

# A Novel Approach to Digital Watermarking, Exploiting Colour Spaces

By Frédéric Lusson

June 11, 2011

A thesis presented for the award of Master of Computer Science (Research) at Letterkenny Institute of Technology, Donegal, Ireland

June 2011

Submitted to the Higher Education and Training Awards Council (HETAC)

## Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of a Master of Computer Science (Research), is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of Candidate \_\_\_\_\_ Date \_\_\_\_\_

I hereby certify that all the unreferenced work described in this thesis and submitted for the award of Master of Computer Science (Research), is entirely the work of \_\_\_\_\_. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Signature of Supervisor \_\_\_\_\_ Date \_\_\_\_\_

Signature of Supervisor \_\_\_\_\_ Date \_\_\_\_\_

## Acknowledgments

I would like to acknowledge the many people who contributed to the completion of this masters. Firstly I would like to show my gratitude to my supervisors, Karen Bailey and Mark Leeney, for their invaluable help, guidance and patience. They were instrumental in providing their knowledge in the areas of steganography and image processing.

None of this would have been possible without the support of my family, in particular my wife and my two daughters. I am so grateful for your understanding and for giving me the time and space necessary to complete this thesis, while keeping the family cocoon together. Thank you.

I would also like to thank the founder of the Strand 1 Research Program - Council of Directors IOTI for their support in carrying out this work.

Finally, I wish to extend my appreciation to Damien McKeever, Betty and Kevin Quinlan and Paul Early, for helping me in producing a readable document in English and to Maeve Diver and Isabel Stephenson for their help in sourcing the referenced articles and papers.

# Abstract

Watermarking is the process of embedding information in a carrier in order to protect the ownership of text, music, video and images, while steganography is the art of hiding information.

Normally watermarks are embedded in images but remain visible in the majority of commercial image databases, such as Getty ([gettyimages.ie](http://gettyimages.ie)) or iStock Photo ([istockphoto.com](http://istockphoto.com)). Watermarked images display ownership information in the form of copyright notices super-imposed on the image itself. However this leaves traditional watermarking techniques vulnerable to tampering. Thus the advantage of using steganographic techniques for watermarking is that the watermark is resistant to detection and consequently to tampering.

Robustness is a characteristic of critical importance, in order that a watermark is to survive image manipulation and enhancement processes, as well as intentional attacks, to ensure piracy is prevented.

A review of digital image-based steganography and watermarking techniques is carried out in this document. This investigation reveals that most watermarking algorithms demonstrate partial resistance to attacks.

The aim of this work is to produce a novel hybrid digital watermarking technique, based on the exploitation of both the RGB and the YCbCr colour spaces, using spatial domain techniques. A text watermark is embedded in the YCbCr colour space, while an image watermark is embedded in the RGB colour space. Results demonstrate that the proposed hybrid technique can withstand levels of geometric attacks and processing attacks up to a point where the commercial value of the images tested would be lost. Results also demonstrate technical and performance improvements over existing methods, in terms of security and algorithm efficiency, while taking inspiration from steganography, to avoid drawing attention to the fact that an image contains hidden information.

## Abbreviations

GIF	Graphics Interchange Format
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics
BMP	Bitmap
HVS	Human Visual System
LSB	Least Significant Bit
DCT	Discrete Cosin Transform
FT	Fourier Transform
DFT	Discrete Fourier Transform
STFT	Short Time Fourier Transform
WT	Wavelet Transform
DWT	Discrete Wavelet Transform
QT	Quantization Table
PSNR	Peak Signal to Noise Ratio
NMSE	Normalized Mean Squared Error
SSIM	Structural Similarity measuring the similarity between two images.
LPM	Log Polar Mapping
CIE	French acronym for Commission Internationale de l'Eclairage.

JND	Just Noticeable Difference.
CRT	Cathode Ray Tube, describing the technology inside an analog computer monitor or television set.
PAL	Phase Alternation Line
SECAM	Sequentiel Couleur Avec Memoire (Sequential Colour with Memory)
ASCII	American Standard Code for Information Interchange
PoVs	Pair of Values
GNU-GPL	GNU General Public Licence is a widely used free software license, originally written by Richard Stallman for the GNU project
GUI	Graphical User Interface

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Organisation . . . . .	2
<b>2</b>	<b>DIGITAL IMAGE WATERMARKING AND STEGANOGRAPHY</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Watermarking . . . . .	5
2.3	Steganography . . . . .	7
2.4	Conclusion . . . . .	10
<b>3</b>	<b>DIGITAL IMAGE ATTACKS</b>	<b>12</b>
3.1	Introduction . . . . .	12
3.2	Image processing attacks . . . . .	12
3.3	Geometric transformation . . . . .	13
3.4	Cryptographic attack . . . . .	14
3.5	Protocol attack . . . . .	14
3.6	Benchmarking tools . . . . .	15
3.6.1	Stirmark . . . . .	15
3.6.2	Optimark . . . . .	15
3.6.3	Certimark . . . . .	16
3.6.4	Checkmark . . . . .	16
3.7	Conclusion . . . . .	16
<b>4</b>	<b>DATA HIDING AND STEGANALYSIS</b>	<b>17</b>
4.1	Introduction . . . . .	17
4.2	Spatial Domain Methods . . . . .	19
4.2.1	Least Significant Bit Substitution . . . . .	20
4.2.2	Additive Method . . . . .	21
4.2.3	Histogram . . . . .	22

4.2.4	Remarks . . . . .	23
4.3	Frequency Domain . . . . .	24
4.3.1	Signal Transformation . . . . .	24
4.3.2	Discrete Cosine Transform . . . . .	27
4.3.3	Discrete Fourier Transform . . . . .	28
4.3.4	Discrete Wavelet Transform . . . . .	30
4.3.5	Remarks . . . . .	37
4.4	Adaptive watermarking . . . . .	38
4.4.1	Remarks . . . . .	41
4.5	Steganalysis . . . . .	43
4.5.1	Visual observation . . . . .	44
4.5.2	Colour Palette Observation . . . . .	45
4.5.3	Chi Square / Pair of Values (PoVs) Observation . . . . .	46
4.5.4	Regular Singular (RS) Steganalysis . . . . .	46
4.5.5	DCT domain Steganalysis . . . . .	47
4.5.6	Remarks . . . . .	47
4.6	Conclusion . . . . .	48
<b>5</b>	<b>COLOUR SPACES AND WATERMARKING</b>	<b>51</b>
5.1	Introduction . . . . .	51
5.2	Colour Spaces in the context of watermarking . . . . .	51
5.3	RGB & YCbCr Colour Spaces . . . . .	54
5.3.1	RGB Colour Space . . . . .	54
5.3.2	YUV, YIQ and YCbCr Colour Spaces . . . . .	56
5.4	Conclusion . . . . .	58
<b>6</b>	<b>IMPLEMENTATION</b>	<b>60</b>
6.1	Introduction . . . . .	60
6.2	Methodology . . . . .	61
6.2.1	Motivations . . . . .	61
6.2.2	Embedding Phase . . . . .	66



6.2.3	Extraction Phase . . . . .	71
6.3	Image Attacks . . . . .	76
6.4	Measure of Invisibility . . . . .	77
6.5	Measure of Robustness . . . . .	79
6.6	Steganalysis . . . . .	80
6.7	Conclusion . . . . .	80
<b>7</b>	<b>RESULTS AND ANALYSIS</b>	<b>81</b>
7.1	Introduction . . . . .	81
7.2	Image Database . . . . .	81
7.3	Invisibility Analysis . . . . .	83
7.4	Robustness Analysis . . . . .	86
7.4.1	JPEG Compression Attack . . . . .	86
7.4.2	JPEG 2000 Compression Attack . . . . .	89
7.4.3	Noise Addition Attack . . . . .	91
7.4.4	Resizing Attacks . . . . .	93
7.4.5	Rotation Attacks . . . . .	95
7.4.6	Filtering and Histogram Attacks . . . . .	97
7.4.7	Median Filter and Self Similarity Attacks . . . . .	100
7.4.8	Cropping Attacks . . . . .	101
7.4.9	Collage Attacks . . . . .	103
7.4.10	Clipping Attacks . . . . .	104
7.4.11	Remarks . . . . .	105
7.5	Security Analysis . . . . .	105
7.6	Capacity Analysis . . . . .	110
7.7	Complexity Analysis . . . . .	111
7.8	Conclusion . . . . .	112
<b>8</b>	<b>CONCLUSION - RECOMMENDATIONS</b>	<b>114</b>
8.1	Overall Conclusion . . . . .	114
8.2	Recommendations for future work . . . . .	115

## List of Figures

1	The different embodiment disciplines of information hiding. (Adapted from Cheddad [9], the blue path indicates the goal of this study) . . . . .	3
2	A general digital image watermarking system . . . . .	7
3	Pixels bit substitution . . . . .	20
4	Additive watermarking in the spatial domain . . . . .	21
5	Frequency comparison . . . . .	25
6	Example of embedding a 3 bit watermark in the frequency domain of an image using DCT . . . . .	26
7	Subband Coding Algorithm . . . . .	33
8	DWT Frequency decomposition . . . . .	34
9	Two level DWT using a grey-scaled “lena” Image and the Daubechies filter [34] . . . . .	36
10	First level DWT decomposition in RGB colour channels . . . . .	52
11	RGB Colour Space [34] . . . . .	55
12	YCbCr Colour Space . . . . .	57
13	Original and watermarked “lena” image compared . . . . .	62
14	Original and extracted watermark . . . . .	63
15	Extracted watermark after JPEG compression . . . . .	63
16	Watermark Embedding in the RGB channels . . . . .	67
17	ASCII watermark embedding in the CbCr channels . . . . .	70
18	Watermark extraction from the CbCr channels . . . . .	72
19	Watermark extraction from the RGB channels . . . . .	75
20	Host images and watermark image . . . . .	82
21	MATLAB GUI comparing the original “lena” image and “lena” after the proposed hybrid watermarking method is performed . . . . .	84
22	NMSE and Correlation values of watermark at various JPEG quality factors . . . . .	89

23	NMSE and Correlation values of watermark at various JPEG 2000 quality factors . . . . .	91
24	NMSE and Correlation values of watermark at various noise addition levels . . . . .	93
25	NMSE and Correlation values of watermark at various image resizing levels . . . . .	95
26	NMSE and Correlation values of watermark at various image rotations . . . . .	97
27	Visible pattern on the extracted watermark image . . . . .	106
28	embedding and extraction time . . . . .	112
29	Human Visual Spectrum [33] . . . . .	130
30	HSI Colour Space [34] . . . . .	133
31	MATLAB GUI . . . . .	141
32	Test images comparison - noise addition attacks . . . . .	171
33	Test images comparison - JPEG compression attacks . . . . .	173
34	Test images comparison - JPEG2000 compression attacks . . . . .	174
35	Test images comparison - resizing attacks . . . . .	176
36	Test images comparison - rotation attacks . . . . .	179
37	Test images comparison - filtering, histogram attacks . . . . .	180
38	Test images comparison - cropping attacks . . . . .	181
39	Test images comparison - collage attacks . . . . .	182
40	Test images comparison - clipping attacks . . . . .	183

# 1 INTRODUCTION

## 1.1 Background

The unprecedented increase in piracy and digital criminality over the past 10 years has stimulated interest in the field of watermarking to enhance protection against violations of copyrighted digital material, such as digital images. According to a recent study carried out by TERA Consultants for the International Chamber of Commerce and made public in March 2010, the European creative industries lost around 9.9 billion euros and over 186,000 jobs in 2008 because of piracy, mainly digital piracy [77].

Watermarking, steganography and encryption are closely linked and sometimes combined when hiding information. The work presented here focuses on watermarking and steganography in digital images and to limit the scope of the work, does not discuss any other type of media.

Watermarking aims at identifying the creator, owner or distributor of a digital document, whereas steganography aims at hiding digital information into a digital document. Although their objectives are slightly different, watermarking and steganography are closely related, as they use similar methods to embed the required information into digital images.

Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG) and Portable Network Graphics (PNG) represent the most popular image formats on the Internet [85]. The PNG file format was created as the open-source successor to the GIF, which is a proprietary format. Most of the watermarking and steganography techniques are developed to exploit these three different image structures and they also often use Bitmap (BMP) images as intermediary results or for evaluation purpose. Such image files use the .bmp file extension.

Over the last 15 years, many watermarking methods have been developed and tested with the aim of providing reliable ways of proving image ownership. Surveys detailing the most popular watermarking techniques can be

found in the literature [60, 43]. This document does not attempt to give a comprehensive review of all the watermarking and steganographic techniques developed over the past 15 years, as there is an impressive amount of research in this area. Rather, this document discusses the most significant steps and techniques developed in the context of watermark invisibility and robustness in hiding information in digital images, in order to propose a novel watermarking technique approach. This technique will be based on advances in steganography, and may be of interest as an addition to the current state of the art techniques in this area of research.

## 1.2 Organisation

Chapter 2 introduces watermarking and steganography related to digital images. Chapter 3 presents the potential attacks that a digital image can be subjected to and the various benchmarking tools, which have been developed to measure the efficacy of watermarking and steganography algorithms in resisting these attacks. Chapter 4 discusses the main methods used in digital image watermarking and digital image steganography and compares the major embedding techniques (watermarking and steganography), developed in this area, with their benefits and drawbacks. Most of the studies revolve around the exploitation of grey scale images. However, their application to colour images might not be completely adequate since they do not take into consideration the full implication of the Human Visual System (HVS) and in particular its sensitivity to colour brightness and perception. In order to explore this further, Chapter 5 presents the different colour spaces, their relation to the HVS and some research done in this area. This results in the description of an algorithm in Chapter 6, designed to achieve robustness and the methodology used to address the shortcomings of other grey scale algorithms. Chapter 7 presents and analyses the experiment test results. Finally, Chapter 8 draws the conclusions and discusses potential improvements and future work.

## 2 DIGITAL IMAGE WATERMARKING AND STEGANOGRAPHY

### 2.1 Introduction

This chapter defines Watermarking and Steganography and outlines the fundamental differences and objectives that each tries to achieve. Although not considered in this project, Fingerprinting is briefly discussed in the conclusion.

Three techniques are inter-linked, i.e. steganography, watermarking and cryptography. The first two are quite difficult to differentiate especially for those working in areas outside this domain. Figure 1 and Table 1 may help in clarifying the differences.

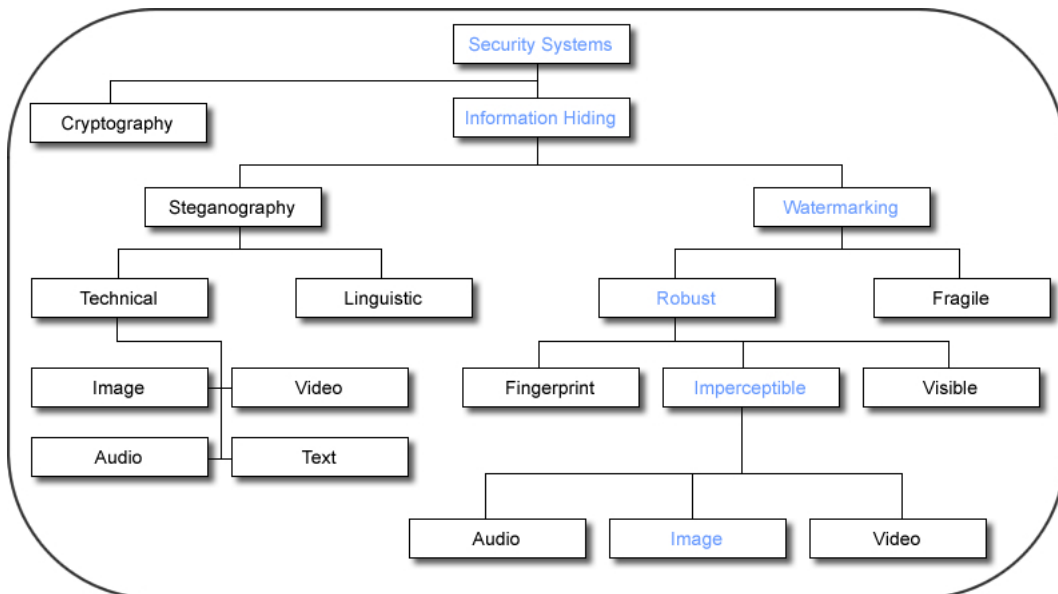


Figure 1: The different embodiment disciplines of information hiding. (Adapted from Cheddad [9], the blue path indicates the goal of this study)

Criterion/Method	Watermarking	Steganography	Encryption
Carrier	mostly image/audio/video files	any digital media	usually text based, with some extensions to image files
Secret data	watermark	payload	plain text
Key	optional	optional	necessary
Input files	at least two, unless in self-embedding	at least two unless in self-embedding	one
Detection	usually informative i.e., original cover or watermark is needed for recovery	blind	blind
Authentication	usually achieved by cross correlation	full retrieval of data	full retrieval of data
Objective	copyright preserving	secret communication	data protection
Result	watermarked-file	stego-file	cipher-text
Concern	robustness	detectability/capacity	robustness
Type of attacks	image processing	steganalysis	cryptanalysis
Visibility	sometimes	never	always
Fails when	it is removed/replaced	it is detected	de-ciphered
Relation to cover	usually becomes an attribute of the cover image. The cover is more important than the message.	not necessarily related to the cover. The message is more important than the cover.	N/A
Flexibility	cover choice is restricted	free to choose any suitable cover	N/A
History	ancient, except its digital version	very ancient, except its digital version	modern era

Table 1: Comparison of steganography, watermarking and encryption

## 2.2 Watermarking

The word watermarking is derived from the much older notion of placing a visible watermark on paper [86]. Watermarking was originally designed as an artifact to identify a specific paper maker or to discourage printed currency counterfeiting. One of the earliest watermarks was found in an Italian paper made in Bologna in 1282, and this technique quickly spread throughout Europe [6]. Paper watermarking is still in use today, in more elaborate forms such as currency notes. It is also used to signify that a paper is of high-quality.

In the digital era, a digital image watermarking process consists of embedding information into a host image (also called a cover image) so as to prove the authenticity of the image. More recently, watermarks in images are invisible to the viewer, mainly so that they don't get in the way of an image rather than to avoid detection. The more recent design processes aim at fulfilling characteristics such as invisibility, robustness, security, capacity, and complexity:

- Invisibility: the watermark cannot be detected with the human eye.
- Robustness: the embedded information is robust if it can be extracted reliably, even if the image has been modified (but not destroyed completely). Robustness thus signifies the resilience of the watermark in an image to incidental changes or image operations. This implies that it will be possible to extract the watermark after the image has been subjected to transformations and that the watermark will be identifiable.
- Security: a watermarking algorithm is considered secure if the embedded information cannot be destroyed, detected or forged, given that the attacker has full knowledge of the watermarking technique, has access to at least one piece of marked data material, but does not know the



secret key. Only the intended audience which possesses the proper key or which has knowledge of the embedding procedure can successfully extract the valid watermark.

- Capacity: describes the volume of information (usually in bits) that can be embedded. It addresses also the possibility of embedding multiple watermarks.
- Complexity: describes the effort and time needed to embed and retrieve a watermark. This parameter is essential for real time commercial applications and is usually measured in terms of computing power used over time.

There are two major steps in the digital watermarking process, as can be seen in Figure 2:

- Watermark embedding: the watermark is inserted into a host image (an encryption technique may be used).
- Watermark extraction: the watermark is separated from the host image (a decryption technique may be used).

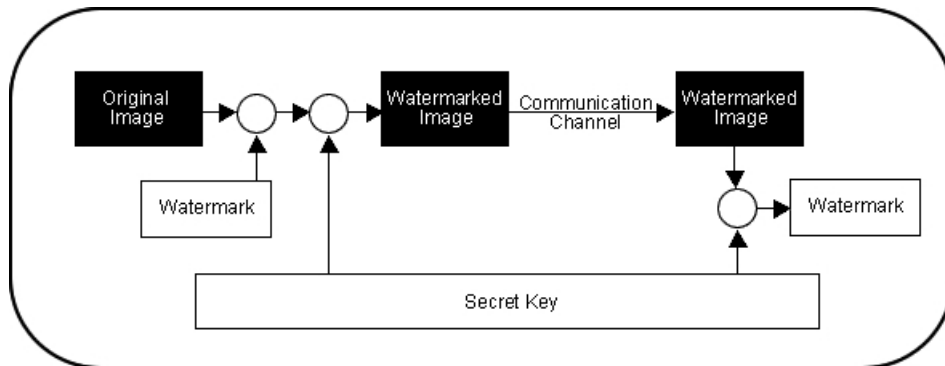


Figure 2: A general digital image watermarking system

## 2.3 Steganography

Steganography is the art and science of invisible communication [9]. This is accomplished through hiding information in other information, thus hiding the existence of the communicated information. The advantage of steganography, over cryptography alone, is that messages do not attract attention to themselves. Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

The word steganography is derived from the Greek words “stegos” meaning “cover” and “grafia” meaning “writing”, defining it as “covered writing” [9]. In image based steganography, the information is hidden exclusively in images. The idea and practice of hiding information has a long history. The Greek historian Herodotus wrote of Histaeus, who needed to communicate with his son-in-law in Greece. He shaved the head of one of his most trusted slaves and tattooed the message onto the slave’s scalp. When the slave’s hair grew back the slave was dispatched with the hidden message.

In the Second World War the Microdot technique was developed by the

Germans. Information, especially photographs, was reduced in size until it was the size of a typed full stop. Extremely difficult to detect, a normal cover message was sent over an insecure channel with one of the full stop on the paper containing hidden information. Today steganography is mostly used on computers with digital data being the carriers and networks being the high speed delivery channels.

Although steganography is an ancient subject, its modern formulation is often given in terms of the prisoners' problem proposed by Simmons [74], where two inmates wish to communicate in secret to hatch an escape plan. All of their communication passes through a warden who will throw them in solitary confinement should any covert communication be suspected. The warden, who is free to examine all communication exchanged between the inmates, can either be passive or active. A passive warden simply examines the communication to try and determine if it potentially contains secret information. If it is suspected that a communication contains hidden information, a passive warden takes note of the detected covert communication, reports this to some outside party and lets the message through without blocking it. An active warden, on the other hand, will try to alter the communication with the suspected hidden information deliberately, in order to remove the information.

A good steganographic algorithm should comply with a few basic requirements [9], including:

- **Invisibility:** the most important requirement is that a steganographic technique has to be invisible. The goal of steganography is to avoid drawing attention to the transmission of a hidden message. If suspicion is raised, then this goal is defeated.
- **Payload capacity:** unlike watermarking, which needs to embed only a small amount of copyright information, steganography aims at hidden communication and therefore requires sufficient embedding capacity

- Robustness against statistical attacks: statistical steganalysis is the practice of detecting hidden information through applying statistical tests on image data. Many steganographic algorithms leave a “signature” when embedding information that can be easily detected through statistical analysis. (More on steganalysis in Chapter 4)
- Robustness against image manipulation: during the communication process, the image may undergo changes by an active warden in an attempt to remove hidden information. Manipulation of the image can be effected before it reaches its destination. Depending on the manner in which the message is embedded, these manipulations may destroy the hidden message. It is preferable for steganographic algorithms to be robust against either malicious or unintentional changes to the image. For example JPEG compression can alter the data and accidentally destroy the hidden message.
- Independent of file format: with many different image file formats used on the Internet, it might seem suspicious that only one type of file format is continuously communicated between two parties. A powerful steganographic algorithm thus possesses the ability to embed information in any type of file. This also solves the problem of not always being able to find a suitable image at the right moment, in the right format to use as a cover image.
- Unsuspicious files: this requirement includes all characteristics of a steganographic algorithm that may result in images that are not used normally and may cause suspicion. Abnormal file size, for example, is one property of an image that can result in further investigation of the image by a warden.

The fundamental requirement of steganographic systems is that the stego-image must be as close as possible to the original image so that it does

not raise any suspicion. Embedding capacity and invisibility are the two major requirements which are widely studied in the various steganography techniques [50, 89, 64, 10, 14]. Resistance to attacks is not emphasized as much.

Over the past decade, three significant surveys on steganographic techniques have been published. Johnson et al. [38] published their extensive survey in 1999, in “Information Hiding”. Since then steganographic techniques have evolved a lot. Bailey et al. [5] evaluated the different spatial steganographic techniques, based on the Least Significant Bit (LSB) technique, applied to GIF images (published in 2006). A very comprehensive and up-to-date survey (published in 2009) on image steganography can be found in Cheddad et al. [9].

## 2.4 Conclusion

Watermarking is closely related to steganography. However, watermarking is mainly concerned with the protection of intellectual property, thus watermarking algorithms have different requirements than steganography. In watermarking all of the instances of an image are “marked” in the same way. The kind of information hidden in images when using watermarking is usually a signature to signify origin or ownership for the purpose of copyright protection. With fingerprinting, different, unique marks are embedded in distinct copies of the carrier object that are supplied to different customers. This enables the intellectual property owner to identify customers who break their licensing agreement, by supplying the property to third parties. In watermarking, the fact that information is hidden inside images may be public knowledge, sometimes it may even be visible, while in steganography the imperceptibility of the information is crucial. A successful attack on a steganographic system consists of an adversary observing that there is information hidden inside a file, while a successful attack on a watermarking

system would not be to detect the mark, but to remove it. Fingerprinting, although briefly mentioned here as a way of identifying ownership, is outside of the scope of this research.

Traditional steganography conceals information. Watermarks extend information and become an attribute of the cover image. Digital watermarks may include such information as copyright, ownership, or license. In steganography, the object of communication is the hidden message. In digital watermarking, the object of communication is the cover. It is also important to note that watermarking and steganography techniques can be used on a variety of digital media. This research is focused solely on digital images.

Before exploring the various techniques used, it is necessary to understand the type of attacks to which a digital image can be subjected. This is the purpose of the next section.

## 3 DIGITAL IMAGE ATTACKS

### 3.1 Introduction

In order to identify the weaknesses of the various watermarking techniques, one needs to understand the different types of attacks on digital images. Attacks can be unintentional or intentional. Intentional attacks are usually more difficult to survive than the unintentional attacks [73].

Attacks may further be categorised as malicious, if their goal is to remove the watermark or make it unrecoverable. Such attacks can be blind (not knowing the algorithm used for watermarking) or informed (exploiting knowledge of algorithm used for watermarking).

Non-malicious attacks on the other hand can be defined as transformations during normal use of image manipulation, such as compression, geometric and temporal manipulations, digital to analogue conversion, noise reduction or removal of part of the image (cropping).

Cox et al. [22] have discussed extensively which level of robustness is appropriate to the type of watermarking application. They mention some of the attacks and their countermeasures.

Voloshynovskiy et al. [81] and Shih [73] (p. 51-61), have classified attacks into 4 categories: (1) interference and removal attacks (image processing attacks), (2) geometrical attacks, (3) cryptographic attacks and (4) protocol attacks. This section briefly describes each attack category and the benchmarking tools that are available to compare algorithm robustness against these attacks.

### 3.2 Image processing attacks

- Filtering is the process of applying a filter to the frequency domain. Sharpening filter attack, blurring filter attack and Gaussian filter attack are examples of filter attacks.

- Re-modulation is the process of removing the noise from an image. It is an effective attack with little distortion.
- JPEG Coding distortion is a popular compression algorithm.
- JPEG 2000 Compression uses wavelet based technology, resulting in a high compression ratio without the blocky visual effect of JPEG compression.

### 3.3 Geometric transformation

Geometrical attacks, while not directly aimed at removing the watermark, instead try to either weaken it or disable its detection. This can be done using image manipulation programs such as Macromedia Fireworks or Adobe Photoshop. They potentially introduce local jittering or local geometrical bending in addition to a global geometrical transformation. As a consequence, most watermark detectors will lose synchronization with the embedded information and therefore these attacks are also referred to as synchronization attacks. They include:

- Scaling: the process of down sampling by reducing the length and width of an image, followed by up sampling through interpolation.
- Rotation: a clockwise or anti-clockwise angle rotation is applied to an image.
- Clipping: a portion of an image is kept, the rest is removed.
- Linear transformation: achieved by applying a linear transformation matrix.



- Bending: this technique was originally proposed by Petitcolas [59]. An image is interpolated after nonlinear geometric distortions are applied. A small amount of compression and noise are also added.
- Warping: it is a pixel by pixel remapping such that any shapes portrayed in the image have been significantly distorted.
- Perspective projection: parallel lines converge, the object size is reduced. As a consequence the object shape is not preserved.
- Collage: a combination of different image parts.

### **3.4 Cryptographic attack**

The aim of Cryptographic attack is to find the encrypted key used for embedding the watermark, if any. An exhaustive key search may be used as a strategy against any encrypted data by an attacker. It involves systematically checking all possible keys until the correct key is found. This technique, better known as brute-force attack, is very process-intensive. But once found, the watermark can be overwritten. It is a deliberate attack, also called a malicious attack.

### **3.5 Protocol attack**

The aim of Protocol attack is to cause ambiguity regarding true ownership of the image, by removing the original watermark and reinserting another one in its place. It is part of the intentional attacks (malicious attacks). Approaches employed most frequently are filter models. Removing noise from the marked image using median or high pass filtering, are methods considered very likely to succeed [73].

## 3.6 Benchmarking tools

To prevent unauthorised use of digital content using digital watermarking, the robustness of the watermarking should be evaluated in detail. Benchmark tools have been developed for this purpose. These tools can show which attack will break the embedded digital watermarks.

### 3.6.1 Stirmark

Stirmark is a generic tool [61], developed in 1997, to evaluate the robustness of watermarking algorithms. It determines an overall score by applying various types of attacks available (9 in total).

The first version of Stirmark introduced random bilinear geometric distortions to de-synchronise watermarking algorithms. Then several versions followed improving the original attack but also introducing a longer list of tests. In January 1999 Stirmark 3.1 was released as a benchmark tool. It allows for fair evaluation procedures for watermarking systems.

Stirmark can be considered also as a generic steganalysis tool performing removal of the hidden message. It simulates a resampling process, by introducing the same kind of errors into an image as printing it on a high quality printer and then scanning it again with a high quality scanner. If information embedded by an algorithm into an image does not survive the Stirmark process, then the steganographic technique used should be considered unacceptably easy to break.

### 3.6.2 Optimark

Optimark is a benchmarking tool for still image watermarking algorithms that was developed in the Artificial Intelligence and Information Analysis Laboratory at the Department of Informatics, Aristotle University of Thessaloniki, Greece [75].

The attacks that are currently included in Optimark are the following: Cropping, Line and Column Removal, General Linear Transformation, Scal-

ing, Shearing, Horizontal Flip, Rotation, Auto cropping and Auto scale, Sharpening, Gaussian Filtering, Median, JPEG.

### **3.6.3 Certimark**

Certimark is a benchmarking platform for certification of watermarking algorithms at European level, resulting from the collaboration of 14 academic and industrial partners [7]. It intends to perform the certification process on watermarking algorithms, thus becoming the benchmark of reference. This project is still at an early stage of development.

### **3.6.4 Checkmark**

Checkmark provides a benchmarking tool to evaluate the performance of watermarking techniques [81]. It provides additional attacks unavailable in Stirmark.

## **3.7 Conclusion**

Stirmark is by far the most used in watermarking studies for providing a large array of attacks, using automated tests. Kamiya et al. [39] however criticise the current benchmarking tools for not reflecting the reality of potential combined attacks. They propose a new benchmark tool that supports evaluations using many images. As it is still in its infancy, no recent studies using Kamiya's alternative have been found.

In the next chapter, the review the main methods used in steganography and watermarking to hide information in digital images is presented.

## 4 DATA HIDING AND STEGANALYSIS

### 4.1 Introduction

The major distinction made in the study of watermarking for digital images is between visible watermarks and invisible ones. Visible watermarks are used to mark a digital image in a clearly detectable way, in order to give a general idea of what the image looks like, while preventing any commercial use of that particular image. In this case, the purpose is to prevent any unauthorised use of an image by adding an obvious identification key, which removes the image's commercial value. On the other hand, invisible watermarks are used for author identification in order to determine the origin of an image. They can also be used in the detection of unauthorised image copying, either to prove ownership or to identify a customer. The invisible scheme does not intend to forbid any access to an image but rather to tell if a specified image has been used without its owner's formal consent or if the image has been altered in any way. This approach is certainly the one that has received the most attention in the past ten years.

Following this line of investigation, it is possible to choose many ways for hiding information.

Watermarking can be accomplished by simply feeding the following code into a DOS emulated Windows command prompt [9]:

```
C:\> Copy original_image.jpg /b + watermark.txt /b combined_image.jpg
```

This code appends any text found in the file “watermark.txt” into the JPEG image file “original\_image.jpg” and produces the combined image “combined\_image.jpg”. The idea behind this is to exploit the recognition of EOF (End of file). In other words, the watermark is appended after the EOF tag. Opening “combined\_image.jpg” in any image editing application will just display the picture ignoring anything coming after the EOF tag. However, when opened in Notepad for example, the text watermark reveals itself after displaying some data bytes. The embedded message does not

impair the image quality. Neither image histograms nor visual perception can detect any difference between the two images due to the secret message being hidden after the EOF tag. Whilst this method is simple, a range of online steganography software uses it (Camouflage, JpegX). Unfortunately, this simple technique would not resist any editing to the combined image nor any attacks by steganalysis experts.

Another simple implementation is to append hidden data to the image's Extended File Information, which is a standard used by digital camera manufacturers to store meta data information in the image header file, e.g. the make and model of a camera. Unfortunately, it is as unreliable as the previous method because it is very easy to overwrite such information.

Very early research focused on LSB insertion in the spatial domain (pixel level) of images for its simplicity and its potentially large capacity. Later scientific research considered the frequency domain and the quantisation of coefficients.

Research conducted for the purpose of this work would indicate that watermarking and steganography techniques can be classified into three main categories:

- Spatial Domain methods
- Frequency Domain methods
- Adaptive methods

Adaptive methods are treated as a special case here, because they can either be applied to the spatial domain or to the frequency domain.

The following sections examine each domain methodology and analyse their impact on achieving the optimum watermarking requirements, defined in the previous chapter.

## 4.2 Spatial Domain Methods

Spatial domain methods concern the modification of a pixel value directly on the spatial domain of an image [54]. All studies referred to in this section are applied to either JPG or BMP images.

