

Watermarking Digital Images in the Frequency Domain: Performance and Attack Issues

Maria Chroni Angelos Fylakis Stavros D. Nikolopoulos

Department of Computer Science & Engineering
University of Ioannina
GR-45110 Ioannina, Greece
{mchroni,afylakis,stavros}@cs.uoi.gr

Abstract. In this work we propose an efficient model for watermarking images that are intended for uploading on the web under intellectual property protection. Headed to this direction, we recently suggested a way in which an integer number w which being transformed into a self-inverting permutation, can be represented in a two dimensional (2D) object and thus, since images are 2D structures, we propose a watermarking algorithm that embeds marks on them using the 2D representation of w in the frequency domain. In particular, we propose a watermarking technique that uses the 2D representation of self-inverting permutations and utilizes marking at specific areas thanks to partial modifications of the image's Discrete Fourier Transform (DFT). Those modifications are made on the magnitude of specific frequency bands and they are the least possible additive information ensuring robustness and imperceptiveness. We have experimentally evaluated our algorithms using various images of different characteristics under JPEG compression, Gaussian noise addition, and geometric transformations. The experimental results show an improvement in comparison to the previously obtained results and they also depict the validity of our proposed codec algorithms.

Keywords: Watermarking techniques, Image watermarking algorithms; Self-inverting permutations; 2D representations of permutations; Encoding; Decoding; Frequency domain; Experimental evaluation.

1 Introduction

Internet technology, in modern communities, becomes day by day an indispensable tool for everyday life since most people use it on a regular basis and do many daily activities online [1]. This frequent use of the internet means that measures taken for internet security are indispensable since the web is not risk-free [2, 3]. One of those risks is the fact that the web is an environment where intellectual property is under threat since a huge amount of public personal data is continuously transferred, and thus such data may end up on a user who falsely claims ownership.

It is without any doubt that images, apart from text, are the most frequent type of data that can be found on the internet. As digital images are a characteristic kind of intellectual material, people hesitate to upload and transfer them

via the internet because of the ease of intercepting, copying and redistributing in their exact original form [4]. Encryption is not the problem's solution in most cases, as most people that upload images in a website want them to be visible by everyone, but safe and theft protected as well. Watermarks are a solution to this problem, since thanks to them someone can claim the property of an image if he previously inserted one in it.

Watermarking. Digital watermarking (or, hereafter, watermarking) is a technique for protecting the intellectual property of a digital object; the idea is simple: a unique marker, which is called *watermark*, is embedded into a digital object which may be used to verify its authenticity or the identity of its owners [6, 7]. More precisely, watermarking can be described as the problem of embedding a watermark w into an object \mathcal{O} and, thus, producing a new object I_w , such that w can be reliably located and extracted from \mathcal{O}_w even after \mathcal{O}_w has been subjected to transformations [7]; for example, compression, scaling or rotation in case where the object is an image.

In the image watermarking process the digital information, i.e., the watermark, is hidden in image data. The watermark is embedded into image's data through the introduction of errors not detectable by human perception [8]; note that, if the image is copied or transferred through the internet then the watermark is also carried with the copy into the image's new location.

Motivation. Intellectual property protection is one of the greatest concerns of internet users today. Digital images are considered a representative part of such properties so we consider important, the development of methods that deter malicious users from claiming others' ownership, motivating internet users to feel more safe to publish their work online.

Image Watermarking, is a technique that serves the purpose of image intellectual property protection ideally as in contrast with other techniques it allows images to be available to third internet users but simultaneously carry an "identity" that is actually the proof of ownership with them. This way image watermarking achieves its target of deterring copy and usage without permission of the owner. What is more by saying watermarking we don't necessarily mean that we put a logo or a sign on the image as research is also done towards watermarks that are both invisible and robust.

Our work suggests a method of embedding a numerical watermark into the image's structure in an invisible and robust way to specific transformations, such as JPEG compression, Gaussian noise addition, and geometric transformation.

Contribution. In this work we present an efficient and easily implemented technique for watermarking images that we are interested in uploading in the web and making them public online; this way web users are enabled to claim the ownership of their images.

What is important for our idea is the fact that it suggests a way in which an integer number can be represented with a two dimensional representation (or, for short, 2D representation). Thus, since images are two dimensional ob-

jects that representation can be efficiently marked on them resulting the watermarked images. In a similar way, such a 2D representation can be extracted for a watermarked image and converted back to the integer w .

Having designed an efficient method for encoding integers as self-inverting permutations, we propose an efficient algorithm for encoding a self-inverting permutation π^* into an image I by first mapping the elements of π^* into an $n^* \times n^*$ matrix A^* and then using the information stored in A^* to mark specific areas of image I in the frequency domain resulting the watermarked image I_w . We also propose an efficient algorithm for extracting the embedded self-inverting permutation π^* from the watermarked image I_w by locating the positions of the marks in I_w ; it enables us to reconstruct the 2D representation of the self-inverting permutation π^* .

