codewords are randomly generated with each element from a Gaussian distribution and thus there is a high possibility that this set of codewords may not be uniformly distributed across the surface of the hyper-sphere. Also in the case for Angle QIM [OLJPG05] (refer to Section 2.8.3), the codewords are obtained by quantising a set of angles which defines the direction of any vector in the multidimensional space, e.g. the longitude and latitude in 3-D. This

leads to a suboptimal distribution of codewords (in 3-D) as there is a concentration of codewords at the poles. In contrast, with a good spherical code, the minimum distance between any pair of codewords should be maximised, thus leading to a uniform distribution of the codewords over the surface of the sphere [CS98]. Therefore in this chapter, assuming that the distribution of the codewords is non-uniform when dirty paper trellis code is used, the influence of the above-mentioned factors on the distribution of codewords is discussed so as to enhance the performance of dirty paper trellis codes.

To generate a better distribution of codewords, related work by Ungerboeck [Ung82] is useful as it enables the symbols associated with the arcs to be well separated using a concept called trellis coded modulation (TCM). Prior to the introduction of TCM, the channel coding stage and the modulation stage (refer to Section 2.1) are carried out individually. To improve the performance of trellis coding, Ungerboeck proposed TCM which combines these two stages. For a traditional trellis (depicted in Figure 3.3), the output symbols associated with each arc are binary. Although the output symbols of the arcs for a TCM trellis is also binary, there is a further mapping (i.e. modulation) carried out such that each output symbol corresponds to a signal in a modulation constellation, e.g. a point in the 8-PSK modulation. The concept of TCM is described in Section 5.1.

A TCM trellis is similar to traditional trellis in that both perform a one-to-one mapping between the input messages and the output codewords, i.e. one message corresponds to only one unique codeword. In order to use a TCM trellis for the purpose of dirty paper watermarking, a slight modification can be done to the TCM trellis so that the resulting TCM trellis enables a one-to-many mapping between input message bits and output codewords. The discussion on this modification is covered in Section 5.2.

The TCM trellis proposed by Ungerboeck [Ung82] deals with signals which are two dimensional, i.e. the signal point associated with each arc consists of 2 elements. Although it can still be applied for watermarking purposes, a higher signal dimension is very much preferred because, in general, performance and fidelity are improved. In particular, Pietrobon *et al* [PDL⁺90] provided an extension to Ungerboeck's work by using a higher dimensional arc label, thus favouring its use for dirty paper watermarking. A description of Pietrobon's work and its application to dirty paper watermarking are found in Section 5.2.

Having determined how to use TCM for dirty paper watermarking, Section 5.3 focuses on practical watermarking schemes using TCM. Firstly, in Section 5.3.1, vector quantisation (VQ) is used as a preliminary test to see how well TCM dirty paper trellis (TCM DPTC) and the original dirty paper trellis (random DPTC) perform. Since vector quantisation (VQ) can introduce a substantial distortions to the cover Work, c_o , another embedding scheme, which causes much lesser distortions but achieves a fixed robustness at the same time, is preferred. The iterative embedding algorithm mentioned in [MDC04] is suitable and is described in Section 5.3.2. Section 5.3.3 examines the use of this iterative embedding scheme for TCM DPTC and random DPTC and compares the performance between them.

5.1 Trellis Coded Modulation

Let us first ignore the *dirty paper nature* of a dirty paper trellis, i.e. the distinction between bold and non-bold arcs. We then have a trellis with several arcs entering/leaving each node. From a communications perspective, this means that more than one bit is mapped onto a reference pattern (i.e. signal point) per stage in a trellis. Figure 5.1 gives an example of a convolutional encoder that generates a trellis shown in Figure 5.2 with 4 states and 4 arcs leaving/entering each node. From Figures 5.1 and 5.2, the input consists of two bits at each time instance and the output is a three-bit symbol.

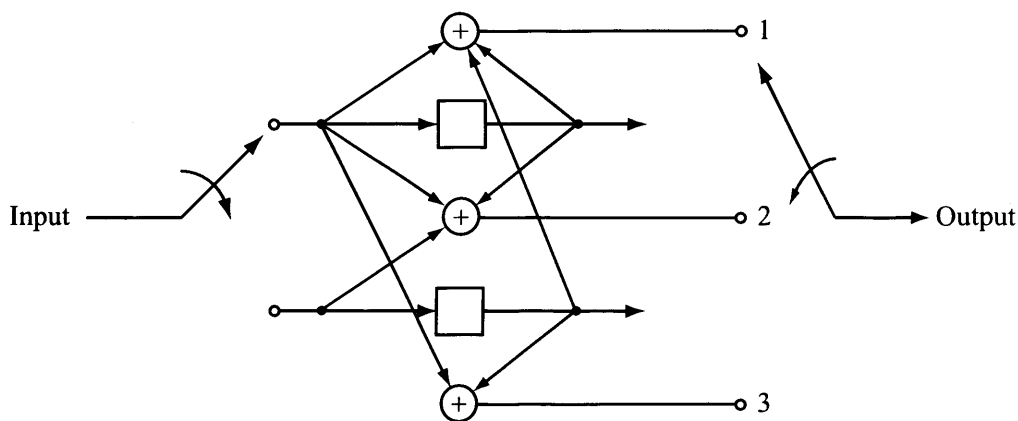


Figure 5.1: Convolutional encoder that takes 2 input bits at a time and produces 3 output bits.

In Section 2.1, we have seen that different coding schemes, e.g. linear block codes and convolutional codes, can bring about coding gains, i.e. performance improvement

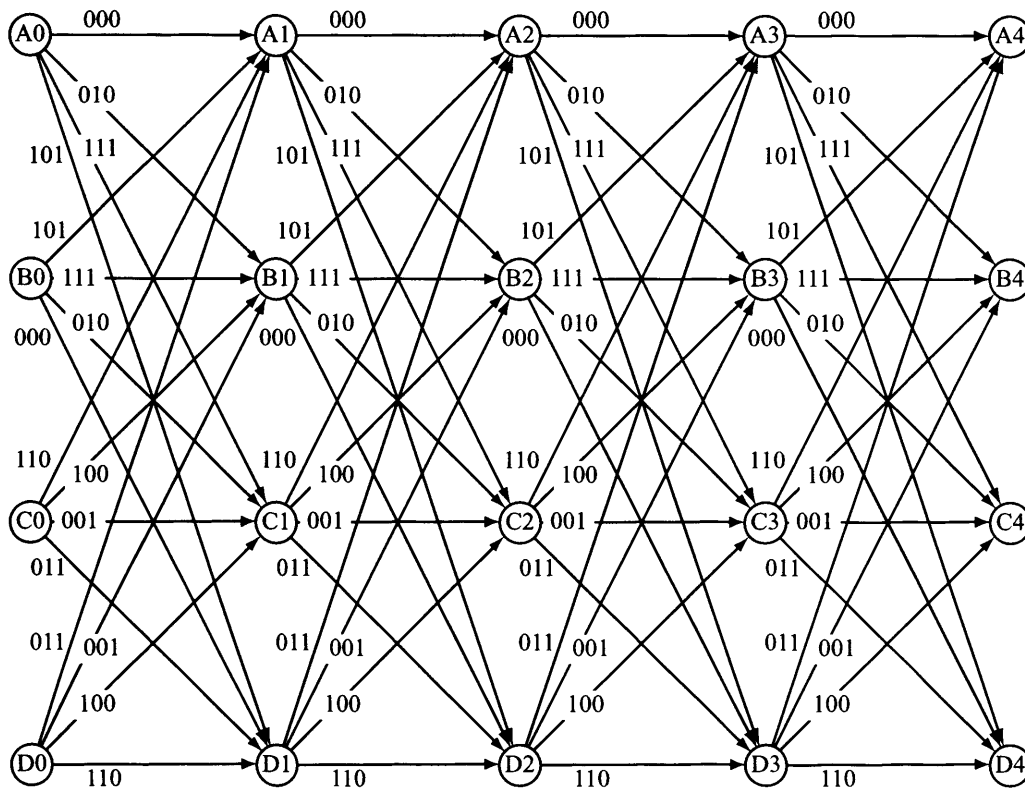


Figure 5.2: Trellis diagram for the convolutional encoder in Figure 5.1.

in terms of the difference in signal-to-noise ratio (SNR) required to achieve a specified error rate. However, the channel coding stage and the modulation stage (e.g. frequency shift keying (FSK), amplitude shift keying (ASK), and phase shift keying (PSK)) are usually carried out separately as shown in the general communications model of Figure 2.3. Ungerboeck [Ung82] proposed to combine these two stages into a single stage and showed that the performance of the resulting communication system can be improved. For example in [Ung82], Ungerboeck considered a transmission of 2 (information) bits per time interval and compared a coded 8-PSK modulation (with 8 symbols) and an uncoded 4-PSK (with 4 symbols). He showed that a theoretical gain of 7 dB can be achieved by using the former in an error-free transmission (assuming unlimited coding and decoding capability) over the latter with an additive white Gaussian noise channel at a symbol-error probability of 10^{-5} . Note that an error-free transmission for an uncoded 4-PSK requires an infinite SNR, hence a transmission with a hypothetical error rate is assumed. By increasing the number of symbols further (from 8 symbols onwards), only an additional gain of 1.2 dB can be achieved. Since other modulation

schemes revealed a similar gain in channel capacity, Ungerboeck concluded that most of the gain in channel capacity can be achieved by just doubling the number of symbols.

In order to achieve the gain in terms of channel capacity, Ungerboeck used trellis coding (i.e. convolutional coding) with the Viterbi decoding algorithm as the coding/decoding scheme. Assuming that, at time t , k bits are to be transmitted using an alphabet \mathcal{A} of 2^{k+1} symbols, a convolutional encoder with rate $R = k/(k+1)$ can be used to expand the k input bits and then map the resulting $(k+1)$ -long symbols into the symbol set of the alphabet \mathcal{A} . This entire framework, which was originally proposed by Ungerboeck [Ung82] and depicted in Figure 5.3, is commonly referred to as trellis coded modulation (TCM).

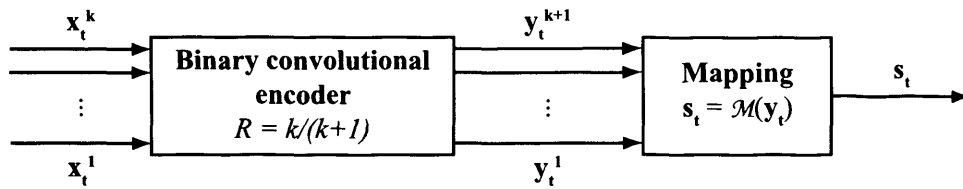


Figure 5.3: Trellis coded modulation (TCM): A binary convolutional encoder with rate $R = k/(k+1)$ is fed at time t with a k -bit binary word \mathbf{x}_t and a mapping function defines which symbol s_t to transmit depending on the $(k+1)$ -bit encoded output \mathbf{y}_t .

The transmitted sequence of symbols (i.e. codewords), $\{s_t\}$, corresponds to a path in the trellis. When this sequence $\{s_t\}$ is corrupted by additive white Gaussian noise (AWGN), the decoder may make wrong decisions by determining a path in the trellis that differs by at least one transition from the correct path, i.e. producing a decoded sequence of symbols $\{s'_t\}$. Such mistakes are called error-events and their occurrences are related to the free distance defined as

$$d_{\text{free}} = \min_{\{s_t\} \neq \{s'_t\}} \left[\sum_{t=1}^L \|s_t - s'_t\|^2 \right]^{\frac{1}{2}} \quad (5.1)$$

between all pairs of codewords $\{s_t\}$ and $\{s'_t\}$, whose paths start from the same state and end at the same state, that can be produced by the encoder. In other words, the free distance is defined as the minimum Euclidean distance separation between paths that diverge from the same state at the beginning of the trellis and remerge at the same state at the end of the trellis. The closer the pair of codewords, the more susceptible the code

is to errors. If *maximum likelihood* (ML) decoding, such as Viterbi decoding, is applied, the probability of such errors will asymptotically approach, at high signal-to-noise ratio (SNR), the lower bound

$$P_e \gtrsim N_{\text{free}} \cdot Q(d_{\text{free}}/2\sigma) \quad (5.2)$$

where N_{free} denotes the average number of error-events with free distance d_{free} and $Q(\cdot)$, or sometimes known as the Q-function, is the Gaussian error probability function defined as

$$Q(u) = \frac{1}{\sqrt{2\pi}} \int_u^{\infty} e^{-x^2} dx, u \geq 0 \quad (5.3)$$

Figure 5.4 shows how the Q function, $Q(u)$, varies with the variable u . From this figure, it is clear that the larger the value of u , the smaller the value of $Q(u)$. Hence from Equation 5.2, a large free distance, d_{free} , will give rise to a small value of $Q(u)$, which in turn will result in a smaller error probability P_e . Therefore the trellis should be designed in such a way that the free distance, d_{free} , is maximised since a larger d_{free} means a lower error probability.

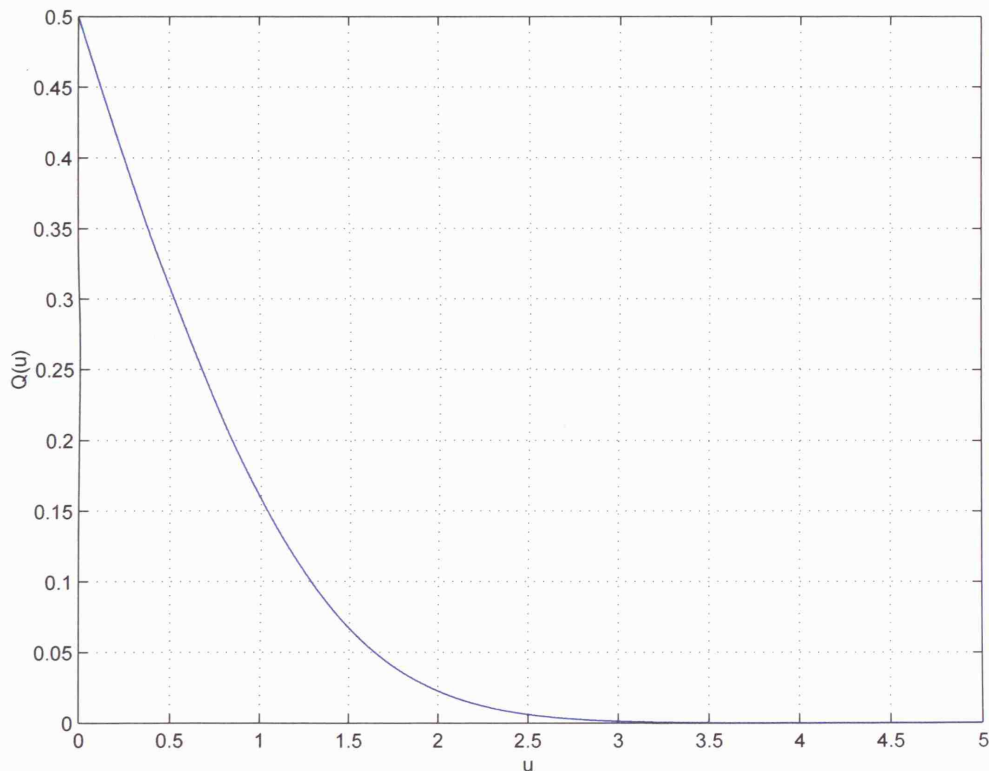


Figure 5.4: The Q-function, $Q(u)$, as a function of the variable u .

