

On the Application of Coding to Adaptive Steganography: Robustness, Efficiency, and Coding Loss



Christopher Stephen Kin-Cleaves
Exeter College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2020

Abstract

Steganography is the process by which we hide *payloads* (secret messages) into *covers* (inconspicuous mediums of data) resulting in *stego objects* (original mediums of data with subtle changes to encode our message), with the aim of these objects appearing to be indistinguishable from the cover. Steganography aims to compromise the adversaries' ability to classify an object as innocent or not. The most successful steganography is adaptive embedding, which uses a distortion function to associate individual cover changes with a cost of change (higher costs implying a more detectable change), and a code which embeds the payload whilst minimising the sum of distortion (a proxy for detectability).

This thesis explores approaches to codes, for adaptive embedding on the subjects of robustness, efficiency, and simulating. Since the wide adoption of adaptive embedding, the significant focus of most researchers in the field has been on distortion algorithms. Our work has aimed to explore the potentials around codes instead, building on the highly successful *Syndrome Trellis Codes* (STC) algorithm.

On the area of robustness, we explore the ability to create stego objects able to withstand the noisy (non-adversarial) channel. Since syndrome codes are fragile by design, syndrome-encoded stego objects are a bad choice: amplifying the channel error rate into unacceptably-high payload error rates.

Our work on improving the efficiency of STCs looks at how $\mathbb{H}$ matrix construction can influence the efficiency of the algorithm. Expanding on existing research around minimising outliers, we look at how moving from a block-based matrix to one constructed with rows can improve the embedding efficiency.

Finally, we explore the area around simulated steganography. Because of speed constraints, many researchers omit the use of a code, to speed up experiments. The justification is clear: not using a code saves both time and experimental complexity, when you can simulate stego objects on the *rate-distortion bound*. If we cannot achieve the rate-distortion bound with a code, and more parameters are influencing this than which are taken into account when simulating, is this a sensible option? We propose a method which samples the code with the given parameters and is then able to simulate objects not on the rate-distortion bound, but at the distortion and detectability of a code.

Keywords— Steganography, steganalysis, adaptive embedding, simulated embedding, robustness

Acknowledgements

I want to express my most sincere gratitude to Dr Andrew Ker. Without his advice, very patient explanations, and mentoring, this work would not be possible. His academic rigour and standards have helped me improve substantially over the years, alongside the fruitful discussions about steganography, computer science, and mathematics. My time as a DPhil student has been very enjoyable, and I owe a large part of this to Dr Ker. Alongside this, I'd like to thank Tomás Pényv and Patrick Bas for their contributions and expertise.

This DPhil would not be possible without the generous funding and financial support from the Engineering and Physical Sciences Research Council (EPSRC), the Department of Computer Science at the University of Oxford, and Exeter College.

To my friends who have supported me during my DPhil, I'd like to thank you for all the DPhil distractions, support, laughs, and fond memories over the years. Finally, a heartfelt thank you to my family. Without your constant support and guidance, I'd have never made it this far. Thank you for keeping me optimistic and encouraging me all the way.

Contents

1	Introduction	5
1.1	Historical Steganography	6
1.2	Steganography Today	9
1.3	Definition of Steganography	10
1.4	Distortion	14
1.5	Roadmap of Thesis	19
1.5.1	Research Topic	19
1.5.2	Overview of Research Chapters	20
2	Background and Literature Survey	24
2.1	Notation	25
2.2	Covers	26
2.2.1	RAW Images	26
2.2.2	JPEG Images	27
2.2.3	Video, Sound, and Other Covers	30
2.3	Embedding Operations	32
2.3.1	LSB Replacement	33
2.3.2	LSB Matching (± 1)	35
2.4	Distortion Functions	36
2.4.1	Non-adaptive	37
2.4.2	WOW - Wavelet Obtained Weights	38
2.4.3	UNIWARD - UNiversal WAvelet Relative Distortion	39
2.4.4	J-UNIWARD - JPEG UNIWARD	40
2.4.5	MiPOD - Minimising the Power of Optimal Detector	41
2.5	Optimality	42
2.5.1	Additive Distortion Functions	43
2.5.2	Coding Loss	45
2.6	Legacy Codes	47
2.6.1	Matrix Embedding	48
2.6.2	F5 and nsF5	52

2.7	Syndrome Trellis Codes	54
2.7.1	Binary Syndrome Trellis Codes	55
2.7.2	Non-Binary and Multi-Layered Syndrome Trellis Codes	60
2.8	Forward Error Correction	63
2.8.1	The Noisy Channel	63
2.8.2	Steganography and FEC	67
2.8.3	Minimum Distance of Block Codes	68
2.8.4	Hamming Codes	69
2.8.5	Reed-Solomon	69
2.8.6	Convolutional Codes	70
3	Robust Syndrome Coding	74
3.1	Statement of the Problem	74
3.2	Robustness/Distortion Trade-off	78
3.2.1	Benchmarking the Trade-off	82
3.2.2	Using a Subset of the Cover	83
3.2.3	Reducing the Constraint Height	86
3.2.4	Increasing the message length	88
3.3	Dual Syndrome Trellis Codes	89
3.3.1	Decoding and Error Correction	90
3.3.2	Finding Optimal Values of η_s , η_m , $\mathbf{s}$, and h	91
3.4	Analysis	92
3.5	Conclusion	94
4	Irregular Syndrome Trellis Codes	103
4.1	Statement of the Problem	103
4.2	Divergence from Optimal Change Probabilities	107
4.2.1	Single-cover Single Value Divergence	107
4.2.2	Single-cover Multiple Value Divergence	108
4.2.3	Binary Entropy of Changes	108
4.2.4	Distortion	109
4.3	Irregular Syndrome Trellis Codes	110
4.3.1	Segment Based Matrices	113
4.3.2	Tiling the Matrix	114
4.3.3	η - Density of the Segment	116
4.3.4	γ - the Skew of the Matrix	118
4.4	Analysis	120
4.4.1	8×8 , $n = 64$ Cover Crops	120
4.4.2	64×64 , $n = 4096$ Cover Crops	121
4.4.3	Favicons	123
4.5	Conclusion	125

5	Simulating Suboptimal Embedding	129
5.1	Statement of the Problem	130
5.1.1	Simulating at the Rate-Distortion Bound	131
5.1.2	Coding Loss is not Equal	132
5.2	Sampling Coding Loss	133
5.2.1	Determining n_m and n_c	135
5.3	Analysis	138
5.3.1	Per Cover	138
5.3.2	Distortion Functions	140
5.3.3	Detectability	141
5.4	Conclusion	143
6	Conclusion	144
6.1	Contribution of this thesis	145
6.1.1	Robust Steganography	145
6.1.2	Irregular Syndrome Trellis Codes	146
6.1.3	Simulating Suboptimal Steganographic Embedding	147
6.2	Further Work	148
6.2.1	Further Investigation on the Construction of $\mathbb{H}$	148
6.2.2	Non-Binary Sub-Optimal Simulation	149
6.2.3	Lazy evaluation of trellis for high q value embedding	150
	Appendices	167
A	Syndrome Coding	168
B	(7,4) Hamming Codes	170

Chapter 1

Introduction

Steganography is the technique for covert communication, allowing a sender and a receiver to transmit a secret message, by embedding their message into inconspicuous objects. Hiding the message in plain sight [1], a steganographic system aims to minimise the ability of a passive warden to classify any communication as either an innocent object or a stego object. Modern steganographic systems seek to embed hidden messages in digital media such as images, digital audio, and more. Thus the primary aim of a steganographic system is to conceal the communication of a hidden message itself, yielding security through obscurity.

In this chapter, we introduce the reader to a summary of steganography: its historical uses dating back thousands of years, to more modern-day steganography, before formally defining the problem. We then move to introduce *Adaptive Steganography*, which will be the focus of this thesis, and then introduce the plan for the remainder of this thesis.

1.1 Historical Steganography

The word steganography has a Greek origin, which is derived from the Greek word ‘steganos’, translated as ‘covered’ [1, 2]. The first documented use of steganography was by Histiaeus, the exiled tyrant of the ancient Ionian city of Miletus. By shaving the head of his most trusted slave, Histiaeus was able to tattoo a message to his scalp. Once the slave’s hair had grown back and subsequently concealed Histiaeus’ message, he travelled to the city inconspicuously. When inside, the slave shaved his head, revealing the hidden message, encouraging the city’s regent, Aristagoras to revolt against the Persian rulers [1].

The word steganography was first used by Johannes Trithemius (1462-1516), in the book *Steganographia* [3]. Written in 1499, and published in Frankfurt in 1606, *Steganographia* is formed of three volumes; two on encoding messages using steganography, and cryptography, and the third, widely assumed to be on magic, is a stego object itself, containing mundane hidden messages [4, 5]. After publishing, the Catholic Church placed *Steganographia* on the *libri prohibiti*, the hidden, or prohibited books, where it sat from 1609 to 1900.

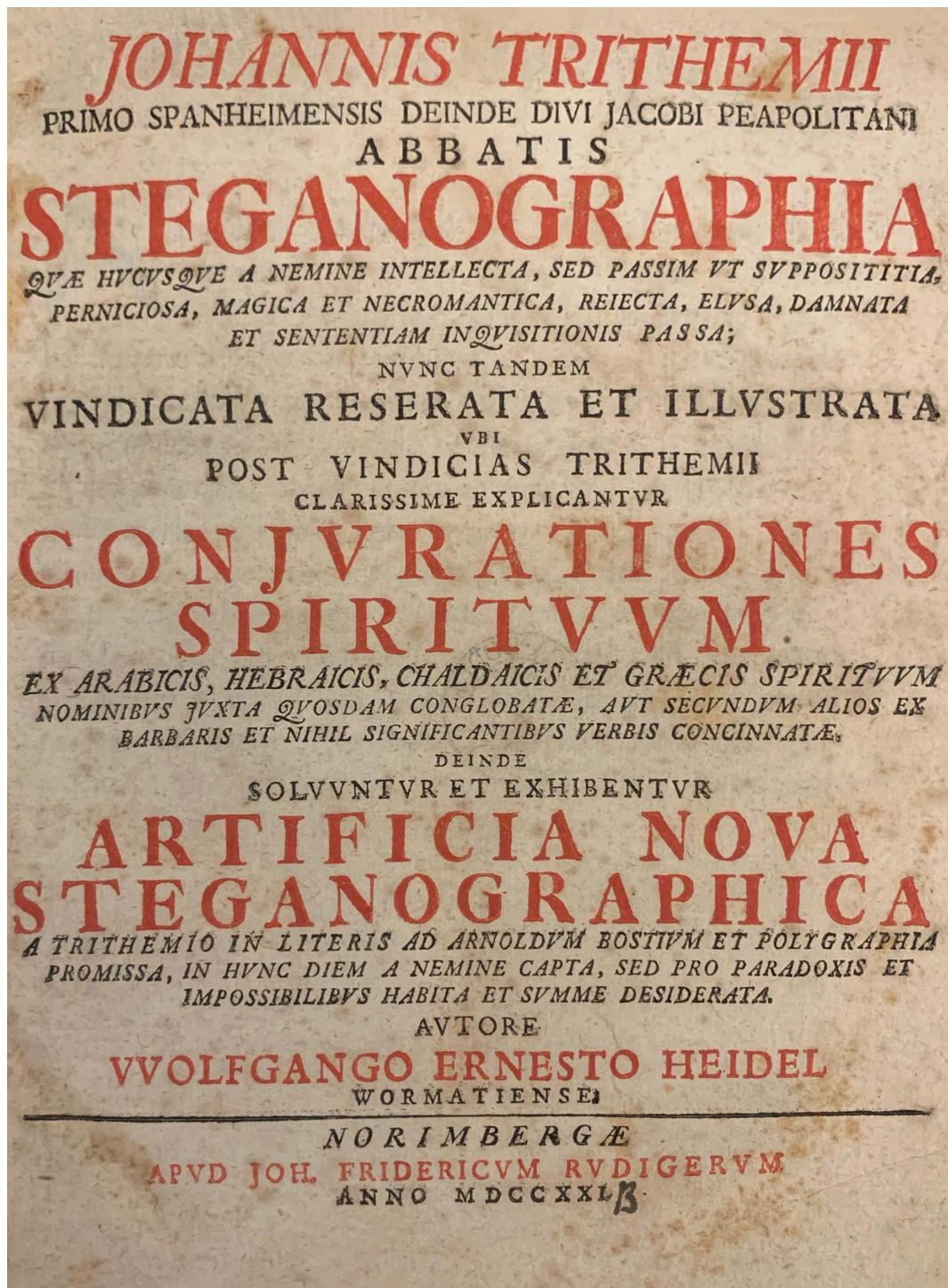


Figure 1.1: Title page of *Steganographia* by Johannes Trithemius [3]. Reproduced by kind permission of the Bodleian Libraries, University of Oxford.

The use of an acrostic is an example of linguistic steganography [6]. For example, if we took each letter in our message, and wrote a note whereby each new sentence started with the next character of our message:

Another day in paradise, here on the island. Tomorrow we set sail for the mainland to get to the festival. Today we're relaxing in the late sun. Again, the weather is fortunate. Cold days are far past. Kind regards to you all!

Taking the first letter from each sentence would return the true meaning of the message:

ATTACK.

An improvement in linguistic steganography was the *Cardan's Grille*. First developed in China, it was reinvented by Jérôme Cardan. The Cardan Grille is a mask which obscures a body of text, revealing the real message. Because there is no obvious pattern to the words or letters, it is less conspicuous to any warden, yet upon placing the grille over the note, the real message is revealed. In this case, the grille itself is the secret key [6, 1].

In World War One, the process of using *invisible ink* was developed. Spies would write sensitive letters, penned in chemical compounds, to hide the true nature of the document; these compounds would only develop, and become visible with the application of a specific developer [7]. The adversary reacted by applying broad range developers in small strokes to suspected letters, an early example of steganalysis. Later, in World War Two, microdots were introduced. Through the advances in lens and reprographic technology, minute dots could be printed to the underside of letters, or under postage stamps.

Invisible to the naked eye, these dots could be used to transmit small messages, undetected [8].

1.2 Steganography Today

The use of steganography itself, is a paradox: in the case of parties successfully using steganography, their presence goes unnoticed. Since steganography aims to hide the very evidence of the existence of the system, whether or not steganography is being actively used in modern times, is itself, a mystery.

With the growth of the internet and the practice of sharing media files becoming one of the most common uses of the internet, it would be safe to assume the conditions for steganography to be used have become increasingly favourable. Since the advent of *Web 2.0*, user participation and media sharing has sky rocketed [9]. Some 79% of internet users in the United States of America use social media platform *Facebook* [10], where media sharing is one of the most prominent use cases.

The New York Times reported that steganography might have been used in plotting U.S. embassy attacks in Paris [11]. The use case of steganography in evading state surveillance lends itself nicely; large scale surveillance is costly and challenging, the steganographer wishes to be inconspicuous. In 2010 the FBI deported suspected Russian spies, after uncovering a domestic spy network. Using steganography, they communicated over public channels, by hiding information in images [12].

A similar, but different technique to steganography, is *watermarking*. Watermarking seeks to add a visible, or invisible payload to the media, which

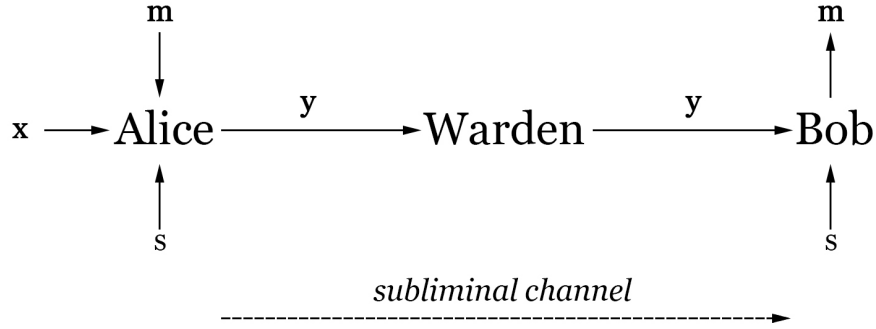


Figure 1.2: Representation of the prisoners' problem

conveys information. The existence of the watermark can be to indicate ownership or proof of authenticity [1]. The significant difference between watermarking and steganography is in steganography the adversary wins if the payload is detected, whereas in watermarking the adversary wins if the payload is erased.

1.3 Definition of Steganography

Simmons formulated steganography as the prisoners' problem, although not named as steganography but rather the subliminal channel [13]. As depicted in Fig. 1.2 it introduces three characters: Alice-sender, Bob-receiver, and the prison warden-adversary. The scenario is as follows. Alice wishes to communicate a secret message to Bob; they are allowed to communicate freely; however, the prison warden monitors all messages. If Alice and Bob openly discuss an escape plan, the warden will cut the communication channel and penalise them both. To discuss an escape plan, Alice must hide her secret message m inside the input cover x using the shared secret s , which results in the stego object y , this is passed to Bob, who can extract the hidden

message from the stego object as he also knows s .

The warden's inability to classify Alice and Bob's communication as either a cover or stego object is the inherent security of the stego system. We call the warden's classification problem *steganalysis*. In the previous scenario, we described that the warden must only classify if Alice and Bob are communicating via the subliminal channel to win. In the event of the warden detecting the use of the subliminal channel, we consider the stego system broken, whereas, in cryptography, we consider the cryptosystem broken if the adversary can extract the message. Steganalysis classification is often based on a set of rich features of a natural image: the measure of how a stego object diverges from the expected norm (measured by the affinity to the feature set) is used to classify the stego object as innocent or not [14]. Thus, the game is two-player, with Alice and Bob on one team, against the other team: the warden.

If we apply Kerckhoffs' principle to the subliminal channel, then we assume that the adversary (the prison warden) has full knowledge of Alice and Bob's steganographic algorithm, but not their shared secret keys [1]. However, unlike cryptography, the application and adherence to Kerckhoffs' principle is subtle [15]; what is the whole definition of the stego system? Whilst this thesis is principally focused on distortion minimisation, as a proxy for reducing detectability, in terms of the two-player game: Alice and Bob, and the warden, we assume the following:

1. Keys - Alice and Bob share a secret key known only to them.
2. Message length - Alice, Bob, and the warden all know the message

length.

3. Cover - Once Alice creates some stego object from a cover, the cover is destroyed and hence unavailable to the warden.
4. Cover source - Although the cover is unavailable to the warden, the warden also has access to Alice's cover source.

Principally, there are three methods in which Alice may practically embed her message into a cover: cover selection, cover the synthesis and cover modification. In this thesis, we shall only consider cover modification¹; Alice modifies an existing cover to convey her secret message, resulting in the stego object. Cover modification is the most practical approach of steganography [1], because of its versatility and ability to communicate significant payloads. With cover modification steganography, the aim is to maximise the payload size (secret message), whilst keeping our stego object appearing innocent to the adversary.

To formalise our problem mathematically, we define two functions *Emb* (the embedding function) and *Ext* (the extraction function) for a cover $\mathbf{x}$,

¹In Cover Selection, Alice utilises the rejection sampler. Alice rejects all covers, except for the cover in which the message Alice wishes to communicate, already exists [1]. Considered perfect-steganography, Alice will sample covers until a perfect cover is observed. This method can only be attempted for the smallest of messages. Using a message-digest function and the shared secret, Alice samples and rejects all covers until a cover is found where the digest matches the secret message. However, rejection-sampling requires exponential samples with respect to message length [16]; therefore, for any practical scenario, it is doubtful that Alice would be able to sample a perfect cover. Cover Synthesis is the approach of synthesising a cover that contains the hidden message itself. This could be achieved through creating a mosaic, or fractal images, which may be able to convey significant payloads, however, there is no practical solution for natural images. Further to this, it may be detectable in itself, if Alice suddenly started to send generated images, as opposed to natural images.

message $\mathbf{m}$, and shared secret s ;

$$Emb_s : (\mathbb{A}^n \times \mathbb{B}^m) \longrightarrow \mathbb{A}^n \quad (1.1)$$

$$Ext_s : \mathbb{A}^n \longrightarrow \mathbb{B}^m, \quad (1.2)$$

where the cover $\mathbf{x} \in \mathbb{A}^n$ is the n sized set of cover values. In practice, cover values could be 16-bit unsigned integer DCT values from a JPEG, 8-bit signed integer pixel intensities from a RAW image, or some other type (explained in more detail in Section 2.2). The message $\mathbf{m} \in \mathbb{B}^m$ is the m sized set of message bits, which most commonly are binary, or ternary symbols.

Emb_s takes the shared secret s , the input cover $\mathbf{x}$, and secret message $\mathbf{m}$, returning the stego object $\mathbf{y}$, which has the same length as the input cover:

$$\mathbf{y} = Emb_s(\mathbf{x}, \mathbf{m}), |\mathbf{y}| = |\mathbf{x}|. \quad (1.3)$$

The extraction function Ext_s takes the shared secret s , and the vector $\mathbf{y}$ ², and returns the secret message $\mathbf{m}$:

$$Ext_s(\mathbf{y}) = \mathbf{m}. \quad (1.4)$$

For correctness Emb_s and Ext_s both satisfy

$$Ext_s(Emb_s(\mathbf{x}, \mathbf{m})) = \mathbf{m}. \quad (1.5)$$

²In practice, the extraction and embedding functions would likely need to share the length of the payload, or use a pre-known fixed payload length [17].

1.4 Distortion

Under the assumption we must make some changes to our cover, to embed our message, it would seem logical that if we made fewer changes, it would make our stego object harder to detect. Indeed older steganographic schemes attempted to minimise the number of changes made, to embed a message. The widely used metric for comparing non-adaptive stego systems is the embedding efficiency e . Defined as the length of the message divided by the number of changes made to the stego object:

$$e = \frac{m}{\sum_{i=1}^n [x_i \neq y_i]}, \quad (1.6)$$

where the *Iverson bracket* denotes a 1 if the proposition is satisfied:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise} \end{cases}, \quad (1.7)$$

and m is the length of the message ***m*** we wish to embed. However, this scheme has a large flaw; are all changes equal? If we compared two potential changes, the first is a pixel which has low predictability but is surrounded by other pixels with low predictability; hence the area is noisy. The second case is a pixel which is again low in predictability, but surrounded by very predictable pixels, for example in a smooth area of an image; a plain blue sky or similar. The question posed by the adversary is, is this an example of a deliberate change? Or is the pixel some noisy residual caused by the camera

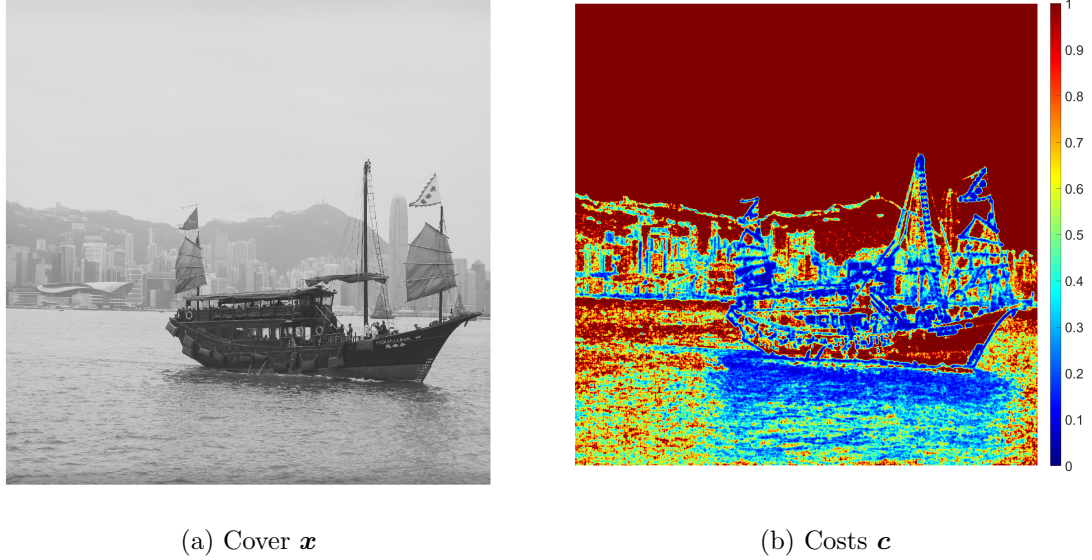


Figure 1.3: Example of a cover, and the costs generated using a distortion function.

sensor, or image compression technique? One would argue the first case is much more plausible and innocent, whilst the latter is seeming much more conspicuous. Because of this, we say that changes are not all equal. This is the foundation of adaptive steganography.

Almost all current adaptive-steganographic methods aim to minimise a user-defined distortion function $\mathbf{D}$, a distortion function provides costs of change. We assume the total distortion of a stego object is a proxy for detectability [1], with the distortion functions aiming to exploit the structure, and complexity of the input cover, providing a heuristic-based cost per element change. A distortion function takes a cover $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, and returns a set of real costs $\mathbf{c} = \{c_1, c_2, \dots, c_n\}$, $\mathbf{c} \in \mathbb{R}^n$. The distortion cost of modifying cover values x_i , resulting in the stego value y_i , is c_i , where this cost reflects the impact to detectability of the change. A higher cost signifies

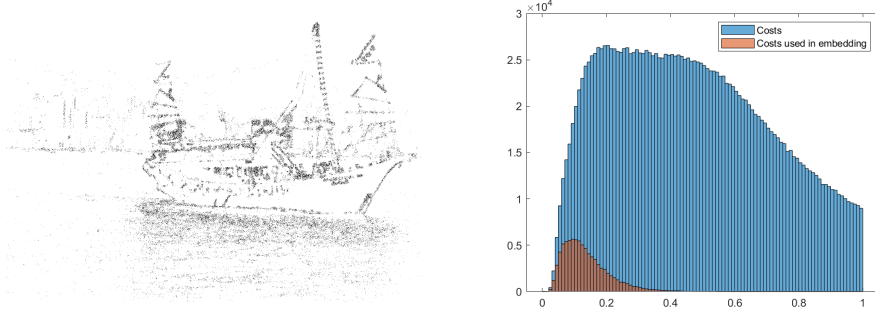
the change is more detectable than some other smaller cost. Expanding on non-adaptive steganography, we introduce the notion that not all changes are equal, and that as well minimising the overall number of changes, we wish to select the changes which introduce the least distortion: lowering the deviation from the original cover, with respect to the distortion model.

We can formalise our distortion function as:

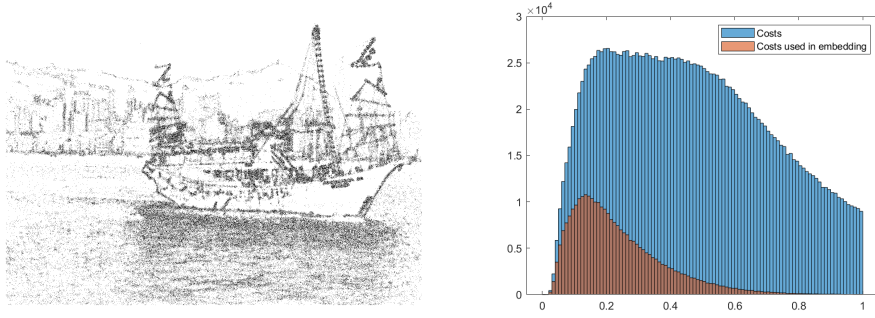
$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n c_i[x_i \neq y_i], \quad \mathbf{c} \in \mathbb{R}^n, \quad (1.8)$$

for a given cover $\mathbf{x}$ we have the set $\mathbf{y} \in \mathcal{Y}$, which is the set of all stego objects which can be generated from the input cover. The stego system which minimises the distortion $D(\mathbf{x}, \mathbf{y})$ should pick the stego object $\mathbf{y}$ which has the lowest distortion³. This format of adaptive steganography is additive, in that the sum of per-value costs are summed, which fundamentally cannot account for interacting changes. Whilst additive distortion cannot account for interacting changes, it is dramatically less computationally complex since, on each modification of a cover value, the distortion cost map must be re-generated to account for the change. In this thesis, we shall exclusively refer to the additive form of adaptive steganography.

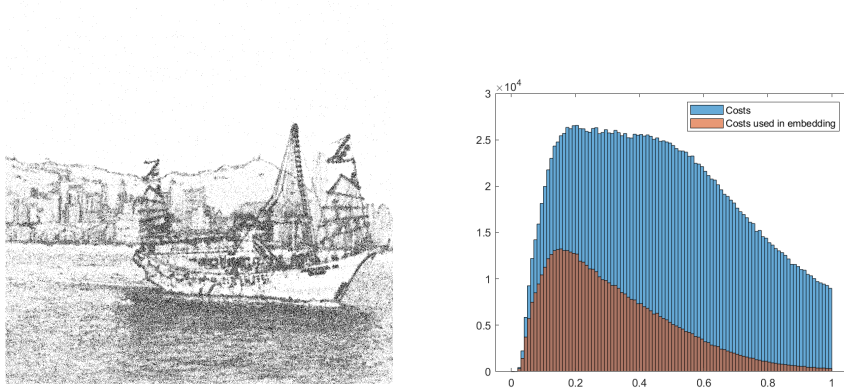
³When the costs $\mathbf{c}$ are all equal to 1, this distortion function simply maximises the embedding efficiency e , and takes the form of (1.6).



(a) Left - cover changes, right - change cost distribution, $\alpha = 0.1$



(b) Left - cover changes, right - change cost distribution, $\alpha = 0.3$



(c) Left - cover changes, right - change cost distribution, $\alpha = 0.5$

Figure 1.4: Changes made (pixel changes marked in black), and the distribution of costs associated with the respective changes.

There are three parts to an adaptive steganography system: the distortion function, the code, and the embedding function⁴. A code seeks to increase the embedding efficiency through minimising the number of changes we make to hide our message (minimising (1.8), subject to satisfying (1.5)), and distortion functions aim to give an (additive) cost associated with changing an individual cover value. The embedding operation realises these changes to the cover, resulting in the stego object. If we combine all three, we wish to use the code to minimise the total sum of distortion whilst embedding our message into the cover, resulting in the stego object. In Fig. 1.3, we show a cover, and the cost map—the result of a distortion function. We can see that higher costs are in red, these are values in which the predictability is high if we assume the warden can also determine this predictability then we want to avoid making changes here. The low-cost areas are in blue, and correspond to textured or low predictability areas of the cover, if we make changes here, the warden will be less able to distinguish the change.

As the payload rate α increases, we force the code to pick higher-cost areas to change, to embed our message in the cover. Therefore, we expect distortion to monotonically increase as the payload rate α increases (which is expected as per (1.8)). In Fig. 1.4 we show a comparison of payload rates $\alpha \in \{0.1, 0.3, 0.5\}$. As shown, the changes at lower values of α focus on the lowest cost areas, and as α increases, the code must utilise higher cost values in order to satisfy (1.5).

⁴There is a code which solves this in pseudo linear time, the Syndrome Trellis Code, as introduced in Section 2.7.

1.5 Roadmap of Thesis

We now briefly discuss the outline of the remainder of this thesis. We begin with a broad literature survey in Chapter 2, where we cover steganographic embedding, distortion functions, and coding algorithms. We discuss optimality with respect to steganography and how we can benchmark different steganographic systems. Finally, we look at *Forward Error Correcting Codes*, given they feature prominently in Chapter 3.

1.5.1 Research Topic

The overall topic of this thesis is the application of codes to adaptive steganography. Codes play an integral part of any stego system and have been instrumental to adaptive steganography. Our thesis started by exploring robust steganography, and it quickly became apparent that robustness achieved by error correcting codes was in contention with distortion minimisation of embedding codes. The desirable property of an adaptive steganography system is its ability to minimise distortion, this is achieved by considering many possible changes to the cover in order to hide the payload. However, robustness requires these possible changes to be far apart in order to have error correcting properties. The critical element here was the construction of the parity check matrix used during embedding. Because of this, our second chapter looked on how we could improve the parity check matrix generation, in order to further improve the distortion minimisation during embedding. Our benchmark for this was to consider the theoretic bound,

the rate-distortion bound. Comparing the real embedding using a code to the rate distortion bound can be done through quantifying the coding loss. When observing coding loss, it became apparent many researchers choose not to embed, but rather simulate embedding, neglecting the impact of coding loss. This prompted our third and final research chapter to look at ways we could simulate embedding, whilst mirror the impact of coding—the coding loss.

1.5.2 Overview of Research Chapters

In this thesis, we present three chapters of research, each addressing its research question relating to the application of codes to adaptive steganography. In each of these chapters, we introduce the statement of the problem, any background information the reader may be required first to understand before we introduce our method, analysis of our method, and conclusion where we discuss the impact of said research. The research chapters are as follows:

Chapter 3 - Robust Syndrome Coding In this chapter, we explore the role of robustness within steganography. As previously mentioned, in digital watermarking, robustness is critical: an adversary should not be able to remove the watermark from the media, else the watermark has failed. In steganography, we typically consider the noiseless case, in which Alice and Bob can freely communicate a stego object, or cover object, with the assumption that the object received by Bob is exactly the object sent by Alice. However, if the channel which Alice and

Bob communicate through has incidental noise, our message hidden in the stego object would be rendered invalid. Adaptive steganography by design is fragile and susceptible to even the lowest error rates on a channel. We consider the case in which a stego object can be re-compressed, which is very common for social media platforms. Because of this, the noise is non-adversarial, but it will still damage a message hidden in a stego object. Unlike digital watermarking, steganography does not require robustness by default; however, if our public channel is noisy, we must strike a balance between robustness, and distortion minimisation. Our method, *Dual Syndrome Trellis Codes*, modifies the state of the art Syndrome Trellis Code to create distortion minimised robust stego objects, which can withstand small error rates on the channel. Typically the error rate we expect to encounter is from JPEG recompression at the same quality factor. We evaluate our method as the ability to achieve robustness whilst minimising distortion. Chapter 3 is an adaptation and extension of the work presented in the paper ‘*Adaptive Steganography in the Noisy Channel with Dual-Syndrome Trellis Codes*’ at the 2018 IEEE International Workshop on Information Forensics and Security (WIFS), in which the paper won best student paper.

Chapter 4 - Irregular Syndrome Trellis Codes Syndrome Trellis Codes have become very popular with steganographers, approaching the optimum bound (*rate-distortion-bound*), with the algorithm operating in linear time, parameterised by the *constraint height*. In this chapter, we look at how the algorithm can produce *positive outliers*, cover val-

ues which are changed with a higher probability, than expected, and how the occurrence of these outliers is focused at the start and end of the cover. Positive outliers pose two problems; an attack may be formulated which seeks to exploit the existence of these outliers, and a chance which is made with a higher occurrence than the expected change probability dictated by the rate-distortion-bound dictates coding loss. By modifying the parity check matrix used during embedding, we show that we can reduce the occurrence of positive outliers. Our method *Irregular Syndrome Trellis Codes* allows for irregular parity check matrices to be used during embedding, meaning a larger parity check matrix (the equivalent of increasing the constraint height) can be used in the areas in which we expect positive outliers. The work in this chapter includes work presented at the Electronic Imaging, Media Watermarking, Security, and Forensics 2019 conference, from the paper titled ‘*Reducing Coding Loss with Irregular Syndrome Trellis Codes*’, and continued work finished after publication on Favicon steganography. Whilst our method showed improvement on small-sized covers, which allowed us to reduce distortion further and mitigate positive outliers, on larger covers, our method’s improvement was negligible.

Chapter 5 - Simulating Suboptimal Embedding Because of the complexities and cost in terms of time; many researchers choose to simulate steganographic embeddings, rather than use a code to create real objects. The cost in time is two-fold: researchers must invest significant time to establish a full steganographic system which creates stego ob-

jects, as well as the embedding process taking significantly more time than simulating. However, simulating stego objects has a critical flaw, in that coding loss is eradicated, with the object being created at the rate-distortion-bound. The argument for this is that the state of the art approaches the bound asymptotically. Our research suggests that whilst it does approach the bound, the distance to the bound (the coding loss) is dependent on a multitude of factors. These factors include; the distortion function, constraint height of the STC, and the payload rate. Therefore, comparing two distortion functions by simulating, may favour one distortion function over another. In this chapter, we propose a lightweight sampling method, which samples the coding loss dependant on the aforementioned parameters. We show that for a very small sample size, an accurate estimation of a *coding loss factor*, the amount in which we must inflate the original payload, to simulate the real-world coding loss, can be found. The result of this work allows researchers to create accurate simulations, which reflect real stego objects, both in terms of distortion and detectability. The work of this chapter was presented in a paper ‘*Simulating Suboptimal Steganographic Embedding*’ at the 2020 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSEC), in which the paper won best student paper.