It is worth noting that although digital watermarking has made considerable progress and became a popular technique for copyright protection of multimedia information [8], our work proposes something new. We first point out that our watermarking method incorporates such properties which allow us to successfully extract the watermark w from the image I_w even if the input image has been compressed with a lossy method, scaled and/or rotated. In addition, our embedding method can transform a watermark from a numerical form into a two dimensional (2D) representation and, since images are 2D structures, it can efficiently embed the 2D representation of the watermark by marking the high frequency bands of specific areas of an image. The key idea behind our extracting method is that it does not actually extract the embedded information instead it locates the marked areas reconstructing the watermark's numerical value.

We have evaluated the embedding and extracting algorithms by testing them on various and different in characteristics images that were initially in JPEG format and we had positive results as the watermark was successfully extracted even if the image was converted back into JPEG format with various JPEG compression ratios. We had also positive results on Gaussian noise addition and geometric transformation attacks. All the algorithms have been developed and tested in MATLAB environment [9].

2 Theoretical Framework

In this section we first describe discrete structures, namely, permutations and self-inverting permutations, and briefly discuss a codec system which encodes an integer number w into a self-inverting permutation π . Then, we present a transformation of a watermark from a numerical form to a 2D form (i.e., 2D representation) through the exploitation of self-inverting permutation properties.

2.1 Self-inverting Permutations

Permutations may be represented in many ways [10]. The most straightforward is simply a rearrangement of the elements of the set $N_n = \{1, 2, \dots, n\}$; in this way we think of the permutation $\pi = (5, 6, 9, 8, 1, 2, 7, 4, 3)$ as a rearrangement

of the elements of the set N_9 such that “1 goes to 5”, “2 goes to 6”, and so on [10, 11]. Hereafter, we shall say that π is a permutation over the set N_9 .

Definition 2.1.1. Let $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ be a permutation over the set N_n , $n > 1$. The inverse of the permutation π is the permutation $q = (q_1, q_2, \dots, q_n)$ with $q_{\pi_i} = \pi_{q_i} = i$. A *self-inverting permutation* (or, for short, SiP) is a permutation that is its own inverse: $\pi_{\pi_i} = i$.

By definition, a permutation is a SiP (self-inverting permutation) if and only if all its cycles are of length 1 or 2; for example, the permutation $\pi = (5, 6, 9, 8, 1, 2, 7, 4, 3)$ is a SiP with cycles: (1, 5), (2, 6), (3, 9), (4, 8), and (7).

2.2 Encoding Numbers as SiPs

There are several systems that correspond integer numbers into permutations or self-inverting permutation [10]. Recently, we have proposed algorithms for such a system which efficiently encodes an integer w into a self-inverting permutations π and efficiently decodes it. The algorithms of our codec system run in $O(n)$ time, where n is the length of the binary representation of the integer w . The key-idea behind our algorithms is mainly based on mathematical objects, namely, bitonic permutations [12].

2.3 2D and 2DM Representations

In the 2D representation, the elements of the permutation $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ are mapped in specific cells of an $n \times n$ matrix A as follows:

- number $\pi_i \longrightarrow$ entry $A(\pi_i^{-1}, \pi_i)$

or, equivalently, the cell at row i and column π_i is labeled by the number π_i , for each $i = 1, 2, \dots, n$. Figure 1(a) shows the 2D representation of the self-inverting permutation $\pi = (6, 3, 2, 4, 5, 1)$.

Based on the previously defined 2D representation of a permutation π , we next propose a two-dimensional marked representation (2DM representation) of π , which is an efficient tool for watermarking images. In our 2DM representation, a permutation π over the set N_n is represented by an $n \times n$ matrix A^* as follows:

- the cell at row i and column π_i of matrix A^* is marked by a specific symbol (in our implementation we use the asterisk “*”), for each $i = 1, 2, \dots, n$.

Figure 1(b) shows the 2DM representation of the permutation π . It is easy to see that, since the 2DM representation of π is constructed from the corresponding 2D representation, there is one symbol in each row and in each column of the matrix A^* . It is also easy to see that we can extract π from A^* in linear time (i.e., linear in the size of matrix A^*); we call `Extract_pi_from_2DM` the extraction algorithm.

	1	2	3	4	5	6
1						1
2			2			
3		3				
4				4		
5					5	
6	6					

(a)

	1	2	3	4	5	6
1						*
2			*			
3		*				
4				*		
5					*	
6	*					

(b)

Fig. 1. The 2D and 2DM representations of the self-inverting permutation $\pi = (6, 3, 2, 4, 5, 1)$.

Remark 2.3.1. Since the permutation π is a self-inverting permutation, its 2D matrix A has the following property: $A(i, j) = j$ if $\pi_i = j$ and $A(i, j) = 0$ otherwise, $1 \leq i, j \leq n$. Thus, the corresponding matrix A^* is symmetric.

Hereafter, we shall denote by π^* a SiP and by n^* the number of elements of π^* .

2.4 The Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the frequency domain, while the input image is the spatial domain equivalent. In the image's fourier representation, each point represents a particular frequency contained in the image's spatial domain.

Typically, in our method, we are interested in the magnitudes of DFT coefficients. The magnitude $|F(u, v)|$ of the Fourier transform at a point is how much frequency content there is [14].

3 The Frequency Domain Approach

Having described an efficient method for encoding integers as self-inverting permutations using the 2DM representation of self-inverting permutations, we next describe codec algorithms that efficiently encode and decode a watermark into the image's frequency domain [15, 16, 14].