One of the simpler approaches to hiding data within an image file is LSB insertion. Using this method, the binary representation of the hidden data is computed and LSB of each byte within the cover image is overwritten. As an example, three adjacent pixels (nine bytes) are shown with the following encoding:

```
10010101 00001101 11001001
10010110 00001111 11001010
10011111 00010000 11001011
```

Now suppose the following 9 bits of data 101101101 are hidden. If these 9 bits are overlaid on the LSB of the 9 bytes above, the following is obtained (where the bits in bold have been changed):

```
10010101 00001100 11001001
10010111 00001110 11001011
10011111 00010000 11001011
```

Note that the 9 bits have been hidden but at a cost of only changing 4, or approximately 50%, of the LSBs.

This section outlines two main approaches to embedding in the spatial domain applied to watermarking studies.

### 4.2.1 Least Significant Bit Substitution

It is an algorithm based on pre-defined LSB substitution of pixels which composes an image as described by Shih [72], Celik [13] and Cvejic [23]. This is illustrated in the above example. The watermark (secret message) itself is converted into a bit stream before each bit is inserted in the predefined bit positions of pixels, part of the image as seen in Figure 3.

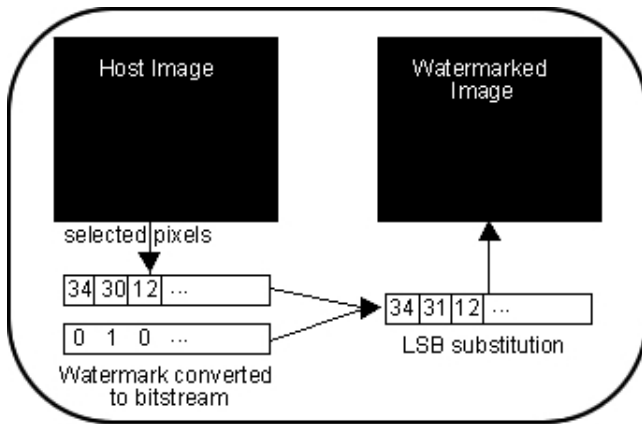


Figure 3: Pixels bit substitution

The image pixel value is decomposed into an array of 8 bit values. If the watermark bit value is 1, the corresponding image pixel LSB value is set to 1, else it is set to 0. The values 34, 30 and 12 in Figure 3 represent pixel values in the range 0 to 255.

There is a trade-off between preserving the image quality versus information hiding (watermark or secret message) payload, although it is generally accepted that modifying the LSB of each pixel does not visually alter image quality. A reasonable capacity is a third the size of the host image original size [73] (p. 34). This algorithm, presented by Shih [72] and Celik [13], is easy to break, by flipping the least significant bit of every pixel of the image, or by embedding a new watermark on top of the current one. On the other hand, it is easy to implement and it requires less processing power. To al-

leviate this concern, other algorithms [23] have been introduced whereby a private key is used to define where the bit value should be embedded (LSB, LSB2 or LSB3). Varying the bit position used, makes it a lot more difficult to find which bit is used to embed the watermark bit.

#### 4.2.2 Additive Method

This method basically adds an amount of the watermark value into each pixel value (rather than using bits of the pixel) composing the image, as seen in Figure 4.

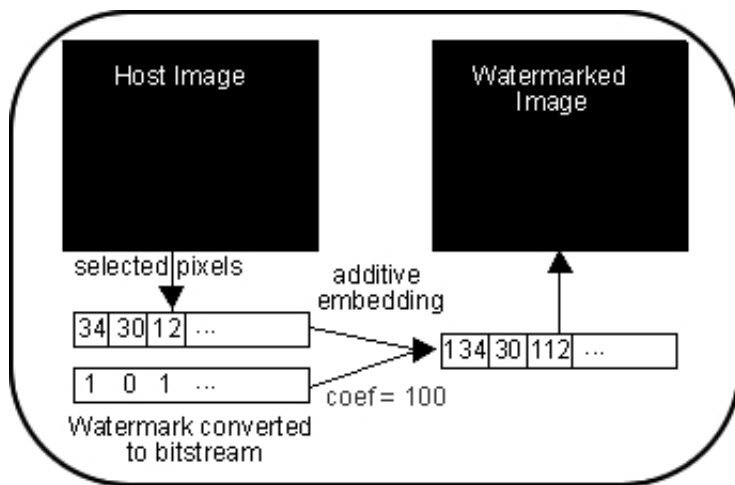


Figure 4: Additive watermarking in the spatial domain

The watermark is first converted to a bit stream. For a watermark bit value of 1, the image pixel value is increased with a predefined coefficient (100 in this example), so that for a pixel value of 34, if the watermark bit is 1, the final image pixel value becomes 134. If the watermark bit is 0, the original image pixel value remains unchanged. After the additive process, if a value becomes greater than 255, it is thresholded at a value of 255. The higher the coefficient, the more robust the watermarking technique, but the more perceptible it becomes [47]. Further this method also usually requires the original image in order to extract the watermark.



To improve imperceptibility, Lin [47] uses a block of pixels instead of individual pixels. This process shows an increased probability in recovering the hidden data, after the combined image is exposed to various attacks. However, the original image is needed to extract the watermark, since one does not know which blocks were used to embed the watermark. In this scenario, the original image needs to be stored securely, and so be easily accessible to perform the operation. Also, there is a significant loss of embedding capacity, due to the fact that a block of pixels is used rather than individual pixels. This might be a problem as regards steganography but it is not a major hurdle as regards watermarking, for the simple reason that the amount of information to be hidden is typically small. For example, a watermark can be a social security number, which uniquely identifies a person.

### 4.2.3 Histogram

Histogram equalisation is used in image processing to adjust contrasts [33]. The aim of this technique is to better distribute intensity values on the histogram. This allows for image areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

Histogram-based data-hiding is another commonly used watermarking scheme. In its simplest form, pre-defined histogram values are used to embed the watermark. Chrysochos et al. [12] chose a blind algorithm with an asymmetric key to embed the watermark into histogram values. They show that after embedding, the histogram shape is mainly preserved. They also demonstrate their algorithm to be robust against geometrical attacks such as rotation, flipping, translation, aspect ratio changes and resizing, warping, shifting, drawing and scattered tiles, as well as their combinations. They did not test their algorithm against compression nor against filtering attacks. In addition, the data hiding capacity is very much restricted to 127 bits (for

grey-scale images) and 384 bits for colour images.

Such a scheme has the advantage of recovering the original cover image from the combined image. In addition, a modified histogram does not affect the visual perception of the image. The main drawback of this technique is that the embedding strategy can be detected more easily, just by comparing the histogram shape of the original image versus the watermarked image. Chrysochos and Bayley [12, 6] suggest that the main advantage of histogram based data hiding is its robustness to rotations and other geometric transformations. On the other hand, the main difficulty associated with this technique is that there is a non-linear relationship between its representation and the pixel representation.

#### **4.2.4 Remarks**

It is well accepted in the literature that the LSBs of a digital image can be changed without degrading the perceived quality. LSB methods can generally be characterised as requiring low computation power, while potentially hiding large amounts of data. There is however a trade off between the embedding capacity and the visual impact, due to image distortion, in particular with additive methods. Image processing operations can destroy part of the watermark due to the fact that the embedded watermark might be localised to small portions of an image, making this technique not very robust. A solution to this problem is to distribute the watermark around the entire image - or significant parts of - the image, which, if removed or altered, would degrade its commercial value. For example, HVS characteristics can be applied to hide the watermark information, although success is based on favorable image characteristics. While it is not an issue in steganography (one can choose whatever image is most suited for better results), this can become an obstacle in watermarking.

One potential problem with any of the LSB methods is that they can be discovered visually by an adversary who is looking for unusual patterns, or

by using steganalysis tools.

LSB manipulation is a fast and relatively inexpensive way of hiding information but it tends to be vulnerable to spatial changes resulting from image processing or lossy compression. Such compression is a key advantage that JPEG images have over other formats. High quality images can be stored in relatively small files using JPEG compression method. LSB's natural shortcoming regarding weaknesses against image manipulations such as JPEG compression, has led researchers to look into the frequency domain. The next section discusses these techniques.

### **4.3 Frequency Domain**

Signal transformation is defined and its role in extracting valuable information from a digital signal presented.

#### **4.3.1 Signal Transformation**

To obtain further information from a signal, that is not readily available in the raw signal, mathematical transformations are applied. Fourier transforms are the most popular transformations [76]. Most of the signals in practice are time-domain signals in their raw format, so that whatever the signal is measuring, is a function of time (time-amplitude representation of the signal on the x and y axis). In many cases, the most distinguished information is hidden in the frequency content of the signal.

The frequency spectrum of a signal shows what frequencies exist in the signal. Rapid changes in the signal are known as high frequencies, whereas drawn out changes are known as low frequencies. The frequency is measured in cycles per second, or Hertz. Looking at Figure 5, the first representation is a sine wave at 3 Hz, the second one at 10 Hz.

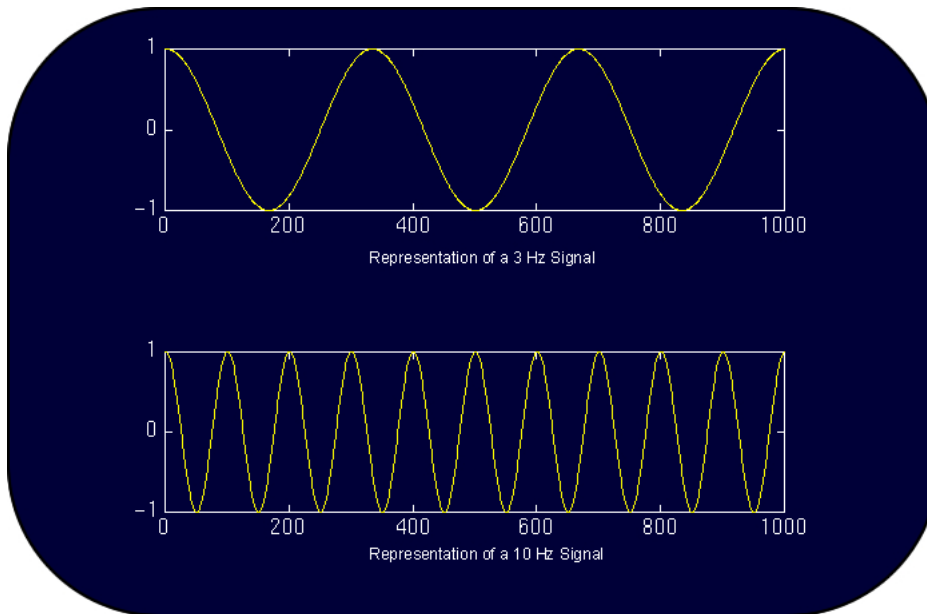


Figure 5: Frequency comparison

The Fourier Transform (FT) is a way of showing how much of each frequency is in a signal. The frequency axis starts from zero, and goes up to infinity and each frequency has an amplitude value. Frequency information is valuable because, very often, the information that cannot be readily seen in the time-domain can be seen in the frequency domain. A typical example of this is ECG (Electrocardiography), where the typical shape of a healthy ECG is well known and any deviation is very often symptomatic of an underlying pathology. Although FT is one of the most popular transforms used (especially in electrical engineering), it is not the only one. Hilbert transform, Short Time Fourier Transform (STFT), Wigner distributions, the Radon Transform, the Wavelet Transform (WT), are others. Each technique has its own area of application. FT and WT are of particular interest to image processing because they are reversible transforms, i.e., they allow going back and forward between raw and processed signals.

The following section describes the three frequency transforms used in data hiding techniques in the frequency domain of digital images and it discusses some research done with each technique.

The watermark is introduced into the frequency coefficients of the transformed image as described by Huang [36], either by Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT). Additive and multiplicative watermarking methods are used to embed the information into the frequency coefficients. The substitution method is similar to the one described in the spatial domain. Except that frequency coefficients are used rather than pixels, as seen in Figure 6 and Table 2.

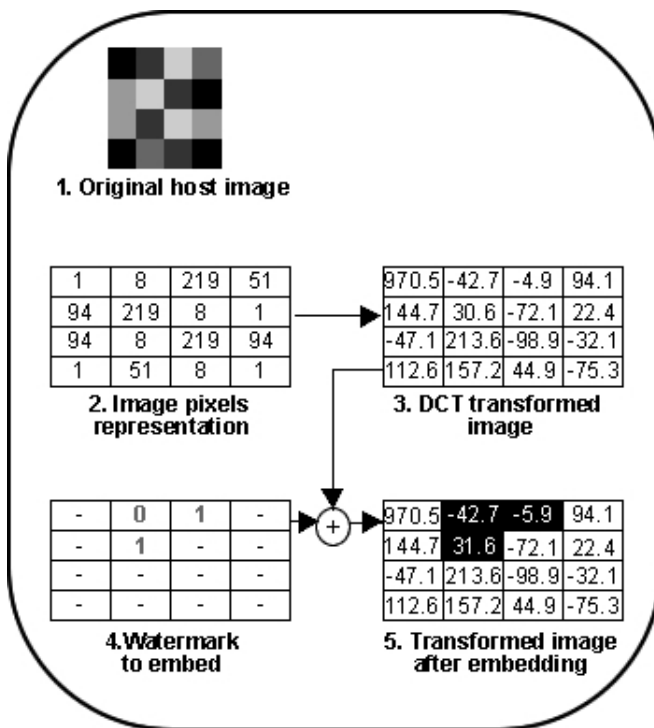


Figure 6: Example of embedding a 3 bit watermark in the frequency domain of an image using DCT

Watermark (bits)	Original image coeff.	Integer part	Binary format	Watermarked binary	Watermarked coeff.
0	-42.7	42	00101010	00101010	-42.7
1	30.6	30	00011110	00011111	31.6
1	-4.9	4	00000100	00000101	-5.9

Table 2: LSB substitution into the coefficients of the transformed image in Figure 4.

Additive watermarking is used extensively in the literature due to its simplicity.

### 4.3.2 Discrete Cosine Transform

The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance. The DCT transforms a signal or image from the spatial domain to the frequency domain. With an input image, A, the coefficients for the output image B, are:

$$B(k_1, k_2) = \sum_{i=1}^{N_1-1} \sum_{j=0}^{N_2-1} 4 \cdot A(i, j) \cdot \cos\left[\frac{\pi \cdot k_1}{2 \cdot N_1} \cdot (2 \cdot i + 1)\right] \cdot \cos\left[\frac{\pi \cdot k_2}{2 \cdot N_2} \cdot (2 \cdot j + 1)\right]$$

where the input image is  $N_2$  pixels wide by  $N_1$  pixels high,  $A(i, j)$  is the intensity of the pixel in row  $i$  and column  $j$ ,  $B(k_1, k_2)$  is the DCT coefficient in row  $k_1$  and column  $k_2$  of the DCT matrix.

All DCT multiplications are on real numbers [11, 1, 44, 91]. DCT is widely used with image compression such as JPEG lossy compression, because it has a strong “energy compaction” property [69]: most of the signal information tends to be concentrated in a few low-frequency components of the DCT. Frequency components with minimal values are discarded, leaving only the “significant contributors” of an image. DCT based algorithms are more robust to JPEG lossy compression which is also based on the DCT. Unfortunately, these DCT based schemes are not robust to basic transformations.

Zhu et al. [90] have proposed an improved DCT scheme based on perceptually shaping watermark block-wise, using localized gain factor for each block. The distortion created by the watermark addition is measured using Watson’s DCT based visual model for each block [24], rather than the entire image. An adjustment strategy is then implemented in order to seek the best trade off between robustness and imperceptibility. Their experimental results seem to concur with those of Voloshynovskiy [81]. However they differ in the choice of block size upon which to base their computations. Li and Wang [45] proposed the modification of the Quantisation Table (QT) part of the JPEG and used the middle frequency coefficients to hide the message. The aim of quantisation is to retain the valuable information while eliminating the “not so important” information.

Diaz and Grana Romay [70] have proposed a multi-objective genetic algorithm which searches the best localisation of the DCT of an image to place the mark-image-DCT-coefficients for minimal visual distortion and optimal robustness. They measured the results of this algorithm based on the Pareto-Front, which represents the trade-off between image fidelity and robustness. Predicting attack type and strength is however not a simple matter. It is therefore very unlikely that this algorithm would fit all conditions.

### 4.3.3 Discrete Fourier Transform

The Fourier Transform is an important image processing tool which is used to decompose an image. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

The DFT is the sampled Fourier Transform and therefore does not contain

all frequencies forming an image, but only a set of samples which is large enough to fully describe the spatial domain image. The number of frequencies corresponds to the number of pixels in the spatial domain image, i.e. the images in the spatial and Fourier domain are of the same size.

For a square image of size  $N \times N$ , the two-dimensional DFT is given by:

$$F(k, l) = \sum_{i=1}^{N-1} \sum_{j=0}^{N-1} f(i, j) e^{-i2\pi(\frac{ki}{N} + \frac{lj}{N})}$$

where  $f(i, j)$  is the image in the spatial domain and the exponential term is the basis function corresponding to each point  $F(k, l)$  in the Fourier space. The equation can be interpreted as: the value of each point  $F(k, l)$  is obtained by multiplying the spatial image with the corresponding base function and summing the result.

The basis functions are sine and cosine waves with increasing frequencies, i.e.  $F(0,0)$  represents the DC-component of the image which corresponds to the average brightness and  $F(N - 1, N - 1)$  represents the highest frequency. Low frequencies are responsible for the general grey-level appearance of an image over smooth areas, while high frequencies are responsible for detail (edges and noise) [33]. The DFT, based on fast Fourier transform methodology, uses phase modulation instead of magnitude components to hide messages, since phase modulation has less visual effect. The output of the transformation represents the image in the Frequency Domain. This methodology has been used by Chi-Man [16], Kim [41], Kutamura [42], Qiang [68] and Zheng [93], which demonstrate that DFT is preferable to DCT when it comes to dealing with geometric manipulations such as cropping and translation.



#### 4.3.4 Discrete Wavelet Transform

“DWT is any wavelet transform for which the wavelets are discretely sampled” [84]. Unlike the Fourier transform (sinusoidal functions), wavelet transforms are based on small waves of varying frequency and of limited duration. As with other wavelet transforms, a key advantage it has over DFT and DCT is its temporal resolution: it captures both frequency and location information (location in time). Consequently, features that might go undetected at one resolution might be easier to spot at another [76].

The WT was developed as an alternative to the Short Time Fourier Transform. Various high pass and low pass filters are applied to the signal, to filter out either high frequency or low frequency portions of the signal. To better understand the reasons behind the popularity of using this transform to hide information in images, the DWT needs to be explored in more depth. A concrete example will be used to clarify the point.

Let’s suppose a signal has frequencies up to 1000 Hz. The first stage is to split up the signal into two parts by passing the signal through a highpass and a lowpass filter (using admissibility condition) which results in two different versions of the same signal:

- The portion of the signal corresponding to 0-500 Hz (low pass portion), and
- The portion of the signal corresponding to 500-1000 Hz (high pass portion).

Either the low pass portion or the high pass portion is used, and the same operation is repeated. This is called decomposition. Assuming that the low pass portion is used, the resultant is three sets of data, each corresponding to the same signal at frequencies 0-250 Hz, 250-500 Hz, 500-1000 Hz. Taking the low pass portion again and passing it through low and high pass filters

and this results in four sets of signals corresponding to 0-125 Hz, 125-250 Hz, 250-500 Hz, and 500-1000 Hz. The same operation is repeated until a pre-defined condition is met. The result is a number of signals, which actually represent the same signal, but all corresponding to different frequency bands. Which signal corresponds to which frequency band is known, and if all are put together and plotted on a 3-D graph, time will be in one axis, frequency in the second and amplitude in the third axis.

The main reason why researchers have switched to WT from STFT is that STFT gives a fixed resolution at all times, whereas WT gives a variable resolution as follows: higher frequencies are better resolved in time, and lower frequencies are better resolved in frequency. This means that, a certain high frequency component can be located better in time (with less relative error) than a low frequency component. On the contrary, a low frequency component can be located better in frequency compared to high frequency component.

The decomposition of the signal into different frequency bands is simply obtained by successive high pass and low pass filtering of the time domain signal [76]. The original signal  $x[n]$  is first passed through a half band high pass filter  $g[n]$  and a low pass filter  $h[n]$ . After the filtering, half of the samples can be eliminated according to the Nyquist's rule, since the signal now has a highest frequency of  $\pi / 2$  radians instead of  $\pi$ . The signal can therefore be sub-sampled by 2, simply by discarding every other sample. This constitutes one level of decomposition and can mathematically be expressed as follows:

$$y_{high}[k] = \sum_n x[n].g[2k-n]$$

$$y_{low}[k] = \sum_n x[n] \cdot h[2k-n]$$

where  $y_{high}[k]$  and  $y_{low}[k]$  are the outputs of the high pass and low pass filters, respectively, after sub sampling by 2.

This decomposition halves the time resolution since only half the number of samples now characterises the entire signal. However, this operation doubles the frequency resolution, since the frequency band of the signal now spans only half the previous frequency band, effectively reducing the uncertainty in the frequency by half. The above procedure, which is also known as the subband coding, can be repeated for further decomposition. At every level, the filtering and sub sampling will result in half the number of samples (and hence half the time resolution) and half the frequency band spanned (hence double the frequency resolution). Figure 8 illustrates this procedure, where  $x[n]$  is the original signal to be decomposed, and  $h[n]$  and  $g[n]$  are low pass and high pass filters, respectively. The bandwidth of the signal at every level is marked on the figure as "f".

As illustrated in Figure 7, assume that the original signal has 512 sample points, spanning a frequency band of zero to  $\pi$  rad/s. At the first decomposition level, the signal is passed through the highpass and lowpass filters, followed by sub-sampling by 2. The output of the highpass filter has 256 points (hence half the time resolution), but it only spans the frequencies  $\pi/2$  to  $\pi$  rad/s (hence double the frequency resolution). These 256 samples constitute the first level of DWT coefficients as illustrated in Figure 8. The output of the lowpass filter also has 256 samples, but it spans the other half of the frequency band, frequencies from 0 to  $\pi/2$  rad/s.

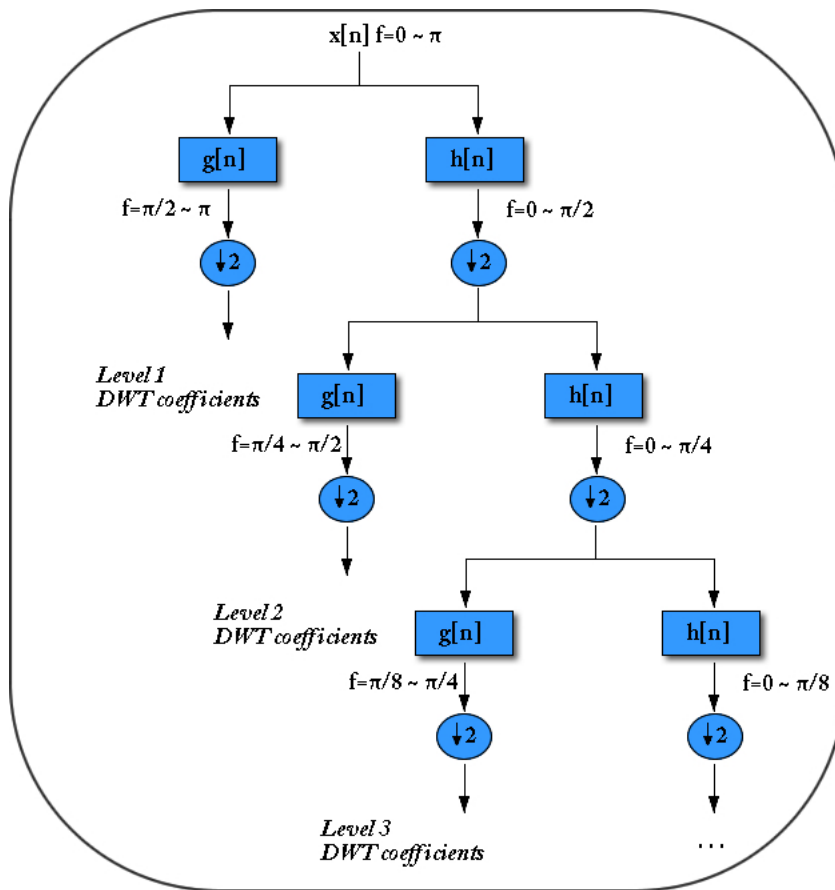


Figure 7: Subband Coding Algorithm

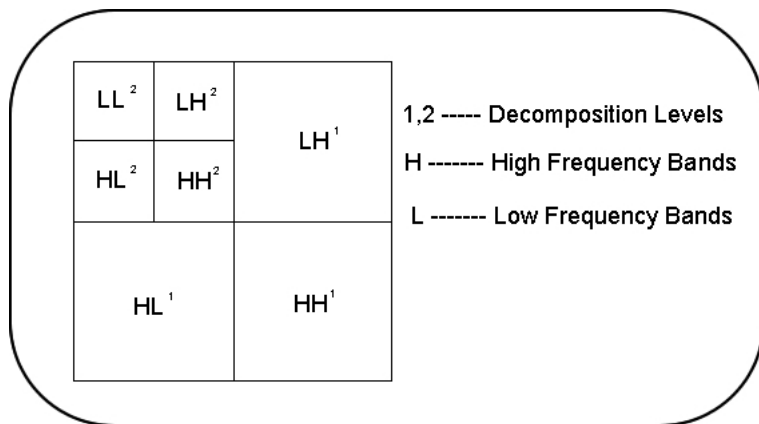


Figure 8: DWT Frequency decomposition

This signal is then passed through the same lowpass and highpass filters for further decomposition. The output of the second low-pass filter followed by subsampling has 128 samples spanning a frequency band of 0 to  $\pi/4$  rad/s, and the output of the second highpass filter followed by subsampling has 128 samples spanning a frequency band of  $\pi/4$  to  $\pi/2$  rad/s. The second highpass filtered signal constitutes the second level of DWT coefficients. This signal has half the time resolution, but twice the frequency resolution of the first level signal. In other words, time resolution has decreased by a factor of 4, and frequency resolution has increased by a factor of 4 compared to the original signal.

The DWT of the original signal is then obtained by concatenating all coefficients starting from the last level of decomposition. The DWT will then have the same number of coefficients as the original signal. The frequencies that are most prominent in the original signal will appear as high amplitudes in that region of the DWT signal that includes those particular frequencies.

The difference between this transform and the Fourier transform is that the time localisation of these frequencies will not be lost. However, the time localisation will have a resolution that depends on which level they appear. If the main information of the signal lies in the high frequencies,

the time localisation of these frequencies will be more precise, since they are characterised by a greater number of samples. If the main information lies only at very low frequencies, the time localisation will not be very precise, since few samples are used to express signal at these frequencies.

This procedure in effect offers a good time resolution at high frequencies, and good frequency resolution at low frequencies. The frequency bands that are not very prominent in the original signal will have very low amplitudes, and that part of the DWT signal can be discarded without any major loss of information. This is how DWT provides a very effective data reduction scheme (JPEG 2000 compression).

The DWT provides a powerful insight into an image's spatial and frequency attributes. Hence, the DWT has gained a lot of popularity (most of the recent research on digital grey scaled image watermarking is based on DWT [26, 49, 53]), for the fast transformation approach that translates an image from spatial domain to frequency domain while still providing robustness. An example of such a decomposition on "Lena" image can be seen in Figure 9.

In a general way, the watermark is inserted in the transform coefficients. The insertion process may be separated in 3 phases: computation of the DWT coefficients (using various filters such as Haar, Daubechies [34]), addition of the watermark to those coefficients (for example modifying those that are above a given threshold in the sub-bands other than the low pass sub-band) and compute the inverse DWT to reconstruct the watermarked image. Ghannam et al. [31] proposed a variant, where embedding is performed in two bands representing low and high frequency components in order to achieve both imperceptibility and robustness. Zao, Chen and Liu have proposed a combination of frequency domain transform [92], in order to benefit from advantages of both DCT and DWT.

DWT watermarking schemes are robust to JPEG and JPEG2000 compression. Another advantage is that they facilitate determining the salient

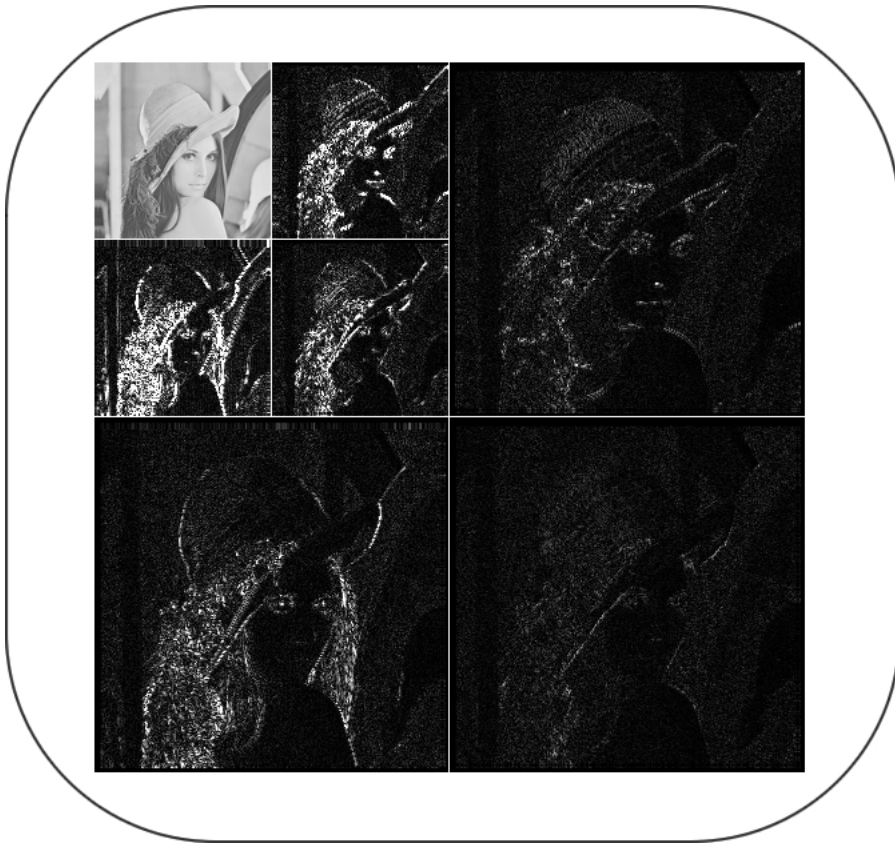


Figure 9: Two level DWT using a grey-scaled “lena” Image and the Daubechies filter [34]

areas of an image (i.e. the perceptually most significant information) where the strength of the embedded watermark is adjusted.

#### 4.3.5 Remarks

A frequency domain approach appears more attractive in general, in terms of robustness to compression and filtering than the Spatial Domain, because it decorrelates the spatial relationship between each pixel. A watermark embedded into the low frequencies increases the risk of becoming visible. On the other hand, because compression and filtering affects the high frequencies, targeting the high frequencies for watermark embedding increases the risk of watermark destruction. Embedding a watermark in the medium frequencies is a good compromise in particular when distributed across the entire image. It is a more robust technique since the embedded watermark is spread out [20].

The hidden information is generally difficult to detect, but on the other hand the watermarking payload must be small (compared to spatial domain watermarking) because of a higher risk of image distortion, hence a higher potential detection risk.

The location of the large absolute values (significant coefficients) would make watermarking even more robust. Noise addition, for example, increases the number of significant coefficients [19]. Unfortunately, images in the vast majority of cases do not contain so many significant coefficients, so watermarking capacity is limited. One answer is to artificially increase the amount of significant coefficients, through the use of (block based) chaotic map [92] to break the local spatial similarity of an image [73] (p122). However, the capacity remains limited and the risk of embedding visibility is increased.

Embedding in the DWT domain has shown promising results and certainly outperforms DCT embedding in terms of compression survival [9]. However, recent advances in the understanding of the HVS has opened new



avenues of research, which are the subject of Chapter 5.

## 4.4 Adaptive watermarking

All aforementioned watermarking methods hide the watermark in the spatial or the frequency domain, by addition, multiplication or replacement. To do so, a fixed raster (pixel grid or fixed size block image division for example) is used for embedding and extracting the watermark. Further research in the HVS suggests that exploiting certain image characteristics (corners, edges, luminance) might protect the embedded data from deliberate attacks. The hypothesis is as follow: if the watermark is hidden in regions of an image less likely to be modified because of their intrinsic value, the watermark's survival probability would be increased. Methodologies which use visually significant regions in an image to hide the watermark [3, 57, 67] are classified in this section.

Cox et al. [20] have indicated that watermarks should be embedded into regions with large magnitudes in the frequency coefficients, since geometric processing affects regions with low coefficients.

Chen et al. [14] proposed a LSB-based solution to embed the hidden message into pixels located in the image edges. They combined the fuzzy edge detector with the Canny edge detector to increase the embedding payload. They also claim this technique to be resistant to statistical analysis (steganalysis), due to the pixel selection approach they use. Although the image quality is preserved after embedding, this algorithm limits the amount of data to hide relative to the amount of edges available in the image. Images with smooth colour and intensity transitions would probably not be suited to this algorithm.

Region-based embedding is a technique that embeds a watermark over the region of an image, to spread out the message, where the embedding capacity is restricted to the chosen block size. Miller [56] has used the technique of informed coding and embedding with a similar problem of low embedding

capacity.

Mathias Schlauweg et al. [71] proposed an algorithm where the watermarking position is determined by the image content, using textels (texture elements), particularly the grey-level blob, which has the property of being scale invariant, therefore resistant to geometric attacks. By using a pre-defined significant blob and measuring the distance from the other chosen blobs, they demonstrated that this technique provides a better resistance to a wider range of geometric transformation attacks and the embedded watermark remained perceptually invisible due to the high masking effect. A disadvantage of this technique, is that the distance computed between the reference point and the other blobs must be available in order to extract the hidden information. This technique is also computer-resource-intensive, due to the complexity of this algorithm.

Zhiwei et al. [89] presented a method based on wavelet and modulus functions. They used image block division and wavelet decomposition to compute each block modulus, to decide where to embed the secret information and how much should be embedded so that it cannot be perceived by the human eye. Adapting the embedding capacity to the image texture proved to increase embedding capacity while maintaining good imperceptibility. It currently remains unclear how well this algorithm performs against geometric attacks.

Lou et al. [50] proposed an adaptive steganography scheme, capable of providing for a large embedding capacity, while preserving the visual quality of an image. They use the variation among the immediate neighbouring pixel values to predict the embedding capacity of each pixel. However, they did not measure the robustness of their algorithm against attacks.