Chapter 2

Background and Literature Survey

This chapter introduces the reader to the relevant work surrounding the area of this thesis. As previously mentioned, we can break an adaptive stego system into three disjoint components:

1. **Distortion functions** - Defines costs associated with changing cover values. Can be additive, or non-additive.
2. **Code** - Aims to minimise the impact of encoding the message into the cover. Minimise some user-defined distortion function.
3. **Embedding functions** - Applies changes to the cover, resulting in the stego object.

The combination of all three components forms an adaptive stego system, as shown in Fig 2.1.

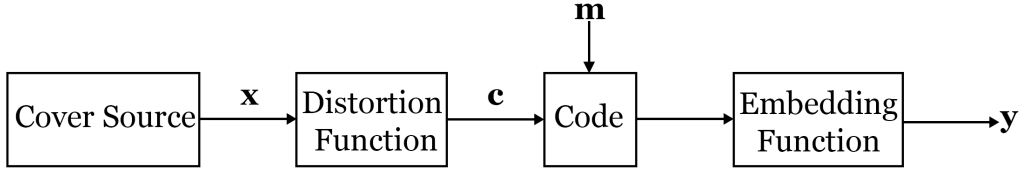


Figure 2.1: Depiction of an adaptive stego system.

Notation will be defined in Section 2.1. We first review covers, including uncompressed and compressed images, as well as text and audio-based covers in Section 2.2. We then look at embedding operations including LSB replacement/matching, F3 and F4 in Section 2.3. In Section 2.4 we introduce the reader to several distortion functions, both adaptive and non-adaptive. With distortion functions introduced, and before we explore codes, we look at the rate-distortion bound, the theoretic bound for any stego system in Section 2.5 for both additive, and non-additive distortion functions. To supplement the background on embedding functions and distortion functions, we survey legacy and state of the art steganographic coding schemes in Section 2.6 and Section 2.7 respectively. Finally, in Section 2.8 we look at error-correcting codes and the notion of robustness.

2.1 Notation

We shall denote vectors as lowercase boldface $\mathbf{x}$, with the 1st element of $\mathbf{x}$ denoted x_1 . Matrices are written in uppercase double struck; $\mathbb{H}$, indexed as $\mathbb{H}_{i,j}$.

2.2 Covers

The use of images as covers in steganography has long been the norm for several reasons; prevalence - images are one of the most common files formats transported over the internet, density - containing large amounts of discrete information, and inherent noise; images are naturally noisy. For these reasons, images are a good choice of covers for steganography. Largely these fall into two categories: spatial which we discuss in Section 2.2.1, and JPEG images which we look at in Section 2.2.2.

Outside of image steganography, there exist many other suitable formats, including some more than images. In Section 2.2.3 we look briefly at non-image steganography, including sound, video, and linguistic steganography. Work in this thesis will involve both RAW and JPEG Images; however, we include video and other cover forms as background information.

2.2.1 RAW Images

RAW images can then be further divided into two categories, raster and palette formats. Palette formats store a pre-defined (quantised) selection of up to 256 colours stored with some given precision: the palette, with each pixel intensity, then stored as a reference to one of the palette colours. This format lends itself well to images with low colour depth: vector graphics, line illustrations, computer-generated graphics, and cartoons. Raster formats are relatively simple formats; pixel intensities are stored in a grid forming the

shape of the image with a pre-defined accuracy, usually one byte per channel per pixel [1]. As such, for a greyscale image, one byte is required for each pixel intensity, with the range of values $\{0, \dots, 255\}$. Three colour (true colour) images are stored as a triple of red green blue (*RGB*) intensities, with each value again in the range $\{0, \dots, 255\}$. Standard formats for RAW images are PNG, BMP, and TIFF files, which may have their compression types. For the remainder of this thesis, we shall only use raster formats when referring to RAW images.

2.2.2 JPEG Images

JPEG (Joint Photographic Experts Group) steganography is particularly popular due to a simple reason; JPEGs are easily the most commonly used image format on the internet [18]. Because of the vast availability of input covers, it is not surprising that much of the current literature would focus on JPEG steganography. JPEGs are an ideal image format for natural images because the human eye is less sensitive to minute changes in colour intensity [1]. It is also the case, that many digital cameras, including smartphones, produce almost exclusively, JPEG images, and do not offer the Raw uncompressed image. JPEG compression can be thought of as two stages; lossy, and lossless compression. As the name suggests, lossy compression involves the permanent truncation (through quantisation) of accuracy, whilst the loss-less stage is reversible, given no information is lost [19].

The lossy stage is compromised of four stages:

1. **Colour space transformation** - The raw input pixel values are con-

verted from RGB values to the YC_rC_b colour space. Y is the luminance channel, C_r and C_b the red and blue chrominance channels respectively. A linear transform from RGB to YC_rC_b can be expressed as:

$$Y = 16 + (65.481 \cdot R + 128.553 \cdot G + 24.966 \cdot B) \quad (2.1)$$

$$C_b = 128 + (-37.797 \cdot R - 74.203 \cdot G + 112.0 \cdot B) \quad (2.2)$$

$$C_r = 128 + (112.0 \cdot R - 93.786 \cdot G - 18.214 \cdot B). \quad (2.3)$$

2. **Down-sampling and division into MCUs** - First the C_rC_b channels are down-sampled since they are less dominant on how we as humans perceive the image, than the Y channel. Starting with a 16×16 block of values, if sub sampled by a factor of 2, one 16×16 block is sampled into an 8×8 block for the C_rC_b channels, whilst the Y channel is not sub sampled. This yields 4 Y channel blocks, 1 C_r block, and 1 C_b block, commonly referred to as a 4 : 1 : 1. The 8×8 blocks are referred to as Minimum Coded Units (MCU).

3. **DCT transform.** - The resulting YC_rC_b MCUs are transformed from the spatial domain values to the transform domain, by applying a forward discrete cosine transformation. For each MCU of spatial values $S[i, j], i, j = 0, \dots, 7$, we obtain $S_{DCT}[u, v], u, v = 0, \dots, 7$ as:

$$S_{DCT} = \sum_{i,j=0}^7 \mathbb{D}[i, j; k, l] S_{DCT}[i, j], \quad (2.4)$$

where $\mathbb{D}$, the DCT coefficient matrix is defined as:

$$\mathbb{D}[i, j; k, l] = \frac{\mathbf{w}[k]\mathbf{w}[l]}{4} \cos \frac{\pi}{16}k(2i+1) \cos \frac{\pi}{16}l(2j+1), \quad (2.5)$$

and $\mathbf{w}[0] = \frac{1}{\sqrt{2}}, \mathbf{w}[> 0] = 1$. The DCT transformed values $\mathbb{S}_{DCT}$ now form a matrix in which $\mathbb{S}_{DCT}[0, 0]$ is the lowest frequency of the block, as we move away from $\mathbb{S}_{DCT}[0, 0]$ in any direction the frequencies increase, with $\mathbb{S}_{DCT}[7, 7]$ being the highest frequency of the block.

4. **Quantization** - The quantization step involves rounding and dividing each MCU in $\mathbb{S}_{DCT}$ by a quantization matrix $\mathbb{Q}_{q_f}[i, j], i, j = 0, \dots, 7$. At this stage the JPEG quality factor $q_f = 1, 2, \dots, 100$ is used to generate $\mathbb{Q}_{q_f}$

$$\mathbb{Q}_{q_f}[i, j] = \begin{cases} \max(1, 2 \cdot \mathbb{Q}_{50}[i, j] \cdot (1 - \frac{q_f}{100})), & q_f > 50 \\ \min(255, \frac{\mathbb{Q}_{50}[i, j]}{q_f}), & q_f \leq 50, \end{cases} \quad (2.6)$$

where $\mathbb{Q}_{50}$ is the JPEG standard quantization matrix. The effect of the quantization step causes the quantized DCT values to lose more information towards values in the bottom right hand corner of each MCU. Which is critical in allowing the lossless compression stages to be effective.

At this stage in the compression, steganographers may manipulate and modify the quantised DCT values of the JPEG image, as further changes are at no risk of being lost since the remaining stages of the compression are lossless. Next, the quantised DCT values are each scanned in a natural order.

The first element of the scan is the DC value, which has the most influence on the MCU's appearance; the remaining 63 values are AC values. DC values from sequential MCU's are compressed using differential pulse code modulation (DPCM), and AC values inside each MCU compressed with run-length encoding (RLE). The output of the DPCM and RLE compression is encoding using Huffman coding. In practice, most steganography algorithms avoid modifying the DC values, in particular, zero values, since they have a significantly higher effect on the MCU than modifications to AC values [20]. Again, to reduce the impact of embedding, it is common to embed only in the Y channel.

2.2.3 Video, Sound, and Other Covers

There exist several cover types outside of images, which have good attributes for steganography. Indeed many of the reasons we use images as covers are also applicable to other cover types. The density of information, the innocent nature of their widespread distribution, and inherent noise are all characteristics of Web 2.0 multimedia. As mentioned in Section 1.2, since the advent of Web 2.0, users are the largest contributor of media on the internet, with social media users encouraged to upload and share media of all types; images, video, sound, and written text.

1. **Linguistic Steganography** - Predating image steganography significantly, as mentioned in Section 1.1. A common strategy is the replacement of words which are considered synonymous, a form of cover modification. Other strategies include deleting adjectives, or re-ordering sen-

tences [21]. With the rising popularity of Twitter, a platform for users to micro-blog messages of 140 characters or less, Twitter forms a very promising platform for linguistic steganography [22]. The approach of batch-based steganography; spreading the stego payload across several covers lends itself well to Twitter [23]; many users send frequent bursts of messages.

2. **Video** - Significantly more dense than images, video files typically have larger compression ratios, and subsequently, more noise, given file size constraints. Typically video steganography falls into two categories; embedding into the pixels of individual frames of video, or embedding into the motion vectors between frames of video [24, 25]. Methods from image steganography can apply to video steganography, with the STC algorithm being used to embed into H.264 macroblock partitions of videos [26] or the use of wavelet decomposition for embedding [27]. Perhaps the most inhibiting factor of video steganography is the computational complexity of video editing.
3. **MP3** - There has been an influx of lossless audio sharing sites, allowing users to upload their own music, or remixes, without transcoding or modifying the uploaded media. Because of this, these platforms form ideal channels for audio steganography [28]. The MP3 encoding format proposed by the Moving Pictures Expert Group (MPEG) uses partial discarding of inaudible frequencies, and approximations to compress audio with limited human perception. It is comprised of psychoacoustic analysis, sub-band filtering, and MDCT transform and quantiza-

tion [29]. The psychoacoustic model shows that human perception to differing audio frequencies is not evenly split, as such some bands of frequencies dominate others. A commonly used method to embed into MP3 files is to use phase coding [29, 30]. Since human hearing is more sensitive to amplitude rather than phase, slight changes in the phase of an audio file are rendered unnoticeable to humans [30].

4. **VoIP** - Whilst music streaming services strive for high-quality audio streams, the area of *Voice over Internet Protocol* (VoIP) services utilise low bit rate audio streams [31]. The reasoning for this is simple, in a question of priority, users prefer to be able to communicate in real-time, over audio quality. As such, low bit-rate codecs such as *Internet Low Bit-Rate Codec* (iLBC), *G.723.1*, or *G729.A*, are used [32]. However, this poses a challenge, in steganography, it is typically desirable to have a high-density cover source, where it is likely the use of high compression ratios will be required, resulting in lots of cover entropy. Because of this, a typical LSB modification of cover values would not be suitable [32]. One of the most promising methods of VoIP steganography is *Quantization Index Modulation* (QIM), which modifies the quantisation of coefficients, during encoding. [33].

2.3 Embedding Operations

To realise any change to the cover, in order to encode a message, an embedding strategy must be in place. With the covers as described in Section 2.2 we now look at how these covers can be manipulated to encode a message.

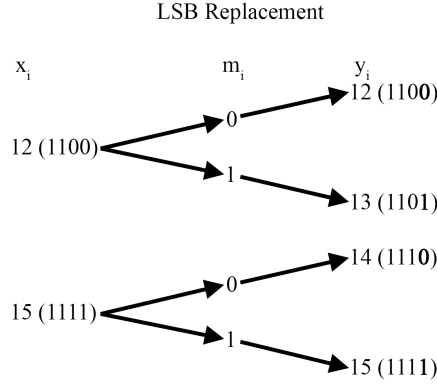


Figure 2.2: LSB modification under LSB Replacing.

As mentioned previously, embedding forms one of the three core components of an adaptive steganographic system. However, before the adoption of adaptive steganography, embedding strategies provided security from the adversary by spreading the payload within the stego object. In this section, we first introduce LSB Replacement in Section 2.3.1, of the earliest, and primitive forms of embedding into the least significant bits of covers. We then look at LSB Matching, which mitigated an easily identifiable trait of LSB Replacement, in Section 2.3.2. LSB Matching, also called ± 1 embedding, allows for ternary embedding and will be used in this thesis as the primary embedding operation.

2.3.1 LSB Replacement

One of the earliest, and most commonly used digital steganographic methods is least significant bit (LSB) embedding [1, 2]. Where the cover $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$ is a sequence of integers, e.g. the pixel intensities in a spatial image, or the DCT coefficients in a JPEG image. For a cover with n_b -bit

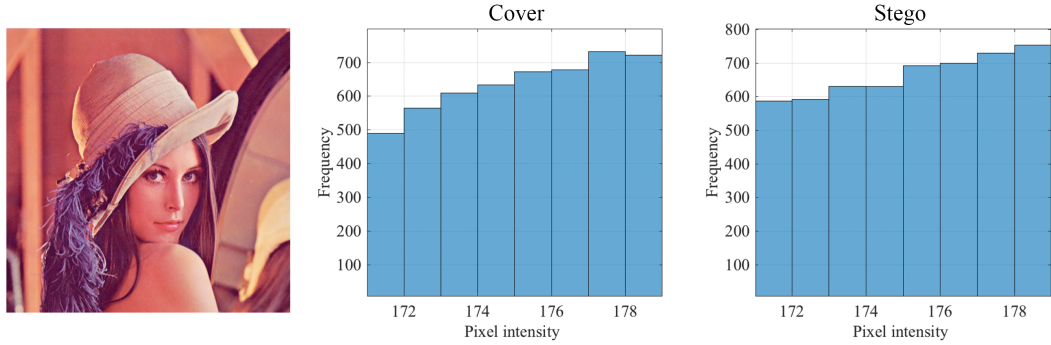


Figure 2.3: From left: original cover, histogram of pixel intensities before, and after LSB replacement.

accuracy, the cover value x_i is in the range $0 \leq x_i \leq 2^{n_b-1}$. We denote the binary representation of x_i as $x_i = \{b_1, b_2, \dots, b_{n_b}\}$, with the LSB being b_{n_b-1} . In the case of a 8-bit image, a cover values $x_i = \{b_1, b_2, \dots, b_7\}$, and $\text{LSB} = b_7$. To make modifications to the LSB we flip the LSB when required:

$$\text{LSBFlip}(x_i) = x_i + 1 - 2 \pmod{2(x_i)}. \quad (2.7)$$

We can now express LSB Replacement embedding as:

$$\mathbf{y} = \sum_{i=1}^m y_i = \begin{cases} \text{LSBFlip}(x_i) & \text{iff } \text{mod } 2(x_i) \neq m_i \\ x_i & \end{cases}, \quad (2.8)$$

an example can be seen in Fig. 2.2. The use of a shared secret aids security by creating a pseudo-random walk of the cover. Instead of a naive sequential embedding which would focus the embedding in one part of the image, the changes would be distributed across the cover if $m < n$ [34].

Whilst the minor modification of the least significant bits are unnoticeable by humans [1], LSB replacement modifications are apparent when targeted by

steganalysis attacks. This method introduces an easily identifiable artefact, as it groups pairs of pixels together, over a range of v values the pairs coefficients of value $\{0, 1\}, \{2, 3\}, \dots \{v-1, v\}$ will equalise in their frequency [1]. In Figure 2.3, we see how the pixel intensities group together into pairs as predicted. Because of the embedding operation, this effect has been widely exploited to detect LSB replacement [34].

2.3.2 LSB Matching (± 1)

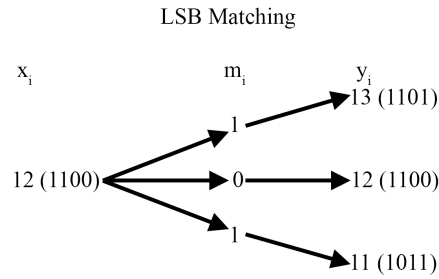


Figure 2.4: LSB modification under LSB Matching.

Whilst in the case of replacement, the LSB of a cover value is replaced with the next bit of the message, under LSB Matching, we randomly increase or decrease the cover value in the event of a flip, since both the cover value ± 1 has the same LSB as the message bit. LSB Matching is also referred to as ± 1 embedding. In Fig. 2.4 we show how this works in practice. The simple change to LSB matching somewhat reduces the effect as highlighted in the previous section, by equalising the spread of values which the stego pixels can take [1].

± 1 embedding can easily be changed to allow embedding in the ternary space. If each cover value x_i is expressed by its $\bmod 3$ representation, then

the three embedding operations for ± 1 embedding $(-1, 0, +1)$ can be used to express the three ternary symbols $(0, 1, 2)$ [35].

2.4 Distortion Functions

In this section, we look at the state of the art distortion functions in both the spatial and JPEG domains. Any distortion function aims to proxy detectability, to differentiate more detectable changes from less detectable changes by their relative cost. Of course, if we had perfect knowledge of the statistical distribution of our cover source, then the need for a distortion function would be redundant: we could sample the cover source to find a stego object which conveys our message [36]; however, this would require a number of samples exponential in the length of the payload. This method would provide perfectly secure steganography since, by definition, the stego object would be indistinguishable from the cover source distribution. However, natural digital images are non-stationary, and any empirical cover source is incongniseable [37]. Since the cover source model is unknown to both the steganographer and the warden, we must use an approximation of the cover source [38]. Instead, we must use distortion functions which are reliant on minimising some user-defined empirical statistical detectability [39].

One option available to the steganographer is to utilise side information. As the name suggests, side-informed distortion functions utilise side information available to the sender, to generate a set of costs. Similarly, a non-side informed distortion function does not use side-information. The side information, if used, is not required by the receiver to decode the stego object,

an example of side information may be access to the full raw image (before any compression) [38]. Many side-informed JPEG distortion functions utilise the quantisation and rounding errors introduced during JPEG compression [40, 41, 20, 42], only available to the sender, if the steganographer has the raw uncompressed image. Side-informed distortion functions generally out-perform their non-side informed counterparts, and many distortion functions are available in both side-informed and non-side-informed versions.

In this section, we introduce the reader to non-adaptive and adaptive distortion functions. Work in this thesis is limited to the following distortion functions; WOW, UNIWARD, J-UNIWARD, MiPOD. Other relevant distortion functions include, but are not limited to Block Entropy Distortion (EBS) [42], Uniform Embedding Distortion (UED) [40], High-pass, Low-pass, Low-pass (HiLL) [43], and Clustered Modification Directions for Color Components (CMD-C) [44].

2.4.1 Non-adaptive

As mentioned in Section 1.4, before any adaptive distortion functions were introduced the goal was to minimise the number of embedding changes. We can express this as:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n [x_i \neq y_i]. \quad (2.9)$$

However, if we wanted to incorporate the magnitude of the change between x_i and y_i , then we could use the form:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|^\gamma, \quad (2.10)$$

where $\gamma \geq 1$ parameterises the distortion function. For example, $\gamma = 1$ is the L_1 norm:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|, \quad (2.11)$$

and $\gamma = 2$ is the L_2 norm:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|^2. \quad (2.12)$$

Non-adaptive distortion functions pre-date any other distortion functions which follow the distortion minimisation paradigm and were used exclusively during the introduction of Syndrome Trellis Codes [45].

2.4.2 WOW - Wavelet Obtained Weights

The Wavelet Obtained Weights distortion function, or *WOW* [46] is a model-free distortion function based on the directional residuals computed from a set of high-pass filters. These high-pass filters compound how predictable the resulting pixel value (after embedding) is, with respect to its neighbour values. When the predictability of a pixel is low in every direction (e.g. textured areas) the resulting cost is low. Whereas when the pixel values

are predictable in at least one direction, the cost of change is high. Each direction is assessed using the filter set, $\mathcal{B}_n = \{\mathbf{K}^{(1)}, \dots, \mathbf{K}^{(n)}\}$ containing n high-pass L_2 normalised directional filters. For a cover $\mathbf{x}$, the i^{th} embedding suitability is calculated as

$$\xi_{ij}^{(k)} = |\mathbf{R}^{(k)}| \star |\mathbf{R}^{(k)} - \mathbf{R}_{[ij]}^{(k)}|, \quad (2.13)$$

where $\mathbf{R}^{(k)}$ is the k^{th} residual $k = 1 \dots n$, $\mathbf{R}^{(k)} = \mathbf{K}^{(k)} \star \mathbf{X}$, and $\star$ is a mirror-padded convolution. The embedding suitabilities $\xi_{ij}^{(k)}, k = 1 \dots n$ are then aggregated to additive costs using the reciprocal Hölder norm:

$$c_i^{(p)} = \left(\sum_{k=1}^n |\xi_{ij}^{(k)}|^p \right)^{-\frac{1}{p}}, \quad (2.14)$$

for a given p . Holub et al. found experimentally the value $p = -1$ to be best.

2.4.3 UNIWARD - UNiversal WAVElet Relative Distortion

Proposed by Holub et al., UNiversal WAVElet Relative Distortion, *UNIWARD* [20] is a wavelet-based distortion function which derives costs in a similar fashion to WOW. However, unlike WOW, UNIWARD computes relative costs and hence is suitable to be used in the arbitrary domain. A cost is higher if the value is more predictable, in the cover, with respect to nearby coefficients. Predictable areas, such as smooth sections of an image (for example a clear blue sky), carry a higher cost, than those of a noisy area. Based on high and low pass wavelet decomposition filters denoted

h and g respectively, based on horizontal, vertical, and diagonal kernels; $\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \mathbf{K}^{(3)}$ used to measure the relative smoothness of the coefficients in one direction, where

$$\mathbf{K}^{(1)} = h \cdot g^T, \mathbf{K}^{(2)} = g \cdot h^T, \mathbf{K}^{(3)} = g \cdot g^T. \quad (2.15)$$

By computing the directional residuals for a pair of pixels u, v in the spatial domain $W^{(k)} = \mathbf{K}^{(k)} \star \mathbf{x}$. Given a cover $\mathbf{x}$, stego object $\mathbf{y}$, and the stabilising constant $\sigma > 0$ we compute the distortion as

$$D(\mathbf{x}, \mathbf{y}) \triangleq \sum_{k=1}^3 \sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \frac{|W_{uv}^{(k)}(\mathbf{x}) - W_{uv}^{(k)}(\mathbf{y})|}{\sigma + |W_{uv}^{(k)}(\mathbf{x})|}, \quad (2.16)$$

where $\mathbf{x}$ is the input image of dimensions $n_1 \times n_2$. We then compute the additive distortion values c_i as

$$c_i \triangleq D(\mathbf{x}, x_{\sim i} y_i), \quad (2.17)$$

where $x_{\sim i} y_i$ denotes the cover image $\mathbf{x}$ with the i^{th} value changed, and $D(\mathbf{x}, \mathbf{x}) = 0$. Holub et al., originally proposed using $\sigma = 2^{-6}$ for S-UNIWARD [20], however, in later work, a value of $\sigma = 1$ was proven to be more secure, and avoided S-UNIWARD being too sensitive to content [47].

2.4.4 J-UNIWARD - JPEG UNIWARD

J-UNIWARD is the JPEG distortion function based on UNIWARD. Unlike most other JPEG distortion functions, J-UNIWARD supports embedding

into all DCT values, including zero value DCT values, and DC values. To adapt the previously mentioned distortion function in Section 2.4.3 for JPEG images, we decompress the cover and stego object $\mathbf{x}, \mathbf{y}$, into the spatial domain, denoted as $J^{-1}(\mathbf{x}), J^{-1}(\mathbf{y})$, and then apply the UNIWARD distortion function as:

$$D(\mathbf{x}, \mathbf{y}) \triangleq D(J^{-1}(\mathbf{x}), J^{-1}(\mathbf{y})). \quad (2.18)$$

The above distortion function is non-additive, since any change to a DCT value will affect not only the other values in the DCT block, but other values in the same wavelet direction. To use J-UNIWARD as an additive distortion function, we associate the distortion of changing a single value as the change in distortion between the cover and stego object. Distortion costs are generated for all values $x_I \in \mathbf{x}$ for $y_i = x_i \pm 1$, such to modify the i^{th} bit of a cover, leaving all other values we use

$$\rho(\mathbf{x}, y_i) \triangleq D(\mathbf{x}, \mathbf{x}_{\sim i} y_i), \quad (2.19)$$

where $\mathbf{x}_{\sim i} y_i$ denotes the cover image $\mathbf{x}$ with the i^{th} coefficient changed, and $D(\mathbf{x}, \mathbf{x}) = 0$.

2.4.5 MiPOD - Minimising the Power of Optimal Detector

MiPOD, proposed by Sedighi et al. is a model-based distortion function for spatial domain images [48]. By modelling the image noise residuals as a se-

quence of independently quantized Gaussian variables, with varying variance, MiPOD generates a per-element distortion cost based upon the impact to the cover. First the *deflection coefficient* ϱ_i for the i^{th} pixel value is computed as

$$\varrho_i = 2 \sum_{i=1}^n \sigma_i^{-4} \beta_i^2, \quad (2.20)$$

where σ_i^2 is the variance, and β_i the change rate for the i^{th} pixel. The change rates $\beta_i, i = 1 \dots n$ can be found by solving the modified PLS constraint:

$$\beta_i \sigma_i^{-4} = \frac{1}{2\lambda} \ln \frac{1 - 2\beta_i}{\beta_i}, \quad i = 1, \dots, n, \quad m \leq \sum_{i=1}^n H(\beta_i). \quad (2.21)$$

The per element cost c_i can then be expressed as:

$$c_i = \ln \frac{1}{\beta_i - 2}. \quad (2.22)$$

Hence MiPODs per-cover value cost c_i is directly influenced by the associated change rate β_i , instead of a heuristic cover model.

2.5 Optimality

For any steganographic system, it is useful to benchmark the performance of the code. The rate-distortion bound is a term used to express the relationship between the communication rate of a channel and distortion [1]. Steganographic codes which communicate on the rate-distortion bound cannot be improved, and are said to be optimal. In reality, we approach the rate-distortion bound asymptotically.

In the distortion minimisation framework, there exists two paradigms: additive, and non-additive distortion functions. In additive distortion functions, costs are static and derived before any changes are made. Fundamentally, this means additive costs cannot account for interacting changes [49]. Because STCs are known to approach the rate-distortion bound asymptotically, and that non-additive distortion functions can be relatively easily expressed in additive form [43], additive distortion functions remain the most popular amongst researchers. Nevertheless, most adaptive steganography schemes cluster changes close together [50], which would seem to suggest that additive distortion functions should be more easily detectable. In fact, the opposite is true at least for now [51], although this may be down to limitations in coding for non-additive distortion minimisation.

In this section, we derive the rate-distortion bound for additive distortion functions, and give two measures for a performance of a code (coding loss) in terms of entropy, and distortion which will be used throughout this thesis.

2.5.1 Additive Distortion Functions

For a cover $\mathbf{x} = \{x_1, \dots, x_n\}$ an additive distortion function D gives the additive change costs $\mathbf{c} = \{c_1, \dots, c_n\}$; changes are binary and independent. Suppose that we communicate a message with entropy m bits. The rate-distortion bound arises from the solution to the Payload-Limited Sender (PLS) optimisation problem:

$$\underset{p_1, \dots, p_n}{\text{Minimise}} \quad d = \sum_{i=1}^n p_i c_i, \quad (2.23)$$

such that

$$m \leq \sum_{i=1}^n H(p_i), \quad (2.24)$$

where H is the binary entropy function:

$$H(x) = -x \log_2 x - (1 - x) \log_2 (1 - x), \quad (2.25)$$

and then p_i is the optimal probability of changing x_i . PLS embeds a fixed message $\mathbf{m}$ into an arbitrary cover $\mathbf{x}$, whilst aiming to minimise the distortion $D(\mathbf{x}, \mathbf{y})$. Alice wishes to communicate her existing message, by embedding it into a cover, with the minimum distortion. Throughout this thesis, we denote the minimum objective d by $PLS(\mathbf{c}, m)$, where $PLS: \mathbf{covers} \times \mathbb{R} \rightarrow \mathbb{R}$.

The same probabilities are the solution to the dual Distortion-Limited Sender (DLS) problem:

$$\underset{p_1, \dots, p_n}{\text{Maximise}} \quad h = \sum_{i=1}^n H(p_i), \quad (2.26)$$

such that

$$\sum_{i=1}^n p_i c_i \leq d, \quad (2.27)$$

where d a distortion budget; in this case the aim is to communicate the maximum entropy. DLS although less common than PLS [45], is useful for batch-steganography, and is a more content aware form of embedding. DLS allows Alice to communicate with a fixed ceiling of distortion. DLS max-

imises the payload length that can be communicated with a pre-determined acceptable amount of distortion d . We denote the maximum objective h by $DLS(\mathbf{c}, d)$, where $DLS: \mathbf{covers} \times \mathbb{R} \rightarrow \mathbb{R}$.

The solution to (2.23) and (2.27) is

$$p_i = \frac{e^{-\lambda c_i}}{1 + e^{-\lambda c_i}} \quad (2.28)$$

for some λ such that the payload- or distortion-constraint is equality. λ can be found by conducting a binary search [1].

The maximum payload we can communicate is $H(\mathbf{p})$, where $\mathbf{p}$ is some embedding probabilities. However, since we know this is an upper-bound, any real stego system can only approach this.

2.5.2 Coding Loss

We might hope that the distortion realised by some code $Code(\mathbf{x}, \mathbf{m})$, where $Code: \mathbf{covers} \times \mathbf{messages} \rightarrow \mathbb{R}$ returns the distortion cost of embedding is close to the bound, i.e.

$$Code(\mathbf{x}, \mathbf{m}) \approx PLS(\mathbf{x}, m), \quad m = |\mathbf{m}|. \quad (2.29)$$

Indeed, for some $\mathbf{m}$ the code distortion can even be lower (for example a cover which already happens to contain message $\mathbf{m}$). But we typically witness *coding loss*: in the case of PLS, the code distortion exceeds the bound; in the case of DLS, the code can only embed a message shorter than m . Whilst we commonly see steganographic coding loss expressed as the *entropy coding*

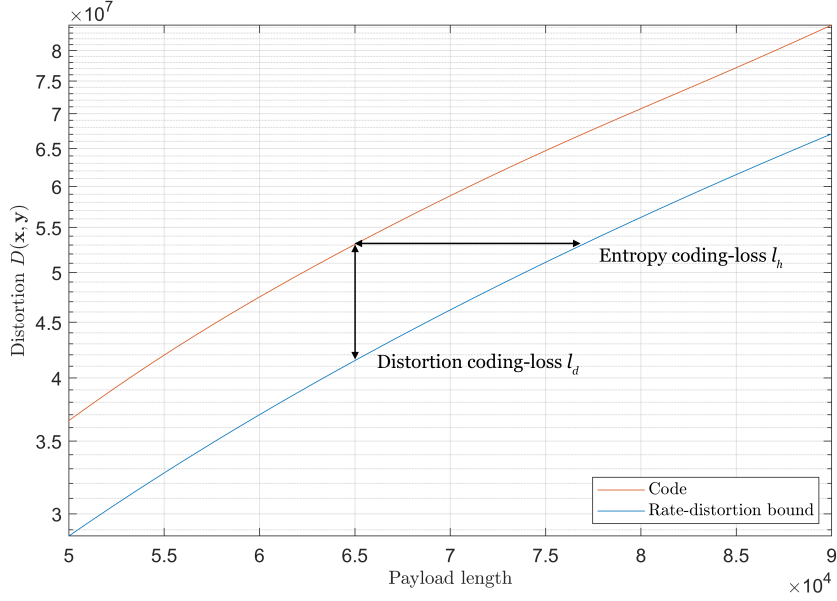


Figure 2.5: Illustration of the rate-distortion bound, and a code, showing coding loss as both l_d - distortion coding loss, and l_h - entropy coding loss.

loss l_h [45, 52, 40]

$$l_h = 1 - \frac{m}{DLS(\mathbf{x}, Code(\mathbf{x}, \mathbf{m}))}, \quad (2.30)$$

for the purpose of indicating loss on a distortion level, we also will use a slight variant, *distortion coding loss* l_d :

$$l_d = \frac{Code}{PLS(\mathbf{x}, m)} - 1. \quad (2.31)$$

In Fig. 2.5, we show coding loss as the gap between the rate-distortion bound and the distortion achieved by some code. The distance between the two series is the coding loss; vertical for distortion—the amount of additional distortion for some fixed payload length, and horizontal—the amount of en-

tropy we lose, when using a code, in contrast to the rate-distortion bound.

2.6 Legacy Codes

As mentioned earlier, an adaptive steganography system can be thought of as three problems: finding the probabilities of change for each cover element (defining the per element change distortion measure), using a code to minimise distortion whilst encoding a message, and finally using an embedding method to realise the changes to the cover. In this section, we will look at the second problem only, how do we minimise a user-defined distortion function.

As stated in Section 1.4, since we must make modifications to our cover in order to convey our message, and we wish to minimise our detectability, it is obvious we want to minimise the number of changes we make, to convey our payload. By using a code, we aim to maximise e by reducing the number of changes per the length of the payload. Note how the embedding efficiency is merely minimising the number of embedding changes, as previously denoted in (1.6).

If we first consider not using a code, then at worst, we must make the length of the payload number of changes to the cover. In the binary space, if we assume the binary representation of the cover and message is uniformly distributed, then on average we would expect to make $\frac{1}{2}$ changes per cover value. If we embedded two message values into two cover values, we would have the expected change of $(1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2}) = 1$. Then we have an embedding

efficiency:

$$\frac{m}{\sum_{i=1}^n [x_i \neq y_i]} = \frac{2}{1} = 2. \quad (2.32)$$

In this section, we introduce the reader to various coding techniques which pre-date the current state of the art. Whilst this thesis does not use these codes, they contain useful background to aid the reader to understand the state of the art, which is described in the next section. We start with generalised matrix embedding, before introducing F5 and nsF5—some of the first uses of matrix embedding in steganography¹.

2.6.1 Matrix Embedding

Crandall [53] introduced the notion of matrix encoding as a code to improve the embedding efficiency. An example of matrix embedding would be to split the input cover $\mathbf{x}$ into vectors of length three, and the secret message $\mathbf{m}$ into vectors of length two. Thus, to embed the message sub-vector into an input cover sub-vector, the following scheme is used;

$$\mathbf{y} = \begin{cases} \{x_1, x_2, x_3\} \leftrightarrow (m_1 = x_1 \oplus x_2) \wedge (m_2 = x_2 \oplus x_3) \\ \{\overline{x_1}, x_2, x_3\} \leftrightarrow (m_1 \neq x_1 \oplus x_2) \wedge (m_2 = x_2 \oplus x_3) \\ \{x_1, \overline{x_2}, x_3\} \leftrightarrow (m_1 = x_1 \oplus x_2) \wedge (m_2 \neq x_2 \oplus x_3) \\ \{x_1, x_2, \overline{x_3}\} \leftrightarrow (m_1 \neq x_1 \oplus x_2) \wedge (m_2 \neq x_2 \oplus x_3) \end{cases}. \quad (2.33)$$

¹F5, nsF5, and MMx are complete steganographic systems, and not just codes. However, they pre-date the distortion minimisation framework, and their codes are good early examples, so we include them here.