3.1 Embed Watermark into Image

We next describe the embedding algorithm of our proposed technique which encodes a self-inverting permutation (SiP) π^* into a digital image I . Recall that, the permutation π^* is obtained over the set N_{n^*} , where $n^* = 2n + 1$ and n is the length of the binary representation of an integer w which actually is the image's watermark [12].

The watermark w , or equivalently the self-inverting permutation π^* , is inserted in the frequency domain of specific areas of the image I . More precisely, we mark the DFT's magnitude of an image's area using two ellipsoidal annuli, denoted hereafter as “Red” and “Blue” (see, Figure 2). The ellipsoidal annuli are specified by the following parameters:

- P_r , the width of the “Red” ellipsoidal annulus,
- P_b , the width of the “Blue” ellipsoidal annulus,
- R_1 and R_2 , the radiuses of the “Red” ellipsoidal annulus on y -axis and x -axis, respectively.

The algorithm takes as input a SiP π^* and an image I , and returns the watermarked image I_w ; it consists of the following steps.

Algorithm Embed_SiP-to-Image

Input: the watermark $\pi^* \equiv w$ and the host image I ;

Output: the watermarked image I_w ;

Step 1: Compute first the 2DM representation of the permutation π^* , i.e., construct an array A^* of size $n^* \times n^*$ such that the entry $A^*(i, \pi_i^*)$ contains the symbol “*”, $1 \leq i \leq n^*$.

Step 2: Next, compute the size of the input image I , say, $N \times M$, and cover the image I with an imaginary grid C with $n^* \times n^*$ grid-cells C_{ij} of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$, $1 \leq i, j \leq n^*$.

Step 3: For each grid-cell C_{ij} , compute the Discrete Fourier Transform (DFT) using the Fast Fourier Transform (FFT) algorithm, resulting in a $n^* \times n^*$ grid of DFT cells F_{ij} , $1 \leq i, j \leq n^*$.

Step 4: For each DFT cell F_{ij} , compute its magnitude M_{ij} and phase P_{ij} matrices which are both of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$, $1 \leq i, j \leq n^*$.

Step 5: Then, the algorithm takes each of the $n^* \times n^*$ magnitude matrices M_{ij} , $1 \leq i, j \leq n^*$, and places two imaginary ellipsoidal annuli, denoted as “Red” and “Blue”, in the matrix M_{ij} (see, Figure 2). In our implementation,

- the “Red” is the outer ellipsoidal annulus while the “Blue” is the inner one. Both are concentric at the center of the M_{ij} magnitude matrix and have widths P_r and P_b , respectively;
- the radiuses of the “Red” ellipsoidal annulus are R_1 (y -axis) and R_2 (x -axis), while the “Blue” ellipsoidal annulus radiuses are computed in accordance to the “Red” ellipsoidal annulus and have values $(R_1 - P_r)$ and $(R_2 - P_r)$, respectively;
- the inner perimeter of the “Red” ellipsoidal annulus coincides to the outer perimeter of the “Blue” ellipsoidal annulus;
- the values of the widths of the two ellipsoidal annuli are $P_r = 2$ and $P_b = 2$, while the values of their radiuses are $R_1 = \lfloor \frac{N}{2n^*} \rfloor$ and $R_2 = \lfloor \frac{M}{2n^*} \rfloor$.

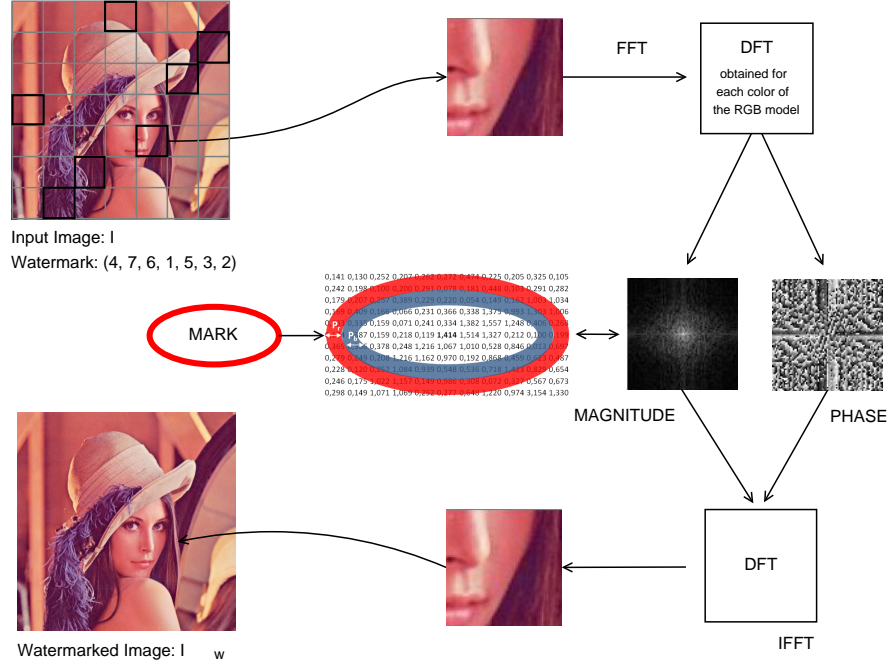


Fig. 2. The embedding process.

