AN ADAPTIVE PIXEL VALUE ORDERING BASED REVERSIBLE WATERMARKING SCHEME FOR IMAGE AUTHENTICATION

by

Gabriel Melendez Melendez

Thesis submitted as partial requirement for fulfilment of the degree of

MASTER IN SCIENCE, COMPUTER SCIENCE

at

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)

November, 2018

Santa María Tonantzintla, Puebla, CP 72840

Advisors:

PhD. René Armando Cumplido Parra

PhD. Alicia Morales Reyes

Computer Science Department, INAOE

Agradecimientos

Esta investigación fue realizada gracias al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo otorgado a través de la beca No. 450812 y al Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE) por todas las facilidades brindadas a lo largo de estos dos años, expreso mi más sincero agradecimiento a ambas instituciones.

Las palabras no son suficientes para agradecer a mis asesores, el Dr. René Armando Cumplido Parra y la Dra. Alicia Morales Reyes, quienes en todo momento me brindaron su acertada orientación y compartieron su conocimiento para finalizar en buenos términos este trabajo de investigación.

Agradezco a la Dra. Lil María Xibai Rodríguez Henríquez, a la Dra. Kelsey Alejandra Ramírez Gutiérrez y al Dr. Ignacio Algredo Badillo, por sus comentarios, críticas y observaciones realizadas a esta tesis.

A mis amigos: Jessica, Daniel, Bruno y Javier, por todos los buenos momentos que compartimos dentro y fuera del instituto. A David, Ania y la familia Musa, agradezco su apoyo y consejos brindados a lo largo de mi formación.

Finalmente, agradezco a mi familia por su infinito amor, comprensión y apoyo incondicional en todo momento.

Contents

List of Figures

List of Tables

Resumen

La protección de información dentro de un sistema de comunicación es de suma importancia ya que un contenido digital puede caer en manos no autorizadas durante su transmisión. El uso de marcas de agua para ocultar información en imágenes es un tema por el que los investigadores han apostado durante los últimos años. Un esquema de marca de agua convencional distorsiona permanentemente la imagen que se utiliza para transportar la información oculta. Proponer esquemas reversibles se vuelve importante en aplicaciones donde el uso de la imagen original es indispensable para la toma de decisiones en el extremo receptor de la comunicación, ya que proporcionan la ventaja de extraer la información oculta y eliminar la distorsión introducida, recuperando la imagen original usada para transportarla.

En este trabajo se desarrolla PVO-GA (Pixel Value Ordering Genetic Algorithm), un esquema de marca de agua reversible para la autenticación de imágenes usando la técnica de ordenamiento de pixeles para la inserción de la marca de agua. El esquema propuesto mejora las dos características que deben considerarse en la evaluación de un esquema reversible: imperceptibilidad y capacidad de inserción, respecto a trabajos en el estado del arte. Dado que el espacio de soluciones es expandido, se integra un algoritmo genético evolutivo para la búsqueda de los parámetros adecuados del esquema propuesto.

Abstract

Protecting information within communication systems is quite important since a digital content may be manipulated by unauthorized people during transmission. Image watermarking is a hiding information technique that has been thoroughly investigated by the cybersecurity community in recent years. A traditional watermarking scheme permanently distorts an image in order to hide information, which is why reversible watermarking schemes (RWS) emerged for application domains where original image is compulsory when involving decision making at the receiver end of the communication. Reversible watermarking schemes have the capability to extract the hidden information and remove the introduced distortion, such that original image is recovered.

In this work, a reversible watermarking scheme for images authentication called PVO-GA (Pixel Value Ordering Genetic Algorithm), which is based on pixel value ordering technique to embed the watermark is developed. The proposed scheme improves the two important characteristics in reversible watermarking, such as imperceptibility and embedding capacity reported in state of the art works. A fine grained genetic algorithm is incorporated in order to optimize the insertion process since solutions space is expanded. The proposed genetic algorithm finds the optimal/near optimal PVO-GA parameters, which provide an effective trade-off between imperceptibility and embedding capacity properties.

1

Introduction

In recent years, the infrastructure for the use of local networks and the Internet has increased significantly. Nowadays digital content distribution is carried out easily, so an image, song, video or document can be transferred among users in only a few seconds. Leaving aside disadvantages arising from this growth, the appropriate use of this infrastructure causes that more and more organizations, such as banks, schools, hospitals even governments, take advantage of moving several operations to an electronic environment. Thus, protecting the information that is transferred when these communication systems are interacting is a situation that must be considered.

It is necessary to guarantee at receiver end of the communication, that received digital content is the one originally sent from the transmitter end, this security service is known as integrity. In this way, received information is ensured of not being manipulated during transmission process. Additionally, it is also important to provide authentication, if this security service is provided, the receiver is assured that the received information originates from a trusted party.

Image watermarking gives a solution to the above mentioned problems. A traditional digital watermarking system embeds the information that is going to be protected within an image, slightly modifying its features, such that this information is unnoticed by malicious people. Then a protected watermarked image is sent. Finally, at the receiver end

of the communication, the hidden message is extracted from the watermarked image for its interpretation. However, the watermarked image that serves as cover to transport the hidden information remains altered. There are some application domains, such as medical or military image processing, where it is necessary to recover the original image after data extraction, since it is essential for decision making. The development of reversible watermarking schemes is important because they are able to recover the hidden message and also to rebuild digital content used to transmit the information, using control information generated during the insertion process, as it is shown in Fig. 1.1.

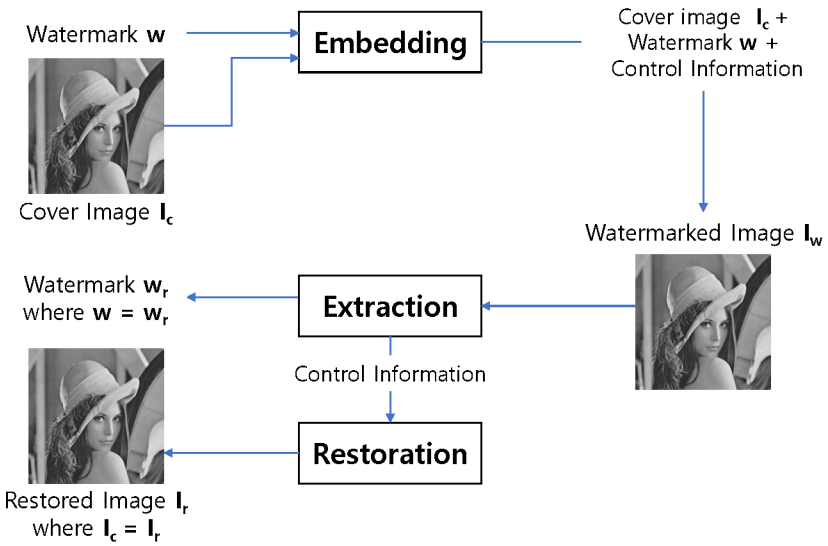


Figure 1.1: Reversible watermarking scheme.

During the design of a reversible watermarking scheme, it is important to consider some properties in the insertion and extraction processes. The embedding capacity and imperceptibility are opposite features in a watermarking system (Cox et al., 2008), so when one feature is being improved the other is compromised and therefore, reversible watermarking research focuses on finding a suitable trade-off between these properties (Khan et al., 2014b), while ensuring the original signal reversibility. On the one hand, the imperceptibility feature is evaluated using metrics such as Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR) or Structural Similarity Index (SSIM), which provide objective information about how much distortion is embedded into a watermarked image,

compared with the original cover image. On the other hand, embedding capacity property is evaluated by indicating the number of inserted bits.

Watermarking systems provide potential benefits over other techniques. First, watermarks are imperceptible, so malicious people could not realize the existence of hidden information. Second, the watermark travels with its associated cover work. Finally, the watermarks undergo the same transformations as its associated cover work, so it could be possible to learn something about the applied attacks.

The performance of reversible watermarking schemes depend on their parameters. It is desired to develop adaptive schemes since proving different parameters using the same embedding method could improve the imperceptibility or embedding capacity. For instance, Xuan et al. (2002, 2004, 2005) embed the information into the low frequency coefficients of Integer Wavelet Transform (IWT), these coefficients are compressed using a threshold T_c in order to reduce the distortion caused after embedding. If T_c is increased more coefficients are compressed, however control information increases causing imperceptibility to be affected. Testing all possible parameters of the method is time consuming. Evolutionary computation has been used to deal with this issue (Arsalan et al., 2012, 2017; Naheed et al., 2014). These intelligent techniques find an optimal balance between embedding capacity and imperceptibility, which are the two important features in reversible watermarking schemes. The same happens with Pixel Value Ordering (PVO) based methods (Li et al., 2013; Peng et al., 2014; Wang et al., 2015), where performance depend on parameters as block size and at least one threshold. When block size is increased, imperceptibility levels are improved, however embedding capacity is decreased. When block size is decreased, the opposite occurs. Proving all possible block sizes and thresholds is computationally expensive, which motivated to expand the solutions space and develop a genetic algorithm to intelligently find an optimal set of parameters, which provide a good trade-off between imperceptibility and embedding capacity.

1.1 Problem statement

Information embedding within a signal, particularly within an image, alter its features causing distortion. Although introduced distortion may be minimal, in most watermarking schemes the signal used to hide information does not recover its original form after extraction process.

Certain application domains, such as medical or military, prohibit permanent loss of the signal, they are sensitive to embedded distortion since using the original signal is indispensable for decision making. It is also important to consider the embedding capacity offered by the method since when it is try to improve imperceptibility feature, the embedding capacity is decreased. Therefore, it is necessary to propose reversible watermarking schemes to ensure original signal recovery after hidden message extraction. Moreover, since the watermarked image could be manipulated during transmission, it is necessary to inform at the receiver that the received digital content is authentic and it was sent by a trusted user. In this way, restored image and recovered information from received watermarked image can be certainly used.

1.2 Motivation

Easy digital content distribution leads organizations to do their operations electronically, thus protecting the information used in these operations is a task that must be further investigated. Most digital watermarking schemes are not reversible, therefore it is necessary to propose reversible schemes to reach all possible application domains in which watermarking schemes are involved.

There exists improvement areas for current reversible watermarking schemes like provide tampering localization or find a trade-off among watermarking properties as imperceptibility and embedding capacity (Khan et al., 2014b).

1.3 Objectives

1.3.1 Principal objective

To develop an adaptive reversible watermarking scheme for image authentication that allows to find an effective trade-off among embedding capacity and imperceptibility.

1.3.2 Specific Objectives

- To review and select reversible watermarking embedding and extraction algorithms.
- To develop a reversible watermarking scheme allowing image authentication at the receiver end of the communication.
- To develop an algorithm that allows to find the optimal parameters of the scheme, which provide the best trade-off among watermarking properties.
- To evaluate the performance of the proposed scheme over existing schemes using benchmark images and metrics reported in the state of the art.

1.4 Methodology

The following project stages are proposed to achieve the objectives mentioned above.

- **Review and selection of methods:**
  During the first stage of the project, watermarking methods are reviewed considering reversible schemes. Data insertion and extraction algorithms that are used in the proposed scheme, are selected according to results reported in previous works, including embedding capacity and imperceptibility properties.
- **Design and implementation:**
  Selected algorithms are implemented to evaluate their performance in terms of

PSNR and number of inserted bits.

During selected algorithms implementation, areas of opportunity are identified in order to improve their performance.

The novel proposed scheme is developed using modified insertion and extraction algorithms.

- **Evaluation and analysis of results:**
  Benchmark images are used to embed watermarks of different size. Watermarked images are evaluated using PSNR and inserted number of bits.

  Finally, the proposed scheme is compared against related works, considering embedding capacity and imperceptibility properties.

1.5 Document organization

The rest of this document is organized as follows. In Chapter 2, theoretical foundations for the understanding of this thesis are presented. Chapter 3 introduces several significant and closely related works in the area. Papers are classified into three different categories, according to watermarking insertion algorithms. The proposed reversible watermarking scheme is presented in detail in Chapter 4. Experimental results of proposed scheme together with a comparison against some state of the art works are presented in Chapter 5. Finally, conclusions and future work of this research thesis are mentioned in Chapter 6.

2

Background

This chapter provides the reader with the basics for understanding the topics addressed throughout this thesis. First, digital watermarking subject is explained including main features, applications and the standard metrics used to evaluate reversible watermarking schemes performance. Image authentication is provided by using a Message Authentication Code (MAC), then, it is explained what a hash function is and how it can be used to provide authentication. Finally, the chapter concludes by mentioning conceptual basis of evolutionary computation, which is used in proposed scheme.

2.1 Digital Watermarking

In recent years, progress related to communication technologies allows information exchange in a simple fashion. Thus, public and private organizations move part of their operations to an electronic environment. Protecting information in those communication systems is a task that can be addressed using security technologies such as cryptography, steganography or digital watermarking (Cox et al., 2008).

Digital watermarking is a research area which aims to embed a secret message into a digital content known as cover work, which can be an image, audio, video, document, etc. The message is a digital code known as watermark. Sometimes watermarking and steganography terms can be confused since both are information hiding techniques.

However, there are some properties that identify each one. The most important difference is that in watermarking systems, the message provides useful information about its associated cover work, unlike steganographic systems where the message is not related to it (Cox et al., 2008). In this thesis, an image is used as cover work, so a cover image with an embedded watermark is known as watermarked image.

Research interest related to digital watermarking techniques started in the 90s. Watermarking systems consist of two important processes: embedding and extraction. Fig. 2.1 shows a common digital watermarking system. Let w be a watermark and I_c the cover image in which w will be inserted using the embedding algorithm W_E, the corresponding watermarked image is defined to be $I_w = W_E(I_c, w)$. Typically, I_w is transmitted or recorded. At the receiver, I_w is received and the watermark w_r is obtained using the corresponding extraction algorithm W_D. So watermark is extracted as: $w_r = W_D(I_w)$, where $w_r = w$.

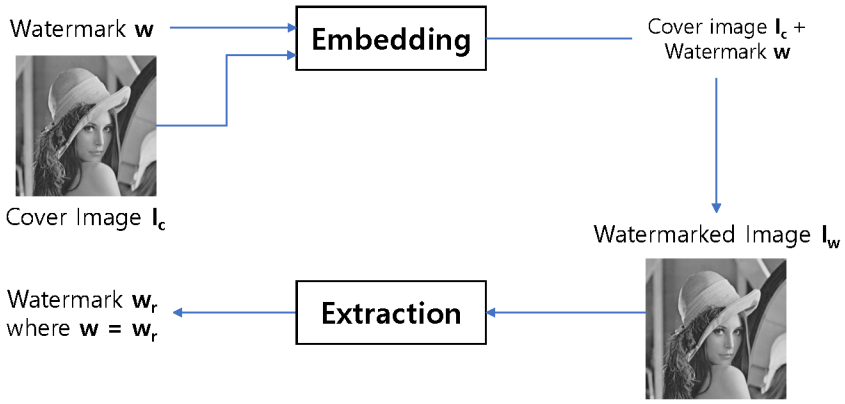


Figure 2.1: A general image watermarking system.

Information embedding produces distortion within a cover image. Although introduced distortion can be minimal, in certain application domains such as medical or military, using the original image is mandatory for decision making and a permanent loss in the signal is prohibited (Shi et al., 2016). Reversible watermarking schemes provide a solution to this problem since the embedding process is strategically performed, so that at the receiver end of the communication, it is possible to extract the watermark and also the

watermarked image can be recovered to its original form after extraction and restoration stages using control information generated in the embedding process, see Fig. 1.1. Let RW_E be a reversible embedding algorithm. For a given cover image I_c and a watermark w, its corresponding watermarked image is defined to be $I_w = RW_E(I_c, w)$. At the receiver, both I_c and w are obtained by applying the corresponding reversible extraction algorithm as: $(I_r, w_r) = RW_D(I_w)$, where $I_r = I_c$ and $w_r = w$.

2.1.1 Watermarking properties

Digital watermarking schemes must meet some requirements. Depending on the application for which the scheme is designed, some properties are prioritized over others. Below, the main properties that should be considered when designing a watermarking scheme are described (Cox et al., 2008).

- **Imperceptibility** - This property is very important since invisibility gives advantages to watermarking over other techniques to keep the information secret. Whenever a watermark is embedded into the cover image, it causes distortion. Watermarked image and cover image must be perceptually similar, thus introduced distortion should be the lowest possible.

- **Embedding capacity** - This property refers to the maximum number of information bits that can be embedded into the cover image.

- **Robustness** - Resistance level to attacks.

In general, reversible watermarking methods are considered fragile, it means that they are not able to resist any possible attack (Shi et al., 2016). In contrast with digital watermarking schemes, where robustness is considered a top priority. Therefore, improvement of reversible watermarking schemes is primarily based upon making a good imperceptibility versus embedding capacity trade-off (Khan et al., 2014b).

2.1.2 Applications

Watermarking techniques can be used in a wide variety of application domains since it is possible to hide additional information as part of the watermark. Watermarking is distinguished from other techniques by three important characteristics (Cox et al., 2008).

- Information embedding is performed in a way that cover media is minimally altered to avoid attracting the attention of malicious people.
- Watermark travels together with its associated cover work without the need of sending each item separately.
- The watermark undergoes the same transformations as the content in which it is embedded. So it is possible to learn something about transformations applied to the cover content, by analyzing the extracted watermark.

Therefore, watermarking schemes are ideal for applications such as:

- **Broadcast monitoring** — In 1997, Japanese TV advertisers detected that thousands of their commercials were never shown on air despite having paid for them. Human observers were used to watch the broadcast and record what they saw or heard, however it was costly and imprecise. Digital watermarking is an alternative to solve this problem. Identification information is embedded into TV and radio commercials to monitor transmissions using watermark detectors, which is less expensive and more effective for the advertisers.
- **Content Authentication** — Digital works could be easily altered by malicious people. Watermarking is used to verify the integrity of its associated cover work. Authentication is the process of identify if the received digital content is exact as it was sent (Saini and Shrivastava, 2014). This process can be performed using cryptographic techniques such as MACs or digital signatures, however as mentioned

above, watermarking offers potential benefits. In this case, watermarks are embedded into the cover work itself, removing any need of store or send the work and their authentication information separately.

- **Owner identification** — Digital content authors must protect their works (images, videos or songs) to claim them when copyrights are infringed. Watermarking schemes are used to embed the owner information into the works. By extracting the watermark, it is possible to identify the owner of a specific digital work.

- **Transaction tracking**- In this case, watermarks are embedded into a specific work in every transaction taking place in the copy history of the work itself. For instance, legal copies of a movie could be watermarked using a different watermark in each copy. If someone leaks a copy of the movie to the Internet, it could be possible to identify the user who leaked it by extracting the watermark.

- **Copy control**- Watermarking is also used to prevent illegal actions as copying of digital works. For example, a watermark could be embedded into a song or movie to notify a recorder(whit a watermark detector) that a copy is legal.

2.2 Evaluation metrics

Reversible watermarking schemes are evaluated considering imperceptibility and embedding capacity properties. Metrics to evaluate each property are described next.

2.2.1 Imperceptibility evaluation

As mentioned in previous sections, embedding process causes distortion in the cover image, introduced distortion is noticeable when watermarked image quality is evaluated. Although sometimes a subjective analysis between cover and watermarked images is enough, in most cases it is necessary to use numerical metrics to measure distortion levels caused.

MSE

Distortion introduced in watermarking embedding process, can be obtained by computing the difference or distance between the original cover image and the watermarked image. Mean Squared Error or MSE is the simplest distortion metric to measure it (Eskicioglu and Fisher, 1995). The MSE is computed using Eq. 2.1.

$$MSE(I_c, I_w) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (I_{c_{i,j}} - I_{w_{i,j}})^2 \tag{2.1}$$

Where:

I_c = Cover image

I_w = Watermarked image

N, M = Images size

If there are no differences between compared images, MSE will return 0, so a MSE closer to 0, implies a better watermark imperceptibility.