In order to achieve a larger free distance, d_{free} , Ungerboeck proposed a two-step optimisation process: “mapping by set partitioning” followed by searching for an appropriate binary convolutional encoder. Note that this system design is a reverse sequence of that in Figure 5.3. Let us first understand what “mapping by set partitioning” is. Consider an alphabet, \mathcal{A} , consisting of several symbols which may be transmitted by the encoder. One desired property is that these symbols are as separated as possible to avoid confusion at the receiver side. Furthermore, since the underlying goal is to obtain a code that is invariant to valuemetric scaling in digital watermarking, equi-energetic symbols, such as phase shift keying (PSK) symbols will be considered.

As an illustration, assume that an alphabet, \mathcal{A} , consists of 8PSK symbols, i.e. $|\mathcal{A}| = 2^{k+1} = 8$ symbols in the alphabet \mathcal{A} , where $k = 2$. Once the alphabet, \mathcal{A} , has been defined, TCM successively partitions the alphabet, \mathcal{A} , into subsets with increasing minimum subset distance $\Delta_1 < \Delta_2 < \dots$ between the symbols belonging to these subsets. Figure 5.5 provides an example of such a partitioning on the alphabet of 8PSK symbols. At each partition level, the considered set of symbols is further divided into two subsets so that the minimum distance within the subsets increases. At the end, the obtained subsets are arranged along a binary tree. Thus it is possible to label each branch of this tree with a $\log_2(|\mathcal{A}|)$ -bit label.

Now that the partitioning of the alphabet, \mathcal{A} , has been done to maximise the minimum subset distance, the next step is to search for a binary convolutional encoder which maximises the free distance d_{free} . The output of such a binary convolutional encoder, at time t , corresponds to a $\log_2(|\mathcal{A}|)$ -bit label, which is associated to one of the symbols involved during set partitioning. In other words, each of the arcs in the trellis has an associated $\log_2(|\mathcal{A}|)$ -bit label, determined by the convolutional encoder, which is then mapped to one of the $|\mathcal{A}|$ symbols. Having known how the arcs of the trellis are mapped to the symbols, this step is equivalent to enumerating all possible connectivity configurations and retaining the one configuration with the largest free distance, d_{free} , i.e. the maximum distance between two closest codewords with the same start and end states, as defined in Equation 5.1.

A convolutional code can be represented by a set of parity-check polynomials $[\mathbf{H}^{k+1}(D) \dots \mathbf{H}^1(D)]$. Any binary sequence \mathbf{y}_t output by the convolutional encoder

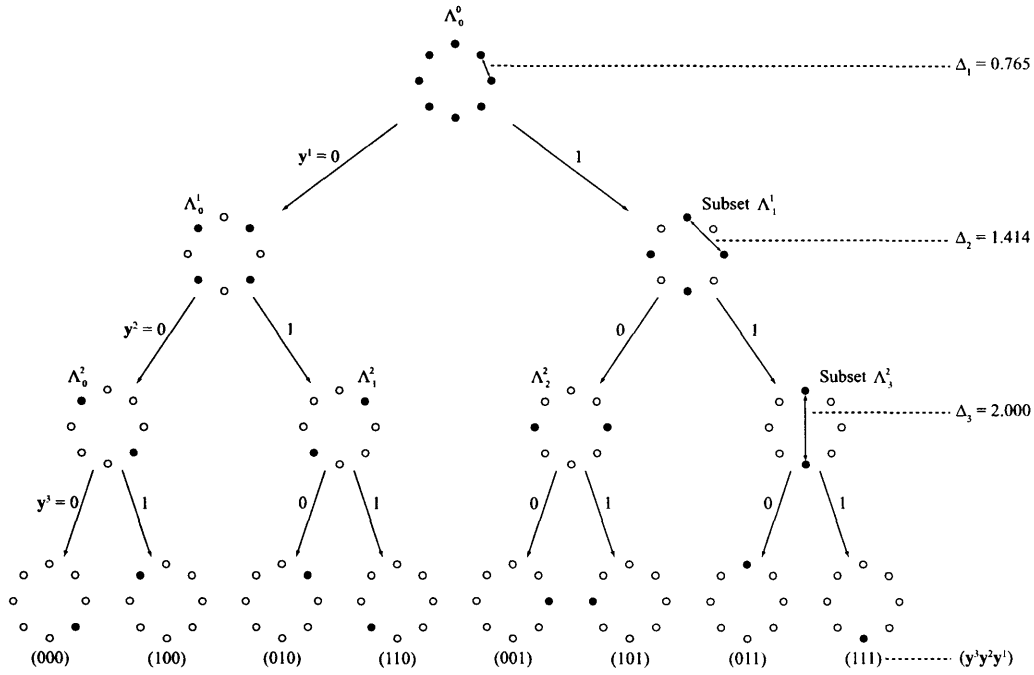


Figure 5.5: Partitioning of 8PSK symbols. At each partition level, the considered set of symbols is further divided into two subsets so that the minimum distance within each resulting subset is increased ($\Delta_1 < \Delta_2 < \Delta_3$). This results in a binary tree whose branches are labelled with a 3-bit label.

satisfies the parity check equation:

$$[\mathbf{y}_t^{k+1} \dots \mathbf{y}_t^1] \cdot [\mathbf{H}^{k+1}(D) \dots \mathbf{H}^1(D)]^T = 0 \quad (5.4)$$

where \mathbf{y}_t^i is the i -th bit of the output symbol \mathbf{y}_t at time t , and D is a time delay introduced by a delay element. Refer to Section 3.1.2 for a more detailed description of convolutional codes.

A straightforward way to perform the exhaustive search is to enumerate all possible parity-check polynomials [Ung82]. These polynomials have the following form:

$$\mathbf{H}^j(D) = \sum_{i=0}^{\nu} h_i^j D^i, \quad 1 \leq j \leq k+1 \quad (5.5)$$

$$\text{with } h_0^j = h_{\nu}^j = \begin{cases} 0, & j \neq 1 \\ 1, & j = 1 \end{cases}$$

where the h_i^j 's are binary coefficients and ν is the constraint length of code, i.e. the number of delay elements in the finite-state machine. The larger the value of ν , the

more *memory* the code has. The exhaustive search then reduces to enumerating all possible combinations of binary coefficients in order to identify the set of coefficients that gives the trellis structure with the highest free distance d_{free} . For example, Figure 5.6 shows a convolutional encoder with rate $2/3$ and constraint length $\nu = 3$ that gives the largest free distance, d_{free} , and the symbols used are from the alphabet of 8PSK symbols given in Figure 5.5. The input to this convolutional encoder is 2-bit long while its output, which selects one of the symbols from the alphabet of 8PSK, is 3-bit long. The resulting TCM trellis is shown in Figure 5.7 where the labels of the arcs leaving a node are listed (in the same order) on the left of the node, e.g. the arcs leaving node Λ_0^0 (from top to bottom) are having labels 0, 4, 2 and 6 (i.e. 000, 100, 010, 110) respectively. Upon closer observation, the arcs leaving each node belong to one of the two subsets, Λ_0^1 and Λ_1^1 , of Figure 5.5, and are independent of y_t^1 , i.e. the first bit of the 3-bit label $y_t^3 y_t^2 y_t^1$. The other two bits of the 3-bit label, i.e. y_t^2 and y_t^3 , determine which arc (out of 4) is selected. For instance, arcs leaving node $A0$ have $y_t^1 = 0$ and the arc with label 4 is selected if $y_t^3 = 1$ and $y_t^2 = 0$. It should be noted that the free distance for the trellis shown in Figure 5.7 can be computed using the pair of highlighted paths, both starting at state $A0$ and ending at state $A3$.

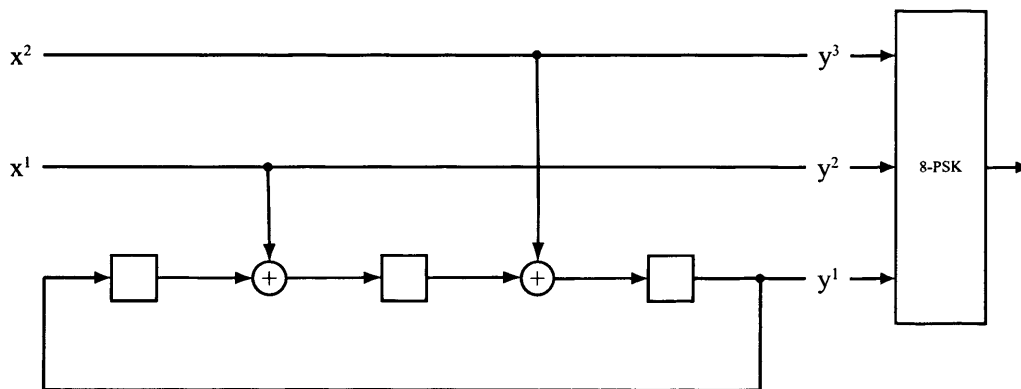


Figure 5.6: Binary convolutional encoder with rate $2/3$ and constraint length $\nu = 3$ giving the largest free distance d_{free} .

In general, if a convolutional encoder with rate $k/(k+1)$, but of any constraint length ν , is used for TCM, the symbols used for mapping will come from an alphabet of (2^{k+1}) -PSK symbols. Furthermore, the first bit, y_t^1 , of the $(k+1)$ -bit output label is the same for all arcs leaving a particular node and the remaining k bits select an arc (out

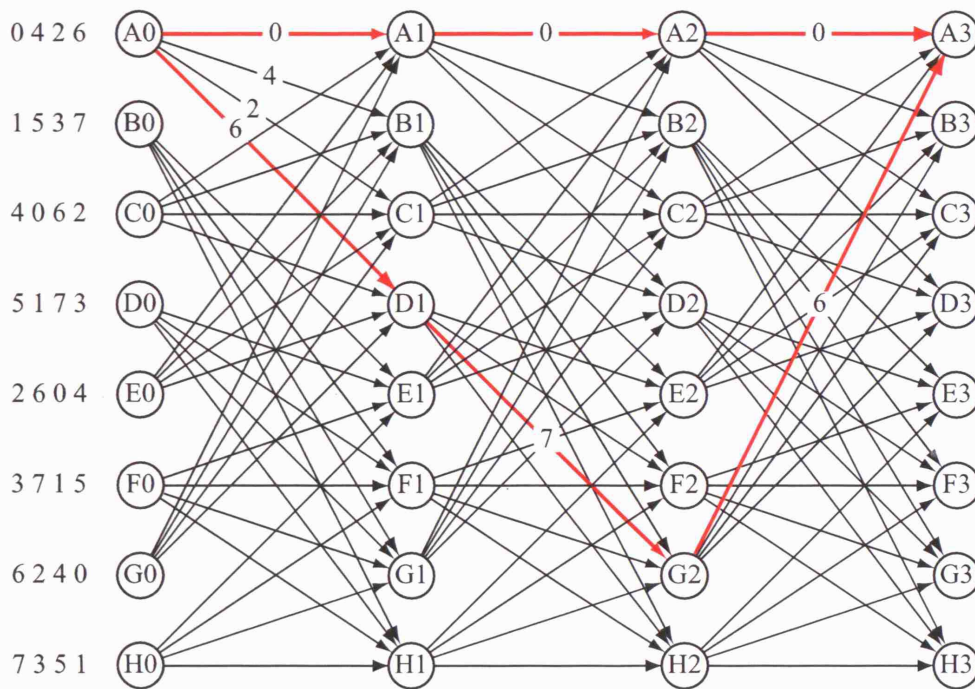


Figure 5.7: The 8-state TCM trellis, which is generated by the convolutional encoder in Figure 5.6, uses the alphabet of 8PSK symbols whose partitioning is shown in Figure 5.5.

of 2^k) leaving that particular node. In other words, regardless of the number of states of this TCM trellis, the number of arcs per state remain unchanged at 2^k and only one bit, i.e. y_t^1 , of the $(k + 1)$ -bit label has the same value for all the arcs leaving a node.

The entire optimisation process ensures that the codewords are as separated as possible. From [Ung82], it has been shown that by using trellis coded modulation (TCM), there is significant coding gains over conventional uncoded modulation. For example, by using trellis-coded 8PSK modulation, a coding gain of 3.0 dB to 5.7 dB (depending on the constraint length ν) can be achieved over conventional uncoded 4PSK. Note that this gain is less than the theoretical gain of 7 dB mentioned earlier because unlimited coding and decoding capability is assumed in the derivation of the theoretical gain. A larger constraint length ν not only gives rise to a larger coding gain (i.e. more powerful codes), but also increases the complexity of a Viterbi decoder exponentially. As such, constraint lengths of up to 9 are used in practical applications (see Section 8.2.8

in [Pro01]¹).

5.2 TCM Dirty Paper Trellis

In trellis coded modulation (TCM), the transmitted sequence of symbols, $\{s_t\}$, is designed to be as separated as possible. This sequence of symbols are the codewords or watermark patterns, w_m , used in the watermarking context. To use TCM in our dirty paper trellis framework, there is still one issue to be addressed: how to assign a *dirty bit* (bold vs. non-bold) to the arcs of the trellis? For fidelity reasons, it is desired that all the codewords encoding the same message, m , are well separated, i.e. all the paths through the trellis encoding the same message should be as separated as possible. Using Figures 5.5 and 5.7 as an illustration, the arcs leaving each node belong to one of the two subsets, either Λ_0^1 or Λ_1^1 , and therefore have the same first bit, i.e. $y_t^1 = 0$ (or 1) depending on which node the arcs leave from, in its 3-bit label $y_t^3 y_t^2 y_t^1$. The other two bits y_t^2 and y_t^3 in the label determine which symbol (or arc) is chosen from the corresponding subset, and hence any one of these bits can be selected as the dirty bit. The bit chosen as the dirty bit should ensure the resulting symbols are separated as far as possible. Since the partitioning of subset Λ_0^1 or Λ_1^1 in Figure 5.5 maximises the distance between the symbols in the resulting subsets, i.e. $\Lambda_0^2, \Lambda_1^2, \Lambda_2^2$, and Λ_3^2 , the second bit, i.e. y_t^2 , is selected as the dirty bit. In other words, if y_t^2 is chosen as the dirty bit, Figure 5.7 will undergo a slight change to produce the modified trellis as shown in Figure 5.8 and the resulting trellis can then be used for watermarking applications. In general, when using a set of coded MPSK symbols, where M denotes the number of symbols in the alphabet \mathcal{A} , the second bit of the $\log_2(|\mathcal{A}|)$ -bit label is selected as the dirty bit using the argument described earlier.

So far, we have only considered TCM trellises having arc labels which are 2-dimensional, i.e. each arc is associated with a 2-sample symbol. However, for digital watermarking, it is a common practice to use many more samples per arc, e.g. a few tens. In this perspective, previous work by Pietrobon *et al* [PDL⁺90] is of particular interest. They investigated the use of $\mu \times$ MPSK symbols, i.e. the concatenation of μ 2-D MPSK symbols, for TCM which results in an alphabet of M^μ symbols of dimension

¹Note that the definition of constraint length mentioned is slightly different from that used throughout this thesis.