The areas covered by the “Red” and the “Blue” ellipsoidal annuli determine two groups of magnitude values on M_{ij} (see, Figure 2).

Step 6: For each magnitude matrix M_{ij} , $1 \leq i, j \leq n^*$, compute the average of the values that are in the areas covered by the “Red” and the “Blue” ellipsoidal annuli; let $AvgR_{ij}$ be the average of the magnitude values belonging to the “Red” ellipsoidal annulus and $AvgB_{ij}$ be the one of the “Blue” ellipsoidal annulus.

Step 7: For each magnitude matrix M_{ij} , $1 \leq i, j \leq n^*$, compute first the variable D_{ij} as follows:

- $D_{ij} = |AvgB_{ij} - AvgR_{ij}|$, if $AvgB_{ij} \leq AvgR_{ij}$
- $D_{ij} = 0$, otherwise.

Then, for each row i of the matrix M_{ij} , $1 \leq i, j \leq n^*$, compute the maximum value of the variables $D_{i1}, D_{i2}, \dots, D_{in^*}$ in row i ; let $MaxD_i$ be the max value.

Step 8: For each cell (i, j) of the 2DM representation matrix A^* of the permutation π^* such that $A_{ij}^* = “*”$ (i.e., marked cell), mark the corresponding

grid-cell C_{ij} , $1 \leq i, j \leq n^*$; the marking is performed by increasing all the values in magnitude matrix M_{ij} covered by the “Red” ellipsoidal annulus by the value

$$AvgB_{ij} - AvgR_{ij} + MaxD_i + c, \quad (1)$$

where $c = c_{opt}$. The additive value of c_{opt} is calculated by the function f (see, Subsection 3.3) which returns the minimum possible value of c that enables successful extracting.

Step 9: Reconstruct the DFT of the corresponding modified magnitude matrices M_{ij} , using the trigonometric form formula [14], and then perform the Inverse Fast Fourier Transform (IFFT) for each marked cell C_{ij} , $1 \leq i, j \leq n^*$, in order to obtain the image I_w .

Step 10: Return the watermarked image I_w .

In Figure 2 we demonstrate the main operations performed by our embedding algorithm. In particular, we show the marking process of the grid-cell C_{44} of the Lena image; in this example, we embed in the Lena image the watermark number w which corresponds to SiP (6, 3, 2, 4, 5, 1).

3.2 Extract Watermark from Image

In this section we describe the decoding algorithm of our proposed technique. The algorithm extracts a self-inverting permutation (SiP) π^* from a watermarked digital image I_w , which can be later represented as an integer w .

The self-inverting permutation π^* is obtained from the frequency domain of specific areas of the watermarked image I_w . More precisely, using the same two “Red” and “Blue” ellipsoidal annuli, we detect certain areas of the watermarked image I_w that are marked by our embedding algorithm and these marked areas enable us to obtain the 2D representation of the permutation π^* . The extracting algorithm works as follows:

Algorithm Extract_SiP-from-Image

Input: the watermarked image I_w marked with π^* ;

Output: the watermark $\pi^* = w$;

Step 1: Take the input watermarked image I_w and compute its $N \times M$ size. Then, cover I_w with the same imaginary grid C , as described in the embedding method, having $n^* \times n^*$ grid-cells C_{ij} of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$.

Step 2: Then, again for each grid-cell C_{ij} , $1 \leq i, j \leq n^*$, using the Fast Fourier Transform (FFT) get the Discrete Fourier Transform (DFT) resulting a $n^* \times n^*$ grid of DFT cells.

Step 3: For each DFT cell, compute its magnitude matrix M_{ij} and phase matrix P_{ij} which are both of size $\lfloor \frac{N}{n^*} \rfloor \times \lfloor \frac{M}{n^*} \rfloor$.

Step 4: For each magnitude matrix M_{ij} , place the same imaginary “Red” and “Blue” ellipsoidal annuli, as described in the embedding method, and compute as before the average values that coincide in the area covered by the “Red” and the “Blue” ellipsoidal annuli; let $AvgR_{ij}$ and $AvgB_{ij}$ be these values.

Step 5: For each row i of C_{ij} , $1 \leq i \leq n^*$, search for the j_{th} column where $AvgB_{ij} - AvgR_{ij}$ is minimized and set $\pi_i^* = j$, $1 \leq j \leq n^*$.

Step 6: Return the self-inverting permutation π^* .

Having presented the embedding and extracting algorithms, let us next comment on the function f which returns the additive value $c = c_{opt}$ (see, Step 8 of the embedding algorithm).

3.3 Function f

In our watermarking model, the embedding algorithm amplifies the marks in the “Red” ellipsoidal annulus by adding the output of the function f . What exactly f does is returning the optimal value that allows the extracting algorithm under the current requirements, such as JPEG compression, noise addition, to still be able to extract the watermark from the image.

The function f takes as an input the characteristics of the image and the parameters R_1 , R_2 , P_b , and P_r of our proposed watermark model (see, Step 5 of embedding algorithm and Figure 2), and returns the minimum possible c_{opt} that added as c to the values of the “Red” ellipsoidal annulus enables extracting (see, Step 8 of the embedding algorithm). More precisely, the function f initially takes the interval $[0, c_{max}]$, where c_{max} is a relatively great value such that if c_{max} is taken as c for marking the “Red” ellipsoidal annulus it allows extracting, and computes the c_{opt} in $[0, c_{max}]$.