PSNR

Peak Signal-to-Noise Ratio or PSNR is the most commonly used metric to measure introduced distortion in decibels between cover and watermarked images (Khan et al., 2014b). This metric is based on the MSE metric.

For a given cover image I_c and a watermarked image I_w of size $N \times M$, the PSNR between I_c and I_w is defined as Eq. 2.2.

$$PSNR(I_c, I_w) = 10log_{10} \frac{MAX^2}{MSE(I_c, I_w)} \tag{2.2}$$

Where MAX is the maximum value that image pixels can take, so for an 8-bit deep image this value is 255.

When I_c and I_w images are identical, MSE metric returns 0, therefore PSNR goes to infinite, so a higher PSNR value means a better watermark imperceptibility.

SSIM

Structural Similarity Index or SSIM is another objective metric usually used to evaluate image quality (Hore and Ziou, 2010).

SSIM is considered to be correlated to the human visual system (HVS), since this metric combine three components such as loss of correlation, luminance distortion and contrast distortion.

For a given cover image I_c and a watermarked image I_w, the SSIM measure between I_c and I_w is defined using Eq. 2.3.

$$SSIM(I_c, I_w) = l(I_c, I_w)c(I_c, I_w)s(I_c, I_w) \tag{2.3}$$

where:

$$l(I_c, I_w) = \frac{2\mu_{I_c}\mu_{I_w} + C_1}{\mu_{I_c}^2\mu_{I_w}^2 + C_1} \tag{2.4}$$

$$c(I_c, I_w) = \frac{2\sigma_{I_c}\sigma_{I_w} + C_2}{\sigma_{I_c}^2\sigma_{I_w}^2 + C_2} \tag{2.5}$$

$$s(I_c, I_w) = \frac{\sigma_{I_cI_w} + C_3}{\sigma_{I_c}\sigma_{I_w} + C_3} \tag{2.6}$$

Eq. 2.4 represents luminance mean. Eq. 2.5 is the contrast measured by standard deviation. Eq. 2.6 is the structure comparison, which measures correlation coefficient between cover and watermarked images. Finally, C_1, C_2 and C_3 are positive constants to avoid null denominator (Hore and Ziou, 2010).

SSIM metric is within $[0, 1]$ range. If there is no correlation between compared images, SSIM function will return 0. On the other hand, if compared images are identical, SSIM function will return 1. A SSIM value close to 1 corresponds to a better image quality.

2.2.2 Embedding capacity evaluation

Embedding capacity refers to a watermarking feature, which indicates the maximum number of information bits that can be embedded within a cover image.
Commonly this property is evaluated by directly indicating the number of embedded bits, however, it is possible to compute the number of bits per pixel or BPP that are inserted into the cover image (Cox et al., 2008).
Given a cover image I_c of size $N \times M$ with a maximum embedding capacity of P bits, the BPP is computed using Eq. 2.7.

$$BPP(I_c) = \frac{P}{N \times M} \tag{2.7}$$

It is desired to improve both the imperceptibility and the embedding capacity, however the more the number of inserted bits, the introduced distortion increases. So the research focuses on finding a trade-off among these two properties.
The proposed scheme aims to perform image authentication. This process is accomplished by using a Message Authentication Code or MAC, which is obtained using a keyed hash function. In next section it is briefly explained what a hash function is.

2.3 Hash function

A cryptographic hash is a one-side function that takes as input a bit string of arbitrary length and returns a fixed-size bit string called digest (Ferguson et al., 2011). Let H be a hash function and s a binary string of arbitrary length, the corresponding digest is defined as $d = H(s)$.
Typically, d goes from 128 to 1024 bits, which are far fewer bits, compared to the thousands or millions of bits that could be in the input string. Hash functions must be resistant to collisions, that is to say, to avoid producing the same digest for two different input strings. However, a hash function is not collision free, but it is desired that if a

collision exists, it must be very hard to find, therefore a cryptographic hash function can provide assurance of data integrity (Stinson, 2005).

A keyed hash function can be used to obtain a Message Authentication Code or MAC. In this case, suppose that users U_t and U_r share a secret key k to make d dependant on the key. Then, an authentication code for s can be computed by U_t as $d_{MAC} = H(k, s)$. Later, the pair (s, d_{MAC}) can be transmitted from U_t to U_r over an insecure channel. When U_r receives the pair (s, d_{MAC}), he can verify if $d_{MAC} = H(k, s)$. If this condition holds, s and d_{MAC} were not altered by an adversary during transmission and U_r is assured that the message s originates from U_t.

Keyed hash functions have been used in the reversible watermarking subject to solve the problem of content authentication, since a minimal modification in the input causes the produced MAC to change drastically, even if only one bit is modified. For example, Fridrich et al. (2001) and Vleeschouwer et al. (2003), use MD5 hash function, while Honsinger et al. (2001) uses SHA-1 hash function. They compute a MAC to be embedded into cover image itself. In this way, after extraction and recovery process, the restored image is used to produce a new MAC using the same key as in embedding and then, it is compared with the extracted one.

Reversible watermarking schemes performance depend on their parameters. It is desired to provide an effective trade-off between imperceptibility and embedding capacity. However proving all possible parameters of one specific method could be time consuming. Evolutionary computing has been used to deal with this problem, which motivated to develop the Genetic Algorithm(GA) proposed in Chapter 4. In next section it is explained how a GA works.

2.4 Evolutionary computing

Evolutionary computing is a set of algorithmic techniques that belong to the soft computing category and has been investigated for some decades. It is possible to develop computational algorithms simulating biological behaviors. Evolutionary computing is

inspired in the evolutionary process to offer problem solutions, establishing an environment populated by individuals who strive to survive and reproduce (Bäck, 1996). A set of individuals is defined according to applied genetic operators to generate new individuals. The goal is to generate improved individuals every time they interact, following an objective function known as fitness function.

Evolutionary computing is suitable for search optimization, modelling and simulation problems since the interaction of individuals is done by heuristic functions designed to approximate optimal/near optimal solutions (Eiben et al., 2003).

The most widely known type of evolutionary algorithm is the Genetic Algorithm or GA. Using a GA to solve a problem implies that the designer of it adopts the main decisions of a general GA framework and only specifies the details. In next section it is explained how a GA should be designed.

2.4.1 Genetic Algorithm

A GA is an stochastic search technique typically used for optimization or searching problems and inspired by the natural evolution process which deploys selection, mutation and recombination among individuals (Eiben et al., 2003).

In a genetic algorithm, a solution is represented by a structure known as chromosome or individual, which contains genes. A set of individuals is known as a population, which interacts to evolve individuals by applying genetic operators in order to produce solutions of better quality. The process is repeated for a number of generations or until an individual reaches an established fitness value. Fig. 2.2 shows a general scheme of a traditional genetic algorithm.

There is not an only way to implement a genetic algorithm, since it can be adapted to many application domains, because of possible representations variety and the use of specific operators. However, some standard stages must be considered when designing it (Eiben et al., 2003). Principal stages are briefly explained below.

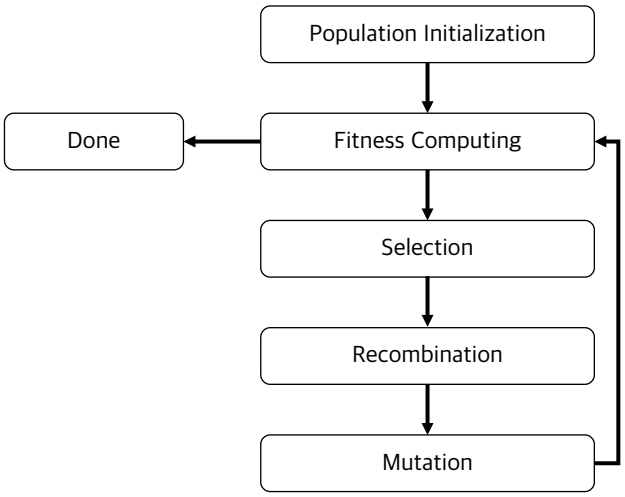


Figure 2.2: A common Genetic Algorithm flowchart.

Individuals representation

The first stage of building a genetic algorithm is to propose a right representation of a possible solution to the problem. This could be considered the most important decision to design a good genetic algorithm, since it must be ensured that proposed representation allows to reach all valid solutions to the problem being solved.

Binary representation is the most basic and one of the simplest representations used in the subject. It consists of a string of binary digits (Eiben et al., 2003). In binary representation it is important to decide the string size which represents the chromosome, beside mentioning its meaning. However, binary representation is not suitable in many problems, so other type of representations had arisen. It is possible to represent a chromosome using integer and floating point representations. Integer representations are suitable to the problem of finding optimal values for a set of variables that all takes integer values, which might be restricted to a finite set or unrestricted, i.e., any possible integer value is permissible. Finally, floating point representations are used in problems where the chromosome values come from a continuous distribution.

Parent selection

The selection operator is responsible for choosing two or more individuals that will be used to produce offspring individuals. Commonly, parent selection is done considering the fitness value of each individual. Ranking selection sorts the population on the basis of fitness, and then selects those individuals with greater fitness value. Ranking selection requires knowledge of the entire population, which could be computationally expensive (Gen and Cheng, 2000).

Tournament selection gives a solution to above problem. In this case, a set of individuals are picked from the entire population randomly. So only picked individuals are ranked and later, the best ones are selected for crossing over.

Recombination

Recombination also known as crossover is another variation operator which aims to create new individuals from information contained within two or more parent individuals. Crossover is an essential operator in genetic algorithms (unlike other evolutionary algorithms) since it is the primary mechanism that provides diversity into population. Crossover operator takes as input two or more parent individuals to exchange their information creating one or more offspring individuals (Eiben et al., 2003).

In binary and integer representations, the information exchange is typically done by randomly defining one or more crossover points for parents partition. Thus, offspring individuals are created by taking alternative segments from parents. In this way offspring individuals inherit parent characteristics, which would result in solutions with better quality than previous ones.

Mutation

Mutation operator is considered a background variation operator that takes as input an individual, which is slightly modified, typically in a random form to create the mutated individual (Eiben et al., 2003).

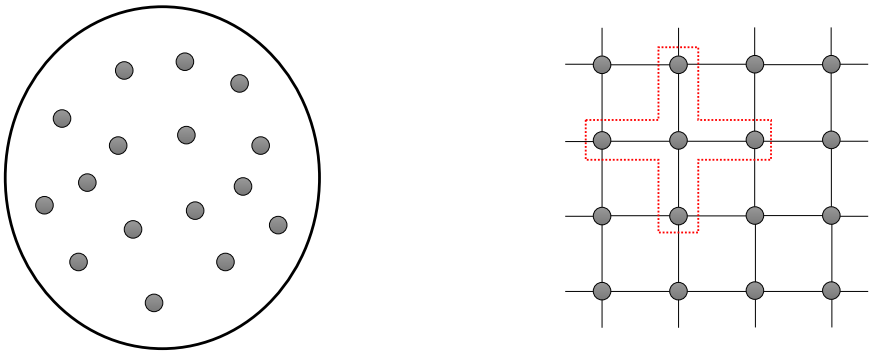
On the one hand, when a binary representation is used, it is common to select some bits to be flipped based on a probabilistic distribution. On the other hand, when an integer representation is used, mutation can be applied using random resetting or creep mutation. Random resetting is used to choose a new permissible value in a random form. Creep mutation works by adding a random positive or negative value to a selected chromosome gene within some allowed range.

Population dynamics

Population is a set of individuals that will evolve by applying genetic operators. It is considered the second most important element of the evolutionary process when designing a genetic algorithm, after individuals representation.

Generational and steady-state models are two different population models identified in the literature. In the generational model, the population contains N_i individuals, every generation N_i offsprings are created by applying genetic operators, after the entire population is replaced by its offsprings. In the steady-state model, only a percentage of the entire population is replaced (Eiben et al., 2003).

The spatial structure of a population is something that needs to be considered when designing a genetic algorithm. Fig. 2.3 shows two commonly used structures. Fig. 2.3a shows a panmictic population, also known as panmixia (Alba and Dorronsoro, 2009); in a panmictic population, the genetic operators are applied to the whole population. Fig. 2.3b shows a decentralized population. Genetic algorithms that use this population structure are known as cellular genetic algorithms (cGA) since a small neighborhood concept is exploited.



(a) Panmictic population (b) Decentralized population

Figure 2.3: Structural space population examples.

It has been empirically demonstrated that cellular genetic algorithms perform more efficiently than genetic algorithms using a panmictic population, because exploration[1] and exploitation[2] of landscapes are carried out in a different way, providing diversity[3] (Alba and Dorronsoro, 2009).

Evaluation function

A genetic algorithm is responsible for individuals interaction to find an optimal/near optimal solution for a given problem. Searching is based on a metric assigned to every individual in the population that measure their quality. The evaluation function is also known as fitness function. Its role is essential since it leads the population search and is also the guide for operators such as parent selection and replacement mechanism (Eiben et al., 2003).

Replacement mechanisms

Since population size is a fixed number of individuals, and after genetic operators application new individuals are created, it is necessary to have a process responsible for

[1]The generation of new individuals in untested regions of the search space(Eiben et al., 2003).
[2]The concentration of the search near to good solutions(Eiben et al., 2003).
[3]Different solutions present in the population(Eiben et al., 2003).

deciding which individuals will survive in the next generation and which ones will be excluded. There are some replacement strategies that have been proposed in the literature, considering individuals information such as age or fitness value (Eiben et al., 2003). Typically, replacement is applied based on fitness value, following strategies such as next:

- Replace P_i individuals with lower fitness value, with P_i new individuals with best fitness value.
- Replace an individual only if its offspring has a better fitness value.

2.5 Summary

In this chapter, conceptual basis to introduce the main areas involved in the proposed reversible watermarking scheme for image authentication were exposed. Digital watermarking fundamentals were discussed. Reversible watermarking differences as an extension to digital watermarking were also explained. Moreover, main features of watermarking schemes were described, with an extended description of the two important properties in reversible watermarking field, which are imperceptibility and embedding capacity. Measures for performance evaluation in reversible watermarking schemes were presented. Finally, evolutionary computation was addressed focusing on genetic algorithms, which will be used in proposed solution. The next chapter carries out a detailed review on state of the art related work.

3

State of the art

This chapter describes the principal image reversible watermarking techniques identified in the literature. Initially, the pioneer works are exposed. Then, some papers are classified into three main groups, according to the technique they use to insert the watermark such as compression, histogram modification, and error expansion based methods. Pixel Value Ordering (PVO) technique, in which the proposed scheme is based, appears as an extension of prediction error expansion methods, so a detailed explanation to understand how it works is carried out.

3.1 Reversible watermarking methods

Reversible watermarking schemes have received increasing research interest in recent years (Khan et al., 2014b). The first solution when addressing the hiding information problem in a reversible way was proposed in Barton (1997). The patent exposed a method to embed authentication information within a digital data block. So an image is partitioned in pixel blocks to compute a digital signature from the high-order bits. The LSBs (Least Significant Bits) of block pixels are modified to embed a bit string composed by a block's digital signature and a compressed version of the original bits that were altered. The insertion of compressed original bits allows reversibility while digital signatures provide authentication.

Honsinger et al. (2001) proposed a reversible technique based on $modulo - P$ addition, which is computed using the watermark bit b and the pixel value Pix in which b will be inserted to obtain the watermarked pixel $Pix_w = (b + Pix) modulo - P$. In the case of a p-bit deep image, $P = 2^p$ (e.g. for an 8-bit image, $P = 256$). Modulo-P addition is used to make sure that watermarked image values will always be in the allowed range, solving underflow[1] and overflow[2] problems. However, modulo-P addition causes salt and pepper noise[3] on pixel values near to 0 and 255. In this scheme, they also used a digital signature and control information to ensure image authentication and reversibility.

Most reversible watermarking techniques use control information since their function arguments may vary. Control information is embedded as a part of the watermark. This information is extracted and used to recover the original image at the receiver end of the communication, so generated control information should be the minimal possible (Khan et al., 2014b).

Since then, many reversible watermarking schemes have been proposed, using new, extended or improved versions of existing watermarking embedding techniques. Research on reversible watermarking subject mainly focuses on finding a trade-off between imperceptibility and embedding capacity properties.

In the next sections, the most relevant reversible watermarking works are presented. They are classified into three identified groups: compression, histogram modification, and error expansion based methods.

3.1.1 Compression based methods

In order to reach image reversibility, embedding space must be extended to include control information generated during the insertion process. Compression based methods try to find that space by compressing parts of a cover image. Table 3.1 shows compression based methods identified in the literature.

[1] A pixel exceeds the upper bound.

[2] A pixel exceeds the lower bound

[3] A certain amount of pixels in the image become either black or white, specially in the darkest or lightest areas of an image, so watermark imperceptibility could be affected.

Fridrich et al. (2001) proposed a reversible scheme based on the spatial domain to embed the watermark. In order to find additional space, bit-planes are identified to be loosely compressed. Then, a MAC is computed using MD5 hash function to be inserted together with a compressed version of the original bits that were modified, so additional information can not be embedded.

Author	Year	Domain	Authentication	Technique
Barton	1997	Spatial	Yes	LSB modification
Honsinger et al.	2001	Spatial	Yes	Add, subtract, modulus
Fridrich et al.	2002	Spatial / Frequency	Yes	LSB, DCT
Xuan et al.	2002	Frequency	No	IWT
Xuan et al.	2004	Frequency	No	Companding IWT
Yang et al.	2004	Frequency	No	Companding DCT
Xuan et al.	2005	Frequency	No	Companding IWT
Usman and Khan	2009	Frequency	No	DCT
Arsalan et al.	2011	Frequency	Yes	IWT, GA, Companding
Memon et al.	2011	Frequency	Yes	IWT, Companding
Ko et al.	2012	Frequency	No	DCT
Mohammed and Khoo	2013	Frequency	No	IWT
Arsalan et al.	2015	Frequency	Yes	IWT, GP, Companding
Nguyen et al.	2016	Frequency	Yes	DWT
Shih and Zhong	2016	Frequency	Yes	DCT, ROI
Weng and Pan	2016	Frequency	No	IHWT
Parah et al.	2017	Spatial	Yes	Pixel to block

Table 3.1: Methods based on compression.

Usually, compression based methods work in the frequency domain, using coefficients of transforms as Discrete Cosine Transform (DCT) Yang et al. (2004); Usman and Khan (2010); Ko et al. (2012); Shih and Zhong (2016), Discrete Wavelet Transform (DWT) Nguyen et al. (2016) or Integer Wavelet Transform (IWT) Xuan et al. (2002, 2004, 2005); Memon et al. (2011); Weng and Pan (2016). Initially, the cover image is transformed into frequency domain by computing a transform. Later, coefficients are compressed to create the embedding space. Every watermark bit is appended to the compressed coefficients in their binary representation. Watermarked image is obtained by applying the inverse transform. At the receiver end of the system, secret information bits are obtained by com-

puting corresponding transform to the watermarked image, LSB of every watermarked coefficient is extracted. Then, coefficients are expanded to their original values to ensure reversibility. Finally, the original image is recovered by applying the inverse transform to expanded coefficients.

Recently, Arsalan et al. (2012, 2017) proposed using evolutionary algorithms to find a threshold set to compress IWT coefficients. By adopting intelligent search, coefficients of different areas within the image can be compressed using different thresholds, unlike Xuan et al. (2005), where all coefficients are compressed using only one threshold. Using a genetic algorithm or genetic programming allows testing different parameter configurations of the algorithm, without the need to carry out an exhaustive search into the whole solutions space.

As seen above, many compression based methods have been proposed, however there are some identified drawbacks. On the one hand, the works that uses the spatial domain, provide a smaller embedding capacity compared with the works that embed the watermarks in the frequency domain. On the other hand, the methods that use the frequency domain compress all the coefficients of a transform's sub-band to embed the watermark bits, which cause more distortion.