A skin tone detection steganography algorithm is proposed by Cheddad et al. [10], which demonstrates robustness to attacks, while keeping the stego data invisible, by embedding in skin regions of an image. This is perfectly suited to steganography where the cover image can be specifically chosen with

skin attributes in it. Unfortunately, this technique is not generic enough to suit watermarking, where one has no choice when it comes to choosing the cover image.

Genetic Algorithms are an important optimisation technique [35], in the area of evolutionary computation. Khan et al. [40] have suggested the idea of exploiting the HVS, combined with genetic algorithms, to structure a watermark based on the cover image and the intended attack, to optimise imperceptibility and robustness. Although results show significant robustness over a few specific attacks, they are not predictable by nature, therefore, it remains to be seen if genetic algorithms are a practical solution to counteract multiple attacks, while trying to preserve invisibility. One way to resist attacks is to invert attack distortions at the decoding end. Gilani et al. [32] concentrated on increasing the robustness of a watermarking system by estimating a watermarked image distortion, using distortion estimation functions based on Genetic Programming. Results show superior performance compared to the previously proposed technique by Piva et al [63]. However, the technique uses known attacks to generate the estimation functions, i.e. the original image is tested against the attacked image to generate the best estimation function. There are a lot of undefined variables that can interfere with the process, such as what attack or combination of attacks should the host image be protected against, and for each attack, what is the degree of the attack. To produce the results, Gilani et al. had to pre-define the attacks, which unfortunately does not reflect the reality.

More recently, Autrusseau et al. [3] have demonstrated the usefulness of combining the advances in the understanding of the HVS with a Fourier space watermarking technique, in providing good robustness properties when subjected to many kinds of distortions. Mohanty et al. [57] suggest extracting the most perceptually important region of an image to embed the watermark, using a combination of HVS metrics such as intensity, contrast, location and edges.

Cong [18] uses Canny edge detection to isolate pixels of significance in the representation of an image (feature points matching) so that even after attacks, the watermark can be retrieved by a blind process and identified.

In the domain of exploiting the HVS for steganography, Chen et al. [14] have demonstrated an interesting hybrid edge detector algorithm (a combination of fuzzy edge detector with Canny edge detector) to improve payload capacity and invisibility, in the spatial domain, using an LSB embedding technique.

#### **4.4.1 Remarks**

Many variations of the spatial and frequency methods, or a combination of both, have been explored by the scientific community in attempt to improve watermark robustness. Table 3 summarises the advantages and disadvantages of each.

<b>Method</b>	<b>Properties</b>
Spatial Domain	<p>Simple, low processing required.</p> <p>Large payload without altering the host image visual aspect.</p> <p>Less robust against lossy compression and filters.</p> <p>Less robust against rotation, cropping and translation.</p> <p>Less robust against noise.</p> <p>Many work mainly on the BMP format.</p> <p>Tendency to be more sensitive to steganalysis.</p>
Frequency Domain	<p>Computational complexity.</p> <p>Less prone to attacks at the expense of capacity.</p> <p>Breach of second order statistics.</p> <p>Breach of coefficients distribution.</p> <p>Not as robust against geometric attacks.</p> <p>Not as robust against noise addition.</p> <p>Robust to compression.</p> <p>Variable sensitivity to steganalysis</p>
Adaptive Embedding	<p>Computational complexity.</p> <p>Small embedding space at the benefit of robustness.</p> <p>Variable resistance to rotation, translation, cropping and noise addition.</p> <p>Resistance to lossy compression, when using the DWT.</p> <p>Performs better than DCT algorithms in keeping the carrier distortion to a minimum.</p> <p>Ability to embed secret data into different orientation act as an additional secret key.</p> <p>More resistant to steganalysis.</p>

Table 3: Methods comparison

## 4.5 Steganalysis

As presented in Chapter 2, security is an important characteristic of any watermarking scheme. It may seem obvious, but it is important to note that an invisible watermark attracts less attention than a visible one. Similarly, if one does not suspect the presence of a watermark, one is less likely to try to remove it, or to interfere with it. It is in the context of watermark security that steganalysis comes into play.

The goal of steganalysis is to identify suspected information in digital streams and to determine whether or not they have hidden messages encoded in them, and, if possible, to recover the hidden information. The various problems handled by steganalysis are:

- Identification of an embedding algorithm.
- Detection of the presence of hidden message in a cover signal.
- Estimation of embedded message length.
- Prediction of location of hidden message bits.
- Estimation of secret key used in the embedding algorithm.
- Estimation of parameter of embedding algorithm.
- Extraction of hidden message.
- Or simply the destruction of any hidden message, without trying to recover the message.

In the context of steganography, steganalysis is also used to determine whether a message is secure, in which case the steganography algorithm is successful.

As seen in the previous sections, there are many different possible ways to embed data. In order to find the presence of a hidden message, one solution would be to attempt to reverse all possible embedding techniques to see whether or not a hidden message is present. But this would be a monumental task to undertake. Fortunately, just as statistics came to the rescue of code-breakers, so too does it help stegananalysts. The basis of any feasible stegananalytical investigation is in finding a set of easily measurable characteristics which change when a message is embedded into images. For digital image steganalysis, these characteristics are generally based on the statistics of the potential cover object. Such a characteristic is the high correlation among neighbouring pixels of an image. Image pixel data have statistical properties, which are disturbed by the process of embedding. Other image characteristics which can be disturbed after data hiding are colour composition and luminance. These are exploited in steganalysis of images. Various techniques for steganalysis are described in the next sections.

#### **4.5.1 Visual observation**

One method for detecting the existence of hidden messages is to look for obvious and repetitive patterns which may point to positive identification. An approach used to identify such patterns is to compare the original image with the watermarked image, using the naked eye.

Most steganographic and watermarking algorithms studied for the purpose of this research, embed the message bits either sequentially or in some pseudo-random fashion. In most algorithms, the message bits are chosen independently of the image content. In the case of using simple visual inspection, if the image contains homogeneous areas, or areas where the pixel colour is saturated at either 0 or 255, one can look for suspicious artifacts, such as a grainy structure or appearance, too few colours causing colour blocks and a lack of texture or a lack of continuity in the colour.

Even though the artifacts cannot be readily seen, one bit-plane (for example, the LSB plane) can be plotted and inspected. This attack is especially applicable to palette images for LSB embedding in indices to the palette. If, at the same time, the message is embedded sequentially, one can have a convincing argument towards the presence of a hidden message in an image.

Another method consists of selecting a specific area of an image and magnifying it, to try to find any lack of continuity where it is expected, or a lack of colour variation where it is expected.

Finally, it is interesting to note that distortions introduced into an image may resemble JPEG compression noise for instance. This “noise” can become quite obvious when the combined image is compared to the original cover images. Without the benefit of using the cover image, such noise may pass for an integral part of the image and go unnoticed. Although visual detection is simple to do, it is hard to automate [87].

#### **4.5.2 Colour Palette Observation**

Palette-based images such as GIF images, are a class of images for which steganalysis methods have been proposed in the past. Since pixel values in a palette image are represented by indices into a colour look-up table which contains the actual colour RGB value, even minor modifications to these indices can result in annoying artifacts. Visual inspection or simple statistics from such stego-images can yield enough tell-tale evidence, to discriminate between stego and cover-images, since an 8 bit colour table only contains 128 different colours.

To counteract this, embedding techniques proposed in EzStego (a steganography tool downloadable from [www.stego.com](http://www.stego.com), but no longer available), first sorts the colour palette so that the colour differences between consecutive colours are minimised. It then embeds the message bits in the LSB of the colour indices in the sorted palette, therefore minimising the visual artifacts.

Fridrich et al. [27] show that precise results in detection are obtained



using colour palette observation, but at the cost of expensive processing time and sometimes with problems locating the exact embedded region.

#### **4.5.3 Chi Square / Pair of Values (PoVs) Observation**

When using LSB substitution, during the embedding process, fixed sets of PoVs emerge. A pixel with an original value of 2 would become 3 if the bit to embed were 1, for example. It would remain 2 if the bit to embed were 0. Using this logic, Pfitzman and Westfield [87] introduced a powerful statistical attack that can be applied to any steganographic technique, in which a fixed set of PoVs are flipped into each other to embed the message bits. This method is based on statistical analyses of PoVs exchanged during message embedding. As the number of pixels for which LSB has been replaced increases, the frequencies of both values of each PoV tend to become equal. So for example if an image has 50 pixels that have a value 2 and 100 pixels that have a value 3, then after LSB embedding of the entire LSB plane the expected frequencies of 2 and 3 are 75 and 75 respectively. This of course is when the entire LSB plane is modified. However, as long as the embedded message is large enough, there will be a statistically discernible flattening of PoV distributions and this fact is exploited by their steganalysis technique.

#### **4.5.4 Regular Singular (RS) Steganalysis**

Statistical measures on LSBs for detecting level of embedding is, on its own, unreliable, as the LSB bit plane does not contain any easily recognisable structure [27]. And even though it appears random, it has some relation with other bit planes. RS Steganalysis exploits this property. Fridrich et al. [27] developed a steganalytic technique based on this for detection of LSB embedding in colour and grey-scale images. They analyse the capacity for embedding lossless data in LSBs. Randomizing the LSBs decreases this capacity. To examine an image, they define Regular groups (R) and Singular groups (S) of pixels based on specific properties. Then with the help of relative frequencies of these groups in the given image, they try to predict

the levels of embedding, in the image obtained from the original image with LSBs flipped and the image obtained by randomising LSBs of the original image.

#### 4.5.5 DCT domain Steganalysis

A well known algorithm named F5 is used to store the information in DCT coefficients leading to change in DCT histogram. Fridrich et al. [30, 28] have demonstrated that this change is proportional to the level of embedding. They have also shown that, if an image is cropped by 4 rows and 4 columns, one can recover the original DCT histogram.

The basic assumption here is that the quantised DCT coefficients are robust to small distortions and after cropping the newly calculated DCT coefficients will not exhibit clusters due to quantisation. Also, because the cropped stego image is visually similar to the cover image, many macroscopic characteristics of the cover image will be roughly preserved. After predicting DCT coefficient's histogram in the original image and comparing it with that of a stegoed image, the hidden message length can be calculated. Tools such as Outguess [66] have been developed to counter this attack. Fridrich et al. [29] have developed techniques using blockiness introduced in images due to histogram equalisation, to expose Outguess' counter-measure.

#### 4.5.6 Remarks

The steganalysis techniques described in the previous sections are generally specific to a particular embedding algorithm. Avcibas et al. [4] attempted to detect the presence of hidden information regardless of the technique used, essentially by building up a classifier based on a training set of cover-images and stego-images created from a variety of steganography algorithms. This was done by identifying which statistical feature of an image is disturbed after the steganographic algorithm is applied. The accuracy of this technique was reported by the author to be quite acceptable.

StegDetect [65] is a stand alone utility developed by Niels Provos, a doc-

toral graduate of the University of Michigan. Licensed under the GNU General Public License (GNU GPL), it is available for download in source code form and as a Microsoft Windows executable binary. Stegdetect can find hidden information in images using steganography schemes implemented in F5, Invisible Secrets, JPHide, Outguess, Camouflage, and JSteg. It also contains a Graphical User Interface called Xsteg.

StegSecret [58] is a steganalysis open source project (GNU/GPL). It is a Java-based multi-platform steganalysis tool which detects hidden information embedded by popular steganographic methods. It detects EOF, LSB, DCTs and other techniques. The author, Alfonso Muñoz, is currently working on implementing algorithms, to detect other steganography algorithm.

More information on steganalysis tools is made available on the Computer Forensics, Cybercrime and Steganography Resources website [?, ?].

## 4.6 Conclusion

Although research in steganography did not attract as much interest as watermarking in terms of the number of studies, the literature review conducted on both steganography and watermarking techniques, exhibits similar conclusions:

- LSB embedding in the spatial domain is simpler to implement and requires less processing power, while being generally less resistant to compression and filtering transformations. Embedding in the spatial domain of an image offers larger embedding capacity, but the hidden information seems easier to detect using statistical analysis tools.
- DCT, DFT and more recently DWT techniques have shown they better resist compression and filtering attacks, but are not performing as well when it comes to resisting geometric attacks. Embedding in the DWT domain seems to outperform DCT embedding, especially in terms of compression survival. Embedding in Frequency Domain makes hidden

information more difficult to detect, but the embedding capacity is directly proportional to the increased distortion of an image.

- Isolating and using regions of interest in an image to embed the data is a way of increasing the robustness of an algorithm to the detriment of capacity.

Invisibility, robustness and capacity are often used to evaluate watermarking methods. A preferred algorithm, as discussed in Chapter 2, should show a good trade-off between these three requirements.

Placing watermark information into the perceptually significant portions of data guarantees robustness against large numbers of attacks like compression, filtering, and scaling. Furthermore, placing such information into the perceptually insignificant portions guarantees robustness against attacks like histogram equalization [55].

As mentioned in Cox et al. [22], “robustness to geometrical transforms remains one of the most difficult outstanding areas of watermarking research”. Various algorithms and theories have been proposed for a watermark to resist geometric transformations such as rotation, scaling and transformation. A survey from Zheng et al. [88] has shown that each current method has its own advantages and disadvantages.

As seen in the previous sections, there are multiple data hiding methodologies and algorithms, each solving a particular facet of the watermarking problem, while no technique outperforms the others from all points of view. Zheng et al. [88] have published an extensive survey of Rotation Scale Translation (RST) invariant image watermarking algorithms in 2007 (summarised results in Appendix A).

Having reached this point, it becomes clear that most of the watermarking and steganography schemes are applied to grey-scale images, or colour images first transformed into grey-scale images before the embedding phase would

occur. However their application to colour images might not be completely adequate since they do not take into consideration the implication of the Human Visual System and in particular its sensitivity to colour brightness and perception. In order to explore this, further research into the HVS and the various colour models was needed. The outcome of this investigation is outlined in the next section.

## 5 COLOUR SPACES AND WATERMARKING

### 5.1 Introduction

More recent watermarking studies [57, 48] have turned their attention to colour images rather than grey-scale images. Effectively, colour may be more than just an extension of grey scale. It is considered as a key element for a number of image processing systems. Photoshop and Macromedia Fireworks are such examples of image processing software. In particular, colour space transforms have played a central role in coding, compression and transmission applications, in television, video and image processing. Colour also plays a major role in pattern recognition and digital multimedia [34], where colour based-features and colour segmentations have proven effective in indexing and retrieving image content.

Alternative colour spaces from the traditional RGB have also been studied. This is motivated in part by the fact that the exploitation of colour spaces (in digital video compression in particular) offers important colour information redundancy, which can be used for the purpose of hiding information without it being perceptible to the human eye.

The aim of this chapter is to explore in more detail how some colour spaces may be exploited for the benefit of providing a more robust watermarking scheme.

### 5.2 Colour Spaces in the context of watermarking

Liu and Chou [48] have compared the efficacy of a watermarking scheme between three different colour spaces (YCbCr, XYZ, CIElab). The algorithm used the frequency domain, extracting the wavelet coefficient with highest perceptual redundancy in each colour band, as reproduced in Figure 10, using MATLAB [34].



Figure 10: First level DWT decomposition in RGB colour channels

To do so, the DWT is performed independently on each colour channel. Depending on the subband targeted, filtering blocks of varying sizes are applied and the minimum value of the Just Noticeable Coefficients (JND) are targeted for embedding. Finally the inverse DWT is performed to rebuild the combined image.

The objectives of using the above technique are firstly to ensure watermark invisibility, and secondly to evaluate the robustness against attacks in each colour space. The watermark used is a black and white image of size 20 by 40 pixels. The Peak Signal to Noise Ratio (PSNR) values above 40db for each colour space and the more subjective visual inspection between the

original and the combined image demonstrates the watermark invisibility.

The combined image is then attacked using JPEG compression, low-pass filtering, zero-mean Gaussian noise addition, scaled down by a factor of 4 and scaled up by a factor of 4.

The authors use the bit error rate between the original and the extracted watermark after attack, which is simply a correlation resulting from the comparison between each watermark bit (original versus extracted).

After compression attacks, the bit error rate of the extracted watermark is the lowest for the watermarking scheme that is carried out in the YCbCr colour space. Results show that watermarks hidden in the YCbCr and XYZ colour spaces in particular, are better recovered after JPEG compression attacks.

Similar results are noticed with Gaussian noise attacks. These results demonstrates that the YCbCr and XYZ colour spaces have large amount of perceptual redundancy for colour pixels in this colour space. The larger the extent of perceptual redundancy, the greater the strength of the watermark signal that can be embedded, and the higher the robustness of the embedded watermark.

Although the variety of attacks is quite limited, the YCbCr colour space shows better overall robustness to attacks while preserving the watermark invisibility. This study also shows that there are special considerations to follow during the development of a watermarking algorithm in non-RGB colour spaces. When processing information in a non-RGB colour space, it is important not to create combinations of values which result in the generation of invalid RGB colours. e.g, given that RGB has a normalised value of (1, 1, 1), the resulting YCbCr value is (235, 128, 128) as per the mathematical formula in section 5.3.2. If Cb and Cr are manipulated to create a value of (235, 64, 73), the corresponding RGB value becomes (0.6, 1.29, 0.56), i.e., the green value exceeds 1. Robustness tests done against geometric attacks appear limited. Previous studies suggest that DWT algorithms perform well



against compression and filtering. This study confirms this but goes no further.

### 5.3 RGB & YCbCr Colour Spaces

In this section, the adaptation of human visual perception to colours in the different colour spaces is discussed.

A colour space is a method by which one can specify, create and visualise colour. As humans, colour is defined by its attributes of brightness, hue and colour intensity. A computer defines a colour in terms of the excitations of red, green and blue phosphors on the Cathode Ray Tube (CRT) faceplate. A printing press defines a colour in terms of the reflectance and absorbency of cyan, magenta, yellow and black inks on the paper.

Since the HVS has a limited sensitivity in perceiving visual information, it is well understood that there exists a considerable amount of perceptual redundancy in colour images [33]. The perceptual redundancy of a particular colour is represented by a colour region in which each colour cannot be distinguishable; i.e. the colour difference is close to zero. Through making the embedded watermarks part of the perceptual redundancy in colour images, watermark insertion can be achieved with transparency.

The three most popular colour models are RGB (used mostly in computer graphics), YIQ, YUV or YCbCr (used in video systems) and CMYK (used in colour printing). Only RGB and YCbCr colour spaces will be presented here for their relevance to the methodology discussed in the next chapter. See Appendix B for a presentation of other colour spaces. All mathematical formulae presented in the following sections (and Appendix B) were taken from the Commission Internationale de l'Eclairage (CIE) [8].

#### 5.3.1 RGB Colour Space

RGB colours are known as additive primary colours, because a colour is produced by adding different quantities of the three components, red, green,

and blue [33]. The RGB colour space is widely used throughout computer graphics. It is the most prevalent colour space as colour displays use red, green and blue, which are the three primary additive colours. They are represented by a three-dimensional, Cartesian coordinate system (see Figure 11).

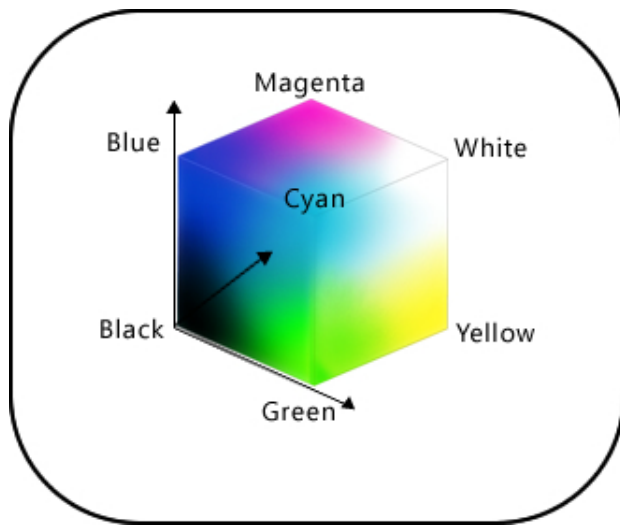


Figure 11: RGB Colour Space [34]

The different colours are points within the boundaries of this cube, with values ranging from  $[0; 1]$  (or  $[0; 255]$  in the digital world). While RGB channel format is a natural scheme for representing real-world colour, each of the three channels is highly correlated with the other two. This can be demonstrated by independently viewing the R, G, and B channels of a given image, where the entire image is clearly defined in each channel. The RGB colour space is device dependent. For example the colour produced using pixel values will alter as the brightness and contrast on a computer screen changes.

Two main conditions apply to the RGB model:

- Any of the three components must be of equal bandwidth to generate any colour within the RGB colour cube,
- Modifying the intensity of a pixel requires the modification of the three channel values.

### 5.3.2 YUV, YIQ and YCbCr Colour Spaces

These are the television transmission colour spaces. YUV is used in European televisions, while YIQ is used in Northern American television. The luma information of YUV, YIQ and YCbCr is stored in the Y channel, while the colour information is stored in the U and V channels (for YUV), I and Q channels (for YIQ) and Cb and Cr channels (for YCbCr). The YIQ is derived from the YUV colour space. They are both used for analogue video (PAL and SECAM). The YCbCr colour space, developed as part of ITU-R BT.601 (world wide digital component video standard), is a scaled and offset version of the YUV colour space. Digital video is a series of digital images displayed in rapid succession at a constant rate.

The basic equation applied to convert between RGB and YCbCr is:

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = -0.172R - 0.339G + 0.511B + 128$$

$$Cr = 0.511R - 0.428G - 0.083B + 128$$

The conversion from YCbCr back to RGB is as follow:

$$R = Y + 1.371(Cr - 128)$$

$$G = Y - 0.698(Cr - 128)$$

$$B = Y + 1.732(Cb - 128)$$

It is assumed Y is within the range [0; 1], and Cb and Cr are within the range [-0.5; 0.5] if Red, Green, and Blue are within the range [0; 1]

The fact that YCbCr is applied in digital video (YUV and YIQ are for analogue video), suggests that the YCbCr colour space could be used to good

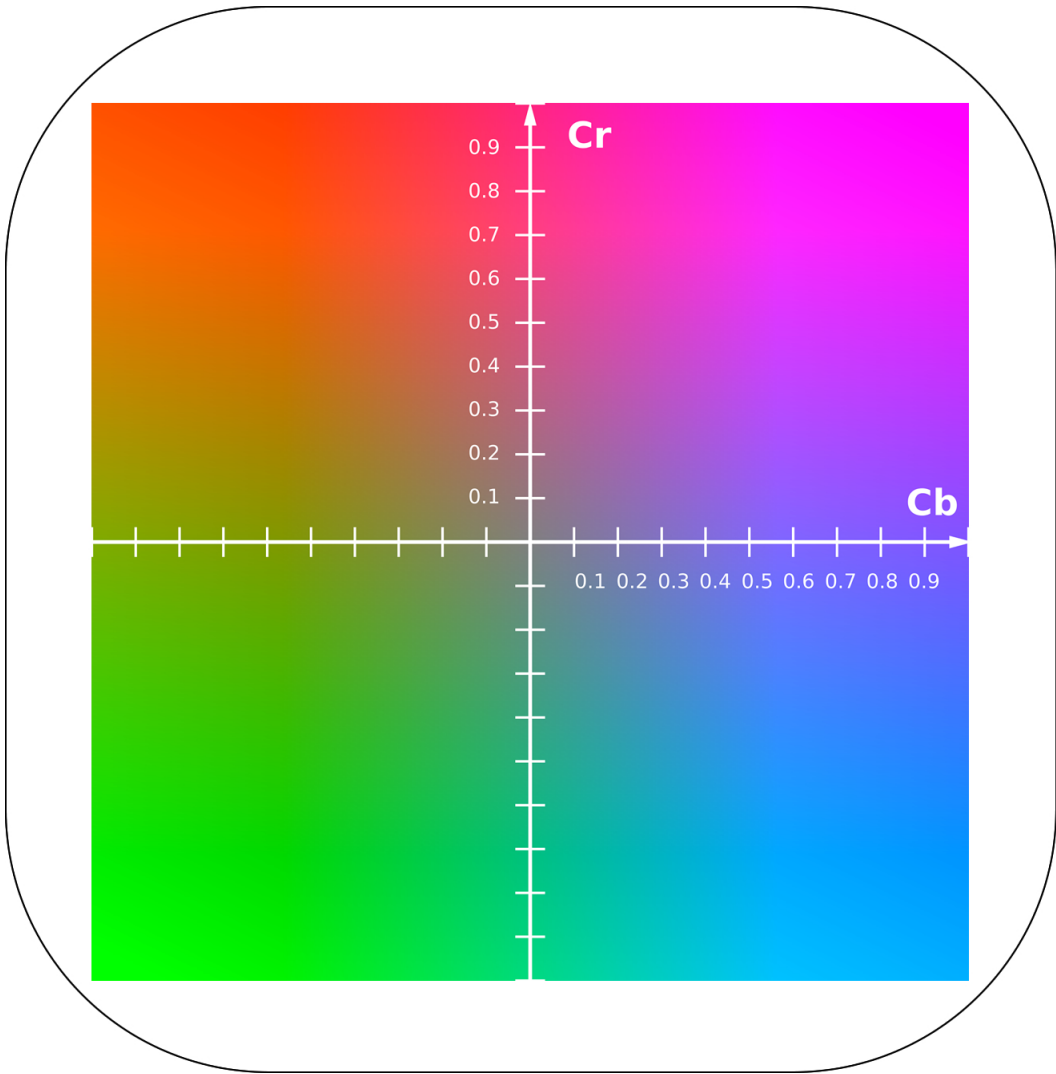


Figure 12: YCbCr Colour Space

effect with still digital images:

- Digital images are the building block of digital video
- There is a linear relationship between RGB and YCbCr

## 5.4 Conclusion

YCbCr is a component colour space used by digital video. Unlike the RGB model, YCbCr breaks the visual information into black and white (luma) signal and two colour components. It separates luminance from chrominance (lightness from colour). With many more rods than cones, the human eye is more attuned to brightness and less to colour differences. Hence the YCbCr colour system allows more attention to be paid to Y, and less to Cb and Cr. As a result, using Cb and Cr values to embed the watermark, rather than the Y channel, should achieve watermark invisibility.

Using the same line of thought, it may be possible to use a blind watermarking technique in the YCbCr colour domain. One would assume that the extra information needed to extract the original watermark without prior knowledge of the embedding process, will not affect the invisibility quality of the proposed scheme. The second important point to make about the YCbCr colour space is that the translation between it and the RGB colour space is linear and simple, therefore requiring little processing.

The RGB colour space will also be exploited for the simple fact that it is readily available to use on computer systems, therefore requiring little computation overhead. Since it is important to preserve the correlation between each component (R, G, B), the same amount of watermark will be embedded in each colour channel respectively, using an additive LSB technique. Previous research in grey-scale image watermarking has shown that it is more robust to spread the watermark across the entire image space rather than

localising it to specific image regions. For this reason, the watermark image will be rescaled to the original image size before embedding. It is hoped that using a hybrid watermarking system will improve the general robustness against most digital image attacks.

## 6 IMPLEMENTATION

### 6.1 Introduction

In many of the studies reviewed, watermarking algorithms are based either on an additive, a substitution, a multiplicative or a quantisation process, at pixel bit or coefficient levels. The watermark is extracted from the marked image either blindly or with a secret key or with the original image.

Most of the watermarking schemes using keys are symmetric (i.e. the embedding and detection keys are identical). While several methods have been proposed to watermark grey level images, only a few have been designed for colour images. Most of the time, these methods integrate colour information and the HVS by using histogram and quantization schemes, frequency domain transforms or spatial domain processing.

In light of the literature survey, it was originally proposed to use an hybrid watermarking scheme (in the spatial and frequency domains). This method and the results obtained from it are briefly discussed. However, due to disappointing results particularly after JPEG compression where the watermark was lost, it was decided to implement a new hybrid (JPEG image watermark and ASCII watermark) scheme in the RGB and the YCbCr colour spaces respectively, using the additive LSB technique (in RGB) and the pixel value addition technique (in YCbCr), following investigation of studies in various colour spaces.

The creation of a software tool to implement and evaluate the aforementioned novel approach against attack, was realised using MATLAB. It is a powerful matrix based programming language used mainly for scientific simulation programs. Use was made of MATLAB's comprehensive image processing and wavelet toolbox code library. Throughout this chapter, references are made to the code used in the implementation. Each code module is a distinct method, representing a main step in the watermark embedding and extraction process, in order to improve the code's modularity, readability

and comprehensibility.

The following section describes in detail the watermark's embedding and extraction algorithm, followed by the evaluation methodologies.

## 6.2 Methodology

### 6.2.1 Motivations

Some fundamental differences between steganography and watermarking have made the choice of basing the proposed watermarking algorithm on a strong steganographic model quite difficult:

- Steganography algorithm emphasis is generally more on capacity and invisibility rather than robustness, as opposed to watermarking techniques which in order to be successful must show invisibility and resistance to attacks as a priority.
- With steganography one has the benefit of choosing the host image, which can help in developing the most appropriate algorithm to exploit image specific features, as seen for example in the skin tone algorithm proposed by Cheddad et al. [10]. With watermarking, no such choice exists.

As seen in chapter 4, exploiting the frequency domain with DWT naturally strengthens the watermarking algorithm against JPEG compression and filtering attacks. On this basis, a second level DWT algorithm was developed as a test, using Matlab, to verify the natural robustness of watermarks to image compression. To do so, the image was processed based on the following pseudo-code:

```
[cA1,cH1,cV1,cD1] = dwt2(cover_image,'haar');  
k = value added to original coefficient;
```



```

for pixel_position=1 to watermark_total_length
    if (watermark[pixel_position] == 0)
        if(cH1[pixel_position] >= 0)

            cH1[pixel_position] = cH1[pixel_position]+k;
        end
        if(cV1[pixel_position] >= 0)

            cV1[pixel_position] = cV1[pixel_position]+k;
        end
    end
end
end
watermarked_image = idwt2(cA1,cH1,cV1,cD1,'haar',[cover_img_height,cover_img_width]);

```

The MATLAB code written to implement this algorithm can be seen in Appendix C.

The original grey-scaled image “lena” and the black and white watermark image “copyright”, of size 50 by 20 pixels, were used for the purpose of this test. After embedding the watermark, the original image and the combined image were compared visually (see Figure 13).



Figure 13: Original and watermarked “lena” image compared

The PSNR was also computed, with a result of 57.7 db, confirming that the original image could not be distinguished from the combined image obtained after the embedding process. The embedding process time and the extraction process time were both similar with a time value of 1.2 seconds. The original and the recovered watermark after embedding and extraction are compared in Figure 14.

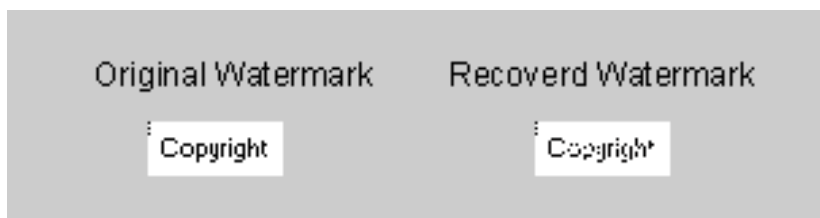


Figure 14: Original and extracted watermark

Although there is a slight loss of quality, the recovered watermark is clearly identifiable.

MATLAB was used to perform a JPEG compression test on the combined image. The watermarked “lena” image was compressed by a ratio of 10% and 20%, using the built in JPEG-compression-MATLAB-method. The watermark was then extracted (see Figure 15).

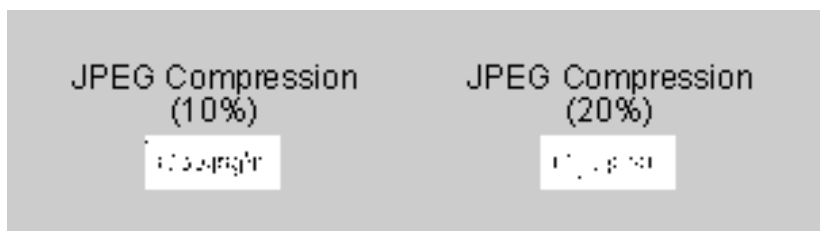


Figure 15: Extracted watermark after JPEG compression

Results obtained after JPEG compression were very disappointing. The embedded watermark has lost its original resolution, implying that the DWT implementation used for the purpose of this test cannot be used reliably for

the purpose of watermarking. This unexpected discovery prompted a re-think to the approach.

Watermarking algorithms studied by Cox [21] demonstrate that in most cases they only resolve some of the potential attacks. A practical example of this is found in Adobe Photoshop, in the form of a tool, designed by Digimarc, to embed a numerical tag into photos, using wavelet encoding techniques, which illustrates its resistance to many basic distortions but not all: for example rotating an image by 45 degrees before applying blurring and sharpening filters will destroy the watermark.

To increase the chances of watermark survival after multiple attacks, a combinative watermarking approach is suggested by Shih [73](p. 65-75), based on previous research done in this area by Tsai [78] and Shih [72], to provide for high capacity watermarks, by splitting the watermark into two parts; one part is embedded into the spatial domain while the other part is embedded into the frequency domain. The LSB technique is adopted in the spatial domain, using a pseudo random number generator to locate the pixels for the embedding process. Embedding in the lower frequency components to avoid loss after compression is the technique chosen in the frequency domain. A random permutation of the watermark is also used, showing a better resistance to cropping, in particular. Chemak et al. [15], encode the watermark and embed the result in both the frequency and the spatial domain. They use the 5/3 wavelet decomposition adapted for JPEG2000 compression. Images transformed using this algorithm are naturally resistant to this kind of attack. It also has the advantage of not losing coefficient value precision as it is an integer to integer conversion. They propose to modify pixels luminance values using the LSB2 (second least significant bit) as their embedding technique in the spatial domain.

Both studies suggest an improved robustness by combining frequency and spatial domains in their algorithm, to the detriment of performance, due to higher complexity. However, it is suggested that the success of a watermark-

ing scheme should be founded not only on robustness and invisibility, but also on the simplicity of its algorithm, to enable better commercial viability. While processing power (CPU and memory) increases all the time and becomes cheaper with every new hardware release, image manipulations require large amount of memory and fast CPU cycles to be efficient. This is not an issue when serving individual needs. However, for a commercial entity such as Getty Images, which stores millions of images, algorithm efficiency would certainly be an important decision factor. For example, if it takes 2 seconds to embed a watermark into an image using algorithm 1 and it takes 6 seconds using algorithm 2, it would cost three times more in hardware investment to choose the less efficient algorithm. If the most efficient algorithm can perform as well as the less efficient one (in terms of robustness and invisibility), the final decision becomes easy to make.