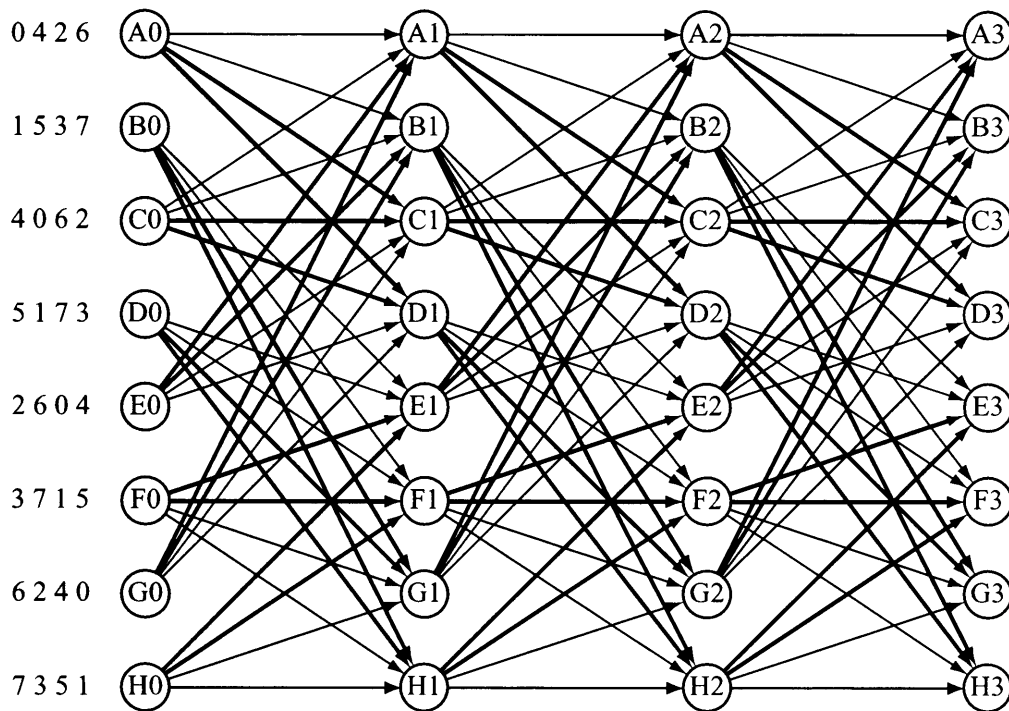


Figure 5.8: The dirty paper 8-state TCM trellis, which is generated by the convolutional encoder in Figure 5.6, uses the alphabet of 8PSK symbols whose partitioning is shown in Figure 5.5.

2μ . For security reasons, it may be required that the alphabet is not disclosed to the public, i.e. it is necessary to have a secret key, K , to have access to the alphabet. This can be easily done by applying a pseudo-random rotation to a generic known alphabet such as the one described in [PDL⁺90]. In some sense, it is similar to the dither term introduced in QIM schemes [CW01].

For this multi-dimensional TCM, the steps for searching for a TCM trellis with the maximum free distance, d_{free} , is the same as that mentioned by Ungerboeck, i.e. “mapping by set partitioning” followed by searching for an appropriate binary convolutional encoder. However, higher dimensional TCM might not be able to increase the minimum subset distance at each partition level during set partitioning. In such cases, the partitioning process should lead to a maximum reduction in the number of nearest neighbours within the smaller subsets. In practice, with $\mu \times$ MPSK symbols, it is possible to perform partitioning and mapping in a systematic manner by making use of block codes [PDL⁺90]. Since this partitioning process is complicated, it is out of the

context of this thesis and we will just use the results on mapping and trellis structure from [PDL⁺90]. Interested readers are encouraged to refer to [PDL⁺90]. By using this new multi-dimensional TCM, Pietrobon *et al* [PDL⁺90] have shown coding gains in the range of 1.76 dB to 5.33 dB (depending on the constraint length ν) can be achieved by using a coded 2×8 PSK modulation in 4-D.

For illustration of a higher dimensional TCM, let us consider a coded 2×8 PSK modulation in 4-D, i.e. 4 samples per label. In this case, the coded symbols come from a concatenation of two 8PSK symbols and hence there are 64 different symbols in all, i.e. $M^\mu = 8^2$. The coded symbols can be represented as a 6-bit label $y_t^6 y_t^5 y_t^4 y_t^3 y_t^2 y_t^1$. As explained earlier, the first bit, y_t^1 , is the same for all the arcs leaving a given node. To confirm that, in this case, the fidelity is best when using the second bit of the binary symbol, i.e. y_t^2 , output by the convolutional encoder, we conducted an experiment using synthetic signals with various bit of the 6-bit label as the *dirty bit*. Normally distributed random cover Works, c_o , of length $4L = 4000$, with each element having unit variance, were generated. Next, a dirty paper trellis, designed using TCM (called a TCM DPTC as opposed to random DPTC, which refers to the original dirty paper trellis), with 128 states and 32 arcs per state was generated using 2×8 PSK symbols based on Tables VII and XXIV reported in [PDL⁺90]. Using the relevant data from these tables, the binary convolutional encoder with the mapping block for the described TCM DPTC is shown in Figure 5.9. Note that the front portion of the 6-bit label, i.e. $y_t^6 y_t^5 y_t^4$, selects a symbol from an 8PSK constellation while the end portion of the label, i.e. $y_t^3 y_t^2 y_t^1$, corresponds to another symbol from the 8PSK constellation. Random messages of $L = 1000$ bits were then generated and the dirty paper trellis was constructed based on different bits of the 6-bit label used as the *dirty bit*. The path in the dirty paper trellis which encoded the message and had the highest linear correlation with the cover Work, c_o , using the Viterbi decoder correspond to the watermark pattern, w_m , that was eventually embedded using the direction quantisation (or vector quantisation) approach described as follows:

$$c_w = \frac{|c_o|}{|w_m|} w_m. \quad (5.6)$$

This embedding process is called vector quantisation because the cover Work, c_o , which is in the form of a vector, is “quantised” to the nearest codeword, i.e. the water-

mark pattern w_m , on the surface of the sphere.

The mean square error (MSE), defined as

$$MSE = \frac{|c_o - w_m|^2}{4L}, \quad (5.7)$$

is then computed. This experiment was repeated 10000 times to obtain a fairly accurate average value of the mean square error for different assignments of the *dirty* bit. Their mean square error values can be found in Table 5.1. From the table, it is clear that, in this case, if the *second* bit (i.e. y_t^2) of the 6-bit label by the convolutional encoder is chosen as the dirty paper label of each arc, the distortion is minimised, assuming vector quantisation is used as the embedding method. Therefore the second bit of the 6-bit label, y_t^2 , output by the convolutional encoder will be used as the *dirty* bit for TCM DPTC using $2 \times 8PSK$, having 128 states and 32 arcs per state.

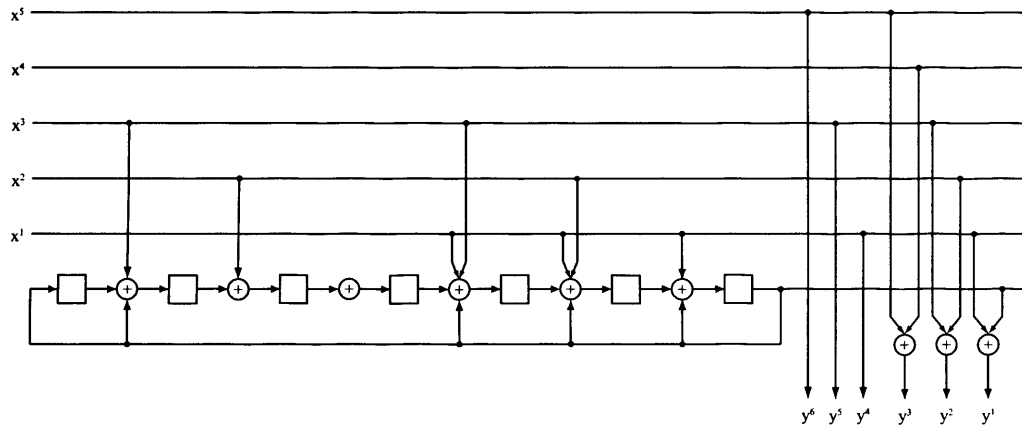


Figure 5.9: A convolutional encoder that generates a dirty paper TCM trellis having 128 states, 32 arcs per state, and $2 \times 8PSK$ symbols for every arc.

	2 nd bit (y_t^2)	3 rd bit (y_t^3)	4 th bit (y_t^4)	5 th bit (y_t^5)	6 th bit (y_t^6)
MSE	0.3651	0.3706	0.4228	0.4462	0.5745

Table 5.1: Average mean square error between cover Works, c_o , and watermarked Works, c_w , with different bits of the 6-bit label output by the convolutional encoder as the *dirty* bit. If the second bit of the 6-bit label is chosen as the *dirty* bit, the distortion caused during vector quantisation embedding is the minimum.

Before attempting to use TCM for dirty paper watermarking purposes, we need to

ensure that TCM performs better than a random dirty paper trellis. Since the performance of a trellis can be determined from its free distance (see Equation 5.2), the free distances of TCM and dirty paper trellis are compared. The free distance of a trellis is obtained by exhaustively searching the trellis for a pair of different codewords, i.e. paths, (having the same start and end states) which gives the smallest Euclidean distance between them. Note that the search for free distance disregards the dirty paper nature of the trellis, i.e. it does not matter what bit ('0' or '1') is associated to a particular arc. Due to the fact that dirty paper trellis is randomly generated, i.e. the symbols of the arcs are random, there is a need to obtain several free distance values since different generated dirty paper trellises lead to different free distances. In this case, 1000 free distance values are gathered so that a fairly accurate average can be produced. In order for fair comparison, the number of states, S , the number of arcs per state, A , and the number of samples per arc, N , for both TCM and dirty paper trellises are kept the same, e.g. at $S = 128$, $A = 32$ and $N = 4$. In particular, the TCM trellis is the one generated by the convolutional encoder in Figure 5.9. Upon completion of the search for the free distances, d_{free} , of TCM and dirty paper trellises, we found that $d_{free} = \sqrt{2} = 1.4142$ for TCM whereas $d_{free} = 0.2070$ (an average value) for dirty paper trellis. It can be deduced that, in the communications perspective, TCM performs much better than dirty paper trellis.

5.3 Watermarking Using TCM DPTC

From the previous section, we have seen that how a trellis, which is based on the design of trellis coded modulation (TCM), can be altered for use as a dirty paper trellis called TCM DPTC. Using a coded 2×8 PSK modulation, the second bit of the output of the convolutional encoder can be chosen as the *dirty* bit. Before jumping straight into any practical TCM DPTC application, we first consider a simple scenario so as to gauge the performance of TCM DPTC over random DPTC. In Section 5.3.1, a simple experimental setup involving synthetic signals and vector quantisation (VQ) as the embedding method is discussed.

Since vector quantisation (using dirty paper trellis) is unsuitable for practical watermarking due to its high distortion, an iterative embedding strategy, described in [MDC04], which provides a fixed robustness, is used instead. This iterative em-

bedding algorithm is described in Section 5.3.2 with Section 5.3.3 illustrating the use of this algorithm on TCM DPTC and random DPTC.

5.3.1 Vector Quantisation

To verify that the proposed TCM DPTC outperforms a randomly coded design, experiments were carried out with synthetic signals. Normally distributed random cover Works c_o of length $4L = 4000$, with unit variance, were generated. Next, two kinds of dirty paper trellises with 128 states, 32 arcs per state and 4 samples per arc were generated: either following the original design [MDC04], or following a TCM-guided design shown in Figure 5.9. Random messages of $L = 1000$ bits were then generated and embedded using the direction quantisation approach described in Equation 5.6. Subsequently, the watermarked cover c_w is corrupted by additive white Gaussian noise (AWGN) with variance σ_n^2 . The signal-to-noise ratio (SNR) is defined as $SNR = 10 \log_{10}(|c_w|^2 / 4L\sigma_n^2)$. The extracted message obtained by Viterbi decoding is finally compared to the message which has been actually embedded and the bit error rate (BER) is computed. Additionally, the Message Error Rate (MER) and the Path Error Rate (PER) are also computed. The MER relates to the probability of extracting *all* the bits of the message, whereas the PER indicates the probability that the detected path through the trellis is identical to the one chosen for embedding. This experiment is repeated for different SNR values and for each SNR, it is repeated 10000 times to obtain meaningful statistics.

The results are reported in Figures 5.10 and 5.11. When considering the Figure 5.10, it is difficult to make any statement in favour or against the TCM design. The two curves cross over which suggests that there is no best design, i.e. alternative designs have to be chosen depending on the SNR regime. This is a bit surprising with respect to previous results in digital communications. However, it should be kept in mind that the BER is computed by considering only a single bit per arc, i.e. the *dirty bit* instead of 6 bits from its label $(y_t^6 y_t^5 y_t^4 y_t^3 y_t^2 y_t^1)$. On the other hand, Figure 5.11 clearly shows that the TCM design always outperforms its random counterpart. For any SNR regime, there is less chance to deviate from the correct path with a TCM DPTC than with a random one. This is because the free distance, d_{free} , is significantly higher for the TCM design, as seen in the Section 5.1. This was the whole purpose behind the construction

of these codes.

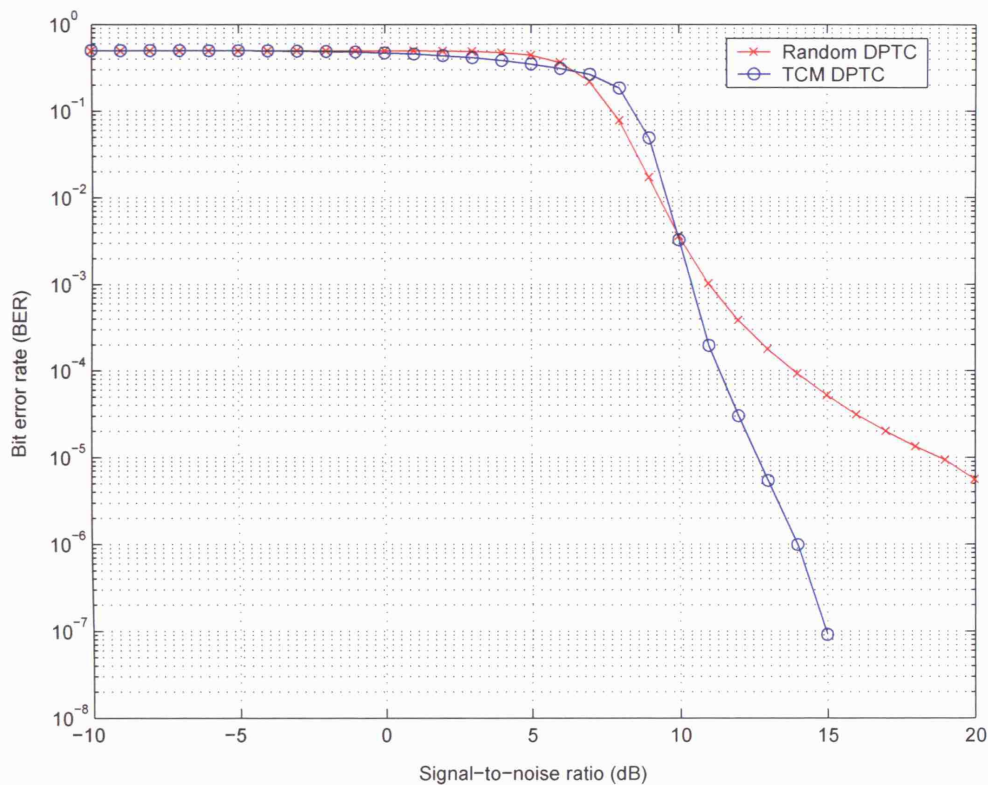


Figure 5.10: BER performance comparison between random DPTC and TCM DPTC with synthetic signals and direction quantisation embedding.