With this scheme, we are now embedding two message bits from our secret message into three elements of our input cover. Although this may seem wasteful in terms of cover utilisation, we see that there is no situation in which more than a single modification of the three cover coefficients are changed. The situation where no changes are required to $\mathbf{x}$ occurs with probability $\frac{1}{4}$. Therefore, we have an expected number of changes $(0 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4}) = \frac{3}{4}$, and thus we have an embedding efficiency:

$$e = \frac{m}{\sum_{i=1}^n [x_i \neq y_i]} = \frac{2}{3/4} = 2.67. \quad (2.34)$$

Matrix embedding increases our embedding efficiency over not using a code. Its efficiency improvements come from the scheme used to minimise the overall amount of changes incurred to the input cover. Critically, the code also distributes the changes over the values, rather than focusing the changes in the sequential order the cover values are visited in. We can see that in matrix encoding, no single stego value contains the individual message values; thus the message values are contained in both the stego values and their position, i.e., the ordering of sub-vectors from the input cover $\mathbf{x}$. Matrix, or syndrome embedding has the common notation $(1, l, p)$, whereby we denote embedding p bits of payload into $l = 2^p - 1$ bits of cover, whilst making 1 change [1]. For the receiver to extract the message each cell is multiplied by the matrix $\mathbb{H}$:

$$\mathbb{H} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \mathbf{m}' = \begin{bmatrix} m_1 = ((x_1 \cdot 1) + (x_2 \cdot 1) + (x_3 \cdot 0)) \bmod 2 \\ m_2 = ((x_1 \cdot 0) + (x_2 \cdot 1) + (x_3 \cdot 1)) \bmod 2 \end{bmatrix}. \quad (2.35)$$

Over time the literature changed from referring to this method as matrix embedding and moved to syndrome embedding. Since, for an arbitrary sized matrix $\mathbb{H}$, we are ultimately solving the condition $\mathbb{H}\mathbf{y} = \mathbf{m}$.

The use of binary hamming codes for syndrome (matrix) embedding, allows Alice to embed differing length payloads into her cover, by splitting her message $\mathbf{m}$ into sub-vectors of length p , and our input cover $\mathbf{x}$ into sub-vectors of length $l = 2^p - 1$. Denoted $(1, l, p)$, we are able to embed $\mathbf{m}$ into $\mathbf{x}$ using a $p \times 2^p - 1$ binary matrix $\mathbb{H}$ (the shared secret), constructed of $2^p - 1$ columns of non-zero binary p -tuples.

We wish to modify the $2^p - 1$ elements of $\mathbf{x}$, so that $\mathbb{H}\mathbf{y}$ (the syndrome of $\mathbf{y}$) satisfies $\mathbb{H}\mathbf{y} = \mathbf{m}$. To do this, first Alice computes $\mathbb{H}\mathbf{x} = \mathbf{m}$, if this condition holds, then Alice communicates the input cover $\mathbf{x}$ as the stego object requiring no changes. If the condition does not hold, then Alice calculates $\mathbb{H}\mathbf{x} - \mathbf{m}$ to find the resultant column in $\mathbb{H}$ which needs to be removed from the syndrome. Finally, to solve $\mathbb{H}\mathbf{x} = \mathbf{m}$, we flip the i^{th} value of $\mathbf{x}$, which corresponds to the column we wished to remove from the syndrome. This results in the stego object $\mathbf{y}$, which Alice can communicate, as $\mathbb{H}\mathbf{y} = \mathbf{m}$ [1]. For a worked example of Syndrome Coding, see Appendix A.

We make a change with probability $1 - \frac{1}{2^p}$, giving us an embedding effi-

Message size (p)	Cover elements ($2^p - 1$)	α	e
1	1	1.000	2.000
2	3	0.667	2.667
3	7	0.429	3.429
4	15	0.267	4.267
5	31	0.161	5.161
6	63	0.095	6.095
7	127	0.055	7.055
8	255	0.031	8.031
9	511	0.018	9.018
10	1,023	0.010	10.010

Table 2.1: Syndrome embedding efficiency

ciency

$$e = \frac{m}{\sum_{i=1}^n [x_i \neq y_i]} = \frac{p}{1 - 1/2^p}. \quad (2.36)$$

For a message $\mathbf{m}$ of length 3, we would have an embedding efficiency $e = 3.429$ [1].

Whilst maximising e we also must maintain α , the embedding ratio, defined as

$$\alpha = \frac{m}{n}. \quad (2.37)$$

Given that syndrome embedding allows for a variable-length payload, we see that in Table 2.1 that increasing p yields an increasing embedding efficiency, whilst simultaneously the embedding ratio α decreases. The sender would need to determine their desired α rate, then split their cover, and message into the respective p , and $2^p - 1$ sub-vectors, and embed the $\frac{m}{p}$ syndrome

operations. It may appear at first that syndrome embedding provides a large increase on embedding efficiency over matrix embedding or not using a code, however, in order to convey a reasonably sized message, the α required to do so would quickly force the embedder into selecting a p value with a lower embedding efficiency.

2.6.2 F5 and nsF5

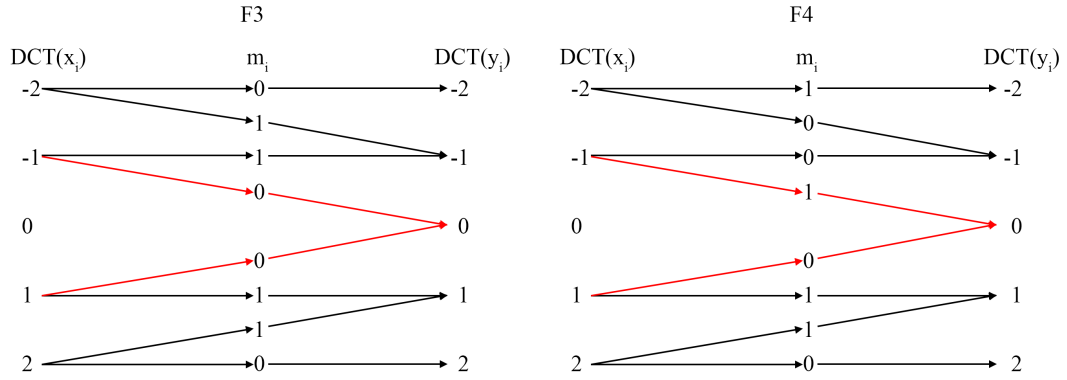


Figure 2.6: F3 (left) and F4 (right) DCT value modification, shrinkage embedding operations are in red.

F5 [54] was the first practical implementation of matrix embedding in steganography. *F5* was an improvement on the *F4* and *F3* algorithms [54], which used DCT modification strategies shown in Fig. 2.6. While a χ^2 test for overwritten bits would not detect the changes made by *F3* [55], *F3* had problem with *shrinkage*, decreasing DCT values of -1 and 1 result in the DCT value 0: the receiver cannot determine if the 0 was the result of an embedding change, or existed before embedding. In the event of a 0 DCT value from embedding, the same message bit is re-embedded until a resulting DCT value is non-zero. *F4* improved on *F3* by removing shrinkage, and making bit flips

symmetrical [55].

To perform the F5 embedding we first estimate the capacity c of the cover as:

$$c = n_{AC} - h[0] - \frac{(h[1] + h[1])}{2}, \quad (2.38)$$

where n_{AC} is the number of AC DCT values in the cover, and $h[i]$ is the number of AC DCT values $= i$. We then find the $(1, n, k)$ code which provides a suitable embedding rate $R(k)$ to encode our message in the cover, where $n = 2^k - 1$, $k > 0, k \in \mathbb{Z}$:

$$R(k) = \frac{k}{n} = \frac{1}{n} \cdot \text{ld}(n + 1) = \frac{k}{2^k - 1}. \quad (2.39)$$

For example, for a cover with $c = 1024$, and message of length $m = 200$, the required embedding rate is 5.12%. Iterating over values of k , we can find that $k = 7$ has a high enough embedding rate to encode our message, therefor the $(1, 127, 7)$ code is used. Using a PRNG seeded with the shared secret, the message is XOR-ed with a randomly generated binary stream, and a random walk of the cover values is taken, excluding DC and AC values equal to 0 skipped. Finally, for each k message bits, we take the next $2^k - 1$ cover values along the random walk and matrix encode using the $(1, n, k)$ code. If shrinkage occurs, the same k message bits are encoded into the next $2^k - 1$ cover values.

Whilst F5 was a significant step forward for steganography, it was shown to be insecure [56]. By cropping the JPEG stego object, and re-compressing

using the same quantisation matrices, an estimation on the number of modified non-zero AC DCT values can be found. To alleviate this, a new algorithm no-shrinkage F5 (*nsF5*) was designed to eliminate the shrinkage introduced during embedding [57]. To do this, the F5 embedding operation was changed to use *Wet Paper Codes* [58].

Wet Paper Codes aim to allow the sender to utilise side information not available to the receiver, if we consider this scenario for F5, this would be in the event of shrinkage. To alleviate this problem, Wet Paper Codes use syndrome coding (similar to matrix embedding, as shown earlier in this section). Using a random linear code of dimension $p = 18$, we encode m message bits into n AC DCT values, of which $n_{\mathbb{Z}^*} > m$ are all non-negative. Much like F5, the cover values (including zero AC DCT values) are visited using a random walk and separated into blocks of size $n/(m/p) = np/m$, of which on average $n_{\mathbb{Z}^*}p/m$ will be non-zero. Next a random binary matrix $\mathbb{D}$ of dimensions $p \times np/m$ is generated, and the cover $\mathbf{x}$ is modified such that the resulting stego vector $\mathbf{y}$ satisfies $\mathbb{D}\mathbf{y} = \mathbf{m}$ [1, 56]. By using Wet Paper Codes, nsF5 avoids the problem of shrinkage and is more secure.

2.7 Syndrome Trellis Codes

In this section, we describe Syndrome Trellis Codes in detail, and briefly, non-binary/multi-layered STCs. The Syndrome Trellis Code algorithm [59, 45] can embed a message into a cover using a single syndrome, minimising the distortion in linear time. Unlike syndrome encoding as described in previous sections, STCs uses a single syndrome, instead of multiple smaller syndromes.

$$\hat{\mathbb{H}} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad \mathbb{H} = \begin{bmatrix} 1 & 0 & & & & & & & \\ 1 & 1 & 1 & 0 & & & & & \\ & & 1 & 1 & 1 & 0 & & & \\ & & & & 1 & 1 & 1 & 0 & \\ & & & & & & 1 & 1 & 1 & 0 \\ & & & & & & & & 1 & 1 \end{bmatrix}$$

Figure 2.7: Demonstrating how the sub-matrix $\hat{\mathbb{H}}$ is tiled to generate the matrix $\mathbb{H}$.

The binary STC will be one of the focuses of this thesis, and the background is extensive, to prepare the reader.

2.7.1 Binary Syndrome Trellis Codes

With naive syndrome embedding, we would require $2^p - 1$ cover values to hide p bits of a message. STCs allows for us to use a single syndrome encoding, hiding arbitrary sized messages into arbitrarily sized covers. Using a sparse banded matrix, STCs perform near the rate-distortion bound [45]. To encode a message $\mathbf{m}$ of length m into some cover $\mathbf{x}$ of length n , we first generate the matrix $\mathbb{H}$ of dimensions $m \times n$. As previously mentioned, $\mathbb{H}$ is sparse and banded. It is formed by tiling a sub-matrix $\hat{\mathbb{H}}$ of dimensions $h \times k$, where h is the *constraint height* (h parameterizes the STC, and will be covered in further detail later), and $k = \left\lfloor \frac{n}{m} \right\rfloor$. The matrix is formed using a PRNG seeded with the shared secret.

The $\hat{\mathbb{H}}$ matrix is tiled in order to create the matrix $\mathbb{H}$, where each column of $\mathbb{H}$ contains exactly one column of $\hat{\mathbb{H}}$, and each subsequent $\hat{\mathbb{H}}$ is tiled from the next row. An example of this tiling can be seen in Fig. 2.7. However,

Algorithm 1 $\hat{\mathbb{H}}$ Greedy Tiling

```

1: procedure GREEDYTILING(coverLength, k, k')
2:    $\alpha \leftarrow \text{ceil}(\text{coverLength}/k)$ 
3:   if  $(\alpha \times k) = \text{coverLength}$  then return  $\alpha, 0$ 
4:    $\beta \leftarrow 0$ 
5:   while  $((\alpha \times k) + (\beta \times k')) \neq \text{coverLength} + w'$  do
6:      $\alpha = \alpha - 1$ 
7:      $\beta = \beta + 1$ 
   return  $\alpha, \beta - 1$ 

```

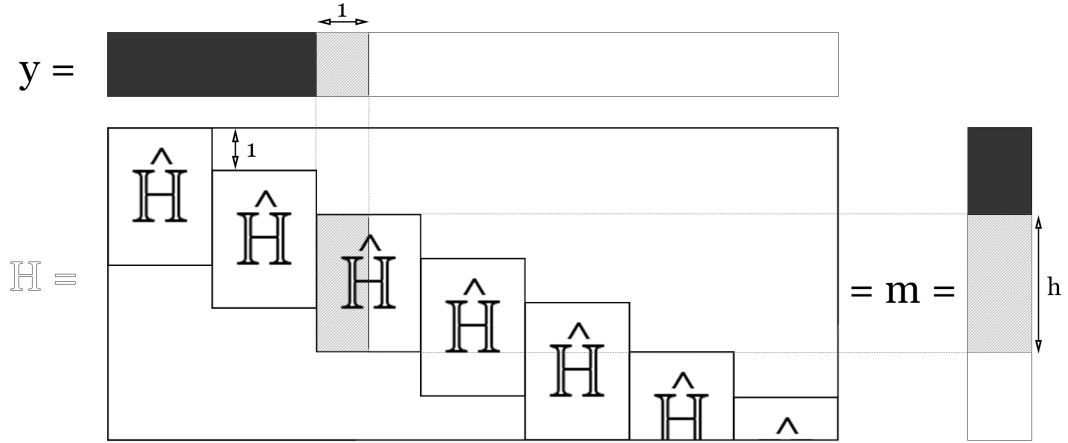


Figure 2.8: Tiling of the $\hat{\mathbb{H}}$ matrices, and how $\mathbb{H}$ allows for a subset of the input values to affect a sliding window subset of the syndrome. Darkened regions showing the already set portion of $\mathbf{y}$, the shaded region showing how a single value modification can affect the result of h message bits.

in practice a combination of two $\hat{\mathbb{H}}$ matrices of width k , $k' = k + 1$ will be required for non integer values of $\frac{n}{m}$. A tiling algorithm for this can be seen in Algorithm 1.

To encode our message, we wish to find solutions which satisfy $\mathbb{H}\mathbf{y} = \mathbf{m}$, where $\mathbf{y}$ is the minimum distortion leader in the coset of all possible solutions. Given that we are solving the matrix-vector multiplication $\mathbb{H} \cdot \mathbf{y}$, then there are 2^y possible solutions with a small subset solving $\mathbb{H}\mathbf{y} = \mathbf{m}$. However, as

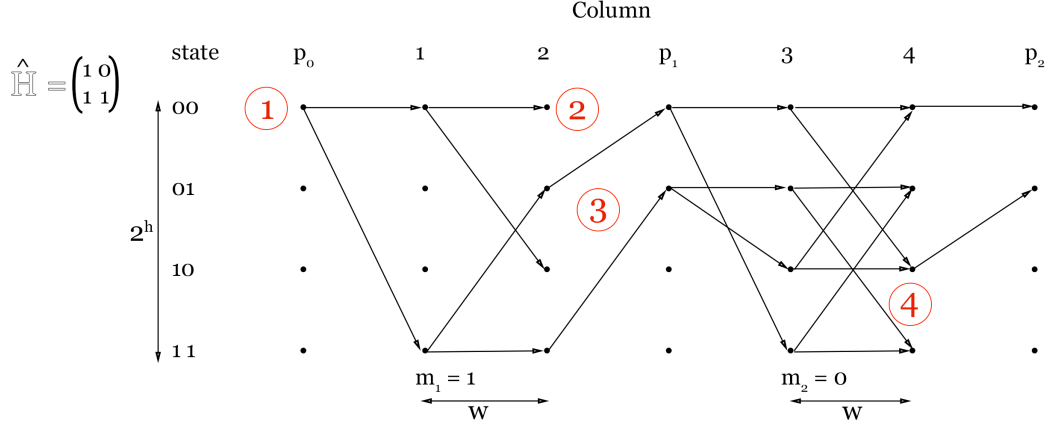


Figure 2.9: Trellis after generating the first two blocks (length w) from the start state p_0 , and two prune state p_1 , and p_2 . (1) - All paths start from the state p_0 , outgoing horizontal and diagonal edges represent 0 and 1 respectively, are the assignment of y_i . (2) At the end of the block we prune any state where the state labels LSB does not match the blocks message value, as such states 00, and 10 are pruned. (3) Non-pruned states have their state label bit-shifted to represent the change from m_i to m_{i+1} . (4) State labels allow sub-optimal (higher overall distortion) equivalent paths to be pruned.

seen in Figure 2.8, there are at most h non-zero elements in each column of $\mathbb{H}$, then there are only $2^h m$ solutions in the coset. As such, the time complexity is $O(2^h)$ not $O(2^n)$.

To find the coset of solutions, we build a trellis using the Viterbi algorithm [60] to express each possible permutation of the transformation $\mathbf{x} \rightarrow \mathbf{y}$, which satisfy $\mathbb{H}\mathbf{y} = \mathbf{m}$. The trellis is a grid of states, of dimensions $(n + m) \times 2^h$. States represent distinct partial syndromes, more notably they allow for analogue paths, i.e. paths of different stego value assignments that have the same forward effect on new cover value. To understand how states allow for equivalence, we first introduce the state labelling notation; as seen in Figure 2.9, we have 2^h rows in the trellis, representing the 2^h max-

imum different states at each assignment $y_i \rightarrow x_i$. State labels are binary representations, formed with the following assignment:

$$S' = \begin{cases} S \oplus \mathbb{H}_{*,i} \text{ iff } y_i = 1 \\ S \text{ otherwise} \end{cases}, \quad (2.40)$$

where S' denotes the new state label, S the previous state label, $\mathbb{H}_{*,i}$ is the i^{th} column in $\mathbb{H}$, and y_i is the i^{th} value in y . In the trellis (Fig. 2.9), we note that paths assigning $y_i = 0$ are horizontal edges, and $y_i = 1$ assignments are diagonal edges. When two edges meet at the same state, the lowest cost assignment is used, and higher cost edge is discarded (N.B. Figure 2.9 does not include the notion of costs for simplicity).

Columns in the trellis are labelled p_i or c_i , indicating a prune, or non-prune column respectively. There are w non-prune columns, and one prune column in each block b ; $\{c_i, \dots, c_{i+w}, p_b\}$. Non-prune columns represent the $x_i \rightarrow y_i$ assignment across the 2^h states. Prune columns prune any state for which the LSB of the state label does not match the respective message coefficient m_b since the LSB will not change any more after this block, non-pruned states have their label bit-shifted to represent the change of the partial syndrome from affecting the m_b to the m_{b+1} message coefficient. The exception to the above is the initial state p_0 which only represents the start of the trellis, and as such has no incoming edges.

In Fig. 2.10 we show an example of STC encoding, with a cover $\mathbf{x} = \{1, 0, 1, 1, 1, 1, 0, 1, 1, 0\}$, costs $\mathbf{c} = \{3, 1, 2, 5, 9, 2, 4, 1, 7, 3\}$, and message $\mathbf{m} = \{1, 0, 1, 0, 1\}$. The matrices $\mathbb{H}$, and $\hat{\mathbb{H}}$ used are introduced in Fig. 2.7. The

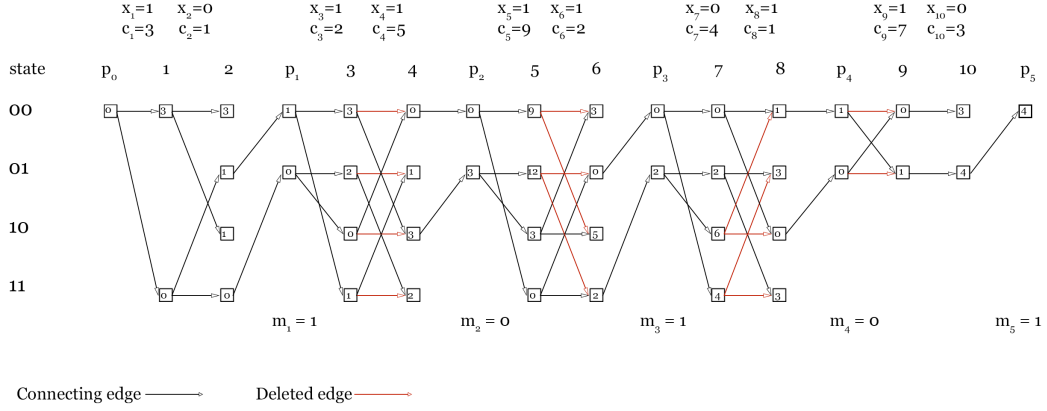


Figure 2.10: Example STC encoding, as per matrices in Fig. 2.7. State labels indicate the minimum cost to reach the respective path. Horizontal and diagonal edges represent $y_i = 0$, and $y_i = 1$ respectively, and subsequently the addition or not of the i th row in $\mathbb{H}$ to the syndrome. Red edges are deleted edges, indicating they reach a state at a sub-optimal cost.

encoding finds the lowest distortion stego object $\mathbf{y} = \{1, 0, 1, 1, 1, 1, 0, 0, 1, 0\}$ with cost $D(\mathbf{x}, \mathbf{y}) = 4$.

As mentioned earlier, the constraint height h parameterises the STC, as well as dictating the time complexity. Since the trellis contains 2^h states, increasing h increases the number of partial syndromes the STC can consider, as such the number of deleted edges also increases. Because of this, the embedding efficiency of the STC increases asymptotically towards the rate-distortion bound as h increases. In Fig. 2.11, we show this trade-off, as expected, the time complexity increases exponentially as h increases, whilst the embedding efficiency increases sub-linearly.

Aside from the constraint height, the contents of $\hat{\mathbb{H}}$ also has a large contribution to the efficiency of STCs. Each column in $\hat{\mathbb{H}}$ is required to be unique; otherwise, the trellis would contain two or more identical paths. It has also been found that setting the first and last rows equal to 1 in the sub-matrix

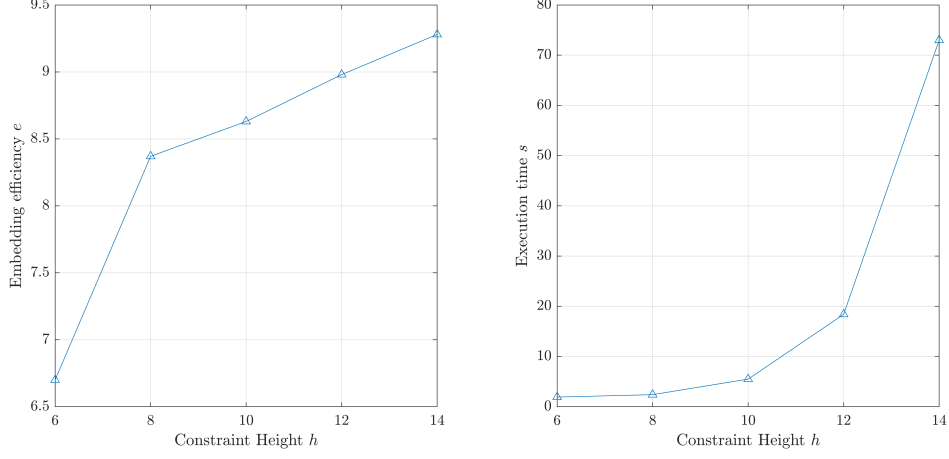


Figure 2.11: Results for 100 randomly selected images from the BOSS base [61], encoded with a pseudo randomly generated binary message, $\alpha = 1/52$, constant distortion profile $D(\mathbf{x}, \mathbf{y} = \sum_{i=1}^n = [y_i \neq x_i])$. Left - Embedding efficiency $e = m/D(\mathbf{x}, \mathbf{y})$ over different constraint heights. Right - Average execution times for a varying constraint height.

$\hat{\mathbb{H}}$ yields the best embedding efficiency in 97% of 1000 codes, with $h = 7$, and $w = 4$ [59].

2.7.2 Non-Binary and Multi-Layered Syndrome

Trellis Codes

So far, we have only considered binary embedding, or $q = 2$. However, there is no reason q cannot increase, and therefore allow us to embed with a higher alphabet [1]. Commonly, steganographers are using ± 1 embedding (whereby allowing an element of the cover to be increased or decreased by 1, or left unchanged), by using ternary ($q = 3$) embedding. To achieve this, Alice takes the q -ary representation of her cover elements; in the case of ternary, Alice takes the modulo three value of each element. Because a ternary symbol has

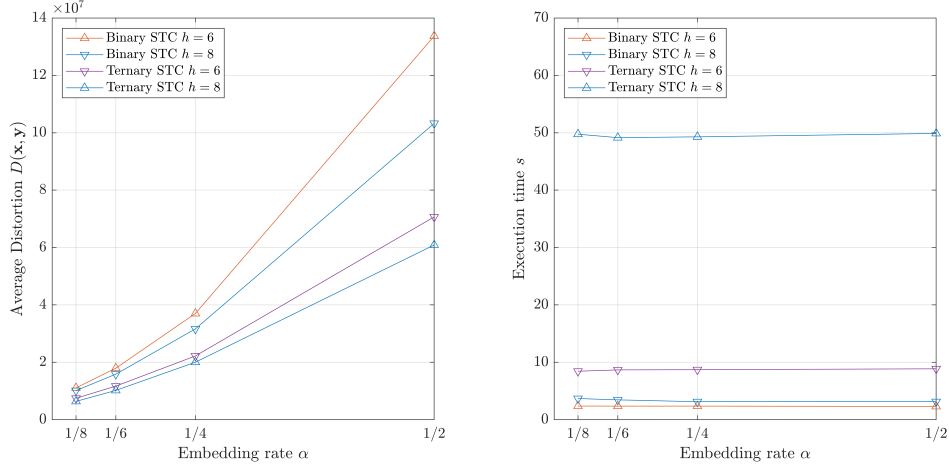


Figure 2.12: Results for 100 randomly selected images from the BOSS base [61], encoded with a pseudo randomly generated binary message, $\alpha \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}\}$, J-UNIWARD distortion function. Left - Average distortion over varying constraint heights. Right - Average execution times for a varying constraint heights.

three possibilities, represented by modifying the cover element by $-1, 0, +1$, Alice is able to embed $\log_2 3$ bits per ternary symbol [1]. Since $\frac{2}{3}$ cases result in a change, we can achieve an embedding efficiency of $\frac{\log_2 3}{2/3} = 2.37$, without the use of a code; a significant improvement on an efficiency of 2, when using binary. By increasing the value of q , we can increase our embedding efficiency. However, larger values of q carry a higher risk of being more detectable, than smaller changes with a smaller q [1].

There is a relatively small change required to STCs, in order to allow for non-binary embedding. Firstly, the number of states required in the trellis will grow from 2^h to q^h , and each cover value x_i is the $\bmod 3$ representation of the original cover value. The matrix $\hat{\mathbb{H}}$ will no longer be binary, but be composed of the symbols $[0, q - 1]$, and thus, so will $\mathbb{H}$. Finally, the state

transition as denoted in (2.40), will change from using XOR to:

$$S' = \begin{cases} \text{mod } 3(S + 2\mathbb{H}_{*,i}) \text{ iff } y_i = 2 \\ \text{mod } 3(S + \mathbb{H}_{*,i}) \text{ iff } y_i = 1 \\ S \text{ otherwise} \end{cases} \quad . \quad (2.41)$$

The above changes will allow the STC to compute q -ary changes; however, the time complexity will grow significantly, from 2^h to q^h .

The alternative is to use *Multi-Layered STCs* (ML-STC) [62]. The idea behind ML-STCs is that a higher alphabet (larger values of q), can be embedded using binary operations. By using k bits per cover value, k individual layers can be embedded into, using multiple passes of the STC. For example, the ternary symbols $\{0, 1, 2\}$ can be represented in binary as $\{00, 01, 10\}$ respectively. To embed using this scheme, the payload must first be split into the two layers used (*MSB*, and *LSB*), then embedded using a binary STC. In order to embed into two independent layers pseudo-simultaneously, the *MSB* layer is first encoded, then the resulting values in the *LSB* layer which must take specific value are fixed using Wet Paper Codes [63]. Notably, the amount of payload in each layer must be communicated to the receiver [62]. ML-STCs are also used in The Gibbs Construction [52], a framework for embedding using non-additive costs. Unlike other algorithms previously mentioned, The Gibbs Construction does not minimise an additive distortion function. Instead, The Gibbs Construction minimises a non-additive distortion function of the cover $\mathbf{x}$.

2.8 Forward Error Correction

In information theory, we define a code as a process whereby an original message is transformed into a set of accepted symbols, which can be communicated across a channel. The purpose of doing so could be for efficiency (reducing the bandwidth required), or robustness (adding redundancy for error detection and or correction). In this thesis, we shall only investigate the latter and its application in robust steganography. In particular, the subsections relating to The Noisy Channel, Steganography and FEC, and Reed-Solomon codes will be used heavily in Chapter 3, whilst convolutional Codes are the basis of the Syndrome Trellis Code algorithm.

The foundations of both information theory and communication theory have roots in Shannon's 1948 publication *A Mathematical Theory of Communication* [64]. In the landmark paper, Shannon identified the *channel capacity*, and proved that reliable communication is achievable for some transmission rate below the channel capacity.

2.8.1 The Noisy Channel

A channel in practice can take many forms: a satellite link between continents, an optical disk, or even fibre optic networks. The common characteristics of a channel are that a sender may communicate to a receiver over the channel. The channel is said to be *noisy* if it is not always the case, that whatever is sent on the channel, is received [65]. In Fig. 2.13, we show a depiction of the noisy channel. The *channel error* p_e is the error rate of the

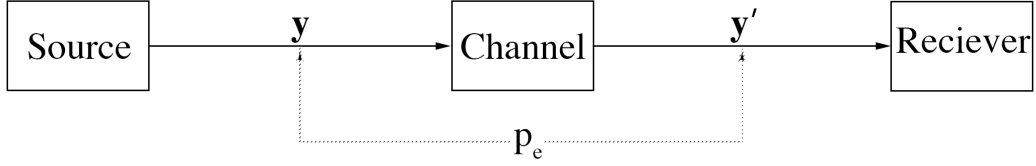
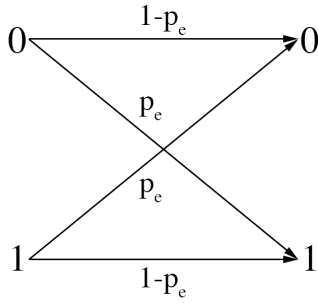


Figure 2.13: Illustration of the noisy channel. A sender communicates by sending some message $\mathbf{y}$ over the channel, with the channel error p_e . The receiver receives $\mathbf{y}'$.

Binary Symmetric Channel



Binary Erasure Channel

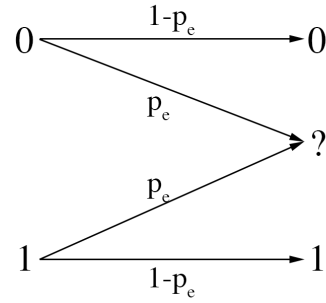


Figure 2.14: Binary Symmetric and Erasure Channel, with the channel error p_e .

channel.

Within the binary domain, there are two common types of channels: *the binary symmetric channel*, and *the binary erasure channel*. From Fig. 2.14 we can see in the binary symmetric channel, a bit is transmitted over the channel with error p_e , as such the receiver receives the original bit with probability $1-p_e$, and the flipped bit with probability p_e , the channel capacity $c = H(p_e)$ [64]. In the erasure channel, a bit is transmitted; however, the receiver may, or may not receive a bit. The receiver receives the bit with probability $1-p_e$, but the bit is not received with probability p_e , as such,

the channel capacity is $c = H(1 - p_e)$. During this thesis, we only consider the binary symmetric channel.

For example, if Alice transmits the following binary sequence:

1 1 0 0 1 0 1 0 1 1 1 0 0 0 0 1 0,

however, Bob receives:

1 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1 0,

then we express this as two problems:

1. **Detection of errors** - As Bob only receives the mutated sequence, without side-information, he cannot distinguish if the sequence is valid or not, and where (if any) errors have occurred.
2. **Correction of errors** - Should Bob be able to identify the errors, he may attempt to correct the errors, again only if he has side-information to aid him.

The required side-information is added through the use of extra information added to transmissions, commonly referred to as parity or redundant information. For example, if Alice was to repeat each bit 3 times, then we would add redundancy in the form of additional bits which aid Bob. Say Alice sends the sequence:

1 1 1 0 0 0 1 1 1,

whereas Bob receives:

1 0 1 0 0 0 0 1 1,

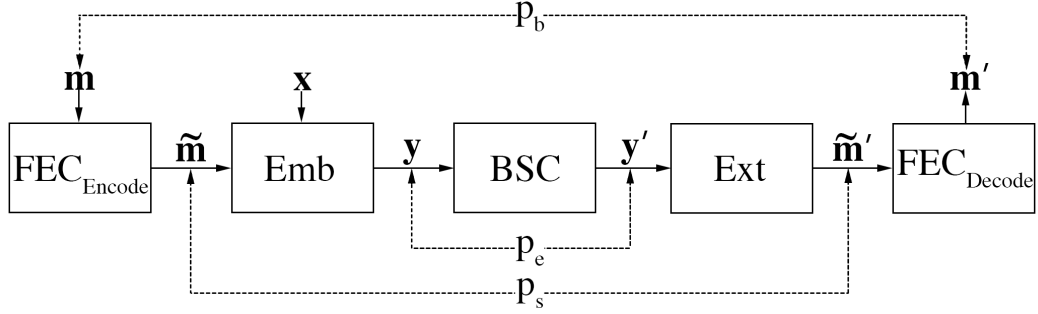


Figure 2.15: Illustration of the noisy stego channel. A message $\mathbf{m}$ is transmitted over the noisy channel (error rate p_e), inside the stego object, the resulting error rate of the secret message after error correction is p_b .

in this case, Bob takes each 3-bit chunk of the received sequence, as long as more than half the bits are valid, Bob can take the most common value as the valid value:

1 1 1 0 0 0 1 1 1.

As such, each triple of bits forms a *code word*. A code word is an accepted word from the language of the code, in this example, the only acceptable codewords are 000 and 111. When Bob receives any code word which is not either of these values, he knows the received codeword contains an error.

The specific measure for error correction is the *bit-error rate* (BER). In the binary symmetric channel (of which will be the sole focus during this thesis), the bit-error-rate is defined as error bits/transmitted bits. For example, communicating 100 bits, with 20 errors would have a 0.2BER. The measure of any channel error is also expressed as bit-error rates.

2.8.2 Steganography and FEC

Whist, the area of robust steganography, is relatively unexplored, combining forward error correction with a stego system can be achieved relatively easily. The obvious choice for combining forward error correction is to encode the secret message as a codeword first, before embedding the codeword in the stego object.

In Fig. 2.15 we show a depiction of the noisy stego channel. We introduce the forward error correction functions FEC_{Encode} , and FEC_{Decode} , which encode some message $\mathbf{m}$ resulting in the codeword $\tilde{\mathbf{m}}$, and decode the received codeword $\tilde{\mathbf{m}}'$ resulting in the corrected message $\mathbf{m}'$, respectively. As previously introduced, we have some steganographic embedding and extraction functions Emb and Ext . Since the Emb and Ext functions have no prior information on messages themselves, they are agnostic to the forward error correction functions, they simply embed or extract the message given. Unlike other examples in this thesis, which we assume are noiseless, in this scenario, we have some binary symmetric channel, with the error rate p_e , of which the stego object is exposed to, the warden, has access to the unchanged stego object $\mathbf{y}$. The received stego object $\mathbf{y}'$ is the result of the stego object being communicated across the noisy channel, and will as such, contain errors. Because of this, the extracted uncorrected message $\tilde{\mathbf{m}}'$ will also contain errors, the error rate between the forward error corrected message $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{m}}'$ is p_s . After error correction, the final message $\mathbf{m}'$ has an error rate p_b , from the original message $\mathbf{m}$. By using error correction, we hope the error rate p_b is lower than the p_s .