3.1.2 Histogram modification based methods

Reversible watermarking schemes have also been addressed by histogram modification strategies, see Table 3.2.

Te main idea of this approach is to compute a histogram considering the cover image pixels or differences between them, to modify the generated bins. Usually, histogram modification based methods use a peak bin to embed information while other bins are shifted to create the embedding space. Depending on the approach, it is decided which bins will be used to carry information and which ones will be shifted to ensure reversibility.

Author	Year	Domain	Authentication	Technique
Vleeschouwer et al.	2001	Spatial	No	Histogram
Vleeschouwer et al.	2003	Spatial	No	Histogram
Xuan et al.	2006	Frequency	No	IWT, Histogram
Ni et al.	2006	Spatial	No	Histogram
Lin et al.	2008	Spatial	No	Histogram
Ni et al.	2008	Spatial	Yes	Histogram, blocks
Gao et al.	2009	Spatial	Yes	Histogram, blocks
Kim et al.	2009	Spatial	No	Histogram
Tsai et al.	2009	Spatial	No	Histogram
Li et al.	2010	Spatial	No	Histogram
Gao et al.	2011	Spatial	No	Histogram
An et al.	2012	Spatial	Yes	Histogram
Huang et al.	2013	Spatial	No	Histogram
Khan et al.	2014	Spatial	Yes	Histogram
Wu et al.	2015	Spatial	No	Histogram
Gao et al.	2017	Spatial	Yes	ROI areas, Histogram

Table 3.2: Methods based on histogram modification.

In order to reach reversibility, Vleeschouwer et al. (2001) proposed for the first time to apply histogram modifications to embed information in images. The cover image is partitioned into blocks. Pixels blocks are pseudo-randomly distributed on two equal-sized groups, A and C. Each histogram group is calculated and mapped to a circle, where positions are indexed based on pixels luminance levels. Watermark bits are embedded by applying slight rotations to the circles' axes, which represents the distribution mass of pixel values in each group. Rotations in group A are carried out clockwise to insert a '1' bit and anticlockwise to insert a bit '0', while group C rotations are performed in opposite way. Since pixels are pseudo-randomly distributed into two groups, axes are close to each other, so reversibility is reached by applying rotations that result in the smallest difference between calculated axes. Circular interpretation of histogram provides a solution to underflow/overflow problems. However, this technique causes salt and pepper effect on the histogram bin values 0 and 255. To solve this problem, Vleeschouwer et al. (2003) proposed to use bijective transformations together with modulo-256 addition.

After this work, new schemes emerged in which the histogram of a cover image is computed to modify their generated bins. Xuan et al. (2006) used the IWT coefficients of high frequency subbands to compute the histogram, a bin is selected to carry information while the other bins are shifted. Ni et al. (2006) propose to increase in one unit those bins between the zero and peak grayscale values to embed the information bits into the peak bin, however embedding capacity is small. Lin et al. (2008) increased the embedding capacity by computing the histogram of differences between adjacent pixels to embed the information bits into the zero bin, which is the histogram peak. Ni et al. (2008) used error correction codes together with permutations to avoid using modulo-256 addition, overcoming the salt and pepper effect. Gao et al. (2009) improved the reversibility of the scheme proposed by Ni et al., they use a location map to discard unsuitable areas of the image in the embedding process. Kim et al. (2009) exploited the high spatial correlation between neighboring pixels by generating a sub-sampled image and modifying the difference histogram, an embedding range is selected to shift its outer regions. Tsai et al. (2009) proposed to use a prediction technique to create a residual histogram, embedding capacity is increased by carrying information in the peak and zero pairs. Li et al. (2010) computed the histogram of adjacent pixel differences to embed information in the peak point. Gao et al. (2011); An et al. (2012) used statistical characteristics of images to improve the performance of histogram based methods using different kind of images. Huang et al. (2013) exploited the high correlation existing in smooth areas of medical images by modifying a difference histogram. Recently, Kamran et al. (2014) presented a high capacity reversible watermarking approach using down-sampled images. Wu et al. (2015) developed a reversible watermarking method based on contrast enhancement, which excludes some histogram bins from shifting process to enhance the contrast in regions of interest. Finally, Gao et al. (2017) proposed a reversible watermarking scheme for tamper localization, which computes a histogram considering regions of interest. Tamper localization is done by embedding a feature-bit matrix. In order to achieve contrast enhancement, a peak-pair bins are selected to be expanded.

In contrast with compression based methods, where it is necessary to embed a copy of the

original bits that were modified, methods based on histogram modification reduce the size of control information generated in embedding process since it is only necessary to know which bins contain information and which ones do not, however embedding capacity is smaller as usually watermark bits are embedded only within the peak bin.

3.1.3 Error expansion based methods

Tian (2003) proposed a new reversible technique, called Difference Expansion (DE). The cover image is divided into pairs of adjacent pixels (p_l, p_r). Difference between p_l and p_r is computed to obtain $d_e = p_l - p_r$. Every watermark bit b is appended as a new LSB of d_e. Finally, pairs of pixels (p_l, p_r) are modified considering the average of original values and the generated expanded difference. Image redundancy is exploited by computing differences between neighbor pixels to discover embedding space. Usually, neighbor pixels are similar, so generated differences are small and by coding the watermark bits into this differences, introduced distortion will be lower.

Thodi and Rodríguez (2004) proposed a well-known technique in the reversible watermarking subject, called Prediction Error Expansion (PEE). Unlike Tian, who uses differences between pairs of adjacent pixels to embed the watermark, Thodi and Rodríguez use a prediction error computed from a three pixels context. Using a predictor takes better advantage of the redundancy in the cover image. Many reversible schemes have been proposed based on this technique, see Table 3.3.

Author	Year	Domain	Authentication	Technique
Tian	2003	Spatial	No	Difference Expansion
Thodi and Rodríguez	2004	Spatial	No	PEE
Thodi and Rodríguez	2007	Spatial	No	PEE
Chen et al.	2009	Spatial	No	PEE
Sachnev et al.	2009	Spatial	No	PEE
Ou et al.	2010	Spatial	No	PEE, Histogram
Tudoroiu et al.	2011	Spatial	Yes	PEE
Coltuc	2011	Spatial	No	PEE
Luo et al.	2011	Spatial	No	PEE
Coltuc	2012	Spatial	No	PEE
Li et al.	2013	Spatial	No	PVO

Peng et al.	2014	Spatial	No	PVO
Dragoi and Coltuc	2014	Spatial	No	PEE
Naheed et al.	2014	Spatial	Yes	PEE, Interpolation
Wang et al.	2015	Spatial	No	PVO
Qu and Kim	2015	Spatial	No	PVO
Siddiqa and Khan	2015	Spatial	No	PEE
Ou et al.	2016	Spatial	No	PVO, PEE
He et al.	2017	Spatial	No	PVO
Ou et al.	2017	Spatial	No	PEE
He et al.	2017	Spatial	No	PEE

Table 3.3: Methods based on error expansion.

Chen et al. (2009) operated with a predictor of full context, i.e, to predict one pixel they used all its 8 adjacent pixels as context, which produced smaller prediction errors. Later Sachnev et al. (2009) proposed a predictor using a rhombus pattern. Ou et al. (2010) presented a linear predictor to obtain the initial prediction, which is modified by computing the variance of adjacent pixels to decrease prediction errors. Tudoroiu et al. (2011) developed the Median Edge Detector (MED) predictor using three pixels as context. Coltuc (2011, 2012) improved imperceptibility levels by splitting the prediction error into the current pixel and its context. Since modified pixels after prediction error embedding may suffer loss of correlation, Luo et al. (2011) proposed using a compensation to deal with this problem. Dragoi and Coltuc (2014) used an adaptive prediction of pixels that do not belong to smooth regions of the images. Naheed et al. (2014) created an image using interpolation to compute the prediction errors. They exploited the image pixel values correlation by incorporating two intelligent techniques: a GA and Particle Swarm Optimization (PSO). Recently Siddiqa and Khan (2015) proposed using a rhombus predictor to exploit the variance of neighbouring pixels.

Prediction error techniques are computationally less expensive since they work in the spatial domain, avoiding the computation of some transform or the use of compression algorithms. This approach is emerging since proposed techniques are promising in achieving a trade-off between embedding capacity and imperceptibility properties.
The challenge for prediction error based schemes is to predict the pixel values in the most accurate way possible. The more accurate the prediction value, the smaller the error generated. Therefore, the distortion introduced into a cover image will be smaller when encoding a watermark bit into the error.

Pixel Value Ordering (PVO) based methods

PVO is a reversible strategy proposed by Li et al. (2013). The method combines two known techniques: histogram modification and prediction error expansion to reversible embed information.
First, a cover image is partitioned into non-overlapped blocks of size $n \times m$. Each block B contains x pixels, that is to say $x = n \times m$. For a given block, their pixel values $(B_{(1)} \dots B_{(x)})$ are sorted in ascending order to obtain $(B_{sorted(1)} \dots B_{sorted(x)})$, such that: $B_{sorted(1)} \leq \dots \leq B_{sorted(x)}$, $sorted(i) < sorted(j)$ if $B_{sorted(i)} = B_{sorted(j)}$ and $i < j$. The second largest value $B_{sorted(x-1)}$ is used to predict the maximum $B_{sorted(x)}$. Prediction error is computed using Eq. 3.1.

$$PE_{max} = B_{sorted(x)} - B_{sorted(x-1)} \tag{3.1}$$

A histogram is created using PE_{max} values. By subtracting the second largest value ($B_{sorted(x-1)}$) to the maximum one ($B_{sorted(x)}$), the histogram always will contain zero and positive bins, that is to say, bins with values from 0 to 255 (for an 8-bit deep image). Since bin with $PE_{max} = 1$ is typically the histogram peak (Li et al., 2013), it is used to carry the watermark bits, while the other bins (bins larger than 1) are shifted to create the embedding space ensuring reversibility.

A watermark bit $b \in \{0, 1\}$ is embedded into prediction error PE_{max} using Eq. 3.2.

$$\tilde{PE}_{max} = \begin{cases} PE_{max} & \text{if } PE_{max} = 0, \\ PE_{max} + b & \text{if } PE_{max} = 1, \\ PE_{max} + 1 & \text{if } PE_{max} > 1 \end{cases} \tag{3.2}$$

Subsequently, maximum pixel value $B_{sorted(x)}$ is modified using Eq. 3.3.

$$\tilde{B}_{sorted(x)} = B_{sorted(x-1)} + \tilde{PE}_{max} = \begin{cases} B_{sorted(x)} & \text{if } PE_{max} = 0, \\ B_{sorted(x)} + b & \text{if } PE_{max} = 1, \\ B_{sorted(x)} + 1 & \text{if } PE_{max} > 1 \end{cases} \tag{3.3}$$

In this way pixel values $B_{sorted(1)} \dots B_{sorted(x-1)}$ are not modified, while $B_{sorted(x)}$ remains unchanged or increases in one unit. Therefore, the pixel value order of each block is maintained after data embedding to ensure reversibility. A similar embedding process is performed to embed an additional bit into the smaller block pixel value. So this technique allows embedding a maximum of two watermark bits per block.

Peng et al. (2014) increase the embedding capacity of the scheme proposed by Li et al. (2013). Maximum and minimum prediction errors are computed considering pixel locations. Therefore, maximum prediction error is calculated using Eq. 3.4.

$$D_{max} = B_u - B_v \tag{3.4}$$

where:

u = min$(sorted(x), sorted(x-1))$

v = max$(sorted(x), sorted(x-1))$

A histogram is created using D_{max} values. Unlike the method proposed by Li et al., where $PE_{max} = B_{sorted(x)} - B_{sorted(x-1)}$, the new predictor causes the histogram to take from -255 to 255 bin values (for an 8-bit deep image). So bin 0 will be the histogram peak

since bin 1 now is distributed into bins 1 and -1.

Bins 0 and 1 are used to carry the watermark bits and bins greater than 1 or smaller than 0 are shifted to create embedding space. A watermark bit $b \in \{0, 1\}$ is embedded into the prediction error D_{max} using Eq. 3.5.

$$\tilde{D}_{max} = \begin{cases} D_{max} + b & \text{if } D_{max} = 1, \\ D_{max} + 1 & \text{if } D_{max} > 1, \\ D_{max} - b & \text{if } D_{max} = 0, \\ D_{max} - 1 & \text{if } D_{max} < 0 \end{cases} \tag{3.5}$$

Finally, the maximum pixel value $B_{sorted(x)}$ is modified using Eq. 3.6.

$$\tilde{B}_{sorted(x)} = B_{sorted(x-1)} + \mid \tilde{D}_{max} \mid = \begin{cases} B_{sorted(x)} + b & \text{if } D_{max} = 1, \\ B_{sorted(x)} + 1 & \text{if } D_{max} > 1, \\ B_{sorted(x)} + b & \text{if } D_{max} = 0, \\ B_{sorted(x)} + 1 & \text{if } D_{max} < 0 \end{cases} \tag{3.6}$$

A similar process is performed to embed an extra watermark bit into the prediction error D_{min}. Algorithm 1 is applied to embed up to two watermark bits into a given block B.

Algorithm 1 Mark a block B using PVO (Peng et al., 2014).

Input : Block B of size $n \times m$,
Watermark w

1: **procedure** MARKBLOCK(B, w)
2: $x \leftarrow n \times m$
3: $B_a \leftarrow$ reshape$(B, [1 \dots x])$ ▷ Reshape B to an array of size [1 … x]
4: $B_{sorted}, sorted \leftarrow$ sort(B_a) ▷ Sorted array: B_{sorted}, indexes: $sorted$

// Watermark embedding into prediction error D_{max}
5: $u = \min(sorted(x), sorted(x-1))$
6: $v = \max(sorted(x), sorted(x-1))$
7: $D_{max} \leftarrow B_a(u) - B_a(v)$
8: **if** $D_{max} = 0$ **or** $D_{max} = 1$ **then**

Algorithm 1 (continue) Mark a block B using PVO (Peng et al., 2014).

```
        b ← w.getbit()                                  ▷ Get a new watermark bit
        B_sorted(x) ← B_sorted(x) + b                   ▷ Embed b into maximum value
    else
        B_sorted(x) ← B_sorted(x) + 1                   ▷ Histogram shifting
    end if
// Watermark embedding into prediction error D_min
    s = min(sorted(1), sorted(2))
    t = max(sorted(1), sorted(2))
    D_min ← B_a(s) − B_a(t)
    if D_min = 0 or D_min = 1 then
        b ← w.getbit()                                  ▷ Get a new watermark bit
        B_sorted(1) ← B_sorted(1) − b                   ▷ Embed b into minimum value
    else
        B_sorted(1) ← B_sorted(1) − 1                   ▷ Histogram shifting
    end if
    B_u ← inversesort(B_sorted, sorted)                 ▷ Move pixels to their original position
    B ← reshape(B_u, [1...n][1...m])                    ▷ reshape B_u to a block of size n × m
end procedure
```

Using this strategy also ensures that pixel values $B_{sorted(1)} \dots B_{sorted(x-1)}$ are not modified, while $B_{sorted(x)}$ remains unchanged or increases in one unit. Thus, the pixels order is maintained.

The watermark extraction is performed as follows. Suppose that the sorted pixels of a watermarked block B_w are $(Bw_{sorted(1)} \dots Bw_{sorted(x)})$, where $Bw_{sorted(x)} = \tilde{B}_{sorted(x)}$. The modified prediction error is computed as $\tilde{D}_{max} = Bw_u - Bw_v$, where (u, v) is defined in Eq. 3.4.

- If $\tilde{D}_{max} > 0$, it is known that $Bw_u > Bw_v$. Thus, $sorted(x) < sorted(x-1)$, $u = sorted(x)$ and $v = sorted(x-1)$.
  - If $\tilde{D}_{max} \in \{1, 2\}$, the hidden bit is $b = \tilde{D}_{max} - 1$ and the original maximum pixel value is $B_{sorted(x)} = Bw_u - b$.
  - If $\tilde{D}_{max} > 2$, there is no a hidden bit here and the original maximum pixel value is $B_{sorted(x)} = Bw_u - 1$.

- if $\tilde{D}_{max} \leq 0$, it is known that $Bw_u \leq Bw_v$. Thus, $sorted(x) > sorted(n-1)$, $u = sorted(x-1)$ and $v = sorted(x)$.

  - If $\tilde{D}_{max} \in \{0, -1\}$, the hidden bit is $b = -\tilde{D}_{max}$ and the original maximum pixel value is $B_{sorted(x)} = Bw_v - b$.

  - If $\tilde{D}_{max} < -1$, there is no a hidden bit here and the original maximum pixel value is $B_{sorted(x)} = Bw_v - 1$.

Algorithm 2 is applied to remove the watermark bits embedded into B_w.

Algorithm 2 Watermark bits removal from B_w (Peng et al., 2014).

Input : Watermarked block B_w of size $n \times m$,
Watermark w_e, in which extracted bits will be appended

procedure REMOVEWATERMARKBITS(B_w, w_e)
  $x \leftarrow n \times m$
  $Bw_a \leftarrow$ reshape$(B_w, [1...x])$ ▷ Reshape B_w to an array of size [1...x]
  $Bw_{sorted}, sorted \leftarrow$ sort(Bw_a) ▷ Sorted array: Bw_{sorted}, indexes: $sorted$
  // Watermark extraction from prediction error $\tilde{D}_{max}$
  $u = \min(sorted(x), sorted(x-1))$
  $v = \max(sorted(x), sorted(x-1))$
  $\tilde{D}_{max} \leftarrow Bw_a(u) - Bw_a(v)$
  if $\tilde{D}_{max} > 0$ **then**
    if $\tilde{D}_{max} = 1$ **or** $\tilde{D}_{max} = 2$ **then**
      $b \leftarrow \tilde{D}_{max} - 1$ ▷ Get hidden bit
      append b to w_e
      $Bw_{sorted}(x) \leftarrow Bw_a(u) - b$ ▷ Original pixel is recovered
    else
      $Bw_{sorted}(x) \leftarrow Bw_a(u) - 1$ ▷ Histogram shifting
    end if
  else
    if $\tilde{D}_{max} = 0$ **or** $\tilde{D}_{max} = -1$ **then**
      $b \leftarrow -\tilde{D}_{max}$ ▷ Get hidden bit
      append b to w_e
      $Bw_{sorted}(x) \leftarrow Bw_a(v) - b$ ▷ Original pixel is recovered
    else
      $Bw_{sorted}(x) \leftarrow Bw_a(v) - 1$ ▷ Histogram shifting
    end if
  end if

Algorithm 2 (continue) Watermark bits removal from B_w (Peng et al., 2014).

// Watermark extraction from prediction error $\tilde{D}_{min}$
$s = min(sorted(1), sorted(2))$
$t = max(sorted(1), sorted(2))$
$\tilde{D}_{min} \leftarrow Bw_a(s) - Bw_a(t)$
if $\tilde{D}_{min} > 0$ **then**
  if $\tilde{D}_{min} = 1$ **or** $\tilde{D}_{min} = 2$ **then**
    $b \leftarrow \tilde{D}_{min} - 1$ ▷ Get hidden bit
    append b to w_e
    $Bw_{sorted}(1) \leftarrow Bw_a(t) - b$ ▷ Original pixel is recovered
  else
    $Bw_{sorted}(1) \leftarrow Bw_a(t) - 1$ ▷ Histogram shifting
  end if
else
  if $\tilde{D}_{min} = 0$ **or** $\tilde{D}_{min} = -1$ **then**
    $b \leftarrow -\tilde{D}_{min}$ ▷ Get hidden bit
    append b to w_e
    $Bw_{sorted}(1) \leftarrow Bw_a(s) - b$ ▷ Original pixel is recovered
  else
    $Bw_{sorted}(1) \leftarrow Bw_a(s) - 1$ ▷ Histogram shifting
  end if
end if
$B_u \leftarrow$ inversesort$(Bw_{sorted}, sorted)$ ▷ Move pixels to their original position
$B \leftarrow$ reshape$(B_u, [1...n][1...m])$ ▷ reshape B_u to a block of size $n \times m$
end procedure

After extraction process, the watermark is obtained and removed from the watermarked blocks. As a result, pixels recovered their original values (i.e., the original image is restored), so reversibility is ensured.