5.3.2 Iterative Embedding Algorithm

The performance of TCM DPTC using synthetic signals, with vector quantisation (VQ) as the embedding method, suggests that TCM DPTC is to be preferred. Hence we are prepared to employ TCM DPTC in practical watermarking scenarios. However, vector quantisation can cause large distortions if it is chosen as the embedding method. An alternative to vector quantisation is the iterative embedding algorithm described in [MDC04]. Iterative embedding is similar to vector quantisation (VQ) except that the embedding process determines a point within the detection region containing the selected watermark pattern, w_m , as the watermarked Work, c_w , so as to provide a certain robustness to additive white Gaussian noise (AWGN). This way, not only the watermarked Work c_w is resistant to a certain amount of Gaussian noise depending on how robust the user wants it to be, the distortion caused to the original cover Work

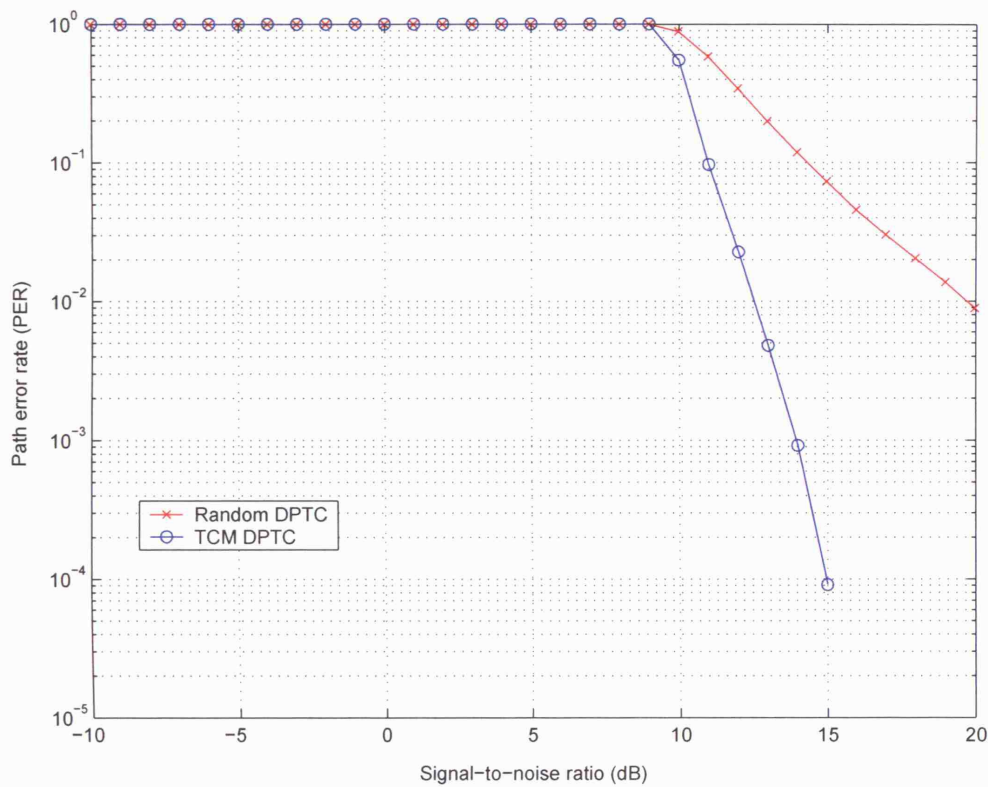


Figure 5.11: PER performance comparison between random DPTC and TCM DPTC with synthetic signals and direction quantisation embedding.

is reduced considerably compared to vector quantisation (VQ). This subsection first describes the iterative embedding algorithm used in [MDC04] before applying it to TCM DPTC in Section 5.3.3.

To understand how iterative embedding [MDC04] works, let us consider a simple system with only two possible messages, each represented by a different vector with \mathbf{g} as the “good” vector (i.e. the watermark pattern \mathbf{w}_m) and \mathbf{b} as the “bad” vector. \mathbf{g} is the vector to be embedded into the cover Work \mathbf{c}_o whereas \mathbf{b} is the vector that is hopefully not detected from the watermarked Work \mathbf{c}_w . The detector returns the message associated with \mathbf{g} if $\mathbf{g} \cdot \mathbf{c}_w > \mathbf{b} \cdot \mathbf{c}_w$, where $\mathbf{g} \cdot \mathbf{c}_w = \sum_i \mathbf{g}[i] \mathbf{c}_w[i]$ is the correlation between \mathbf{g} and \mathbf{c}_w .

Assuming that the distortions applied to a watermark Work \mathbf{c}_w are modelled as additive white Gaussian noise (AWGN)², \mathbf{n} , which is a length $L \times N$ vector whose ele-

²Refer to [CMB01] for a justification of this assumption.

ments are drawn independently from a Gaussian distribution of mean zero and variance σ_n^2 , i.e. $\mathcal{N}(0, \sigma_n^2)$, the detector will receive $c_{wn} = c_w + n$. In this case, in order for correct detection, the condition $g \cdot c_{wn} > b \cdot c_{wn}$ has to be met.

Before plunging straight into the iterative embedding algorithm, consider first the geometric interpretation of the embedding region, given the good vector g (i.e. watermark pattern w_m) to be embedded, as illustrated in Figure 5.12. This figure shows a Voronoi diagram representing six detection regions for different vectors indicated with 'X'. If the watermarked Work, c_w , is to resist a certain minimum noise of variance σ_n^2 , c_w has to be a certain distance away from the edge of the detection region for g , i.e. indicated by the boundary of the gray region. The gray region indicates the set of images that has a robustness σ_{max} , i.e. resistance to noise of variance at least σ_{max}^2 . The solid circle indicates an original cover Work c_o , and the triangle shows the closest possible watermarked Work c_w that has an acceptable robustness. The ideal embedder moves c_o straight to the closest point in the acceptable embedding region, to produce a watermarked Work c_w .

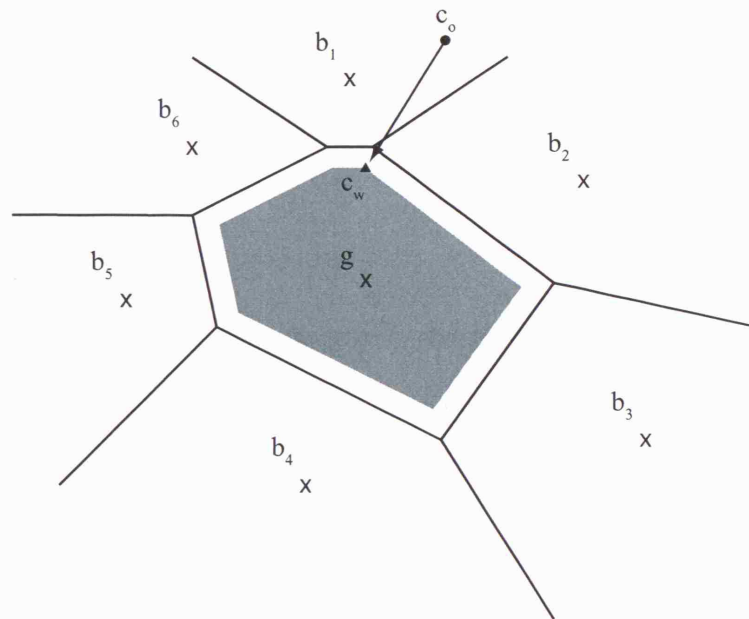


Figure 5.12: Geometric interpretation of the embedding region defined by specifying a fixed robustness σ_{max} .

However, it is difficult to implement such an algorithm to find the optimal wa-

termarked Work, c_w , as illustrated in Figure 5.12 since the detection boundaries are not easily determined. Therefore Miller *et al.* [MDC04] proposed a sub-optimal iterative algorithm mentioned in [MDC04]. Assume that there is a black-box watermark encoder, $W(m)$, that maps a given message into a watermark signal, g , and also a black-box watermark detector, $D(c_w)$, that maps a given image, c_w , into another watermark signal with which the image has the highest correlation. Given a cover Work, c_o , a message, m , to embed, and a target robustness σ_{max} , the complete, general version of the iterative embedding algorithm (which is a form of informed embedding since the cover Work c_o is used during the production of the watermarked Work c_w) is as follows:

1. Let $g = W(m)$, $c_w = c_o$, $\sigma_n = 0$, and $j = 0$.
2. Let $b = D(c_w + n)$, where n is a random vector with each element drawn independently from a Gaussian distribution with mean zero and variance σ_n^2 .
3. If $b \neq g$, modify c_w as follows:

$$d = \frac{g - b}{|g - b|} \quad (5.8)$$

$$\alpha = d \cdot (c_w + n) \quad (5.9)$$

$$c_w \leftarrow c_w - \alpha d \quad (5.10)$$

Then reset j to 0 and go back to *Step 2*.

4. If $b = g$ and $j < 100$, then increment j and return to *Step 2*.
5. If $b = g$ and $j = 100$, then reset j to 0. If $\sigma_n < \sigma_{max}$, increase σ_n by δ and return to *Step 2*. Otherwise, terminate.

The sequence of this algorithm can be shown geometrically in Figure 5.13. Before starting the iterative process, the standard deviation, σ_n , of every element in the noise vector, n , is initialised to zero. In the first iteration, the watermarked Work, c_w , is assigned the cover Work, c_o , which lies in the detection region of b_1 and hence the bad vector $b = b_1$ in *Step 2*. The watermarked Work, c_w , is then moved to the detection boundary between g and b_1 . At the end of the first iteration, c_w lies in the detection region of b_2 . Then in the second iteration, $b = b_2$ and c_w is moved to the detection

boundary between \mathbf{g} and \mathbf{b}_2 . Now that the watermarked Work, \mathbf{c}_w , can be detected to possess the good vector \mathbf{g} , but it is still susceptible to any noise that may be present during the transmission of the watermarked Work \mathbf{c}_w . Hence in the next step, the standard deviation, σ_n , of the noise vector, \mathbf{n} , is incremented by a pre-assigned increment, δ , and this noise vector, \mathbf{n} , is added to \mathbf{c}_w to produce a corrupted version of the watermarked Work \mathbf{c}_{wn} . If \mathbf{c}_{wn} is still within the good detection region (i.e. in the detection region of \mathbf{g}), then increment the counter j to indicate that it has attained a robustness to noise with a standard deviation of σ_n . However, being able to withstand that amount of noise once does not mean that \mathbf{c}_w has already achieved that level of robustness. Therefore \mathbf{c}_w has to be tested by several noise vectors, \mathbf{n} , in order to be certain that it can withstand that amount of random noise in most situations. If \mathbf{c}_{wn} is outside of the good detection region as denoted by a solid square in Figure 5.13, then \mathbf{c}_w has to move further into the good detection region by the distance in which \mathbf{c}_{wn} is away from the boundary between \mathbf{g} and \mathbf{b}_2 . The counter j is then reset to 0 and the iterative process of testing \mathbf{c}_{wn} with various noise vectors, \mathbf{n} , continues. The algorithm terminates when σ_n reaches σ_{max} and the counter j reaches 100, i.e. the final watermarked Work, \mathbf{c}_w , is very certain to withstand an additive white Gaussian noise (AWGN) with a standard deviation of σ_{max} for each element in the noise vector \mathbf{n} .

Now that the iterative embedding algorithm is described and the working principles explained, we can then go on to use this algorithm with TCM DPTC in Section 5.3.3.

5.3.3 Fixed Robustness Embedding

To evaluate the performance of TCM DPTC under more realistic conditions, another round of experiments with a database of 1000 grayscale images of dimension 240×368 has been considered. First, a feature vector is extracted from each image to serve as the cover Work \mathbf{c}_o . This is done by computing the 8×8 block discrete cosine transform (DCT) and extracting 4 middle-frequency AC coefficients, as shown in Figure 5.14, from each of the $L = 1380$ DCT blocks in the image. Middle frequency AC coefficients are chosen so that they are not vulnerable to signal processes such as lowpass filtering, i.e. attenuating high frequency components of a content, and lossy compression, i.e. eliminating perceptually non-salient components of an image or audio. The

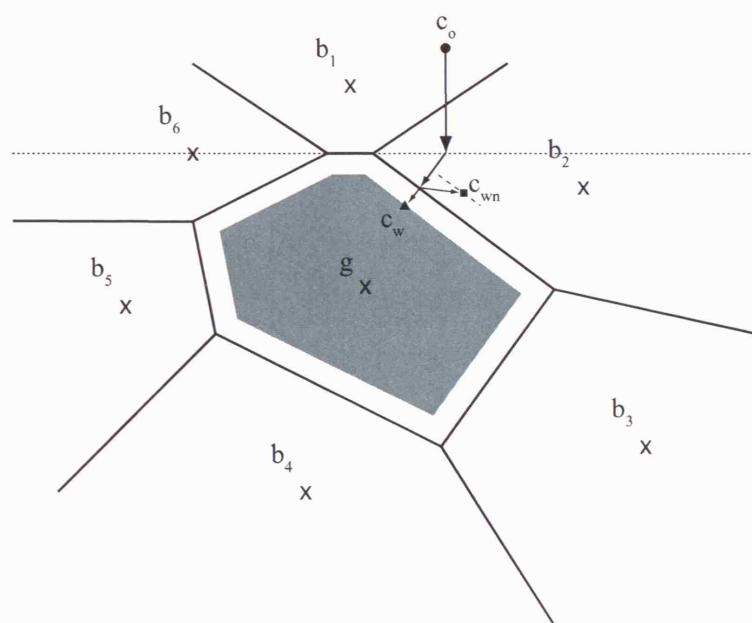


Figure 5.13: Geometric interpretation of the sub-optimal, iterative algorithm for informed embedding.

extracted vector of length $4L$ is then pseudo-randomly shuffled according to the secret key K to obtain the cover Work, c_o , which will subsequently be considered for watermarking. Once again, two kinds of dirty paper trellises with 128 states and 32 arcs per state are considered: random DPTC on one hand and TCM-DPTC using 2×8 PSK symbols [PDL⁺90] on the other hand. Subsequently, a L -bit long random message is generated and the codeword, w_m , which is obtained by using the modified trellis as described in Section 3.3, is embedded. In practice, the crude quantisation embedding technique given in Equation 5.6 introduces strong visual distortion. Therefore, the iterative embedding algorithm described in 5.3.2 has been preferred. Instead of quantising the cover Work, c_o , to the centre of the desired detection region, the algorithm basically brings the Work inside this region by a margin set by a specified robustness parameter. This parameter can be adjusted to control the distortion induced by the watermarking process. In our experiments, this distortion was kept fixed as measured by the mean square error (MSE). The statistics of the observed MSE are given in Table 5.2. The distortion has been kept on average around 0.83. However, more importantly, it should be noted that the variance is significantly lower with TCM DPTC than with

random DPTC. Once again, this confirms that TCM induces a better distribution of the codewords on the surface of the hypersphere.