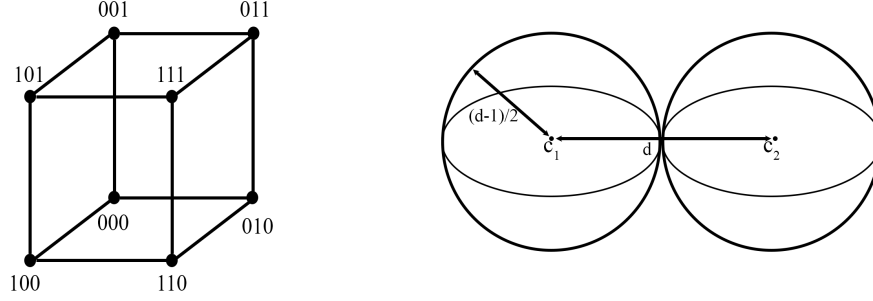


Figure 2.16: (Left) a 3-bit binary cube displaying all binary words. (Right) two codes $\mathbf{c}_1$, and $\mathbf{c}_2$ represented as spheres.

2.8.3 Minimum Distance of Block Codes

A block code in general takes k information bits, and adds $n - k$ parity bits to form a code word of length n , such that $n > k$, and the rate k/n [66]:

$$Source = \{c_1, c_2, \dots, c_k\} \rightarrow \{c_1, c_2, \dots, c_k, c_{n-k}, c_{n-k+1}, \dots, c_n\}. \quad (2.42)$$

The *minimum distance* d of a block code is the minimum hamming distance d_H , between any two code words $\mathbf{c}_1$ and $\mathbf{c}_2$ of length n , where the hamming distance is:

$$d_H(\mathbf{c}_1, \mathbf{c}_2) = \sum_{i=1}^n [c_{1_i} \neq c_{2_i}]. \quad (2.43)$$

In Fig. 2.16 we show a 3-bit binary cube representing all the words from a nominal code, the minimum distance between any words in the code is the distance between the points. For example, the distance between 011 and 110, is 2. The minimum distance of a code dictates its error detection and

error correction capabilities [65]. For a code which has minimum distance d , we can detect at most $d - 1$ errors, and correct $\frac{d-1}{2}$ errors. As shown in Fig. 2.16, if we depict code words as spheres with a radius $\frac{d-1}{2}$, then any noisy transmission of a code word $\mathbf{c}$ resulting in the received code word $\mathbf{c}'$ can be corrected to $\mathbf{c}$, if the distance between the two is less than $\frac{d-1}{2}$.

2.8.4 Hamming Codes

Hamming Codes were first introduced by Richard Hamming [67] whilst working for Bell Labs, after becoming increasingly frustrated by the errors introduced by punch card readers. They are a class of simple linear parity-check codes, with a message of length $k = 2^r - r - 1$, a code word exists of length $n = 2^r - 1$ for any $r \geq 2$ [66].

The (7,4) Hamming code has a minimum distance $d = 3$ and thus can correct a single error, and detect up to two errors per block. For a more detailed explanation of the (7,4) code, see Appendix B. Hamming Codes are suited to low error rate environments and are still used in ECC memory.

2.8.5 Reed-Solomon

Introduced by Irving S. Reed and Gustave Solomon, Reed Solomon (RS) codes are block-based error-correcting codes, and a subset of BCH codes [66]. RS codes are particularly well-suited for *burst errors*. Burst errors as the name suggests, are contiguous errors across multiple symbols. An (n, k) RS code takes k information symbols and adds $n - k$ parity symbols to make a codeword of length n , with the minimum distance of $d = n - k + 1$. RS codes

can detect up to $n - k$ errors, and correct $\frac{n-k}{2}$ errors [68]. Therefore, as k increases, the length of the RS code increases, as does the error-correcting performance [68]. RS codes are also a *maximum distance separable* (MDS) code, any code which meets the *Singleton bound* with equality is MDS, where the Singleton bound is:

$$k + d \leq n + 1. \quad (2.44)$$

Unlike Hamming Codes, RS codes are *systematic*, as the k information symbols are not changed, with the $n - k$ parity symbols appended to the end to form the code word.

Typical applications for RS codes include optical disks (CDs, DVDs, Blu-Ray). Since disks are prone to physical damage, errors often occur in bursts. By using two RS codes, a relatively weak inner code, interleaved into a more powerful outer code results in the *Cross Interleaved Reed Solomon Code* (CIRC).

2.8.6 Convolutional Codes

A convolutional code is a code generated by applying the input symbols over a Boolean polynomial function which outputs a continuous code. They are denoted (n, k, m) , where the rate of the code is k/n , generating n output bits per k bits received, and m is the *constraint length*, or memory order [69]. As such, the output of a convolutional code is dependant on the previous states of the coder.

An example of a convolutional code is the $(2, 1, 2)$ code as shown in Fig-

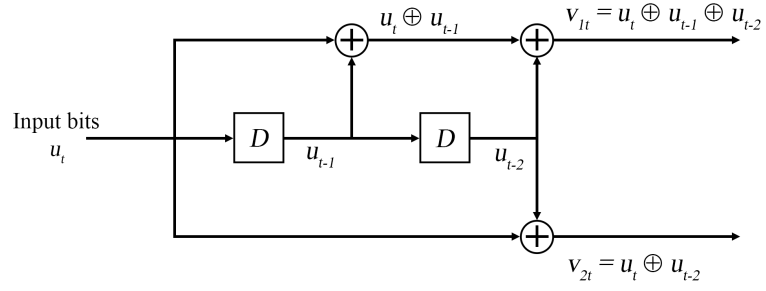


Figure 2.17: (2, 1, 2) convolutional encoder.

Table 2.2: Example encoding of the convolution encoder from Fig. 2.17

u	u_{t-1}	u_{t-2}	$u_t \oplus u_{t-1}$	v_{1t}	v_{2t}
True	False	False	True	True	True
False	True	False	True	True	False
True	False	True	True	False	False
False	True	False	True	True	False
True	False	True	True	False	False
True	True	False	False	False	True

ure 2.17. The code takes $k = 1$ input bit, and generates $n = 2$ output bits, therefor its rate is $\frac{1}{2}$. In this example the encoder has two *shift registers*. Upon receiving a bit u_t , the shift register outputs the bit u_{v-1} i.e. the bit which precedes u_t . The outputs of these shift registers are XOR'd together, with two outputs v_{1t} and v_{2t} . In Table 2.2 we show the example encoding with the input:

$$\mathbf{u} = \{101011\}, \quad (2.45)$$

resulting in the output code:

$$\mathbf{v} = \{111000100001\}. \quad (2.46)$$

We can express convolutional codes as *linear time-invariant* (LTI) systems:

$$v_j = u_1 * g_j^{(1)} + u_2 * g_j^{(2)} + \cdots + u_k * g_j^{(k)} = \sum_{i=1}^k u_i * g_j^{(i)}, \quad (2.47)$$

where $*$ is the convolution operation, and $g_j^{(i)}$ is the *impulse response*. The impulse response $g_j^{(i)}$ is found by inputting the impulse sequence $\{1, 0, 0, \dots, 0\}$ at the i^{th} input and observing the j^{th} output. We can then create the gener-

ator matrix for the code:

$$\mathbb{G} = \begin{bmatrix} \mathbb{G}_0 & \mathbb{G}_1 & \mathbb{G}_2 & \dots & \mathbb{G}_m \\ & \mathbb{G}_0 & \mathbb{G}_1 & \dots & \mathbb{G}_{m-1} & \mathbb{G}_m \\ & & \mathbb{G}_0 & \dots & \mathbb{G}_{m-2} & \mathbb{G}_{m-1} & \mathbb{G}_m \\ & & & \ddots & & & \ddots \end{bmatrix}, \quad (2.48)$$

where $\mathbb{G}_l$ is the $k \times n$ sub matrix:

$$\mathbb{G}_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{2,l}^{(1)} & \dots & g_{n,l}^{(1)} \\ g_{1,l}^{(2)} & g_{2,l}^{(2)} & \dots & g_{n,l}^{(2)} \\ & \dots & & \dots \\ g_{1,l}^{(k)} & g_{2,l}^{(k)} & \dots & g_{n,l}^{(k)} \end{bmatrix}. \quad (2.49)$$

Taking the convolutional encoder example from Fig. 2.17 we have the following generator matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & & & \\ & & 1 & 1 & 1 & 0 & 1 & 1 & \\ & & & 1 & 1 & 1 & 0 & 1 & 1 \\ & & & & 1 & 1 & 1 & 0 & 1 & 1 \\ & & & & & 1 & 1 & 1 & 0 & 1 & 1 \\ & & & & & & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (2.50)$$

Chapter 3

Robust Syndrome Coding

This chapter focuses on robust adaptive steganography. We show how there is a trade-off between distortion, and robustness, and how the stenographer can strike a balance between the two. We look at how codes can dramatically increase the error rate of the payload embedded, when the stego object is transmitted over the noisy channel, and how we can change our embedding to minimise the impact of the channel noise, whilst still minimising distortion. Our contribution, *Dual Syndrome Trellis Codes*, are a modification of the original Syndrome Trellis Code algorithm, which can embed two syndromes, one the payload, and the other for error correction.

3.1 Statement of the Problem

Steganography, unlike digital watermarking, does not typically strive for robustness. The adversary is a passive detector, and the communication chan-

nel noiseless [1]. This models real-world problems of communicating a hidden message inside, for example, an image shared on a social media platform.

However, some image-sharing websites and social media platforms recompress uploaded images or add visible watermarks. This introduces a non-adversarial element of noise into the channel. Unfortunately, even a small amount of noise is often catastrophic for adaptive algorithms [15]. Such algorithms employ Syndrome Trellis Codes [45, 59] as introduced in Section 2.7.1, which carry the payload $\mathbf{m}$ as the syndrome of a block-structured parity-check matrix $\mathbb{H}$. However, unless $\mathbb{H}$ is extremely sparse, one error in the channel causes many errors in the syndrome, so that even small amounts of noise in the channel lead to high bit error rates in the payload.

To illustrate this phenomenon, we measured the error rate of the Least Significant Bit (LSB) of DCT coefficients in JPEG files that have been decompressed and recompressed with a matching quality factor and identical JPEG library. This represents the least-damaging form of recompression. Although most of the JPEG process is invertible, the final decompression step of clamping and rounding pixel values means that compression is not an idempotent operation. Errors are more numerous in JPEGs with a higher quality factor where the quantisation is finer, as rounding noise is more likely to tip a coefficient into the next quantisation bin.

In Fig. 3.1, we show the bit error rate induced by this recompression, for JPEG quality factors between 50 and 100, measured on the 512×512 images in the BOSSBase [61] set. We also show the bit error rate for payload if STCs are used to embed payloads of 26214 and 52429 bits (0.1 and 0.2 bits per pixel, respectively). The constraint height of the STC was $h = 8$; as we

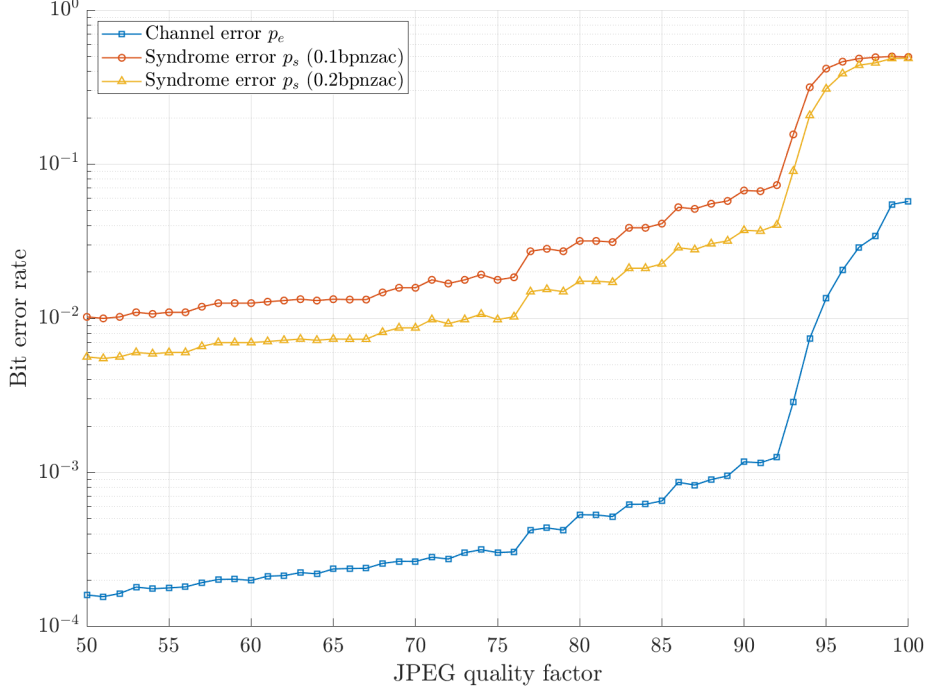


Figure 3.1: Bit error rates p_e for coefficient LSBs in recompressed JPEGs, as the quality factor varies, and corresponding error rates of adaptive steganography.

shall see later, this is a significant factor in the fragility of STC embedding. Note also that the *shorter* payload is more fragile. However, there are few alternatives to STCs if we want to reduce distortion.

In this chapter, we examine the relationship between the error rate of a binary symmetric channel (BSC) and the error rate in the syndrome and examine methods for error correction in adaptive steganographic algorithms. We emphasise that, unlike digital watermarking methods, we consider only non-adversarial noise. There is relatively little literature on this problem, and we show that it has unexpected lessons for adaptive steganography: where possible, STCs should be avoided when the channel is noisy, but if distortion

must be reduced then there are ways to mitigate STC fragility: lower the constraint height and parity-check density and use only as much cover as necessary to meet a distortion constraint.

There is a wealth of literature on digital watermarking [6], which aims for robustness against an active attacker. However, there is little written about robust steganography robust and almost all work predates adaptive embedding.

Marvel et al. [70] builds on an early type of steganography that requires the receiver to know the cover object. In their system, called ‘SSIS’, the receiver estimates the cover from the stego object using image restoration techniques. Within their hiding mechanism, low-rate Reed Solomon (RS) codes are used to correct the errors of imperfect restoration. This is not robust steganography as such, but it illustrates the most basic idea of applying forward error correction to a payload before embedding it into a channel that might be noisy. A similar architecture is employed by ‘YASS’, proposed by Solanki et al. [71], where noise is introduced by using ‘big blocks’ that do not align with JPEG blocks. The errors are corrected using a Repeat-Accumulate (RA) code.

Zhang and Wang [72] use cyclic codes to both improve embedding efficiency of, and to introduce error correction to, the resulting stego objects. Predating the technology of STCs, the distortion heuristic is that all changes have equal cost. The authors make clear that there is a trade-off between embedding rate and robustness because increasing the embedding rate forces codewords to be closer together. Ref. [73] proposes a similar idea with a distortion heuristic that is not constant: some blocks are not used for em-

bedding, in which case their symbols are erased. Either RS or RA codes are used to remove errors caused by the selective avoidance of high-cost blocks.

Separately from the practical applications of steganography, Hopper et al. [74] formalise a complexity-theoretic approach to steganography. This work is most famous for its analysis of the rejection sampler, a low-bandwidth but provably perfectly-secure steganography scheme; less famously, the same paper also proposes a modification to work in noisy channels. The authors identify an upper-bound on the steganographic capacity of noisy channels, but the abstract model is far from the practice of steganography in digital images. Rejection sampling is also computationally expensive, so we do not consider it further.

The most recent work on steganography in noisy channels is [75]. Published just before the introduction of STCs, it uses syndrome coding with the parity-check matrix of a $(7, 4)$ Hamming code. Like YASS, some blocks are avoided; furthermore, the matrix embedding allows the number of embedding changes to be reduced. This is equivalent to costs of 1 or ∞ . To reduce errors, RA codes are using inside the syndrome, i.e. forward error correction is applied before embedding. A soft decoder is used.

3.2 Robustness/Distortion Trade-off

Fundamentally, minimising distortion can be adverse to maximising robustness. Choosing codewords which minimise distortion requires finding a codeword close to the existing cover, whilst choosing a codeword which maximises robustness requires picking a codeword that is far from other codewords. An

essential step for creating a robust stego system is to choose appropriate parameters. Should we use the entire cover? How does the constraint height affect error rates? Should we pad the message?

We restrict our attention to binary codes. Let $\mathbf{m}$ be the message, of l bits, to be hidden in cover $\mathbf{x}$ of n LSBs. To reduce errors, $\mathbf{m}$ may be expanded using forward error correction to $\tilde{\mathbf{m}}$, of length $\tilde{l}$. This is embedded using adaptive steganography to create a stego object whose LSBs are $\mathbf{y}$ of length n . After transmission, the stego object is corrupted to $\mathbf{y}'$, decoded as message $\mathbf{m}'$.

Let us examine why the bit error rate of STC adaptive steganography is so high in Fig. 3.1. According to the STC algorithm, $\mathbf{m} = \mathbb{H}\mathbf{y}$ and $\mathbf{m}' = \mathbb{H}\mathbf{y}'$. Recall that, with STCs, $\mathbb{H}$ is constructed from copies of $\hat{\mathbb{H}}$ as shown in Fig. 2.8.

Suppose that the errors in $\mathbf{y}'$ are due to a BSC with error rate p_e ¹. We define p_s , the syndrome error rate, as the bit error rate of $\mathbb{H}\mathbf{y}$ and derive it in terms of p_e , l , n , and the average number of 1s in each column of $\hat{\mathbb{H}}$, denoted a .

There is an elementary argument as follows. On average there will be np_e errors in $\mathbf{y}'$. Suppose that one error affects bit i of $\mathbf{y}$; this adds column i of $\mathbb{H}$ to $\mathbf{m}$, which corrupts on average a bits of $\mathbf{m}$. Although some errors cancel out, a first-order approximation to the number of errors in $\mathbf{m}'$ is therefore

¹Errors in digital media might occur in bursts, but we can mitigate this by visiting the cover and stego objects in a keyed pseudorandom order. Later, we will also assume that errors in the syndrome are also independent, as the syndrome can be visited in a random order.

anp_e , and so the error rate of the syndrome is

$$p_s = p_e \frac{an}{l} + O(p_e^2). \quad (3.1)$$

A more sophisticated argument assumes that each entry of $\mathbb{H}_{ij}$ are iid Bernoulli with parameter a/l (we ignore the block structure that is present in STCs). Since the probability that an error (denoted $*$) occurs in some bit of the syndrome is:

$$P(*) = P_e, \quad (3.2)$$

and that bit of the syndrome is a 1 with probability:

$$P(1) = \frac{a}{l}, \quad (3.3)$$

then the probability that a syndrome bit is both a 1, and is affected by an error is:

$$P(1 \wedge *) = P_e \frac{a}{l}. \quad (3.4)$$

Thus the probability that $m'_j \neq m_j$ is the probability that bit j is subject to an odd number of flips, which is the probability that $X \sim \text{Bi}(n, q)$ is odd:

$$P(\text{bit error}) = P\left(\text{Bi}\left(n, P_e \frac{a}{l}\right) \text{ is odd}\right). \quad (3.5)$$

The probability generating function gives us:

$$\begin{aligned}
\mathbb{E}[s^x] &= \sum_{x=0}^n s^x \binom{n}{x} q^x (1-q)^{n-x}, \\
&= \sum_{x=0}^n \binom{n}{x} (sq)^x (1-q)^{n-x}, \\
&= (sq + 1 - q)^n.
\end{aligned} \tag{3.6}$$

Then:

$$\begin{aligned}
\mathbb{E}[-1^x] &= 1 \cdot P(x=0) \\
&\quad + (-1) \cdot P(x=1) \\
&\quad + 1 \cdot P(x=2) \\
&\quad + (-1) \cdot P(x=3) \\
&\quad + \dots,
\end{aligned} \tag{3.7}$$

and as such:

$$P(x \text{ is odd}) = 1 - \frac{1 + \mathbb{E}[-1^x]}{2} = \frac{1}{2}(1 - (1 - 2q)^n). \tag{3.8}$$

Finally we can express the expected error rate of the syndrome p_s as:

$$p_s = \frac{1 - (1 - 2ap_e/l)^n}{2} = p_e \frac{an}{l} + O(p_e^2). \tag{3.9}$$

Setting aside other error-correction techniques, this suggests three elementary ways to increase the robustness of adaptive syndrome coding, all of which decrease p_s : reduce n (do not use the entire cover); reduce a (decrease

the constraint height of the STC); increase l (pad out the message in some way).

3.2.1 Benchmarking the Trade-off

For the experiments in this chapter, we used 1000 of the 512×512 grayscale images from BOSSBase [61], JPEG compressed with quality factor 75. Broadly informed by Fig. 3.1, we will simulate a noisy channel by flipping LSBs of the stego DCT coefficients at error rate $p_e = 10^{-3}$. For adaptive embedding, we use the J-UNIWARD [20] costs, and we simulate payloads of length $l \in \{2000, 4000, 8000, 12000, \dots, 40000\}$ bits.

We will examine the robustness/distortion trade-off empirically, displaying the results in plots such as Fig. 3.2: on log scales, average total J-UNIWARD cost (distortion) on the x -axis and the bit error rate of the syndrome p_s (later the corrected bit error rate p_b) on the y -axis. For various STC parameters, later augmented by forward error correction, we plot the average distortion $D(\mathbf{x}, \mathbf{y})$ and average error rate for each payload; naturally, increasing payloads cause more distortion. The best robustness/distortion trade-off is found with a graph to the left and below the others, but we must also take into account the absolute amount of distortion (a low bit error rate is not secure if the embedding is easily detectable). To relate distortion to detectability, we computed the average distortion costs of payloads corresponding to 0.1 and 0.4 bits per nonzero DCT coefficient, when embedding using binary STCs. These are 8.4×10^5 and 6.8×10^6 , respectively. We plot these vertical lines on each distortion-error chart. In [76], on BOSSBase

BPNZAC	$D(\mathbf{x}, \mathbf{y})$	QF 75		QF 95	
		JRM	JPSRM	JRM	JPSRM
0.1	835,806	0.4632	0.4350	0.4923	0.4920
0.4	6,804,747	0.2376	0.1228	0.3951	0.3246

Table 3.1: Bits per non-zero coefficient and average distortion, for 1,000 BOSSBase images, with detection error E_{OOB}) at JPEG quality factors 75, and 95.

images, Holub found that these payload rates were detectable by JRM features [77] with error rates of 0.4632 and 0.2376, respectively. This suggests that average distortion of 10^7 is almost certainly too high, and below 10^6 probably sufficiently low for mostly-undetectable embedding.

Holub benchmarks different JPEG steganographic methods against an array of detectors. From this we found the average distortion budget for our cover set of 1,000 BOSSBase [61] for 0.1 and 0.4BPNZAC, as shown in Table 3.1. For the remainder of the plots in this chapter, we have added vertical lines indicating the equivalent distortion for 0.1 and 0.4BPNZAC for a JPEG QF of 75.

3.2.2 Using a Subset of the Cover

Equation (3.1) shows that the syndrome error rate can be reduced by reducing n ; the STC is applied to only $n' \leq n$ locations of the cover (and the other $n - n'$ left unchanged). In order to convey the entire message we require $n' \geq l$; in the extreme case $n' = l$ the STC becomes trivial, and we resort to an operation such as LSB Matching. Lowering n' , we would expect to improve the robustness/distortion trade-off by reducing fragility.

We parameterise this by

$$n' = l + r(n - l), \quad r \in [0, 1], \quad (3.10)$$

and plot the distortion-robustness trade-off in Fig. 3.2. Here the constraint height of the STC is $h = 1$: see the following section for why we made this choice. The line $r = 0$ is below the others: the clear message is that for best robustness/distortion trade-off, do not use STCs at all. However, not using STCs does have a distortion penalty, and for the case $r = 0$ even the smallest message of 2000 bits causes distortion of approximately 3×10^6 , which is probably detectable by existing methods. If both payload length and distortion constrain the embedder, they must use an adaptive method to reduce the latter.

We can approximate the distortion in Syndrome Trellis Code embedding, when a constraint height $h = 1$ is used. Given $h = 1$, there is no overlap in the tiling of $\hat{\mathbb{H}}$ matrices when generating $\mathbb{H}$;

$$\mathbb{E} \left[(D(\mathbf{x}, \mathbf{y})) = \frac{l}{2} \left(F\left(\eta \frac{n}{l}\right) \right) \right], \quad (3.11)$$

where $F(x)$ is the monotonic convex function of the average cost of encoding x bits into our cover.

If we consider (3.1), then it could be perceived that reducing the cover size, or using a subset of the cover would reduce the error rates p_s , and p_b . Let us first consider using half the cover for our encoding without any error correction, expanding on (3.11) to approximate distortion of using half the

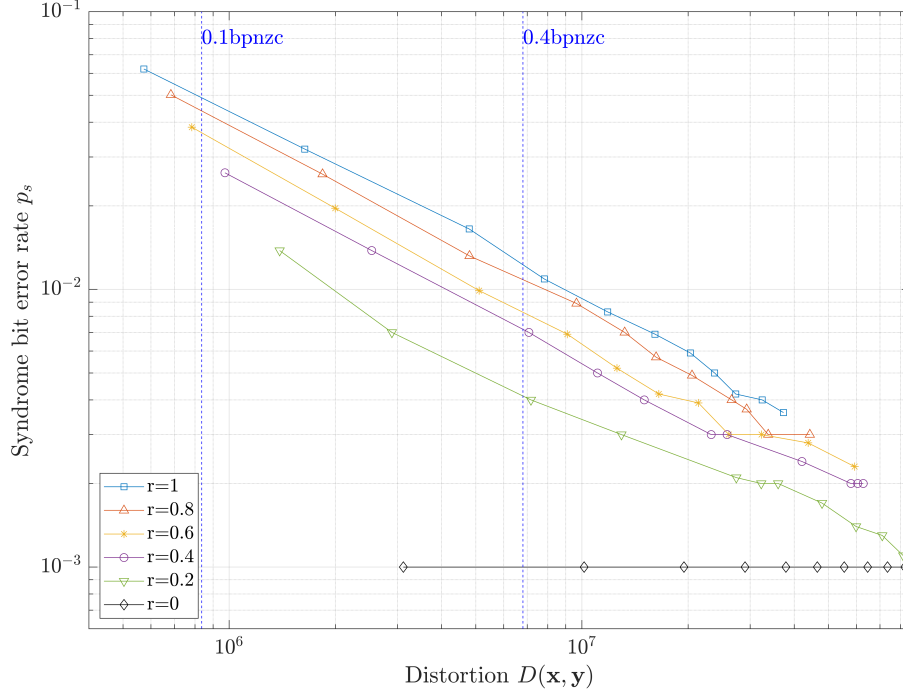


Figure 3.2: p_s vs $D(\mathbf{x}, \mathbf{y})$ for varying values of r .

cover would give us

$$D(\mathbf{x}, \mathbf{y}) = \frac{l}{2} F(\eta \frac{n}{2l}). \quad (3.12)$$

For the case of encoding in both halves of the cover, with the same message length we would expect

$$D(\mathbf{x}, \mathbf{y}) = \frac{l}{2} F(\eta \frac{n}{l}). \quad (3.13)$$

As the message lengths are the same in both examples, using some forward error-correcting code, we chose a code which adds $2k$ parity bits to the message, resulting in the larger error corrected message $\mathbf{m}'$. If we now

adjust (3.12) (3.13) for $\mathbf{m}'$, we get

$$D(\mathbf{x}, \mathbf{y}) = \frac{l}{2} + k F\left(\eta \frac{n}{2(l+2k)}\right), \quad (3.14)$$

for encoding in half the cover, and

$$D(\mathbf{x}, \mathbf{y}) = \frac{l}{2} + k F\left(\eta \frac{n}{l+2k}\right), \quad (3.15)$$

for encoding in the both halves of the cover. Since $F(x)$ is convex, encoding in the entire cover will provide a lower distortion, for the same amount of robustness.

3.2.3 Reducing the Constraint Height

The number of ones in a column of $\mathbb{H}$ is determined by the constraint height and the density of the generator matrix $\hat{\mathbb{H}}$, which we denote η . In the notation of Eq. (3.1), $a = h\eta$.

From (3.1), it becomes clear that lowering the constraint height would decrease the syndrome error rate. As suggested by Filler et al. [45, 59], the STC algorithm finds better stego objects as the constraint height increases. By better we mean lower distortion, however, when we consider robust adaptive stego objects, it is best to minimise the constraint height, as the increased syndrome error rate p_s outweighs the lower distortion.

To examine these parameters, we tested STCs with constraint heights $h \in \{1, 2, 3, 5, 8\}$, with $\eta \in \{0.25, 0.5, 0.75\}$ for $h = 1$, and the same other parameters as for Fig. 3.2, showing the results in Fig. 3.3. Eq.(3.1) predicts

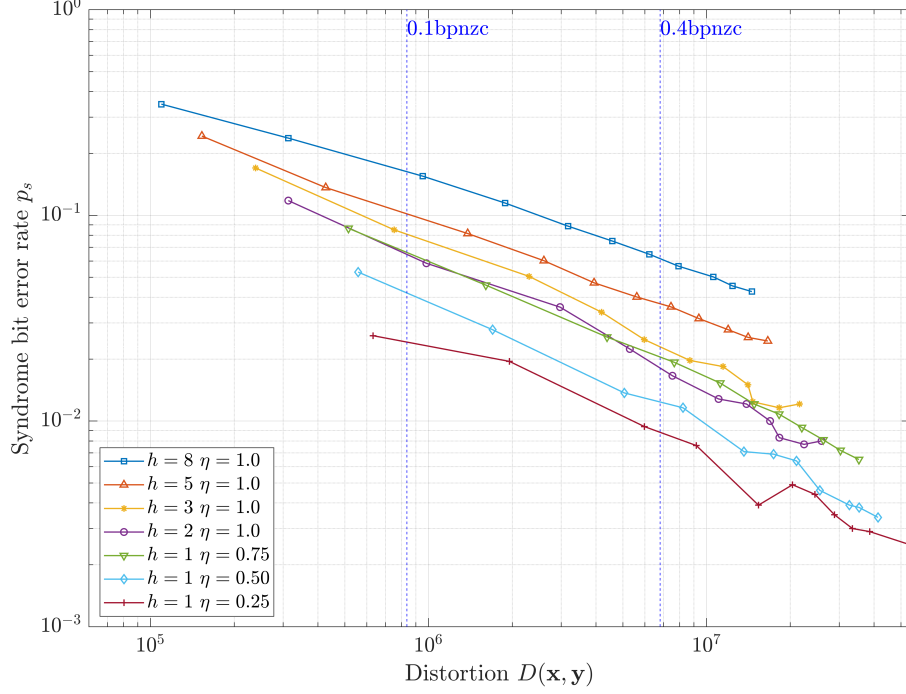


Figure 3.3: p_s vs $D(\mathbf{x}, \mathbf{y})$ for varying values of h and η .

that reducing h or η results in lower p_s ; our understanding of STCs as approaching random linear codes as $h \rightarrow \infty$, and zero columns of $\hat{\mathbb{H}}$ as unused locations in the cover, predicts that reducing h or η results in higher distortion. This is what we observe, and furthermore that reducing h or η is a better trade-off between distortion and syndrome error rate (in that error rates are lower for each corresponding distortion), but if detectability is a concern then there is a limit to the embedder's ability to thin out the matrix $\mathbb{H}$.

It is clear that a lower constraint height, or a sparser matrix $\mathbb{H}$ reduces the syndrome error rate, at the cost of increased distortion. However, if the aim is to minimise syndrome error, the constraint height and density of $\mathbb{H}$

should be reduced.

3.2.4 Increasing the message length

We have determined that the most robust STCs are used as little as possible, and with a constraint height of $h = 1$ so that they become block codes (the trellis is trivial).

The final way to reduce p_s , from (3.1), is to increase l . Like the other options, this inevitably increases distortion. A simple method is to pad the payload with random garbage, but if the message is to be expanded, it makes sense to use the extra bits for forward error correction. We now introduce $\tilde{\mathbf{m}}$ of length $\tilde{l} \geq l$ as the result of $\text{FEC}_{\text{Encode}}$: some forward error correction encoding function on the message $\mathbf{m}$. In Fig 3.4, we show how a practical stego system using forward error correction could work, where forward error correction pads the message. As well as the syndrome error rate p_s , we also have p_b , the message bit error rate after error correction. We reiterate that error correction inside syndrome has two advantages: the decoding process removes some errors, but also the increased syndrome length reduces the syndrome error rate, thanks to (3.1). We will benchmark one example of FEC (RS codes) in Section 3.4. RS codes are a maximum distance separable code, a (n, k) RS code with n information bits, and k parity bits can correct $(n-k)/2$ errors. Therefore, as k increases, the length of the RS code increases, as does the error-correcting performance [68].

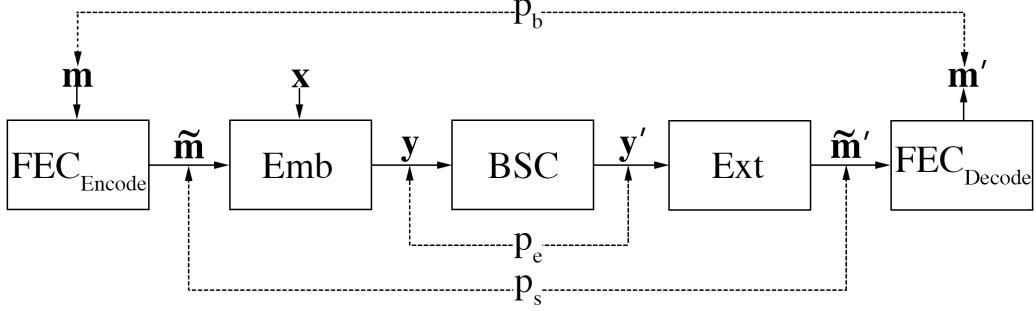


Figure 3.4: Notation for layering FEC within steganography.

3.3 Dual Syndrome Trellis Codes

We develop STCs into *Dual-Syndrome* Trellis Codes, that solve for two syndromes simultaneously. We formalise the embedding as

$$Emb(\mathbf{x}, \mathbf{m}) = \underset{\substack{\mathbb{H}_s \mathbf{y} = \mathbf{s} \\ \mathbb{H}_m \mathbf{y} = \mathbf{m}}}{\operatorname{argmin}} D(\mathbf{x}, \mathbf{y}), \quad (3.16)$$

where $\mathbf{s}$ is the shared syndrome of length l , $\mathbb{H}_s$ and $\mathbb{H}_m$ are matrices generated by tiling sub-matrices $\hat{\mathbb{H}}_s$, and $\hat{\mathbb{H}}_m$ respectively.

We define $\hat{\mathbb{H}}_m \in \{0, 1\}^w$ as the $1 \times w$ vector, used to generate $\mathbb{H}_m$, containing η_m ones on average in each column. $\hat{\mathbb{H}}_s$ is the $h \times w$ matrix used to generate $\mathbb{H}_s$, containing η_s ones on average in each column. As the matrix $\hat{\mathbb{H}}_m$ is of height one, the value of m_i is the mod2 accumulation of the w columns of $\hat{\mathbb{H}}_m$ multiplied by the respective values of $\mathbf{y}$. To encode this, we increase the number of states from 2^h to 2^{h+1} , allowing each state to have a representation for the partial syndrome setting the m_i to 0 or 1. Thus, the h LSBs of the state label represent the syndromes equivalence for setting the value of s_i , and the MSB represents the value of m_i . Upon coding w

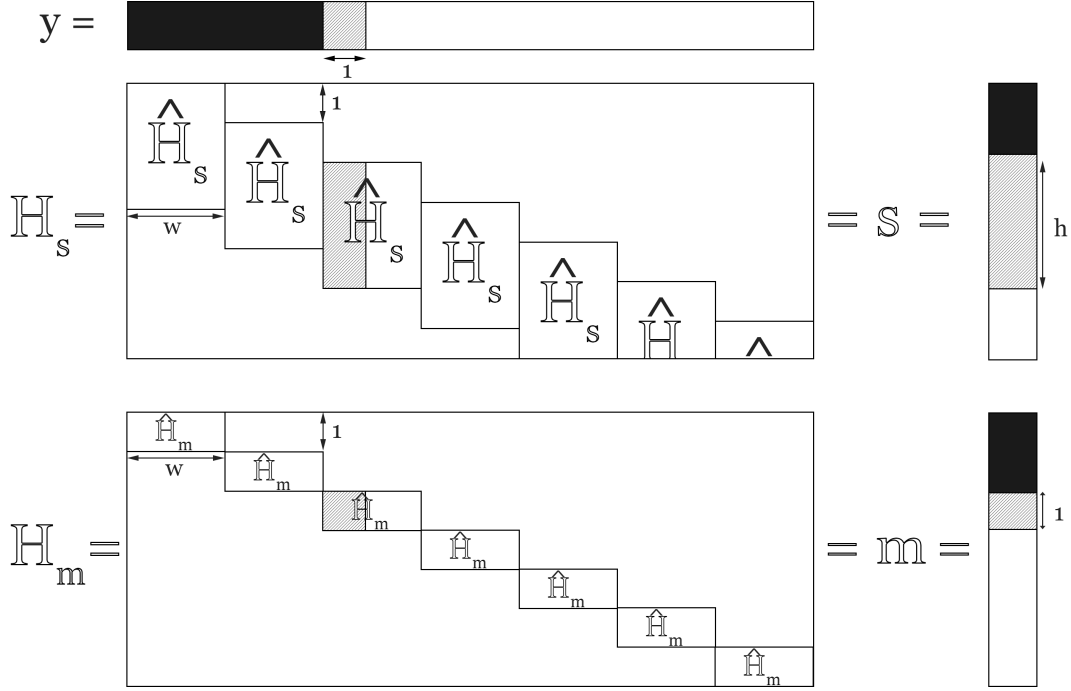


Figure 3.5: Partial DSTC encoding, darkened regions showing the set portion of $\mathbf{y}$, shaded regions showing how a single cover value can affect the result of h coefficients in $\mathbf{s}$ the FEC syndrome, and 1 coefficient in $\mathbf{m}$ the message syndrome.

coefficients we prune, only accepting states which LSB matches the shared syndrome s_i value, as this value will not change any more, and the MSB matching the value of the secret message m_i .

