Computing and Informatics, Vol. 29, 2010, 357–387

EMBEDDING STRENGTH CRITERIA FOR AWGN WATERMARK, ROBUST AGAINST EXPECTED DISTORTION

Vesna Vučković

Faculty of Mathematics
Studentski trg 16
11000 Belgrade, Serbia
e-mail: vesnav@matf.bg.ac.rs, vesnav@math.rs

Manuscript received 4 February 2008; revised 25 May 2009
Communicated by François Brémond

Abstract. In this paper we engage in AWGN watermark for grayscale image (the message is embedded by adding of white Gaussian noise matrix; detection is blind, correlation based). We search criteria for "the best" (minimal one which guaranties watermark detectability) embedding strength for watermark robust against expected attack. These criteria we find for AWGN watermarks, which are embedded in spatial or in transform domains; for one bit message or for a longer message; into whole image or into some of its coefficients. This paper peculiarity is that we do not propose new watermarking algorithm; for well known, robust algorithm we find the best embedding strength for robust watermark.

Keywords: Digital watermark, AWGN, embedding effectiveness, lossy compression, spread spectrum technique, watermark robustness

1 INTRODUCTION

During the last fifteen years, a lot of methods for grayscale images robust watermarking have been proposed. Classification of these methods is usually made according to the embedding domain used. So, embedding techniques have been proposed

- in spatial domain [1, 2, 3]
- in transform domain, first of all in DCT and block DCT [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15], Fourier [16, 17, 18] and wavelet [19, 20, 21, 22] domains.

Also, there are “hybrid methods”, where watermark is embedded in two different domains (for example, in spatial and transform domain [23].

Another classification is based on the necessity of original image availability in the detection phase:

- Algorithms with informed detection require original image presence in the detector – in algorithms with blind detection, detectors do not require original image.
- Algorithms with informed detection [5, 6, 7] are nowadays rarely used, and whenever it is possible (with respect to watermarking application), they are replaced with blind ones [1, 2, 13, 15, 22, 24, 25, 26]. In some applications, original image presence in detector is especially undesirable. Particularly, this applies to watermarking application of ownership proving in a court dispute. Informed detection here usually leads to algorithm invertibility, reverse engineering possibility, and famous Craver attack [32, 33, 34].

One important watermarking algorithms class is based on spread spectrum technique [3, 5, 6, 7, 16, 27, 28, 29, 30, 31] – embedding by spreading the information about each message bit across several image pixels, or across the whole image. Watermark embedding in such algorithms is usually performed by adding of some pseudorandom data vector to the image.

Among such algorithms, an interesting class is represented by those based on additive white Gaussian noise (AWGN) embedding, with blind detection based on correlation between image and embedded message. To these techniques, Cox et al. in [29] devote a lot of space (principally they concentrate on watermark embedding in spatial domain). In our paper we will use the term *AWGN watermarks* for such watermarks.

Important property of digital watermark, which is dedicated to copyright protection, is its *robustness*. The watermark needs to remain detectable after common image operations (lossy compression, contrast/brightness changing, cropping, rotation ...). In this text, we'll call these operations *watermark (removal) attacks*, irrespective if they were performed without or with the intention to remove the watermark from the image.

A lot of watermarking algorithms robust against these attacks are proposed. All the just mentioned algorithms are robust. AWGN watermarking algorithm is not an exception – it exhibits good robustness characteristics against many removal attacks.

Clearly, *not every individual watermark embedded by robust technique is robust:* a watermark, to be robust, needs to be embedded strongly enough. However, watermark which is embedded too strongly will damage image quality. It is important to find a real measure for embedding strength.

This is the theme of our paper: we do not propose new watermarking algorithm, but for well known, robust algorithm, we determine “the best” embedding strength – minimal, that ensures message detectability practically with 100 % probability after expected attack.

The procedure of finding attacks to which our image was exposed is beyond the scope of this paper. Here we assume that it is known which attack occurred and what kind of procedure should be done (if necessary) in this situation. *We only determine necessary embedding strength for watermarks to remain detectable after performed attacks.*

This paper arose from investigation how strongly the watermark needs to be embedded to be robust against expected lossy compression; precisely, how strongly we need to embed it into image which then will be exposed to known compression (e.g. DjVu or JPEG), in order the watermark stays detectable in a compressed file. That's why the most part of this text is dedicated to lossy compression attack. Only in Section 7 we deal with embedding, robust against other attacks.