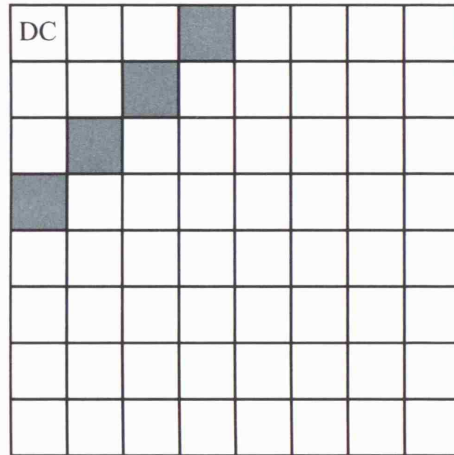


Figure 5.14: The 4 middle-frequency DCT coefficients (shaded in gray) used for embedding. The top-left corner DCT coefficient is the DC term.

	Mean	Variance
Random DPTC	0.8305	0.3083
TCM-DPTC	0.8295	0.1958

Table 5.2: Statistics of the MSE with both codes under study.

The watermarked images, c_w , are then degraded by additive white Gaussian noise with standard deviation σ . During the watermark detection process, a feature vector, c_{wn} , is extracted from each corrupted image. Next, the Viterbi decoder is fed with c_{wn} and several results are stored, e.g. the number of error bits between the embedded and the extracted message, the occurrence of a message error, and the occurrence of a path error. These values are then averaged for all images to estimate the BER, MER and PER for a given standard deviation σ . Repeating this process for different values of σ , it is then possible to produce the Figures 5.15 and 5.16. Once again, by looking at Figure 5.15, the comparison between the two designs is not easy. Although, TCM always outperforms its random counterpart, the distance between the two curves is fluctuating, thus giving an unclear conclusive statement. However, one should keep in mind that with DPTC, only one message bit is associated with each arc compared to several coded

bits per arc in the actual communications case. Therefore what is important is whether all the bits of the embedded message have been correctly retrieved. When looking at Figure 5.16, it is clear that the TCM design achieves greater robustness compared to the random one. With comparable embedding distortion, 80% (as suggested in [KP99]) of the TCM-DPTC watermarks survive noise addition up to a standard deviation of $\sigma = 1.5$, whereas random DPTC watermarks are only able to cope with noise addition up to $\sigma = 1$. In other words, for a given distortion budget, it is possible to go further into the detection region with TCM DPTC as compared to random DPTC. This is a direct consequence of a better distribution of the codewords.

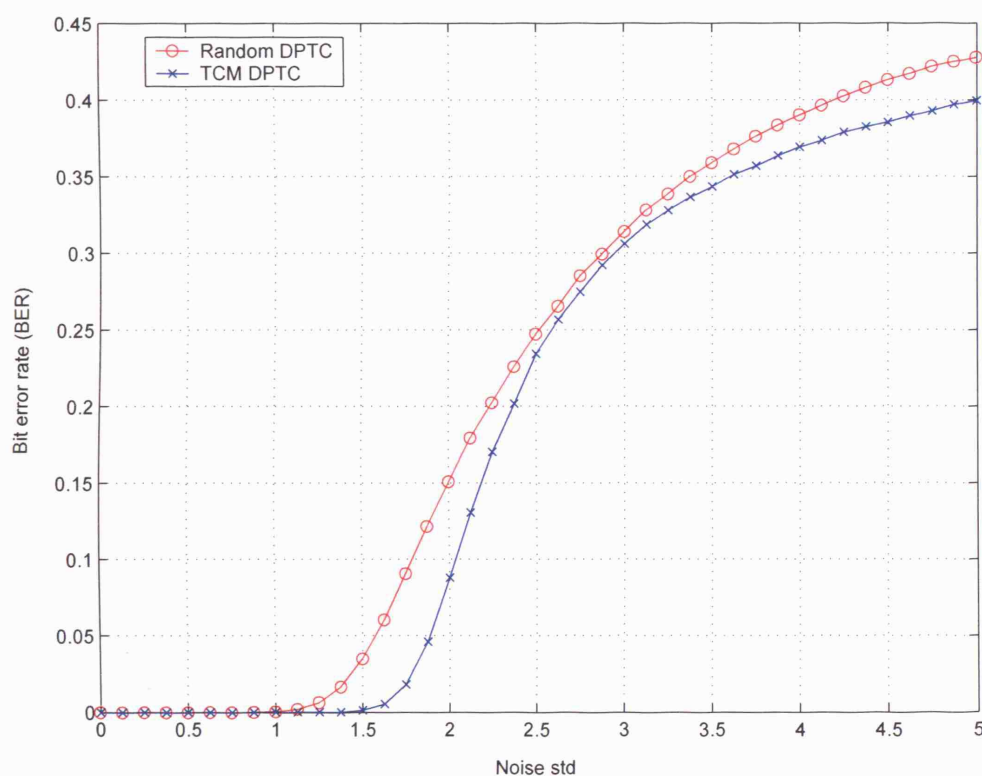


Figure 5.15: BER performance between random DPTC and TCM DPTC with real images and iterative embedding .

5.4 Summary

Dirty paper trellis codes (DPTC) are a form of spherical codes. In previous work, a practical implementation using DPTC has been proposed to generate such a code in a high dimensional space [MDC04]. However, the original design was random,

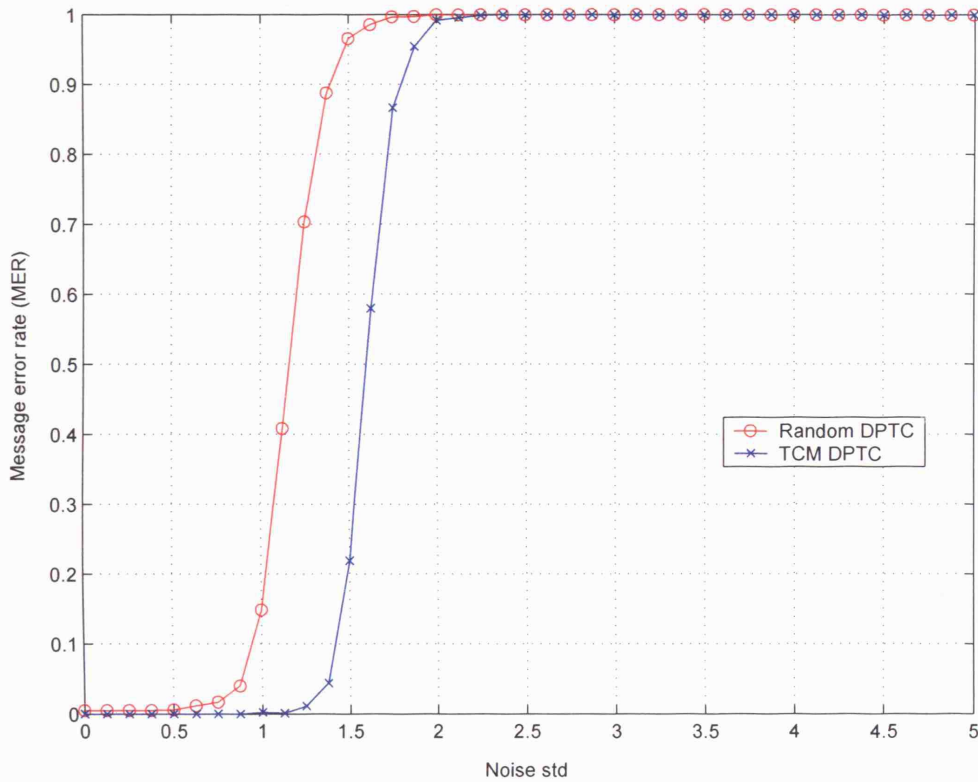


Figure 5.16: MER performance between random DPTC and TCM DPTC with real images and iterative embedding .

thus possibly leading to a suboptimal distribution of the codewords on the unit sphere. Ideally, the codewords should be well separated to optimise the fidelity-robustness watermarking trade-off. In this chapter, relevant work on trellis coded modulation (TCM) [Ung82, PDL⁺90] has been revisited to produce a better DPTC. In our proposed TCM DPTC design, each arc was labelled with a concatenation of 2-D MPSK signals as suggested in [PDL⁺90]. Experimental results have validated that codewords generated by TCM DPTC are more regularly distributed on the unit sphere and consequently ensure higher performance in terms of MER compared to random DPTC. Nevertheless, it is not clear whether this approach leads to an optimal distribution of the codewords. On the other hand, designing *good* spherical codes is also critical when suboptimal embedding techniques are used. For instance in [LCD05], the fixed robustness embedding region is approximated by the same hyperbola for all detection regions even though these regions are not equally shaped. In other words, the hyper-

bolic embedding region will not guarantee a fixed robustness and this is not desirable. Furthermore, as mentioned in the original paper [Ung82], TCM can be used with binary symbols instead of spherical symbols. Recent studies have reported that such an approach can significantly improve the performances of traditional quantisation-based watermarking schemes [ABPGM05].

Chapter 6

Using Perceptual Distance to Improve the Selection of Dirty Paper Trellis Codes for Watermarking

As mentioned in Section 2.3, the watermark embedding process consists of two stages: message coding and modification. During the message coding stage, the watermark pattern, w_m , chosen depends on the *cost function*, i.e. the cost of choosing an arc. So far, the cost function considered in Chapters 4 and 5 is the linear correlation between the cover Work c_o and the reference patterns, w_r , associated with the arcs regardless of the embedding strategy used in the modification stage. It is logical to choose the reference pattern with the highest linear correlation¹ with the cover Work c_o since less distortion, in terms of Euclidean distance, is expected after watermark embedding.

In certain scenarios, the watermarked Work, c_w , has to be within a specified fidelity, i.e. the distortion between the cover Work, c_o , and the watermarked Work, c_w , has to be within a specified limit. What are the ways to measure the distortion between two images? The most common way is to compute the mean square error (MSE) between them as has been seen in Chapter 5. However, mean square error is known to be a poor way to measure perceptual distortion [Gir93]. To illustrate this, consider Figure 6.1. In this figure, the original image is distorted by two different processes, namely an addition of white noise and an addition of lowpass filtered noise. The computed MSE for the first and second distorted images are *16.18* and *16.11* respectively. Although the

¹Having the highest linear correlation is equivalent to saying having the smallest Euclidean distance.

resulting distorted images show similar distortions as measured by MSE, the human eye does not perceive the same distortion.

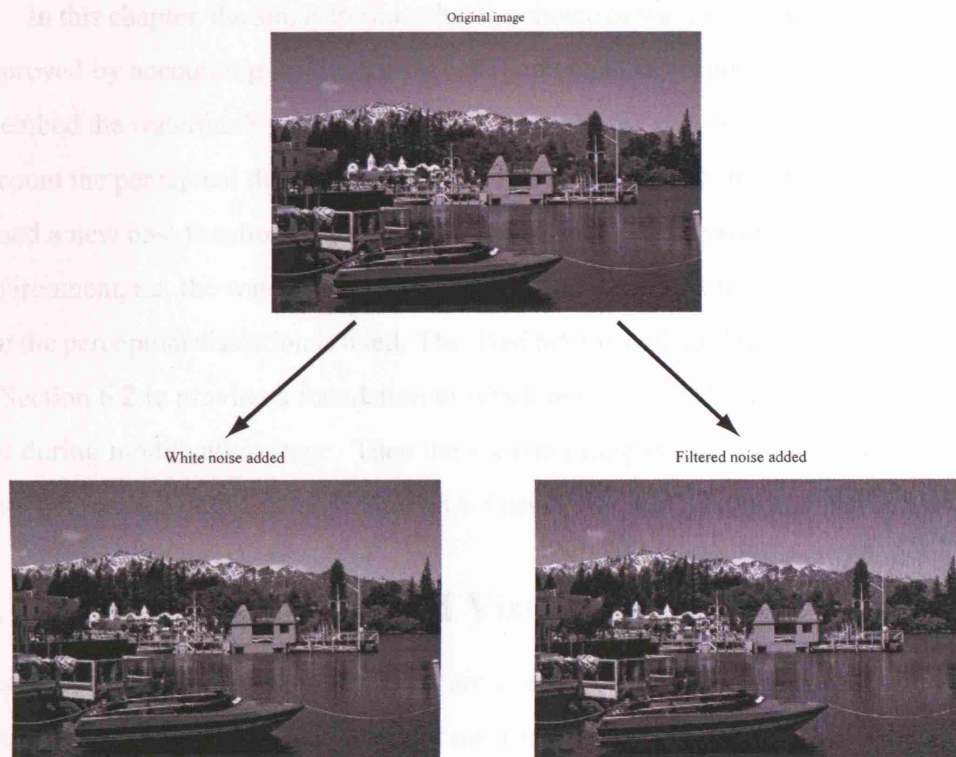


Figure 6.1: Illustration of a case in which mean square error (MSE) is a poor measure of perceptual distortion.

Since MSE is not a good measure of perceptual distortion, better perceptual models had been designed to follow the response of the human visual system (HVS). One such perceptual model is Watson's DCT-based visual model, which estimates the perceptual changes in individual DCT terms of an image block before combining those estimates into a single estimate of perceptual distance, called Watson distance, $D_{wat}(c_o, c_w)$, where c_o is the cover Work and c_w is the watermarked Work. The Watson distance is measured in terms of the number of *just noticeable differences* (JNDs). A JND is defined as the fidelity difference typically considered as the minimum that is generally perceptible. The computation of perceptual distance in terms of Watson distance is described in Section 6.1. In order to show that this model is better than MSE at estimating the perceptual distortion, the images in Figure 6.1 are used for comparison. The image with white noise added has a Watson distance of 62.57 whereas that

with filtered noise added has a Watson distance of *108.18*, thus showing the perceptual model correctly shows the difference in the perceptual quality.

In this chapter, the aim is to show that the choice of watermark pattern, w_m , can be improved by accounting for the subsequent perceptual distortion that will be incurred to embed the watermark pattern w_m . In other words, the cost function should take into account the perceptual distortion caused by that particular arc. In view of this, we proposed a new cost function and an improvement can be observed under a fixed fidelity environment, i.e. the watermark pattern, w_m , is embedded with a certain strength such that the perceptual distortion is fixed. The fixed fidelity embedding strategy is described in Section 6.2 to provide a foundation in which the watermark pattern, w_m , is embedded during modification stage. Then the watermarking system based on the proposed new cost function is evaluated in Section 6.3 using the fixed fidelity embedding scheme.

6.1 Watson's DCT-based Visual Model

Because of the fact that a lot of images are stored in the JPEG format, Watson's DCT-based visual model [Wat93] is useful for measuring the perceptual fidelity by estimating the perceptual distance, $D_{wat}(c_o, c_w)$, where c_o is the original image and c_w is the distorted version of c_o . Perceptual distance is measured in terms of *just noticeable distance (JND)*.

Watson's model first divides the entire image into 8×8 blocks of pixels. If the image is denoted by \mathbf{c} , the i, j^{th} pixel in block number k is denoted by $\mathbf{c}[i, j, k]$, $0 \leq i, j \leq 7$. The image is transformed into the DCT domain and is denoted by \mathbf{C} . Therefore $\mathbf{C}[i, j, k]$, $0 \leq i, j \leq 7$, denotes a DCT coefficient in the k^{th} block. Obviously, $\mathbf{C}[0, 0, k]$ is the DC term of the k^{th} block.

Watson's model consists of four components, namely sensitivity, luminance masking, contrast masking, and pooling.