Hence, this study proposes a completely different hybrid approach, built upon the assumption that combining both spatial features and colour space might improve robustness. So rather than utilising a grey-scale image, advantage of the colour redundancy that the YCbCr colour space offers will be taken, as discussed in section 5.2. A relatively simple algorithm for improved performance has been focused on, inspired by two techniques (additive pixel value and LSB substitution) widely used in steganography. The following combined RGB and YCbCr watermarking technique is proposed:

- Non blind, additive pixel value in the RGB components, using a JPEG black and white watermark (50 by 50 pixels).
- Blind, LSB substitution in the Cb and Cr components of the YCbCr colour space, using an ASCII text watermark. This technique enables to hide more information, without compromising invisibility.

The main motivation in adopting this approach is to study:

- How well any of the two watermarks survive a large battery of attacks,
- What impact embedding in the two different colour space has on watermark invisibility,
- How difficult it is to detect and potentially destroy the watermarks,
- How efficient this algorithm is in terms of processing time.

It is important to note that the proposed algorithm did evolve during the development phase from its original conception, to address practical problems met along the way. Explanations and justifications are provided where appropriate in the following sections.

### **6.2.2 Embedding Phase**

The process of watermark embedding is divided into two phases.

#### **Phase 1**

First the image watermark “FL” is resized to the host image size, in order to distribute the watermark evenly into the host image, to increase resistance to attacks, as shown in the literature review. The RGB Image is decomposed into three matrices corresponding to R, G and B channels, so is the resized watermark (see Figure 16).

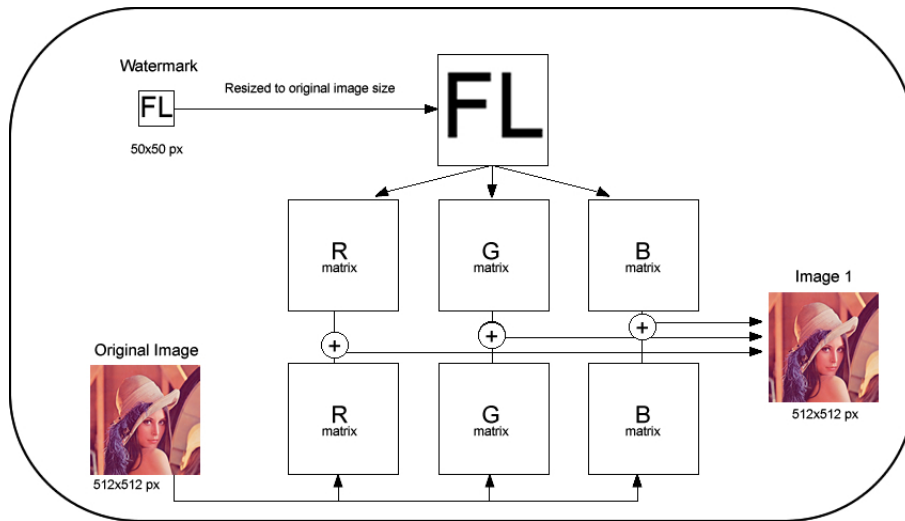


Figure 16: Watermark Embedding in the RGB channels

For each host image R, G, B matrix value, the equivalent watermark R, G, B matrix weighted value (watermark value \* k) is added so that:

- Combined\_image\_R is equal to  $\min(\text{original\_R} + (\text{watermark\_R} * k), 255)$
- Combined\_image\_G is equal to  $\min(\text{original\_G} + (\text{watermark\_G} * k), 255)$
- Combined\_image\_B is equal to  $\min(\text{original\_B} + (\text{watermark\_B} * k), 255)$

where k is a constant that increases the strength of the watermark. k equals to 0.01 is found to be the optimum in this proposed algorithm. Below this value, the watermark becomes imperceptible after extraction. Above this value, the watermark becomes visible in the combined image (Image 1).

Each new computed value is then recalibrated to an integer, in the range 0 to 255, before recombining the 3 channels together to form Image 1.

For example:

if original\_R is equal to 254 and watermark\_R is equal to 254, then the rounded value of combined\_image\_R is equal to 256 (254 + (254 \* 0.01)), which is clearly outside the upper range of 255. In this case, combined\_image\_R is set to 255.

Here is the pseudo-code (See the function insertWatermark, in Appendix D):

```

resizeimage( watermark, [original_image_width, original_image_length] );
for width = 1 to total_image_width
    for length = 1 to total_image_length

        red_pixels[length, width] = original_image( length, width, 1 ) +
            (resized_watermark_image( length, width, 1 ) * 0.01);
        green_pixels[length, width] = original_image( length, width, 2 ) +
            (resized_watermark_image( length, width, 2 ) * 0.01);
        blue_pixels[length, width] = original_image( length, width, 3 ) +
            (resized_watermark_image( length, width, 3 ) * 0.01);

    end

end

combined_image = combine(red_pixels, green_pixels, blue_pixels);
combined_image = min( combined_image, 255 );

```

## Phase 2

The combined RGB image (Image 1 in Figure 16) is then converted to YCbCr.

The basic equation applied to convert between RGB and YCbCr is:

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = -0.172R - 0.339G + 0.511B + 128$$

$$Cr = 0.511R - 0.428G - 0.083B + 128$$

The 3 component vectors (Y, Cb, Cr) are extracted. The ASCII watermark is converted to a bit stream, using 8 bits for each ASCII character.

Although not implemented here, the watermark can be encrypted before the binary conversion to make it unreadable, if discovered.

Only the Cb and Cr channels are selected to embed the watermark. Each channel is converted into a 2D matrix, which is segmented into blocks. The optimal block size, which shows the best compromise between capacity, imperceptibility and retrieval after attacks, is defined. The reason for using blocks is based on the fact that there is a better statistical chance of recovering the watermark after geometric transformation. On the other hand, this may increase the risk of detection due to the repeated patterns, if the block size is guessed. Setting every LSB of a block to the same value naturally leaves a “print”, which may be detected using steganalysis tools for example. One way of guessing the block size would be to use iterations starting with a block the size of the image. After each iteration, the block size is reduced by one pixel, until a pattern is discovered. A practical solution to this potential problem (although not implemented for simplicity and clarity) would be to alternate the use of the least significant bit (LSB) and the second least significant bit (LSB2) for embedding the watermark, based on a private key randomly generated, containing a stream of ones and zeros equal to the length of the watermark binary stream, so that for each watermark bit:

- If the corresponding private key bit position is zero, the LSB is used
- If the corresponding private key bit position is one, the LSB2 is used

This would presume that the private key is output to the user when it is randomly created and it has to be associated with the original image used. A relational database storing both information within the same record would ensure that such a record is safely kept.

Each block position is computed as seen in Figure 17 (based on the algorithm proposed by Lin and Delp [47]).



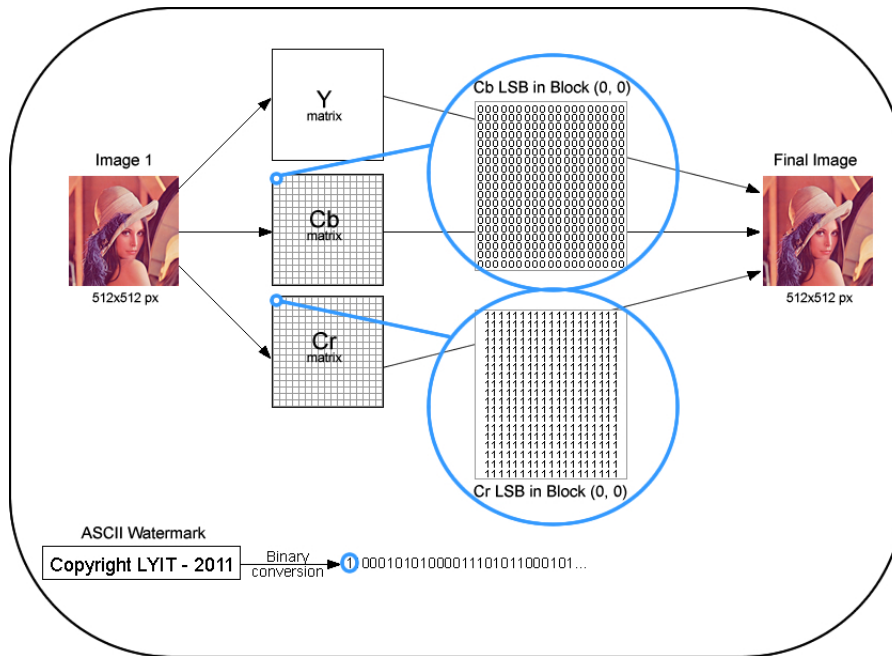


Figure 17: ASCII watermark embedding in the CbCr channels

1. For every even pixel value of the combined image, if the watermark bit is set to 1, then the LSB of Cb is set to 0 for each Cb element in the block. The LSB of Cr is set to 1.
2. For every odd pixel value of the combined image, if the watermark bit is set to 1, then the LSB of Cb is set to 1 for each Cb element in the block. The LSB of Cr is set to 0.

This is repeated until the watermark binary stream is fully embedded. The combined image is then converted back to RGB and saved. The conversion from YCbCr back to RGB is as follow:

$$R = Y + 1.371(Cr - 128)$$

$$G = Y - 0.698(Cr - 128)$$

$$B = Y + 1.732(Cb - 128)$$

An important consideration regarding this method in the YCbCr is the limited length of the text watermark, depending on the image size and the block size. This is clearly demonstrated by the following equation:

$$\text{max\_watermark\_character} = \frac{\left(\frac{\text{image\_size}}{\text{block\_size}}\right)^2}{8}$$

where 8 is the number of bits used to define one single ASCII character.

Here is the pseudo-code (See the function `encodInYCbCr`, in Appendix D):

```
[Y, Cb, Cr] = rgb2ycbcr(image(R, G, B));
watermark = ascii_to_binary(text);
position = 0;
for each block in image
    for each lsb in the block
        if (watermark_bit[position] == 1)
            Cb_lsb = 0;
            Cr_lsb = 1;
        else
            Cb_lsb = 1;
            Cr_lsb = 0;
        end
    end
    if(position > watermark_bit.length) stop;
    else position++;
end
combined_image = ycbcr2rgb(Y, Cb, Cr);
```

### 6.2.3 Extraction Phase

The watermark extraction is also done in two phases.

#### Phase 1

The watermarked RGB image (Watermarked Image in Figure 18) is converted to YCbCr following the same equation referenced in phase 2 of section 6.2.2. The 3 component vectors (Y, Cb, Cr) are extracted.

Only the Cb and Cr channels are selected to extract the watermark. Each channel is converted into a 2D matrix, which is segmented into blocks the same block size used during the embedding phase.

Each block position is computed as seen in Figure 18.

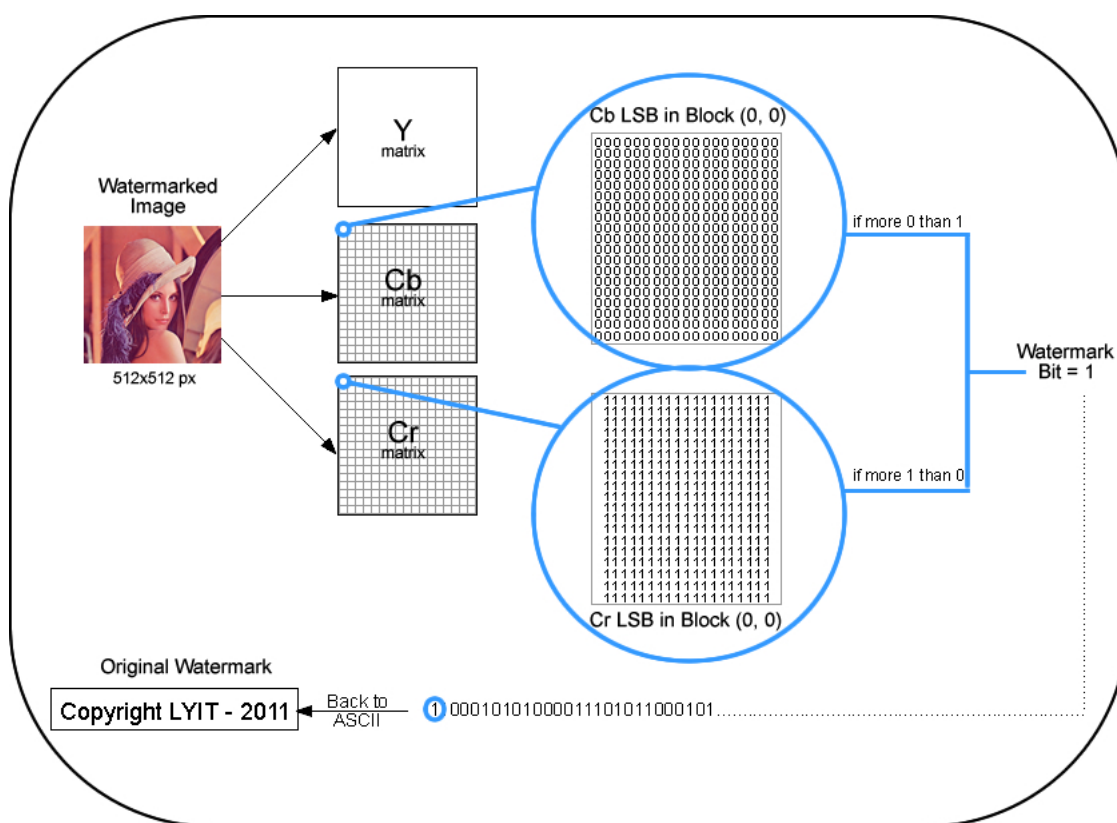


Figure 18: Watermark extraction from the CbCr channels

For each block in Cb and Cr channels in even positions (even rows and columns starting at position [0, 0]):

- In Cb the following condition is tested: the count of bits equal to 0 is greater than the count of bits equal to 1.
- In Cr the following condition is tested: the count of bits equal to 1 is greater than the count of bits equal to 0.
- If the two assumptions are true, then the watermark bit is set to 1.
- If the two assumptions are false, then the watermark bit is set to 0.

The same process is reproduced for all odd block positions (odd rows and columns starting at position [1, 1]), but this time the opposite values are taken in each channel, such as:

- In Cb the following condition is tested: the count of bits equal to 1 is greater than the count of bits equal to 0.
- In Cr the following condition is tested: the count of bits equal to 0 is greater than the count of bits equal to 1.
- If the two assumptions are true, then the watermark bit is set to 1.
- If the two assumptions are false, then the watermark bit is set to 0.

The watermark bit stream obtained is converted back to ASCII characters. The watermarked image is then converted back to RGB, before starting phase 2.

Here is the pseudo-code (See the function `decodeInYCbCr`, in Appendix D):

```
[Y, Cb, Cr] = rgb2ycbcr(image(R, G, B));
position = 0;
for each block in image
```

```

count_0 = 0;
count_1 = 0;
for each pixel in block

    if(Cb_lsb == 0 or Cr_lsb == 1) then count_1++;
    if(Cb_lsb == 1 or Cr_lsb == 0) then count_0++;
end
if(count_1 > count_0)
    watermark_bit[position] = 1;
else
    watermark_bit[position] = 0;
end
if(position > watermark_bit.length)
    stop;
else
    position++;
end

end
watermark_text = binary_to_ascii(watermark_bit);
image[R, G, B] = ycbcr2rgb([Y, Cb, Cr]);

```

## Phase 2

The combined image RGB (containing the watermark) is decomposed into each colour channel as seen in Figure 19. The original image is also decomposed into each colour channel.

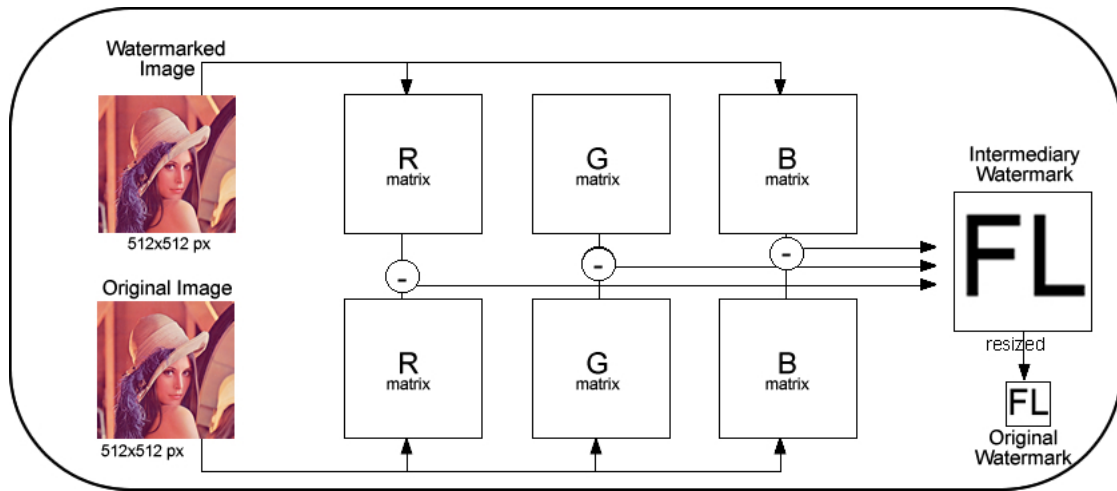


Figure 19: Watermark extraction from the RGB channels

For each R, G, B matrix value from the original image, the equivalent watermarked image R, G, B matrix weighted value (combined image value \* k) is subtracted so that:

- $\text{Watermark\_R} = \text{original\_R} - (\text{combined\_image\_R} * k)$
- $\text{Watermark\_G} = \text{original\_G} - (\text{combined\_image\_G} * k)$
- $\text{Watermark\_B} = \text{original\_B} - (\text{combined\_image\_B} * k)$

where k is the same constant used in the embedding process ( $k = 0.01$ ). Each new computed watermark value is then recalibrated to be in the range 0 to 255, before recombining the 3 resultant channels together to form the original watermark. The watermark obtained is finally resized to its original size.

Here is the pseudo-code (See the function `extractWatermark`, in Appendix D):

```
for width = 1 to total_image_width
```

```

for length = 1 to total_image_length

    red_pixels[length, width] = original_image( length, width, 1 ) -
        (watermarked_image( length, width, 1 ) * 0.01);
    green_pixels[length, width] = original_image( length, width, 2 ) -
        (watermarked_image( length, width, 2 ) * 0.01);
    blue_pixels[length, width] = original_image( length, width, 3 ) -
        (watermarked_image( length, width, 3 ) * 0.01);

end

end
large_watermark = combine(red_pixels, green_pixels, blue_pixels);
watermark_FL = min( large_watermark, 255 );
watermark_FL = downsize_to_original(watermark_FL);

```

After phases 1 and 2, the two watermarks are extracted and clearly identified. Embedding into the YCbCr allows one to extract the ASCII watermark without needing the original image, nor any information regarding the watermark. This is called a blind technique. However, the original image was needed in order to extract the original watermark from the RGB channels. It is called a non blind (or informed) technique. The benefit of using a blind technique in terms of image management is appreciated, as there is no need to store the original image.

### 6.3 Image Attacks

A series of attacks have been applied for each main digital image attack category described in Chapter 3. To do so, some attacks have been automated (scaling, rotation, compression, noise addition) using MATLAB (see code in Appendix D). The proposed algorithm has also been tested against JPEG 2000 compression attacks.

Manual attacks have been applied using Macromedia Fireworks, a popular image manipulation software used to design and enhance digital images. These attacks cover low pass filtering, gaussian blur, change of brightness and contrast, cropping, sharpening and histogram attacks.

Finally, Stirmark was also used to test against self similarity attacks, median filter attacks and to measure the strength of the proposed algorithm.

## 6.4 Measure of Invisibility

It is generally agreed that, despite the sophistication of the HVS, detecting differences between two images is difficult to do objectively. One person may perceive subtle variations that another will not notice. For this reason, mathematical formulae have been defined to rationalise these subtle differences. Many studies use the PSNR based on the mean-squared error (MSE) between two images, which is computed from the RGB colour components. The PSNR value is high when the difference between the cover image and the stego image is small. Above 38db, the human eye cannot notice the decline of image quality [89]. The PSNR is computed as follows:

$$PSNR = 10 \log_{10} \left( \frac{C_{max}^2}{MSE} \right)$$

where  $C_{max}$  represent the highest pixel value present in the image (maximum of 255).

For a cover image whose width and height are  $M$  and  $N$ ,  $MSE$  is defined as:

$$MSE = \frac{1}{M \times N} \sum_{x=1}^M \sum_{y=1}^N (S_{xy} - C_{xy})^2$$

where  $x$  and  $y$  are the image co-ordinates,  $S$  is the generated watermarked image and  $C$  is the cover image.

Recently, Lukac and Plataniotis [52] have demonstrated that since the PSNR is a component average and treats equally the errors whatever the



image content might be, it is not correlated closely enough with the human perception. If the watermark is precisely embedded into textured regions or edges, the PSNR is then inadequate to measure the image quality. Wang [82] illustrates this point by showing images which have been transformed but retain the same PSNR measure, where one mark may be invisible while another may be drastically visible. Several alternatives have been proposed [25, 82] to overcome the PSNR limitation. Wang [82] presented a Structural Similarity (SSIM) index, as a method of improving the similarity measure between two images. The SSIM index is a full reference metric, in other words, the measuring of image quality is based on an initial uncompressed or distortion-free image as reference. The SSIM metric is calculated on various windows of an image. The measure between two windows  $x$  and  $y$  of common size  $N$  by  $N$  is:

$$\text{SSIM}(x,y) = \frac{2(\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

where:

$\mu_x$  is the average of  $x$ ;

$\mu_y$  is the average of  $y$ ;

$\sigma_x^2$  is the variance of  $x$ ;

$\sigma_y^2$  is the variance of  $y$ ;

$\mu_{xy}$  is the covariance of  $x$  and  $y$ ;

$c_1 = (k_1L)^2$ ,  $c_2 = (k_2L)^2$  are two variables to stabilise the division with weak denominator;

$L$  is the dynamic range of the pixel values (typically  $2^{\#bits/pixel} - 1$ ;

$k_1 = 0.01$  and  $k_2 = 0.03$  by default;

In order to evaluate the image quality this formula is applied only on the luma component. The resultant SSIM index is a real number value between

-1 and 1, and the value 1 is only reachable if two sets of data are identical. Typically it is calculated on window (block) sizes of 8 by 8. The window can be displaced pixel-by-pixel on the image but it is proposed to use only a subgroup of the possible windows to reduce the complexity of the calculation.

To conclude, both the PSNR and the SSIM metrics will be used to measure the visual differences between the original image and the combined image. The PSNR has been used extensively in the past as a reference metric. The SSIM is much more recent and is a candidate as a reasonable alternative.

## 6.5 Measure of Robustness

In order to measure the robustness of the proposed technique, apart from the visual watermark comparison (original and extracted after attacks), an objective way of measuring the watermark distortion after subjecting the combined image to various attacks was required. These attacks include common image processing operations such as filtering, compression, histogram equalization, intensity adjustment, gamma correction and geometric transformations like cropping, scaling, and rotation. One of the most popular difference distortion measures is the Normalized Mean Squared Error (NMSE) metric which is defined as:

$$NMSE = \sum_{ij} (y(i, j) - x(i, j))^2 / \sum_{ij} x^2(i, j)$$

This metric is used to evaluate the distortion which has occurred in the extracted watermark logo after attacks. A correlation coefficient, computed by matching the pixel similarity between the original watermark and the extracted one, is also used. A correlation coefficient value of 1 means 100 percent match. The MATLAB code to compute both metrics is listed in Appendix D, under the function named “wSimilarity”.

Regarding the extracted ASCII watermark, the most objective method is that the text must match exactly what was embedded in the YCbCr in order to prove robustness to attacks. Although presence of ASCII characters after extraction demonstrates that ASCII text was embedded in the first place, failure will be considered if more than a single character mismatch occurs, as the text may not make sense any longer.

## **6.6 Steganalysis**

To complete this experimentation, an evaluation of how detectable the proposed algorithm is, is performed using Stegdetect [65] and StegSecret [58]. Both are often cited and used in steganography research studies. They can detect a wide range of steganography algorithms. The current source code version of Stegdetect is 0.6, released in September 2004. The latest beta release (17/12/2007) of StegSecret is used. Both tools provide potentially interesting feedback regarding detection, given that it's known the proposed technique is inspired by LSB steganography.

## **6.7 Conclusion**

The next chapter presents the test results and analysis.

## 7 RESULTS AND ANALYSIS

### 7.1 Introduction

In this chapter, the results of the experiments are reported and their interpretation is given. Experiments were completed on a computer running Microsoft Windows Vista, equipped with an Intel Core 2 Duo 2 GHz (Gigahertz) CPU (Central Processing Unit) and 2 GB of RAM (Gigabytes of Random Access Memory).

The strengths and weaknesses of the proposed algorithm were analysed, as regards invisibility, robustness to attacks, potential detection using steganalysis tools and potential destruction using Stirmark. Although seven test images have been used during experimentation (results can be seen in Appendix E), only results based on “lena” image will be presented in this section. But first the reasons for the choice of images to test with will be explained.

### 7.2 Image Database

“It is important to test an image watermarking software on many different images and for fair comparison, the same set of sample images should always be used” [62].

Digital images can exhibit different characteristics, interesting from the signal processing point of view: textured or smooth areas, size, synthetic, with straight edges, sharp, blur, brightness and contrast. It is difficult to get an exhaustive list of classes of pictures, and stock photo companies have a lot of difficulties in setting up a satisfactory and uniform index.

Some image databases already exist for image processing research. The USC-SIPI Image Database [80] is an example of such a database where one can find the “classics”, e.g lena, baboon, peppers, and so on. Although, the copyright status of these images is not stated clearly, it was decided to use these classics for testing the proposed algorithm:

- They have been used extensively to test various steganography and watermarking algorithms.
- In order to compare fairly the strengths and weaknesses of the proposed algorithm to others, the same set of colour images had to be chosen.

Seven colour images were used in total, each of size 512 by 512 pixels: “crown”, “girl”, “lena”, “baboon”, “plane”, “peppers”, and “boat”, as shown in Figure 20.

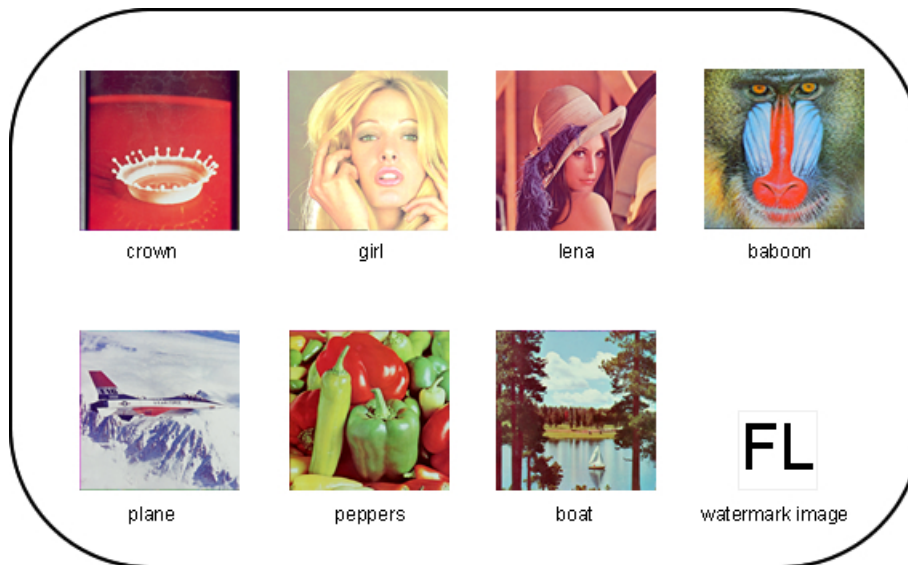


Figure 20: Host images and watermark image

Each image went through the same testing process:

- The image was first loaded individually into the MATLAB Graphical User Interface (GUI). See Appendix D for code reference of the GUI.
- A 50 by 50 pixels, black and white JPEG image (“FL”) was used as the watermark image, to embed in the RGB space of the image, as shown in Figure 20.

- An ASCII text “Copyright LYIT - 2011” was used as the second watermark, to embed in the YCbCr colour space of the image.
- After the embedding process, an invisibility test was performed on the combined image, after inserting only one watermark, then the other, then both. See Appendix D for code reference of the PSNR and SSIM computation.
- Then each image was submitted to a battery of attacks. After each attack, the extraction success of each watermark was reported. See Appendix D for code reference of the test attacks using MATLAB and Appendix E for test results.
- The combined image was submitted to two steganalysis tools (StegDetect and StegSecret), to measure the probability of detection, by comparing the original image steganalysis results with the combined image steganalysis results.
- Finally the combined image was submitted to Stirmark attacks to measure the degree of survival of each watermark.

“lena” was chosen as this experiment reference image because it is the most widely used image throughout the literature. The image attack results for the other images are available in Appendix E.

### 7.3 Invisibility Analysis

A visual test was performed to try to detect differences between the original image and the combined image, for each image. As seen in Figure 21 for the particular case of image “lena”, the original image is “4.2.0.4-lena.tiff”

and the watermark image (after embedding “fl\_small.jpg” and the ASCII text “Copyright LYIT - 2011”) is “watermarked\_img.bmp”.

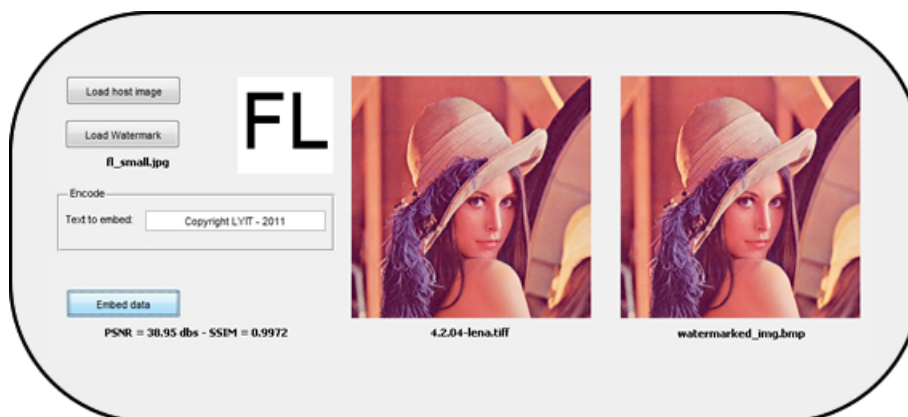


Figure 21: MATLAB GUI comparing the original “lena” image and “lena” after the proposed hybrid watermarking method is performed

Ten persons, chosen randomly amongst family and friends, were given one minute to look at the two images only, shown in Figure 21. Any evidence that could distinguish the original from the watermarked image was previously removed, such as the file name which appears below each image, in Figure 21. Each person was then asked if they could find any differences between the two images, and if yes to point them out. Four out of the ten persons thought that the two images were the same. The remaining six suspected the two images were not the same, but when asked to point out the differences, no one was able to point them out with 100% conviction. 100% conviction means without any hesitation nor change of opinion.

Having completed this simple subjective test, the PSNR and SSIM values were also computed, after individual watermarks were embedded, and then with the hybrid watermark. To do so, three tests were run in parallel:

- A first test measured the PSNR and SSIM of the combined image after embedding the image watermark in the RGB only.

- A second test measured the PSNR and SSIM of the combined image after embedding the ASCII watermark in the YCbCr only.
- A third test measured the PSNR and SSIM of the combined image after embedding both watermarks.

A SSIM value of 1 means the two images compared are identical. A PSNR value greater than 35 decibels means the two images compared are not visibly different. Table 4 summarises the results.

Colour Space	RGB	RGB	YCbCr	YCbCr	Hybrid	Hybrid
Images	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
crown	39.3	0.9909	48.29	0.9972	38.77	0.9882
girl	40.36	0.9988	48.86	0.9981	39.81	0.9967
lena	39.47	0.9991	48.55	0.9982	38.95	0.9972
plane	38.83	0.9995	47.93	0.9976	38.31	0.9971
boat	39.36	0.9989	48.34	0.9985	38.88	0.9973
pepper	38.94	0.9916	48.01	0.9982	38.43	0.9899
baboon	39.47	0.9995	48.56	0.9995	38.96	0.9989

Table 4: PSNR and SSIM Results

Looking at the results, the first observation to make is that there is very little difference between the PSNR values, from one image to another, within each colour space, although each image tested has different characteristics to the others. For example, the PSNR values range between 38.83 and 40.36 in the RGB, they range between 47.93 and 48.86 in the YCbCr and they range between 38.31 and 39.81 using the hybrid embedding.

Embedding in the YCbCr only, also shows a higher PSNR value compares to the RGB only. This finding might be explained by two reasons:

1. LSB embedding in the YCbCr has less visual impact than the additive method in the RGB.



2. The image watermark binary payload is significantly greater than the ASCII watermark binary payload.

The SSIM values are all around 0.99, showing no appreciable difference between the original image and the watermarked image, regardless of the embedding technique used and the colour space chosen. These results tally with the visual inspection, in demonstrating the invisibility of the proposed watermarking scheme. The comparable results, despite using very different images (from the set of seven images), also suggests that the proposed hybrid watermarking algorithm should remain invisible regardless of the kind of image used.

## 7.4 Robustness Analysis

This section analyses each watermark survival after attacks against the watermarked image “lena”. For each attack, the NMSE and the Correlation coefficient (R) are computed. Both metrics are used to evaluate the distortion of the watermark image (“FL” logo) after attacks. As well as these two metrics, a visual inspection on the extracted watermarks is performed to judge their survival. Attacks are performed using various tools:

- MATLAB: annotated with (M).
- Macromedia Fireworks: annotated with (F).
- Stirmark: annotated with (S).

### 7.4.1 JPEG Compression Attack

JPEG is one of the most widely used compression algorithms and any wa-

termarking system should be resilient to some degree of compression. JPEG compression with different quality factors are applied to the watermarked image “lena”. Table 5 summarises the results.