Li et al. show that larger sized blocks provide better imperceptibility levels, however embedding capacity is decreased. On the other hand, smaller sized blocks causes more distortion and provide a greater embedding capacity. Deciding block size and threshold values is a situation that should be considered, since modifying these parameters directly impacts on methods performance.

In order to find a trade-off between imperceptibility and embedding capacity, Wang et al. (2015) proposed a PVO strategy based on dynamic pixel block partition. The cover image is partitioned into 4×4 sized blocks. A couple of thresholds T_1 and T_2 is used to classify blocks in rough, normal and flat blocks according to their noise level. Rough block prob-

ably represents a textured area, so it is unsuitable to embed information, then this type of blocks are discarded in the embedding process. Normal block is a moderately smooth area of the image, thus up to two information bits can be embedded in it using the predictor proposed by Peng et al. (2014). Because of a flat block represents a smooth region of the image, it is divided into four 2×2 sized sub-blocks, so up to eight information bits can be embedded because each 2×2 sub-block can carry two bits.
Wang et al. merge features from two different block sizes to enhance the performance of pixel value ordering based embedding technique, however the block size is restricted to using only 4×4 and 2×2.

3.2 Discussion

Considering state of the art research works, many reversible watermarking schemes have been addressed using compression, histogram modification and error expansion techniques, which examples are shown in three previous subsections. However, papers could be classified according to the insertion domain. On the one hand, there exists works in frequency domain that use IWT, DWT or DCT coefficients transforms to embed the watermark. On the other hand, works based on spatial domain use techniques that directly affects one or more LSBs of image pixels. As shown above, many schemes do not provide image authentication, so it is important to consider including it in the design of a new scheme.

Arsalan et al. (2012, 2017) proposed using evolutionary computing in the reversible watermarking field. Evolutionary algorithms have been useful to find an optimal set of embedding method parameters, without performing an exhaustive search.

Li et al. (2013) proposed the PVO technique. This strategy have been recently studied in order to decrease distortion and increase the embedding capacity when a watermark is inserted into the cover image. Parameters used by the PVO methods are

the block size and one or more thresholds, which are used to decide how the block will be treated in the insertion process. However, Li et al. (2013); Peng et al. (2014); Wang et al. (2015) determined PVO parameters by performing iterative searches within a restricted solution space of smaller block sizes.

Finally, in recent identified works, there is a tendency to manipulate the cover image in blocks of pixels and use prediction error expansion strategies, since this technique provides a versatile insertion capacity and better distortion levels than other techniques (Khan et al., 2014b). In this thesis a reversible watermarking scheme based on pixel value ordering technique for image authentication is proposed. Some improvement areas in PVO based methods were identified, such as distortion reduction and embedding capacity increment. Solutions space can be expanded by increasing the block sizes, so an intelligent technique is presented to optimize this process. In next chapter all details related to proposed scheme are exposed.

4

Proposed Scheme

This chapter detail the proposed reversible watermarking scheme called PVO-GA. Chapter 3 explained reliance of PVO techniques performance on parameters as block size and thresholds for blocks classification. In order to improve PVO's performance, the proposed scheme introduces modifications at image pre-processing stage and an unequal sized block partition strategy. The solutions space is expanded due to using larger block sizes, so a genetic algorithm is deployed to find optimal/near optimal PVO-GA parameters. At the transmitter, a watermarked image is generated to provide image authentication using a message authentication code, which is extracted at the receiver to perform the authentication phase.

4.1 Overview

The proposed scheme is based on two main processes. On the one hand, the insertion process, which is carried out at the transmitter end of the system. On the other hand, the extraction and authentication process, which is performed at the receiver. An overall flowchart of the proposed scheme is shown in Fig. 4.1. During the insertion process, a watermark w and a cover image I_c are prepared to be used by the embedding algorithm. Later, the watermark is embedded using the predictor proposed by Peng et al. (2014) to produce a watermarked image I_w. Two modifications are introduced to improve the performance of PVO technique: an image pre-processing stage and an unequal sized block

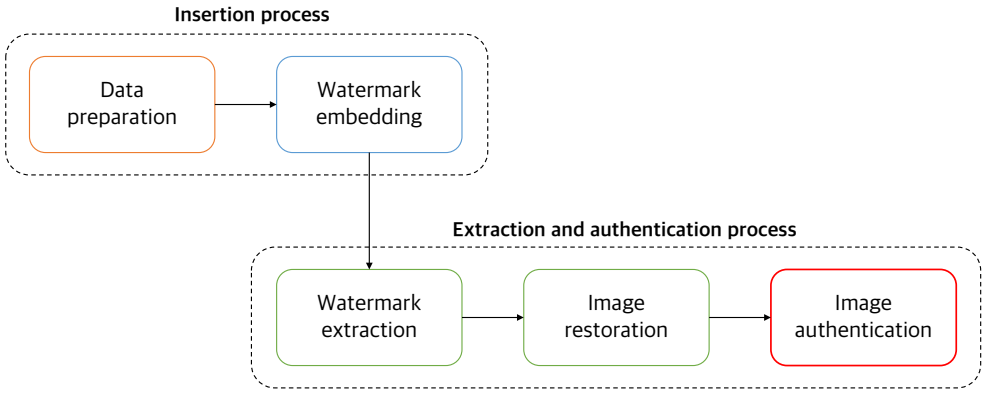


Figure 4.1: Proposed scheme general flowchart.

partition strategy, which is based on the dynamic pixel block partition proposed by Wang et al. (2015). Proposed modifications allow to use larger sized blocks, which result in a better trade-off between imperceptibility and embedding capacity. During the extraction process, a watermark w_e is first extracted from I_w. After watermark extraction, I_c is restored. Finally, the image authentication phase is carried out. Every phase of the proposed scheme is detailed in next sections.

4.2 Insertion process

The proposed insertion process is based on some stages to optimize PVO embedding performance. In order to provide image authentication, the message to be hidden m_h and I_c are used to get a message authentication code MAC_t, using a secret key k and SHA-256 algorithm. I_c undergoes an image pre-processing to obtain the pre-processed image I_{pr}. In this phase, a recovery bit sequence R_s is generated. R_s will be used after watermark extraction process to recover the original image I_c. R_s is compressed using arithmetic coding to obtain R_{sc}. Thus, MAC_t and m_h make up w while R_{sc} is part of control information used to recover the original image. Finally, w is inserted into I_{pr} using the predictor proposed by Peng et al. (2014).

In contrast to the method proposed by Wang et al. (2015), where the cover image is

partitioned in 4×4 sized blocks, we divided I_{pr} in blocks of size $n \times m$, which can be subdivided into four sub-blocks using unequal sized block partition strategy. A compact fine grained genetic algorithm is deployed to find optimal block size and thresholds to be used in embedding process. The insertion process is shown in Fig. 4.2. Every stage of proposed insertion process is detailed in next subsections.

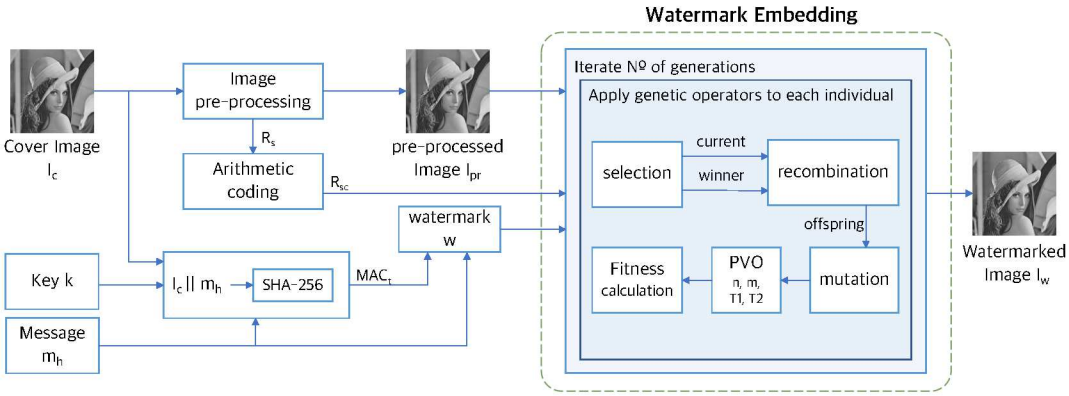


Figure 4.2: PVO-GA insertion process.

4.2.1 Message authentication code calculation

Exact authentication refers to the task of verifying that a work has not been altered at all since it left a trusted party (Cox et al., 2008). Image must be considered as altered, even if only one bit has been modified. In order to provide protection to m_h and I_c, we compute a MAC by concatenating I_c in its string representation together with m_h, as it is shown in Fig 4.3.

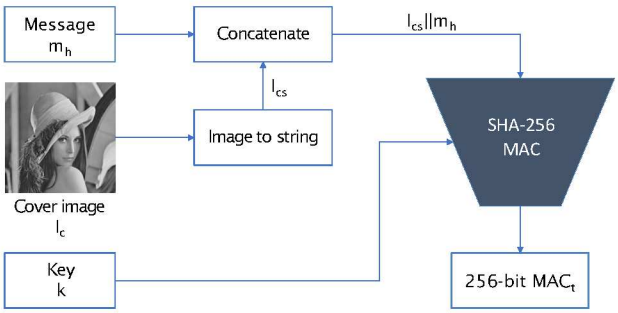


Figure 4.3: Message Authentication Code calculation.

Given a cover image I_c, a message m_h and a secret key k, to guarantee the integrity of both I_c and m_h, a message authentication code is computed. Image pixel values are converted to a string I_{cs}, then I_{cs} and m_h are concatenated to get $I_{cs}\|m_h$, which will be SHA-256MAC function's argument using k. The 256-bit MAC_t value will be part of the watermark w that will be inserted into the image.

4.2.2 Image pre-processing

PVO methods distort minimum and maximum values from a data block in at most one unit (Li et al., 2013; Peng et al., 2014; Wang et al., 2015). Minimum value is decreased while maximum value is increased to achieve reversibility. Previous works identify those blocks that contain pixel values 0 or 255 to create a location map, which is compressed and appended as part of control information. The identified blocks are omitted in the insertion process because they can cause underflow or/and overflow problems (i.e. values produced outside allowed range), however embedding capacity could be affected since those blocks are not being analyzed.

Considering this, an image pre-processing stage is proposed. For a given cover image I_c, when its histogram is calculated, it will contain from 0 to 255 gray scale values. In this stage, all pixel values 0 and 255 are increased and decreased in one unit respectively, so that after modification, the histogram range becomes from 1 to 254 grayscale values. A pre-processed image I_{pr} and a recovery sequence R_s are obtained by applying the Algorithm 3 to the cover image I_c.

Algorithm 3 Image pre-processing function.

Input : Cover image I_c of size $N \times M$
Output : Pre-processed image I_{pr} of size $N \times M$,
Recovery sequence R_s

1: **function** IMAGEPREPROCESSING(I_c)
2: $cont \leftarrow 0$; $R_s \leftarrow$ empty string
3: $I_{pr} \leftarrow I_c$
4: **for** $i \leftarrow 1$ **to** N **do**
5: **for** $j \leftarrow 1$ **to** M **do**
6: **if** $I_{pr}(i,j) = 0$ **or** $I_{pr}(i,j) = 255$ **then**

Algorithm 3 (continue) Image pre-processing function.

```
            if I_pr(i,j) = 0 then
                I_pr(i,j) ← 1                          ▷ Increase pixel value
            else
                I_pr(i,j) ← 254                        ▷ Decrease pixel value
            end if
            cont ← cont + 1
            R_s(cont) ← '1'                            ▷ Append 1 to indicate modification
        else if I_pr(i,j) = 1 or I_pr(i,j) = 254 then
            cont ← cont + 1
            R_s(cont) ← '0'                            ▷ Append 0 to indicate that pixel was not changed
        end if
    end for
end for
return I_pr, R_s
end function
```

The proposed image pre-processing is explained with the help of an example as shown in Fig. 4.4. This figure shows original and modified image pixel values obtained after applying the proposed modification.

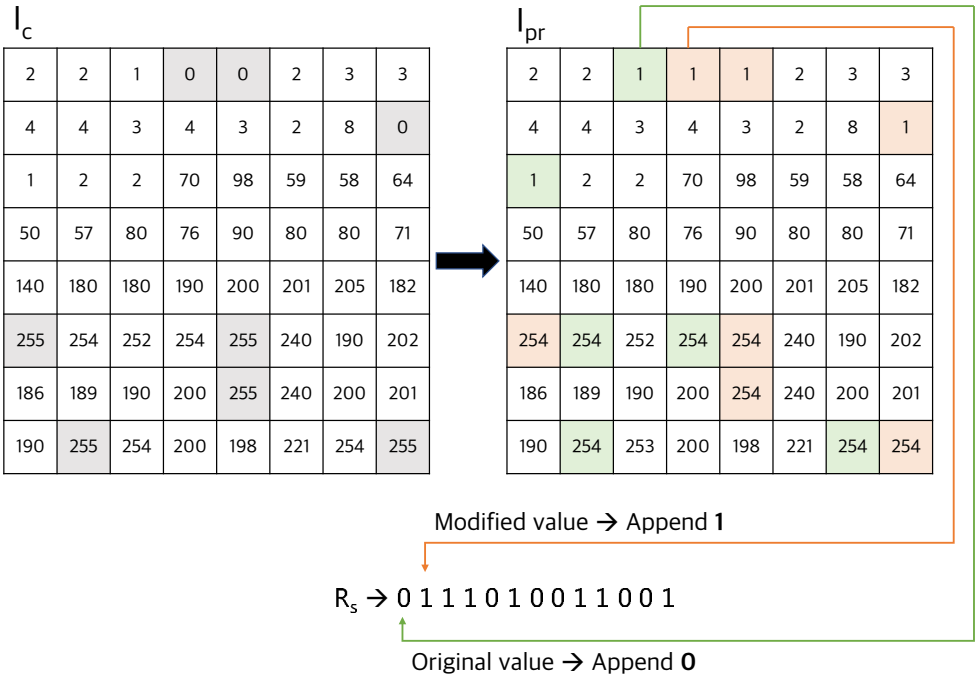


Figure 4.4: Image pre-processing and recovery sequence calculation example.

Every identified pixel with value 0 or 255 existing in I_c is highlighted in gray color. After image pre-processing, these pixels become 1 and 254 respectively, which are shown in orange color, while original values 1 and 254 in I_{pr} are shown in green color. All pixels 1 from I_{pr} are due to 0's and 1's pixels in I_c, in the same way all pixels 254 from I_{pr} are due to 254's and 255's pixels in I_c. To preserve reversibility, a recovery binary sequence R_s is generated in the process. I_{pr} is scanned in raster order (from left to right and top to bottom) to identify all 1's and 254's pixel values, when some of these two values is found and its correspondent pixel in I_c differs, a 1 bit is appended to R_s, otherwise a 0 bit is appended. Thus, the recovery sequence required for obtaining I_c is $R_s = 0111010011001$. When I_c does not contain pixels with values 0 or 255, I_{pr} will be identical to I_c, then R_s is not required since I_c is not modified. The original cover image I_c is obtained by following the Algorithm 4.

Algorithm 4 Image post-processing function.

Input : Pre-processed image I_{pr} of size $N \times M$,
Recovery sequence R_s
Output : Cover image I_c of size $N \times M$

```
1:  function ImagePostProcessing(I_pr, R_s)
2:      cont ← 0
3:      for i ← 1 to N do
4:          for i ← 1 to M do
5:              if I_pr(i, j) = 1 or I_pr(i, j) = 254 then
6:                  cont ← cont + 1
7:                  if R_s(cont) = '1' then
8:                      if I_pr(i, j) = 1 then          ▷ 1 indicates that pixel was modified
9:                          I_pr(i, j) ← 0              ▷ Pixel 1 is restored to 0
10:                     else
11:                         I_pr(i, j) ← 255            ▷ Pixel 254 is restored to 255
12:                     end if
13:                 end if
14:             end if
15:         end for
16:     end for
17:     I_c ← I_pr
18:     return I_c
19: end function
```

First, I_{pr} is scanned in raster order, when a pixel with value 1 or 254 is found, a bit from R_s is obtained. If the obtained bit is 1, this means that pixel was modified by image pre-processing function, then pixel value is modified to 0 or 255 respectively, otherwise pixel is not modified.

Image pre-processing stage is performed to obtain a grayscale image with pixel values from 1 to 254. Since PVO techniques modify minimum and maximum values of each block in at most one unit, all blocks can be analyzed in embedding process, which improves embedding capacity.

4.2.3 Unequal sized block partition strategy

As it is mentioned in Chapter 3, the performance of PVO reversible watermarking schemes relies on algorithm parameters as block size $n \times m$ and one or more thresholds. Wang et al. (2015) proposed combining two different block sizes. First, cover image is divided into 4×4 sized blocks to calculate their noise level NL, which indicate if a block represents a normal, flat or textured area in the image using two thresholds T_1 and T_2, where $T_1 > T_2$. If $NL > T_1$, block is considered as rough texture block, then it is omitted for embedding because this block type causes more distortion. If $T_1 \geq NL > T_2$, block is considered as normal block, so up to two data bits can be embedded into 4×4 sized block using the strategy of Peng et al. (2014). Finally, if $NL \leq T_2$, block is considered as flat (smooth texture) block. This block type, in theory, contains more pixels redundancy, so up to eighth data bits can be embedded by partitioning the entire block into four 2×2 sized sub-blocks.

According to Wang et al. (2015), it can be observed that classifying blocks to use two different block sizes is a good deal that is essential to find a trade-off between imperceptibility and payload capacity obtained when a watermark is embedded. However, it is known that the larger the block size, the smaller the distortion introduced (Li et al., 2013; Peng et al., 2014), therefore the proposed insertion phase uses block sizes larger than 4×4.

To guarantee reversibility, using floor and ceiling functions to obtain the four sub-blocks is essential, since block sizes may be of unequal dimensions. Thus, for a given block B of size $n \times m$, the dimensions of its four sub-blocks are calculated with Equations 4.1 - 4.4.

$$sb1_n = \lfloor \frac{n}{2} \rfloor, sb1_m = \lfloor \frac{m}{2} \rfloor \tag{4.1}$$

$$sb2_n = \lfloor \frac{n}{2} \rfloor, sb2_m = \lceil \frac{m}{2} \rceil \tag{4.2}$$

$$sb3_n = \lceil \frac{n}{2} \rceil, sb3_m = \lfloor \frac{m}{2} \rfloor \tag{4.3}$$

$$sb4_n = \lceil \frac{n}{2} \rceil, sb4_m = \lceil \frac{m}{2} \rceil \tag{4.4}$$

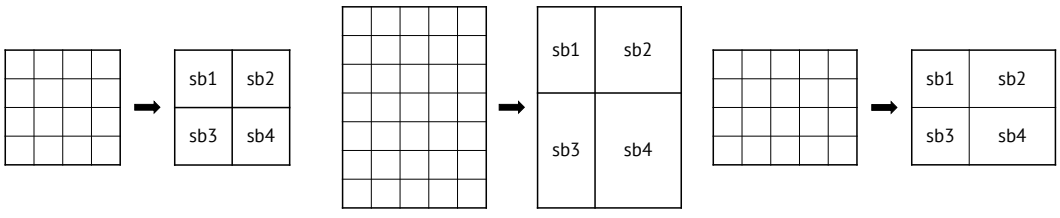
Eq. 4.1 is applied to obtain the dimensions of the first sub-block $sb1$, height and width of B are divided by 2 using floor function, to know how many pixels of each dimension (n, m) will be taken. The complementary parts are computed using ceiling function. Eq. 4.2 is applied to compute the dimensions of $sb2$, here ceiling function is used to take the remaining part of m since the first part was taken to create $sb1$. When $sb3$ dimensions are computed using Eq. 4.3, ceiling function is used to take the remaining part of n. Finally, $sb4$ dimensions are calculated using Eq. 4.4, where only ceiling function is used to take the last part of B. Algorithm 5 is used to obtain four sub-blocks from B. When each sub-block dimensions are computed, every sub-block is created by taking from n_{start} to n_{end} pixel rows and from m_{start} to m_{end} pixel columns from B.