Note that, c_{max} allows extracting but because of being great damages the quality of the image (see, Figure 3). We mentioned relatively great because it depends on the characteristics of each image. For a specific image it is useless to use a c_{max} greater than a specific value, we only need a value that definitely enables the extracting algorithm to successfully extract the watermark.

We next describe the computation of the value c_{opt} returned by f ; note that, the parameters P_b and P_r of our implementation are fixed with the values 2 and 2, respectively. The main steps of this computation are the following:

- (i) Check if the extracting algorithm for $c = 0$ validly obtains the watermark $\pi^* = w$ from the image I_w ; if yes, then the function f returns $c_{opt} = 0$;
- (ii) If not, that means, $c = 0$ doesn't allow extracting; then, the function f uses binary search on $[0, c_{max}]$ and computes the interval $[c_1, c_2]$ such that:
 - $c = c_1$ doesn't allow extracting,
 - $c = c_2$ do allow extracting, and
 - $|c_1 - c_2| < 0.2$;
- (iii) The function f returns $c_{opt} = c_2$;

As mentioned before, the function f returns the optimal value c_{opt} . Recall that, optimal means that it is the smallest possible value which enables extracting $\pi^* = w$ from the image I_w . It is important to be the smallest one as that minimizes the additive information to the image and, thus, assures minimum drop to the image quality.

4 Experimental Evaluation

In this section we present the experimental results of the proposed watermarking codec algorithms which we have implemented using the general-purpose mathematical software package Matlab (version 7.7.0) [9]. We tested our codec algorithms on various 24-bit digital color images of various sizes (from 200×130 up to 4600×3700) and quality characteristics. Many of the images in our image repository were taken from a web image gallery [17] and enriched by some other images different in characteristics.

In this work we used JPEG images due to their great importance on the web, since they are small in size, while storing full color information (24 bit/pixel), and can be easily and efficiently transmitted. Moreover, robustness to lossy compression is an important issue when dealing with image authentication. It should be observed that the design goal of lossy compression systems is opposed to that of watermark embedding systems. The Human Visual System (HVS) attempts to identify and discard perceptually insignificant information of the image, whereas the goal of the watermarking system is to embed the watermark information without altering the visual perception of the image [21].

In order to evaluate the quality of the watermarked image obtained from our watermarking method we used two objective image quality assessment metrics, namely the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index Metric (SSIM). Our aim was to prove that the watermarked image is closely related to the original (image fidelity [5]), because watermarking should not introduce visible distortions in the original image as that would reduce images' commercial value.

The PSNR metric is the ratio between the reference signal and the distortion signal, i.e., watermark, in an image given in decibels (dB). It is well known that, PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). The higher the PSNR value the closer the distorted image is to the original or the better the watermark conceals. It is a popular metric due to its simplicity, although it is well known that this distortion metric is not absolutely correlated with human vision. The SSIM image quality metric, developed by [18], is considered to be correlated with the quality perception of the HVS [19]. The highest value of SSIM is 1, and it is achieved when the original I and watermarked image I_w are identical.

4.1 Performance

Initially, we had to choose the appropriate values for the parameters of the quality function f . In our implementation we set both of the parameters P_r

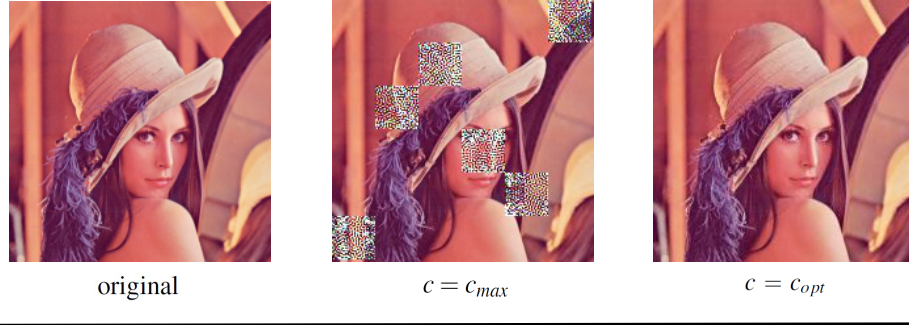


Fig. 3. The original image of Lena and its two watermarked images with $c = c_{max}$ and $c = c_{opt}$; the watermark corresponds to SiP (6,3,2,4,5,1).

and P_b equal to 2 (see, Section 3.3). Recall that, the value 2 is a relatively small value which allows us to modify a satisfactory number of pixels in order to embed the watermark and successfully extract it, without affecting images' quality. Note that, for great in size images, a smaller width reduces the strength of the watermark. There isn't a distance between the two ellipsoidal annuli as that enables the algorithm to apply a small additive information to the values of the "Red" annulus. The two ellipsoidal annuli are inscribed to the rectangle magnitude matrix, as we want to mark images' cells on the high frequency bands.

We mark the high frequencies by increasing their values using mainly the additive parameter $c = c_{opt}$ because alterations in the high frequencies are less detectable by human eye [20]. What is more, in high frequencies most images contain less information.