6.1.1 Sensitivity

The sensitivity component is based on a frequency sensitivity table, \mathbf{t} . Each table entry, $\mathbf{t}[i, j]$, is approximately the smallest amount of change in the corresponding DCT coefficient in a block that will produce one *JND*. The eye is more sensitive to a frequency if that value, $\mathbf{t}[i, j]$, is small, and less sensitive otherwise. This sensitivity table is de-

pendent on several parameters and it is derived in [AP92]. The set of parameters has been determined so that the resulting frequency sensitivity table, shown in Table 6.1, can be used in experiments mentioned in this report.

1.40	1.01	1.16	1.66	2.40	3.43	4.79	6.56
1.01	1.45	1.32	1.52	2.00	2.71	3.67	4.93
1.16	1.32	2.24	2.59	2.98	3.64	4.60	5.88
1.66	1.52	2.59	3.77	4.55	5.30	6.28	7.60
2.40	2.00	2.98	4.55	6.15	7.46	8.71	10.17
3.43	2.71	6.34	5.30	7.46	9.62	11.58	13.51
4.79	3.67	4.60	6.28	8.71	11.58	14.50	17.29
6.56	4.93	5.88	7.60	10.17	13.51	17.29	21.15

Table 6.1: DCT frequency sensitivity table.

6.1.2 Luminance Masking

If the average intensity of the 8×8 block is brighter, the Watson's model adapts by adjusting a DCT coefficient by a larger amount before being noticed. In Watson's model, this is achieved by adjusting the sensitivity table, $t[i, j]$, for each block, k , according to the block's DC term. The luminance-masked threshold, $t_L[i, j, k]$, is given by

$$t_L[i, j, k] = t[i, j](C_o[0, 0, k]/C_{0,0})^{a_T}, \quad (6.1)$$

where a_T is a constant with a suggested value of 0.649, $C_o[0, 0, k]$ is the DC coefficient of the k^{th} block of the original image, and $C_{0,0}$ is the average value of all the DC coefficients in the image. Equation 6.1 indicates that brighter regions can accommodate larger changes before being noticed.

6.1.3 Contrast Masking

Contrast masking is the reduction of visibility of a change in one frequency due to the energy present in that frequency. This affects the luminance-masked threshold, $t_L[i, j, k]$, and results in a masking threshold, $s[i, j, k]$, given by

$$s[i, j, k] = \max\{t_L[i, j, k], |C_o[i, j, k]|^{w[i,j]} t_L[i, j, k]^{1-w[i,j]}\}, \quad (6.2)$$

where $w[i, j]$ is a constant between 0 and 1. Watson chooses a value of $w[i, j] = 0.7, \forall i, j$. The masking threshold called slacks, $s[i, j, k]$, is the estimated amount of change in individual terms of the block DCT that result in one JND.

6.1.4 Pooling

Comparison between the original image, c_o , and a distorted image, c_w , is done by first computing the differences between their DCT coefficients $C_w[i, j, k]$ and $C_o[i, j, k]$ respectively. These differences are then scaled by their respective slacks and eventually combined into a single perceptual distance given by

$$D_{wat}(c_o, c_w) = \left(\sum_{i,j,k} \left| \frac{C_w[i, j, k] - C_o[i, j, k]}{s[i, j, k]} \right|^p \right)^{\frac{1}{p}}, \quad (6.3)$$

where $p = 4$ is a value recommended by Watson.

6.2 Fixed Fidelity Embedding

In Section 5.3, a fixed robustness scheme is used during the watermark embedding stage to provide a fixed resistance to additive white Gaussian noise (AWGN). However, in some watermarking applications, the fidelity of the watermarked Work, c_w , is of paramount importance and a fixed fidelity constraint is preferred. In this scenario, a perceptual model is required to measure distortion and thus provide a basis for fixed fidelity embedding. In particular, we consider the Watson's DCT-based visual model [Wat93], described in Section 6.1, which estimates the number of JNDs (just noticeable differences) between images.

The simplest method to achieve a fixed perceptual distance, D_{target} , during embedding is to allow automatic adjustment of the global embedding strength α . This method uses a simple watermarking algorithm, E_PERC_GSCALE, mentioned in Section 7.4 of [CMB01] and is described as follows.

The watermarked Work is

$$c_w = c_o + w_a \quad (6.4)$$

where c_o is the original cover Work and w_a is the added mark. The added mark is given by $w_a = \alpha w_m$, where w_m is the message mark.

The perceptual distance between c_w and c_o , as measured by the Watson's model, $D_{wat}(c_o, c_w)$, is a linear function of α . Let C_w , C_o and W_m be the block DCT of c_w ,

\mathbf{c}_o , and \mathbf{w}_m , respectively. Because the block DCT is a linear transform, we have

$$\mathbf{C}_w = \mathbf{C}_o + \alpha \mathbf{W}_m \quad (6.5)$$

According to the pooling function used in the Watson's model (Equation 6.3), the perceptual distance between \mathbf{c}_w and \mathbf{c}_o is estimated as

$$D_{wat}(\mathbf{c}_o, \mathbf{c}_w) = \sqrt[4]{\sum_{i,j,k} \left(\frac{C_w[i, j, k] - C_o[i, j, k]}{s[i, j, k]} \right)^4} \quad (6.6)$$

where s is an array of slacks based on the estimated sensitivity function of the eye and the masking properties of \mathbf{c}_o . Substituting $\alpha W_m[i, j, k]$ for $C_w[i, j, k] - C_o[i, j, k]$ gives

$$D_{wat}(\mathbf{c}_o, \mathbf{c}_w) = \sqrt[4]{\sum_{i,j,k} \left(\frac{\alpha W_m[i, j, k]}{s[i, j, k]} \right)^4} \quad (6.7)$$

$$= \alpha \sqrt[4]{\sum_{i,j,k} \left(\frac{W_m[i, j, k]}{s[i, j, k]} \right)^4} \quad (6.8)$$

$$= \alpha D_{wat}(\mathbf{c}_o, \mathbf{c}_o + \mathbf{w}_m) \quad (6.9)$$

Thus, to set $D_{wat}(\mathbf{c}_o, \mathbf{c}_w)$ equal to a desired perceptual distance, D_{target} , we obtain α as

$$\alpha = \frac{D_{target}}{D_{wat}(\mathbf{c}_o, \mathbf{c}_o + \mathbf{w}_m)} \quad (6.10)$$

In practice, round-off and clipping (i.e. underflow and overflow) can cause watermarks embedded with this α to have different numbers of JNDS. To mitigate this problem, we perform an exhaustive search of values between 0.2α and 1.1α , looking for one that, with round-off and clipping, yields the closest value to the desired number of JNDS.

Figure 6.2 illustrates how fixed fidelity embedding is carried out. The surface of a hypersphere is divided into many detection regions, each consisting of a codeword in its centre. The bad codewords are denoted by \times while the good codeword, i.e. \mathbf{w}_m , is denoted by \blacksquare . The fixed fidelity region can be approximated by an elliptical boundary (in blue) with the cover Work, \mathbf{c}_o , denoted by \bullet , in the centre. Anywhere on this elliptical boundary represents a Work with the same fidelity measured by the Watson model. The size of the elliptical region varies according to the allowed distortion.

Since fixed fidelity embedding is defined by Equation 6.4, the embedder searches for a point on the elliptical boundary that lies along the direction towards the correct

codeword, w_m , taking reference from the cover Work, c_o . Hence the watermarked Work, c_w , is the point denoted by \blacktriangle that satisfies the conditions.

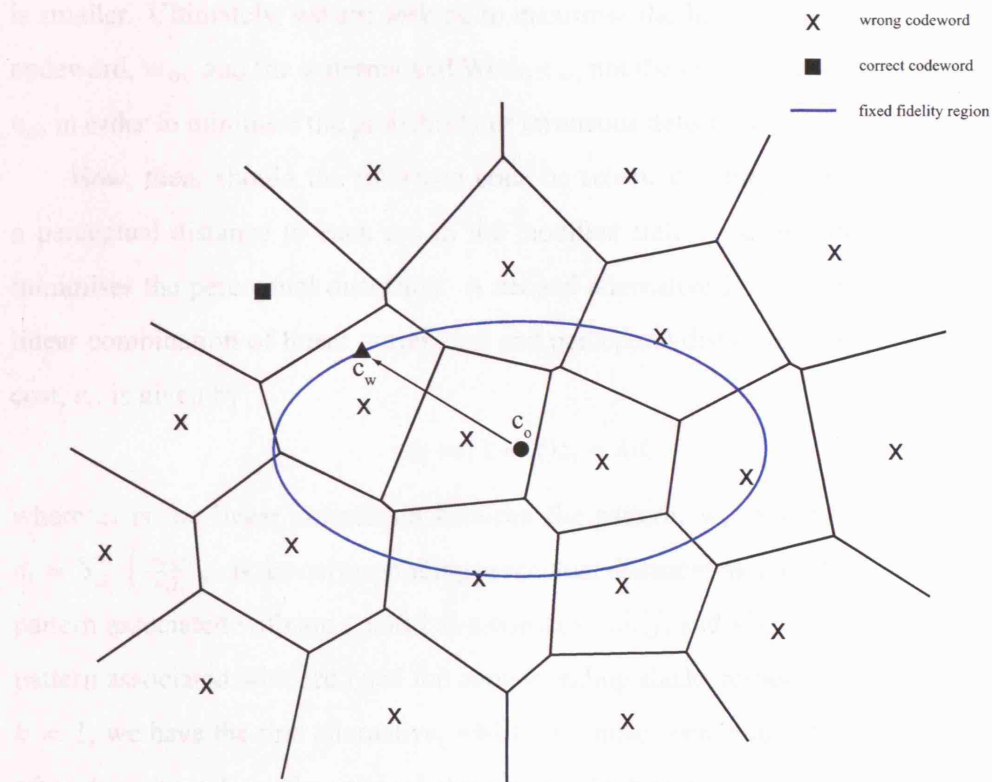


Figure 6.2: An illustration of fixed fidelity embedding.

6.3 New Cost Function

When we use the combination of informed coding and informed embedding based on the scheme used in Section 6.2, the watermark embedder first selects the preferred codeword from the set of codewords that represent the desired message. The preferred codeword is chosen to maximise the linear correlation between the pattern to be embedded and the cover Work c_o . The watermark detector also uses linear correlation to determine the most likely message.

Once the preferred codeword is selected, it is embedded with a strength, α , that is chosen to maintain a fixed fidelity. Unfortunately, there is no clear relationship between linear correlation and perceptual distance. Thus, it is entirely possible that a codeword chosen to have a high correlation with the cover Work, may, in fact, be embedded with a low strength due to the perceptual distortion that it introduces. Conversely, it may

transpire that a codeword that has a smaller linear correlation with the cover Work, c_o , can be embedded with a high strength if the corresponding perceptual distortion is smaller. Ultimately, we are seeking to maximise the linear correlation between the codeword, w_m , and the watermarked Work, c_w , not the original unwatermarked Work, c_o , in order to minimise the probability of erroneous detection.

How, then, should the informed code be selected? One alternative is to assign a perceptual distance to each arc in the modified trellis and find the codeword that minimises the perceptual distortion. A second alternative is to assign a cost that is a linear combination of linear correlation and perceptual distance. In particular, the arc cost, e_i , is given by:

$$e_i = (1 - k)z_i - kd_i \quad (6.11)$$

where z_i is the linear correlation between the pattern, w_r , and the cover Work c_o , $d_i = \sum_j \left(\frac{w_r[j]}{s[j]} \right)^4$ is the corresponding perceptual distance² as a result of embedding the pattern associated with arc i , and k is a constant. $w_r[j]$ and $s[j]$ are j -th element of the pattern associated with arc i and the corresponding slacks respectively. Clearly, when $k = 1$, we have the first alternative, which minimises perceptual distortion only. And when $k = 0$ we have the original algorithm, which maximises the linear correlation. Other alternatives are possible but are not considered in this thesis. Note that although it is desirable to normalise both z_i and d_i , the normalisation factors for z_i and d_i obtained from any one image are not sufficiently accurate. A huge amount of statistics has to be computed from a large collection of varying images and a large number of randomly generated dirty paper trellises to ensure accuracy in the obtained normalisation factors. Hence it is difficult for normalisation to be carried out.

To determine whether the performance of (i) linear correlation alone, (ii) perceptual distance alone or (iii) a linear combination of the two is superior, we conducted an experiment using a dirty-paper trellis with 64 states and 64 arcs per state so as to compare with the results reported in [MDC04].

Watermark embedding proceeds as follows:

1. Compute the discrete cosine transform (DCT) of each 8×8 block of an image, c_o , to obtain C_o .

²Refer to Equation 6.3. The quartic root is removed in this case since individual d_i 's have to be added while traversing the trellis.

2. Extract the 12 lowest-frequency AC terms of each block, as shown in Figure 6.3, to form a single, $12 \times N_b$ length vector, \mathbf{v} , where N_b is the number of blocks in the image. The vector \mathbf{v} is referred to as the extracted vector.
3. Use the dirty-paper trellis to encode the desired message, m , into a watermark vector, \mathbf{w}_m . This was done by running Viterbi's algorithm on the extracted vector, \mathbf{v} , using a trellis modified for message m . Refer to Section 3.3 to see how a modified trellis is obtained. The cost associated with an arc is given by Equation 6.11.
4. Embed \mathbf{w}_m into \mathbf{v} with informed embedding: $\mathbf{v}_w = \mathbf{v} + \alpha \mathbf{w}_m$, where α is the global embedding strength and is chosen such that the fidelity of the watermarked Work, c_w , is fixed at a desired Watson distance D_{target} .
5. Place the values of \mathbf{v}_w into the corresponding low-frequency AC terms of the block-DCT of the cover Work, C_o to produce the watermarked Work C_w .
6. Convert the image, C_w , back into the spatial domain to obtain the watermarked Work c_w .

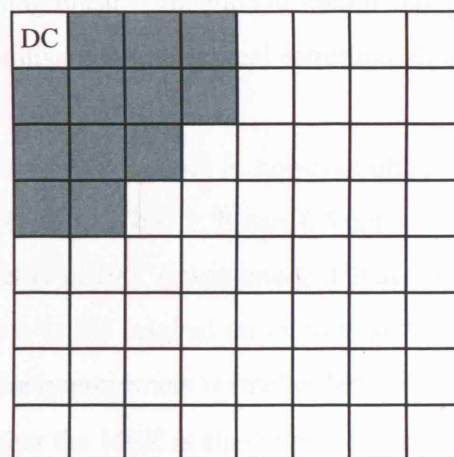


Figure 6.3: The 12 lowest-frequency DCT coefficients (shaded in gray) used for embedding. The top-left corner DCT coefficient is the DC term and is not used for embedding.

Watermark detection proceeds as follows:

1. Extract a vector, \mathbf{v}' , from the watermarked image in the same manner as in steps 1 and 2 of the embedding algorithm.
2. Apply the Viterbi algorithm to \mathbf{v}' , by using the whole trellis, to identify the path whose code vector yields the highest linear correlation.
3. Record the decoded message, m' , associated with the corresponding path.

To evaluate the different algorithms, we used a database of 2000 images, each of dimension 240×368 . Thus, the number of 8×8 blocks is $N_b = 1380$. The bit error rate (BER) and message error rate (MER) are computed. The Message Error is defined as being zero if all 1380 bits are correctly decoded and one, otherwise.

The BER and MER are shown in Figure 6.4 and Figure 6.6 for three values of fidelity, i.e. three values of embedding strength, α and for k ranging from 0 to 1. The percentage improvements (over the scheme in which linear correlation is used as the cost function) in BER and MER are illustrated in Figure 6.5 and Figure 6.7. Figure 6.8 shows an original image and three watermarked images with three different fidelity, i.e. Watson distances 30, 50, and 100.