Algorithm 5 Block partition algorithm.

Input : Block B of size $n \times m$
Output : Sub-blocks $sb1$, $sb2$, $sb3$ and $sb4$ of size $sb\#_n \times sb\#_m$

function GETSUBBLOCKS(B)
// Equations 4.1-4.4 are applied to obtain each sub-block dimensions
  $sb1_n \leftarrow \lfloor \frac{n}{2} \rfloor$; $sb1_m \leftarrow \lfloor \frac{m}{2} \rfloor$
  $sb2_n \leftarrow \lfloor \frac{n}{2} \rfloor$; $sb2_m \leftarrow \lceil \frac{m}{2} \rceil$
  $sb3_n \leftarrow \lceil \frac{n}{2} \rceil$; $sb3_m \leftarrow \lfloor \frac{m}{2} \rfloor$
  $sb4_n \leftarrow \lceil \frac{n}{2} \rceil$; $sb4_m \leftarrow \lceil \frac{m}{2} \rceil$
// Sub-blocks creation $B([n_{start} : n_{end}], [m_{start} : m_{end}])$
  $sb1 \leftarrow B([1 : sb1_n], [1 : sb1_m])$
  $sb2 \leftarrow B([1 : sb2_n], [sb1_m + 1 : sb1_m + sb2_m])$
  $sb3 \leftarrow B([sb1_n + 1 : sb1_n + sb3_n], [1 : sb3_m])$
  $sb4 \leftarrow B([sb2_n + 1 : sb1_n + sb4_n], [sb3_m + 1 : sb3_m + sb4_m])$
  return $sb1$, $sb2$, $sb3$, $sb4$
end function

Fig. 4.5 shows three different partition examples. On the one hand, when $n = m$, and both are even numbers, the produced sub-blocks will be of the same dimensions. On the other hand, when dimensions are of unequal size (i.e. $n \neq m$), the produced sub-blocks will be different. Fig. 4.5a shows a 4×4 sized block. After applying partition using Algorithm 5, there will be four 2×2 sized sub-blocks. Fig. 4.5b shows a 7×5 sized block, which will be result in four unequal sized sub-blocks. Finally, in Fig. 4.5c it is shown a 4×5 sized block that after partition will result in two 2×2 sized sub-blocks and two 2×3 sized sub-blocks.



(a) $n \times m = 4 \times 4$ (b) $n \times m = 7 \times 5$ (c) $n \times m = 4 \times 5$

Figure 4.5: Partition strategy examples.

By adopting the proposed partition strategy, the method clearly will combine different sized sub-blocks, and therefore it will take advantage of features offered by each sub-block to find a trade-off between imperceptibility and embedding capacity. Additionally,

Wang et al. (2015) scheme is a particular case of this partition strategy, that is, when the block size of B is 4×4.

Noise level is computed strategically to preserve reversibility since blocks classification must be identical in watermark embedding and extraction. Algorithm 6 is used to compute the noise level of a given block B.

Algorithm 6 Noise level computation.

Input : Block B of size $n \times m$
Output : Noise level NL

function COMPUTENOISELEVEL(B)
  $sb1, sb2, sb3, sb4 \leftarrow$ GETSUBBLOCKS(B)

  $sb1_s \leftarrow$ second minimum value in $sb1$
  $sb2_s \leftarrow$ second minimum value in $sb2$
  $sb3_s \leftarrow$ second minimum value in $sb3$
  $sb4_s \leftarrow$ second minimum value in $sb4$

  $sb1_g \leftarrow$ second maximum value in $sb1$
  $sb2_g \leftarrow$ second maximum value in $sb2$
  $sb3_g \leftarrow$ second maximum value in $sb3$
  $sb4_g \leftarrow$ second maximum value in $sb4$

  $NL \leftarrow max(sb1_g, sb2_g, sb3_g, sb4_g) - min(sb1_s, sb2_s, sb3_s, sb4_s)$
  return NL
end function

After watermark embedding maximum/minimum pixel values of every watermarked block could be increased/decreased in one unit respectively using the predictor of Peng et al. (2014). Second maximum/minimum values are not modified, so they are used to compute NL since are identical in watermark embedding and extraction.

In PVO-based methods, a watermark is embedded block by block. The proposed embedding process is performed by using Algorithm 7. A pre-processed image I_{pr} is partitioned into blocks of size $n \times m$. Every block B is classified into rough, normal or flat block using two thresholds T_1 and T_2 as it is proposed by Wang et al. (2015), however the proposed partition strategy is able to deal with block sizes greater than 4×4, since Equations 4.1-4.4 guarantee reversibility. When B is classified as flat block, four sub-blocks are

obtained from it using Algorithm 5, so each sub-block $sb1, sb2, sb3$ and $sb4$ is used to embed watermark bits. When B is classified as normal block, the complete B is used to embed watermark bits. Finally, if B is classified as rough block, it is not used to embed watermark bits, since represents a textured area of the image.

Algorithm 7 Watermark Embedding.

Input : Pre-processed image I_{pr} of size $N \times M$,
Compressed recovery sequence R_{sc},
Watermark w,
Block size $n \times m$,
Thresholds T_1, T_2
Output : Watermarked image I_w

function WATERMARKEMBBEDING($I_{pr}, R_{sc}, w, n, m, T_1, T_2$)
  $Blocks \leftarrow$ partition(I_{pr}, n, m) ▷ Divide I_{pr} into blocks of size $n \times m$
  for all $B \in Blocks$ **do**
    $NL \leftarrow$ COMPUTENOISELEVEL(B)
    if $T_2 \geq NL$ **then** ▷ Flat block
      $sb1, sb2, sb3, sb4 \leftarrow$ GETSUBBLOCKS(B)
      MARKBLOCK($sb1, w$)
      MARKBLOCK($sb2, w$)
      MARKBLOCK($sb3, w$)
      MARKBLOCK($sb4, w$)
    else if $T_2 < NL \leq T1$ **then** ▷ Normal block
      MARKBLOCK(B, w)
    else if $NL > T_1$ **then** ▷ Rough block
      Do nothing
    end if
    if all w bits were embedded **then**
      $B_{last} \leftarrow$ number of last watermarked block
      $C_i \leftarrow n \| m | T_1 \| T_2 \| B_{last} \|$length($R_{sc}$)$\| R_{sc}$ ▷ Control information
      $C_{il} \leftarrow$ length(C_i)
      $LSB_r \leftarrow$ record the LSBs of the first C_{il} pixels in I_{pr}
      replace the LSBs of the first C_{il} pixels in I_{pr} by C_i
      continue embedding LSB_r in the remaining blocks starting on $B_{last} + 1$
    end if
  end for
  $I_w \leftarrow$ join($Blocks$) ▷ Create the watermarked image using $Blocks$
  return I_w
end function

When all watermark bits are inserted, control information is generated using next specifications:

- Block size n (4 bits), m (4 bits),
- Thresholds T_1 (8 bits), T_2 (8 bits),
- Number of last watermarked block B_{last} (15 bits),
- Length of R_{sc} (14 bits),
- Compressed recovery sequence R_{sc} (length(R_{sc}) bits)

The LSBs of the first (53 + length(R_{sc})) pixels in I_{pr} are recorded into a binary sequence LSB_r and replaced by C_i bits. Finally, the sequence LSB_r is embbeded in the remaining blocks of I_{pr} to obtain the watermarked image I_w.

4.2.4 Search space

As it is discussed in previous sections, PVO methods use some important parameters as block size dimensions n, m and at least one threshold. Finding the optimal/near optimal set of parameters is not a simple task when the search space is large. On the one hand, Li et al. (2013) and Peng et al. (2014) restrict their search space to a maximum block size of 5×5 and use a single threshold T within a range from 1 to 255. So there are $4,080$ possible solutions. On the other hand, Wang et al. (2015) restricts their search space by setting 4×4 sized blocks, however threshold T_1 takes values from 2 to 255 and T_2 from 1 to $T_1 - 1$, resulting in $32,385$ possible solutions.

In this thesis work, the search space is expanded by allowing a maximum block size of 16×16, and following Wang et al. (2015) idea, two thresholds are used to classify blocks. So there are $5,473,065$ possible solutions in proposed scheme. Carrying out an exhaustive search to find ideal parameters of the algorithm is a very difficult task, since it involves many insertion processes that are computationally expensive. Therefore, an stochastic searching technique such a genetic algorithm is proposed to optimize this process.

4.3 Fine grained GA as optimizer

In Chapter 2, it is explained why genetic algorithms are a computational tool commonly used for optimization and search problems. In the previous section it was shown that

by allowing 16×16 sized blocks, the solutions space is greatly expanded. In this section, a GA is proposed as a searching technique to find the optimal parameters from the extended searching space.

In this work, an integer individual representation is used. Each individual consists of four parameters, which are block dimensions n, m and thresholds T_1, T_2. Thus, a four elements integer array is used.

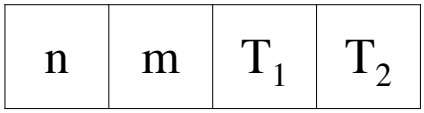


Figure 4.6: Integer individual representation.

Individuals' quality is calculated using Eq. 4.5. A GA will search for those individuals that maximize the fitness function, which is the PSNR value of a watermarked image using corresponding individual parameters.

$$fitness = \begin{cases} PSNR(I_c, I_w) & \text{if all watermark bits are embedded} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

Commonly, population size is a genetic algorithm parameter that remains fixed along generations (Eiben et al., 2003). Population consists of 16 individuals connected in a toroidal way. Fig. 4.7 shows a population distributed over a grid with four rows and four columns. A fine grained distributed GA is deployed to tackle the extended search space due to their better performance reported when dealing with continuous or combinatorial problems (Alba and Dorronsoro, 2009). During every generation (a selection-recombination-mutation-replacement cycle) each current individual interacts with four neighbor individuals as it can be observed in Fig. 4.8.

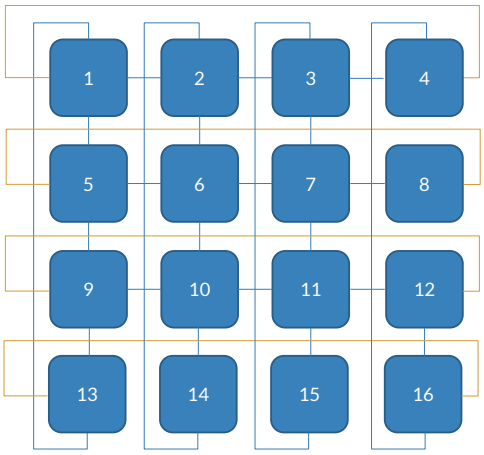


Figure 4.7: Proposed spatial population structure.

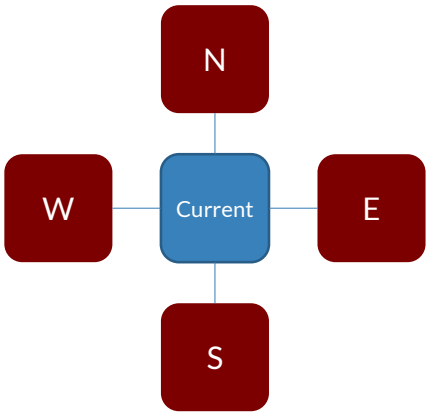


Figure 4.8: Individuals interaction.

Population initialization is performed by creating individuals in a random way, respecting the ranges of values allowed for each parameter.

Selection operator

In Chapter 2, tournament selection, which is a useful mechanism commonly used in genetic algorithms was described. It is based in a local competition to choose the best individuals based on their fitness value. Algorithm 8 is used to select the winner individual. Fitness values from North, South, East, and West neighbors are evaluated. The best

neighbor individual is selected for recombination with the current individual (solution located in neighbourhood center, see Fig 4.8).

Algorithm 8 Selection operator.

Input : North individual P_N,
South individual P_S,
East individual P_E,
West individual P_W
Output : Winner individual P_w

function SELECTION(P_N, P_S, P_E, P_W)
  if $P_N.fitness > P_S.fitness$ **then**
    $P_w \leftarrow P_N$
  else
    $P_w \leftarrow P_S$
  end if
  if $P_E.fitness > P_W.fitness$ **then**
    $P_w \leftarrow P_E$
  end if
  if $P_W.fitness > P_w.fitness$ **then**
    $P_w \leftarrow P_W$
  end if
  return P_w
end function

Crossover operator

Crossover operator is applied using Algorithm 9. Two parent individuals are partitioned to exchange part of their information and produce an offspring individual. A crossover point cp is randomly chosen, then portions between P and P_w up to this point are exchanged to create the offspring P_o.

Algorithm 9 Crossover operator.

Input : Current individual P,
Winner individual P_w
Output : Offspring individual P_o

function CROSSOVER(P, P_w)
  $option \leftarrow$ random integer in [1,2]
  $cp \leftarrow$ random integer in [1,3] ▷ Crossover point
  if $option = 1$ **then**
    P_o from first gene to gene $cp \leftarrow P$ from first gene to gene cp
    P_o from gene $cp + 1$ to last gene $\leftarrow P_w$ from gene $cp + 1$ to last gene

Algorithm 9 (continue) Crossover operator.

```
    else
        P_o from first gene to gene cp ← P_w from first gene to gene cp
        P_o from gene cp + 1 to last gene ← P from gene cp + 1 to last gene
    end if
    return P_o
end function
```

Fig. 4.9 illustrates a crossover operator example. For clarity, one parent is shown in green color and the other one in orange. In this case, the crossover point is 1. Thus, up to this point, an offspring would have from green parent only n parameter and from orange parent m, T_1 and T_2 parameters. Exchanging genetic information creates an offspring, randomly mixing information from both parents.

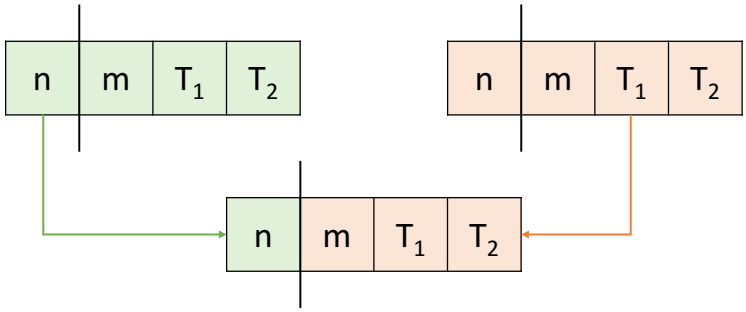


Figure 4.9: Crossover operator example.

Mutation operator

The mutation operator is applied to offspring P_o using Algorithm 10. As a result, this operator returns a slightly modified individual P_m. Commonly, the applied modification is random and unbiased. This operator is applied to P_o after recombination. For this work creep mutation is used, as shown in Chapter 2. A parameter from P_o is randomly chosen to be slightly modified, always within corresponding permitted ranges for integer representation. Proposed modifications for creep mutation are presented below.

- Block size mutation

  $n = n + mut_{block}$, where $mut_{block} \in \{-2, -1, 1, 2\}$ randomly selected.

$m = m + mut_{block}$, where $mut_{block} \in \{-2, -1, 1, 2\}$ randomly selected.

- Thresholds mutation

  $T_1 = T_1 + mut_{threshold}$, where $mut_{threshold} \in \{-10, -9, -8, \ldots, 8, 9, 10\}$ randomly selected.

  $T_2 = T_2 + mut_{threshold}$, where $mut_{threshold} \in \{-10, -9, -8, \ldots, 8, 9, 10\}$ randomly selected.

Algorithm 10 Mutation operator.

Input : Offspring individual P_o,
List of already evaluated individuals E
Output : Mutated individual P_m

function MUTATION(P_o, E)
  $P_m \leftarrow P_o$
  repeat
    $parameter \leftarrow$ random integer in $[1, 4]$
    $mut_{bloque} \leftarrow$ random integer in $[-2, 2]$
    $mut_{threshold} \leftarrow$ random integer in $[-10, 10]$
    if $parameter = 1$ **then**
      $P_m.n \leftarrow P_m.n + mut_{bloque}$, as long as $4 \geq P_m.n \leq 16$
    else if $parameter = 2$ **then**
      $P_m.m \leftarrow P_m.m + mut_{bloque}$, as long as $4 \geq P_m.m \leq 16$
    else if $parameter = 3$ **then**
      $P_m.T_1 \leftarrow P_m.T_1 + mut_{threshold}$, as long as $2 \geq P_m.T_1 \leq 255$
    else
      $P_m.T_2 \leftarrow P_m.T_2 + mut_{threshold}$, as long as $1 \geq P_m.T_2 < P_m.T_1$
    end if
  until $P_m \notin E$
  append P_m to E
  return P_m, E
end function

In order to avoid evaluating a solution more than once and in consequence to reduce computational cost, a list of solutions already evaluated is kept. After reproduction (crossover and mutation) if an offspring belongs to that list, mutation is carried out again. In this way, it is ensured that in each generation different solutions will be tested.

The current individual is replaced only if the offspring returned by mutation operator has better fitness value, otherwise the current individual will survive. This condition

ensures that the best individuals in the population will survive during each generation. The process is repeated up to a number of established generations is reached.

The complete PVO-GA insertion process, which consider all previous stages is performed using Algorithm 11. It is required a cover image I_c, a message to be hidden m_h and a secret key k to compute the MAC_t. The watermark w is created by concatenating m_h and MAC_t. After that, I_c undergoes the proposed image pre-processing to get I_{pr} and R_s, which is compressed using arithmetic codding. Finally, the proposed GA is used to intelligently embed w into I_{pr} using the unequal sized block partition strategy. It is expected that the PSNR will be increased after each generation. At the end of the simulation, the GA will return an optimal or near optimal solution, which represents the best PVO parameter configuration that produces the best result for a given payload.

Algorithm 11 PVO-GA insertion process.