The quality function f returns the factor c , which has the minimum value c_{opt} that allows the extracting algorithm to successfully extract the watermark. In fact, this value c_{opt} (see, Formula 1) is the main additive information embedded into the image. Depending on the images and the amount of compression, we need to increase the watermark strength by increasing the factor c . The value of c increases as the quality factor of JPEG compression decreases. It is obvious that the embedding algorithm is image dependent. It is worth noting that, the c_{opt} values are small for images of relatively small size while these values increase as we move to images of greater size.

To demonstrate the differences on watermarked image quality, with respect to the values of the additive factor c , we watermarked the original image lena.jpg and we embedded a watermark with $c = c_{max}$ and $c = c_{opt}$, where $c_{max} \gg c_{opt}$ (see, Figure 3); in the watermarked image in the middle we used $c = c_{max}$ for illustrative purposes.








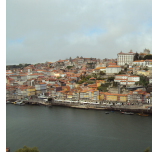


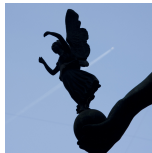
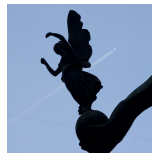
Name / Size	Original	Watermarked	Name / Size	Original	Watermarked
Baboon.jpg 200 x 200			lbook.jpg 200 x 200		
	$c_{opt} = 2.7$	PSNR = 41.5 SSIM = 0.977		$c_{opt} = 2.5$	PSNR = 42.2 SSIM = 0.963
Trattoria.jpg 500 x 500			City.jpg 500 x 500		
	$c_{opt} = 6.2$	PSNR = 46.3 SSIM = 0.984		$c_{opt} = 3.4$	PSNR = 51.5 SSIM = 0.993
Aquarium.jpg 1024 x 1024			Statue.jpg 1024 x 1024		
	$c_{opt} = 5.8$	PSNR = 56.2 SSIM = 0.997		$c_{opt} = 11.9$	PSNR = 50.3 SSIM = 0.977

Fig. 4. Some original images and their corresponding watermarked ones; for each image, its size and its c_{opt} , and PSNR and SSIM values are also shown, for $Q = 55$.

4.2 Attack Issues

In this section we present the experimental results of our watermarking method under several attacks. In fact, we test the robustness of our method after applying the following attacks:

- (A) JPEG Compression
- (B) Gaussian Noise
- (C) Geometric Transformations

Recall that, for the evaluation process we use the PSNR and SSIM metrics.

(A) JPEG Compression. The quality factor (or, for short, Q -factor) is a number that determines the degree of loss in the compression process when saving an image. In general, JPEG recommends a quality factor of 75–95 for visually indistinguishable quality difference, and a quality factor of 50–75 for merely acceptable quality. We compressed the images with Matlab using `imwrite` with different JPEG quality factors; we present results for $Q = 85$, $Q = 75$, $Q = 65$, and $Q = 55$.

Filename	PSNR				SSIM			
	Q = 85	Q = 75	Q = 65	Q = 55	Q = 85	Q = 75	Q = 65	Q = 55
Lena.jpg	54.04	50.10	46.82	44.86	0.997	0.993	0.986	0.981
Baboon.jpg	49.19	46.17	42.48	41.53	0.995	0.989	0.980	0.977
Trattoria.jpg	67.79	60.59	53.50	46.36	0.999	0.999	0.996	0.984
Aquarium.jpg	65.19	61.20	58.26	56.18	0.999	0.999	0.998	0.997
Ibook.jpg	51.47	47.78	44.76	42.21	0.994	0.987	0.976	0.963
City.jpg	57.20	52.86	48.63	51.54	0.998	0.995	0.987	0.993
Statue.jpg	63.58	58.40	54.90	50.30	0.998	0.995	0.990	0.977

Table 1. The PSNR and SSIM values of the original and watermarked images, for compression of qualities $Q = 85$, $Q = 75$, $Q = 65$, and $Q = 55$.

Our watermarked images have excellent PSNR and SSIM values. In Figure 4 we present six images of different sizes, along with their corresponding PSNR and SSIM values. Typical values for the PSNR in lossy image compression are between 40 and 70 dB, where higher is better. In our experiments, the PSNR values of 90% of the watermarked images were greater than 40 dB. The SSIM values are almost equal to 1, which means that the watermarked image is quite similar to the original one, which explains the method’s high fidelity.

In Table 1 we demonstrate the PSNR and SSIM values of some images that are used in this work. We observe that these values are decreasing on smaller quality factors. Also, as the additive value $c = c_{opt}$ increases for each quality factor, the quality decreases. Moreover, the additive value c that embeds robust marks for qualities $Q = 85$, $Q = 75$ and $Q = 65$, does not result in a significant image distortion as the tables suggest.