3.3.1 Decoding and Error Correction

As shown in (3.16) $\mathbf{y}$ is encoded to match both the shared syndrome $\mathbf{s}$, and secret message $\mathbf{m}$. However due to the noisy channel Bob receives $\mathbf{y}'$, the perturbed stego object. To correct $\mathbf{y}'$ we use an STC to solve $\mathbb{H}_s \mathbf{y}' = \mathbf{s}$. By using the hamming distance distortion function $D_H(\mathbf{y}', \mathbf{y}'')$, we use the Viterbi algorithm to find the closest code-word $\mathbf{y}''$, which has the correct

$\eta_m \backslash \eta_s$	$p_s \times 10^{-2}$				$D(\mathbf{x}, \mathbf{y}) \times 10^6$			
	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
0.2	0.19	0.22	0.24	0.40	36.66	36.21	36.35	46.00
0.4	0.26	0.24	0.28	0.35	26.20	31.38	31.30	39.15
0.6	0.44	0.41	0.43	0.39	18.35	19.40	20.69	28.30
0.8	0.57	0.57	0.41	0.39	17.37	18.37	20.75	29.19

Figure 3.6: p_s and $D(\mathbf{x}, \mathbf{y})$ for varying values of η_m , and η_s .

syndrome $\mathbf{s}$:

$$\mathbf{y}'' = \underset{\mathbb{H}_s \mathbf{y}'' = \mathbf{s}}{\operatorname{argmin}} D_H(\mathbf{y}', \mathbf{y}''). \quad (3.17)$$

We then get the corrected secret message $\mathbf{m}' = \mathbb{H}_m \mathbf{y}''$.

3.3.2 Finding Optimal Values of η_s , η_m , $\mathbf{s}$, and h

In Section 3.2.3 we showed that increasing the sparsity of $\mathbb{H}$ reduces p_s at a cost of increasing the distortion. In DSTCs we have two independent matrices $\mathbb{H}_s$ and $\mathbb{H}_m$. To find the optimal values of the sparsity, we performed DSTC embedding with η_s and $\eta_m \in \{0.2, \dots, 0.8\}$, using the same cover set and channel error as previous experiments. Syndrome error rates and distortion are displayed in Fig. 3.6, from this we selected $\eta_m = 0.4$, and $\eta_s = 0.2$.

During our experiments, we observed a curious phenomenon: in cover objects, syndromes are non-uniform, often favouring zeros. We can attribute this to JPEG compression creating more zeros than nonzeros. In traditional STCs, the steganographer has no choice over the syndrome – it is the message – but in DSTCs we could choose $\mathbf{s}$ to require fewest changes. Since $\mathbb{H}_s$ is secret, $\mathbf{s}$ need not be, and it seems safe to choose $\mathbf{s} = \mathbf{0}$, slightly reducing

distortion.

DSTCs allow for a constraint height to be used as would be in a regular STC. In practice, we found that although codewords produced with a $h > 1$ had a lower distortion, they had worse correcting performance. When $h = 1$, the codeword that satisfies the constraint (3.16) is similar to that which solves just $\mathbb{H}_s y = \mathbf{s}$, therefore, in the correction step $\mathbf{y}'$ is corrected to $\mathbf{y}''$, a codeword close to $\mathbf{y}$. When $h > 1$, the codeword which satisfies (3.16) is less similar to the one which satisfies just $\mathbb{H}_s y = \mathbf{s}$, meaning our correction step is less effective.

Deciding the composition of the shared syndrome $\mathbf{s}$ was an intriguing problem. Usually, the steganographer has no choice over the syndrome, as he is using the syndrome to convey his secret message. Interestingly this is close to the rejection sampler: sampling from the cover set to find the cover which contains the syndrome we wish to send, requiring exponential samples with respect to message length [16]. However, with using the shared syndrome we can pick the syndrome which is close to the cover. Choosing a syndrome close to the existing syndrome of the cover reduces the distortion, therefore, we should pick $\mathbf{s}$ as the pseudorandom vector, which is close to the existing cover syndrome.

3.4 Analysis

As this chapter aims to create robust adaptive stego objects, the balance of distortion minimisation and robustness is critical. For our experiments we took the same image set as used earlier and encoded them both with

DSTC methods as proposed in Section 3.3, and forward error correction in the syndrome as described in Section 3.2.4.

We first tested the effect of r —the cover utilisation, subjecting stego objects to a BSC with $p_e = 10^{-3}$, and DSTCs and STCS using $h = 1$, and $r \in \{0.75, 1.0\}$. In Figs. 3.7 and 3.8 we show how our proposed method is able to generate more robust stego objects per the amount of distortion, for smaller messages.

Using DSTCs, we embedded a smaller set of messages ($l \in \{1000, 1500, \dots, 6000\}$), and RS ($k \in \{200, 216\}$) encoded messages. For comparison, we used RS codes ($k \in \{120, 128\}$) before embedding using a traditional STC. We chose the values of k for STCs + RS codes, to be similar in average distortion to our proposed method so that we could draw comparisons. We also included the case $r = 0$ ($l = n$, plain overwriting), with no error correction.

Considering our findings in Section 3.2, a distortion over 10^7 is most likely very detectable, and for payloads below this threshold our proposed method outperforms STCs + RS codes. In Fig. 3.9 we see that for longer payloads, STCs + RS codes outperforms our proposed method, but this is easily detectable. We can attribute this fall off as typical of FEC: as the code length increases, the bit-error-rate approaches zero.

Showing the effect of lowering the k values of the RS code, we compared DSTC+RS and STC+RS over the same values of (n, k) RS code parameters, and using message lengths $l \in \{2000, 2500, \dots, 5500\}$. Using the same cover set as earlier, we show the results using a channel error $p_e = 10^{-3}$ in Figures 3.10 and 3.11, and $p_e = 10^{-2}$ in Figures 3.12 and 3.13. We show that

our method (with few exceptions) can produce a lower message bit error rate, for a given amount of distortion, when compared to STC+RS on a channel error of $p_e = 10^{-3}$. More interestingly, on a higher channel error, $p_e = 10^{-2}$ our method consistently and significantly outperforms STC+RS codes. We attribute this to the two codes used in our method; a relatively strong outer code—the Dual-Syndrome coding which can reduce the randomly spread errors, and a weaker inner code—the RS encoded payload which can reduce burst errors. The combination of the two codes is clearly very powerful, and is analogous to an interleaved CIRC code, as explained in Section 2.8.5.

3.5 Conclusion

STCs are the dominant method for adaptive steganography, but they have very poor robustness to non-adversarial noise. We have explored ways to mitigate their fragility, a topic that has received little attention in the literature. Adding robustness is bound to increase distortion, but robustness and adaptive steganography are a problematic combination: a code with low distortion requires codewords to be close together (so that a stego object can be close to the cover), yet high robustness needs codewords that are far apart. The Dual-Syndrome code attempts to balance these aims, and slightly outperforms the layering of FEC inside an STC with low constraint height for small messages for smaller channel errors, and significantly outperforms for larger channel errors. However, with a constraint height $h = 1$, we are effectively using block codes, and the Dual-Syndrome code is reduced to solving two block codes simultaneously.

If recompression cannot be avoided (and its erasures cannot be predicted, which is a tacit assumption of this chapter), it may be necessary to try adaptive steganography that does not use syndrome codes at all. This is a possible direction for further research.

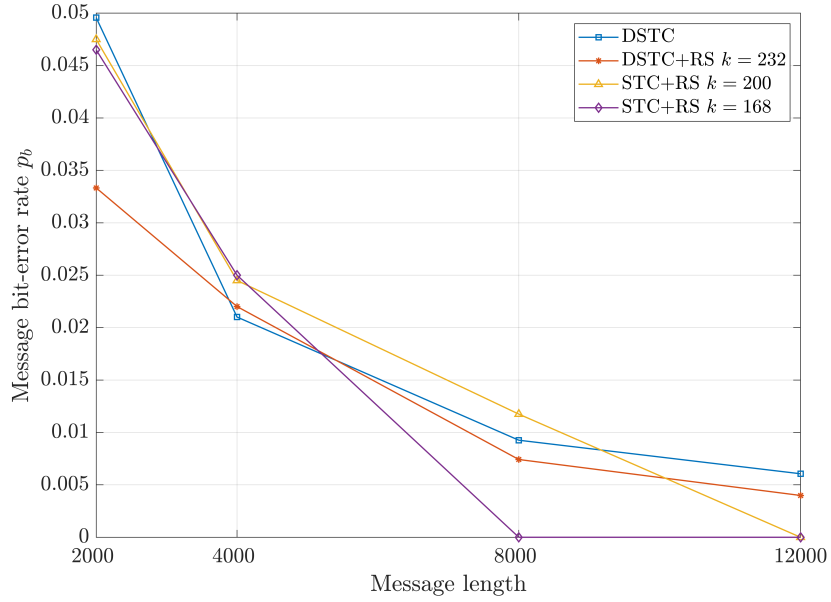
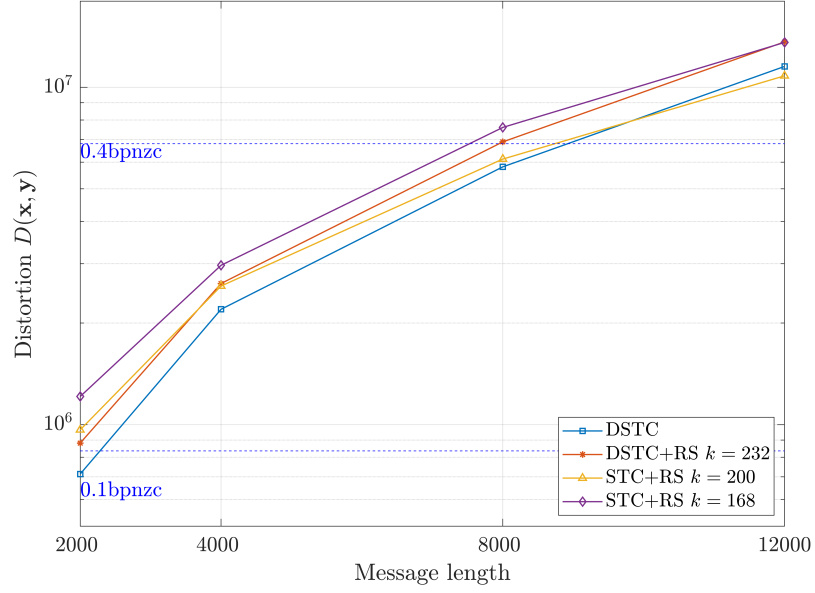


Figure 3.7: Comparison of results for cover utilisation $r = 1.0$.

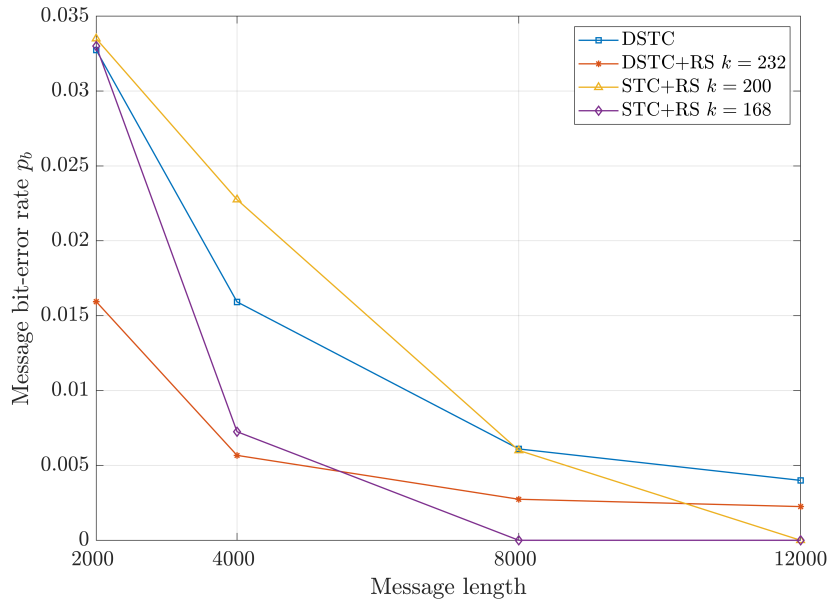
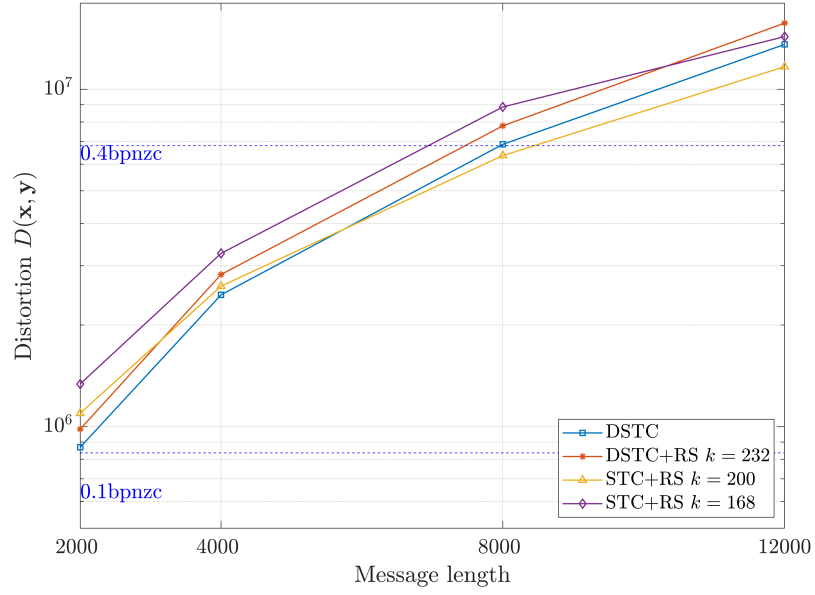


Figure 3.8: Comparison of results for cover utilisation $r = 0.75$.

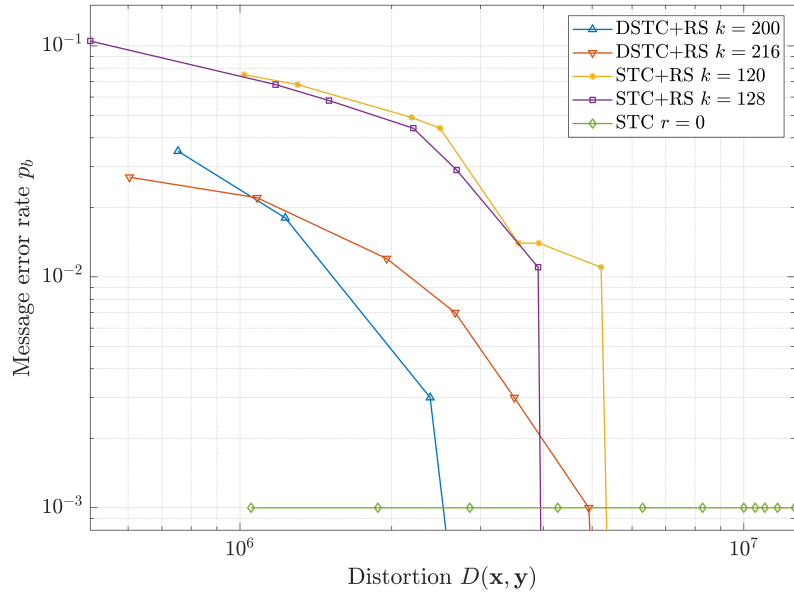


Figure 3.9: Comparison of all the methods explored in this chapter. Average distortion $D(\mathbf{x}, \mathbf{y})$ vs. bit error rate of corrected payload (p_b), for combinations of STC, DSTC, and RS codes, $l \in \{1000, 1500, \dots, 6000\}$ (left to right).

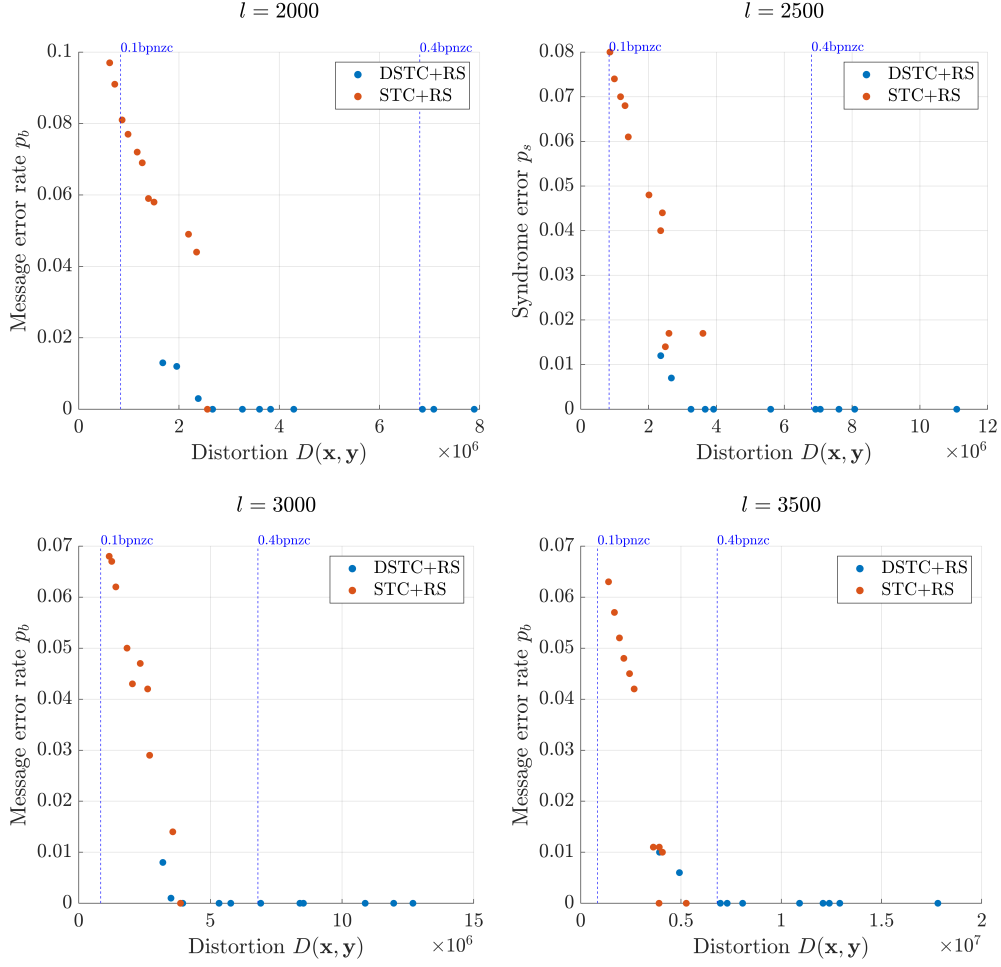


Figure 3.10: Comparison of DSTC+RS and STC+RS of average distortion $D(\mathbf{x}, \mathbf{y})$ and bit error rate of message p_b , for values of $(n, k) \in \{(248, 104), (248, 112), \dots, (248, 232)\}$, and $l \in \{2000, 2500, 3000, 3500\}$. Channel error $p_e = 10^{-3}$.

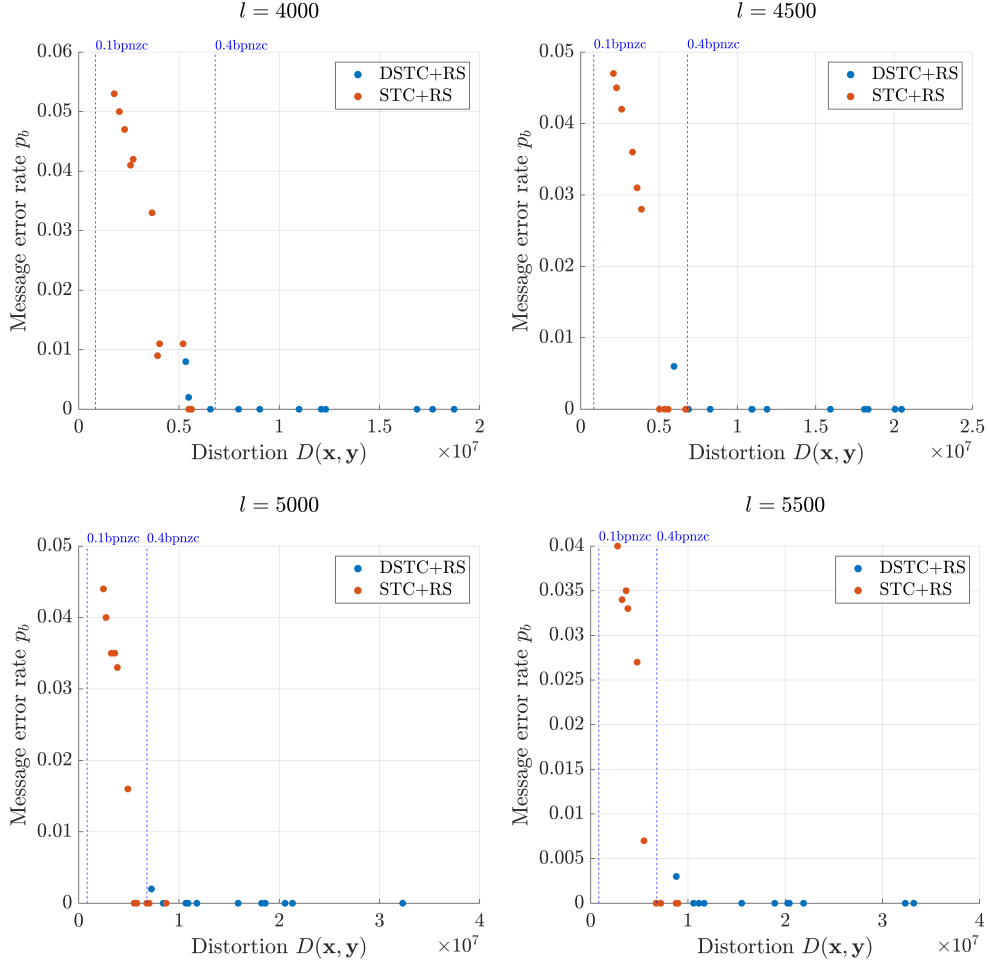


Figure 3.11: Comparison of DSTC+RS and STC+RS of average distortion $D(\mathbf{x}, \mathbf{y})$ and bit error rate of message p_b , for values of $(n, k) \in \{(248, 104), (248, 112), \dots, (248, 232)\}$, and $l \in \{4000, 4500, 5000, 5500\}$. Channel error $p_e = 10^{-3}$.

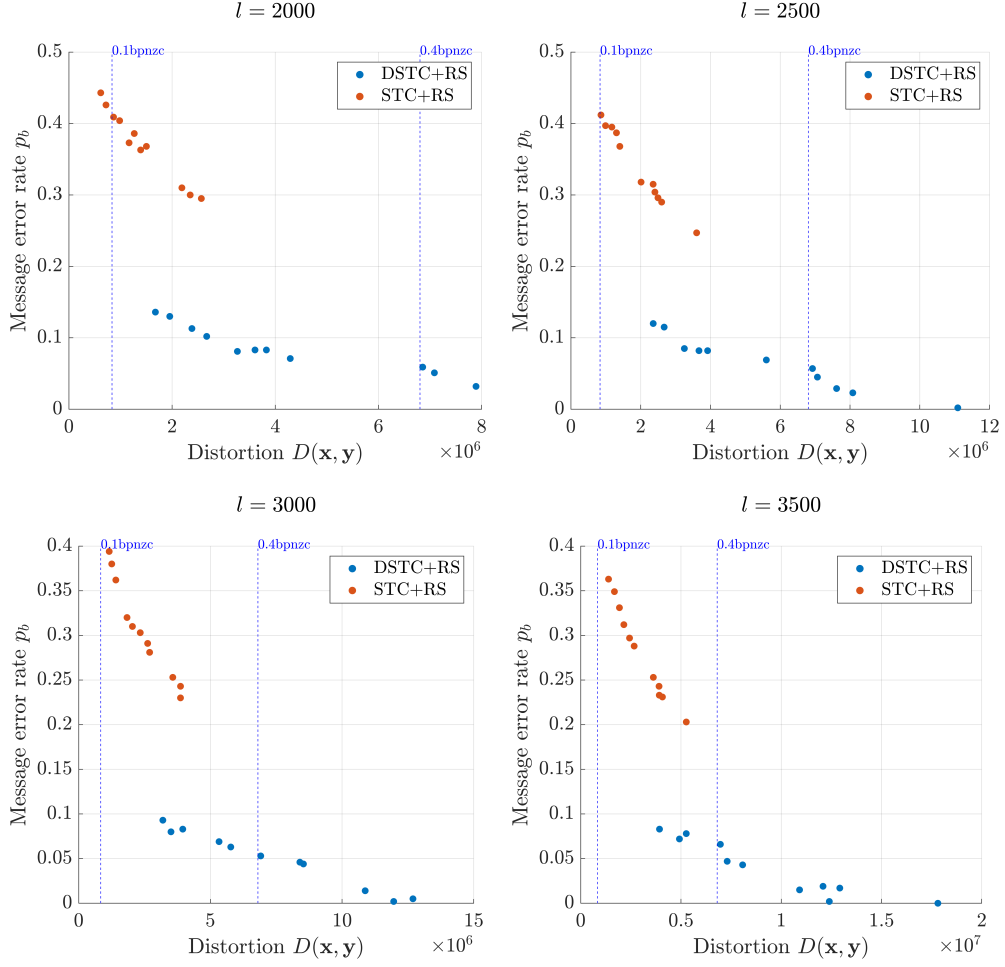


Figure 3.12: Comparison of DSTC+RS and STC+RS of average distortion $D(\mathbf{x}, \mathbf{y})$ and bit error rate of message p_b , for values of $(n, k) \in \{(248, 104), (248, 112), \dots, (248, 232)\}$, and $l \in \{2000, 2500, 3000, 3500\}$. Channel error $p_e = 10^{-2}$.

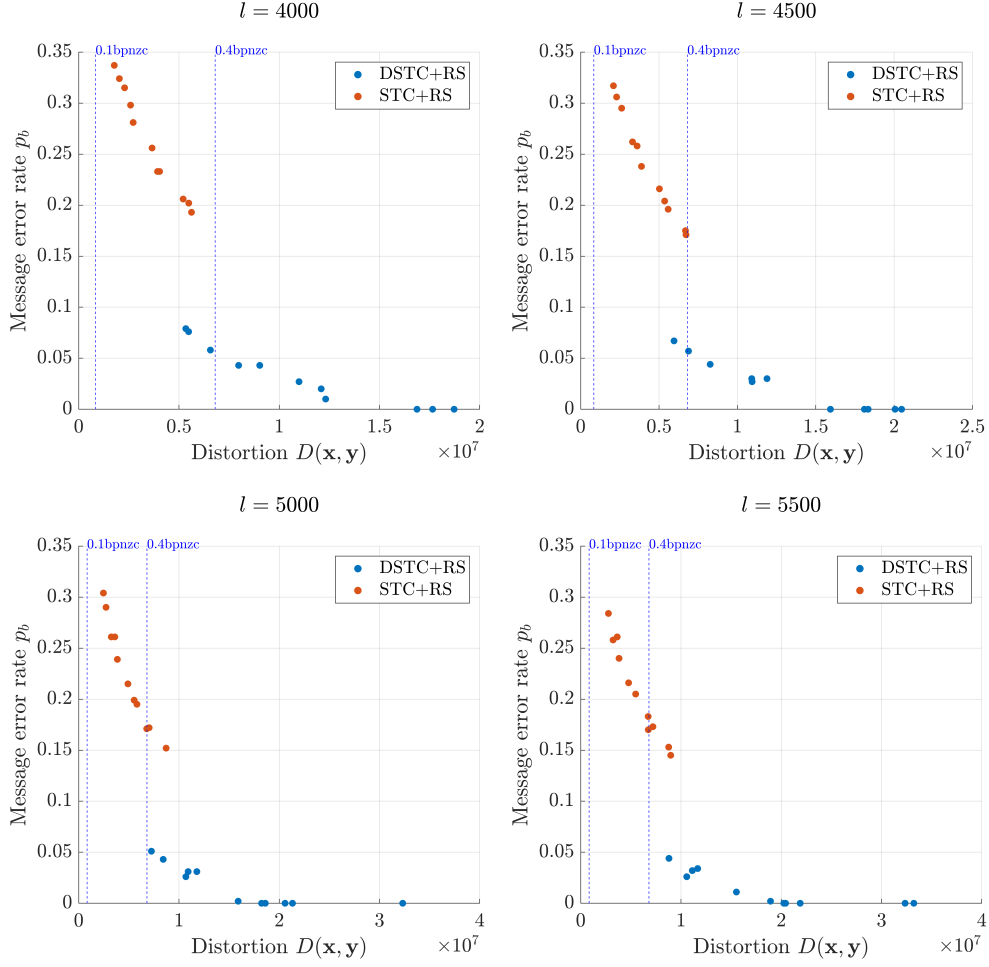


Figure 3.13: Comparison of DSTC+RS and STC+RS of average distortion $D(\mathbf{x}, \mathbf{y})$ and bit error rate of message p_b , for values of $(n, k) \in \{(248, 104), (248, 112), \dots, (248, 232)\}$, and $l \in \{4000, 4500, 5000, 5500\}$. Channel error $p_e = 10^{-2}$.

Chapter 4

Irregular Syndrome Trellis Codes

In this chapter, we look at the causes of coding loss in Syndrome Trellis Codes, and how coding loss can manifest itself through divergence from optimal change probabilities. Our contribution *Irregular Syndrome Trellis Codes* (IR-STC) aims to lower coding loss to increase embedding efficiency.

4.1 Statement of the Problem

As previously mentioned in Section 1.4, the researchers in the field of adaptive steganography have widely adopted distortion minimisation as the state of the art. To generate as secure as possible stego objects, the steganographer has two aims. The first is to preserve the higher-order statistical properties of the cover through a distortion function which accurately models the

cover source; the second is to maximise the embedding rate of the embedding function. By minimising the sum of additive distortion (as a proxy for detectability), we hope to maximise the security of our stego objects.

Improving the embedding efficiency using codes predates the idea of additive distortion functions, as discussed in further detail in Section 2.6. However, the introduction of the STC algorithm brought a framework allowing steganographers to use any additive distortion functions, and achieve close to optimal embedding efficiency. It is not surprising there has been a shift towards developing rich distortion functions as opposed to improving the embedding efficiency further. STCs are elegant in that the constraint height parameterises the embedding performance and time/space complexity of the algorithm h of the matrix $\hat{\mathbb{H}}$. Whilst becoming widely popular in the research community, there has been a smaller focus on the algorithm itself and how it could be improved. Most notably, the impact the construction of the parity check matrix $\mathbb{H}$ has on embedding performance.

We consider that not all changes are equal [1], meaning some changes may involve a higher “cost”, as it is assumed making these changes will introduce a greater amount of distortion to the resulting stego object. The optimal change probabilities for these changes can be derived using either the payload-limited sender constraint, or the distortion-limited sender constraint, as per (2.23)& (2.24) or (2.26)& (2.27), respectively.

From Section 2.5, we know PLS embeds a fixed message $\mathbf{m}$ of length m bits into an arbitrary length cover $\mathbf{x}$ of length n bits, whilst aiming to minimise the distortion $D(\mathbf{x}, \mathbf{y})$. Essentially, Alice wishes to communicate her existing message, by embedding it into a cover, with the minimum dis-

tortion. We quantify the gap between achieved distortion and the optimal distortion as coding loss [52]. The aim of any steganographic coding algorithm, therefore, is to minimise the coding loss. Further, if an algorithm makes a change with higher probability than those dictated by the derived optimal change probabilities, it must make some other changes with a lower probability than the optimum, to generate enough entropy to convey the message. We call these changes with significant divergence from the optimal change probabilities *outliers*.

In this work, we investigate modifications to the STC algorithm to reduce the number of outliers and thus achieve a lower coding loss. We propose a shift from using block sub-matrices for generating the parity check matrix used during STC encoding; instead, we promote using segment-vectors which are tiled to produce the parity check matrix. The tiling of these segments affects the number of outliers and hence the distortion. We show a practical method to generate these matrices and highlight the optimal parameters for embedding. We benchmark our method against the plain STC algorithm and the *Outlier-Corrected STC* as proposed by Köhler et al. [78].

Liu et al. [79] highlight that since the original proposal of the STC algorithm, there has been less work focused on improving the embedding efficiency of the algorithm, and the relationship between STCs and convolutional codes is a poorly known area. However, Liu et al.’s work focuses on reducing the space and time complexity of the STC algorithm by using a minimum span generator, whilst achieving the same embedding efficiency. To achieve this, a minimum trellis is constructed, by converting a regular parity check matrix to ‘minimal-span’ form using Gaussian elimination. The resulting

minimal span generator matrix (MSGM) has an LR property, in that no two rows in the matrix can have the same minimum and maximum span. Because of the structure of the MSGM, a fast solution to $\mathbb{H}\mathbf{y} = \mathbf{m}$ can be found using a minimal trellis, reducing the time and space complexity to find optimal solutions.

Köhler et al. [78] investigates how STCs perform with respect to optimal embedding probabilities. Köhler et al. proposes a variation of the STC algorithm: outlier corrected Syndrome Trellis Codes (OC-STC). By cropping the first $h - 1$ rows of $\mathbb{H}$, the matrix $\mathbb{H}$ becomes symmetrical, as the first and last $h - 1$ rows contain subsets of the matrix $\hat{\mathbb{H}}$. This reduces the payload by $h - 1$, as this is necessary to achieve the cropping. Experimentally, it was found that OC-STCs mitigated the positive outliers (values changed more frequently than optimal). Given negative outliers do not pose an intermediate security risk, the reduced efficiency of values (due to negative outliers still existing) is a sensible trade-off.

There have been several steganographic implementations utilising random-linear codes. Fridrich and Sockal has proposed several schemes including Simplex Codes [80], and Wet Paper Codes [63, 81]. Both methods have since been rendered obsolete by the more powerful STC algorithm, yet both are based on random-linear codes under the premise that random-linear codes can approach the theoretic bound for embedding efficiency [82, p325, Theorem 12.3.5].

4.2 Divergence from Optimal Change Probabilities

In this section, we introduce the reader to several methods of analysis used to quantify the performance attribution of our work.