Input : Cover image I_c of size $N \times M$,
Message m_h,
Secret key k
Output : Watermarked image I_w

function PVOGAINSERTIONPROCESS(I_c, m_h, k)
  $I_{cs} \leftarrow (string)I_c$
  $MAC_t \leftarrow$ SHA256MAC($I_{cs} \| m_h, k$) ▷ Message Authentication Code
  $I_{pr}, R_s \leftarrow$ IMAGEPREPROCESSING(I_c)
  $R_{sc} \leftarrow$ compress(R_s) ▷ Compress R_s using arithmetic coding
  $w \leftarrow m_h \| MAC_t$
  for $i \leftarrow 1$ **to** 16 **do**
    $individual.n \leftarrow$ random integer between $[4, 16]$
    $individual.m \leftarrow$ random integer between $[4, 16]$
    $individual.T_1 \leftarrow$ random integer between $[2, 255]$
    $individual.T_2 \leftarrow$ random integer between $[1, individual.T_1 - 1]$
    $Individuals_i \leftarrow individual$
  end for
  $Population \leftarrow$ connect $Individuals$ as shown in Fig. 4.7
  $E \leftarrow \emptyset$
  for all individual $P \in Population$ **do**
    $I_{wP} \leftarrow$WATERMARKEMBBEDING($I_{pr}, R_{sc}, w, P.n, P.m, P.T_1, P.T_2$)
    $P.fitness \leftarrow$ compute fitness value using Eq. 4.5
    append P to E
  end for

Algorithm 11 (continue)PVO-GA insertion process.

for $i \leftarrow 1$ **to** number of generations **do**
  for all individual $P \in Population$ **do**
    $P_w \leftarrow$ SELECTION(P_N, P_S, P_E, P_W) ▷ Winner
    $P_o \leftarrow$ CROSSOVER(P, P_w) ▷ Offspring
    $P_m, E \leftarrow$ MUTATION(P_o, E) ▷ Mutated
    $I_{wP} \leftarrow$WATERMARKEMBBEDING$(I_{pr}, R_{sc}, w, P_m.n, P_m.m, P_m.T_1, P_m.T_2)$
    $P_m.fitness \leftarrow$ compute fitness value using Eq. 4.5
    if $P.fitness \leq P_m.fitness$ **then**
      $P \leftarrow P_m$
    end if
  end for
end for
$P_b \leftarrow$ individual into $Population$ with best fitness value
$I_w \leftarrow$ WATERMARKEMBBEDING$(I_{pr}, R_{sc}, w, P_b.n, P_b.m, P_b.T_1, P_b.T_2)$
return I_w
end function

Table 4.1 shows the parameter setting used in this thesis.

Parameter	Type
Individual representation	Integer
Population size	16
Number of Generations	20, 100
Replacement mechanism	Keep best
Mutation type	Creep Mutation
Crossover type	One crossover point

Table 4.1: GA parameter setting.

4.4 Watermark extraction and authentication process

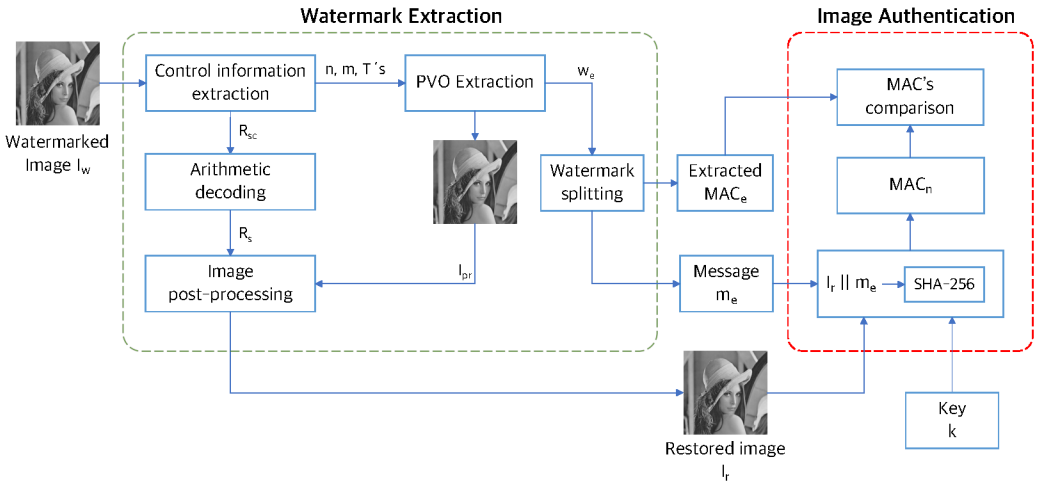


Figure 4.10: PVO-GA extraction and authentication processes.

After I_w is stored or transferred to the receiver, the same procedure as in the embedding side is applied, but in the reverse order for watermark extraction. Algorithm 12 is used to perform the extraction and authentication phases. First, control information is extracted from the watermarked image. Control information contains PVO parameters used during insertion process, those found by the fine grained GA, i.e., n, m, T_1, T_2, the compressed recovery sequence generated during image pre-processing stage R_{sc} and the number of the last block with watermark bits B_{last}. I_w is first divided into blocks of size $n \times m$, then the watermark w_e is extracted. After watermark extraction, I_w should be the same as that obtained during image pre-processing stage because all watermark bits have been removed. So image I_{pr} is created using $Blocks_w$. The recovery sequence R_s is obtained by decompressing R_{sc} using arithmetic decoding. Then I_{pr} undergoes the proposed image post-processing. Finally, the watermark is separated into the the extracted message m_e and the extracted message authentication code MAC_e.

Algorithm 12 Extraction and authentication process.

Input : Watermarked image I_w,
Secret key k
Output : Cover image I_c and Message m_h

function EXTRACTIONANDAUTHENTICATION(I_w, k)
  $Ctrl_{inf} \leftarrow$ get control information
  $n, m, T_1, T_2, B_{last}, length(R_{sc}), R_{sc} \leftarrow$ split($Ctrl_{inf}$)
  $Blocks_w \leftarrow$ partition(I_w, n, m) ▷ Divide I_w into blocks of size $n \times m$
  $block_n \leftarrow 0$; $w_e \leftarrow$ empty string

// Watermark extraction and image restoration
  for all $B_w \in Blocks_w$ **do**
    $block_n \leftarrow block_n + 1$
    $NL \leftarrow$ COMPUTENOISELEVEL(B_w)
    if $T_2 \geq NL$ **then** ▷ Flat block
      $sb1_w, sb2_w, sb3_w, sb4_w \leftarrow$ GETSUBBLOCKS(B_w)
      REMOVEWATERMARKBITS($sb1_w, w_e$)
      REMOVEWATERMARKBITS($sb2_w, w_e$)
      REMOVEWATERMARKBITS($sb3_w, w_e$)
      REMOVEWATERMARKBITS($sb4_w, w_e$)
    else if $T_2 < NL \leq T_1$ **then** ▷ Normal block
      REMOVEWATERMARKBITS(B_w, w_e)
    else if $NL > T_1$ **then** ▷ Rough block
      Do nothing
    end if
    if $block_n = B_{last}$ **then**
      break
    end if
  end for
  $I_{pr} \leftarrow$ join($Blocks_w$) ▷ Create I_{pr} using $Blocks_w$
  $R_s \leftarrow$ decompress(R_{sc}) ▷ Decompress R_{sc} using arithmetic decoding
  $I_r \leftarrow$ IMAGEPOSTPROCESSING(I_{pr}, R_s)
  $m_e, MAC_e \leftarrow$ split(w_e)

// Authentication phase
  $I_{rs} \leftarrow (string)I_r$
  $MAC_n \leftarrow$ SHA256MAC($I_{rs} \| m_e, k$)
  if $MAC_n \neq MAC_e$ **then**
    show message "I_w was altered"
    return 0
  else
    show message "I_w was not altered"
    $I_c \leftarrow I_r$; $m_h \leftarrow m_e$
    return I_c, m_h
  end if
end function

Image Authentication

During the insertion process, I_c is prepared to perform image authentication by embedding a watermark containing a MAC. Authentication phase is carried out at the receiver, after watermark extraction. The input parameters are the recovered image I_r, extracted message m_e, extracted message authentication code MAC_e and the shared secret key k, the same used in the embedding process, as shown in Fig. 4.11.

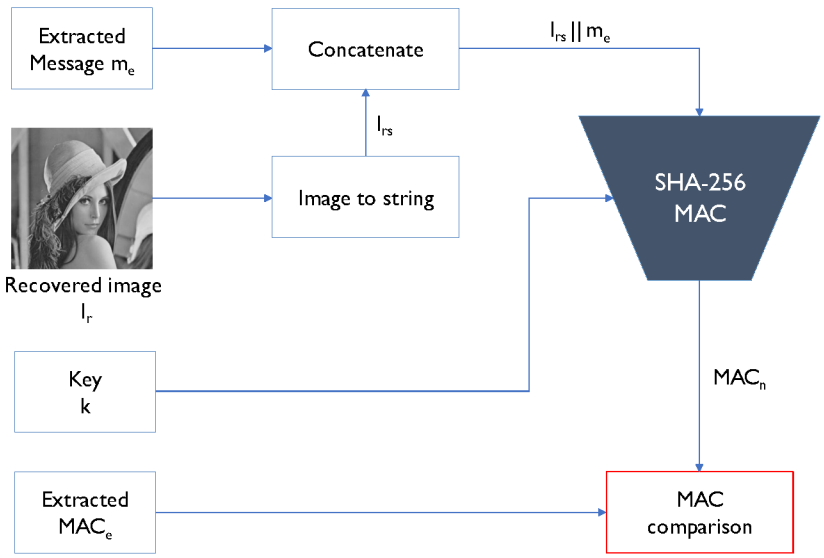


Figure 4.11: Authentication process.

Let MAC_e be the extracted MAC value, I_r the recovered image and m_e the extracted message. Exact authentication is verified by following the next process. First I_r pixels are converted to a string I_{rs}. Extracted message m_e and I_{rs} are concatenated to get $I_{rs}\|m_e$ string, which together with the secret key k used in insertion process will be the argument to the SHA-256MAC algorithm. This function returns a 256-bit MAC value MAC_n. Finally, MAC_n is compared with MAC_e, if they match, the receiver accepts the message and assures himself that the message was originated by a trusted party.

4.5 Discussion

In this chapter, a reversible watermarking scheme based on PVO technique was presented. The proposed scheme allows image authentication by embedding a message authentication code as part of the watermark. We proposed some stages to improve PVO performance such as image pre-processing to increase embedding capacity and unequal sized block partition strategy to reduce the introduced distortion during embedding process. A fine grained GA algorithm was developed to find the best set of parameters of the proposed scheme due to large solutions space. Finally, watermark extraction and authentication processes were explained. In the next chapter the experiments that were carried out to support the proposed scheme are presented.

5

Experiments and Results

In this chapter, an empirical assessment carried out to evaluate the proposed reversible watermarking scheme performance is detailed. Images of two well-known datasets within watermarking field: SIPI and Kodak datasets, are used for embedding. Three experimental stages are defined. Initially, improvement of embedding capacity is evaluated by applying the proposed image pre-processing technique. After, the distortion levels caused by the embedding process are evaluated. On the one hand, 10,000 bits watermarks are inserted into images. On the other hand, different sized watermarks are embedded to show the proposed scheme performance when payload is increased. Experiments are finalized with the GA performance evaluation. A number of simulations are done to analyze PSNR improvement per generation when increasing the watermark size. Also, the GA effectiveness when tackling embedding process is demonstrated by comparing the best final individuals obtained in 100 simulations of the proposed scheme and the best solution offered by Wang et al. (2015) scheme, which restricts block sizes. Finally, PVO-GA performance is compared against other PVO methods such as Li et al. (2013), Peng et al. (2014) and Wang et al. (2015). Distortion levels are shown in terms of PSNR while embedding capacity is indicated with the number of inserted bits. Results show an improvement on distortion levels and embedding capacity when compared against other PVO based methods, besides reducing the computational effort required to find the best parameters. The chapter concludes with a discussion about the obtained results.

5.1 Datasets and setup

Experiments are applied using two benchmark image datasets, which have been used in previous works. This thesis deals with natural images. Although there is not a formal definition of what a natural image is, those images that capture somewhere in the world by using a common digital camera are considered as natural images. Hyvärinen et al. (2009) show that natural images do not follow a uniform distribution, so their pixel values are closely from each other. Then natural images contain a lot of redundancy, unlike most artificial or randomly generated images.

Eight standard images including *Lena*, *Baboon*, *Peppers*, *Airplane*, *Airplane (F16)*, *Boat*, *Lake* and *Aerial* make up the **SIPI** dataset, from Signal and Image Processing Institute, which is shown in Fig. 5.1. All SIPI dataset images are 512×512 pixels size, in grayscale representation and they are available at the University of Southern California site[1]. These images were selected because they have been widely used not only in the watermarking arena, but also in compression, steganography and application domains related to image analysis or image processing.

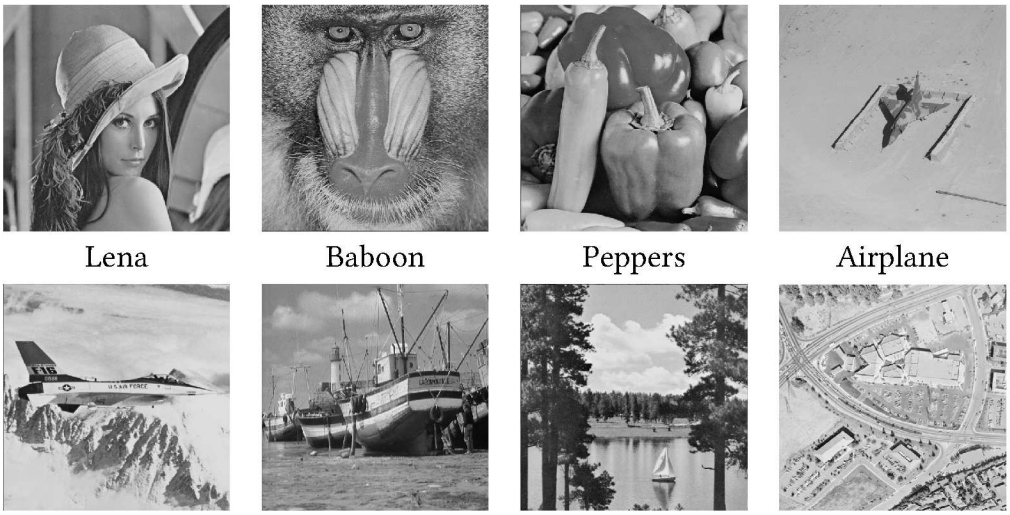


Figure 5.1: SIPI dataset images.

[1] `http://sipi.usc.edu/database/`

Figure 5.2: Kodak dataset images.

Kodak[2] is the second dataset selected for evaluation. Images are shown in Fig. **??**. This data set is provided by the Eastman Kodak Company and contains 24 natural images of 512×768 and 768×512 sizes. These images are available in color representation, so they were transformed to grayscale representation. Every pixel from all images included in both datasets is represented with 8 bits, so pixels can vary from 0 to 255 integer values. The proposed scheme, including embedding, extraction and authentication algorithms was implemented in MATLAB 2017b. The genetic algorithm specifications are shown in Table 4.1. Finally, for evaluation purposes, watermarks are created by generating random binary strings, as in previous works.

5.2 Embedding Capacity

As it was mentioned in Chapter 1, a research challenge related to reversible watermarking schemes is to improve the embedding capacity. We proposed in Chapter 4 an image pre-processing stage, which allows to make available all data blocks for the embedding process. In this experimental setup, benchmark images are used to apply the proposed image pre-processing. This experiment is performed in order to compare embedding capacity that can be offered by images before and after modification.

As it is explained in the previous chapter, the predictor proposed by Peng et al. (2014) is used to embed watermark bits. Maximum embedding capacity for a given block size is reached by setting a threshold value to 255, thus all blocks will be used during insertion process. Therefore, the embedding capacity increment relies on the number of 0's and 255's pixel values within the image.

Fig. 5.2 shows maximum embedding capacity through different block sizes, before and after applying the proposed image pre-processing stage to *kodim24*. This image contains zero pixels with 0 intensity value and $58,932$ pixels with intensity value 255. When image pre-processing is applied to *kodim24*, embedding capacity is increased from $59,579$ bits to $69,249$ bits, i.e., it is increased over 16%, for 2×2 sized blocks, but this increment

[2] `http://r0k.us/graphics/kodak/`

is also maintained for different block sizes.

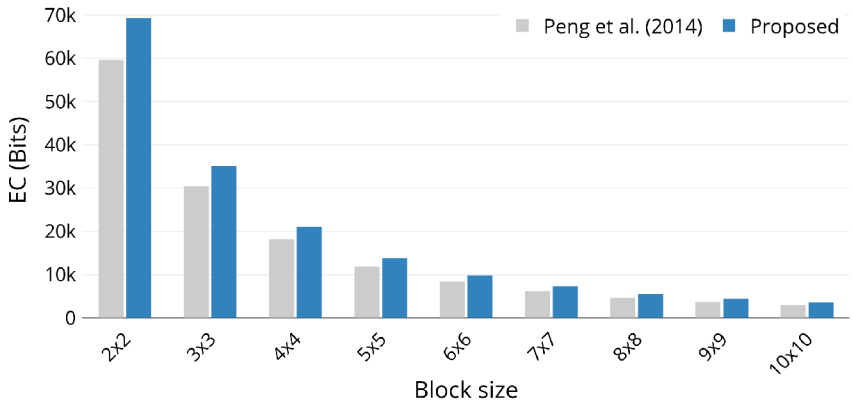


Figure 5.3: Image pre-processing effect on *kodim24*.

Another example is shown in Fig. 5.3. The *kodim13* image contains $8,535$ pixels with intensity value 0 and $2,304$ pixels with value 255. In this case, maximum embedding capacity is increased from $25,383$ bits to $27,522$ bits, that is over 8%. The increment is also maintained for different block sizes as in the previous example.

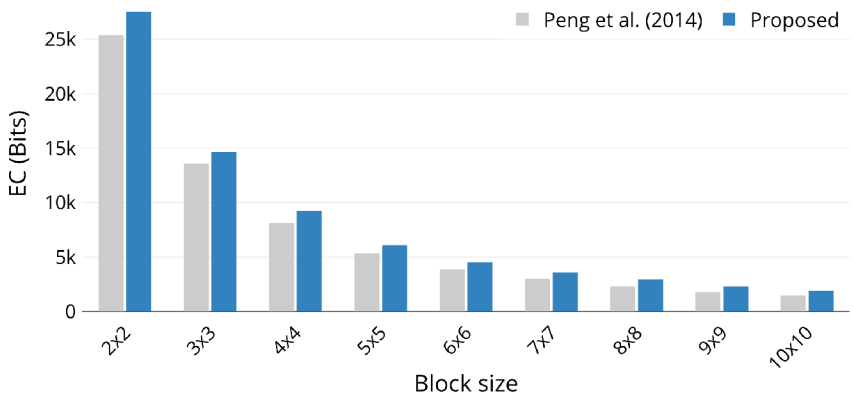


Figure 5.4: Image pre-processing effect on *kodim13*.

Fig. 5.4 shows the maximum embedding capacity for some given block sizes using the *kodim19* image, before and after applying image pre-processing. This image only has 3 pixels with 0 intensity value and 6 pixels with 255 intensity value. In this example, it can be observed that in the worst case, the proposed image pre-processing approach provides

the same embedding capacity as the method proposed by Peng et al.. This happens when images do not contain or contain very few pixels with 0 or 255 intensity values since both images are very similar.

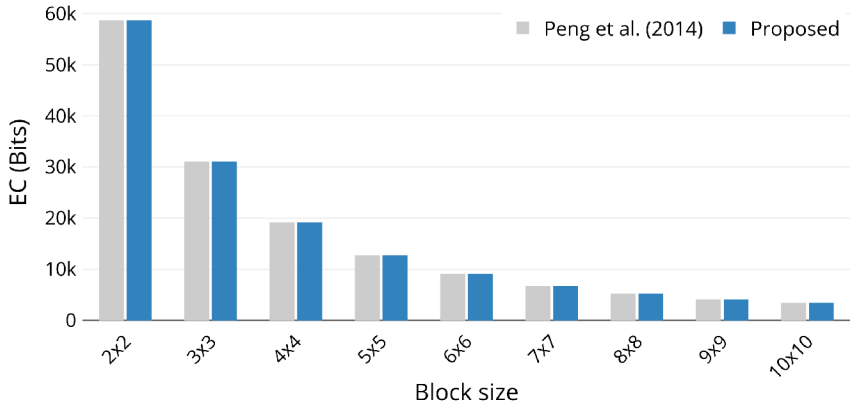


Figure 5.5: Image pre-processing effect on *kodim19*.

From the above examples, it can be observed that embedding capacity improvement depends on the number of pixel values 0 or 255 into cover image. Fig. 5.5 shows the maximum embedding capacity for the entire Kodak dataset using different block sizes. Maximum embedding capacity is obtained when block size is set to 2×2. After image pre-possessing is applied, embedding capacity is increased over 5% in average for every block size.