Figure 6.4 and Figure 6.6 clearly reveal that the BER and MER are worse for $k = 0$ and $k = 1$, i.e. maximising linear correlation or minimising perceptual distortion. The performance when minimising the perceptual distortion alone is much worse than for maximising linear correlation alone.

Significantly improved performance is, however, obtained when a combination or the two measures is used, i.e. $0 < k < 1$. Figure 6.5 indicates that the BER is improved by almost 50% for a fidelity of 100. A watermarked image, c_w , with a Watson distance of 100 is about the same as the original unwatermarked image. For a fixed Watson distance of 50 and 30, the improvement is smaller but still significant.

Figure 6.7 shows that the MER is also improved, this time by almost 25% for a fixed Watson distance of 100. Curiously, the MER continues to increase for values of k up to 0.65, even though the improvement in BER peaks at about 0.1. It is unclear why this is so. As with the BER, the improvement in MER is less when the fidelity is constrained to a Watson distance of 50 and no improvement is measurable for a Watson distance of 30.

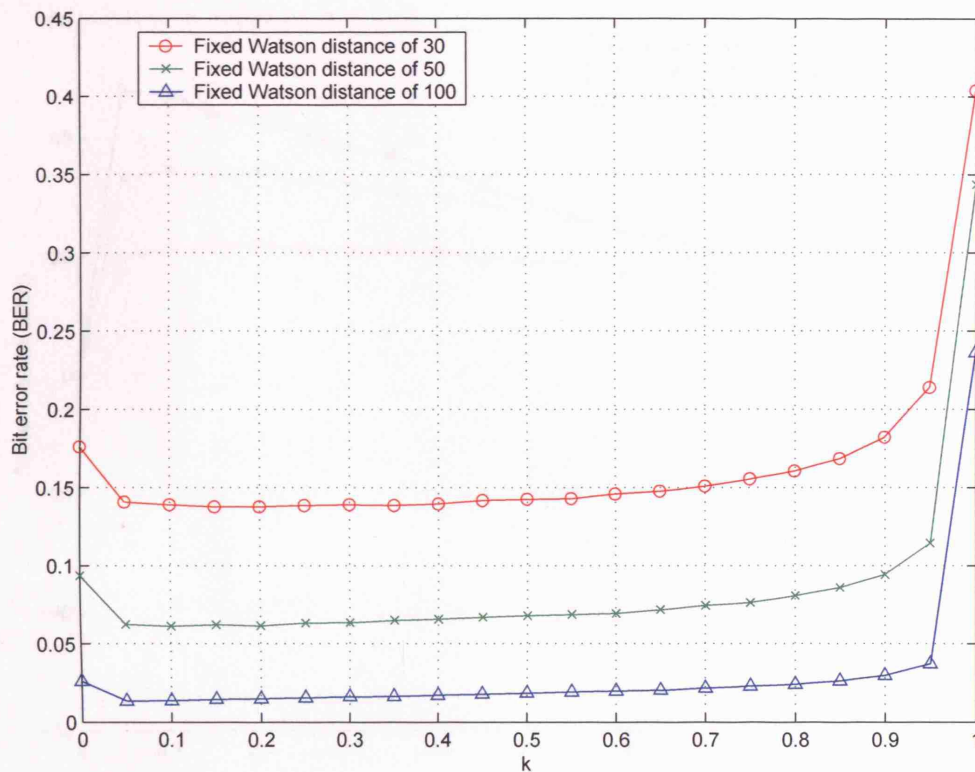


Figure 6.4: The bit error rate (BER) as a function of k for three values of fidelity.

6.4 Summary

We have demonstrated that the choice of codeword to embed in a cover Work, c_o , can be significantly improved by maximising a cost function that is a linear combination of linear correlation and perceptual distortion, rather than linear correlation alone.

Finding the codeword that maximises the linear correlation with the original cover Work, c_o , does not guarantee that the linear correlation is maximised after embedding. This is because said codeword may need to be attenuated more strongly than alternative codewords in order to satisfy a perceptual constraint.

Finding the codeword that minimises the perceptual distortion with the original permits said codeword to be embedded more strongly. However, the codeword may have a very low linear correlation with the cover Work, c_o , and result in very poor performance at the detector.

A linear combination of perceptual distortion and linear correlation was shown to be superior, improving the bit error rate by about 50% and the message error rate by

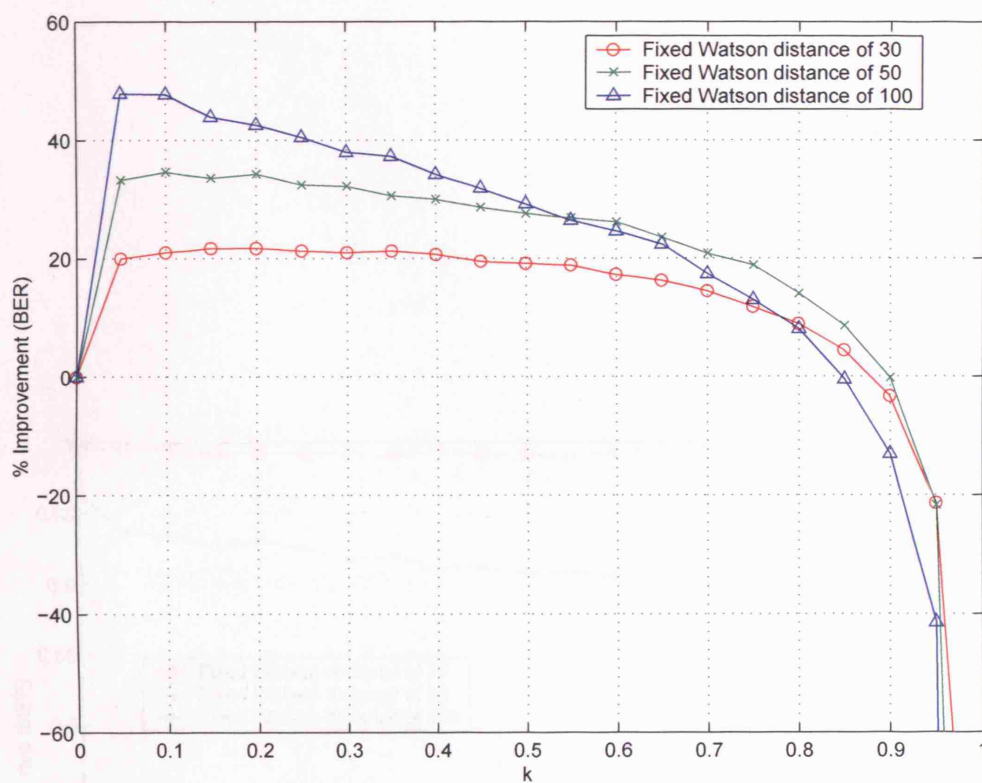


Figure 6.5: The percentage improvement in bit error rate (BER) as a function of k for three values of fidelity.

about 25% for a fixed fidelity of 100.

Further investigation is needed to determine the optimal criterion, which would maximise the linear correlation *after* watermark embedding, subject to a fidelity constraint.

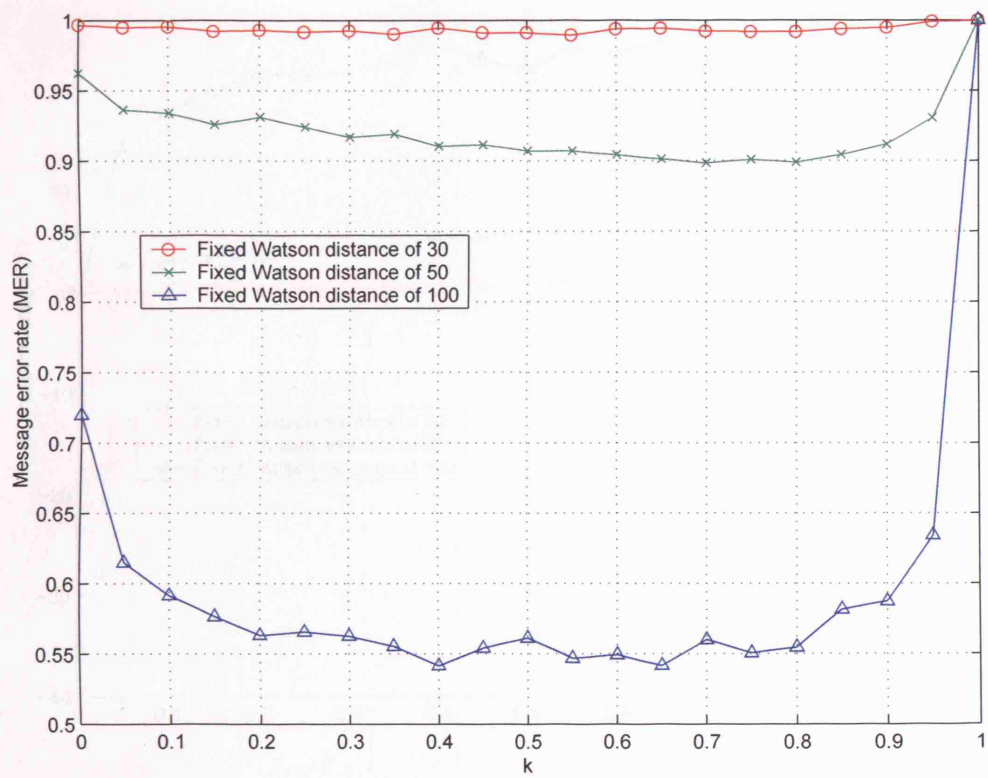


Figure 6.6: The message error rate (MER) as a function of k for three values of fidelity.

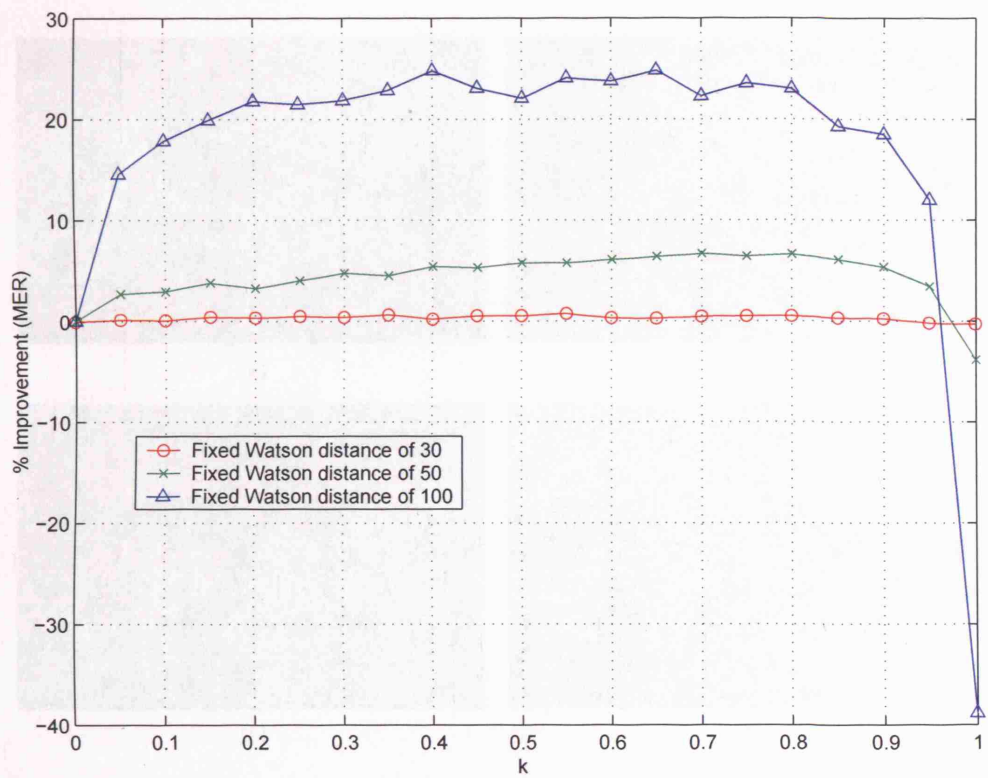


Figure 6.7: The percentage improvement in message error rate (MER) as a function of k for three values of fidelity.

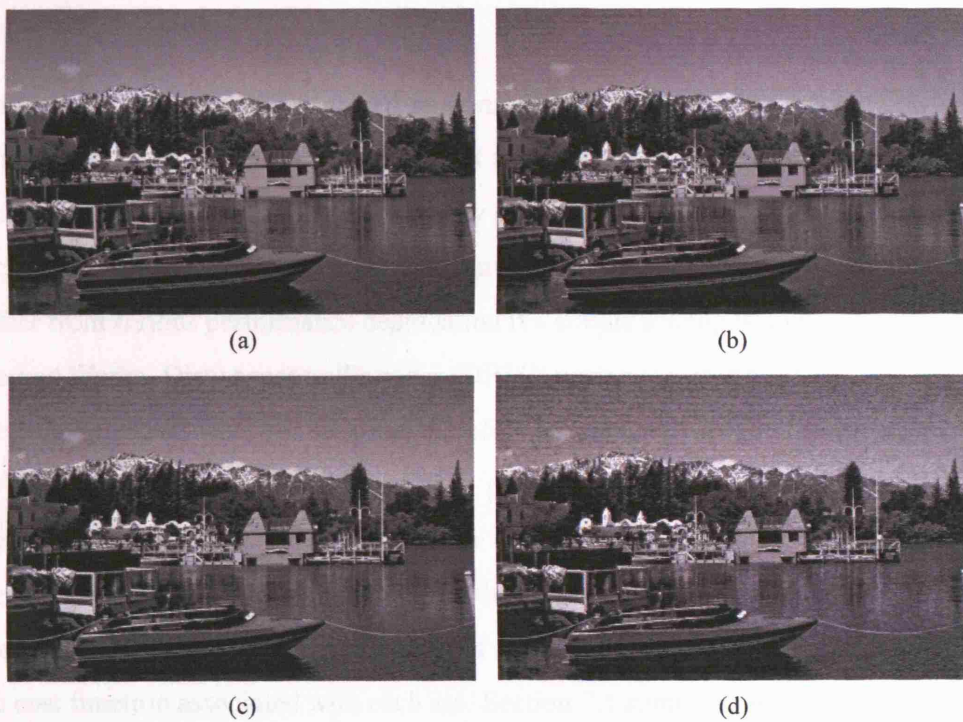


Figure 6.8: Images with different fidelity. (a) is an original image while (b), (c), and (d) are watermarked images with Watson distances 30, 50, and 100 respectively.

Chapter 7

Conclusion

Watermarking has been recognised as communications with side information at the transmitter. The advantage of this is that the interference due to the cover Work is eliminated, thereby increasing the capacity of the watermarking system. Practical watermarking schemes such as lattice codes and syndrome codes were proposed but they suffer from serious performance degradation if a simple scaling is applied to the watermarked Works. Dirty paper trellis codes (DPTC) were suggested as an alternative since they are a form of spherical code and therefore are robust to amplitude scaling.

Many design parameters affect dirty paper trellises. However, the influence of these parameters on the trellis performance was not fully understood. In particular, this thesis examined the following factors: (i) the number of states and the number of arcs per state in the trellis, (ii) the distribution of codewords generated by the trellis, and (iii) the cost function associated with each arc. Section 7.1 summarises the contributions of this thesis in regard to these parameters and suggests some possible future work. All works presented in this thesis were published in international conference proceedings and the list of publications can be found in Appendix A.