4.2.1 Single-cover Single Value Divergence

To analyse the single-cover value divergence from optimal change probabilities as described in (2.28), we can use the Hellinger distance. We use the Hellinger distance as it can be used to describe the similarity of two values from different probability distributions, in the range $[0, 1]$ [83]. Let $\mathbf{p}$ and $\hat{\mathbf{p}}$ be the vectors of optimal and observed change probabilities respectively. Then p_i , and $\hat{p}_i$ are the i^{th} value of the optimal, and observed change probabilities from some stego object $\mathbf{y}$ of length n . For each value of the stego object, we calculate the Hellinger distance as

$$H(p_i, \hat{p}_i) = \sqrt{\sqrt{1 - p_i}\sqrt{1 - \hat{p}_i} + \sqrt{p_i}\sqrt{\hat{p}_i}}. \quad (4.1)$$

The per-element Hellinger distance is the divergence from the optimal change probability. If an embedding implementation generates positive outliers ($\hat{p}_i > p_i$), then there is both a chance of the outliers being used to identify stego objects, and the embedding algorithm cannot be achieving the optimal distortion.

4.2.2 Single-cover Multiple Value Divergence

To analyse outliers across the entire cover, we use the Kullback–Leibler (KL) divergence. The KL -divergence is a measure of how similar two entire distributions are [83], unlike the Hellinger distance, which describes the distance of two individual values. Since $\mathbf{p}$, and $\hat{\mathbf{p}}$ are Bernoulli distributions, the KL -divergence is:

$$KL(\mathbf{p}, \hat{\mathbf{p}}) = \sum_{i=1}^n p_i \log \left(\frac{p_i}{\hat{p}_i} \right) + (1 - p_i) \log \left(\frac{1 - p_i}{1 - \hat{p}_i} \right). \quad (4.2)$$

Therefore, the embedding algorithm with the lowest KL -divergence is one which achieves less divergence from the optimal change probabilities, and a distortion closer to the optimal distortion.

4.2.3 Binary Entropy of Changes

- Using (2.25) we can calculate the binary entropy of the observed embedding changes as:

$$\hat{H}(\hat{\mathbf{p}}) = \sum_{i=1}^n H(\hat{p}_i). \quad (4.3)$$

Since the maximum payload that any embedding algorithm is the entropy of the changes, using the entropy as a theoretical bound allows us to quantify the coding loss of the embedding operation [1, 84]. Thus, the embedding algorithm which generates the lowest binary entropy per the same message embedded has a lower coding loss.

4.2.4 Distortion

As covered in Section 1.4 and Section 2.5, it is understood that there are limitations in adaptive steganography, mainly that interacting changes cannot be accounted for with additive distortion functions. Yet distortion minimisation is well considered a proxy for detectability [59].

For a cover set $\mathbf{X}$ of size k , $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$, we encode the messages $\{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_l\}$ of length m bits, resulting in the distortion result set $\mathbf{Z}$ of length $v = kl$:

$$\mathbf{Z} = \{D(\mathbf{x}_i, \text{Emb}(\mathbf{x}_i, \mathbf{m}_j)) \mid i = 1 \dots k, j = 1 \dots l\} \quad (4.4)$$

Over a large set of embedding, we can consider the variance of the distortion to give further insight into the distribution of distortion values per embedding algorithm. For this we use sample variance:

$$\sigma^2 = \sum_{i=1}^v \frac{1}{n-1} (z_i - \bar{z})^2, \quad (4.5)$$

where z_i is the individual distortion value, and $\bar{z}$ is the average distortion across $\mathbf{Z}$ for a particular embedding algorithm. We can then calculate the standard error of the mean, SE , as:

$$SE = \sqrt{\frac{\sigma^2}{v}}. \quad (4.6)$$

We consider the skewness of the distribution of distortion. Since skewness measures the symmetry of values around the mean, a negative skew implies

there is a greater distribution of values greater than the mean, whereas a positive skew implies a greater distribution of values lower than the mean. We calculate the skewness as:

$$\gamma_1 = \frac{\frac{1}{v} \sum_{i=0}^v (z_i - \bar{z})^3}{(\sigma^2)^{\frac{3}{2}}} \quad (4.7)$$

Since we generate each individual distortion value per-embedding, we can use a Z-test to test the statistical difference between two embedding algorithms. The Z-test determines if two distributions are significantly different (the null hypothesis), given their mean and variance. The Z-test is constructed as follows:

$$z = \frac{\bar{z}_a - \bar{z}_b}{\sqrt{\frac{\sigma_a^2}{v} + \frac{\sigma_b^2}{v}}}, \quad (4.8)$$

for two distortion distributions z_a , and z_b , with the respective means and variances $\bar{z}_a$, $\bar{z}_b$, and σ_a^2 , σ_b^2 . With a resulting value greater than 1.96, or lower than -1.96 (the 95% confidence interval), the null hypothesis is rejected.

4.3 Irregular Syndrome Trellis Codes

From Köhler et al.'s work [78], it became clear that the improvement over the regular STC was due to each row in $\mathbb{H}$ containing the same values. To tile the $\mathbb{H}$ matrix in such a way required $h - 1$ rows to be truncated, reducing the payload by $h - 1$ bits. However, the requirement to truncate $h - 1$ rows only comes from using a $\hat{\mathbb{H}}$ matrix to tile the $\mathbb{H}$ matrix, as proposed in [59, 45]. In

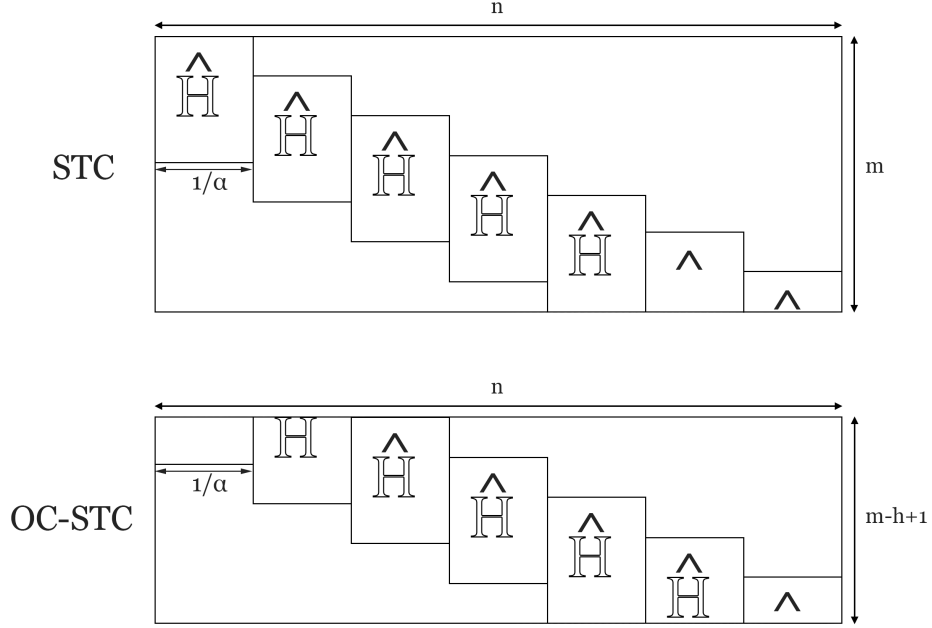


Figure 4.1: $\mathbb{H}$ matrix tiled with $\hat{\mathbb{H}}$ for both STC and OC-STC algorithms.

Fig. 4.1 we show the STC and OC-STC matrix tiling. Critically, in Köhler's proposed Outlier-Corrected STC, each row contains the same sequence, of length $h\frac{1}{\alpha}$, for a constraint height h . However, when tiling the matrix using $\hat{\mathbb{H}}$ matrices, we have a symmetrical tiling, each sub-matrix is of width $\frac{1}{\alpha}$, which requires $h - 1$ rows to be truncated.

If we look at building $\mathbb{H}$ matrices with the sole requirement of containing the same row, or at least the same number of ones, then we relax the constraints on how we can generate said matrices. To generate a $n \times m$ matrix with exactly the same *segment* (sequence of non-zero values) we first fix the length of the segment as $s = h\frac{1}{\alpha}$. Since we have m rows, if we tile each segment with a gap of $\frac{1}{\alpha}$, the length of the matrix would be $\alpha(m - 1) + s$, which is larger than n . However, as we no longer have a constraint of how

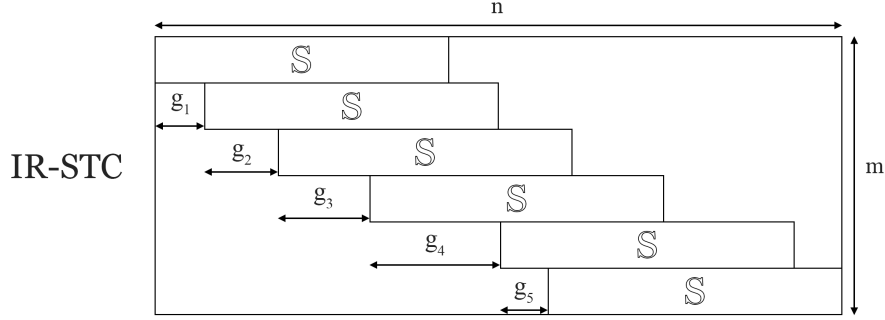


Figure 4.2: $\mathbb{H}$ matrix generated using segments.

we tile the matrix, we can reduce the gaps between segments in any manner, as long as the tiling produces a matrix $\mathbb{H}$ of dimensions $n \times m$. To describe this, let $\mathbf{g} = \{\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_{m-1}\}$ be the set of gaps between each row as shown in Fig 4.2, the *gap profile*. We now have the simple constraint of:

$$n = s + \sum_{i=0}^{m-1} g_i, \quad (4.9)$$

which allows any $n \times m$ sized $\mathbb{H}$ matrix to be constructed, with the same segment, or at least, the same number of ones.

We can modify the STC algorithm to handle the irregular $\mathbb{H}$ matrix in two steps; first, we decompose the matrix $\mathbb{H}$ into irregular blocks at each message bit(s) pruning. Then for each irregular block in the matrix, we generate the sub-trellis of possible syndromes, note how each block may be of different dimensions as shown in Fig 4.3. There now exists more questions about how exactly we should generate these matrices. In this section, we aim to explore the possible questions around the matrix generation and how they impact performance. For all experiments in this paper, we will use costs generated with the WOW distortion function [46], following the experimental setup

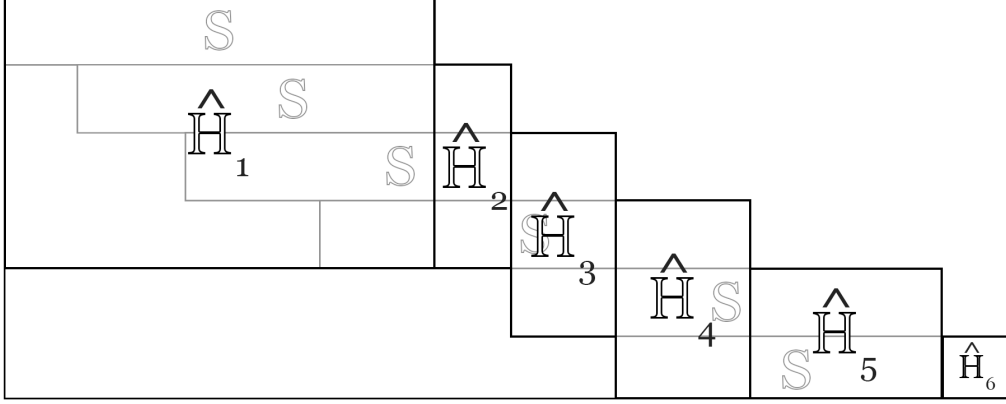


Figure 4.3: An irregular $\mathbb{H}$ matrix can be decomposed into $\hat{\mathbb{H}}$ sub-matrices. Note how sub-matrices can be different widths and heights. Empty spaces are filled with zeros.

of [78]. Our experiments to find optimal parameters will use 1,000 covers from the BOSSbase set [61], cropped at random to 8×8 , $n = 64$ cover crops. Using a message set of 1,000 messages of length $m = 32$.

4.3.1 Segment Based Matrices

With the move to segment-based matrices, our first experiment is to determine the best method to construct the rows of the matrix. From earlier, we discussed how random-linear codes could approach the theoretic bound for embedding efficiency; however, finding the lowest distortion syndrome for a randomly generated matrix would be an NP-hard problem. We can, however, use randomly generated segments to construct the matrix. To test random segments versus repeated segments, we generated 100 $\mathbb{H}$ matrices for randomly generated segments and repeated segments. All segments had the constraint of the first and last $\frac{1}{\alpha}$ values being set to one, to ensure a large

Method	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
Random segments	72,043	1.225×10^{12}	109.40
Repeated segments	71,755	1.197×10^{12}	109.40

Table 4.1: Average distortion, distortion variance, and SE for random and repeated segments in $\mathbb{H}$.

branching factor during trellis construction.

From Fig 4.1, we see that repeated segments slightly out-perform randomly generated segments. To quantify if the difference is significant, we use the Z-test as in (4.8):

$$\frac{72,042.95 - 71,755.34}{\sqrt{\frac{1.225 \times 10^{12}}{2 \times 10^8} + \frac{1.197 \times 10^{12}}{2 \times 10^8}}} \approx 2.614, \quad (4.10)$$

since the result of the z-statistic lies outside of the 95% confidence interval (± 1.96), it is clear that repeated segments significantly outperform random segments.

4.3.2 Tiling the Matrix

As each row in the matrix $\mathbb{H}$ contains exactly one segment of length s , there exists a tiling problem of how best to construct the matrix. Since we have m segments of length s , and we wish to construct a banded matrix to allow partial syndromes to be found in linear time, we wish to find some *gap profile* $\mathbf{g}$ (the placement of gaps between segments) to produce a perfect tiling which also performs best in terms of distortion minimisation.

If we consider the problem as highlighted in 4.9, we can come up with four basic cases:

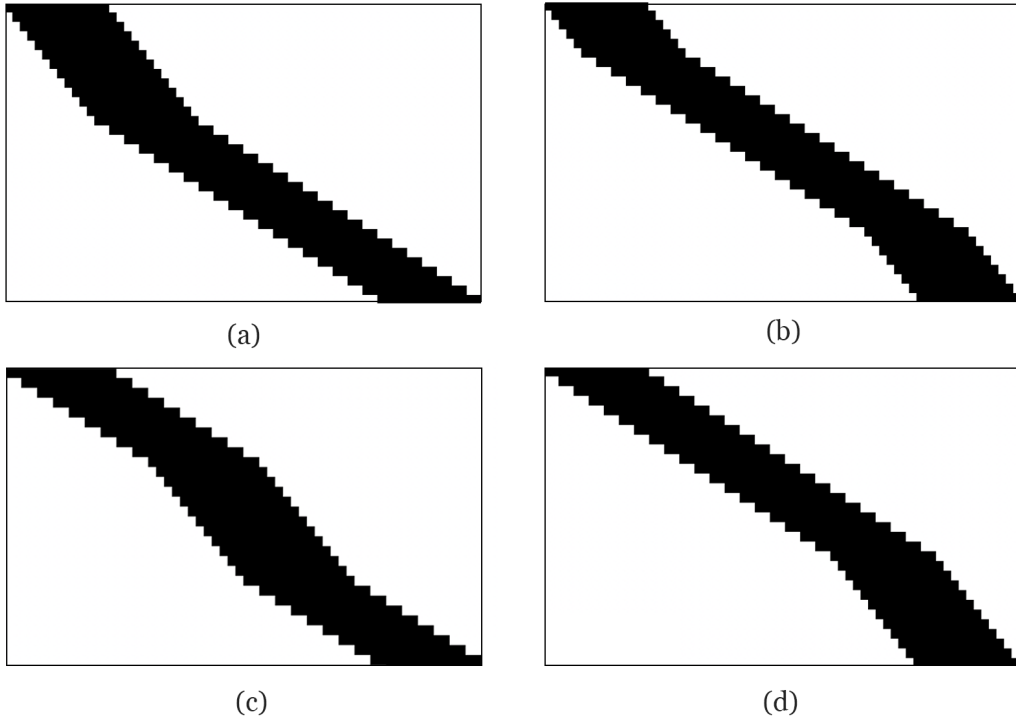


Figure 4.4: The effects of smaller gaps at different parts of the $\mathbb{H}$ matrix; (a) at the start, (b) at the start and end, (c) in the middle, (d) at the end. Dark regions are the non-zero sections of the $\mathbb{H}$ matrix.

- (a) **Smaller gaps at the beginning.** Create the gap profile g , with smaller values at the beginning, creating a denser matrix at the start of the trellis.
- (b) **Smaller gaps at the beginning and end.** Create the gap profile g , with smaller values at the start and end, creating a denser matrix at the start and end of the trellis.
- (c) **Smaller gaps in the middle** Create the gap profile g , with smaller values in the middle, creating a denser matrix at the middle of the trellis.
- (d) **Smaller gaps at the end.** Create the gap profile g , with smaller

Method	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
Smaller gaps at the beginning	74,446	6.932×10^{10}	110.86
...at the beginning and end	72,182	6.145×10^{10}	117.74
...in the middle	76,661	7.943×10^{10}	116.73
...at the end	74,324	6.813×10^{10}	126.04

Table 4.2: Average distortion, distortion variance, and SE for methods of tiling $\mathbb{H}$.

values at the end, creating a denser matrix at the end of the trellis.

The four cases can be seen in 4.4. To test the four cases, we use the same experimental setup as in the previous experiment, however, this time we generate five new $\mathbb{H}$ matrices for each of the four tiling options, using repeated segments given the findings of the last experiment.

From fig 4.2, we can see that compression at the start and end of the matrix gives the best performance by quite a margin. The z-statistic between the two best methods (smaller gaps at the beginning and end, and smaller gaps at the end) is 26.609, indicating a significant performance gain for the former method. We attribute this to the increased density at the start and end of the trellis. With the additional states, the STC algorithm can find more solutions, avoiding positive outlier change probabilities. Reducing these outliers allows a lower distortion since values are not changed with higher observed probability.

4.3.3 η - Density of the Segment

The density of ones in the segments $\mathbf{s}$ will greatly affect the performance. When constructing the trellis, a new state address is given by the XOR

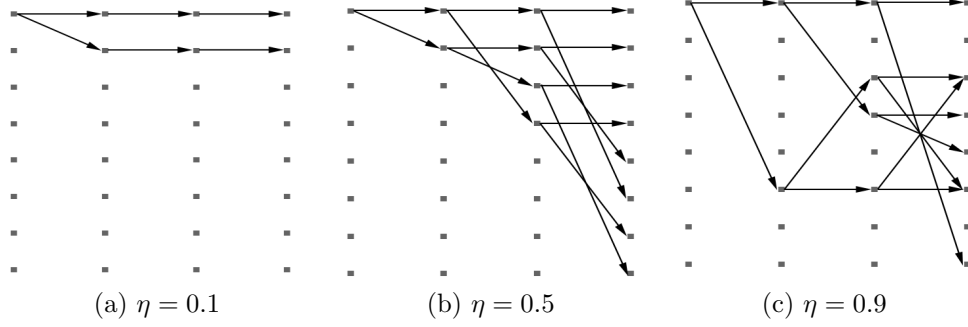


Figure 4.5: Trellis generation for different values of η .

product of the old state label, and the column in $\mathbb{H}$. If the column in $\mathbb{H}$ is entirely zeros or ones, the likelihood that similar or identical columns exist. In the scenario of two columns being identical, a cycle between states occurs, which is wasteful. If x_i and x_j are two cover values for the i^{th} , and j^{th} (identical) columns of $\mathbb{H}$ respectively, then the assignment $x_i = 0, x_j = 0$ and $x_i = 1, x_j = 1$ will create two paths from each state at x_i to exactly one state in x_j . As one path will have lower distortion, we have halved the number of useful paths through the trellis between x_i , and x_j .

To maximise the number of distinct paths through the trellis, we want to generate columns in $\mathbb{H}$ that are as different as possible. As mentioned earlier, as the rate of zeros or ones approaches 1.0, the probability of duplicate columns will increase. First, let η be the rate of ones in each segment $\mathbf{s}$. Specifically the number of ones in each segment $\mathbf{s}$ is $\lfloor \eta s \rfloor$.

To test this, we again use the same experimental setup as before. We generate five $\mathbb{H}$ matrices for each density $\eta \in \{0.2, 0.3, \dots, 0.8\}$.

In Table 4.3 we show the distortion values for the different values of η . As expected, the extreme values of η , when the segments approach entirely

η	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
0.2	93,561	1.150×10^{11}	151.65
0.3	82,484	9.438×10^{10}	137.39
0.4	74,456	6.670×10^{10}	115.47
0.5	73,280	6.189×10^{10}	102.47
0.6	72,476	6.189×10^{10}	111.25
0.7	72,459	6.283×10^{10}	112.09
0.8	73,076	6.394×10^{10}	113.09

Table 4.3: Average distortion, and distortion variance for methods of different values of η .

being constituted of zeros or ones, perform significantly worse. At first, we may expect the value $\eta = 0.5$ to perform best, given the probability of distinct segments is the greatest. However, the more zeros in each column reduce the number of paths through the trellis, and therefore the number of solutions. We attribute this explanation to why the slightly higher $\eta = 0.7$ value performs best. The z-statistic between eta values $\eta \in \{0.6, 0.7\}$ is 0.280 indicating there is not a large significance between the two values.

4.3.4 γ - the Skew of the Matrix

Up to now, we have shown that when we move from tiling $\hat{\mathbb{H}}$ matrices to generate $\mathbb{H}$ we remove certain constraints and allow a more flexible way to generate the matrices. Specifically, we reduce the tiling constraint to as previously introduced in (4.9). In previous sections, we have shown that a gap profile which reduces the space at the start and end of the matrix performs best. We now introduce a method to produce the tiling and parameterise the skew of the matrix. By skew, we refer to how aggressively the density is increased at the start, and end of the matrix. For comparison, a regular STC

Algorithm 2 γ tiling

```
1: procedure TILING( $n, m, \gamma, r$ )
2:    $m' \leftarrow m$ 
3:    $n' \leftarrow n - r$ 
4:   while  $i < m$  do
5:      $\mathbf{g}_i = \text{round}((1 - \gamma) \frac{m'-2}{m-3} \frac{n'}{m'-1})$ 
6:      $n' = n' - \mathbf{g}_i$ 
7:      $m' = m' - 1$ 
   return  $\mathbf{g}$ 
```

γ	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
0.1	72,139	6.206×10^{10}	111.41
0.3	72,139	6.206×10^{10}	111.41
0.5	73,468	6.484×10^{10}	113.88
0.7	78,454	6.083×10^{10}	123.69

Table 4.4: Average distortion, and distortion variance for methods of different values of η .

$\mathbb{H}$ matrix would have no skew. Using $\gamma \in [0, 1]$ we can generate the tiling of the matrix using algorithm 2.

To test values of gamma, we use the same experimental setup as before, generating four $\mathbb{H}$ matrices for the gamma values $\gamma \in \{0.1, 0.3, 0.5, 0.7\}$.

Lower values of gamma are better for distortion minimisation. We attribute this to the lower effective constraint height in the middle of the trellis, at higher gamma values the contrast between the height of the trellis in the middle, versus the start and ends is greater. The z-statistic between the gamma values $\gamma \in \{0.1, 0.5\}$ is 22.07, indicating a significant performance gain for lower values of gamma.

Because of the rounding of the gaps during the tiling, two gamma values can produce the same tiling. This is evident as the values $\gamma \in \{0.1, 0.3\}$

Method	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
IR-STC	67,187	4.791×10^9	69.21
OC-STC	69,756	5.123×10^9	71.58
STC	70,253	5.315×10^9	72.90

Table 4.5: Average distortion, distortion variance, and SE for IR-STC, OC-STC, and STCs, with $8 \times 8, n = 64$ cover crops.

produced the same tiling in our experiment.

4.4 Analysis

In this section, we benchmark our proposed method against the state of the art. We start with cover crops as used in earlier experiments, before moving to larger sized covers; finally, we test our method on favicon images.

4.4.1 $8 \times 8, n = 64$ Cover Crops

To compare our method against the state of the art, we measured achieved distortion and mitigation of outliers. We compared our proposed method to the STC and Outlier-Corrected STC algorithms using the best parameters found in previous sections. Using the $8 \times 8, n = 64$ cover crops from previous experiments, we first benchmarked our method minimisation against the others. The rationale behind using small covers is time complexity. Over a set of 1,000 covers and 1,000 messages, we must do 1,000,000 embeddings per method, before any further parameters such as α , or h are considered. By using smaller covers, we reduced the time required for earlier experiments.

Our results in Table 4.4.1 show that our method does indeed have a lower

distortion. With the reduction in distortion far larger than the standard error, we show our method to out-perform both the STC and OC-STC.

4.4.2 64×64 , $n = 4096$ Cover Crops

To better simulate practical applications we increased the cover sizes. We took a cover set of 1,000 BOSSBase images, cropped at random to 64×64 , $n = 4096$ and embedded 1,000 randomly generated messages with $\alpha \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$. For comparison, we used a STC, and OC-STC with constraint heights $h \in \{7, 8, 9\}$, which for $\alpha = \frac{1}{2}$ gives segments of length $s \in \{14, 16, 18\}$, and for other α gives longer segments. We see that our method does reduce distortion, which we attribute to the mitigation of positive outliers, which in turn, achieves change probabilities closer to the optimal change probabilities. In Table 4.6, we show how our method achieves a slightly lower distortion when compared to the other methods. Our method achieves a slightly lower distortion. From Section 2.5 we know the rate-distortion bound, or *optimal* distortion is given by:

$$\sum_{i=1}^n c_i p_i, \quad (4.11)$$

where c_i is the i^{th} cover value cost of change, and p_i is the probability of change given whilst solving the payload-limited sender, or distortion-limited sender constraints.

Using the distortion coding loss, as defined in (2.31) the IR-STC method reduces distortion coding loss l_d by a tiny amount in the case of $\alpha = \frac{1}{2}$, but as much as 1.35% in the case of smaller payloads ($h = 7, \alpha = \frac{1}{8}$). Our

α	h	Method	$D(\mathbf{x}, \mathbf{y})$	SE	l_d	$\gamma_1(D(\mathbf{x}, \mathbf{y}))$	States reached
$\frac{1}{2}$	7	IR-STC	3,006,231	2,859.02	0.1252	8.174×10^{12}	532,476
		OC-STC	3,008,140	2,851.32	0.1259	8.183×10^{12}	522,362
		STC	3,008,176	2,860.42	0.1259	8.182×10^{12}	521,846
	8	IR-STC	2,954,325	2,812.47	0.1057	7.910×10^{12}	1,064,694
		OC-STC	2,956,160	2,813.89	0.1064	7.918×10^{12}	1,043,962
		STC	2,956,375	2,814.25	0.1065	7.920×10^{12}	1,042,678
	9	IR-STC	2,904,853	2,767.49	0.0872	7.659×10^{12}	2,189,664
		OC-STC	2,906,689	2,768.75	0.0879	7.666×10^{12}	2,093,566
		STC	2,906,893	2,769.12	0.0880	7.668×10^{12}	2,089,980
$\frac{1}{4}$	7	IR-STC	1,084,931	1,101.82	0.1696	1.214×10^{12}	545,268
		OC-STC	1,085,784	1,102.72	0.1706	1.216×10^{12}	519,794
		STC	1,086,099	1,102.72	0.1709	1.216×10^{12}	520,726
	8	IR-STC	1,063,323	1,081.20	0.1463	1.169×10^{12}	1,114,100
		OC-STC	1,064,463	1,082.13	0.1476	1.171×10^{12}	1,037,810
		STC	1,064,639	1,082.13	0.1478	1.171×10^{12}	1,040,518
	9	IR-STC	1,044,247	1,061.04	0.1258	1.129×10^{12}	2,287,604
		OC-STC	1,045,362	1,063.48	0.1270	1.131×10^{12}	2,072,050
		STC	1,045,816	1,063.95	0.1275	1.132×10^{12}	2,078,086
$\frac{1}{8}$	7	IR-STC	423,303	453.54	0.1893	2.057×10^{11}	570,815
		OC-STC	428,071	456.40	0.2027	2.083×10^{11}	519,794
		STC	428,121	456.29	0.2028	2.082×10^{11}	519,152
	8	IR-STC	417,767	445.98	0.1737	1.989×10^{11}	1,137,636
		OC-STC	418,618	446.77	0.1761	1.996×10^{11}	1,025,506
		STC	418,950	446.99	0.1770	1.998×10^{11}	1,032,846
	9	IR-STC	410,051	437.95	0.1520	1.918×10^{11}	2,353,124
		OC-STC	410,943	437.84	0.1545	1.917×10^{11}	2,043,362
		STC	411,229	438.75	0.1553	1.925×10^{11}	2,063,534
$\frac{1}{16}$	7	IR-STC	176,477	186.71	0.1853	3.486×10^{10}	585,668
		OC-STC	176,584	187.67	0.1858	3.522×10^{10}	504,386
		STC	176,603	188.02	0.1859	3.535×10^{10}	512,542
	8	IR-STC	171,817	191.00	0.1632	3.648×10^{10}	1,217,476
		OC-STC	172,342	191.47	0.1658	3.666×10^{10}	1,000,898
		STC	172,497	191.55	0.1665	3.669×10^{10}	1,020,894
	9	IR-STC	168,047	196.21	0.1444	3.850×10^{10}	2,542,532
		OC-STC	168,931	196.01	0.1489	3.842×10^{10}	1,985,986
		STC	169,098	196.19	0.1498	3.849×10^{10}	2,033,886

Table 4.6: Average distortion, distortion coding loss, variance, and number of trellis states reached for IR-STC, OC-STC, and STC, with 64×64 , $n = 4096$ cover crops.

method also has a lower distortion variance (itself an indicator that outliers have been reduced).

To better quantify outlier mitigation, we took a randomly chosen cover from the BOSSbase and cropped it to 64×64 , $n = 4096$, using all three methods, we encoded 10,000 randomly generated messages. We then plot observed change probabilities against optimal change probabilities, for constraint heights $h \in \{7, 9\}$, in Fig. 4.6. We also display the average KL divergence between observed and optimal probabilities, the achieved distortion, and the achieved change entropy. Note that IR-STC reduces the average KL divergence by around 10%, showing that outliers have been mitigated.

On the other hand, we have an increased number of states at the start and the end of our trellis, which determines the algorithm’s time complexity. This is caused by the smaller gaps between segments, effectively making the trellis denser at those points. Critically, this increases the constraint height; however, this is only for a tiny section at the start and end of embedding, because we decomposed the matrix $\mathbb{H}$ into variable-sized blocks. In fact, over the entire cover, we see a minimal increase in the number of states. The encoding process is still (pseudo-)linear time. The small increase in the number of states is seen in Table 4.6.

4.4.3 Favicons

Because our method favours smaller covers, we opted to test our method on *favicons*. Favicons are small images which are automatically downloaded when a web page is loaded [85]. They exist in standard dimensions of 16×16 ,

α	Method	$D(\mathbf{x}, \mathbf{y})$	$\sigma^2(D(\mathbf{x}, \mathbf{y}))$	SE
$\frac{1}{2}$	IR-STC	32,558	3.443×10^{11}	586.81
	OC-STC	43,210	8.631×10^{11}	929.04
	STC	33,440	2.596×10^{11}	509.47
$\frac{1}{4}$	IR-STC	6,717	3.680×10^9	60.66
	OC-STC	6,947	6.708×10^9	81.91
	STC	7,422	4.726×10^9	68.75
$\frac{1}{6}$	IR-STC	2,834	2.210×10^8	14.87
	OC-STC	2,798	2.295×10^8	15.15
	STC	3,203	2.268×10^8	15.06
$\frac{1}{6}$	IR-STC	1,730	4.262×10^7	6.53
	OC-STC	1,742	5.131×10^7	7.16
	STC	2,029	6.015×10^7	7.76

Table 4.7: Average distortion, variance, and standard error for IR-STC, OC-STC, and STC, with 16×16 , $n = 768$ favicons.

32×32 , 48×48 , and 64×64 pixels, both in greyscale and RGB colour. Because of favicons' small cover size, we decided to test the impact of our method here, since clearly, our method favours smaller covers.

To obtain a database of favicon images, we crawled the most popular (by traffic) websites listed on Alexa (<https://www.alexa.com>) until we had sourced 1,000 unique 16×16 , RGB favicons. With each favicon having three colour channels, we extracted costs using WOW for each channel, as such each favicon had $n = 768$ cover values.

Using 1,000 pseudo randomly generated messages for each payload rate $\alpha \in \{\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}\}$, we compared our method, with an STC and OC-STC. In Table 4.7 we show that our method is able to minimise the distortion slightly better than the other methods, consistent with our previous findings. To test detectability, we used *sample pairs* (SP) [86]. A sample pairs detector is more suitable for Favions, since more powerful methods such as feature based

extractors will not work on such small covers. SP produces three estimated change rates denoted:

$$\hat{\alpha}_R, \hat{\alpha}_G, \hat{\alpha}_B, \quad (4.12)$$

one per colour channel, the overall estimated rate of change can be found as:

$$\hat{\alpha} = \sum (\hat{\alpha}_R, \hat{\alpha}_G, \hat{\alpha}_B). \quad (4.13)$$

We can then generate the ROC curves based on the estimated change rate $\hat{\alpha}$ by iterating $\gamma \in \{\min(\hat{\alpha}), \dots, \max(\hat{\alpha})\}$. In Fig. 4.7 we show the ROC curves for IR-STC, OC-STC, and STCs. Although we're able to minimise distortion slightly more than IR-STC and STCs, the detectability remains constant.

4.5 Conclusion

STCs have become incredibly popular over the last few years. We have explored a new way of constructing the matrix $\mathbb{H}$ to achieve closer-to-optimal change probabilities, whilst achieving a lower distortion, without changing the Viterbi algorithm that underlies STCs. The change is to allow the trellis to have variable height, with more height at the start and end. We also stress the value of constructing $\mathbb{H}$ from row segments, rather than blocks. There has been little research into what makes a good parity matrix for STC performance. Whilst our improvement is small, the relative cost in terms of complexity is minimal.

We benchmarked our method against IR-STC and STCs with detectability using sample pairs, when embedding in favicons. Whilst we showed our method was able to slightly improve the distortion minimisation, and reduce positive outliers, this made a negligible impact on detectability.

Other ways in which $\mathbb{H}$ is constructed may also influence the performance of STCs. Note that convolutional codes, when used in error correction, are highly sensitive to the parity-check matrix (and good matrices are often found by exhaustion). Perhaps a steganographic analogy exists.

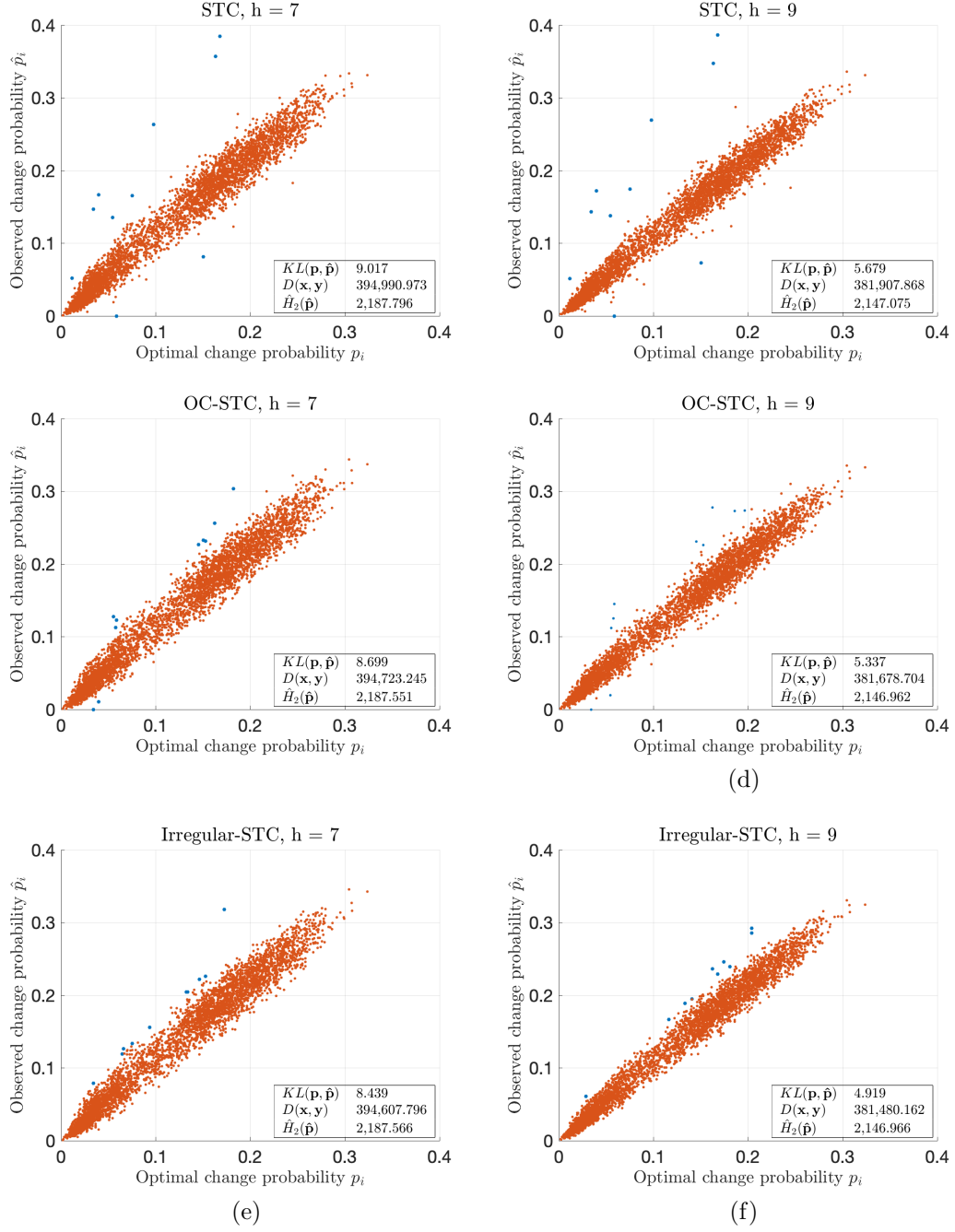


Figure 4.6: Optimal change probability versus observed change probability comparison between our proposed method, STC, and OC-STC. Blue markers the signify values with the largest Hellinger distance from optimal change probabilities.