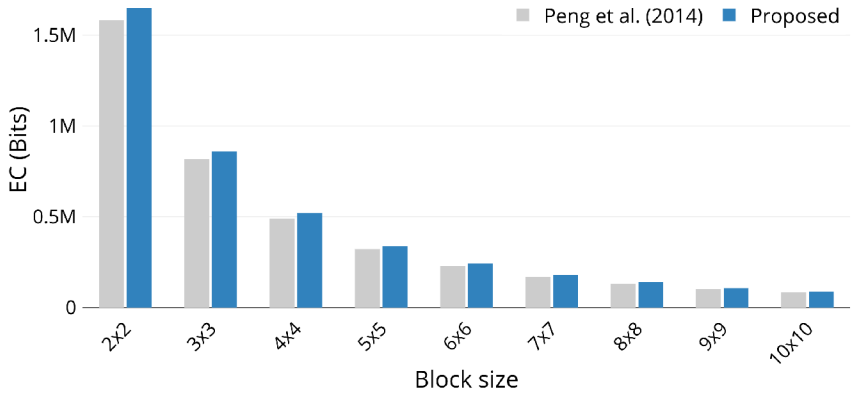


Figure 5.6: Image pre-processing effect on Kodak dataset.

In SIPI dataset, embedding capacity improvement is less than that obtained in Kodak

dataset, see Fig. 5.6. After applying the proposed image pre-processing, embedding capacity is increased only over 1% for every block size. This happens because *Lena*, *Peppers*, *Airplane*, *Airplane (F16)* and *Lake* images does not contain pixels with values 0 or 255, however in *Lake* image, embedding capacity is increased over 11%.

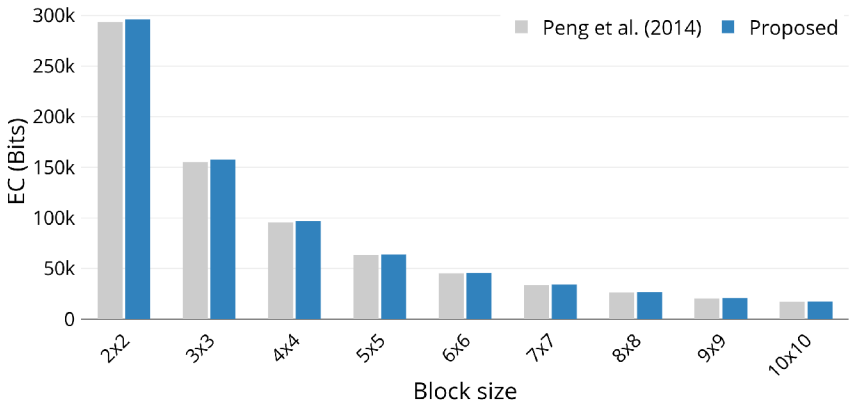


Figure 5.7: Image pre-processing effect on SIPI dataset.

Finally, on the one hand, when an image does not contain pixels with intensity values 0 or 255, embedding capacity is not increased, however control information is reduced since a recovery sequence R_s is not required. On the other hand, a maximum embedding increment up to 16% is achieved on the *kodim24* image, where compressed recovery sequence R_{sc} has a length of 1287 bits.

5.3 Distortion

Li et al. (2013) and Peng et al. (2014) showed that using larger sized blocks generates higher PSNR, but lower embedding capacity and viceversa, thus performance of PVO based methods depends on the block size. In the previous chapter, it was presented an unequal sized block partition strategy to divide an image using blocks of different and/or unequal sizes. Experiments were carried out to show distortion generated by using the proposed strategy.

Watermark embedding

In this experiment all benchmark images were set up to embed a watermark of $10,000$ bits size, in order to measure image quality between cover and watermarked images. Table 5.1 shows imperceptibility levels in terms of PSNR.

	PSNR (dB)				
Image	**Li et al. (2013)**	**Peng et al. (2014)**	**Wang et al. (2015)**	**PVO-GA 20**	**PVO-GA 100**
Lena	60.2872	60.5009	60.4441	**60.7668**	**60.8188**
Baboon	53.58	53.5766	54.657	54.6568	**54.6581**
Peppers	58.904	59.0457	58.9967	**59.1578**	**59.1813**
Airplane	-	62.4642	61.8743	**62.7167**	**62.7521**
Airplane (F16)	62.035	62.9682	63.4406	63.426	**63.5075**
Boat	58.1231	58.3533	58.4528	**58.5649**	**58.5946**
Lake	58.2499	58.9308	59.5689	**59.5746**	**59.5779**
Aerial	60.1965	60.9275	61.3409	**61.3844**	**61.3886**
kodim01	58.9642	62.4256	63.2471	63.0782	**63.2575**
kodim02	62.4327	**64.0137**	63.9118	63.8607	63.9614
kodim03	63.811	**65.3854**	65.0014	65.0316	65.0642
kodim04	61.8899	63.5303	63.0439	63.5275	**63.5525**
kodim05	60.1577	61.9523	62.0582	**62.3026**	**62.3657**
kodim06	61.7535	64.7625	**65.5141**	65.3445	65.508
kodim07	63.4008	**65.0327**	64.6437	64.785	64.8235
kodim08	57.7837	56.49	**60.1349**	58.8368	58.7243
kodim09	62.8425	**63.7188**	62.9094	63.4932	63.5184
kodim10	62.18	**63.1338**	62.788	62.4929	62.5028
kodim11	62.4634	65.0034	**65.0208**	64.8473	64.8696
kodim12	62.8204	64.4459	**64.4482**	64.3122	64.3477
kodim13	55.735	57.3085	58.8127	**58.8817**	**58.978**
kodim14	59.7997	61.4638	**62.1679**	62.0083	62.018
kodim15	62.8961	**64.3013**	63.9792	62.2888	62.2994
kodim16	63.0545	**64.9591**	64.8369	64.7468	64.7666
kodim17	62.0348	63.675	63.6802	**64.0854**	**64.0967**
kodim18	60.3194	60.9583	61.0146	**61.1774**	**61.1843**
kodim19	62.4854	63.4442	63.3429	**63.5157**	**63.549**
kodim20	62.3378	65.0082	64.9853	**65.0253**	**65.0597**
kodim21	63.1035	63.5748	63.5857	63.5814	**63.6048**
kodim22	61.8006	**62.8632**	62.8061	62.7934	62.822
kodim23	63.3594	**64.6141**	64.4515	63.8752	63.9181
kodim24	60.846	61.7416	**63.2831**	59.8931	59.9351
Average SIPI	58.7680	59.5959	59.8469	**60.0310**	**60.0599**
Average Kodak	61.5947	63.0753	**63.3195**	63.0744	63.1136

Table 5.1: 10,000 bits embedding results.

Lets see *Lena* row. The watermark was embedded using PVO strategies proposed by Li et al. (2013), Peng et al. (2014) and Wang et al. (2015), where obtained PSNR levels are 60.2872, 60.5009 and 60.4441 respectively. The proposed PVO-GA method was run setting 20 (PVO-GA 20) and 100 (PVO-GA 100) generations obtaining 60.7668 and 60.8188 PSNR values respectively, which are greater than that obtained with previous methods. It can be observed that PVO-GA outperforms previous methods in most tested images,

best PSNR value for every image is highlighted in bold. Despite the fact that the proposed scheme does not improves distortion in all images, it can be observed that PVO-GA finds similar PSNR levels in most cases. Image pre-processing stage distorts the cover image from the beginning, so images with many intensity 0 and 255 values will be more affected, see *kodim08* and *kodim24*. However, as seen in the previous section, embedding capacity is increased because all data blocks are used during the insertion process, which would be a good way to embed more information.

Embedding distinct watermarks sizes

In this experiment, SIPI dataset images were used to embed different sized watermarks in order to assess the proposed PVO-GA scheme's performance when watermarks size is increased.

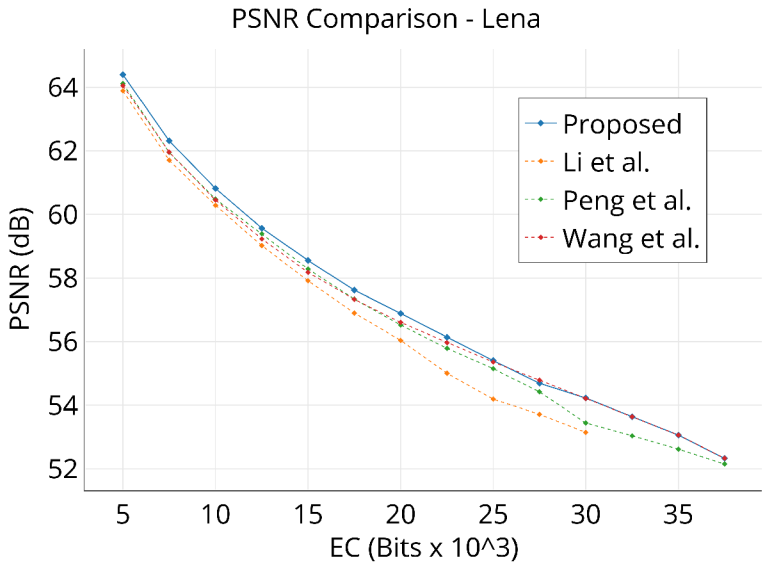


Figure 5.8: PSNR comparison in *Lena* image.

Fig. 5.7 shows PSNR levels using the *Lena* image. This image is considered a medium complexity image, possessing minor smooth areas and high contrast. The figure clearly illustrates that PVO-GA outperforms or has a similar performance compared to other three PVO based methods, for watermarks of different sizes. This is possible because PVO-GA allows to expand the search space considering blocks larger than 4×4. Af-

ter those larger blocks can be divided into unequal sized sub-blocks, so PVO-GA takes advantage of different blocks' features.

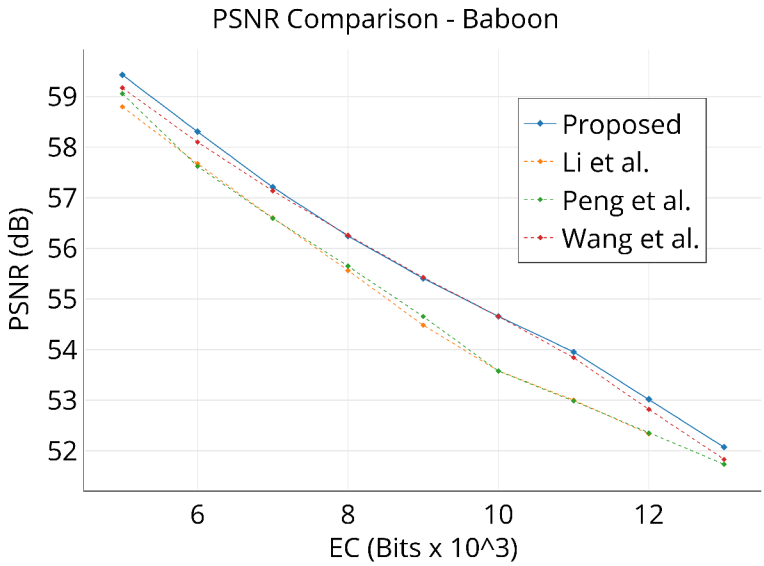


Figure 5.9: PSNR comparison in *Baboon* image.

Fig. 5.8 shows PSNR levels using the *Baboon* image. This image is widely used in image processing because of its complexity, since it is considered a textured image. It can be observed that when embedding capacity is less than $8,000$ bits, PVO-GA outperforms the method proposed by Wang et al. (2015), this happens because PVO-GA provides solutions using block sizes greater than 4×4. On the other hand, from $8,000$ bits to $13,000$ bits embedding capacity, PVO-GA provides solutions within 4×4 sized blocks, as the method of Wang et al. (2015).

It can also be observed that the bigger the watermark, the PSNR levels are improved regarding the method proposed by Wang et al. (2015). It is because distortion caused by image pre-processing is fixed by the insertion process itself. Minimum and maximum pixel values into pre-processed image are 1 and 254 respectively. By implicit characteristics of PVO based methods, for example, where maximum and minimum pixel values are modified, some of these pixels would recover their original values when a 1 watermark bit is inserted.

5.4 Genetic Algorithm Performance

It is mentioned in Chapter 4, that the search space was expanded allowing solutions with block sizes from 4×4 to 16×16, in such a way a genetic algorithm is implemented to search for a solution. Therefore, a thorough empirical assessment is carried out to evaluate the behavior and effectiveness of the proposed genetic algorithm.

Fitness value increment

In this experiment the *kodim01* image is used to embed watermarks using PVO-GA. Watermarks size is increased to analyze genetic algorithm performance to find solutions using different sized watermarks. The proposed GA was executed for 100 generations. Fig. 5.9 shows the mean of 10 genetic algorithm simulations per watermark using the *kodim01* image. It can be observed that PSNR levels increase after every generation. At the end of each simulation, the genetic algorithm produces optimal/near optimal individuals, which gives the best set of parameters, it means the best block size and thresholds values to embed a specific watermark causing minimal distortion.

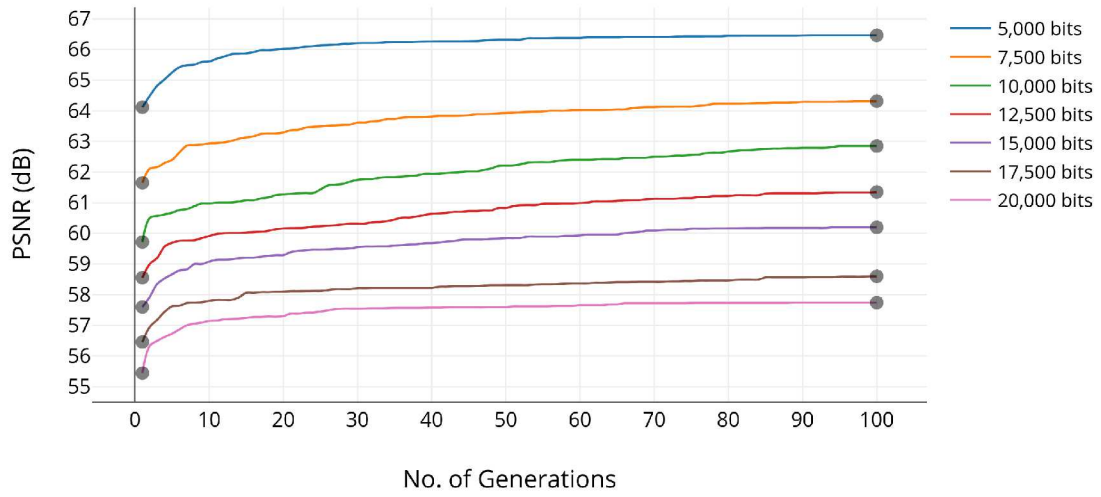


Figure 5.10: Fitness values after each generation in the *kodim01* image.

Genetic Algorithm Effectiveness

In this experiment, PVO-GA is used to embed a $10,000 bits$ watermark in the Lena image. PVO-GA was executed 100 times during 20 and 100 generations. This experiment is carried out to analyze final fitness values provided by the proposed PVO-GA method.

Fig. 5.10 shows PSNR levels when $10,000$ bits are embedded in the Lena image, for 100 executions, 50 executions are carried out setting 20 generations and 50 more setting 100 generations. The red line represents the obtained PSNR using the method of Wang et al., which is 60.4529.

On the one hand, results when PVO-GA is set to run 20 generations are shown in the orange box plot. It can be observed that minimum and maximum obtained PSNR levels are 60.1749 and 60.8004, respectively. For these 50 simulations PVO-GA reported 60.4965 as median and 60.517 as mean with standard deviation of 0.1693. Median and mean are over the PSNR obtained with the method proposed by Wang et al. (2015), which represents the best solution using 4×4 sized blocks.

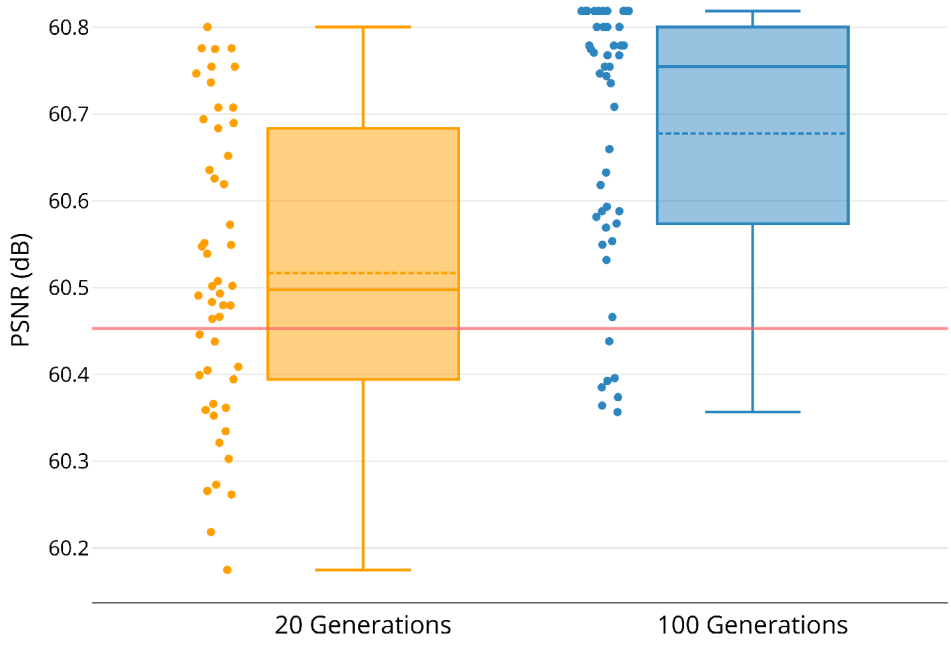


Figure 5.11: PVO-GA final fitness values - 100 simulations for 20 and 100 generations.

On the other hand, the blue box plot shows PSNR levels using PVO-GA set to 100 generations. It can be observed that minimum and maximum obtained PSNR levels are 60.3567 and 60.8188, respectively, which improved results over evolving only 20 generations. In this case PVO-GA obtained 60.7547 as median and 60.6776 as mean with a standard deviation of 0.1536, which also overcomes the PSNR obtained with the method of Wang et al. (2015).

For the above, it is clear that the more generations are allowed, fitness values provided by genetic algorithm will be improved. Fig. 5.10 shows that blue box plot is shorter than orange box plot, since solutions are distributed in a smaller range of PSNR values, this happens because PVO-GA running 100 generations evolves more individuals than running only 20 generations, so more solutions are explored to find the optimal/near optimal individual.

Finally, PVO-GA overcomes the method proposed by Wang et al. (2015) in 32 out of 50 simulations using 20 generations and 47 out of 50 simulations while evolving 100 generations. Despite the fact that, PVO-GA does not overcome reference PSNR at all times, it provides close PSNR levels, however, there is an evident computational cost decrement since PVO-GA evolving 20 generations embed 320 times a watermark while evolving 100 generations embeds $1,600$ times a watermark, a much lower number of insertions compared to Wang et al. (2015), which inserts $32,385$ times a watermark to find the best thresholds.

5.5 Discussion

In this chapter experiments that were carried out to demonstrate PVO-GA performance were detailed. It was shown that proposed reversible watermarking scheme outperforms PVO based methods proposed by Li et al. (2013), Peng et al. (2014), and Wang et al. (2015) in most cases. Initially, we expected to overcome PSNR and number of bits inserted provided by PVO based schemes. Thus, both aims were addressed as follows. Embedding capacity was improved by applying image pre-processing while distortion was reduced

by adopting unequal sized block partition strategy. A genetic algorithm was adopted as search technique because search space was expanded by allowing the use of larger than 4×4 sized blocks. This intelligent search technique allows to find a good trade-off between imperceptibility and embedding capacity properties. The proposed genetic algorithm clearly reduces the number of insertion processes because this number is in function of 16 individuals per number of generations, so when 100 generations are evolved, $1,600$ insertions will be executed, in contrast to the method proposed by Wang et al. (2015), where a greater number of insertions are executed.