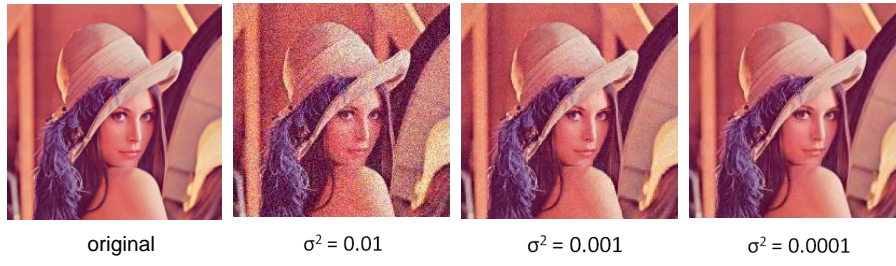


Fig. 5. The original image of Lena and its watermarked images with $\sigma^2 = 0.01$, $\sigma^2 = 0.001$ and $\sigma^2 = 0.0001$;

Filename	$\sigma^2 = 0.01$	$\sigma^2 = 0.001$	$\sigma^2 = 0.0001$
Lena.jpg	24.94	34.75	44.62
Baboon.jpg	24.89	34.79	44.65
Trattoria.jpg	25.04	34.83	44.73
Aquarium.jpg	25.97	35.27	44.81
Ibook.jpg	25.01	34.79	44.62
City.jpg	24.89	34.76	44.70
Statue.jpg	25.37	35.12	44.92

Table 2. The PSNR values of the original and watermarked images, for Gaussian noise with variance values $\sigma^2 = 0.01$, $\sigma^2 = 0.001$, and $\sigma^2 = 0.0001$.

(B) Gaussian Noise We test the robustness of our watermarking model by adding Gaussian noise in the images with $mean = 0$ and different variances σ^2 , that is, we use $\sigma^2 = 0.01$, $\sigma^2 = 0.001$ and $\sigma^2 = 0.0001$. Figure 5 illustrates the original image of Lena and the watermarked images with Gaussian noise of these three variance values. We have to mention that the watermark can be extracted successfully from the attacked image.

Table 2 presents the PSNR values of the original image and the watermarked image with Gaussian noise. As Table 2 and Figure 5 indicate, although Gaussian noise with $\sigma^2 = 0.01$ introduces significant perceptual distortion in images, watermark remains imperceptible.

(C) Geometric Transformations The robustness of the proposed model against geometric attacks was evaluated by applying common geometric attacks, which included rotation, cropping, and scaling.

C.1. Rotation Attacks

It is possible to detect whether the watermarked image has been subject to rotations, thanks to the following two properties of the 2DM representation of self-inverting permutations.

Due to the fact that the 2DM representation that has been used to mark the image is the result of a self-inverting permutation the sequence of the marked cells on the image is not random but there are the two properties that can be used to determine the angle of a watermarked image in respect with the original one and that has to do with the position of the marked cell in the main diagonal of the grid. The two properties are the following:

- The main diagonal of the $n^* \times n^*$ symmetric matrix A^* has always one and only one marked cell.
- The marked cell on the diagonal is always in the entry (i, i) of A^* where: $i = \lceil \frac{n^*}{2} \rceil + 1, \lceil \frac{n^*}{2} \rceil + 2, \dots, n^*$.

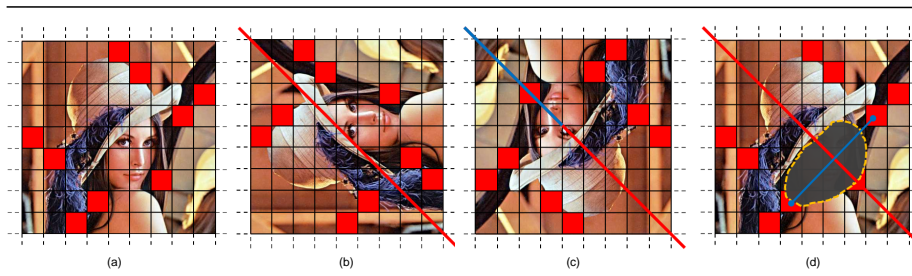


Fig. 6. (a) Watermarked image of Lena, (b) 90 degrees angled image, (c) 180 degrees angled image, and (d) cropped image.

In case the watermarked image has been subject to 90 degree rotation as demonstrated in Figure 6(b) you may notice that the main diagonal has not any cells marked which means that we are dealing with a non valid watermark as there should have been exactly one marked cell.

The second case is when the watermarked image has been subject to 180 degree rotation as demonstrated in Figure 6(c). In this case, beginning with the first property someone may notice that the main diagonal has one and only one marked cell meaning that the first property is satisfied confirming that the image has not been subject to 90 degree rotation.

The diagonal marked cell is situated in the grid's position (i, i) ; in our image $i = 3$ and thus $i < \lceil \frac{n^*}{2} \rceil$ since $n^* = 9$. It is against the second property meaning that the watermarked image has been subject to 180 degree rotation.

C.2. Cropping

Once again thanks to the fact that the 2DM representation a SiP has a symmetric property, i.e., the $n^* \times n^*$ matrix A^* is symmetric, our algorithm successfully extracts the watermark even marked parts of the watermarked image have been lost. This loss can be the result of cropping procedures to certain areas of the image. Recall that, this property is a consequence of the fact that at a self-inverting permutation, each element has its own inverse.

In Figure 6(d) the removed marked part of the image can be recovered as marks using a SiP are symmetric on A^* with respect to the main diagonal. As a result, because of the fact that $A^*(4, 8)$ is marked, $A^*(8, 4)$ is marked as well. Taking that into account we conclude that the lost marked cell is $A^*(8, 4)$ and then we correctly extract the embedded watermark.