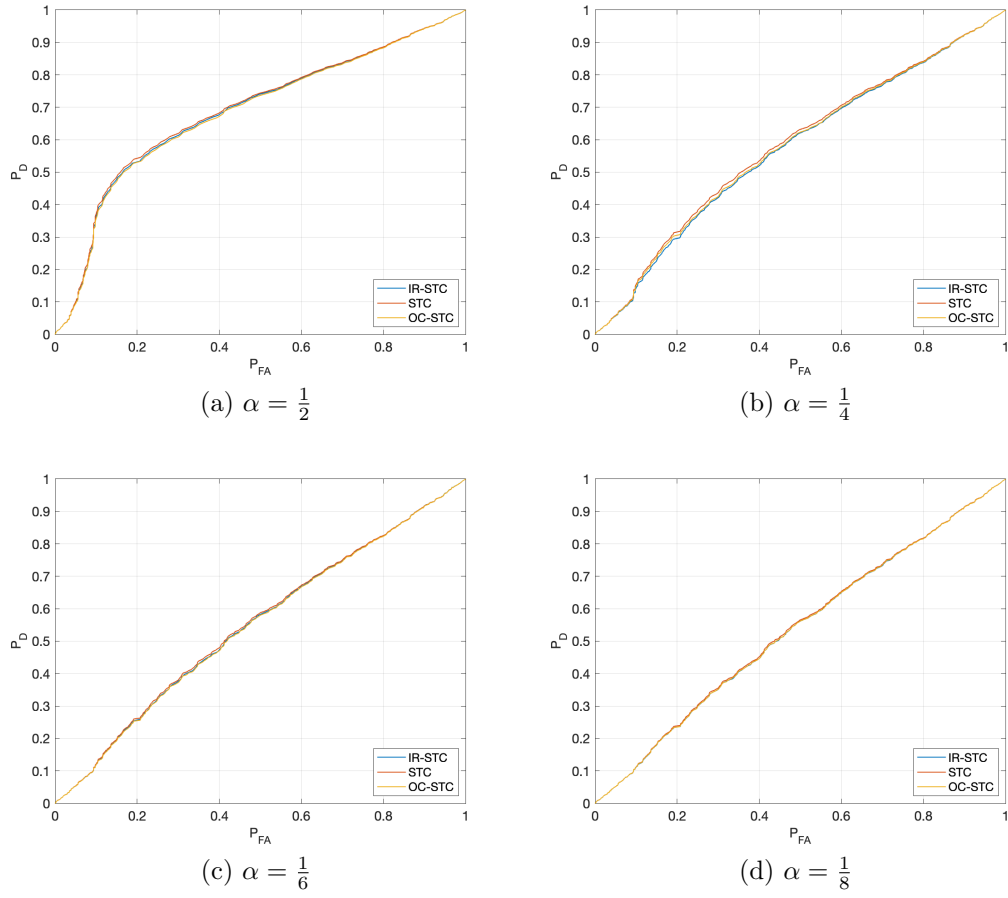


Figure 4.7: Comparison of ROC curves for sample pairs detector, for IR-STC, OC-STC, and STC, with $16 \times 16, n = 768$ favicons.

Chapter 5

Simulating Suboptimal Embedding

In this chapter, we look at how we can improve simulated embedding by being suboptimal. Researchers who wish to benchmark the detectability of steganographic distortion functions typically simulate stego objects. However, the difference (coding loss) between simulated stego objects, and real stego objects are both substantial and dependent on multiple factors. We first identify the factors affecting the coding loss, then through minimal sampling, we can accurately estimate the expected coding loss, allowing us to simulate suboptimal stego objects which are closer representations to real stego objects.

5.1 Statement of the Problem

There are two ways to test a steganographic algorithm that minimises an additive cost function: embed messages using a Syndrome Trellis Code (STC) [45] as described in Section 2.7.1 or simulate an optimal code that meets the rate-distortion bound [52, 36]. In a survey of several top venues for steganography and steganalysis research, we found significantly more work where researchers had simulated [38, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110], as opposed to creating real stego objects [40, 111, 112, 113, 114, 115, 116]. There are two reasons that may have motivated them. First, the STC algorithm is complex to implement and slow to run on extensive data sets. Second, separation of concerns suggests testing the cost function in isolation from the coding.

However, it is well known that the state-of-art STC method does *not* meet the rate-distortion bound. As we shall see, this coding loss is variable, depending on parameters of the STC and the cost function itself. When authors simulate embedding, instead of performing genuine coding, the simulations have less distortion than genuine stego objects. Furthermore, this may advantage one distortion function over another.

Genuine stego objects, on the other hand, can be created using the STC algorithm, which is based on the Viterbi algorithm [45]. The trellis has a parameter h , the *constraint height*, and its realised distortion approaches the rate-distortion bound asymptotically as $h \rightarrow \infty$; however, the time complexity is $O(2^h)$: in practice, $h \in (6, 10)$ provides an acceptable trade-off [45]. In this chapter we use the notation $STC^h(\mathbf{c}, \mathbf{m})$ for the distortion realised by

an STC of constraint height h encoding message $\mathbf{m}$ into cover with costs $\mathbf{c}$.

We might hope that the distortion realized by STCs is close to the bound, i.e.

$$STC^h(\mathbf{c}, \mathbf{m}) \approx PLS(\mathbf{c}, |\mathbf{m}|). \quad (5.1)$$

Indeed, for some $\mathbf{m}$ the STC distortion can even be lower (for example a cover which already happens to contain message $\mathbf{m}$). But we typically witness *coding loss*: in the case of PLS, the STC distortion exceeds the bound; in the case of DLS, the STC can only embed a message shorter than m . Whilst we commonly see steganographic coding loss expressed as the *entropy coding loss* l_h [45, 52, 40]

$$l_h = 1 - \frac{m}{DLS(\mathbf{x}, STC^h(\mathbf{x}, \mathbf{m}))}, \quad (5.2)$$

for the purpose of indicating loss on a distortion level, we also will use a slight variant, *distortion coding loss* l_d :

$$l_d = \frac{STC^h(\mathbf{x}, \mathbf{m})}{PLS(\mathbf{x}, m)} - 1. \quad (5.3)$$

5.1.1 Simulating at the Rate-Distortion Bound

It is often difficult to develop practical steganographic embedding methods that operate efficiently [52]. To speed up the process of researching and developing new state of the art embedding strategies, many steganography researchers are opting to simulate an embedding. That is, to create a stego

object with the embedding impact of that of a new embedding strategy, without creating a practical embedding method. Whilst this has no practical use to a steganographer (who seek to embed messages in covers), to a steganography researcher who wishes to analyse the potential of a new method, simulating an embedding allows them to do so, without the need to implement a practical embedding algorithm.

As covered in section 2.5, we solve the first part of our embedding problem (finding the probabilities of change). By solving for PLS or DLS with (2.23) & (2.24), or (2.26) & (2.27) respectively, and using the individual distortion cost $c_i \in \mathbb{R}$ as generated by the embedding strategy, we then modify each element $x_i \in \mathbf{x}$ with the probability as defined in (2.28). The researcher derives the probabilities of change, and then makes the changes with the prior derived probabilities to the input cover, simulating an embedding strategy without having to create the coding algorithm to perform the embedding.

As previously stated, most steganographers simulate rather than use a real code. Whilst the divergence (coding loss) between the rate-distortion-bound and real codes is well known, it is only approached as differing over the embedding rate α .

5.1.2 Coding Loss is not Equal

The existence of coding loss is well understood, and some literature has compared STC embedding with simulated embedding [117, 46, 57, 20, 51, 118]. However, most researchers continue to use simulated stego objects: as well as neglecting the coding loss that must occur in practice, this could

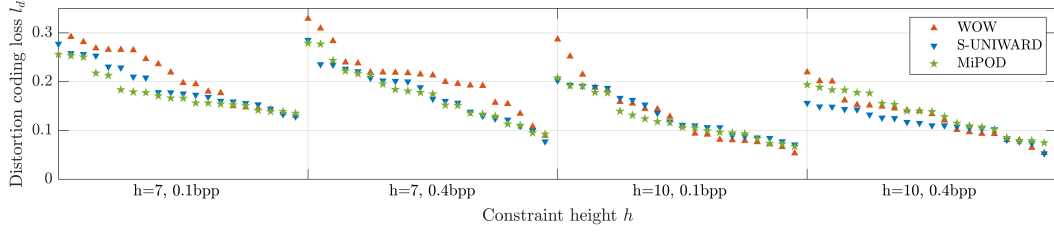


Figure 5.1: Comparison of per-cover distortion coding loss (l_d) for WOW, S-UNIWARD, and Mi-POD. Points are ordered from largest to smallest amount of coding loss along the x-axis for each of the four combinations of h and amount of payload.

advantage one distortion function more than another.

The coding loss of an embedding is dependant on the combination of distortion function, embedding rate, and constraint height. In Fig. 5.1 we show this discrepancy. Taking 100 covers from the BOSSBase [61] ($n = 262144$), we used STCs to encode 100 pseudo-randomly generated messages with payload rates $\alpha = \{0.1, 0.4\}$, using constraint heights $h \in \{7, 10\}$, for the distortion functions WOW [46], S-UNIWARD [20], and MiPOD [48].

We show that we can estimate the expected coding loss, and the adjusted embedding probabilities, through sampling the actual cost generated using STCs from a small cover subset. Doing so, we simulate stego objects which mirror real-world embeddings (corrected), and not the optimal (uncorrected).

5.2 Sampling Coding Loss

Our method aims to *increase* the entropy used in simulated embedding, to reflect the coding loss of STCs. We estimate the coding loss by embedding n_m random messages in each of n_c randomly-chosen covers.

For a given distortion function, constraint height h , and embedding rate

α , let the n_c covers be $\{\mathbf{x}_1, \dots, \mathbf{x}_{n_c}\}$. Rather than estimating l_h directly, we estimate $\tilde{l} = 1/(1 - l_h)$ as the average over covers and messages:

$$\tilde{l} = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{DLS(\mathbf{x}_i, \frac{1}{n_m} \sum_{j=1}^{n_m} STC^h(\mathbf{x}_i, \mathbf{m}_j))}{\alpha|\mathbf{x}_i|}, \quad (5.4)$$

for some n_m messages embedded into each cover $\mathbf{x}_i$ using an STC (with constraint height h), where $|\mathbf{m}_j| = \alpha|\mathbf{x}_i|$ (the *corrected entropy*). Having estimated $\tilde{l}$, we can use it to correct the entropy in simulated embedding, solving (2.23) & (2.24) for $m = \tilde{l}\alpha|\mathbf{x}_i|$ rather than $m = \alpha|\mathbf{x}_i|$. Our expectation is that

$$STC^h(\mathbf{c}, \mathbf{m}) \approx PLS(\mathbf{c}, \tilde{l}|\mathbf{m}|), \quad (5.5)$$

for any cover $\mathbf{c}$ from the same source, and any message $\mathbf{m}$. Hence, $\tilde{l}|\mathbf{m}|$ will be called the *corrected entropy*, and objects simulated using it will be termed *corrected stego objects*.

We can evaluate whether our method is successful by testing whether equation (5.5) holds more closely than (5.1). Because, over a set of images and messages, distortion will vary across orders of magnitude, average or RMS error is an inappropriate benchmark dominated by a few stego objects with large distortion. Instead, we use a Root Mean Square Relative Error

Table 5.1: Variance σ^2 of the corrected entropy $\tilde{l}_\alpha|\mathbf{x}_i|$, for values of n_m , and n_c , using S-UNIWARD, $h = 7$, $\alpha = 0.1$.

$n_m \setminus n_c$	1	2	4	8	16	32
1	45 270	21 439	7716	6655	2322	1851
2	44 543	22 329	10 653	5506	3213	1466
4	46 152	21 213	10 217	6256	2814	1451
8	34 695	25 433	11 884	5162	1891	1203
16	40 921	20 078	13 772	5796	2916	1419

(RMSRE),

$$RMSRE = \sqrt{\frac{1}{N} \sum_i \left(\frac{\tilde{d}_i - d_i}{\tilde{d}_i} \right)^2}, \quad (5.6)$$

where d_i is the true STC cost, $\tilde{d}_i$ is the simulated (uncorrected or corrected) cost, and N is the number of samples.

This will be measured across a set of covers, either the 10,000 standard steganography cover images BOSSBase [61] (all images the same size, $n = 262144$), or 10,000 images from the ALASKA [119] data set (image sizes vary from $n = 262144$ to $n = 1048576$).

5.2.1 Determining n_m and n_c

We first explore the parameters n_m and n_c . To determine which has more effect on the stability of the results, we measured the variance of $\tilde{l}_\alpha|\vec{c}|$ for $\alpha = 0.1$. We tested $n_m \in \{1, 2, 4, 8, 16\}$, and $n_c \in \{1, 2, 4, 8, 16, 32\}$ against an STC with $h = 7$, repeating 100 times the selection of covers from BOSSBase. In Table 5.1 we see that n_m has an almost negligible effect, so that we might

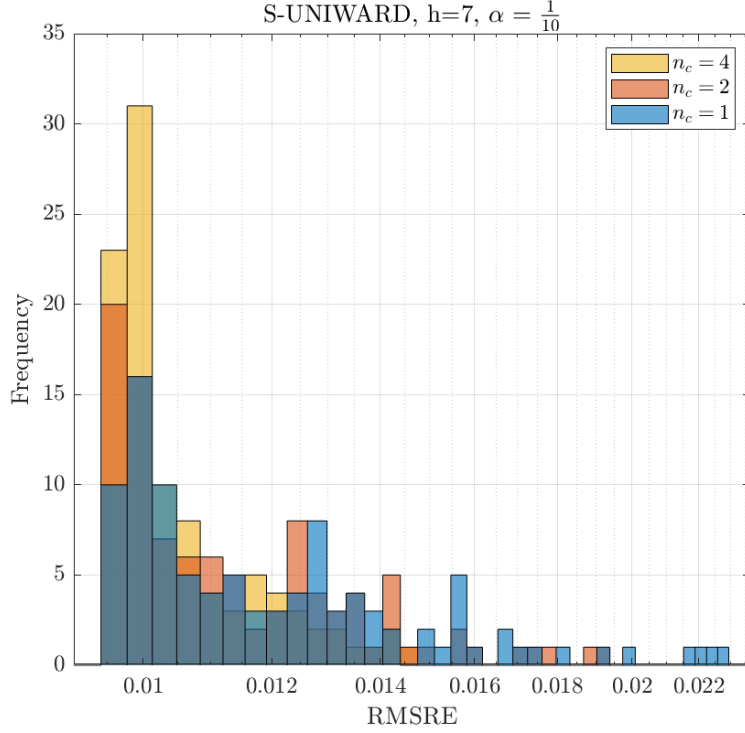


Figure 5.2: Distribution of RMSRE for different values of n_c over 100 simulations, using the S-UNIWARD distortion function, $h=7$.

as well set $n_m = 1$. On the other hand, the variance of the corrected entropy is approximately inversely proportional to n_c . This would be predicted by standard sampling theory [120, §7.3].

Having settled on $n_m = 1$, we can determine the lowest suitable value of n_c , as the user will want to perform the smallest sufficient number of STC embeddings. We expected that very small values of n_c would incorrectly estimate the entropy correction, or be subject to large deviations depending on the covers that were sampled.

Using S-UNIWARD, the BOSSBase cover-set, and an STC ($h = 7$), we found the average STC distortion (true cost) for 16 randomly generated mes-

Table 5.2: RMSRE of per cover distortion values (simulated versus true cost) for uncorrected, and corrected, S-UNIWARD, $h = 7$.

α	RMSRE $\times 10^{-2}$					
	Uncorrected	$n_c = 1$	$n_c = 2$	$n_c = 4$	$n_c = 8$	$n_c = 16$
0.1	16.89	1.36 (0.95, 2.53)	1.20 (0.95, 2.32)	1.06 (0.95, 1.48)	1.02 (0.95, 1.24)	1.00 (0.95, 1.20)
0.2	18.28	0.97 (0.69, 1.68)	0.83 (0.69, 1.64)	0.78 (0.69, 1.21)	0.73 (0.69, 0.91)	0.72 (0.69, 0.85)
0.3	14.25	0.55 (0.40, 0.98)	0.51 (0.40, 1.00)	0.44 (0.40, 0.70)	0.42 (0.40, 0.56)	0.41 (0.40, 0.46)
0.4	14.13	0.18 (0.09, 0.55)	0.14 (0.09, 0.32)	0.12 (0.09, 0.25)	0.11 (0.09, 0.21)	0.10 (0.09, 0.14)
0.5	7.06	0.42 (0.03, 0.81)	0.38 (0.03, 0.66)	0.34 (0.03, 0.50)	0.32 (0.03, 0.42)	0.32 (0.04, 0.36)

sages for each of the embedding rates $\alpha = \{0.1, 0.2, \dots, 0.5\}$. Using (2.23) & (2.24) we found the uncorrected simulated distortion for the original payload rates. For each value of $n_c \in \{1, 2, 4, 8, 16\}$ we found the cover-subset coding loss factor approximation using (5.4) for $n_m = 1$, then the per-cover corrected simulated distortion using (5.5), calculating the RMSRE for each combination of parameters. This experiment was performed 100 times for different random selections of n_c covers.

For values of $n_c \in \{1, 2, 4\}$ we show a histogram of RMSRE values over each of the 100 simulations in Fig. 5.2. We see that as n_c increases, the distribution of RMSRE tends towards 0 as expected; however, even with a value of $n_c = 1$, we see very small values. The variance in the distribution of RMSRE values is very low.

In Table 5.2 we show the RMSRE (above), minimum and maximum (below) RMSRE over all simulations for various values of n_c . We achieve a RMSRE that is at least an order of magnitude lower than uncorrected PLS

embedding. Higher values of n_c have only a small effect on the average outcome, but to our surprise, it is also the case that even $n_c = 1$ – a single sample of cover and message – did not cause large deviations in the accuracy of the simulated distortion. Even in the worst case (out of 100) selection of a single cover, in most cases, an order of magnitude reduction in RMSRE is observed.

5.3 Analysis

Since our method aims to improve the accuracy of simulating, we now benchmark our method in terms of how closely we can simulate real stego objects. We start with simulating individual covers and comparing distortion, simulating a whole cover sets distortion, and finally testing the impact on detectability.

5.3.1 Per Cover

Using the values $n_m = 1$ and $n_c = 1$, in Fig. 5.3 we show the per-cover-message: uncorrected, corrected (averaged over 100 simulations), and corrected (worst-case maximum RMRSE simulation) distortion values plotted against the true per-cover-message average distortion values using $h = 7$, and $h = 10$ respectively. It is clear that our method predicts accurate distortion values which negate the coding loss in the estimation. We emphasise that for each combination of distortion function, constraint height, and embedding rate we performed only a single true STC embedding to estimate the corrected entropy $\tilde{l}_\alpha|\tilde{c}|$, which was then used to simulate our adjusted stego

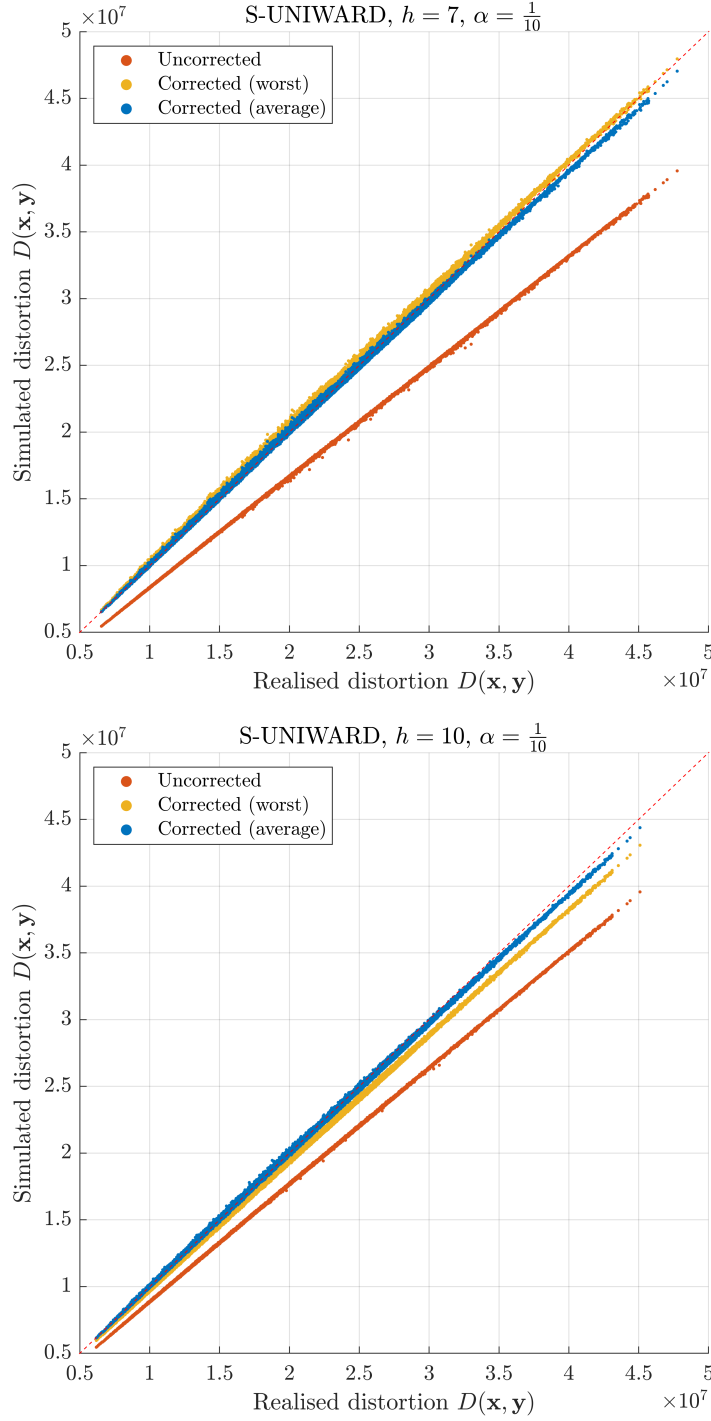


Figure 5.3: Corrected, and uncorrected simulated distortion versus realised distortion values. S-UNIWARD distortion function, STC $h \in \{7, 10\}$, $n_c = 1$, $n_m = 1$.

objects: a small overhead to find an accurate estimation of the true cost.

5.3.2 Distortion Functions

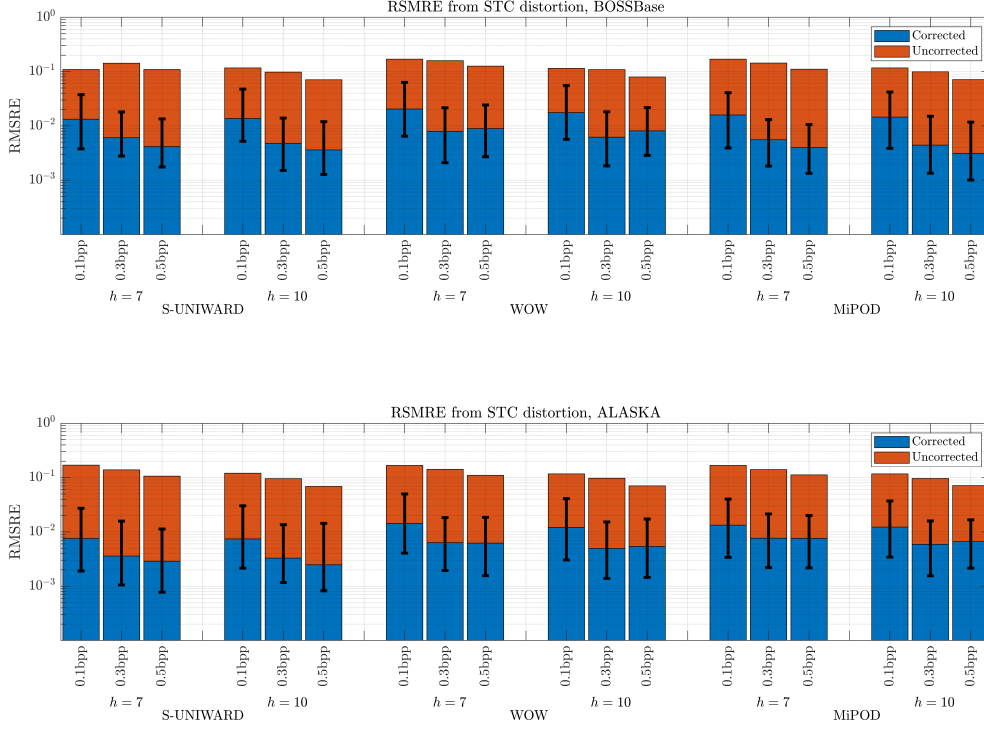


Figure 5.4: Comparison of uncorrected, and corrected RMSRE from achieved distortion, for BOSSBase (top) and ALASKA (bottom), $n_c = 1$, $n_m = 1$. Error bars indicate minimum and maximum RMSRE values obtained over all simulations.

To summarise the results, in Fig. 5.4 we compare our corrected distortion method to the uncorrected PLS method, using 10,000 covers from BOSS-Base, and again with 10,000 more heterogeneous images from ALASKA. We chose ALASKA as our second cover set to show that our method can be used on different sized covers. For a set of distortion functions, constraint heights, and embedding rates, we show the deviation from the true cost using

RMSRE. Our method reduces, typically by an order of magnitude or more, the relative error in distortion compared with a plain PLS method, across all combinations of distortion functions, constraint heights, embedding rates and cover sets.

5.3.3 Detectability

Table 5.3: Average detection error $\tilde{P}_E$, as well as standard deviation amongst folds, for corrected ($n_c = 1$, $n_m = 1$), and uncorrected simulated, compared to STC embedding, S-UNIWARD. All numbers are expressed as percentage points of error.

Method	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
Uncorrected PLS	39.82% (± 0.08)	23.34% (± 0.11)	12.49% (± 0.10)
STC $h = 7$	38.64% (± 0.07)	21.37% (± 0.06)	11.04% (± 0.09)
Corrected $h = 7$	38.55% (± 0.12)	21.30% (± 0.12)	11.03% (± 0.08)
STC $h = 10$	38.99% (± 0.08)	21.99% (± 0.08)	11.50% (± 0.13)
Corrected $h = 10$	38.96% (± 0.08)	21.85% (± 0.11)	11.62% (± 0.10)

So far we have only tested whether our correction method achieves the STC-realized *distortion*; now we test briefly whether it does the same for *detectability*. We benchmarked our corrected method, the uncorrected PLS method, and real STC stego objects, embedding in 10,000 BOSSBase covers at embedding rates $\alpha = \{0.1, 0.3, 0.5\}$. For STC embedding we used $h = \{7, 10\}$, and for the correction we used a single STC realization: $n_c = 1$ and $n_m = 1$. For the detector, we use the Spatial Rich Model with Single Quantization (*SRMQ1*) [91] feature sets with 12,753 dimensions, with the Low-complexity Linear Classifier (*LCLSMR*) [121], using half the cover and stego features for training, and the rest for testing and repeating under 10

splits of training and testing data. We measure detector error by the minimal total probability of error under equal Bayesian priors, P_E defined as:

$$P_E = \frac{P_{FA} + P_{MD}}{2}, \quad (5.7)$$

where P_{FA} is the false alarm rate, and P_{MD} the missed detection rate. The average value of 10 runs of the classifier denoted as $\tilde{P}_E$ is used to evaluate the various methods detectability.

In Table 5.3, we show the error rates, as well as the standard deviation amongst the ten training/testing splits. As expected, under the larger constraint height $h = 10$ the detectability of the STC is slightly closer to a pure PLS simulation, but a significant gap still remains: the simulation was under-distorting the stego objects compared with genuine STC-embedded stego objects. Although the absolute difference between error rates of 23.34% and 21.99% (taking, for example, the case $\alpha = 0.3$) does not seem large, it is statistically significant with $p < 10^{-6}$ if we treat the folds as a random sample. Moreover, it is not uncommon for detection differences of around 1% to be treated as significant when incrementally-better detectors are published [122].

However, when we use our proposed method to increase the PLS entropy to match the STC distortion, PLS embedding gives an error rate of 21.85% that is practically identical to that of real stego objects, and the difference is either not- or marginally-significant ($p > 0.05$ or $p > 0.01$, depending on α and h). Thus our method is simulating stego objects with detectability that matches that of STC-embedded objects, as intended.

5.4 Conclusion

We have shown that our method can create simulated stego objects which are closer in terms of distortion and detectability, to actual stego objects, than those created by uncorrected PLS embedding. Our method can efficiently sample the coding loss of a real stego object and derive an adjusted estimate of the entropy which must be introduced to the simulated stego object, to match the real world stego objects. The result is a low-cost sampling technique which significantly minimises the divergence between simulated and real-world stego objects, allowing the benefits in terms of speed and simplicity of simulating, with a closer representation to actual coding.

Our method also allows the researcher to simulate how their distortion function may behave under different conditions, not just embedding rates. Having used the state of the art STC algorithm, further work may establish if we can simulate different or new additive distortion coding algorithms.

Chapter 6

Conclusion

Since the wide adoption of the additive distortion minimisation framework, adaptive steganography has been compromised mainly of two elements: a user-defined distortion function which provides per cover value costs associated with change, and a code which realises these changes whilst embedding a message. Whilst the current emphasis in research has centred around building better distortion functions, this thesis has focused mainly on topics surrounding coding.

The motivation for this work has come from the belief that there are still improvements to be made in coding for adaptive steganography. Having approached three topics from this area, we have shown there is still improvements to be gained from the research area.

6.1 Contribution of this thesis

In this section, we will discuss and summarise the contributions of this thesis.

6.1.1 Robust Steganography

Chapter 3 focused on robust steganography, an area with little prior work that predated significant advances in adaptive steganography. Adaptive steganography uses distortion functions to discriminate poor changes (changes that would be more likely to render out stego object detectable to the adversary) from good changes, by assigning each change a cost. We then wish to minimise our distortion whilst encoding our message, as such, distortion is a proxy for detectability. Robust steganography is centred around the assumption that stego objects are usually very fragile, and are heavily reliant on the structure remaining unchanged, in order to convey the payload. Because of this, stego objects are susceptible to noise; robust steganography aims to create stego objects which can withstand noisy channels.

With the ever-rising popularity of digital media and social networking platforms, the area of robust steganography would allow for stego objects to be communicated via social networks or media sharing services. We consider the noise coincidental to the channel through JPEG recompression.

My work combined the advances in adaptive steganography with error-correcting codes to produce an embedding algorithm that both minimised distortion and introduced robustness into the stego object. The resulting algorithm, *Dual Syndrome Trellis Codes*, expanded on the state of the art

STC algorithm, to create stego objects able to withstand the amount of channel noise we expect to be introduced by JPEG recompression. The research resulted in the conference paper entitled *Adaptive Steganography in the Noisy Channel with Dual-Syndrome Trellis Codes*, which was presented at the 2018 IEEE International Workshop on Information Forensics and Security (WIFS) conference, in Hong Kong. The paper received very strong reviews and won the best student paper award.

6.1.2 Irregular Syndrome Trellis Codes

Our work on Irregular Syndrome Trellis Codes was motivated by the appearance of positive outliers, cover values which appear to be modified at a larger rate than dictated by optimal change probabilities. If we embed enough random messages into a cover, we would expect an embedding algorithm to make changes to a cover value with a known probability (the optimal change probability). However, since all embedding algorithms incur quantisation to reduced the search space for solutions, the optimal change probabilities are rarely met, causing deviations from these probabilities. The most significant deviations from the optimal change probabilities are referred to as the outliers.

These outliers could be an artefact our adversaries are able to leverage for detection; they also cause an increase to distortion since we are deviating from the optimum. The deviation from optimal change probabilities can be quantified as the coding loss incurred by the embedding algorithm; thus, we wish to minimise the coding loss.

Our work looked at modifying the construction of the parity check matrix used during embedding, to create an irregular-shaped trellis. Whereas a regular STC uses regular shaped blocks outside of the pruning incurred during the last few cover values, our method used variable-width blocks, formed by tiling the matrix in row form. This allowed us to maintain a constraint of each row in the matrix containing the same segment, tiled in an arbitrary manner. Although our method improved on the state of the art, the improvement was minimal. This work was presented at the Electronic Imaging - Media Watermarking, Security, and Forensics 2019 conference. The resulting paper is titled "*Reducing Coding Loss with Irregular Syndrome Trellis Codes*". During experiments, we found that the application of our method is more significant in smaller covers. Continuing this work, we looked at the detectability of Favicons, when combined with our method.

6.1.3 Simulating Suboptimal Steganographic Embedding

In Chapter 5, we looked at the possible drawbacks of simulating stego objects. The motivation for researchers to simulate is clear, it is time-consuming both to set up a real stego system and produce real stego objects. However, there exists a problem: simulating at the rate-distortion bound is an unfair method of analysis. We introduced a lightweight sampling method, which accurately predicts the coding loss as a function of the distortion function $\mathbf{D}$, constraint height of the STC h , and the embedding rate α . Furthermore, once we have an accurate prediction of the coding loss across some cover set, we can

simulate stego objects not at the rate-distortion bound, but closer to the real-world conditions. The result is stego objects which are indistinguishable from real stego objects, both in terms of distortion and detectability.

Our work allows researchers to simulate stego objects for a relatively small cost more accurately. Chapter 5's work resulted in the conference paper *Simulating Suboptimal Steganographic Embedding*, which was presented at the 2020 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSEC) conference, in Denver, CO. The paper received very strong reviews and won the best student paper award.

6.2 Further Work

Over the course of a DPhil, it is natural for new exciting areas of research to become apparent. However, due to the time restrictions in place, we cannot always explore these areas. In this section, we highlight further work, which we think can impact the field.

6.2.1 Further Investigation on the Construction of $\mathbb{H}$

In Chapter 4, we investigated the effect of the $\mathbb{H}$ matrix on Syndrome Trellis Code efficiency. Our method, Irregular Syndrome Trellis Codes showed that an irregular matrix $\mathbb{H}$ could slightly improve on regular matrices. However, an in-depth study of the $\mathbb{H}$ matrices has yet to be completed, and there may be more improvements to be found, once a further understanding of the good properties of the matrix is understood.