6

Conclusions and future work

In this thesis, we have proposed and developed PVO-GA, an adaptive reversible watermarking scheme based on PVO technique for image authentication. Research was focused on three important points.

- To improve PVO embedding capacity.
- To reduce distortion provoked when a watermark is embedded.
- To optimize the insertion process when looking for PVO's optimal parameters.

In order to improve embedding capacity, we proposed an image pre-processing stage, which modifies the pixel luminance values 0 and 255 to 1 and 254 respectively. In this way, all image blocks are available during the insertion process after modification. In the best case embedding capacity was significantly improved over 16%. In the worst case the proposed scheme offered the same embedding capacity as the method proposed by Peng et al. (2014).

Distortion was slightly reduced by adopting an unequal sized block partition strategy. This strategy allows to use larger than 4×4 sized blocks, which directly impacted in the watermarked image quality. Although obtained PSNR values are not significantly increased, they can be considered competitive against other PVO based methods.

A genetic algorithm was implemented to optimize the insertion process performance. A decentralized fine grained version of a genetic algorithm was selected due to its improved searching performance. This evolutionary algorithm searches to find optimal/near optimal parameters. In other words, the best trade-off between image quality and embedding capacity for a given watermark. The number of insertion processes executed is clearly reduced by using the proposed genetic algorithm, therefore the computational cost is also reduced.

A thorough experimental assessment was carried out in order to support the proposed reversible watermarking scheme. Results showed that the proposed PVO-GA is capable of providing an effective trade-off between imperceptibility and embedding capacity, but also it requires fewer insertions processes when compared with existing reversible watermarking schemes based on the same PVO technique. Therefore, computational cost is significantly reduced

Future Work

From the above, the following ideas are presented for future research guidelines.

- PVO methods partition an image in blocks, they are used to embed up to two bits per block. However, embedding relies on prediction error calculation (Eq. 3.1), so there are blocks in which bits are not inserted. As future work it is proposed to study how to ensure the insertion of at least two bits per block by combining another technique such as compression during the insertion process.

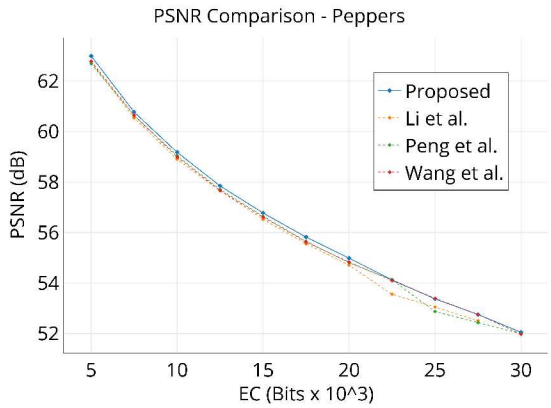
- In this work, exact authentication is used. This is the most basic kind of authentication. If above point is reached, it is worth studying how to add to the proposed scheme the ability to identify those regions of the image that have been corrupted.

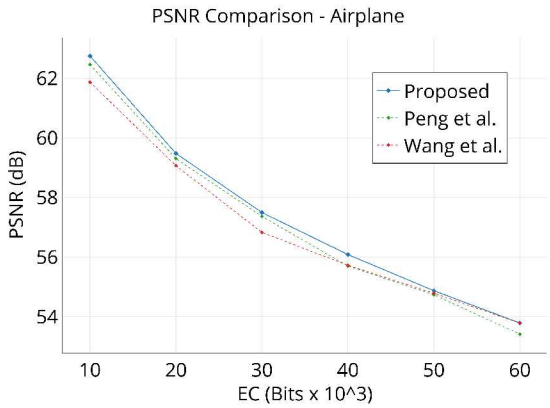
- All PVO based reversible watermarking schemes work in spatial domain. A future direction would be to move to the frequency domain and use coefficients from transforms such as DCT or IWT.

A

Appendix: Obtained PSNR Levels on SIPI Dataset

In Chapter 5, the imperceptibility results for the Lena and Baboon images were presented. This appendix shows the PSNR levels obtained in the other SIPI images when different sized watermarks are embedded.





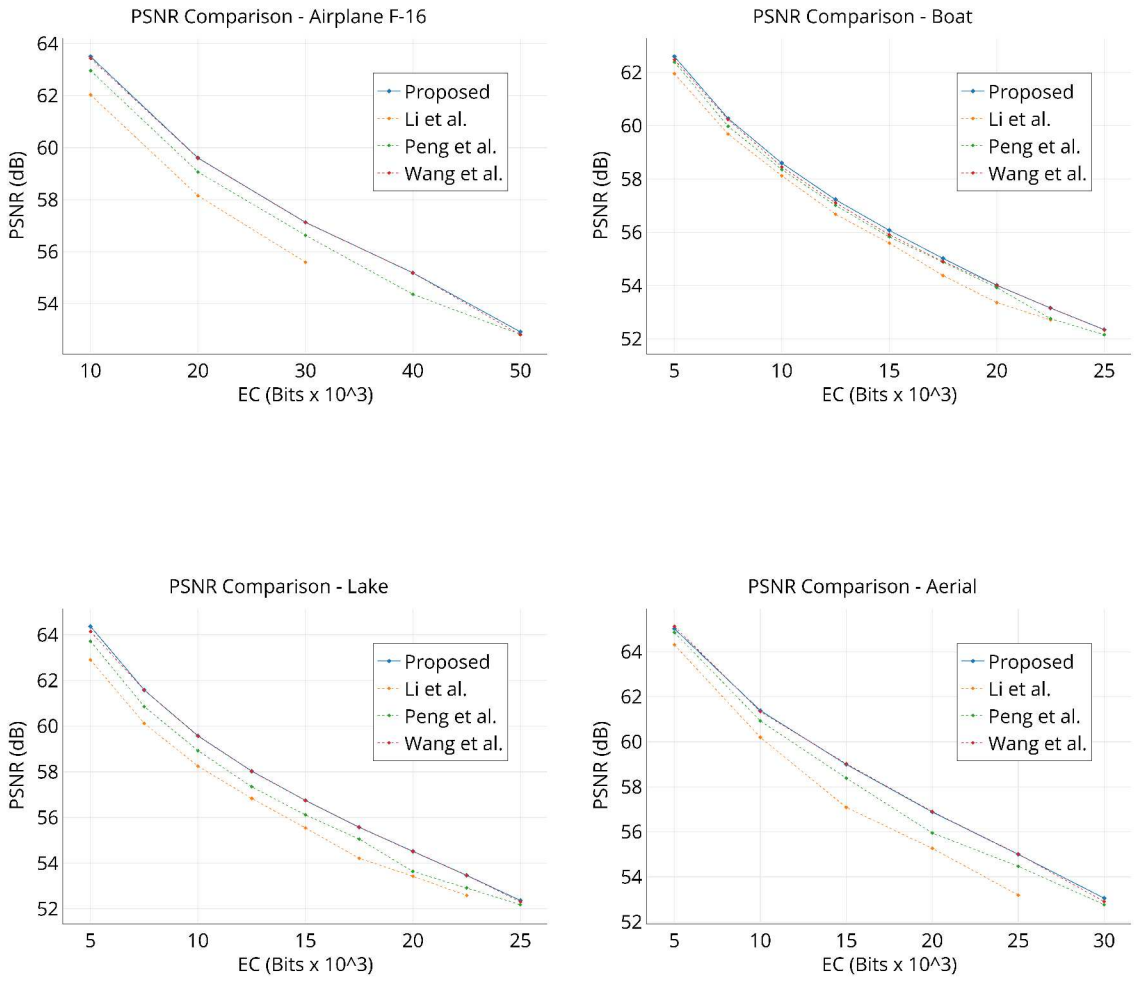


Figure A.1: PSNR results on SIPI dataset images.

Bibliography

Alba, E. and Dorronsoro, B. (2009). *Cellular genetic algorithms*, volume 42. Springer Science & Business Media.

An, L., Gao, X., Yuan, Y., and Tao, D. (2012). Robust lossless data hiding using clustering and statistical quantity histogram. *Neurocomputing*, 77(1):1–11.

Arsalan, M., Malik, S. A., and Khan, A. (2012). Intelligent reversible watermarking in integer wavelet domain for medical images. *Journal of Systems and Software*, 85(4):883–894.

Arsalan, M., Qureshi, A. S., Khan, A., and Rajarajan, M. (2017). Protection of medical images and patient related information in healthcare: Using an intelligent and reversible watermarking technique. *Applied Soft Computing*, 51:168–179.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK.

Barton, J. M. (1997). Method and apparatus for embedding authentication information within digital data. US Patent 5,646,997.

Chen, M., Chen, Z., Zeng, X., and Xiong, Z. (2009). Reversible image watermarking based on full context prediction. *IEEE Int. Conf. on Image processing*, pages 4253–4256.

Coltuc, D. (2011). Improved embedding for prediction-based reversible watermarking. *IEEE Transactions on Information Forensics and Security*, 6(3 PART 2):873–882.

Coltuc, D. (2012). Low distortion transform for reversible watermarking. *IEEE Transactions on Image Processing*, 21(1):412–417.

Cox, I., Miller, M., Bloom, J., Fridrich, J., and Kalker, T. (2008). *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition.

Dragoi, I.-C. and Coltuc, D. (2014). Local-prediction-based difference expansion reversible watermarking. *IEEE Transactions on image processing*, 23(4):1779–1790.

Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

Eskicioglu, A. M. and Fisher, P. S. (1995). Image quality measures and their performance. *IEEE Transactions on Communications*, 43(12):2959–2965.

Ferguson, N., Schneier, B., and Kohno, T. (2011). *Cryptography engineering: design principles and practical applications*. John Wiley & Sons.

Fridrich, J., Goljan, M., and Du, R. (2001). Invertible authentication. In *Security and Watermarking of Multimedia contents III*, volume 4314, pages 197–209. International Society for Optics and Photonics.

Gao, G., Wan, X., Yao, S., Cui, Z., Zhou, C., and Sun, X. (2017). Reversible data hiding with contrast enhancement and tamper localization for medical images. *Information Sciences*, 385:250–265.

Gao, X., An, L., Li, X., and Tao, D. (2009). Reversibility improved lossless data hiding. *Signal Processing*, 89(10):2053–2065.

Gao, X., An, L., Yuan, Y., Tao, D., and Li, X. (2011). Lossless data embedding using generalized statistical quantity histogram. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(8):1061–1070.

Gen, M. and Cheng, R. (2000). *Genetic algorithms and engineering optimization*, volume 7. John Wiley & Sons.

He, W., Cai, J., Zhou, K., and Xiong, G. (2017a). Efficient pvo-based reversible data hiding using multistage blocking and prediction accuracy matrix. *Journal of Visual Communication and Image Representation*, 46:58–69.

He, W., Zhou, K., Cai, J., Wang, L., and Xiong, G. (2017b). Reversible data hiding using multi-pass pixel value ordering and prediction-error expansion. *Journal of Visual Communication and Image Representation*, 49:351–360.

Honsinger, C. W., Jones, P. W., Rabbani, M., and Stoffel, J. C. (2001). Lossless recovery of an original image containing embedded data. US Patent 6,278,791.

Hore, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369.

Huang, L.-C., Tseng, L.-Y., and Hwang, M.-S. (2013). A reversible data hiding method by histogram shifting in high quality medical images. *Journal of Systems and Software*, 86(3):716–727.

Hyvärinen, A., Hurri, J., and Hoyer, P. O. (2009). *Natural image statistics: a probabilistic approach to early computational vision.* Springer.

Kamran, Khan, A., and Malik, S. A. (2014). A high capacity reversible watermarking approach for authenticating images: Exploiting down-sampling, histogram processing, and block selection. *Information Sciences*, 256:162–183.

Khan, A., Malik, S. A., et al. (2014a). A high capacity reversible watermarking approach for authenticating images: exploiting down-sampling, histogram processing, and block selection. *Information Sciences*, 256:162–183.

Khan, A., Siddiqa, A., Munib, S., and Malik, S. A. (2014b). A recent survey of reversible watermarking techniques. *Information Sciences*, 279:251–272.

Kim, K. S., Lee, M. J., Lee, H. Y., and Lee, H. K. (2009). Reversible data hiding exploiting spatial correlation between sub-sampled images. *Pattern Recognition*, 42(11):3083–3096.

Ko, L.-T., Chen, J.-E., Shieh, Y.-S., Scalia, M., and Sung, T.-Y. (2012). A novel fractional-discrete-cosine-transform-based reversible watermarking for healthcare information management systems. *Mathematical Problems in Engineering*, 2012.

Li, X., Li, J., Li, B., and Yang, B. (2013). High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. *Signal Processing*, 93(1):198–205.

Li, Y.-C., Yeh, C.-M., and Chang, C.-C. (2010). Data hiding based on the similarity between neighboring pixels with reversibility. *Digital Signal Processing*, 20(4):1116–1128.

Lin, C.-C., Tai, W.-L., and Chang, C.-C. (2008). Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognition*, 41(12):3582–3591.

Luo, Y., Peng, F., Li, X., and Yang, B. (2011). Reversible image water marking based on prediction-error expansion and compensation. *Proceedings - IEEE International Conference on Multimedia and Expo*, 1:0–5.

Memon, N. A., Khan, A., Gilani, S., and Ahmad, M. (2011). Reversible watermarking method based on adaptive thresholding and companding technique. *International Journal of Computer Mathematics*, 88(8):1573–1594.

Mohammed, R. T. and Khoo, B. E. (2013). Robust reversible watermarking scheme based on wavelet-like transform. In *Signal and Image Processing Applications (ICSIPA), 2013 IEEE International Conference on*, pages 354–359. IEEE.

Naheed, T., Usman, I., Khan, T. M., Dar, A. H., and Shafique, M. F. (2014). Intelligent reversible watermarking technique in medical images using GA and PSO. *Optik - International Journal for Light and Electron Optics*, 125(11):2515–2525.

Nguyen, T.-S., Chang, C.-C., and Yang, X.-Q. (2016). A reversible image authentication scheme based on fragile watermarking in discrete wavelet transform domain. *AEU-International Journal of Electronics and Communications*, 70(8):1055–1061.

Ni, Z., Shi, Y. Q., Ansari, N., Su, W., Sun, Q., and Lin, X. (2008). Robust lossless image data hiding designed for semi-fragile image authentication. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(4):497–509.

Ni, Z. N. Z., Shi, Y.-Q. S. Y.-Q., Ansari, N., and Su, W. S. W. (2006). Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(3):354–362.

Ou, B., Li, X., and Wang, J. (2016). High-fidelity reversible data hiding based on pixel-value-ordering and pairwise prediction-error expansion. *Journal of Visual Communication and Image Representation*, 39:12–23.

Ou, B., Li, X., Wang, J., and Peng, F. (2017). High-fidelity reversible data hiding based on geodesic path and pairwise prediction-error expansion. *Neurocomputing*, 226:23–34.

Ou, B., Zhao, Y., Ni, R., and Cao, G. (2010). A high payload histogram-based reversible wartermarking using linear prediction. *Proceedings - 2010 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2010*, pages 446–449.

Parah, S. A., Ahad, F., Sheikh, J. A., and Bhat, G. M. (2017). Hiding clinical information in medical images: A new high capacity and reversible data hiding technique. *Journal of biomedical informatics*, 66:214–230.

Peng, F., Li, X., and Yang, B. (2014). Improved pvo-based reversible data hiding. *Digital Signal Processing*, 25:255–265.

Qu, X. and Kim, H. J. (2015). Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding. *Signal Processing*, 111:249–260.

Sachnev, V., Kim, H. J., Nam, J., Suresh, S., and Shi, Y. Q. (2009). Reversible watermarking algorithm using sorting and prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(7):989–999.

Saini, L. K. and Shrivastava, V. (2014). A survey of digital watermarking techniques and its applications. *arXiv preprint arXiv:1407.4735*.

Shi, Y.-Q., Li, X., Zhang, X., Wu, H.-T., and Ma, B. (2016). Reversible data hiding: advances in the past two decades. *IEEE Access*, 4:3210–3237.

Shih, F. Y. and Zhong, X. (2016). High-capacity multiple regions of interest watermarking for medical images. *Information Sciences*, 367:648–659.

Siddiqa, A. and Khan, A. (2015). High capacity reversible image watermarking using error expansion and context-dependent embedding. *Electronics Letters*, 51(13):985–987.

Stinson, D. R. (2005). *Cryptography: theory and practice*. CRC press.

Thodi, D. M. and Rodríguez, J. J. (2004). Prediction-error based reversible watermarking. *Proceedings - International Conference on Image Processing, ICIP*, 3(1):1549–1552.

Thodi, D. M. and Rodríguez, J. J. (2007). Expansion embedding techniques for reversible watermarking. *IEEE Transactions on Image Processing*, 16(3):721–730.

Tian, J. (2003). Reversible Data Embedding Using a Difference Expansion. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(8):890–896.

Tsai, P., Hu, Y.-C., and Yeh, H.-L. (2009). Reversible image hiding scheme using predictive coding and histogram shifting. *Signal processing*, 89(6):1129–1143.

Tudoroiu, A., Caciula, I., and Coltuc, D. (2011). Block map implementation of difference expansion reversible watermarking. *ISSCS 2011 - International Symposium on Signals, Circuits and Systems, Proceedings*, pages 321–324.

Usman, I. and Khan, A. (2010). Bch coding and intelligent watermark embedding: Employing both frequency and strength selection. *Applied Soft Computing*, 10(1):332–343.

Vleeschouwer, C., Delaigle, J. F., and Macq, B. (2003). Circular interpretation of bijective transformations in lossless watermarking for media asset management. *IEEE Transactions on Multimedia*, 5(1):97–105.

Vleeschouwer, C. D., Delaigle, J., and Macq, B. (2001). Circular interpretation of histogram for reversible watermarking. *2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564)*, pages 345–350.

Wang, X., Ding, J., and Pei, Q. (2015). A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition. *Information sciences*, 310:16–35.

Weng, S. and Pan, J.-S. (2016). Integer transform based reversible watermarking incorporating block selection. *Journal of Visual Communication and Image Representation*, 35:25–35.

Wu, H.-T., Huang, J., and Shi, Y.-Q. (2015). A reversible data hiding method with contrast enhancement for medical images. *Journal of Visual Communication and Image Representation*, 31:146–153.

Xuan, G., Shi, Y. Q., Yang, C., Zheng, Y., Zou, D., and Chai, P. (2005). Lossless data hiding using integer wavelet transform and threshold embedding technique. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1520–1523. IEEE.

Xuan, G., Yang, C., Zhen, Y., Shi, Y. Q., and Ni, Z. (2004). Reversible data hiding using integer wavelet transform and companding technique. In *International Workshop on Digital Watermarking*, pages 115–124. Springer.

Xuan, G., Yao, Q., Yang, C., Gao, J., Chai, P., Shi, Y. Q., and Ni, Z. (2006). Lossless data hiding using histogram shifting method based on integer wavelets. In *International Workshop on Digital Watermarking*, pages 323–332. Springer.

Xuan, G., Zhu, J., Chen, J., Shi, Y. Q., Ni, Z., and Su, W. (2002). Distortionless data hiding based on integer wavelet transform. *Electronics Letters*, 38(25):1646–1648.

Yang, B., Schmucker, M., Funk, W., Busch, C., and Sun, S. (2004). Integer dct-based reversible watermarking for images using companding technique. In *Security, Steganography, and Watermarking of Multimedia Contents VI*, volume 5306, pages 405–416. International Society for Optics and Photonics.