Compression (%) (M)	NMSE	R	Extracted Watermarks
5	0.024521	0.84	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">UU T NUUT 'N0 °. UTb* WU=Z*6QUx</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; text-align: center;">  </div> </div> <div style="width: 45%;">  <p style="text-align: center; font-size: small;">compression_95_percent.jpg</p> </div> </div>
45	0.024498	0.64	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">V!6éj* ú@*ÆZ4)- EV%þÉ ú¶'éNa</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; text-align: center;">  </div> </div> <div style="width: 45%;">  <p style="text-align: center; font-size: small;">compression_55_percent.jpg</p> </div> </div>
85	0.024067	0.30	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;"> </div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; text-align: center;">  </div> </div> <div style="width: 45%;">  <p style="text-align: center; font-size: small;">compression_15_percent.jpg</p> </div> </div>
95	0.022021	0.09	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;"> </div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; text-align: center;">  </div> </div> <div style="width: 45%;">  <p style="text-align: center; font-size: small;">compression_5_percent.jpg</p> </div> </div>

Table 5: Watermark survival results after JPEG compression

It can be clearly seen that the ASCII text watermark has not survived any level of JPEG compression. However the watermark image “FL” is still perceivable up to 85% of compression (15% JPEG quality factor). At this level, the image has lost its commercial value. A plotted NMSE and correlation coefficient values are illustrated in Figure 22.

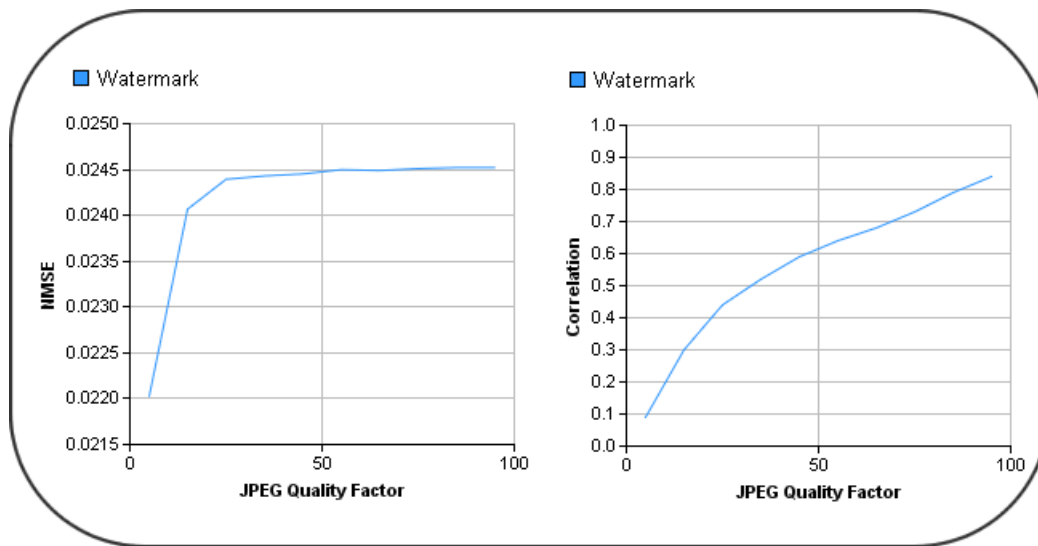


Figure 22: NMSE and Correlation values of watermark at various JPEG quality factors

The correlation coefficient values computed and graphed in Figure 22, reflect very precisely what can be observed of the watermark image after extraction. In light of these results, the proposed algorithm is robust against JPEG compression up to 85% compression.

#### 7.4.2 JPEG 2000 Compression Attack

JPEG2000 is another kind of compression algorithm which uses wavelet instead of the DCT. JPEG 2000 Compressor 1.0 [2], was used for the purpose of this test. Different quality factors were applied to the watermarked image “lena”. Table 6 summarises the results.

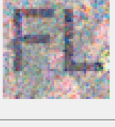
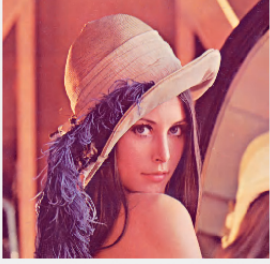
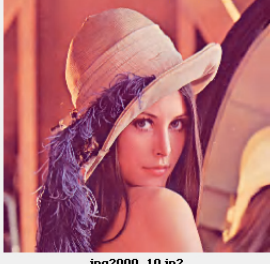
Compression (%) (M)	NMSE	R	Extracted Watermarks
10	0.015541	0.94	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <pre>nM( eW)âU7ñæXx7 P@QWÔ jê ºeb</pre> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>jpg2000_90.jp2</p> </div> </div>
50	0.024503	0.79	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <pre>[z °Ú'Á,þ Fs2¼] D &lt;h Áû « O½</pre> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>jpg2000_50.jp2</p> </div> </div>
80	0.024283	0.63	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <pre>NWúYÁi O?+ Ç*è[ , ìèÚÁv</pre> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>jpg2000_20.jp2</p> </div> </div>
90	0.023991	0.53	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <pre>e É i l wMÚ ] V&lt;R5ÈÔxÁ¼¼- F ää</pre> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>jpg2000_10.jp2</p> </div> </div>

Table 6: Watermark survival results after JPEG compression

It can be clearly seen that the ASCII text watermark has not survived any level of JPEG 2000 compression. However the watermark image “FL” is still perceivable at 90% of data lost in the image (or 10% JPEG 2000 quality factor). A plotted NMSE and correlation coefficient values are illustrated in Figure 23.

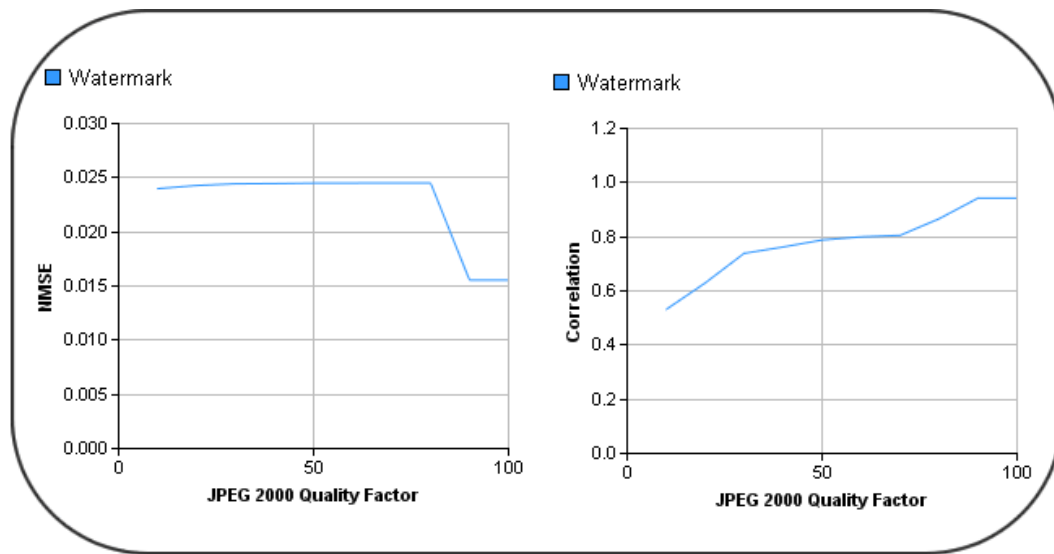


Figure 23: NMSE and Correlation values of watermark at various JPEG 2000 quality factors

The correlation graph in Figure 23 confirms that even after applying a compression ratio of 90%, the watermark image is still clearly identifiable. This demonstrates the proposed algorithm is resistant to JPEG 2000 compression.

#### 7.4.3 Noise Addition Attack

Noise has been added to the watermarked image “lena” at varying degrees, 100% meaning all pixels were modified, 50% meaning half of the number of pixels were modified and so on. The results are illustrated in Table 7.









Attack level (%) (M)	NMSE	R	Extracted Watermarks
100	0.00032	0.86	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <p>Εύλη+&lt;σύ μ ΜεΑ &gt;sb Άέq&amp;LUñ 196j'</p> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">noise_0.bmp</p> </div> </div>
50	0.0087	0.94	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <p>Copyright LYIT - 2011</p> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">noise_1.bmp</p> </div> </div>
33	0.0117	0.95	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <p>Copyright LYIT - 2011</p> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">noise_2.bmp</p> </div> </div>
25	0.01278	0.96	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <p>Copyright LYIT - 2011</p> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">noise_3.bmp</p> </div> </div>

Table 7: Watermark survival results after noise addition

The ASCII text watermark does not survive the noise addition at 100% (all pixels modified), but the watermark image extracted is clearly identifiable at this level of noise addition. This is confirmed by a correlation coefficient of 0.86. In all other cases, both watermarks survive noise addition attacks.

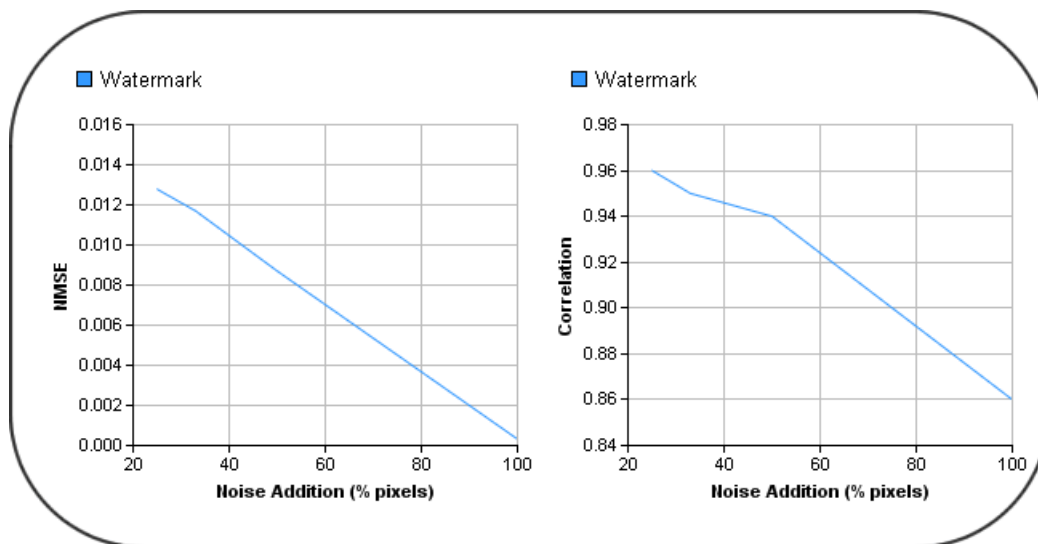


Figure 24: NMSE and Correlation values of watermark at various noise addition levels

#### 7.4.4 Resizing Attacks

To perform this attack, the watermarked image “lena” is first down-sized by a percentage and then up-sized to the original image size, losing information in the process. Table 8 shows the results. Attacks are measured by the percentage of original image size reduction. For example a value of 90 (90% of the original image) means the attack reduces the image size by 10%.







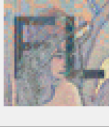
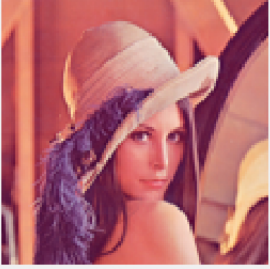

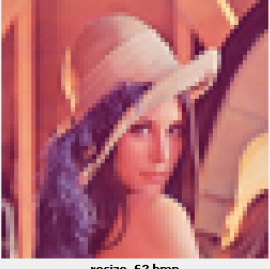
% of original image (M)	NMSE	R	Extracted Watermarks
90	0.024304	0.88	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px;">a1Áj4 ! ÈY0B , 7j0[ ä y, = gÉ</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 100px; text-align: center;">  </div> </div> <div>  <p style="text-align: center; font-size: small;">resize_462.bmp</p> </div> </div>
50	0.024494	0.79	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px;">#s* -Wk+lj 0lye ú</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 100px; text-align: center;">  </div> </div> <div>  <p style="text-align: center; font-size: small;">resize_262.bmp</p> </div> </div>
20	0.024514	0.68	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; text-align: center;">A&lt;</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 100px; text-align: center;">  </div> </div> <div>  <p style="text-align: center; font-size: small;">resize_112.bmp</p> </div> </div>
10	0.024526	0.46	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; text-align: center;">D</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 100px; text-align: center;">  </div> </div> <div>  <p style="text-align: center; font-size: small;">resize_62.bmp</p> </div> </div>

Table 8: Watermark survival results after resizing attacks

Table 8 clearly shows that the ASCII text watermark does not survive any resizing levels. However, the watermark image “FL” is still recognisable after resizing at 20% of the original image size. At this level however, the image has clearly lost its commercial value.

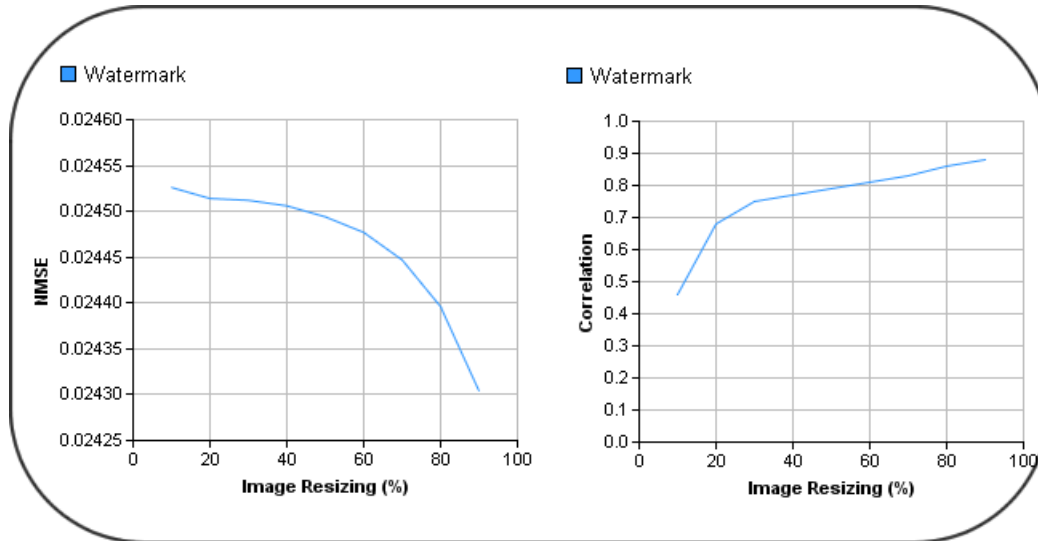


Figure 25: NMSE and Correlation values of watermark at various image resizing levels

#### 7.4.5 Rotation Attacks

Rotation clockwise or anti-clockwise, by a very small amount (0.1 degree), is usually enough to disturb the entire bit map, to such an extent that the embedded information might be lost, but the image commercial value remains. Rotation attacks (between +1.1 and -1 degree) have been performed and the results are summarised in Table 9.

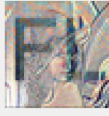

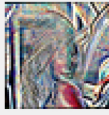

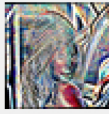





Angle (degree) (M)	NMSE	R	Extracted Watermarks
0	0.024512	0.63	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">rotation_0.0.bmp</p> </div> </div>
1	0.024095	0.08	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">rotation_1.0.bmp</p> </div> </div>
1.1	0.023955	0.03	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">COpÙ*4Ù 4 - Bi HÁÁ</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">rotation_1.1.bmp</p> </div> </div>
-0.5	0.024327	0.27	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">rotation_-0.5.bmp</p> </div> </div>
-1	0.023932	0.12	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyrégit LJITs- 2011</div> <p>Watermark Extracted</p> <div style="display: flex; align-items: center; margin-bottom: 5px;"> <span style="font-size: 2em; margin-right: 10px;">96</span>  </div> </div> <div style="flex: 1;">  <p style="text-align: center; font-size: small;">rotation_-1.0.bmp</p> </div> </div>

Table 9: Watermark survival results after rotation

Between -1 and 1 degree rotation, the ASCII text watermark survives the attack. Outside of this range of attacks, it becomes unreadable. The extracted watermark image is however not very distinct after any attack. Therefore it can be concluded that the proposed watermarking scheme is moderately resistant to rotation attacks. Clockwise rotations and anti clockwise rotations of more than one degree do not allow recovery of either of the two watermarks.

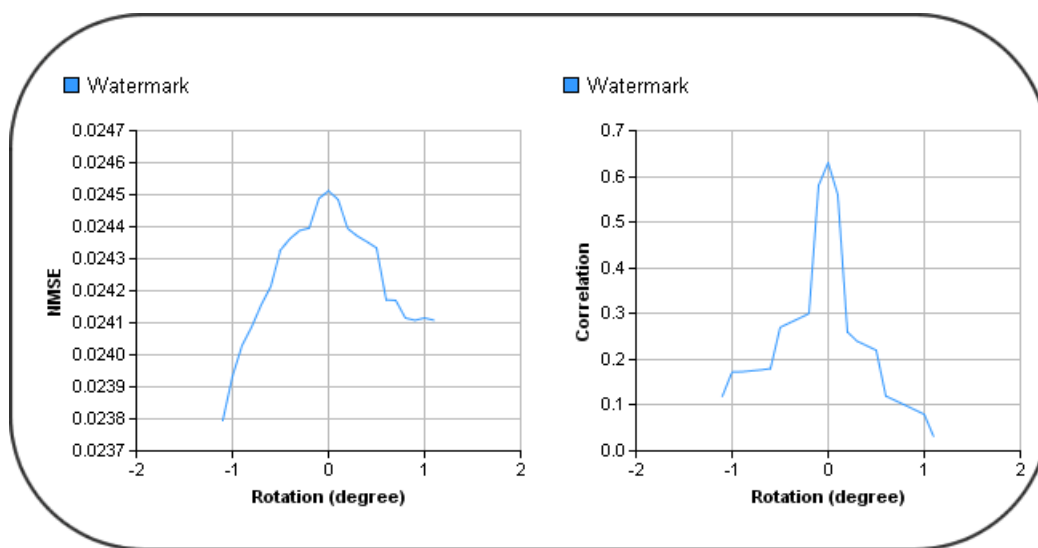


Figure 26: NMSE and Correlation values of watermark at various image rotations

#### 7.4.6 Filtering and Histogram Attacks

All these attacks were performed using Macromedia Fireworks.





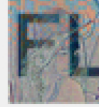

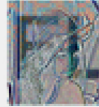

Attacks (F)	NMSE	R	Extracted Watermarks
Brightness contrast	0.0049176	0.88	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; text-align: center;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 80px; text-align: center;">  </div> </div> <div>  <p style="font-size: small; text-align: center;">lena-bright10-contrast10.bmp</p> </div> </div>
HSL	0.019349	0.75	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; text-align: center;">C p'right LÉIT - 2011</div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 80px; text-align: center;">  </div> </div> <div>  <p style="font-size: small; text-align: center;">lena-hue1-saturation1-lightness1.bmp</p> </div> </div>
Gaussian blur 1	0.024438	0.58	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; height: 30px;"></div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 80px; text-align: center;">  </div> </div> <div>  <p style="font-size: small; text-align: center;">lena-gaussian-blur1.bmp</p> </div> </div>
Gaussian blur 2	0.024519	0.37	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 150px; height: 30px;"></div> <p>Watermark Extracted</p> <div style="border: 1px solid gray; padding: 2px; width: 80px; text-align: center;">  </div> </div> <div>  <p style="font-size: small; text-align: center;">lena-gaussian-blur2.bmp</p> </div> </div>

Table 10: Watermark survival results after attacks using Fireworks

Brightness and contrast were increased slightly (by a value of 1) so that the commercial quality of the image remained. From the results in Table 10, it can be seen that both watermarks have survived the attack.

The combined Hue Saturation and Lightness (HSL) attack (increase by a factor of 1 for each) shows that only the watermark image survives.

The Gaussian blur filter attack also demonstrates that only the watermark image survives.





Attacks (F)	NMSE	R	Extracted Watermarks
Histogram	0.012551	0.37	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <pre>ÁW [ aeP~"tBH- pDMÚ ?'b ?G</pre> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p>lena-histo.bmp</p> </div> </div>
Sharpening	0.015023	0.80	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <pre>öp `ä:þí 3~7E8 [-hz- 8W + yJ#</pre> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p>lena-sharpen.bmp</p> </div> </div>

Table 11: Watermark survival results after attacks using Fireworks (continued)

The Histogram attack destroys the ASCII text watermark and the watermark image is barely identifiable.

The Sharpening attack also destroys the ASCII watermark. However the watermark image is well preserved.

### 7.4.7 Median Filter and Self Similarity Attacks

These two attacks are grouped here because they are filter attacks both performed using Stirmark. The self similarity test is performed on RGB, YUV, HSV or LAB colour space. A mask is defined to select which channel to attack. In this case, “s for spatial” was chosen when defining the test.



Attacks (S)	NMSE	R	Extracted Watermarks
Median Filter	0	0	
Self similarity	0	0	

Table 12: Watermark survival results after Stirmark attacks

As can be seen in Table 12, the watermark image is completely destroyed after both attacks. However the ASCII text watermark is preserved 100% in the case of the self similarity attack. This might be explained by the fact that this test is performed on RGB, YUV, HSV or LAB but not on YCbCr. The ASCII watermark is only preserved at 72%, in the case of the median filter attack. Therefore, it can be conclusively deduced that the proposed algorithm is resistant to Stirmark self similarity test. It is resistant

to Stirmark median filter attack but to a lesser degree.

#### **7.4.8 Cropping Attacks**

Macromedia Fireworks (F) was used to conduct these attacks. For each cropping phase, the image centre is preserved and only a percentage of the outer part of the image is removed. In doing so, the visually significant part of the image is assumed to be located towards the centre of the image. Cropping at 10% means 90% of the watermarked image remains. It is worth mentioning that the pixels of both the watermarked image and the image were realigned (synchronised) after attack.




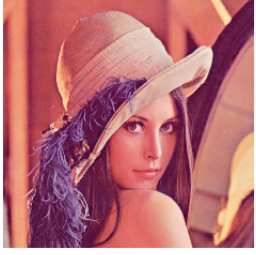
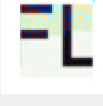
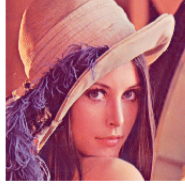
Attack level (%) (F)	NMSE	R	Extracted Watermarks
10	0.00913	0.94	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <p>Copyright LYIT - 2011</p> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>lena_crop_90.bmp</p> </div> </div>
25	0.00813	0.74	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <p>yyyyyyghuaLYÉTI 3a11Á</p> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>crop75.bmp</p> </div> </div>
50	0.0068	0.65	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <p>yyyyyyyyyyáL_úTy 7a17a</p> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p>crop50.bmp</p> </div> </div>

Table 13: Watermark survival results after cropping attacks

As confirmed in Appendix E (Figure 38), there is a strong linear correlation between the level of cropping and the amount of watermark extracted. This is explained by the fact that the watermark “FL” is spread evenly across the image, increasing its chance of survival.

### 7.4.9 Collage Attacks

The Letterkenny Institute of Technology logo was borrowed from their website (www.lyit.ie), with their agreement. It was applied on different areas of the watermarked “lena” image. The LYIT logo suited this experiment well because of its strong colour contrast with the “lena” image. Also because of its width, nearly equivalent to the original image width, the impact of the collage on the original watermark (if any) should not go un-noticed.



Region (F)	NMSE	R	Extracted Watermarks
bottom	0.01482	0.93	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p style="font-size: small; text-align: center;">collage_bottom.bmp</p> </div> </div>
center	0.01486	0.89	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p style="font-size: small; text-align: center;">collage_center.bmp</p> </div> </div>
top	0.01478	0.94	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="width: 45%;">  <p style="font-size: small; text-align: center;">collage_top.bmp</p> </div> </div>

Table 14: Watermark survival results after collage attacks

Results in Table 14 clearly show that the LYIT logo overwrites the RGB pixel values that it covers. It also partially replaces 3 characters (“ght” with “uUT”) of the ASCII watermark, when applied in the area of the image where embedding occurred (collage top). Based on these results, it is reasonable to assume that the proposed watermarking scheme will resist collage attacks unless most of the original image is covered, in which case, its original commercial value would be lost.

#### 7.4.10 Clipping Attacks


Region (F)	NMSE	R	Extracted Watermarks
top	0.011551	0.92	
bottom	0.011874	0.94	

Table 15: Watermark survival results after clipping attacks

Results in Table 15 demonstrate that clipping any area of the image will remove any watermark bits present. However, because the watermark image “FL” is spread across the entire image, it is very likely to survive clipping

attacks, unless a very large portion of the original image is clipped, in which case, its original commercial value would be lost.

#### **7.4.11 Remarks**

In most cases, either the ASCII text watermark or the watermark image “FL” survives after attack. In the case of Stirmark tests and rotation attacks, it is mainly the ASCII watermark that is recovered, while only the watermark image “FL” survives the JPEG and JPEG2000 compression attacks and most filter attacks. In the case of noise addition, both watermarks are recovered, with the exception of the 100% noise addition which destroys the ASCII watermark. These results emphasise the need for such a hybrid algorithm. Test results conducted after geometrical attacks in particular demonstrate the importance to synchronise pixels of original images with pixels of images obtained after attacks. A slight mis-alignment would destroy the watermark image “FL”.

Finally, looking at all graph results in Appendix E, it can be observed that the coefficient R is a much more useful marker of watermark image fidelity, before and after attack, than the NMSE. Although the NMSE is widely used in the literature, significant variations of its values have been noticed, depending on the type of attack performed. This demonstrates, in this particular study, that the NMSE may not be such a useful metric.

### **7.5 Security Analysis**

During this testing, after watermark extraction, a visible pattern on the watermark image itself has been noticed, as seen in Figure 27. This pattern is characterised by the presence of light blue squares and light yellow squares in the upper half of the image. This pattern is a direct result of the algorithm

applied when embedding the ASCII watermark in the YCbCr.



Figure 27: Visible pattern on the extracted watermark image

Although it is only visible when extracting the watermark image “FL”, it was necessary to find out if the tested embedded technique was easily detectable using steganography techniques. To do so, StegDetect and StegSecret have been used.

With each steganalysis tool, each original images was tested first to set a proper benchmark. Each image was then tested after the watermarks were embedded. A sensitivity of 5 for StegDetect was chosen, which is the mid-range sensitivity level. The results are listed in Table 16 and Table 17.

Image Name	Original	Watermarked
lena	jphide(**)	jphide(**)
crown	jphide(**)	jphide(**)
girl	jphide(**)	negative
plane	jphide(**)	jphide(***)
boat	skipped (false positive)	skipped (false positive)
peppers	jphide(*)	skipped (false positive)
baboon	skipped (false positive)	skipped (false positive)

Table 16: StegDetect detection results

It is interesting to note that at a sensitivity level of 5, most of the original images, which are not supposed to contain any hidden information, report a positive detection to the “jphide” method. This raises questions regarding the accuracy of StegDetect. However, apart from the image “plane”, which shows an increased detection probability that the “jphide” algorithm is used, results between the original image and the watermarked image are similar for the other images. These results demonstrate that the proposed algorithm is not detected accurately by StegDetect.

Image Name	Original	Watermarked
lena	no detection	no detection
crown	no detection	no detection
girl	no detection	no detection
plane	no detection	no detection
boat	Hiderman program detected!	Hiderman program detected!
peppers	no detection	no detection
baboon	Hiderman program detected!	Hiderman program detected!

Table 17: StegSecret detection results

Similarly to the previous experiment, the original images “boat” and “baboon” show a positive detection of the Hiderman program. Unfortunately, no useful information could be found on Hiderman’s algorithm, apart from the fact that it is used to protect privacy by hiding information in many different file formats. The result when running the “boat” original and watermarked image displays a message saying “Steganography found at marker position 15547”. The result when running the “baboon” original and watermarked image displays a message saying “Steganography found at marker position 178438”. This also raises questions regarding the detection accuracy of StegSecret. Overall, StegSecret has not managed to detect the proposed watermarking scheme.

Finally, the Stirmark “PSNR test” was used to measure the strength of the proposed watermarking method. If after performing this test, the two

watermarks can be extracted and they are positively identifiable, the strength of this watermarking technique will be proven. Using Stirmark, starting from 0, each step is incremented by 10 until 100 is reached. Table 18 shows only the significant steps.









PSNR test	NMSE	R	Extracted Watermarks
10	0	0.64	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="font-size: small; text-align: center;">w_lena_PSNR_10.bmp</p> </div> </div>
20	0	0.10	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="font-size: small; text-align: center;">w_lena_PSNR_20.bmp</p> </div> </div>
50	0	0	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="font-size: small; text-align: center;">w_lena_PSNR_50.bmp</p> </div> </div>
90	0	0	<div style="display: flex; align-items: flex-start;"> <div style="flex: 1;"> <p>Text Extracted</p> <div style="border: 1px solid gray; padding: 2px; margin-bottom: 5px;">Copyright LYIT - 2011</div> <p>Watermark Extracted</p>  </div> <div style="flex: 1;">  <p style="font-size: small; text-align: center;">w_lena_PSNR_90.bmp</p> </div> </div>

Table 18: Watermarking strength

Using Stirmark as a benchmark to measure the strength of the proposed



algorithm, it can be concluded that only the ASCII text watermark is identifiable throughout each step, showing the strength of the proposed embedding technique in the YCbCr. However the technique used to embed in the RGB is weak, judging by Stirmark standards, as the watermark image is not recoverable from step 20 onwards.

## 7.6 Capacity Analysis

Capacity is more a concern of steganographers rather than those implementing watermarking techniques. As such, it is not essential to embed a lengthy watermark in order to uniquely identify an image to its author. For example, a unique social security number associated to a country identifier and a time stamp or an image name is all that is required to provide uniqueness.

The proposed embedding technique uses two watermarks:

1. An ASCII text which is short: 20 characters in total, each character is 8 bit, therefore 160 bits long.
2. An watermark image: 50 by 50 pixels, each pixel is converted to 8 bit before embedding, therefore 20000 bits long.

Due to the algorithm used to embed in the YCbCr, limits are placed on the ASCII watermark length that can be used. If applied to an image of size 512 by 512 pixels and a block of size 19 by 19 pixels, the maximum number of watermark text characters to embed would be 84. Using smaller block size would increase watermark capacity but to the detriment of robustness, as it was observed during experimentation.

Overall the proposed algorithm provides for a reasonable size watermark, which can be a simple logo combined with a short ASCII text of up to 20

characters long, or two pieces of text of 2500 characters and 20 characters respectively.

## 7.7 Complexity Analysis

Algorithm complexity refers to processing power required to embed and extract the watermark. This is an important factor that any commercial entity would take into consideration, particularly if the volume of images to protect is significant. In order to evaluate the cpu time in seconds (computer resource) that the proposed algorithm is using, the time it takes to embed and extract the watermarks for each image has been computed, on a computer running a dual core 2GHz processor and 4 gigabytes of memory on a 32 bits Operating System running Windows 7. Results are available in Table 19 and graphed in Figure 28.

Images	Embedding time (s)	Extraction time (s)
lena	0.33	0.42
crown	0.37	0.36
girl	0.37	0.41
plane	0.31	0.36
boat	0.33	0.34
peppers	0.34	0.34
baboon	0.36	0.36

Table 19: CPU time used to embed and extract the watermarks

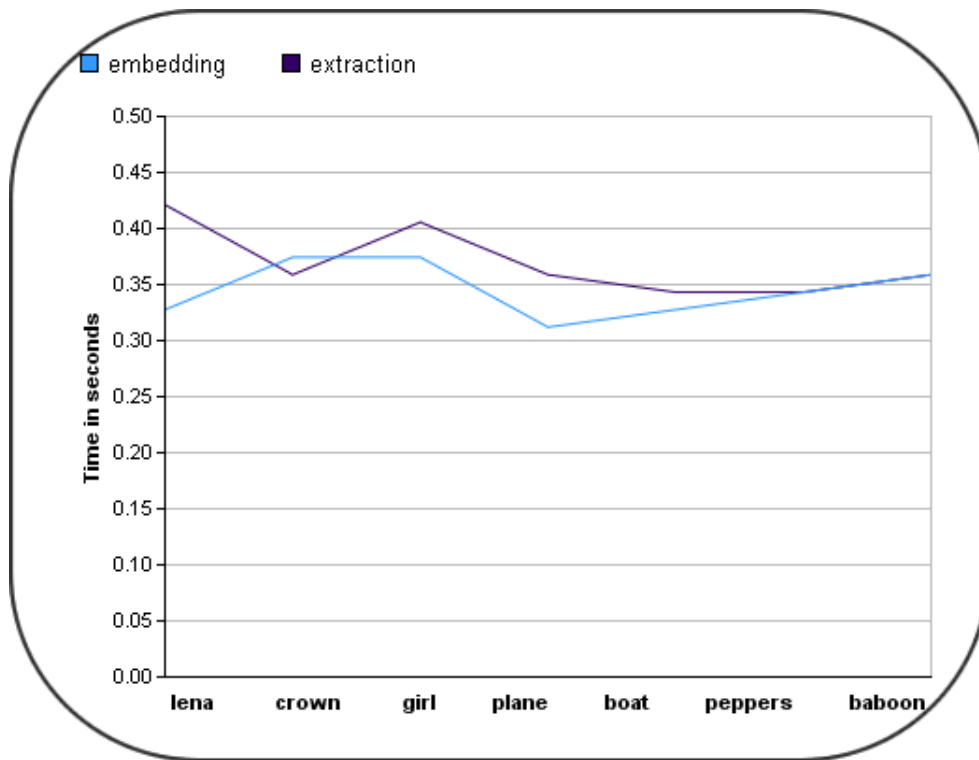


Figure 28: embedding and extraction time

Regardless of the image used, the time it takes to embed is approximately equal to the time it takes to extract the two watermarks, which is around 0.3 to 0.4 seconds. This is significantly faster (four times faster) than the 1.2 seconds it took to embed in the frequency domain using the afore mentioned DWT method.

## 7.8 Conclusion

As observed during the tests, measuring the robustness of a watermark is a difficult task to achieve: “the range of distortion is almost infinite and difficult to model or define” [83]. Having said that, a wide range of attacks have been measured, giving a very comprehensive idea on how well the proposed algorithm can withstand these attacks. Due to the hybrid technique

proposed, the proposed watermarking algorithm can resist a wide range of attacks, with the exception of a useful but limited resistance to rotation and histogram attacks (see Table 21 in Appendix E). In general, it was observed that when one watermark was destroyed, the other remained intact, so in a sense they are complementary. However it is very difficult to predict what an attacker will do and how well this algorithm can resist a combination of attacks, as the variety of such possible attacks is large.