7.1 Contributions and Future Work

Firstly, experimental results on synthetic signals provided a deeper understanding on the influence of different trellis configurations, i.e. the number of states, S , and the number of states per arc, A , on trellis performance. An optimum trellis structure can be selected based on both the bit error rate (BER) performance and the computational cost. For example, in the case of noiseless channel environment, a trellis configuration having $A \leq S$ should be chosen instead of that having $A > S$ because errors are

more difficult to occur for the former compared to the latter. Furthermore, experimental results suggested that trellis configurations $A = S$ (i.e. fully connected trellis) and $A = S/2$ (i.e. half connected trellis) are more desirable since these structures offer similar BER performance compared to others but at a lower computational cost. However, in the situation when the channel is noisy, results have shown that neither a fully connected trellis nor a half connected trellis can guarantee to be an optimum trellis configuration. Whether the channel is noiseless or noisy, the trellis configuration with the lowest computational cost at the desired operating error rate should be selected so as to minimise the time taken during watermark detection.

Secondly, since a well-distributed set of codewords that are generated by a trellis leads to a good trellis performance, we proposed a new type of trellis using trellis coded modulation (TCM) for use to replace the original dirty paper trellis. Experimental results on real images showed that, by using a fixed robustness embedding scheme, this new type of dirty paper trellis, called TCM DPTC, outperforms its original random DPTC. At roughly the same distortion, e.g. mean square error (MSE) equals to 0.83, incurred during watermarking embedding, watermarks embedded using TCM DPTC can survive a noise addition of up to a standard deviation of $\sigma = 1.5$ whereas those using random DPTC can only cope up to $\sigma = 1.0$. This proved that trellis coded modulation leads to an improved distribution of codewords on the hypersphere which in turn leads to an improved in performance.

However, it is unclear if TCM DPTC generates a set of codewords that are optimally distributed on a sphere. Good spherical codes are known to be those with their codewords separated as far as possible from one another. The design of spherical codes is very important if suboptimal embedding techniques are used. For instance in [LCD05], the fixed-robustness embedding region is approximated by the same hyperbola for all detection regions even though these regions are not equally shaped. In other words, the hyperbolic embedding region will not guarantee a fixed robustness. To ensure all codewords are distributed uniformly on a hypersphere's surface, a generic codeword generation mentioned in [Ham96] can be used. This method uses an iterative procedure to ensure that the minimum distance between any pair of codewords increases each time. The procedure is terminated once it has reached a desired separation among the codewords. The resulting set of codewords can be partitioned, in a

similar way to trellis coded modulation (see Section 5.1), for use in dirty paper trellis codes.

Lastly, we proposed a new cost function — a linear combination of linear correlation and perceptual distortion — for use in dirty paper trellis watermarking instead of the traditional cost function — linear correlation — since the latter may be difficult to embed due to perceptual constraints. This means that the selection of the codeword to embed depends on its closeness to the cover Work in terms of linear correlation and perceptual distortion as measured by Watson's distance. Experimental results demonstrated significant improvements in the choice of codeword to embed in a cover Work by maximising this new cost function. The performance, in terms of BER and MER, have improved by about 50% and 25% respectively when the linear combination factor of around $k = 0.1$ and $k = 0.65$ are selected respectively.

It is unclear why the two k values are different when two different error rates, BER and MER, are used. Further investigation is needed to determine the optimal criterion, which would maximise the linear correlation *after* watermark embedding, i.e. $c_w \cdot w$, subject to a fidelity constraint. Alternatively, different perceptual models can be used in place of Watson's model, especially since Watson's model is sensitive to blocking artifacts. For example, Gabor filters by C. J. van de Branden Lambrecht and J. E. Farrell [vF96] and pixel-based model by Voloshynovskiy *et al.* [VHBP99] are of particular interest.

Currently, dirty paper trellises are built on the foundation of convolutional codes. An improvement to dirty paper trellises is to use a more powerful channel code than convolutional codes. In 1993, the concept of turbo codes appeared in the communications community. Since then, it is known that turbo codes [BGT93] perform much better than convolutional codes although this performance improvement comes at the expense of a higher computational cost. In order to achieve low error rates, turbo codes employ an iterative search at the detector to determine the codeword that has been sent. Future work should examine whether dirty paper trellis codes can make use of turbo codes to generate more powerful dirty paper turbo codes. This new class of dirty paper codes might be designed to effectively embed the chosen codeword and then minimise the error rates at the watermark detector through iterative decoding.

Appendix A

Publications

The works presented in this thesis have resulted in the following publications.

1. C. K. Wang, M. L. Miller, and I. J. Cox. Using perceptual distance to improve the selection of dirty paper trellis codes for watermarking. In *Proc. IEEE Int. Workshop on Multimedia Signal Processing*, pages 147–150, Siena, Italy, Sept. 2004.
2. C. K. Wang, G. Doërr, and I. J. Cox. Toward a better understanding of dirty paper trellis codes. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 233–236, Toulouse, France, May 2006.
3. C. K. Wang, G. Doërr, and I. J. Cox. Trellis coded modulation to improve dirty paper trellis watermarking. In *Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 0G1–0G10, San Jose, California, USA, Jan. 2007.

Bibliography

- [ABPGM05] A. Abrardo, M. Barni, F. Pérez-González, and C. Mosquera. Trellis-coded rational dither modulation for digital watermarking. In *Proceedings of the 4th International Workshop on Digital Watermarking*, volume LNCS 3710, pages 351–360, Sept. 2005.
- [AP92] A. J. Ahumada and H. A. Peterson. Luminance-model-based DCT quantization for color image compression. In *Proc. SPIE: Human Vision, Visual Processing, and Digital Display III*, volume 1666, pages 365–374, San Jose, California, USA, Feb. 1992.
- [BBC⁺99] M. Barni, F. Bartolini, V. Cappellini, A. Lippi, , and A. Piva. A DWT-based technique for spatio frequency masking of digital signatures. In *Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents*, volume 3657, pages 31–39, San Jose, California, USA, Jan. 1999.
- [BCK⁺99] J. A. Bloom, I. J. Cox, T. Kalker, Jean-Paul M. G. Linnartz, M. L. Miller, and C. B. S. Traw. Copy protection for DVD video. In *Proc. IEEE*, volume 87, pages 1267–1276, July 1999.
- [BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. IEEE Int. Conf. on Communications*, volume 2, pages 1064–1070, Geneva, Switzerland, May 1993.
- [BRC60a] R. C. Bose and D. K. Ray-Chaudhuri. Further results in error correcting binary group codes. volume 3, pages 279–290, Sept. 1960.

- [BRC60b] R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. volume 3, pages 68–79, March 1960.
- [CKLS97] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoan. Secure spread spectrum watermarking for multimedia. volume 6, pages 1673–1687, Dec. 1997.
- [CL02] A. S. Cohen and A. Lapidoth. The gaussian watermarking game. volume 48, pages 1639–1667, June 2002.
- [CMB00] I. J. Cox, M. L. Miller, and J. A. Bloom. Watermarking applications and their properties. In *Proc. IEEE Int. Conf. on Information Technology: Coding and Computing*, pages 6–10, The Orleans, Las Vegas, Nevada, USA, March 2000.
- [CMB01] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2001.
- [CMM99] I. J. Cox, M. L. Miller, and A. L. McKellips. Watermarking as communications with side information. volume 87, pages 1127–1141, July 1999.
- [Cos83] M. Costa. Writing on dirty paper. volume 29, pages 439–441, May 1983.
- [CPGR00] J. Chou, S. S. Pradhan, L. E. Ghaoui, and K. Ramchandran. Watermarking based on duality with distributed source coding and robust optimization principles. In *Proc. IEEE Int. Conf. on Image Processing*, volume 1, pages 585–588, Vancouver, BC, Canada, Sept. 2000.
- [CPR99] J. Chou, S. S. Pradhan, and K. Ramchandran. On the duality between distributed source coding and data hiding. In *Proc. Thirty-third Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1503–1507, Pacific Grove, CA, USA, Oct. 1999.
- [CS98] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups (Third Edition)*. Springer, 1998.

- [CW98] B. Chen and G. W. Wornell. Digital watermarking and information embedding using dither modulation. In *Proc. IEEE Int. Workshop on Multimedia Signal Processing*, pages 273–278, Redondo Beach, California, USA, Dec. 1998.
- [CW99] B. Chen and G. W. Wornell. An information-theoretic approach to the design of robust digital watermarking systems. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 4, pages 2061–2064, Phoenix, Arizona, USA, March 1999.
- [CW01] B. Chen and G. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. volume 47, pages 1423–1443, May 2001.
- [EBG02] J. Eggers, R. Bäuml, and B. Girod. Estimation of amplitude modifications before SCS watermark detection. In *Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents*, pages 387–398, San Jose, California, USA, January 2002.
- [Eli54] P. Elias. Error-free coding. volume 4, pages 29–37, September 1954.
- [ESZ00] U. Erez, S. Shamai, and R. Zamir. Capacity and lattice-strategies for cancelling known interference. In *Proc. ISITA 2000*, pages 681–684, Honolulu, Hawaii, USA, Nov. 2000.
- [GHW92] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein. *Data Communications Principles*. Plenum Press, New York, 1992.
- [Gir93] B. Girod. What’s wrong with mean-squared error? In A. B. Watson, editor, *Digital Images and Human Vision*, chapter 15, pages 207–220. MIT Press, 1993.
- [Gol49] M. J. E. Golay. Notes on digital coding. volume 37, page 657, 1949.
- [Ham50] R. W. Hamming. Error detecting and error correcting codes. volume 29, pages 147–160, Apr. 1950.

- [Ham96] J. Hamkins. Design and analysis of spherical codes. Technical Report Ph.D. thesis, Univ. of Illinois at Urbana-Champaign, September 1996.
- [Hoc59] A. Hocquenghem. Codes correcteur derreurs. volume 2, pages 147–156, 1959.
- [KP99] M. Kutter and F. A. P. Petitcolas. A fair benchmark for image watermarking systems. In *Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents*, volume 3657, pages 226–239, San Jose, California, USA, Jan. 1999.
- [KP00] S. Katzenbeisser and F. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.
- [LC07] Q. Li and I. J. Cox. Using perceptual models to improve fidelity and provide resistance to volumetric scaling for quantization index modulation watermarking. volume 2, pages 127–139, June 2007.
- [LCD05] L. Lin, I. J. Cox, and G. Doerr. An efficient algorithm for informed embedding of dirty-paper trellis codes for watermarking. In *Proc. IEEE Int. Conf. on Image Processing*, volume 1, pages 697–700, Genova, Italy, Sept. 2005.
- [LKKM03] K. Lee, D. S. Kim, T. Kim, and K. A. Moon. EM estimation of scale factor for quantization-based audio watermarking. In *Proceedings of the 2nd International Workshop on Digital Watermarking, IWDW 2003*, volume LNCS 2939, pages 316–327, Seoul, Korea, October 2003.
- [MDC02] M. L. Miller, G. J. Doerr, and I. J. Cox. Dirty-paper trellis codes for watermarking. In *Proc. IEEE Int. Conf. on Image Processing*, volume 2, pages 129–132, Rochester, New York, USA, Sept. 2002.
- [MDC04] M. L. Miller, G. J. Doerr, and I. J. Cox. Applying informed coding and embedding to design a robust, high capacity watermark. volume 13, pages 792–807, June 2004.

- [MF90] M. W. Marcellin and T. R. Fischer. Trellis coded quantization of memoryless and Gauss-Markov sources. volume 38, pages 82–93, Jan. 1990.
- [MF03] H. S. Malvar and D. A. F. Florencio. Improved spread spectrum: A new modulation technique for robust watermarking. volume 51, pages 898–905, Apr. 2003.
- [OKS04] J. Oostveen, T. Kalker, and M. Staring. Adaptive quantization watermarking. In *Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 296–303, San Jose, California, USA, Jan. 2004.
- [OLJPG05] F. Ourique, V. Licks, R. Jordan, and F. Pérez-González. Angle QIM: A novel watermark embedding scheme robust against amplitude scaling distortions. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 797–800, Philadelphia, PA, USA, March 2005.
- [PCR03] S. S. Pradhan, J. Chou, and K. Ramchandran. Duality between source coding and channel coding and its extension to the side information case. volume 49, pages 1181–1203, May 2003.
- [PDL⁺90] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck, and D. J. Costello. Trellis-coded multidimensional phase modulation. volume 36, pages 63–89, January 1990.
- [pla72] *Playboy Magazine*. Playboy Enterprises International, Inc., November 1972.
- [PR99] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes DISCUS : Design and construction. In *Proc. IEEE Data Compression Conference*, pages 158–167, Snowbird, UT, USA, March 1999.
- [Pra78] W. K. Pratt. *Digital Image Processing*. New York: Wiley, 1978.
- [Pro01] J. G. Proakis. *Digital Communications*. McGraw-Hill, 2001.

- [RDB96] J. J. K. Ruanaidh, W. J. Dowling, and F. M. Boland. Phase watermarking of digital images. In *Proc. IEEE Int. Conf. on Image Processing*, volume 3, pages 239–242, Lausanne, Switzerland, Sept. 1996.
- [RG83] R. Reiningger and J. Gibson. Distributions of the two-dimensional DCT coefficients for images. volume 31, pages 835–839, June 1983.
- [RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. volume 8, pages 300–304, June 1960.
- [Sha48] C. E. Shannon. A mathematical theory of communications. volume 27, pages 379–423, July 1948.
- [Sha58] C. E. Shannon. Channels with side information at the transmitter. volume 2, pages 289–293, 1958.
- [SW73] D. Slepian and J. K. Wolf. Noiseless coding of correlated information sources. volume 19, pages 471–480, July 1973.
- [TD97] B. Tao and B. Dickinson. Adaptive watermarking in the DCT domain. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 2985–2988, Munich, Germany, April 1997.
- [TW01] W. Trappe and L. C. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, 2001.
- [Ung82] G. Ungerboeck. Channel coding with multilevel/phase signals. volume 28, pages 55–67, Jan. 1982.
- [vF96] C. J. van den Branden Lambrecht and J. E. Farrell. Perceptual quality metric for digitally coded color images. In *Proc. European Signal Processing Conference (EUSIPCO)*, pages 1175–1178, Trieste, Italy, Sept. 1996.
- [VHBP99] S. Voloshynovskiy, A. Herrigel, N. Baumgaertner, and T. Pun. A stochastic approach to content adaptive digital image watermarking. In *Proc. Int. Workshop on Information Hiding*, pages 211–236, Dresden, Germany, Sept. 1999.

- [Vit67] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. volume 13, pages 260–269, Apr. 1967.
- [vSTO94] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. A digital watermark. In *Proc. IEEE Int. Conf. on Image Processing*, volume 2, pages 86–89, Austin, Texas, USA, Nov. 1994.
- [Wat93] Andrew B. Watson. Visually optimal DCT quantization matrices for individual images. In *Proc. IEEE Data Compression Conf.*, pages 178–187, Snowbird, Utah, USA, March 1993.
- [WD96] R. B. Wolfgang and E. J. Delp. A watermark for digital images. In *Proc. IEEE Int. Conf. on Image Processing*, volume 3, pages 219–222, Lausanne, Switzerland, Sept. 1996.