C.3. Scaling

In the case where a watermarked image has underwent significant scaling then extracting a watermark may be unsuccessful. In our model if an image has been scaled by a known ratio then each cell of the imaginary grid has underwent exactly the same scaling meaning that the magnitude cell has now a different size as well. Due to this fact, the width of the annuli will be incorrect making

it impossible to calculate the appropriate difference between them. A solution to this would be to use different sized annuli in order to calculate the valid difference between them so that to spot marked areas.

The idea is simple, considering that we know the scaling ratio that the image has undergone we apply the same ratio calculating the new width for the two annuli. So if for example we used $P_b = 2$ and $P_r = 2$ for the width of the “Blue” and the “Red” annulus respectively when we performed the embedding procedure and the image has undergone 50% scaling then in order to extract the embedded watermark from the image we have to use $P_b^* = 1$ and $P_r^* = 1$.

In order to calculate the difference between the same frequency bands, in the second case where the magnitude cell has 50% of the initial size, we use annuli that have 50% less width in comparison with the ones originally embedded.

5 Concluding Remarks

In this paper we propose a watermarking model for embedding invisible watermarks into digital images.

We experimentally tested our codec algorithms on color JPEG images with various and different characteristics. We obtained positive results as the watermarks were invisible, they didn't affect the images' quality and they were extractable despite the JPEG compression and Gaussian noise addition. It is worth noting that the proposed algorithms are robust against rotation or cropping attacks.

The study of our quality function f remains an interesting problem for further investigation; indeed, f could incorporate learning algorithms [22] so that to be able to return the c_{opt} accurately and in a very short computational time.

References

1. Garfinkel, S.: Web Security, Privacy and Commerce. O'Reilly 2nd ed. (2001)
2. Chun-Shien, L., Shih-Kun, H., Chwen-Jye, S., Hong-Yuan, M.L.: Cocktail watermarking for digital image protection. *IEEE Trans. on Multimedia* 4, 209–224 (2000)
3. Davis, J. C.: Intellectual property in cyberspace - what technological / legislative tools are necessary for building a sturdy global information infrastructure?. In: *IEEE Proc. Int'l Symposium on Technology and Society*, pp. 66–74 (1997)
4. O'Ruanaidh, J. J. K., Dowling, W. J., Boland, F. M.: Watermarking digital images for copyright protection. In: *Vision, Image and Signal Processing, IEEE Proceedings* 143, pp. 250–256 (1996)
5. Cox, I.J., Miller, M.L., Bloom, J. A., Fridrich, J., Kalker, T.: *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd ed. (2008)
6. Grover, D.: *The Protection of Computer Software - Its Technology and Applications*. Cambridge University Press, New York (1997)
7. Collberg, C., Nagra, J.: *Surreptitious Software*. Addison-Wesley (2010)
8. Cox, I., Kilian, J., Leighton, T., Shamoon, T.: A Secure, Robust Watermark for Multimedia. In: *Proc. 1st Int'l Workshop on Information Hiding, LNCS 1174*, pp. 317–333 (1996)

9. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image Processing using Matlab. Prentice-Hall (2003)
10. Sedgewick, R., Flajolet, P.: An Introduction to the Analysis of Algorithms. Addison-Wesley (1996)
11. Golumbic, M.C.: Design Patterns: Algorithmic Graph Theory and Perfect Graphs. Academic Press, Inc., New York (1980)
12. Chroni, M., Nikolopoulos, S.D.: Encoding Watermark Integers as Self-inverting Permutations. In: Int'l Conference on Computer Systems and Technologies (Comp-SysTech'10), ACM ICPS 471, pp. 125–130 (2010)
13. Chroni, M., Nikolopoulos, S.D.: An Efficient Graph Codec System for Software Watermarking. In: IEEE Proc. 36th Int'l Conference on Computers, Software, and Applications (STPSA'12), pp. 595–600 (2012)
14. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice-Hall, 3rd ed. (2007)
15. Solachidis, V., Pitas, I.: Circularly Symmetric Watermark Embedding in 2-D DFT Domain. In: IEEE Transactions on Image Processing 10, pp. 1741–1753 (2001)
16. Licks, V., Hordan, R.: On Digital Image Watermarking Eobust to Geometric Transformations. In: IEEE Proc. Int'l Conference on Image Processing 3, pp. 690–693 (2000)
17. Petitcolas, P.: Image Database for Watermarking, <http://www.petitcolas.net/fabien/watermarking/>
18. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image Quality Assessment: From Error Visibility to Structural Similarity. In: IEEE Transactions on Image Processing 13, pp. 600–612 (2004)
19. Hore, A., Ziou, D.: Image Quality Metrics: PSNR vs. SSIM. In: Proc. 20th International conference on pattern recognition, pp. 2366–2369 (2010)
20. Kaur, M., Jindal, S., Behal, S.: A Study of Digital Image Watermarking. Journal of Research in Engineering and Applied Sciences 2, 126–136 (2012)
21. Zain, J.M.: Strict Authentication Watermarking with JPEG Compression (SAW-JPEG) for Medical Images. European Journal of Scientific Research 2, 250–256 (2011)
22. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, 3rd ed. (2010)