In light of the experimentation results, this hybrid algorithm does not impair the image quality, is fairly secure and is very efficient in terms of processing power required to implement.

During the tests, no improvement (nor deterioration) in robustness was observed, by embedding first in the RGB rather than the YCbCr colour space. This suggests that embedding in the RGB and the YCbCr does not significantly affect one over the other.

## 8 CONCLUSION - RECOMMENDATIONS

### 8.1 Overall Conclusion

The aim of this research was to propose a watermarking algorithm, inspired by steganography techniques, to hide the watermark so that it is undetectable and this makes the watermark harder to remove or destroy. It was also to demonstrate an improvement in terms of robustness, over the current methods used.

Results in Chapter 7 show that not only the watermarking technique proposed is undetectable to visual inspection, but that the steganalysis tools used, failed to detect it also. This demonstrates that the hybrid algorithm is unnoticed, even when two separate watermarks are used (with one of them containing a significant number of bytes to hide).

Experiments conducted show that the hybrid watermarking method presented can withstand levels of geometric and processing attacks, up to a point where the commercial value of the images tested would be lost. In fact, no other studies sourced, have demonstrated robustness to such a large array of attacks. It is also interesting to note that despite the current trend, which favors frequency domain embedding (DWT in particular) over spatial domain embedding, it has been demonstrated that the use of spatial domain techniques can perform very well in terms of robustness, while being more efficient in terms of processing.

On the security aspect, the proposed hybrid algorithm was benchmarked against Stirmark, showing good resilience of the ASCII watermark to the PSNR test, at all levels of attacks. In other words, the stringent Stirmark test removed the watermark image “FL”, but the ASCII watermark remained.

Finally, no research experiments that were conducted on all the characteristics of a successful watermarking scheme were found, i.e. invisibility, robustness, security, capacity and complexity. Most of the publications sourced, focus on invisibility and robustness only. In order to increase the

commercial viability of any watermarking scheme, the processing power required should remain as low as possible, in situations where the number of images to protect is large. This characteristic has been clearly demonstrated in this document.

## 8.2 Recommendations for future work

Resistance to cropping, clipping and rotation in particular could be further improved by embedding the ASCII watermark towards the center of the image rather than starting from the top left. Pixels displacement is minimised at the center of an image rather than at its extremity, after image rotation.

Furthermore, the experimental results obtained show that, in most cases, only one of the two watermarks survives. One could therefore suggest that combining rotation with JPEG compression for example, might remove the two watermarks. It would be interesting to investigate alternative hybrid watermarking in the YCbCr colour space, combining spatial and frequency domain embedding, to see if this would improve robustness, in particular to geometric attacks such as rotation. The use of histogram embedding techniques in the YCbCr, would be an interesting focus point for further research.

Another issue raised with the proposed algorithm lies in the fact that it is semi-blind. Although one does not need the original image in order to extract the ASCII watermark, one needs it to extract the watermark image "FL". It is therefore imperative to securely store the original image for the proposed scheme to be successful. This adds complexity to the management of this watermarking model. Further research in this area is necessary in order to make this watermarking scheme completely blind. The use of a private key to determine the exact embedding and extraction location of the watermark might be of benefit. Perhaps the private key could even be used as the ASCII text watermark in the YCbCr.

## References

- [1] Alturki, F., Mersereau R. (2000): 'Robust oblivious digital watermarking using image transform phase modulation', *Image Processing*, vol.2, pp. 84-87
- [2] Anything3d, JPEG 2000 Compressor 1.0 Software. *Referencing*, <http://www.snapfiles.com/get/jpeg2000compressor.html> (last visited 09/05/2011)
- [3] Autrusseau F., Le Callet P. (2007): *Signal Processing*, 87, vol. 6, pp. 1363-1383.
- [4] Avcibas I., Memon N., Sankur B. (2001): 'Steganalysis using image quality metrics', *Security and Watermarking of Multimedia Contents*, San Jose, CA.
- [5] Bailey K., Curran K. (2006): 'An evaluation of image based steganography methods', *Multimedia Tools and Applications*, 30 (1) p. 55-88
- [6] Bayley H. (1966): *The Lost Language of Symbolism*, Citadel Press.
- [7] Certimark. *Referencing*, <http://www.certimark.org/> (last visited 08/06/2011)
- [8] CIE 2 (1990): 'The colourimetry of self luminous displays - a bibliography', *CIE Publication n.87*, Central Bureau of the CIE, Vienna.
- [9] Cheddad A. (2009): 'Digital image steganography: Survey and analysis of current methods', *Signal Process.* doi:10.1016/j.sigpro.2009.08.010

- [10] Cheddad A., Condell J., Curran K., Mc Kevitt P. (2009): 'A Skin Tone Detection Algorithm for an Adaptive Approach to Steganography', *Signal Processing*, 89, pp. 2465–2478.
- [11] Cho J.S. et al. (2000): 'Enhancement of robustness of image watermarks embedding into coloured image, based on WT and DCT', *Information Technology: Coding and Computing*, pp. 483-488.
- [12] Chrysochos E., Fotopoulos V., Skodras A., Xenos M. (2007): 'Reversible Image Watermarking Based on Histogram Modification', *11th Panhellenic Conference on Informatics with international participation*, vol. B, pp. 93-104.
- [13] Celik M.U. et al. (2005): 'Lossless generalised-LSB data embedding'. *IEEE Transactions on Image Processing*, pp. 253-265.
- [14] Chen W.J., Chang C.C., Ngan Le T.H. (2010): 'High payload steganography mechanism using hybrid edge detector', *Expert Systems with Applications* 37, pp. 3292-3301.
- [15] Chemak C., Bouhleb M. S., Lapayre J. C. (2007): 'A new scheme of robust image watermarking: The double watermarking algorithm'. *Proceedings of the 2007 summer computer simulation conference*, San Diego, California, pp. 1201-1208.
- [16] Chi-Man P. (2006): 'A Novel DFT-based Digital Watermarking System for Images', *Signal Processing, 2006 8th International Conference*, vol. 4, Beijing, China.
- [17] Computer Forensics, Cybercrime and Steganography Resources website. *Referencing*, <http://www.forensics.nl/tools> (last visited 15/05/2011)
- [18] Cong J. (2006): 'A novel watermarking algorithm for resistant geometric attacks using feature points matching', *Information Management & Computer Security*, 14, no. 1, pp. 75-98.



- [19] Cox, I.J. (1996): 'A Secure, Robust Watermark for Multimedia.' In *Proc. First Int. Workshop Information Hiding*. Cambridge, UK, p. 185.
- [20] Cox I.J. (1997): 'Secure Spread Spectrum Watermarking for Multimedia.' *IEEE Transactions. Image Processing*, p. 1673-1687.
- [21] Cox I.J., Miller M.L., Bloom J.A. (2001): *Digital Watermarking*. Morgan Kaufmann, San Fransisco, CA.
- [22] Cox I.J., Miller M.L., Bloom J.A. (2002): *Digital Watermarking and Fundamentals*, Morgan Kaufmann, San Fransisco, CA.
- [23] Cvejic N., Seppanen T. (2005): 'Increasing robustness of LSB audio steganography by reduced distortion LSB coding'. *Journal of Universal Computer Science*, p. 56-65.
- [24] Daly S., Watson A. B. (1993): *Digital Images and Human Vision*, The MIT Press.
- [25] Ebner M., Tischler G., Albert J. (2007): 'Integrating Color Constancy Into JPEG2000', *Image Processing, IEEE Transactions*, 16(11), pp. 2697-2706.
- [26] Emek S., Pazarci M. (2006): 'Additive vs. image dependent DWT-DCT based watermarking, in Multimedia Content Representation', *Classification and Security*, p. 98-105.
- [27] Fridrich J., Goljan M., Du R. (2001): 'Reliable Detection of LSB Steganography in Color and Gray-Scale Images', in *Magazine of IEEE Multimedia Special Issue on Security*, October-November 2001, pp. 22-28.
- [28] Fridrich J., Goljan M., Hoge D. (2002): 'Steganalysis of JPEG Images: Breaking the F5 Algorithm', *5th Information Hiding Workshop*, Noordwijkerhout, The Netherlands, 79, pp. 310-323.

- [29] Fridrich J., Goljan M., Hoge D. (2002): 'Attacking the OutGuess', in *Proc. of the ACM Workshop on Multimedia and Security*, Juan-les-Pins, France.
- [30] Fridrich J., Goljan M., Hoge D. (2003): 'New Methodology for Breaking Steganographic Techniques for JPEGs,' in *Proc. SPIE Electronic Imaging*, Santa Clara, CA, Jan 2003, pp. 143-155.
- [31] Ghannam S., Abou-Chadi F.E.Z. (2009): 'WPT versus WT for a Robust Watermarking Technique', *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, no.1.
- [32] Gilani L., Khan A., Mirza A. (2006): 'Distortion Estimation in Digital Image Watermarking using Genetic Programming'.
- [33] Gonzalez R.C., Woods R.E. (2008): *Digital Image Processing, Third Edition*, Prentice Hall.
- [34] Gonzalez R.C., Woods R.E., Eddins S.L. (2009): *Digital Image Processing using MATLAB, Second Edition*, Gatesmark Publishing (USA).
- [35] Huang C., Wu J. (2000): 'A Watermark Optimization Technique based on Genetic Algorithms'. In *Proc. SPIE Electronic Imaging*, San Jose, CA.
- [36] Huang J., Shi Y. (2000): 'Embedding Image Watermarks in DC Components', *IEEE Trans. Circuits and Systems for Video Technology*, p. 974.
- [37] Johnson N.F., Jajodia S. (1998): 'Exploring steganography: Seeing the unseen', *IEEE Computer*, vol. 31, p. 26-34.
- [38] Johnson N.F., Katzenbeisser S.C. (2000): 'A survey of steganographic techniques', in: S. Katzenbeisser and F.A.P. Petitcolas, (ed) *Information*

*hiding techniques for steganography and digital watermarking*, Norwood:  
Artech House, INC

- [39] Kamiya K., Naoe K., Mori T., Iwamura K. (2008). iih-msp, pp. 327-330, *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*.
- [40] Khan A., Mirza A.M. (2005): 'Genetic perceptual shaping: Utilizing cover image and conceivable attack information during watermarking embedding', Elsevier
- [41] Kim B.S., et al. (2003): 'Robust digital watermarking method against geometrical attacks. *Real-Time Imaging*', p. 139-149.
- [42] Kitamura I., Kanai S., Kishinami T. (2001): 'Copyright protection of vector map using digital watermarking method based on discrete Fourier transform', *Geoscience and Remote Sensing Symposium*, vol.3, pp.1191-1193.
- [43] Langelaar G.C., Setyawan I., Lagendijk R.L. (2000): 'Watermarking digital image and video data. A state of the art overview', *IEEE Signal Processing Magazine*, vol. 17 , no 5, pp. 20-46.
- [44] Langelaar G.C., Lagendijk R.L. (2001): 'Optimal differential energy watermarking of DCT encoded images and video', *Image Processing, IEEE Transactions*, p. 148-158.
- [45] Li X., Wang J. (2007): 'A steganographic method based upon JPEG and particle swarm optimization algorithm', *Information Sciences*, 177(15) pp. 3099-31091.
- [46] Li Z., Chen X., Pan X., Zeng X. (2009): 'Lossless data hiding scheme based on adjacent pixel difference', in *Proceedings of the international conference on computer engineering and technology*, pp. 588-592

- [47] Lin E., Delp E. (2004): 'Spatial Synchronisation Using Watermark Key Structure.' *In Proc SPIE Conf. Security, Steganography and Watermarking of Multimedia Contents*. San Jose, CA.
- [48] Liu K.C., Chou C.H. (2007): *International Journal of Computer Science and Network Security*, vol.7, no.7.
- [49] Lou D.C., Shieh J.M., Tso H.X. (2006): 'A robust buyer-seller watermarking scheme based on DWT', *International Journal of Pattern Recognition and Artificial Intelligence*, p. 79-90.
- [50] Lou D.C., Wu N. I., Wang C. M., Lin Z.H., Tsai C.S. (2010): 'A novel adaptive steganography based on local complexity and human vision sensitivity', *Journal of Systems and Software*, pp. 1236-1248.
- [51] Lu Z., Sun S. (2000): 'Digital Image Watermarking Technique Based on Vector Quantization.' *IEEE Electronic Letters*, p. 303.
- [52] Lukac R., Plataniotis K.N. (eds.) (2006): *Color Image Processing: Methods and Applications*. Boca Raton, FL, CRC Press / Taylor & Francis.
- [53] Ni J.Q., et al. (2005): 'A RST-invariant robust DWT-HMM watermarking algorithm incorporating Zernike moments and template', in *Knowledge-Based Intelligent Information and Engineering Systems*, Pt 1, Proceedings, p. 1233-1239.
- [54] Nikolaidisa N., Pitas I. (1998): 'A Method for Watermark Casting on Digital Images'. *IEEE Trans. Circuits and Systems for Video Technology*, p. 775.
- [55] Mehul R., Priti R. (2003): 'Discrete Wavelet Transform Based Multiple Watermarking Scheme,' *Proceedings of IEEE Region 10 Technical Conference on Convergent Technologies for the Asia-Pacific*, Bangalore, India.

- [56] Miller, Doerr, Cox. (2004): 'Applying Informed Coding and Embedding to Design a Robust High-Capacity Watermark.', *IEEE Trans. Image Processing*, p. 792.
- [57] Mohanty S. P., Guturu P. (2008): 'A Novel Invisible Color Image Watermarking Scheme using Image Adaptive Watermark Creation and Robust Insertion-Extraction', *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 5, no. 2, art. 12, pp. 1-24.
- [58] Muñoz A.: 'StegSecret'. *Referencing*, <http://stegsecret.sourceforge.net/> (last visited 04/05/2011)
- [59] Petitcolas F.A.P., Anderson R.J., Kuhn M.G. (1998): 'Attacks on copyright marking systems', in David Aucsmith (Ed), *Information Hiding, Second International Workshop, IH'98*, Portland, Oregon, U.S.A., Proceedings, LNCS 1525, Springer-Verlag, ISBN 3-540-65386-4, pp. 219-239.
- [60] Petitcolas F.A.P., Anderson R.J. (1999): 'Evaluation of copyright marking systems', in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, ICMS '99, vol. I, Florence, Italy, June 1999, pp. 574-579.
- [61] Petitcolas F.A.P. (2000): 'Watermarking schemes evaluation', *I.E.E.E. Signal Processing*, vol. 17, no. 5, pp. 58-64.
- [62] Petitcolas F.A.P.: Image database. *Referencing*, [http://www.petitcolas.net/fabien/watermarking/image\\_database/index.html](http://www.petitcolas.net/fabien/watermarking/image_database/index.html) (last visited 04/05/2011)
- [63] Piva A., et al. (1997): 'DCT-based Watermark Recovering without Resorting to the Uncorrupted Original Image', *Proc Int. Conf. Image Processing*, vol 1, pp 520-523.

- [64] Potdar V.M., Han S., Chang E. (2005): 'Fingerprinted secret sharing steganography for robustness against image cropping attacks', in: *Proceeding of IEEE 3rd International Conference on Industrial Informatics*, Perth, Australia, 10-12 August 2005, pp. 717-724.
- [65] Provos N. "Stegdetect". *Referencing*, <http://www.outguess.org/detection.php> (last visited 17/03/2011)
- [66] Provos N. (2001): 'Defending Against Statistical Steganalysis', in *10th USENIX Security Symposium*, Washington, DC.
- [67] Qi F., Wu J., Shi G. (2009): 'Extracting regions of attention by imitating the human visual system', *icassp*, pp.1905-1908, *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*.
- [68] Qiang C., Huang T.S. (2000): 'Optimum detection and decoding of multiplicative watermarks'. *IEE Transactions on Image Processing*, p.1123-1129.
- [69] Rao K.R., Yip P. (1990): 'Discrete Cosine Transform: Algorithms, Advantages, Applications', *Academic Press*, Boston.
- [70] Sal Diaz D., Grana Romay M. (2005): 'Introducing a Watermarking with Multi-Objective Genetic Algorithm', *GECCO '05 Proceedings of the 2005 conference on Genetic and evolutionary computation*.
- [71] Schlauweg M., Profrock D., Zeibich B., Muller E. (2008): 'Self-Synchronizing Robust Textel Watermarking in Gaussian Scale-Space', *Proceedings of the 10th ACM workshop on Multimedia and security, MM&Sec 2008*, Oxford, UK, ACM 2008.
- [72] Shih F.Y., Wu S.Y. (2003): 'Combinational Image Watermarking in the Spatial and Frequency Domains', *Pattern Recognition* 36, p. 969

- [73] Shih F.Y. (2008): *Digital Watermarking and Steganography - Fundamentals and Techniques*, CRC Press.
- [74] Simmons G. J. (1984): 'The prisoners' problem and the subliminal channel', in *Advances in Cryptology: Proceedings of Crypto 83* (D. Chaum, ed.), pp. 51-67.
- [75] Solachidis V., Tefas A., Nikolaidis N., Tsekeridou S., Nikolaidis A. Pitas, I. (2001): 'A benchmarking protocol for watermarking methods', *IEEE Int. Conf. on Image Processing (ICIP'01)*, pp. 1023-1026.
- [76] Stollnitz E.J., Deroose T.D., Salesin D.H. (1996): *Wavelet for Computer Graphics*, Morgan Kaufmann Publishers, Inc.
- [77] TERA Consultants, 'Building a Digital Economy: The importance of Saving Jobs in the EU's Creative Industries', March 2010. *Referencing*, [http://www.teraconsultants.fr/assets/publications/PDF/2010-Mars-Etude\\_Piratage\\_TERA\\_full\\_report-En.pdf](http://www.teraconsultants.fr/assets/publications/PDF/2010-Mars-Etude_Piratage_TERA_full_report-En.pdf). (last visited 17/03/2011)
- [78] Tsai M. J., Yu K., Chen Y. Z. (2000): 'Joint Wavelet and Spatial Transformation for Digital Watermarking'. *IEEE Trans. Consumer Electronics*, p 241.
- [79] Tsai P., Hu Y.C., Yeh H.L. (2009): 'Reversible image hiding scheme using predictive coding and histogram shifting', *Signal Processing*, 89(6), pp. 1129-1143.
- [80] USC-SIPI Image Database. *Referencing*, <http://sipi.usc.edu/database/database.php> (last visited 04/05/2011)
- [81] Voloshynovskiy S., Pereira S., Iquise V., Pun T. (2001): 'Attack modelling: towards a second generation watermarking benchmark', *Signal Processing*, 81, no. 6, p. 1177-1214.

- [82] Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P. (2004): 'Image quality assessment: From error visibility to structural similarity', *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612.
- [83] Wayner P. (2002): *Disappearing Cryptography, Information Hiding: Steganography & Watermarking, Second Edition*, Morgan Kaufmann Publishers. Copyright 2002 Elsevier Science (USA).
- [84] Wikipedia definition of discrete wavelet transform. *Referencing*, [http://en.wikipedia.org/wiki/Discrete\\_wavelet\\_transform](http://en.wikipedia.org/wiki/Discrete_wavelet_transform) (last visited 17/03/2011)
- [85] Wikipedia image file formats. *Referencing*, [http://en.wikipedia.org/wiki/Image\\_file\\_formats](http://en.wikipedia.org/wiki/Image_file_formats). (last visited 17/03/2011)
- [86] Wikipedia definition of Digital Watermarking. *Referencing*, [http://en.wikipedia.org/wiki/Digital\\_watermarking](http://en.wikipedia.org/wiki/Digital_watermarking). (last visited 17/03/2011)
- [87] Westfeld, Pfitzmann A. (2000): 'Attacks on Steganographic Systems,' *Lecture Notes in Computer Science*, vol. 1768, Springer-Verlag, Berlin, 2000, pp. 61-75.
- [88] Zheng D., Liu Y., Zhao J., El Saddik A. (2007): 'A survey of RST invariant image watermarking algorithms'. *ACM Comput. Surv.* 39, 2, Article 5 (June 2007).
- [89] Zhiwei K., Jing L., Yigang H. (2007): 'Steganography based on wavelet transform and modulus function'. *Journal of Systems Engineering and Electronics*, Vol. 18, No. 3, pp. 628-632.
- [90] Zhu X., Gao Y., Zhu Y. (2006): 'Image-adaptive Watermarking Based on Perceptually Shaping Watermark Blockwise'. *ASIA CCS'06*, Taipei, Taiwan.



- [91] Yi-Chong Z., Soo-Chang P., Jian-Jiun D. (2006): 'DCT-Based Image Protection using Dual-Domain Bi-Watermarking Algorithm', *Image Processing*, pp. 2581-2584
  
- [92] Zhao, Chen, Liu (2004): 'A Chaotic-Based Robust Wavelet-Domain Watermarking Algorithm.' *Chaos, Solitons and Fractals* , p. 22-47.
  
- [93] Zheng S., Zhu Y., Wang X. (2008): 'A New RST-Invariant Watermarking Scheme Based on Texture Features', *e-Forensics*, Adelaide, Australia, p. 21-23.

# Appendix A

Algorithm	Description	Advantages	Problems
Fourier Melin Transform	Use spread spectrum technique for watermark generation, embedding and detection	RST Invariance	Difficult to implement
Phase Correlation and Log Polar Mapping	Watermark is embedded into the LPM domain	RST Invariance, moderate resistance to scaling and compression	Difficulty in embedding the message caused by the LPM and inverse LPM. Original image is needed.
Phase only filtering and Log Polar Mapping	Rotation and scaling induce a shift in LPM domain. A filter is used to compute the possible shift, to reestablish the original image shape size and position.	very robust to RS, good JPEG compression resistance and noise addition.	Original image is needed to deduce the template.
One dimensional Projection and LPM	Based on FM Transform with some properties of the one dimensional projection.	Good RS and compression and noise addition resistance	It is a multiplicative method rather than an additive method, so the optimal coefficients need to be found.

Table 20: Comparison of RST (rotation, scaling and translation) Invariant Image Watermarking Algorithms[88]

<b>Algorithm</b>	<b>Description</b>	<b>Advantages</b>	<b>Problems</b>
Radon Transform	Retrieve geometrical transformations in order to resynchronise the image to its original form	good resistance to RS, JPEG compression and noise addition	Inaccuracy of the transformed coefficient retrieval corrected by extra computation
Template	Templates are located in the middle frequency spectrum	good resistance to RS. Average JPEG compression resistance. Poor noise addition resistance	Problematic accuracy detection of the watermark.
Salient Feature	Local image feature detection used to extract robust feature points.	Good RS and JPEG compression and noise addition resistance.	Limited embedding capacity. Mediocre performance due to algorithm complexity.
Image decomposition	Watermark is embedded into the untransformed domain while location parameters embedding is determined by FM transform	Good resistance to RS, JPEG compression and noise addition	more susceptible to geometric distortions.
Stochastic Analysis	Bispectrum feature vector is used to embed the watermark	Invariant to translation and scaling.	poor resistance to cropping.

Table 21: Comparison of RST (rotation, scaling and translation) Invariant Image Watermarking Algorithms[88] (continued)

# Appendix B

Colour is no longer interpreted as an extension of grey scale. It is considered as a key element for a number of image processing systems. In particular, colour space transforms have played a central role in coding, compression and transmission applications. Colour also plays a major role in pattern recognition and digital multimedia, where colour-based features and colour segmentations have been proved effective in indexing and retrieving image content. The aim of this chapter is to explore in more details how to use colour spaces for the benefit of watermarking.

## Colour in the context of the HVS

To understand precisely the concept of colour information, it is worth reviewing the fundamental properties of the HVS [24]. Colour is defined as an experience in human perception. In physics terms, a colour is the result of an observed light on the retina of the eye. The human eye sees colour by means of cones in the retina. There are three types of cones sensitive to wavelengths that approximately correspond to red, green and blue lights. Together with information from rod cells (which are not sensitive to colour) the cone information is encoded and sent to higher brain centres along the optic nerve. A human eye can recognise colours in the range of 400 nanometres (violet) to 700 nanometres (red) and can adapt to a large variation of illumination levels (see Figure 29).

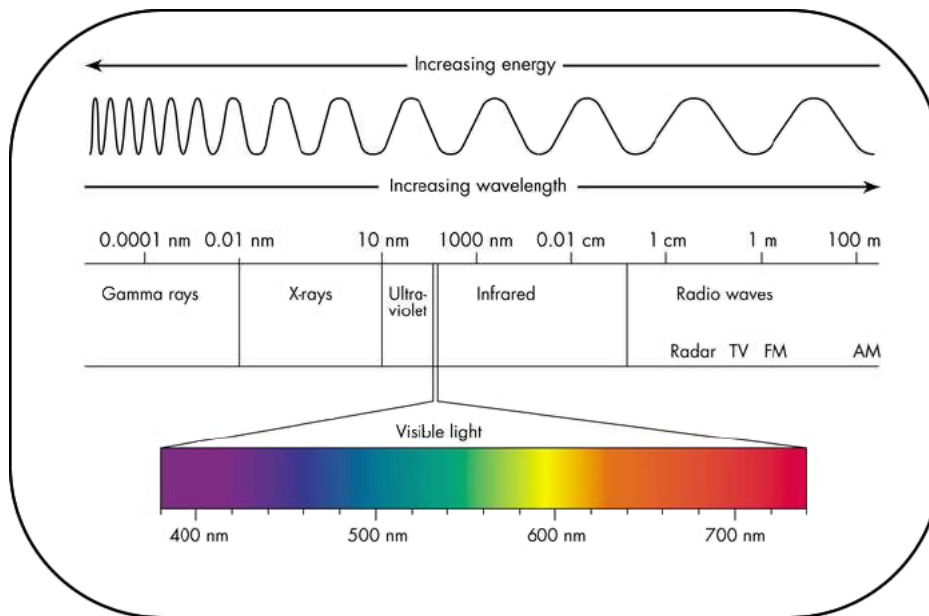


Figure 29: Human Visual Spectrum [33]

The vision system perceives this range of light wavelengths as a smoothly varying rainbow of colours, which is called the visual spectrum. One of the biggest problems in colour image processing is to find the appropriate colour space for the problem being addressed. While the application context often defines the original space (such as RGB for computer images or YCbCr for television video) the insertion space has to be discussed according to the expected properties of the watermark.

## Colour Spaces

In this section, the characteristics of human visual perception to the difference between colours in different colour spaces are discussed.

A colour space is a method by which one can specify, create and visualise colour. Humans define a colour by its attributes of brightness, hue and colourfulness. A computer defines a colour in terms of the excitations of red, green and blue phosphors on the CRT faceplate. A printing press defines a

colour in terms of the reflectance and absorbance of cyan, magenta, yellow and black inks on the paper.

Imagining that each of the three attributes used to describe a colour are axes in a three dimensional space then this defines a colour space. So, a colour space is a mathematical representation of human perception.

Since the HVS has a limited sensitivity in perceiving visual information, it is well believed that there exists quite an amount of perceptual redundancy in colour images. The perceptual redundancy of a particular colour is represented by the perceptually indistinguishable colour region in which each colour cannot be distinguishable; that is, the perceptual colour difference in the perceptually indistinguishable colour region is close to zero. Through making the embedded watermarks part of the perceptual redundancy in colour images, watermark insertion can be achieved with transparency.

The perceptual redundancy inherent in colour images of different colour spaces is estimated based on the numerical colour difference in the uniform colour space. The extent of the perceptual redundancy of a colour varies with the colour space where it is represented. With the varying volume of perceptually indistinguishable colour region in different colour spaces, a watermarking scheme based on the perceptual redundancy is implemented and the corresponding results of robustness of the watermarking scheme are compared.

The three most popular colour models are RGB (used mostly in computer graphics), YIQ, YUV or YCbCr (used in video systems) and CMYK (used in colour printing). But other colour models are presented for completion. Mathematical formulae presented in the following sections were taken from [8].

The RGB and YCbCr colour spaces were developed in Chapter 5. In the following sections you will find other colour spaces for completeness.

## CMY(K) Colour Space

The CMY colour model is used in colour printing and refers to the three inks used (secondary colours of light): cyan, magenta, yellow. On printer devices, a component of black is added to the CMY, and the second colour space is then called CMYK. The black component is actually used because cyan, magenta, and yellow set up to the maximum should produce a black colour. It is said to be subtractive because inks “subtract” brightness from white. The conversion from RGB to CMY is performed using the simple operation:

$$C = 1 - R$$

$$M = 1 - G$$

$$Y = 1 - B$$

This equation shows that light reflected from a surface containing pure cyan does not reflect red, pure magenta does not reflect green and pure yellow does not reflect blue.

The conversion from CMY back to RGB can also be obtained simply by subtracting the individual CMY values from 1. In practice, this later conversion is of little interest, once the ink is on the paper, unless the paper is scanned to be digitized.

## HSI, HSL, HSV and related colour spaces

The representation of the colours in the RGB and CMY(K) colour spaces are designed for specific devices. But for a human observer, they are not useful definitions. For user interfaces a more intuitive colour space, designed for the way one actually thinks about colour is preferred. Such a colour space is HSI: Hue, Saturation and Intensity, which can be thought of as a RGB cube tipped up onto one corner (see Figure 30).

The line from RGB=min to RGB=max becomes vertical and is the intensity axis ( $I$ ). The position of a point on the circumference of a circle around this axis is the hue ( $H$ ) and the saturation ( $S$ ) is the radius from the central

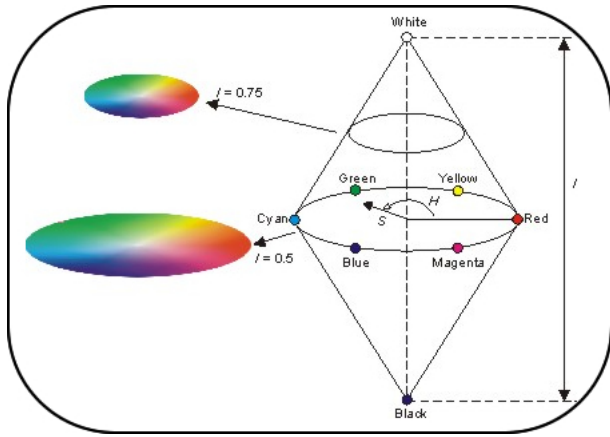


Figure 30: HSI Colour Space [34]

intensity axis to the colour.

The transforms are given below:

$$\text{Hue} = (\alpha - \arctan((\text{Red} - \text{intensity}) * (3^{0.5}) / (\text{Green} - \text{Blue}))) / (2 * \pi)$$

with:

$$\alpha = \pi/2 \text{ if } \text{Green} > \text{Blue}$$

$$\alpha = 3 * \pi/2 \text{ if } \text{Green} < \text{Blue}$$

$$\text{Hue} = 1 \text{ if } \text{Green} = \text{Blue}$$

$$\text{Saturation} = (\text{Red}^2 + \text{Green}^2 + \text{Blue}^2 - \text{Red} * \text{Green} - \text{Red} * \text{Blue} - \text{Blue} * \text{Green})^{0.5}$$

$$\text{Intensity} = (\text{Red} + \text{Green} + \text{Blue}) / 3$$

Note that Intensity must be computed before Hue. If not, it must be assumed that  $\text{Hue} = (\alpha - \arctan((2 * \text{Red} - \text{Green} - \text{Blue}) / ((\text{Green} - \text{Blue}) * (3^{0.5})))) / (2 * \pi)$ .

H, S, L, R, G, and B are within the range of 0 to 1.

Actually, there are many variations on HSI, e.g. HSL, HSV, HCl (chroma / colourfulness), HVC, TSD (hue saturation and darkness). But they all do basically the same thing. The major disadvantage of these models are the conversion complexity which is mainly because the hue is expressed as an angle.



## CIE XYZ Colour Space

The CIE has defined a human "Standard Observer", based on measurements of the colour-matching abilities of the average human eye. Their recommendations are as follow:

- Brightness: The attribute of a visual sensation according to which an area appears to exhibit more or less light.
- Hue: The attribute of a visual sensation according to which an area appears to be similar to one, or to proportions of two, of the perceived colours red, yellow, green and blue.
- Colourfulness: The attribute of a visual sensation according to which an area appears to exhibit more or less of its hue. One can go from a sky blue to a deep blue by changing this attribute.

Using data from measurements made in 1931, a system of three primaries, XYZ, was developed in which all visible colours can be represented using only positive values of X, Y and Z. The Y primary is identical to Luminance, X and Z give colouring information. This forms the basis of the CIE 1931 XYZ colour space, which is fundamental to all colourimetry. Values are normally assumed to lie in the range 0 to 1. Colours are rarely specified in XYZ terms, it is far more common to use chromaticity coordinates, which are independent of the Luminance (Y). The main advantage of CIE XYZ, and any colour space or colour definition based on it, is that it is completely device independent. The main disadvantage with CIE-based spaces is the complexity of implementing them, in addition some are not user intuitive.

## CIE Luv and CIE Lab

In 1976, the CIE defined two new colour spaces to get more uniform and accurate models. The first of these two colour spaces is the CIE Luv whose components are  $L^*$ ,  $u^*$  and  $v^*$ .  $L^*$  component defines the luminance, and  $u^*$ ,  $v^*$  define chrominancy. CIE Luv is mostly used in calculation of small colours or colour differences, especially with additive colours. The CIE Luv colour space is defined from CIE XYZ.

The second, CIE Lab is proposed as a new incorporated colour space in TIFF specifications, where three components are used:  $L^*$  is the luminance,  $a^*$  and  $b^*$  are respectively red/blue and yellow/blue chrominancies. This colour space is also defined with regard to the CIE XYZ colour spaces where,

$$L = 116 \cdot ((Y/Y_n)^{1/3}) - 16 \text{ if } Y/Y_n > 0.008856$$

$$L = 903.3 \cdot Y/Y_n \text{ if } Y/Y_n \leq 0.008856$$

$$a = 500 \cdot (f(X/X_n) - f(Y/Y_n))$$

$$b = 200 \cdot (f(Y/Y_n) - f(Z/Z_n))$$

where

$$f(t) = t^{1/3} \text{ with } Y/Y_n > 0.008856$$

$$f(t) = 7.787 \cdot t + 16/116 \text{ with } Y/Y_n \leq 0.008856$$

## LCH and CIE LSH

CIELab and CIELuv both have a disadvantage if applied to user interfaces, they are unintuitive to use. To solve this the CIE definitions can be used for chroma (c), Hue angle (h) and saturation (s). Hue, chroma and saturation can be derived from CIELuv, and Hue and chroma - but not saturation - can be derived from CIELab (this is because CIELab has no associated chromaticity diagram and so no correlation of saturation is possible).