It is common, when designing convolutional codes, to exhaustively search

the space of possible matrices. Whilst this may apply to constant-rate convolutional codes, to Syndrome Trellis Codes, this becomes increasingly complex: one of the attractions of using an STC is that it can work with arbitrary sized covers and messages. Since the height of $\hat{\mathbb{H}}$ is variable (the constraint height), and the width is also variable (dictated by α), an exhaustive search would require finding many optimal sub-matrices. Instead, the research could be conducted in the area of understanding the properties of $\hat{\mathbb{H}}$, such as short cycles, or large branching factors.

6.2.2 Non-Binary Sub-Optimal Simulation

Whilst in Chapter 5, we primarily used binary STCs for our code, there exists no reason that over codes could not be used. In particular, the multi-layered STC and non-binary STC as introduced in Section 2.7.2 would be of interest.

With multi-layered STCs, since the coding loss is compounded by the extra constraints of embedding using Wet Paper Codes and an STC across two layers of the cover. Interestingly, if the sender was able to sample the expected coding-loss of the multi-layered STC accurately, one may be able to approach the problem of how much payload to split into each layer in a better manner.

For native non-binary STCs, researchers are even more reluctant to generate real stego objects, as the time complexity is far higher than binary and multi-layered STCs. Whilst we assume our method would be directly applicable without change to native non-binary STCs, it would be useful for researchers because of the same rationale as binary STCs.

6.2.3 Lazy evaluation of trellis for high q value embedding

Throughout this thesis there has been a heavy focus on the Syndrome Trellis Code algorithm. Whilst STCs are the state of the art, and it is unlikely for binary embedding that significant improvements can be found, for the non-binary embedding this is not the case. We as researchers have also accepted a constraint height $h \in \{6, 7, \dots, 12\}$ as a reasonable in terms of a distortion-time complexity trade-off. In fact, for the ternary embedding case ($q = 3$), it is almost certainly the case that a multi-layered STC would be used, rather than the native ternary arithmetic STC, because of time complexity.

The ability to embed using high values of q such as 5, 7, or 9 would allow adaptive steganography embedding to consider higher dimension changes. However, the time complexity constraints of the STC algorithm arguably is the limiting factor to researchers exploring this area.

Our preliminary work has identified large inefficiencies in the trellis construction. Mainly that an STC is not much more than a breadth-first search, exploring each possible encoding from within a quantised set; the non-zero h elements per column in $\mathbb{H}$ reduces the embedding operation to $O(q^h)$ operations. This inefficiency has already been highlighted, in the area of the STCs original algorithms basis; the Viterbi decoder, resulting in the Lazy-Viterbi decoder [123]. Using the Lazy-Viterbi decoder, paths through the trellis are expanded in the order of lowest likely hood deviation from the decoders certainty. In the analogous embedding case; paths in the trellis are expanded on the lowest distortion cost basis.

With higher q values, larger changes to the cover elements would likely result in larger distortion values. Because the original STC algorithm expands each path (each assignment of $x_i \rightarrow y_i$), most of these extra paths would likely be cancelled out by a lower cost incoming edge from a lower q value path. However, the search space would increase - there are q^h embedding states per cover element, as such, the STC will decrease distortion as q increases by virtue of expanding more states, but the number of low-cost states as a proportion of all states decreases as q increases; the trellis becomes less efficient.

This work would allow researchers to use values of q , and h previously considered impractical, and opens the path for researchers to see if detectability gains can be made in the higher q space.

To demonstrate this, we implemented a Lazy-Viterbi like embedding algorithm and compared it to a native STC. In Fig 6.1 and Fig 6.2 we show a native STC, and Lazy-Viterbi like algorithm. It is clear that the Lazy-Viterbi algorithm is more efficient, with less cancelled edges being produced, and fewer states explored, whilst still finding low-cost solutions to the embedding problem: of all the accepting states, an equally low distortion path was found for both the STC and Lazy-Viterbi algorithm. In total, the STC algorithm explored 167 states and 420 edges (of which 262 are inactive), compared to the Lazy-Viterbi's 119 states and 131 edges (of which 21 are inactive).

Whilst the above work is limited with a low constraint height ($h = 4$), it is our expectation that the STC performance degrades in $O(q^h)$ and will suffer from a combinatorial explosion in time complexity, whereas the Lazy-

Viterbi algorithm will be much more efficient. This would allow researchers to approach the area of high q embedding.

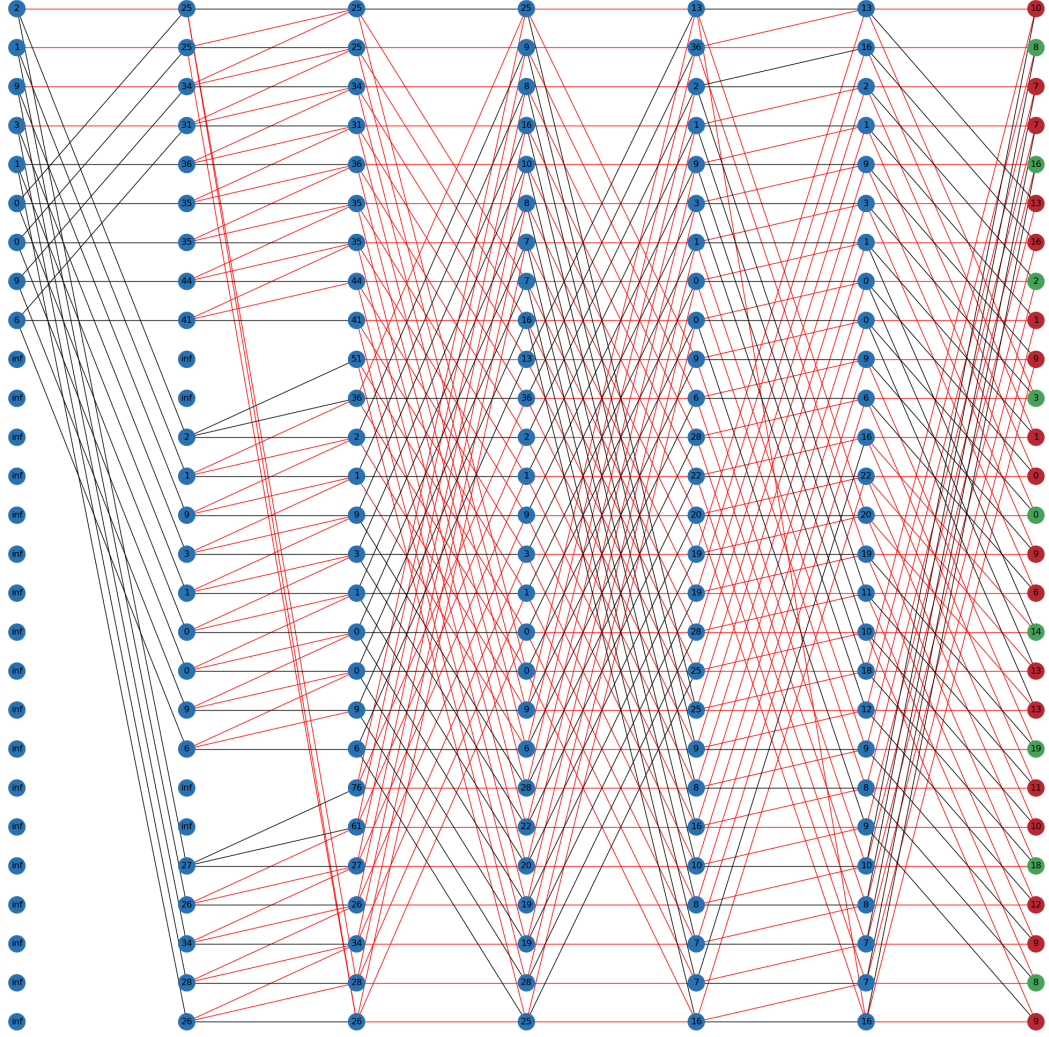


Figure 6.1: STC trellis, $h = 4$, $q = 3$. Red edges denote paths which are cancelled out by lower cost edges. Green and red nodes indicate accepting and non-accepting message states respectively.

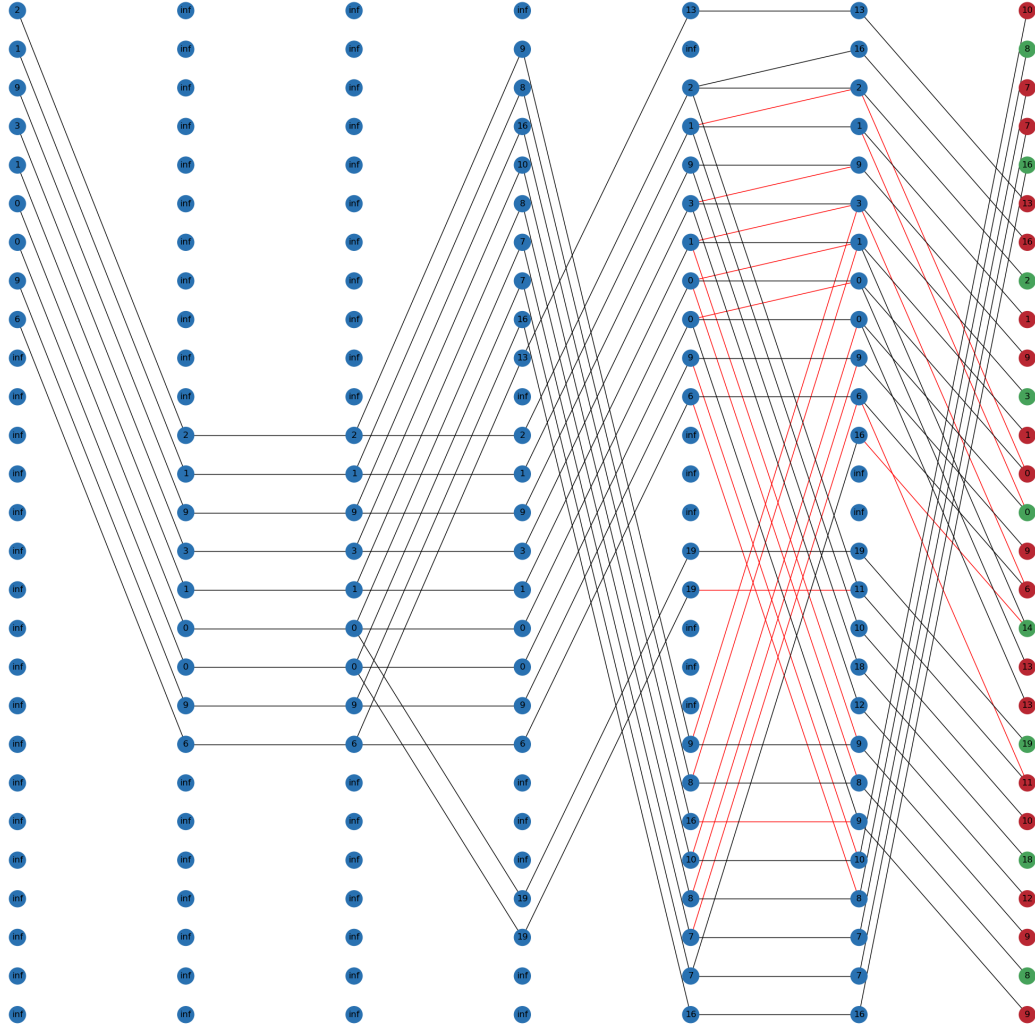


Figure 6.2: Lazy-viterbi trellis, $h = 4$, $q = 3$. Red edges denote paths which are cancelled out by lower cost edges. Green and red nodes indicate accepting and non-accepting message states respectively.

Bibliography

- [1] J. Fridrich. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.
- [2] E. Cole. *Hiding in Plain Sight: Steganography and the Art of Covert Communication*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2003.
- [3] J. Trithemius and W. Heidel. *Steganographia*. Apud Joh. Fridericum Rudigerum, 1721.
- [4] T. Ernst. Schwarzweiße magie: Der schlüssel zum dritten buch der steganographia des trithemius. *Daphnis*, 25(1), 1996.
- [5] J. Reeds. The Ciphers in Book III of Trithemius’s *Steganographia*. *AT & T Labs-Research, New Jersey*, 7932:26, 1998.
- [6] I. Cox, M. Miller, J. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2007.
- [7] D. Kahn. The History of Steganography. In *Information Hiding*, pages 1–5. Springer, 1996.
- [8] N.F. Johnson and S. Jajodia. Exploring Steganography: Seeing the Unseen. *Computer*, 31(2):26–34, 1998.
- [9] C. Fuchs. *Social Media: A Critical Introduction*. Sage, 2017.
- [10] S. Greenwood, A. Perrin, and M. Duggan. Social Media Update 2016. *Pew Research Center*, 11(2), 2016.
- [11] G. Kolata. Veiled Messages of Terror May Lurk in Cyberspace, 2001. Accessed on 16th January 2020.
- [12] N. Shachtman. FBI: Spies Hid Secret Messages on Public Websites, 2010. Accessed on 16th January 2020.

- [13] G. J. Simmons. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, Boston, MA, 1984.
- [14] R. Chandramouli, M. Kharrazi, and N. Memon. Image Steganography and Steganalysis: Concepts and Practice. In *International Workshop on Digital Watermarking*, pages 35–49. Springer, 2003.
- [15] A.D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. Pevný. Moving Steganography and Steganalysis From the laboratory into the Real World. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, pages 45–58. ACM, 2013.
- [16] C. Hundt, M. Liskiewicz, and U. Wolfel. Provably Secure Steganography and the Complexity of Sampling. *Algorithms and Computation*, pages 754–763, 2006.
- [17] T. Pevný and A.D. Ker. Steganographic Key Leakage Through Payload Metadata. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 109–114, 2014.
- [18] Y.Q. Shi, C. Chen, and W. Chen. A Markov Process Based Approach to Effective Attacking JPEG Steganography. In *International Workshop on Information Hiding*, pages 249–264. Springer, 2006.
- [19] N.F. Johnson and S. Jajodia. Exploring Steganography: Seeing the Unseen. *Computer*, 31(2), 1998.
- [20] V. Holub, J. Fridrich, and T. Denemark. Universal Distortion Function for Steganography in an Arbitrary Domain. *EURASIP Journal on Information Security*, 2014(1), 2014.
- [21] M. Grosvald and C. Orgun. Free from the Cover Text: A Human-generated Natural Language Approach to Text-based Steganography. *Journal of Information Hiding and Multimedia Signal Processing*, 2, 05 2011.
- [22] A. Wilson, P. Blunsom, and A.D. Ker. Linguistic Steganography on Twitter: Hierarchical Language Modeling with Manual Interaction. In *Media Watermarking, Security, and Forensics 2014*, volume 9028, page 902803. International Society for Optics and Photonics, 2014.
- [23] A. Wilson and A.D. Ker. Avoiding Detection on Twitter: Embedding Strategies for Linguistic Steganography. *Electronic Imaging*, 2016(8):1–9, 2016.

- [24] K. Wang, H. Zhao, and H. Wang. Video Steganalysis Against Motion Vector-Based Steganography by Adding or Subtracting One Motion Vector Value. *IEEE Transactions on Information Forensics and Security*, 9(5):741–751, May 2014.
- [25] Y. Cao, X. Zhao, D. Feng, and R. Sheng. Video Steganography with Perturbed Motion Estimation. In *International Workshop on Information Hiding*, pages 193–207. Springer, 2011.
- [26] H. Zhang, Y. Cao, X. Zhao, W. Zhang, and N. Yu. Video Steganography with Perturbed Macroblock Partition. In *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, pages 115–122. ACM, 2014.
- [27] Hideki N., Tomofumi F., Michiharu N., and Eiji K. Video Steganography Based on Bit-Plane Decomposition of Wavelet Transformed Video. In Edward J. Delp III and Ping W. Wong, editors, *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 345 – 353. International Society for Optics and Photonics, SPIE, 2004.
- [28] X. Yi, K. Yang, X. Zhao, Y. Wang, and H. Yu. AHCM: Adaptive Huffman Code Mapping for Audio Steganography Based on Psychoacoustic Model. *IEEE Transactions on Information Forensics and Security*, 14(8):2217–2231, Aug 2019.
- [29] G. Litao, A. Akansu, and M. Ramkumar. MP3 Resistant Oblivious Steganography. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 3, pages 1365–1368 vol.3, May 2001.
- [30] M. Atoum, S. Ibrahimn, G. Sulong, A. Zeki, and A. Abubakar. Exploring the Challenges of MP3 Audio Steganography. In *2013 International Conference on Advanced Computer Science Applications and Technologies*, pages 156–161, Dec 2013.
- [31] Y. F. Huang, S. Tang, and J. Yuan. Steganography in Inactive Frames of VoIP Streams Encoded by Source Codec. *IEEE Transactions on Information Forensics and Security*, 6(2):296–306, June 2011.
- [32] Y. Huang, C. Liu, C. Tang, and S. Bai. Steganography Integration Into a Low-Bit Rate Speech Codec. *IEEE Transactions on Information Forensics and Security*, 7(6):1865–1875, Dec 2012.

- [33] Z. Lin, Y. Huang, and J. Wang. RNN-SM: Fast Steganalysis of VoIP Streams Using Recurrent Neural Network. *IEEE Transactions on Information Forensics and Security*, 13(7):1854–1868, July 2018.
- [34] A.D. Ker. Resampling and the Detection of LSB Matching in Colour Bitmaps. In *Security, steganography, and watermarking of multimedia contents VII*, volume 5681, pages 1–15. International Society for Optics and Photonics, 2005.
- [35] J. Fridrich, P. Lisoněk, and D. Soukal. On Steganographic Embedding Efficiency. In *International Workshop on Information Hiding*, pages 282–296. Springer, 2006.
- [36] T. Filler and J. Fridrich. Design of Adaptive Steganographic Schemes for Digital Images. In *Media Watermarking, Security, and Forensics III*, volume 7880, page 78800F. International Society for Optics and Photonics, 2011.
- [37] R. Böhme. *An Epistemological Approach to Steganography*, page 15–30. Springer-Verlag, Berlin, Heidelberg, 2009.
- [38] T. Denemark and J. Fridrich. Side-Informed Steganography with Additive Distortion. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Nov 2015.
- [39] V. Holub and J. Fridrich. Designing Steganographic Distortion Using Directional Filters. In *In Fourth IEEE International Workshop on Information Forensics and Security*, 2012.
- [40] L. Guo, J. Ni, and Y.Q. Shi. Uniform Embedding for Efficient JPEG Steganography. *IEEE Transactions on Information Forensics and Security*, 9(5):814–825, May 2014.
- [41] Y. Kim, Z. Duric, and D. Richards. *Modified Matrix Encoding Technique for Minimal Distortion Steganography*, pages 314–327. Springer Berlin Heidelberg, 2007.
- [42] C. Wang and J. Ni. An Efficient JPEG Steganographic Scheme Based on the Block Entropy of DCT Coefficients. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1785–1788. IEEE, 2012.
- [43] B. Li, M. Wang, J. Huang, and X. Li. A New Cost Function for Spatial Image Steganography. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4206–4210. IEEE, 2014.

- [44] W. Tang, B. Li, W. Luo, and J. Huang. Clustering Steganographic Modification Directions for Color Components. *IEEE Signal Processing Letters*, 23(2):197–201, 2016.
- [45] T. Filler, J. Judas, and J. Fridrich. Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes. *IEEE Transactions on Information Forensics and Security*, 6(3):920–935, 2011.
- [46] V. Holub and J. Fridrich. Designing Steganographic Distortion using Directional Filters. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 234–239, Dec 2012.
- [47] Denemark T., Fridrich J., and Holub V. Further Study on the Security of S-UNIWARD. In Adnan M. Alattar, Nasir D. Memon, and Chad D. Heitzenrater, editors, *Media Watermarking, Security, and Forensics 2014*, volume 9028, pages 38 – 50. International Society for Optics and Photonics, SPIE, 2014.
- [48] V. Sedighi, R. Cogranne, and J. Fridrich. Content-adaptive Steganography by Minimizing Statistical Detectability. *IEEE Transactions on Information Forensics and Security*, 11(2):221–234, Feb 2016.
- [49] T. Filler and J. Fridrich. Steganography Using Gibbs Random Fields. In *Proceedings of the 12th ACM workshop on Multimedia and security*, pages 199–212, 2010.
- [50] J. Fridrich and J. Kodovský. Multivariate Gaussian Model for Designing Additive Distortion for Steganography. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2949–2953, May 2013.
- [51] T. Pevný and A.D. Ker. Exploring Non-Additive Distortion in Steganography. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec ’18, pages 109–114, New York, NY, USA, 2018. ACM.
- [52] T. Filler and J. Fridrich. Gibbs Construction in Steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):705–720, 2010.
- [53] R. Crandall. Some notes on steganography. Steganography Mailing List, available from <http://os.inf.tu-dresden.de/westfeld/crandall.pdf>, 1998.

- [54] A. Westfeld. F5-A Steganographic Algorithm. In *Proceedings of the 4th International Workshop on Information Hiding*, IHW '01, pages 289–302, London, UK, UK, 2001. Springer-Verlag.
- [55] A. Westfeld and A. Pfitzmann. Attacks on Steganographic Systems. In Andreas Pfitzmann, editor, *Information Hiding*, pages 61–76, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [56] J. Fridrich, M. Goljan, and D. Hoge. Steganalysis of JPEG Images: Breaking the F5 Algorithm. In *Revised Papers from the 5th International Workshop on Information Hiding*, IH '02, pages 310–323, London, UK, UK, 2003. Springer-Verlag.
- [57] T. Pevný, T. Filler, and P. Bas. Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. In *Information Hiding*, pages 161–177, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [58] J. Fridrich, M. Goljan, and D. Soukal. Wet Paper Codes With Improved Embedding Efficiency. *IEEE Transactions on Information Forensics and Security*, 1(1):102–110, March 2006.
- [59] T. Filler, J. Judas, and J. Fridrich. Minimizing Embedding Impact in Steganography Using Trellis-Coded Quantization. In *Media Forensics and Security*, page 754105, 2010.
- [60] G.D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
- [61] P. Bas, T. Filler, and T. Pevný. ‘Break Our Steganographic System’: The Ins and Outs of Organizing BOSS. In *Information Hiding*, pages 59–70, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [62] T. Filler and J. Fridrich. Minimizing Additive Distortion Functions With Non-Binary Embedding Operation in Steganography. In *2010 IEEE International Workshop on Information Forensics and Security*, pages 1–6, Dec 2010.
- [63] J. Fridrich, M. Goljan, and D. Soukal. Wet Paper Codes With Improved Embedding Efficiency. In *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, page 607215. International Society for Optics and Photonics, 2006.
- [64] C. Shannon. A Mathematical Theory of Communication. *Bell system technical journal*, 27(3):379–423, 1948.

- [65] W. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge university press, 2010.
- [66] J. Cover, T. and Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.
- [67] R.W. Hamming. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.
- [68] S. Wicker and V. Bhargava. *Reed-Solomon Codes and Their Applications*. John Wiley & Sons, 1999.
- [69] R. Johannesson and K.S. Zigangirov. *Fundamentals of Convolutional Coding*, volume 15. John Wiley & Sons, 2015.
- [70] L. Marvel, C. Boncelet, and C. Retter. Spread Spectrum Image Steganography. *IEEE Transactions on image processing*, 8(8):1075–1083, 1999.
- [71] K. Solanki, A. Sarkar, and B.S. Manjunath. Yass: Yet Another Steganographic Scheme That Resists Blind Steganalysis. In *International Workshop on Information Hiding*, pages 16–31. Springer, 2007.
- [72] X. Zhang and S. Wang. Stego-encoding with Error Correction Capability. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(12):3663–3667, 2005.
- [73] K. Solanki, N. Jacobsen, U. Madhow, B.S. Manjunath, and S. Chandrasekaran. Robust Image-Adaptive Data Hiding Using Erasure and Error Correction. *IEEE Transactions on image processing*, 13(12):1627–1639, 2004.
- [74] N. Hopper, L. von Ahn, and J. Langford. Provably Secure Steganography. *IEEE Transactions on Computers*, 58(5):662–676, 2009.
- [75] A. Sarkar, U. Madhow, and B.S. Manjunath. Matrix Embedding With Pseudorandom Coefficient Selection and Error Correction for Robust and Secure Steganography. *IEEE Transactions on Information Forensics and Security*, 5(2):225–239, 2010.
- [76] V. Holub and J. Fridrich. Challenging the Doctrines of JPEG Steganography. In *Media Watermarking, Security, and Forensics*, 2014.

- [77] J. Kodovský and J. Fridrich. Steganalysis of JPEG Images Using Rich Models. In *Media Watermarking, Security, and Forensics 2012*, volume 8303, page 83030A. International Society for Optics and Photonics, 2012.
- [78] O. Köhler, C. Pasquini, and R. Böhme. On the Statistical Properties of Syndrome Trellis Coding. In *International Workshop on Digital Watermarking*, pages 331–346. Springer, 2017.
- [79] W. Liu, G. Liu, and Y. Dai. Syndrome Trellis Codes Based on Minimal Span Generator Matrix. *Annals of telecommunications-Annales des télécommunications*, 69(7-8):403–416, 2014.
- [80] J. Fridrich and D. Soukal. Matrix Embedding for Large Payloads. *IEEE Transactions on Information Forensics and Security*, 1(3):390–395, 2006.
- [81] J. Fridrich, M. Goljan, and D. Soukal. Steganography via Codes for Memory with Defective Cells. In *43rd Conference on Coding, Communication, and Control*, pages 28–30, 2005.
- [82] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering Codes*. Elsevier, 1997.
- [83] L. Wasserman. *All of Statistics : A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer Science & Business Media, New York, NY, 2013 edition, 2004.
- [84] J. Eggers, R. Bäuml, and B. Girod. A Communications Approach to Image Steganography. In *Proceedings of SPIE: Electronic Imaging 2002, Security and Watermarking of Multimedia Contents IV*, pages 26–37, 2002.
- [85] T. Pevný, M. Kopp, J. Křoustek, and A.D. Ker. Malicons: Detecting Payload in Favicons. *Electronic Imaging*, 2016(8):1–9, 2016.
- [86] A.D. Ker. A General Framework for Structural Steganalysis of LSB Replacement. In *International Workshop on Information Hiding*, pages 296–311. Springer, 2005.
- [87] W. Tang, H. Li, W. Luo, and J. Huang. Adaptive Steganalysis Based on Embedding Probabilities of Pixels. *IEEE Transactions on Information Forensics and Security*, 11(4):734–745, April 2016.

- [88] W. Zhou, W. Zhang, and N. Yu. A New Rule for Cost Reassignment in Adaptive Steganography. *IEEE Transactions on Information Forensics and Security*, 12(11):2654–2667, Nov 2017.
- [89] M. Goljan, J. Fridrich, and R. Cogranne. Rich Model for Steganalysis of Color Images. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 185–190, Dec 2014.
- [90] B. Li, S. Tan, M. Wang, and J. Huang. Investigation on Cost Assignment in Spatial Image Steganography. *IEEE Transactions on Information Forensics and Security*, 9(8):1264–1277, Aug 2014.
- [91] J. Fridrich and J. Kodovsky. Rich Models for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, June 2012.
- [92] V. Holub and J. Fridrich. Random Projections of Residuals for Digital Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, 8(12):1996–2006, Dec 2013.
- [93] W. Tang, B. Li, S. Tan, M. Barni, and J. Huang. Cnn-based Adversarial Embedding for Image Steganography. *IEEE Transactions on Information Forensics and Security*, 14(8):2074–2087, Aug 2019.
- [94] K. Chen, H. Zhou, W. Zhou, W. Zhang, and N. Yu. Defining Cost Functions for Adaptive JPEG Steganography at the Microscale. *IEEE Transactions on Information Forensics and Security*, 14(4):1052–1066, April 2019.
- [95] D. Hu, H. Xu, Z. Ma, S. Zheng, and B. Li. A Spatial Image Steganography Method Based on Nonnegative Matrix Factorization. *IEEE Signal Processing Letters*, 25(9):1364–1368, Sep. 2018.
- [96] J. Ye, J. Ni, and Y. Yi. Deep Learning Hierarchical Representations for Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, Nov 2017.
- [97] K. Chen, W. Zhang, H. Zhou, N. Yu, and G. Feng. Defining Cost Functions for Adaptive JPEG Steganography at the Microscale. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Dec 2016.
- [98] X. Hu, J. Ni, and Y. Shi. Efficient JPEG Steganography Using Domain Transformation of Embedding Entropy. *IEEE Signal Processing Letters*, 25(6):773–777, June 2018.

- [99] P. Bas. Steganography via Cover-Source Switching. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Dec 2016.
- [100] J. Kodovsky and J. Fridrich. Effect of Image Downsampling on Steganographic Security. *IEEE Transactions on Information Forensics and Security*, 9(5):752–762, May 2014.
- [101] W. Tang, S. Tan, B. Li, and J. Huang. Automatic Steganographic Distortion Learning Using a Generative Adversarial Network. *IEEE Signal Processing Letters*, 24(10):1547–1551, Oct 2017.
- [102] W. Tang, B. Li, W. Luo, and J. Huang. Clustering Steganographic Modification Directions for Color Components. *IEEE Signal Processing Letters*, 23(2):197–201, Feb 2016.
- [103] J. Fridrich. Effect of Cover Quantization on Steganographic Fisher Information. *IEEE Transactions on Information Forensics and Security*, 8(2):361–373, Feb 2013.
- [104] Z. Wang, X. Zhang, and Z. Yin. Joint Cover-Selection and Payload-Allocation by Steganographic Distortion Optimization. *IEEE Signal Processing Letters*, 25(10):1530–1534, Oct 2018.
- [105] S. Islam, M. R. Modi, and P. Gupta. Edge-Based Image Steganography. *EURASIP Journal on Information Security*, 2014(1):8, Apr 2014.
- [106] V. Sedighi and J. Fridrich. Effect of Imprecise Knowledge of the Selection Channel on Steganalysis. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec ’15*, pages 33–42, New York, NY, USA, 2015. ACM.
- [107] B. Chen, W. Luo, and P. Zheng. Enhancing Steganography via Stego Post-Processing by Reducing Image Residual Difference. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec’19*, pages 63–68, New York, NY, USA, 2019. ACM.
- [108] J. Yang, D. Ruan, X. Kang, and Y. Shi. Towards Automatic Embedding Cost Learning for JPEG Steganography. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec’19*, pages 37–46, New York, NY, USA, 2019. ACM.

- [109] W. Li, W. Zhang, K. Chen, W. Zhou, and N. Yu. Defining Joint Distortion for JPEG Steganography. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec '18*, pages 5–16, New York, NY, USA, 2018. ACM.
- [110] T. Taburet, P. Bas, J. Fridrich, and W. Sawaya. Computing Dependencies Between DCT Coefficients for Natural Steganography in JPEG Domain. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'19*, pages 57–62, New York, NY, USA, 2019. ACM.
- [111] L. Guo, J. Ni, W. Su, C. Tang, and Y. Shi. Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited. *IEEE Transactions on Information Forensics and Security*, 10(12):2669–2680, Dec 2015.
- [112] T. Denemark and J. Fridrich. Steganography With Multiple JPEG Images of the Same Scene. *IEEE Transactions on Information Forensics and Security*, 12(10):2308–2319, Oct 2017.
- [113] W. Zhang, J. Liu, X. Wang, and N. Yu. Generalization and Analysis of the Paper Folding Method for Steganography. *IEEE Transactions on Information Forensics and Security*, 5(4):694–704, Dec 2010.
- [114] B. Feng, W. Lu, and W. Sun. Secure Binary Image Steganography Based on Minimizing the Distortion on the Texture. *IEEE Transactions on Information Forensics and Security*, 10(2):243–255, Feb 2015.
- [115] Z. Zhao, Q. Guan, H. Zhang, and X. Zhao. Improving the Robustness of Adaptive Steganographic Algorithms Based on Transport Channel Matching. *IEEE Transactions on Information Forensics and Security*, 14(7):1843–1856, July 2019.
- [116] L. Guo, J. Ni, and Y.Q. Shi. An Efficient JPEG Steganographic Scheme using Uniform Embedding. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 169–174, Dec 2012.
- [117] B. Li, X. Wang, M. and Li, S. Tan, and J. Huang. A Strategy of Clustering Modification Directions in Spatial Image Steganography. *IEEE Transactions on Information Forensics and Security*, 10(9):1905–1917, Sep. 2015.

- [118] Z. Zhao, Q. Guan, and X. Zhao. Constructing Near-Optimal Double-Layered Syndrome-Trellis Codes for Spatial Steganography. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '16, pages 139–148, New York, NY, USA, 2016. ACM.
- [119] R. Cogranne, Q. Giboulot, and P. Bas. The ALASKA Steganalysis Challenge: A First Step Towards Steganalysis. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'19, pages 125–137, New York, NY, USA, 2019. ACM.
- [120] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, Belmont, Calif, 2nd ed. edition, 1995.
- [121] R. Cogranne, V. Sedighi, J. Fridrich, and T. Pevný. Is Ensemble Classifier Needed for Steganalysis in High-Dimensional Feature Spaces? In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Nov 2015.
- [122] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich. JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '17, page 75–84, New York, NY, USA, 2017. Association for Computing Machinery.
- [123] J. Feldman, I. Abou-Faycal, and M. Frigo. A Fast Maximum-Likelihood Decoder for Convolutional Codes. In *Proceedings IEEE 56th Vehicular Technology Conference*, volume 1, pages 371–375 vol.1, 2002.

Appendices

Appendix A

Syndrome Coding

Consider the following binary matrix $\mathbb{H}$, message $\mathbf{m}$, and input cover $\mathbf{x}$:

$$\mathbb{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{m} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

The syndrome $\mathbb{H}\mathbf{x}$ evaluates as:

$$\mathbb{H}\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

As $\mathbb{H}\mathbf{x} \neq \mathbf{m}$, we flip the cover value i , which has a subsequent i^{th} column in $\mathbb{H}$ that equals $\mathbb{H}\mathbf{x} - \mathbf{m}$:

$$\mathbb{H}\mathbf{x} - \mathbf{m} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

We flip $\mathbf{x}_2$ to create the stego vector $\mathbf{y}$, of which the syndrome $\mathbb{H}\mathbf{y} = \mathbf{m}$:

$$= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad \mathbb{H}\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Appendix B

(7,4) Hamming Codes

The (7,4) Hamming Code takes $k = 4$ information bits, which results in the code word of length $n = 7$:

$$\{c_1, c_2, c_3, c_4\} \rightarrow \{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}, \quad (\text{B.1})$$

such that the following constraints are met:

$$c_1 \oplus c_2 \oplus c_3 \oplus c_5 = 0 \quad (\text{B.2})$$

$$c_1 \oplus c_3 \oplus c_4 \oplus c_6 = 0 \quad (\text{B.3})$$

$$c_1 \oplus c_2 \oplus c_4 \oplus c_7 = 0. \quad (\text{B.4})$$

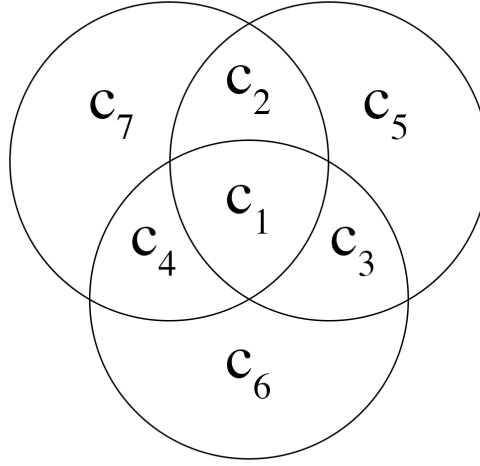


Figure B.1: Representation of the (7,4) Hamming Code parity.

In Fig. B.1 we show the above parity check constraints. If we took the assignment:

$$c_1 = 1, c_2 = 1, c_3 = 0, c_4 = 1, \quad (\text{B.5})$$

then the parity bits would be:

$$c_5 = 0, c_6 = 0, c_7 = 1. \quad (\text{B.6})$$

As such we transmit the code word:

$$1\ 1\ 0\ 1\ 0\ 0\ 1,$$

if a single error occurs, and the resulting code word is received:

$$1\ \mathbf{0}\ 0\ 1\ 0\ 0\ 1,$$

then following the constraints, we can identify c_2 is invalid, and can correct so that the value is flipped back to 1.

More generally, we can express the constraints in B.2 with a parity check matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{B.7})$$

doing so, if we set the values of c_5 , c_6 , and c_7 to 0, we can use the method as outlined in Section A to fix the syndrome required (000), making the required modifications to the code word.