To distinguish between LCH derived from CIELuv and CIELab the values of Hue, H, and Chroma, C, are given the subscripts “uv” if from CIELuv and ab if from CIELab.

For example LCH derived from CIELab is computed as follows:

$$L = L^*$$

$$C = (a^{*2} + b^{*2})^{0.5}$$

$$H = 0 \text{ if } a = 0$$

$$H = (\arctan((b^*)/(a^*)) + k\pi/2) / (2\pi) \text{ if } a \neq 0$$

and

$$(k = 0 \text{ if } a^* \geq 0 \text{ and } b^* \geq 0) \text{ or } (k = 1 \text{ if } a^* > 0 \text{ and } b^* < 0) \text{ or } (k = 2 \text{ if } a^* < 0 \text{ and } b^* < 0) \text{ or } (k = 3 \text{ if } a^* < 0 \text{ and } b^* > 0)$$

LCH derived from CIELuv is computed as follows:

$$L = L^*$$

$$C = (u^{*2} + v^{*2})^{0.5} \text{ or } C = L_s$$

$$H = \arctan[(v^*)/(u^*)]$$

$$H=0 \text{ if } u=0$$

$$H=(\arctan((v^*)/(u^*)) + k\pi/2) / (2\pi) \text{ if } u \neq 0$$

and

$$(k = 0 \text{ if } u^* \geq 0 \text{ and } v^* \geq 0) \text{ or } (k = 1 \text{ if } u^* > 0 \text{ and } v^* < 0) \text{ or } (k = 2 \text{ if } u^* < 0 \text{ and } v^* < 0) \text{ or } (k = 3 \text{ if } u^* < 0 \text{ and } v^* > 0)$$

# Appendix C

DWT Code implementation using MATLAB, developed as part of the original investigation.

## DWT Embedding

```
% start of code used to evaluate DWT watermarking
clear all;clc;
% save start time
start_time=cputime;
% read in lena image
file_name='lena_grey.bmp';
cover_object=double(imread(file_name));
% determine size of watermarked image
Mc=size(cover_object,1);
Nc=size(cover_object,2);
% read in the watermark image and reshape it into a vector
file_name='copyright.bmp';
message=double(imread(file_name));
Mm=size(message,1);
Nm=size(message,2);
message_vector=round(reshape(message,Mm*Nm,1)./256);
[cA1,cH1,cV1,cD1] = dwt2(cover_object,'haar');
k=15; %gain factor

% add pn sequences to H1 and V1 components when watermark pixel is black
for kk=1:length(message_vector)

    if (message(kk) == 0)
        if(cH1(kk) >= 0)
            cH1(kk)=cH1(kk)+k;
        end
        if(cV1(kk) >= 0)
            cV1(kk)=cV1(kk)+k;
        end
    end
end

end
% perform IDWT
watermarked_image = idwt2(cA1,cH1,cV1,cD1,'haar',[Mc,Nc]);
```

```
% convert back to uint8
watermarked_image_uint8=uint8(watermarked_image);
% write watermarked Image to file
imwrite(watermarked_image_uint8,'dwt_watermarked.bmp','bmp');
% display processing time
elapsed_time=cputime-start_time;
disp(elapsed_time)
% calculate the PSNR
psnr=psnr(cover_object,watermarked_image_uint8);
disp(psnr)
% display watermarked image
figure(1) imshow(watermarked_image_uint8,[]) title('Watermarked Image')
```

## DWT Extraction

```
% start of code used to evaluate DWT watermarking
clear all;clc;
% save start time
start_time=cputime;
% read in the watermarked
file_name='dwt_watermarked.bmp';
watermarked_image=double(imread(file_name));
% code used for JPEG compression
%compression = 90;
%file_compressed = sprintf('compression_%d_percent.jpg', compression);
%imwrite( imread(file_name), file_compressed, 'Quality', compression);
%watermarked_image=double(imread(file_compressed));
% end of compression code
% determine size of watermarked image
Mw=size(watermarked_image,1);
Nw=size(watermarked_image,2);
% read in original watermark to get size
file_name='copyright.bmp';
orig_watermark=double(imread(file_name));
% determine size of original watermark
Mo=size(orig_watermark,1);
No=size(orig_watermark,2);
% initialize message to all ones
message_vector=ones(1,Mo*No);
[cA1,cH1,cV1,cD1] = dwt2(watermarked_image,'haar');
k=15; %gain factor
for kk=1:length(message_vector)
    if(cH1(kk) - k >= 0)
        message_vector(kk)=0;
    end
    if(cV1(kk)- k >= 0)
        message_vector(kk)=0;
    end
end
% reshape the message vector and display recovered watermark.
figure(2)
message=reshape(message_vector,Mo,No);
imshow(message,[])
title('Recovered Watermark')
% display processing time
```

```
elapsed_time=cputime-start_time;  
disp(elapsed_time)
```

# Appendix D

Hybrid Watermarking code implementation using MATLAB.

## Graphical User Interface (GUI)

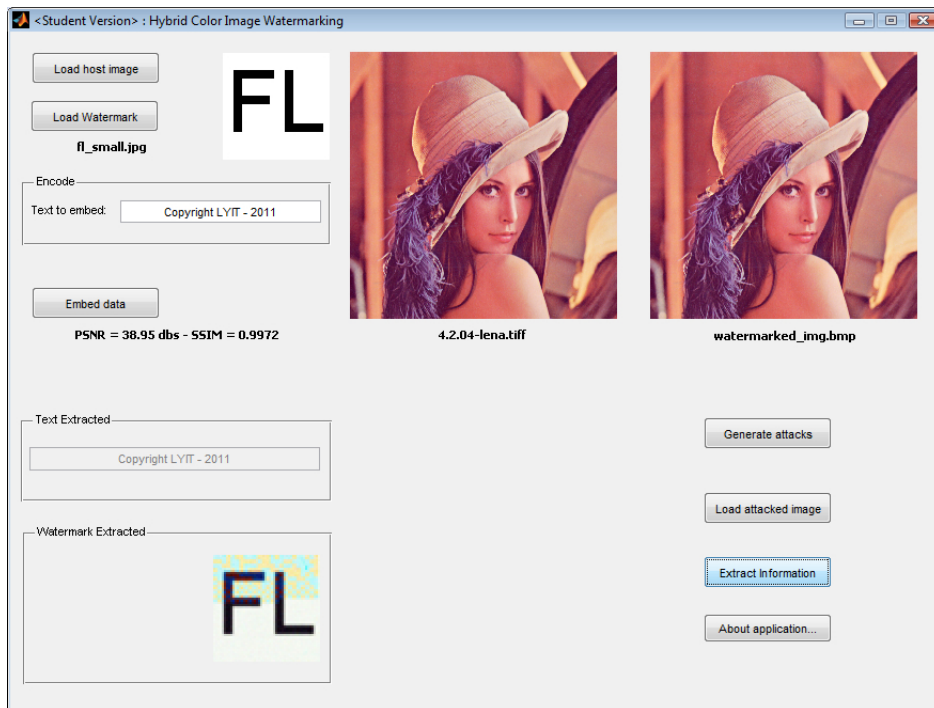


Figure 31: MATLAB GUI



## GUI Code

```
function varargout = mainform(varargin)
% MAINFORM M-file for mainform.fig is our GUI entry point
%     MAINFORM, by itself, creates a new MAINFORM or raises the existing
%     singleton*.
%
%     H = MAINFORM returns the handle to a new MAINFORM or the handle to
%     the existing singleton*.
%
%     MAINFORM('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in MAINFORM.M with the given input arguments.
%
%     MAINFORM('Property','Value',...) creates a new MAINFORM or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before mainform_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to mainform_OpeningFcn via varargin.
%
% OUTPUT:
%     GUI enabling us to insert / extract watermarks and test against
%     attacks
%
% AUTHOR:
%     Frederic Lusson
%
% *****
% Begin initialization code
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @mainform_OpeningFcn, ...
                  'gui_OutputFcn',  @mainform_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```

        gui_mainfcn(gui_State, varargin{:});

end
% End initialization code
% --- Executes just before mainform is made visible.
function mainform_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainform (see VARARGIN)
% Choose default command line output for mainform
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% Prepare space for image display
axes(handles.pic_original); axis off;
axes(handles.pic_combined); axis off;
axes(handles.pic_watermark); axis off;
axes(handles.pic_attacked); axis off;
axes(handles.pic_w_extracted); axis off;
% Clear main Matlab windows
clc;
% Clear all variables
clear all;
% --- Outputs from this function are returned to the command line.
function varargout = mainform_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in btnLoadHostImg.
function btnLoadHostImg_Callback(hObject, eventdata, handles)
global original_image;
% hObject    handle to btnLoadHostImg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
% Invoke file selection window
[filename,pathname]=uigetfile({'*.tiff'; '*.jpg'}, 'Original Image');
% Load original image
original_image = imread(filename);
image(original_image, 'parent', handles.pic_original);
axes(handles.pic_original); axis off;

```

```

% Make button and text area active
set(handles.txtCoder,'Enable','on');
set(handles.btnEncode,'Enable','on');
% Display filename under image
set(handles.lblOutputImage,'String',filename);
% clear decoded string box and also text to encode
set(handles.txtCoder, 'String', '');
set(handles.txtDecoder, 'String', '');
% --- Executes on button press in btnLoadWatermark.
function btnLoadWatermark_Callback(hObject, eventdata, handles)
global watermark_image;
% hObject    handle to btnLoadWatermark (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
global watermark;
% Invoke file selection window
[filename,pathname]=uigetfile({'*.jpg'},'Watermark');
% Load watermark image
watermark_image = imread(filename);
watermark = watermark_image;
image(watermark_image,'parent',handles.pic_watermark);
axes(handles.pic_watermark); axis off;
% Display filename under image
set(handles.lblWatermark,'String',filename);
function txtCoder_Callback(hObject, eventdata, handles)
% hObject    handle to txtCoder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of txtCoder as text
%        str2double(get(hObject,'String')) returns contents of txtCoder
%        as a double
% --- Executes during object creation, after setting all properties.
function txtCoder_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtCoder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
%            called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), ...
get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end

```

```

% --- Executes on button press in btnEncode.
function btnEncode_Callback(hObject, eventdata, handles)
global watermark_image original_image combined_image;
global original_image_y_size original_image_x_size;
% hObject    handle to btnEncode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
start_time = cputime;
original_watermark_image = watermark_image;
image_temporary = original_image; % that's the original image
% read the input from the text box
ascii_codes = unicode2native(get(handles.txtCoder,'String'), 'ISO-8859-1');
% FIRST PART: the image watermark embedding
image_temporary = insertWatermark(image_temporary, original_watermark_image);
% SECOND PART: hide the ascii text into the YCbCr colour space of image
% encodeInYCbCr is built-in function
image_temporary = encodeInYCbCr(ascii_codes, image_temporary);
% display time taken to embed
end_time = cputime - start_time;
disp(['embedding time: ' num2str(end_time)])
image_output = image_temporary;
% Save result in bmp file
imwrite(image_output,'watermarked_img.bmp','bmp'); set(handles.lblCombined,...
    'String','watermarked_img.bmp');
% Load watermarked image: combination of both original + watermark
combined_image = image_output;
image(combined_image,'parent',handles.pic_combined);
axes(handles.pic_combined); axis off;
% Display filename under image
decibels = PSNR(original_image, combined_image);
% Compute SSIM
ssim_val = SSIM(original_image, combined_image);
set(handles.lblPSNR,'String',sprintf('PSNR = %5.2f dbs - SSIM = %4.4f', ...
decibels,ssim_val));
% enable the decode button
set(handles.btnDecode,'Enable','on');
% --- Executes on button press in btnDecode.
function btnDecode_Callback(hObject, eventdata, handles)
% hObject    handle to btnDecode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global combined_image original_image watermark;
image_temporary = combined_image; % this is the original + watermark
start_time = cputime;
% Display decoded text in text area

```

```

% decodeInYCbCr is built-in function
decoded_text = decodeInYCbCr(image_temporary);
set(handles.txtDecoder, 'String', char(native2unicode(decoded_text, 'ISO-8859-1')));
% extract the watermark image
extracted_watermark = extractWatermark(combined_image, original_image);
% display time taken to embed
end_time = cputime - start_time;
disp(['extraction time: ' num2str(end_time)])
image(extracted_watermark,'parent',handles.pic_w_extracted);
axes(handles.pic_w_extracted); axis off;
% compute differences between original watermark image and
% extracted watermark image
gf = wSimilarity(watermark, extracted_watermark, 'all', 'v');
function txtDecoder_Callback(hObject, eventdata, handles)
% hObject    handle to txtDecoder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
function txtDecoder_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtDecoder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
if ispc && isequal(get(hObject,'BackgroundColor'),...
    get(0,'defaultUicontrolBackgroundColor'))

    set(hObject,'BackgroundColor','white');

end
% --- Executes on button press in btnAbout.
function btnAbout_Callback(hObject, eventdata, handles)
% hObject    handle to btnAbout (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
helpdlg({'Spatial & YCbCr Hybrid Watermarking',...
    '- LSB embedding by weighted addition in spatial domain',...
    '- LSB embedding in YCbCr components',',',...
    'Fred Lusson,',',', 'LYIT', 'Letterkenny, Ireland',',',', '2011'});
% --- Executes on button press in btnLoadAttackedImg.
function btnLoadAttackedImg_Callback(hObject, eventdata, handles)
% hObject    handle to btnLoadAttackedImg (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global combined_image original_image;
% Invoke file selection window
[filename,pathname]=uigetfile({'*.bmp'; '*.jpg'}, 'Fred Lusson');
% Load image

```

```

combined_image = imread(filename);
image(combined_image,'parent',handles.pic_attacked);
axes(handles.pic_attacked); axis off;
% Display filename under image
set(handles.lblAttacked,'String',filename);
set(handles.txtDecoder,'String','');
%reset image display to empty
reset(handles.pic_w_extracted); hold off;
axes(handles.pic_w_extracted); axis off;
% enable the decode button
set(handles.btnDecode,'Enable','on');
% --- Executes on button press in btnRunAttacks.
function btnRunAttacks_Callback(hObject, eventdata, handles)
% hObject    handle to btnRunAttacks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global combined_image original_image_y_size original_image_x_size;
% BEGINNING of tests against robustness
runTests(combined_image, original_image_y_size, original_image_x_size);

```

## Function to insert the watermark image “FL”

```
function [combined_image] = insertWatermark(original_image, ...
                                            original_watermark_image)
% insertWatermark Inserts the image watermark into the original RGB
% image
%
% USAGE:
% [combined_image] = insertWatermark(original_image,
% original_watermark_image)
%
% INPUT:
% original_watermark_image: original watermark image.
% original_image: original image to watermark.
%
% OUTPUT:
% combined_image: watermark + original image combined
%
% AUTHOR:
% Frederic Lusson
%
% *****
global original_image_x_size original_image_y_size ...
original_watermark_y_size original_watermark_x_size;
% the percent used to multiply the watermark bit values by before
% adding them to original_image.jpg
% 1.0 = 100
WEIGHTED_COMBINE_VALUE = 0.01;
% grab the x and y resolution for original image
original_image_x_size = size( original_image, 2 );
original_image_y_size = size( original_image, 1 );
original_watermark_x_size = size(original_watermark_image,2);
original_watermark_y_size = size(original_watermark_image,1);
% resize watermark image to be the same same as the original host image
resized_watermark_image = imresize(original_watermark_image,...
                                   [original_image_y_size, original_image_x_size]);
% create a blank image the same size as the original host image
% this will store the new image that is a combination of the original host image
% and watermark
combined_image = zeros(original_image_y_size, original_image_x_size,3);
% add the original host image and the watermark but only give the pixels in
% the resized watermark a weighted value determined by
% WEIGHTED_COMBINE_VALUE
for x_value = 1:original_image_x_size
    for y_value = 1:original_image_y_size
```

```
combined_image(y_value,x_value,1) = original_image(y_value,x_value,1)...
+ (resized_watermark_image(y_value,x_value,1) * WEIGHTED_COMBINE_VALUE);
combined_image(y_value,x_value,2) = original_image(y_value,x_value, 2)...
+ (resized_watermark_image(y_value,x_value,2) * WEIGHTED_COMBINE_VALUE);
combined_image(y_value,x_value,3) = original_image(y_value,x_value,3)...
+ (resized_watermark_image(y_value,x_value,3) * WEIGHTED_COMBINE_VALUE);
end

end
% force the combined image colour values to be between 0 and 255
combined_image = uint8( combined_image );
```



## Function to extract the watermark image “FL”

```
function [watermark_restored_image]=extractWatermark(combined_image,original_image)
% extractWatermark Extracts the image watermark embedded in RGB combined_image
%
% USAGE:
%     [watermark_restored_image] = extractWatermark(combined_image,
%     original_image)
%
% INPUT:
%     combined_image: original image combined with the watermark.
%     original_image: original image to watermark.
%
% OUTPUT:
%     watermark_restored_image: the watermark image scaled to its original
%     dimentionions
%
% AUTHOR:
%     Frederic Lusson
%
% *****
global original_image_x_size original_image_y_size original_watermark_y_size ...
original_watermark_x_size;
WEIGHTED_COMBINE_VALUE = 0.01;
% check needed to extract the embedded watermark by comparing the original
if isempty(original_image)

    [filename1,pathname] = uigetfile('*.jpg','select the original image');
    original_image = imread(num2str(filename1));

end
% create a blank image the same size as the original host image
watermark_restored_big_image = zeros( original_image_y_size, ...
                                     original_image_x_size,3);
combined_image = imresize( combined_image, [original_image_y_size, ...
                                     original_image_x_size] );
% take the values in the combined image and subtract them from the original
% host image values, this will give the values that were added to
% the original host image above, then divide these values by
% WEIGHTED_COMBINE_VALUE in order to try
% to get the original values in the original watermark
for x_value = 1:original_image_x_size

    for y_value = 1:original_image_y_size
```

```

watermark_restored_big_image( y_value, x_value, 1 ) = ...
( combined_image( y_value, x_value, 1 ) - ...
original_image( y_value, x_value, 1 ) ) / WEIGHTED_COMBINE_VALUE;
watermark_restored_big_image( y_value, x_value, 2 ) = ...
( combined_image( y_value, x_value, 2 ) - ...
original_image( y_value, x_value, 2 ) ) / WEIGHTED_COMBINE_VALUE;
watermark_restored_big_image( y_value, x_value, 3 ) = ...
( combined_image( y_value, x_value, 3 ) - ...
original_image( y_value, x_value, 3 ) ) / WEIGHTED_COMBINE_VALUE;
end

end
% force watermark restored image colour values to be between 0 and 255
watermark_restored_big_image = uint8( watermark_restored_big_image );
% resize watermark_restored_big_image to the size of the original
% watermark
watermark_restored_image = imresize( watermark_restored_big_image, ...
[ original_watermark_y_size , original_watermark_x_size ] );

```

## Function to embed the ASCII Watermark in the YCbCr colour space

```
function [image_output] = encodeInYCbCr(ascii_codes, image_temporary)
% encodeInYCbCr converts the ASCII watermark to binary and embedded in
% image_temporary
%
% USAGE:
%     [image_output] = encodeInYCbCr(ascii_codes, image_temporary)
%
% INPUT:
%     image_temporary: image matrix or vector.
%     ascii_codes: text watermark.
%
% OUTPUT:
%     image_output: the original image combined with the watermark
%
% AUTHOR:
%     Frederic Lusson
%
% *****
BLOCK_SIZE = x;
% Generate watermark bits from text area
WATERMARK_MAX_SIZE = 32 * 8;
% has to be a multiple of 8
watermark = zeros(1, WATERMARK_MAX_SIZE);
for l=1:size(ascii_codes,2)
    code_string=num2str(dec2bin(ascii_codes(l),8));
    for m=1:8
        code_string(m);
        watermark(m+((l-1)*8))=eval(code_string(m));
    end
end
% Conversion RGB -> YCbCr
image_temporary_ycbcr = rgb2ycbcr(image_temporary);
% Extract Y, Cb, Cr components
y = image_temporary_ycbcr(:,:,1);
cb = image_temporary_ycbcr(:,:,2);
cr = image_temporary_ycbcr(:,:,3);
% Create LSB matrix for Cb and Cr components
% For 2-bit coding use mod(component,4)
lsbits_cb = mod(cb,2);
lsbits_cr = mod(cr,2);
```

```

% --WATERMARK EMBEDDING ALGORITHM--
bit_position = 1;
% we need this to adjust automatically to the selected image, where width
% and height may vary
[w_size,k_size,p] = size(image_temporary);
for w=1:(w_size / BLOCK_SIZE)

    for k=1:(k_size / BLOCK_SIZE)

        % define our array position (start/end) at each pass
        w_s = ((w - 1) * BLOCK_SIZE) + 1; %width_start
        w_e = w * BLOCK_SIZE; %width_end
        h_s = ((k - 1) * BLOCK_SIZE) + 1; % height_start
        h_e = k * BLOCK_SIZE; % height_end

        % Embed watermark bits in blocks BLOCK_SIZE x BLOCK_SIZE only
        if(mod((w + k), 2) == 0)

            % just a check to ensure we do not go over the max watermark
            % length
            if(bit_position < WATERMARK_MAX_SIZE)

                if(watermark(bit_position) == 1)

                    % If current watermark bit is 1, then LSB of Cb is 0 and LSB
                    % of Cr is 1
                    % For 2-bit coding we can use cr()=cr()+3;
                    cb(w_s:w_e, h_s:h_e) = cb(w_s:w_e, h_s:h_e) - ...
                                            lsbits_cb(w_s:w_e, h_s:h_e);
                    cr(w_s:w_e, h_s:h_e) = cr(w_s:w_e, h_s:h_e) - ...
                                            lsbits_cr(w_s:w_e, h_s:h_e)+1;
                else

                    % If current watermark bit is 0, then LSB of Cb is 1
                    % and LSB of Cr is 0
                    cr(w_s:w_e, h_s:h_e) = cr(w_s:w_e, h_s:h_e) - ...
                                            lsbits_cr(w_s:w_e, h_s:h_e);
                    cb(w_s:w_e, h_s:h_e) = cb(w_s:w_e, h_s:h_e) - ...
                                            lsbits_cb(w_s:w_e, h_s:h_e)+1;
                end
            end
        else
            continue
        end
    end
end

```

```

else

    % Opposite scheme in adjacent blocks
    % just a check to ensure we do not go over the max watermark
    % length
    if(bit_position < WATERMARK_MAX_SIZE)

        if(watermark(bit_position)==1)

            cr(w_s:w_e, h_s:h_e) = cr(w_s:w_e, h_s:h_e) - ...
                                lsbits_cr(w_s:w_e, h_s:h_e);
            cb(w_s:w_e, h_s:h_e) = cb(w_s:w_e, h_s:h_e) - ...
                                lsbits_cb(w_s:w_e, h_s:h_e)+1;

        else

            cb(w_s:w_e, h_s:h_e) = cb(w_s:w_e, h_s:h_e) - ...
                                lsbits_cb(w_s:w_e, h_s:h_e);
            cr(w_s:w_e, h_s:h_e) = cr(w_s:w_e, h_s:h_e) - ...
                                lsbits_cr(w_s:w_e, h_s:h_e)+1;

        end

    end

    else

        continue

    end

    end

    bit_position = bit_position + 1;

end

end

% Create result image
image_temporary_ycbcr(:, :, 1) = y;
image_temporary_ycbcr(:, :, 2) = cb;
image_temporary_ycbcr(:, :, 3) = cr;
% Conversion YCbCr -> RGB
image_output = ycbcr2rgb(image_temporary_ycbcr);

```

## Function to extract the ASCII watermark

```
function [decoded_txt] = decodeInYCbCr(image_temporary)
% decodeInYCbCr Computes the ASCII watermark embedded in image_temporary
%
% USAGE:
%     [decoded_txt] = decodeInYCbCr(image_temporary)
%
% INPUT:
%     image_temporary: image matrix or vector.
%
% OUTPUT:
%     decoded_txt: the watermark ASCII string
%
% AUTHOR:
%     Frederic Lusson
%
% *****
BLOCK_SIZE = x;
% Define starting watermark payload as zero bits
WATERMARK_MAX_SIZE = 32 * 8;
watermark = zeros(1,WATERMARK_MAX_SIZE);
% Conversion RGB -> YCbCr
image_temporary_ycbcr = rgb2ycbcr(image_temporary);
% Extract Y, Cb, Cr components
y=image_temporary_ycbcr(:,:,1);
cb=image_temporary_ycbcr(:,:,2);
cr=image_temporary_ycbcr(:,:,3);
% Create LSB matrix for Cb and Cr components
lsbits_cb = mod(cb,2);
lsbits_cr = mod(cr,2);
% --WATERMARK EXTRACTION ALGORITHM--
bit_position=1;
% Counters used for deciding watermark value in block
counter0=0;
counter1=0;
% we need this to adjust automatically to the selected image, where width
% and height may vary
[w_size,k_size,p] = size(image_temporary);
for block_w=1:(w_size / BLOCK_SIZE)

    for block_k=1:(k_size / BLOCK_SIZE)
        % Read image in blocks
        if mod((block_w + block_k), 2) == 0

            for w = ((block_w-1) * BLOCK_SIZE) + 1:block_w * BLOCK_SIZE
```

```

        for k=((block_k-1)*BLOCK_SIZE)+1:block_k*BLOCK_SIZE
            if lsbits_cb(w,k) < 1
                % For 2-bit coding use lsbits_cb()<2
                counter1=counter1+1;
            else
                counter0=counter0+1;
            end
            if lsbits_cr(w,k) >= 1
                counter1=counter1+1;
            else
                counter0=counter0+1;
            end
        end
    end
else
    for w=((block_w-1)*BLOCK_SIZE)+1:block_w*BLOCK_SIZE

        for k=((block_k-1)*BLOCK_SIZE)+1:block_k*BLOCK_SIZE
            if lsbits_cb(w,k)>=1
                counter1=counter1+1;
            else
                counter0=counter0+1;
            end
            if lsbits_cr(w,k)<1
                counter1=counter1+1;
            else
                counter0=counter0+1;
            end
        end
    end
end
% Decision of watermark bit value in current block
if counter0 > counter1
    watermark(bit_position)=0;
else
    watermark(bit_position)=1;
end
% Move to next position in watermark
bit_position=bit_position+1;
% Reset counters

```

```

        counter0 = 0;
        counter1 = 0;
    end

end

end
% Conversion of binary form of watermark to text displayed in text area
read_bit = 1;
for read_byte=1:(WATERMARK_MAX_SIZE/8)

    ascii_code=0;
    for read_bit=1:8

        bit_decimal_value = watermark(read_bit+((read_byte-1)*8))*...
            2^(abs((read_bit-1)-7));
        ascii_code = ascii_code + bit_decimal_value;
    end
    decoded_txt(read_byte) = ascii_code;
end
end

```



## Function to compute the PSNR

```
function decibels = PSNR(x, y)
% PSNR - compute the Peak Signal to Noise Ratio, defined by :
%       PSNR(x,y) = 10*log10( max(max(x),max(y))^2 / |x-y|^2 ).
%
% INPUT:
%   x is the original image
%   y is the combined image: original + watermarks
%
% OUTPUT:
%   decibels: above 38dbs is considered invisible
%
% AUTHOR:
%       Frederic Lusson
%
% *****
x = double(x); % image width
y = double(y); % image length
% identical images
if(x == y)
    decibels = 100;
% find difference between images
else
    d = mean( mean( (x(:)-y(:)).^2 ) );
    m1 = max( abs(x(:)) );
    m2 = max( abs(y(:)) );
    m = max(m1,m2);
    decibels = 10*log10( m^2/d );
end
```

## Function to compute the SSIM

```
function [mssim, ssim_map] = ssim(img1, img2, K, window, L)
%
% This is an implementation of the algorithm for calculating the
% Structural SIMilarity (SSIM) index between two images
%
% Reference:
%
% Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
% quality assessment: From error visibility to structural similarity,
% IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612,
% Apr. 2004.
%
%
% INPUT :
% (1) img1: the first image being compared
% (2) img2: the second image being compared
% (3) K: constants in the SSIM index formula (see the above
%     reference). default value: K = [0.01 0.03]
% (4) window: local window for statistics (see the above
%     reference). default window is Gaussian given by
%     window = fspecial('gaussian', 11, 1.5);
% (5) L: dynamic range of the images. default: L = 255
%
% OUTPUT:
% (1) mssim: the mean SSIM index value between 2 images.
%     If one of the images being compared is regarded as
%     perfect quality, then mssim can be considered as the
%     quality measure of the other image.
%     If img1 = img2, then mssim = 1.
% (2) ssim_map: the SSIM index map of the test image. The map
%     has a smaller size than the input images. The actual size
%     depends on the window size and the downsampling factor.
%
% USAGE:
% Given 2 test images img1 and img2, whose dynamic range is 0-255
%
% [mssim, ssim_map] = ssim(img1, img2);
%
% *****
% check arguments
if (nargin < 2 || nargin > 5)
    mssim = -Inf;
    ssim_map = -Inf;
```

```

        return;
    end
    if (size(img1) ~= size(img2))
        mssim = -Inf;
        ssim_map = -Inf;
        return;
    end
    [M N] = size(img1);
    if (nargin == 2)
        if ((M < 11) || (N < 11))
            mssim = -Inf;
            ssim_map = -Inf;
            return
        end

        window = fspecial('gaussian', 11, 1.5);
        K(1) = 0.01;
        K(2) = 0.03;
        L = 255;
    end
    if (nargin == 3)
        if ((M < 11) || (N < 11))
            mssim = -Inf;
            ssim_map = -Inf;
            return;
        end
        window = fspecial('gaussian', 11, 1.5);
        L = 255;
        if (length(K) == 2)
            if (K(1) < 0 || K(2) < 0)
                mssim = -Inf;
                ssim_map = -Inf;
                return;
            end;
        else
            mssim = -Inf;
            ssim_map = -Inf;
        return;
    end
end

```

```

end
if (nargin == 4)
    [H W] = size(window);
    if ((H*W) < 4 || (H > M) || (W > N))
        mssim = -Inf;
        ssim_map = -Inf;
        return;
    end
    L = 255;
    if (length(K) == 2)
        if (K(1) < 0 || K(2) < 0)
            mssim = -Inf;
            ssim_map = -Inf;
            return;
        end
    else
        mssim = -Inf;
        ssim_map = -Inf;
        return;
    end
end
end
if (nargin == 5)
    [H W] = size(window);

    if ((H*W) < 4 || (H > M) || (W > N))
        mssim = -Inf;
        ssim_map = -Inf;
        return;
    end

    if (length(K) == 2)
        if (K(1) < 0 || K(2) < 0)
            mssim = -Inf;
            ssim_map = -Inf;
            return;
        end
    else
        mssim = -Inf;
        ssim_map = -Inf;
        return;
    end
end

```

```

        end

end
img1 = double(img1);
img2 = double(img2);
% automatic downsampling
f = max(1,round(min(M,N)/256));
%downsampling by f
%use a simple low-pass filter
if(f>1)

    lpf = ones(f,f);
    lpf = lpf/sum(lpf(:));
    img1 = imfilter(img1,lpf,'symmetric','same');
    img2 = imfilter(img2,lpf,'symmetric','same');
    img1 = img1(1:f:end,1:f:end);
    img2 = img2(1:f:end,1:f:end);

end

C1 = (K(1)*L)^2;
C2 = (K(2)*L)^2;
window = window/sum(sum(window));
mu1 = filter2(window, img1, 'valid');
mu2 = filter2(window, img2, 'valid');
mu1_sq = mu1.*mu1;
mu2_sq = mu2.*mu2;
mu1_mu2 = mu1.*mu2;
sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;
sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;
sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;
if (C1 > 0 && C2 > 0)

    ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./...
        ((mu1_sq + mu2_sq + C1).*(sigma1_sq + sigma2_sq + C2));

else

    numerator1 = 2*mu1_mu2 + C1;
    numerator2 = 2*sigma12 + C2;
    denominator1 = mu1_sq + mu2_sq + C1;
    denominator2 = sigma1_sq + sigma2_sq + C2;
    ssim_map = ones(size(mu1));
    index = (denominator1.*denominator2 > 0);
    ssim_map(index) = (numerator1(index).*numerator2(index))./...
        (denominator1(index).*denominator2(index));
    index = (denominator1 ~= 0) & (denominator2 == 0);
    ssim_map(index) = numerator1(index)./denominator1(index);

```

```
end
mssim = mean2(ssim_map);
return;
```

## Function to generate attacks

```
function runTests(image_output, output_y_size, output_x_size)
% runTests runs image attack on the combine image
%
% USAGE:
%     runTests(image_output, output_y_size, output_x_size)
%
% INPUT:
%     image_output:  the image to test.
%     output_y_size: y axis size of image to test.
%     output_x_size: x axis size of image to test.
%
% OUTPUT:
%     all images that have gone through some processing
%
% DESCRIPTION:
%     List of image attacks:
%     - scaling: from original size to 10 x 10px (each iteration
%     subtracts 10px from previous size
%     - compression: from 5% to 95%
%     - rotation: every 0.1 degree between -0.3 to +0.3
%     - noise addition:
%         0 means every pixel contains added noise
%         1 means ~50% of pixels contain added noise,
%         2 means ~33%, %           3 means ~25%
%
% AUTHOR:
%     Frederic Lusson
%
% *****
% image output is the combined image: original image + image watermark +
% text
% RESIZING the percent to resize the combined image 10%
for i=output_y_size: -50: 50

    resized_image = imresize( image_output, [i, i] );
    imwrite(resized_image, sprintf('attacks/resize_%d.bmp', i),'bmp');

end
% COMPRESSION
imwrite(image_output,sprintf('attacks/compression_100_percent.jpg',100),...
        'Quality', 100); PERCENT_OF_COMPRESSION = 50;
for i=5: 10:100

    imwrite(image_output,sprintf('attacks/compression_%d_percent.jpg',i),...
```

```

        'Quality', i);

end
% ROTATION
for i=-5: 0.1: 5

    image_rotated = imrotate( image_output, i );
    imwrite(image_rotated,sprintf('attacks/rotation_%0.1f.bmp',i),'bmp');

end
% NOISE ADDITION
% specifies the maximum noise value
MAX_NOISE_VALUE = 20;
% specifies the approximate amount of noise
% 0 means every pixel of the combined image will have added noise
% 1 means ~50% will have added noise, 2 means ~33%, 3 means ~25%, ....
for i=0:3

    noise_image = uint8( zeros( output_y_size, output_x_size, 3 ) );
    % create a random noise image
    for x_value = 1:output_x_size
        for y_value = 1:output_y_size

            if round( rand() * i) == 0

                noise_image( y_value, x_value, 1 ) = round( rand() *...
                    (MAX_NOISE_VALUE + 1) );
                noise_image( y_value, x_value, 2 ) = round( rand() *...
                    (MAX_NOISE_VALUE + 1) );
                noise_image( y_value, x_value, 3 ) = round( rand() *...
                    (MAX_NOISE_VALUE + 1) );

            end

        end

    end

    % add the noise image to the combined image
    noise_combined_image = image_output + noise_image;
    imwrite(noise_combined_image,sprintf('attacks/noise_%d.bmp',i),'bmp');

end
end

```



## Function to compute NMSE and R

```
function [gf] = wSimilarity(t, y, fMeasure, options)
% wSimilarity computes similarity measures between the original watermark
% and the extracted watermark after attacks
%
% USAGE:
% [gf] = wSimilarity(t,y)
% [gf] = wSimilarity(t,y,fMeasure)
% [gf] = wSimilarity(t,y,fMeasure,options)
%
% INPUT:
% t: matrix or vector of target values for regression model
% y: matrix or vector of output from regression model.
% fMeasure: a string or cell array of string values representing
% different form of goodness of fit measure as follows:
%
% 'all' - calculates all the measures below
% '1' - normalised mean squared error (nmse)
% '2' - coefficient of correlation (r)
%
% options: a string containing other output options, currently the only
% option is verbose output.
%
% 'v' - verbose output, posts some text output for the
% chosen measures to the command line
%
% OUTPUT:
% gf: vector of goodness of fit values between model output and target
% for each of the strings in fMeasure
%
% EXAMPLES
%
% gf = wSimilarity(t,y); for all statistics in list returned as vector
%
% gf = wSimilarity(t,y,'all','v'); for all statistics in list returned
% as vector with information posted to
% the command line on each statistic
%
% AUTHOR:
% Frederic Lusson
%
% *****
error(nargchk(2,4,nargin));
% reshape matrices into vectors (order of data is not important)
t = reshape(t,1,[]);
```

```

y = reshape(y,1,[]);
if length(t) ~= length(y)

    error('Invalid data size: size of t and y must be same')
end
if nargin > 2

    % check if fMeasure is cell string array or just a string
    if ~iscell(fMeasure) && ischar(fMeasure)
        if strcmp(fMeasure,'all')

            % if the string 'all' is passed in, all the stats are
            % required so make the appropriate cell string array
            fMeasure = {'1' '2'};
            % return all measures
        else

            % otherwise convert string to cell string array of size 1
            fMeasure = {fMeasure};
        end
    else

        % if it is a cell array of strings, check its size
        if size(fMeasure,2) == 1

            % if there is only one element check it is not a request
            % for all measures
            if strcmp(char(fMeasure),'all')

                % if the string 'all' is passed in, all the stats are
                % required so make the appropriate cell string array
                fMeasure = {'1' '2'};
                % return all measures
            end
        end
    end
end
else

    % return all measures if only two inputs, nothing will be posted to
    % the command line
    fMeasure = {'1' '2'};

```

```

end
% remove NaNs from the arrays, avoid modifying them if there are no
% NaNs to prevent reallocation of memory
if sum(isnan(t) | isnan(y))
    inds = ~isnan(t) | ~isnan(y);
    t = t(inds);
    y = y(inds);
end
e = t - y; % Calculate the error
gf = ones(1,size(fMeasure,2)); % preallocate array
for i = 1:size(fMeasure,2)
    switch char(fMeasure(i))
        case '1' % normalised mean squared error
            gf(i) = mean(e.^2)/var(double(t));
            if nargin == 4
                if options == 'v'
                    disp(['normalised mean squared error (nmse): ...'
                        num2str(gf(i))])
                end
            end
        case '2' % coefficient of correlation
            cf = corr2(t,y); % 1 - perfect match
            gf(i) = cf;
            if nargin == 4
                if options == 'v'
                    disp(['coefficient of correlation (r): ' num2str(gf(7))])
                end
            end
        otherwise
            error('Invalid measure in fMeasure(%d):...
                \nIt must be one of the strings {1 2}, ...
                but actually contained "%s"',i,char(fMeasure(i)))
    end
end

```

```
        end
    end
    return;
```

# Appendix E

Attacks	Watermark “FL”	ASCII Watermark
JPEG compression	yes	no
JPEG 2000 compression	yes	no
Noise Addition	yes	no
Resizing	yes	no
Rotation	no	yes (limit -1, +1 degree)
Cropping	yes	yes (limited)
Clipping	yes	yes (limited)
Collage	yes	yes
Brightness contrast	yes	yes
Gaussian Blur	yes	no
Hue Saturation Lightness	yes	no
Histogram	no	no
Sharpening	yes	no
Median Filter	no	yes (90%)
Self Similarity	no	yes
Stirmark Strength Test	no	yes

Table 22: Watermark survival to attacks

Images	“lena”			“crown”			“girl”		
	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
100	0.0003	0.86	0	0.0003	0.84	0	0.0077	0.09	0
50	0.0087	0.94	100	0.0088	0.94	100	0.0139	0.38	100
33	0.0117	0.94	100	0.0118	0.94	100	0.0159	0.50	100
25	0.0128	0.94	100	0.0129	0.94	100	0.0167	0.54	100

Table 23: Results after noise addition attacks

Images	“plane”			“boat”			“peppers”			“baboon”		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
100	0.0003	0.90	0	0.0003	0.90	0	0.0002	0.90	0	0.0003	0.88	0
50	0.0088	0.94	100	0.0088	0.94	100	0.0088	0.94	100	0.0089	0.94	100
33	0.0119	0.94	100	0.0119	0.94	100	0.0118	0.94	100	0.0119	0.94	100
25	0.0129	0.94	100	0.0130	0.94	100	0.0130	0.94	100	0.0130	0.94	100

Table 24: Results after noise addition attacks (continued)

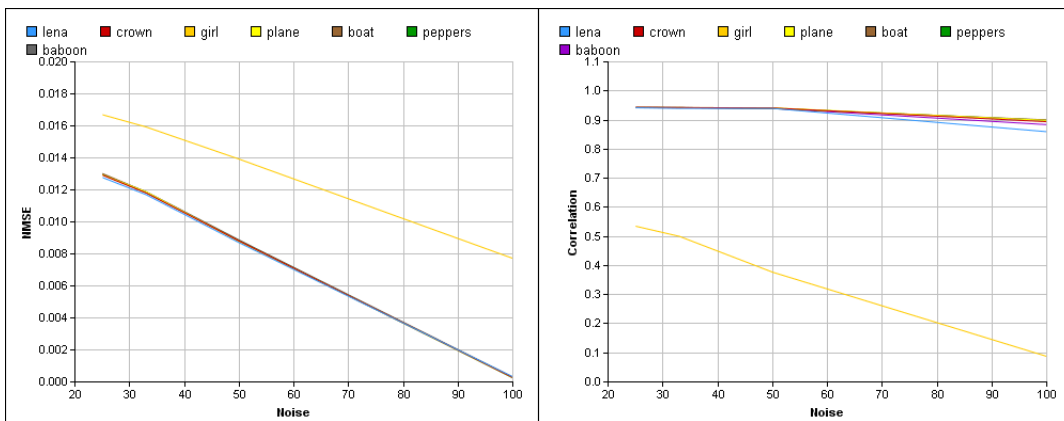


Figure 32: Test images comparison - noise addition attacks

Images	"lena"			"crown"			"girl"		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
5	0.0220	0.09	0	0.0213	0.09	0	0.0224	0.07	0
15	0.0241	0.30	0	0.0232	0.29	0	0.0243	0.18	0
25	0.0244	0.44	0	0.0239	0.44	0	0.0244	0.22	0
35	0.0244	0.52	0	0.0243	0.52	0	0.024	0.25	0
45	0.0244	0.59	0	0.0243	0.57	0	0.0245	0.26	0
55	0.0245	0.64	0	0.0244	0.66	0	0.0245	0.27	0
65	0.0245	0.68	0	0.024	0.68	0	0.0245	0.28	0
75	0.0245	0.73	0	0.0245	0.75	0	0.0245	0.30	0
85	0.0245	0.79	0	0.0245	0.81	0	0.0245	0.32	0
95	0.0245	0.84	0	0.0245	0.86	0	0.0245	0.36	0

Table 25: Results after JPEG compression attacks

Images	"plane"			"boat"			"peppers"			"baboon"		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
5	0.0218	0.11	0	0.0225	0.13	0	0.0227	0.11	0	0.0244	0.16	0
15	0.0222	0.30	0	0.0240	0.29	0	0.0243	0.29	0	0.0245	0.27	0
25	0.0232	0.38	0	0.0243	0.39	0	0.0245	0.41	0	0.0245	0.37	0
35	0.0238	0.55	0	0.0245	0.49	0	0.0245	0.50	0	0.0245	0.44	0
45	0.0239	0.59	0	0.0245	0.55	0	0.0245	0.56	0	0.0245	0.52	0
55	0.0240	0.63	0	0.0245	0.57	0	0.0245	0.61	0	0.0245	0.57	0
65	0.0234	0.64	0	0.0245	0.61	0	0.0245	0.65	0	0.0245	0.61	0
75	0.0240	0.71	0	0.0245	0.64	0	0.0245	0.71	0	0.0245	0.66	0
85	0.0239	0.75	0	0.0245	0.69	0	0.0245	0.77	0	0.0245	0.71	0
95	0.0240	0.82	0	0.0245	0.75	0	0.0245	0.81	0	0.0245	0.73	0

Table 26: Results after JPEG compression attacks (continued)

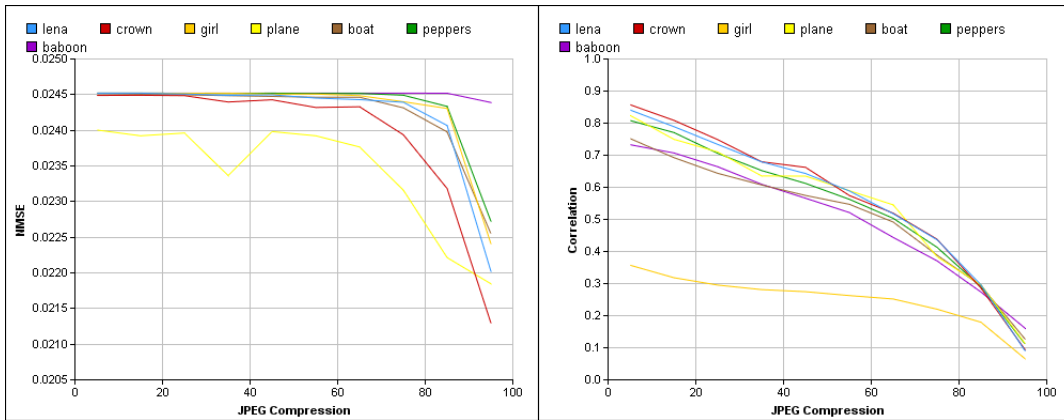


Figure 33: Test images comparison - JPEG compression attacks

Images	"lena"			"crown"			"girl"		
	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
10	0.0240	0.53	0	0.0236	0.64	0	0.0242	0.25	0
20	0.0243	0.63	0	0.0241	0.76	0	0.0243	0.29	0
30	0.0244	0.74	0	0.0242	0.79	0	0.0244	0.32	0
40	0.0245	0.76	0	0.0242	0.82	0	0.0245	0.34	0
50	0.0245	0.79	0	0.0243	0.86	0	0.0245	0.37	0
60	0.0245	0.80	0	0.0243	0.87	0	0.0245	0.38	0
70	0.0245	0.81	0	0.0243	0.88	0	0.0245	0.39	0
80	0.0245	0.87	0	0.0243	0.88	0	0.0245	0.40	0
90	0.0155	0.94	0	0.0151	0.94	0	0.0181	0.60	0

Table 27: Results after JPEG2000 compression attacks



Images	“plane”			“boat”			“peppers”			“baboon”		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
10	0.0222	0.53	0	0.0236	0.39	0	0.0244	0.50	0	0.02	0.26	0
20	0.0227	0.51	0	0.0241	0.50	0	0.0245	0.60	0	0.02	0.38	0
30	0.0232	0.65	0	0.0243	0.55	0	0.0245	0.68	0	0.02	0.44	0
40	0.0235	0.73	0	0.0243	0.57	0	0.0245	0.71	0	0.02	0.50	0
50	0.0236	0.78	0	0.0243	0.59	0	0.0245	0.74	0	0.02	0.55	0
60	0.0238	0.84	0	0.0244	0.63	0	0.0245	0.76	0	0.02	0.61	0
70	0.0240	0.87	0	0.0244	0.68	0	0.0245	0.77	0	0.02	0.65	0
80	0.0240	0.88	0	0.0244	0.72	0	0.0245	0.79	0	0.02	0.66	0
90	0.0151	0.94	0	0.0157	0.94	0	0.0155	0.94	0	0.0155	0.94	0

Table 28: Results after JPEG2000 compression attacks (continued)

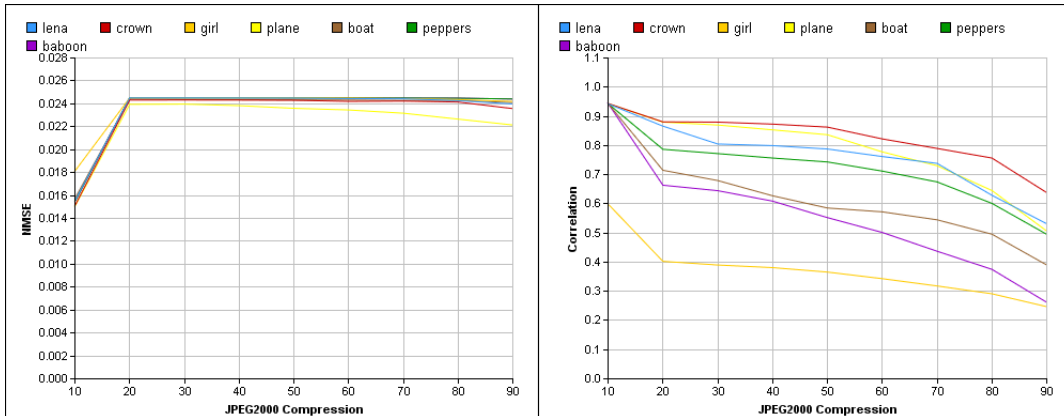


Figure 34: Test images comparison - JPEG2000 compression attacks

Images	“lena”			“crown”			“girl”		
Size (pixel)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
62x62	0.0244	0.46	0	0.0243	0.54	0	0.0245	0.22	0
112x112	0.0245	0.68	0	0.0243	0.72	0	0.0245	0.29	0
162x162	0.0245	0.75	0	0.0242	0.80	0	0.0245	0.32	0
212x212	0.0245	0.77	0	0.0239	0.83	0	0.0245	0.34	0
262x262	0.0245	0.79	0	0.0237	0.85	0	0.0224	0.36	0
312x312	0.0245	0.81	0	0.0235	0.87	0	0.0245	0.38	0
362x362	0.0244	0.83	0	0.0233	0.88	0	0.0245	0.40	0
412x412	0.0244	0.86	0	0.0231	0.90	0	0.0244	0.43	0
462x462	0.0243	0.88	0	0.0227	0.92	0	0.0245	0.46	0

Table 29: Results after image resizing attacks

Images	“plane”			“boat”			“peppers”			“baboon”		
Size (pixel)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
62x62	0.0242	0.40	0	0.0245	0.33	0	0.0245	0.41	0	0.0245	0.24	0
112x112	0.0242	0.60	0	0.0245	0.57	0	0.0245	0.68	0	0.0245	0.41	0
162x162	0.0241	0.68	0	0.0245	0.65	0	0.0245	0.74	0	0.0245	0.49	0
212x212	0.0240	0.73	0	0.0244	0.68	0	0.0245	0.76	0	0.0245	0.53	0
262x262	0.0239	0.77	0	0.0244	0.71	0	0.0245	0.78	0	0.0245	0.56	0
312x312	0.0238	0.80	0	0.0244	0.73	0	0.0245	0.79	0	0.0245	0.59	0
362x362	0.0236	0.83	0	0.0244	0.75	0	0.0245	0.81	0	0.0245	0.62	0
412x412	0.0233	0.86	0	0.0244	0.78	0	0.0245	0.83	0	0.0245	0.65	0
462x462	0.0228	0.89	0	0.0244	0.82	0	0.0244	0.83	0	0.0245	0.70	0

Table 30: Results after image resizing attacks (continued)

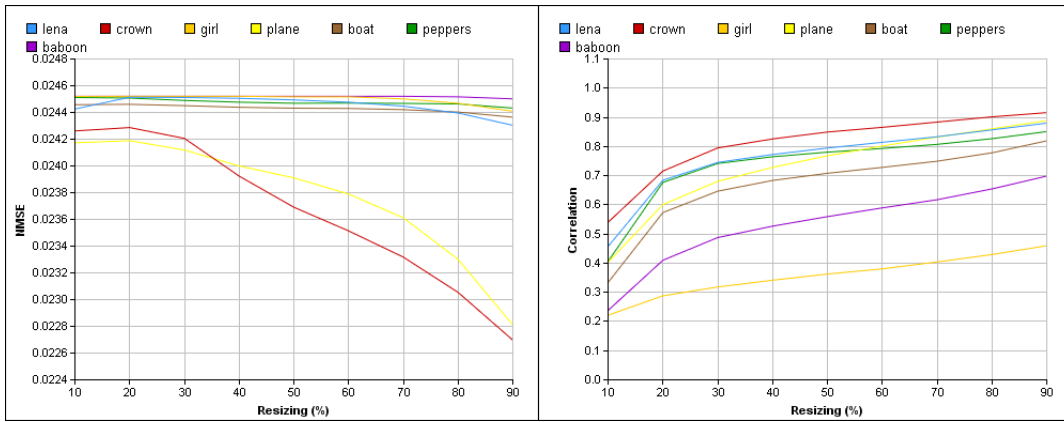


Figure 35: Test images comparison - resizing attacks

Images	"lena"			"crown"			"girl"		
Angle (de- gree)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
-1	0.0239	0.17	0	0.0234	0.36	90	0.0242	0.16	20
-0.9	0.0240	0.17	0	0.0234	0.36	90	0.0242	0.15	20
-0.8	0.0240	0.17	0	0.0234	0.36	90	0.0242	0.15	50
-0.7	0.0241	0.17	0	0.0234	0.36	90	0.0243	0.16	50
-0.6	0.0242	0.18	0	0.0235	0.36	90	0.0243	15	70
-0.5	0.0243	0.27	0	0.0240	0.48	100	0.0244	0.20	100
-0.4	0.0244	0.28	0	0.0237	0.48	100	0.0244	0.21	100
-0.3	0.0244	0.29	0	0.0237	0.48	100	0.0244	0.21	100
-0.2	0.0244	0.30	0	0.0238	0.48	100	0.0245	0.21	100
-0.1	0.0245	0.58	0	0.0238	0.48	100	0.0245	0.31	100
0	0.0243	0.63	0	0.0151	0.94	100	0.0181	0.6	100
0.1	0.0245	0.56	0	0.0239	0.69	100	0.0245	0.31	100
0.2	0.0244	0.26	0	0.0239	0.47	100	0.0244	0.20	100
0.3	0.0244	0.24	0	0.0238	0.46	100	0.0244	0.20	100
0.4	0.0243	0.23	0	0.0239	0.46	100	0.0244	0.19	100
0.5	0.0243	0.22	0	0.0239	0.45	100	0.0244	0.19	100
0.6	0.0242	0.12	0	0.0235	0.34	100	0.0243	0.14	100
0.7	0.0242	0.11	0	0.0235	0.33	100	0.0242	0.13	100
0.8	0.0241	0.10	0	0.0236	0.32	100	0.0242	0.13	100
0.9	0.0241	0.09	0	0.0235	0.32	100	0.0242	0.12	100
1.0	0.0240	0.08	0	0.0236	0.31	100	0.0242	0.12	100
1.1	0.0240	0.03	0	0.0235	0.24	20	0.0240	0.09	20

Table 31: Results after rotation attacks

Images	"plane"			"boat"			"peppers"			"baboon"		
Angle (de- gree)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
-1	0.0241	0.27	60	0.0244	0.19	70	0.0235	0.20	70	0.0245	0.10	70
-0.9	0.0242	0.29	70	0.0244	0.20	70	0.0235	0.21	70	0.0245	0.11	70
-0.8	0.0242	0.30	70	0.0244	0.20	70	0.0236	0.21	70	0.0245	0.11	70
-0.7	0.0242	0.31	80	0.0244	0.21	80	0.0236	0.22	80	0.0245	0.12	80
-0.6	0.0243	0.32	80	0.0244	0.21	80	0.0236	0.22	80	0.0245	0.12	80
-0.5	0.0244	0.45	100	0.0245	0.32	100	0.0241	0.33	100	0.0245	0.22	100
-0.4	0.0244	0.47	100	0.0245	0.34	100	0.0242	0.34	100	0.0245	0.23	100
-0.3	0.0244	0.48	100	0.0245	0.35	100	0.0242	0.35	100	0.0245	0.24	100
-0.2	0.0245	0.50	100	0.0245	0.36	100	0.0242	0.36	100	0.0245	0.24	100
-0.1	0.0244	0.68	100	0.0245	0.57	100	0.0244	0.57	100	0.0245	0.43	100
0	0.0151	0.94	100	0.0153	0.94	100	0.0152	0.794	100	0.0153	0.94	100
0.1	0.0242	0.69	100	0.0245	0.53	100	0.0244	0.57	100	0.0245	0.42	100
0.2	0.0245	0.49	100	0.0245	0.33	100	0.0241	0.35	100	0.0245	0.23	100
0.3	0.0245	0.49	100	0.0245	0.33	100	0.0241	0.33	100	0.0245	0.22	100
0.4	0.0245	0.49	100	0.0245	0.31	100	0.0241	0.32	100	0.0245	0.20	100
0.5	0.0245	0.48	100	0.0245	0.30	100	0.0240	0.30	100	0.0245	0.20	100
0.6	0.0244	0.35	100	0.0244	0.19	100	0.0236	0.21	100	0.0245	0.10	100
0.7	0.0244	0.35	100	0.0244	0.18	100	0.0235	0.20	100	0.0245	0.10	100
0.8	0.0243	0.33	100	0.0244	0.17	100	0.0235	0.19	100	0.0245	0.09	100
0.9	0.0243	0.32	100	0.0244	0.17	100	0.0235	0.19	90	0.0245	0.08	90
1.0	0.0243	0.32	100	0.0244	0.16	100	0.0234	0.18	90	0.0245	0.08	90
1.1	0.0242	0.24	20	0.0243	0.09	10	0.0231	0.13	10	0.0245	0.04	10

Table 32: Results after rotation attacks (continued)

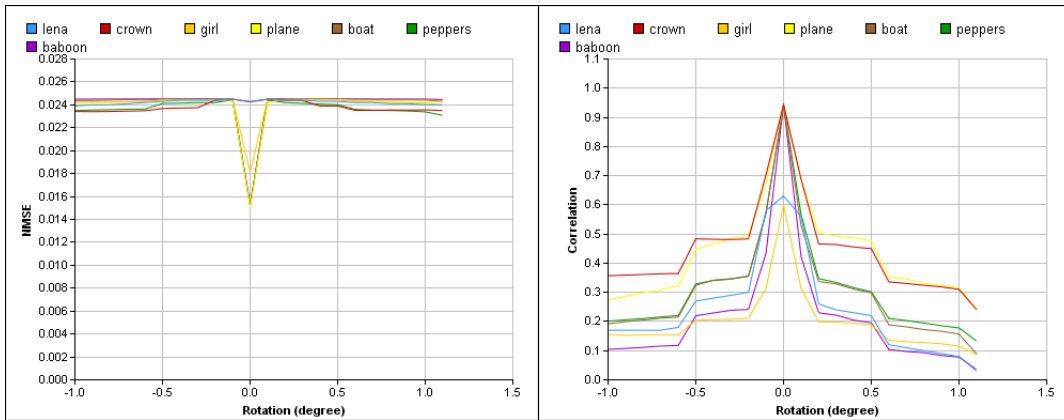


Figure 36: Test images comparison - rotation attacks

Images	“lena”			“crown”			“girl”		
	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
brightness	0.0049	0.88	100	0.0059	0.85	100	0.0181	0.60	100
contrast 1									
brightness	0.0025	0.38	0	0.0020	0.47	0	0.0094	0.14	0
contrast 5									
hue saturation	0.0193	0.75	0	0.0159	0.82	0	0.0244	0.58	0
lightness 1									
gaussian blur 1	0.0244	0.78	0	0.0245	0.85	0	0.0245	0.37	0
gaussian blur 2	0.0245	0.59	0	0.0245	0.73	0	0.0225	0.29	0
histogram	0.0126	0.37	0	0.0201	0.08	0	0.0201	0.16	0
sharpen	0.0150	0.80	0	0.0245	0.86	0	0.0181	0.60	0

Table 33: Results after filters and histogram attacks

Images	“plane”			“boat”			“peppers”			“baboon”		
Attacks	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii	nmse	r	ascii (%)
brightness	0.0058	0.94	100	0.0065	0.86	100	0.0072	0.88	0	0.0062	0.88	100
contrast 1												
brightness	0.0030	0.42	70	0.0059	0.50	0	0.0058	0.43	0	0.0033	0.46	20
contrast 5												
hue	0.0235	0.87	100	0.0209	0.86	100	0.0170	0.83	0	0.0202	0.85	90
saturation												
lightness 1												
gaussian	0.0245	0.73	0	0.0245	0.68	0	0.0245	0.75	0	0.0245	0.54	0
blur 1												
gaussian	0.0245	0.56	0	0.0245	0.50	0	0.0245	0.55	0	0.0245	0.37	0
blur 2												
histogram	0.0111	0.27	0	0.0140	0.25	0	0.0153	0.32	0	0.0206	0.26	0
sharpen	0.0245	0.79	0	0.0245	0.72	0	0.0245	0.81	0	0.0245	0.58	0

Table 34: Results after filters and histogram attacks (continued)

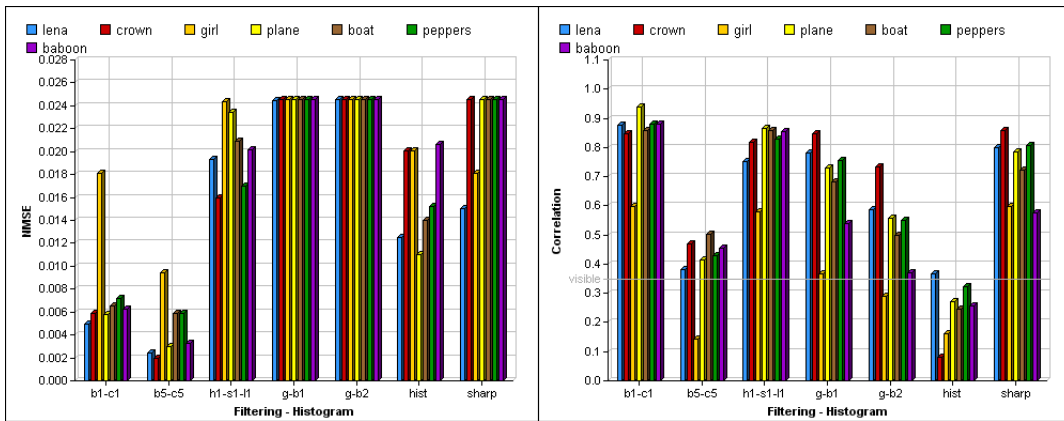


Figure 37: Test images comparison - filtering, histogram attacks

Images	"lena"			"crown"			"girl"		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
10	0.0091	0.94	100	0.0092	0.95	100	0.0101	0.93	100
25	0.0081	0.74	0	0.0082	0.77	0	0.0111	0.72	0
50	0.0068	0.45	0	0.0072	0.49	0	0.0082	0.42	0

Table 35: Results after cropping attacks

Images	"plane"			"boat"			"peppers"			"baboon"		
Level (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
10	0.0091	0.92	100	0.0091	0.94	100	0.0081	0.94	100	0.0127	0.92	100
25	0.0081	0.84	0	0.0081	0.79	0	0.0082	0.72	0	0.0164	0.68	0
50	0.0068	0.44	0	0.0068	0.46	0	0.0067	0.41	0	0.0143	0.40	0

Table 36: Results after cropping attacks (continued)

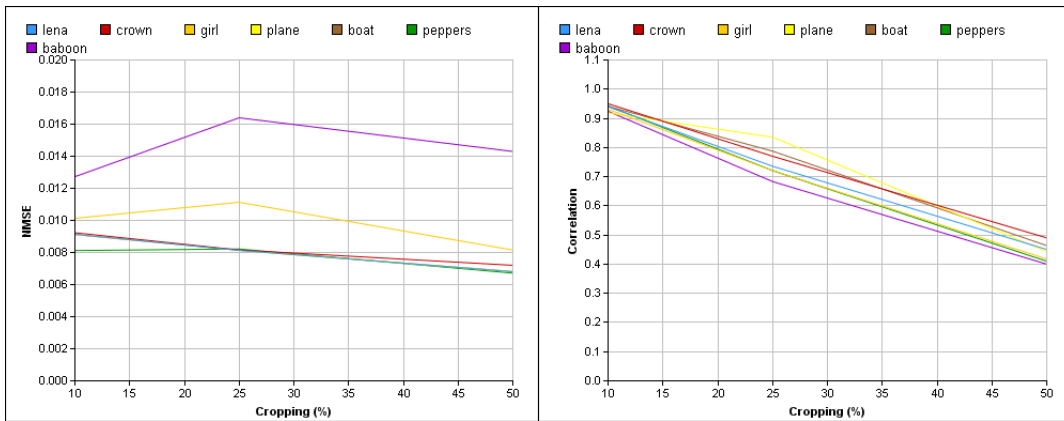


Figure 38: Test images comparison - cropping attacks



Images	"lena"			"crown"			"girl"		
Region	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
bottom	0.0148	0.93	100	0.0148	0.94	100	0.0121	0.91	100
center	0.0149	0.89	100	0.0150	0.89	100	0.0113	0.87	100
top	0.0148	0.94	90	0.0159	0.95	90	0.0102	0.93	90

Table 37: Results after collage attacks

Images	"plane"			"boat"			"peppers"			"baboon"		
Region	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
bottom	0.0148	0.10	100	0.0149	0.93	100	0.0140	0.91	100	0.0153	0.94	100
center	0.0149	0.89	100	0.0150	0.89	100	0.0125	0.86	100	0.0151	0.85	100
top	0.0148	0.96	90	0.0150	0.92	90	0.0140	0.91	90	0.0147	0.95	90

Table 38: Results after collage attacks (continued)

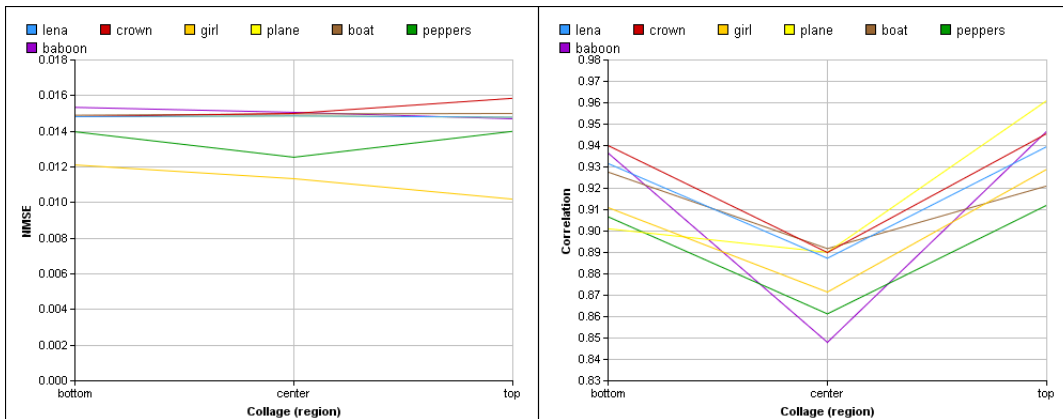


Figure 39: Test images comparison - collage attacks

Images	"lena"			"crown"			"girl"		
Region	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
bottom	0.0119	0.94	100	0.0121	0.94	100	0.0109	0.92	100
top	0.0116	0.92	0	0.016	0.93	0	0.0109	0.91	0

Table 39: Results after clipping attacks

Images	"plane"			"boat"			"peppers"			"baboon"		
Region	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)	nmse	r	ascii (%)
bottom	0.0119	0.91	100	0.0120	0.93	100	0.0132	0.91	100	0.0102	0.76	100
top	0.016	0.93	0	0.0116	0.92	0	0.0127	0.89	100	0.0100	0.73	0

Table 40: Results after clipping attacks (continued)

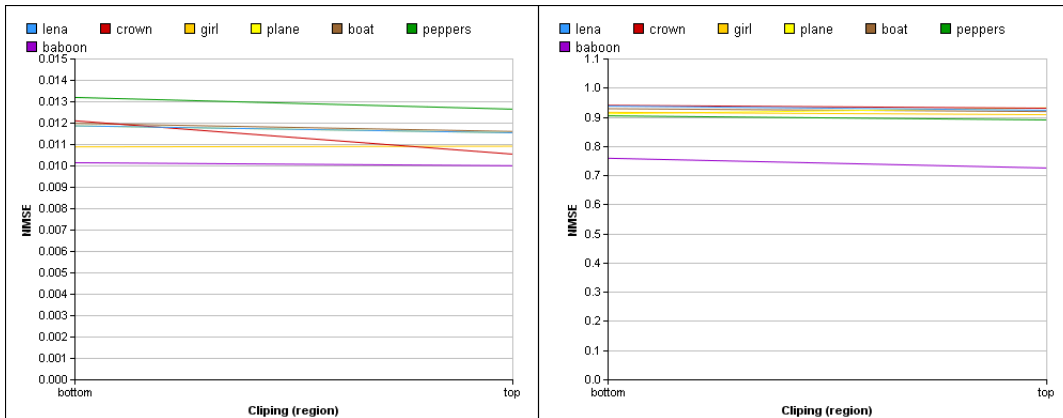


Figure 40: Test images comparison - clipping attacks