Sections 2–4 are dedicated to the simplest AWGN algorithm – spatial domain embedding. In Section 2 we briefly outline AWGN watermark embedding algorithm in spatial domain, in the way it is described in [29]. In Section 3 we consider effective embedding in spatial domain. We discover "the best" embedding strength – mathematically argued minimum strength needed for certain message detection immediately after embedding (effective embedding). Section 4 is dedicated to determining of embedding strength which is needed for a message to survive the expected lossy compression.

In Sections 5 and 6 we consider the same algorithm within the transform domain. The content of Section 5 shows that there is practically not any difference between white Gaussian noise embedding algorithms in spatial and in transform domains. In Section 6 we give several examples of strength coefficients setting, while embedding in some image coefficients, in block DCT transform domain.

In Section 7 we give a short analysis of determining embedding strength for other attacks.

Section 8 concludes this paper.

Note: We will use the known facts about normal distribution. The reader may, in case of need, find them in a book on probability, for example [36].

2 AWGN WATERMARK ALGORITHM

In this section, we briefly outline the AWGN watermark algorithm, as it is described in [29] (spatial domain embedding).

Later, we will analyze other variants of this algorithm too (embedding in different domains and image coefficients). However, this will always be in essence the same algorithm: we add white Gaussian noise to image (or subimage) in some domain; we test the correlation in the same domain in which the watermark is embedded.

2.1 Embedding

We embed one message bit (binary one or zero) into the image in the following way:

$$c_{w1} = c_o + \alpha \cdot r_w, \qquad c_{w0} = c_o - \alpha \cdot r_w$$

where c_{w1} and c_{w0} are image vectors that arise from embedding of binary one or binary zero into the image, c_o is the original image vector before embedding,[1] α is the embedding strength coefficient ($\alpha > 0$) and r_w is the reference pattern.[2]

The *reference pattern* is a vector of the same dimension as the image, which contains (white Gaussian) noise. In this paper, the reference pattern is the pseudorandom vector chosen in accordance with $N(0, 1)$.

The original image, the message bit and the watermark key are used as inputs into the embedding program. The *watermark key* is a number, equal for the embedder and the detector, and it is used as the pseudorandom number generator seed for the reference pattern. Watermarked image is the embedder output.

2.2 Detection

The image and the watermark key (the same as in the embedder) are inputs to the detector. The detected watermark message is the output. As a measure of detection, we use linear correlation between the image and the reference pattern:

$$lc(c, r_w) = \frac{c \bullet r_w}{\|r_w\| \cdot \|r_w\|}$$

($\bullet$ denotes the dot product, and $\|\ \|$ denotes the vector norm).

The detector compares the computed linear correlation value with the threshold value τ, set in advance, and replies:

$$\begin{array}{ll} \text{Binary one is embedded if} & lc(c, r_w) \geq \tau \\ \text{Binary zero is embedded if} & lc(c, r_w) \leq -\tau \\ \text{Nothing is embedded if} & |lc(c, r_w)| < \tau \end{array}$$

2.3 Embedding of Longer Message

We embed into the image a k bits message by embedding repeatedly (k times) one bit message. This embedding we call *embedding message bits one over another.*

[1] We present the grayscale image c_o, dimensionality of $m \times n$ pixels, with one $m \times n$ matrix or, (in fact the same) with one $m \cdot n$ vector. In this paper, we will use the terms *image vector*, *image matrix* and *image* as synonyms (if this will not make confusion).

[2] More precisely, resultant values in matrices must be from permissible set of image pixels values. Therefore, it is more precise to say $c_{w1} = [c_o + \alpha \cdot r_w]_8$ and $c_{w0} = [c_o - \alpha \cdot r_w]_8$. The sign $[\ ]_8$ designates "squeezing" of the result vector coordinates into 8-bits values, i.e. into values from the set $\{0, 1, 2, \ldots, 255\}$.

An alternative solution is *embedding into subimages*: we divide the image into disjoint subimages, and then we embed one bit message into each subimage. In this text, the *subimage* term is understood as any subset of image pixels set. Subimages may, but need not, be composed of adjacent pixels.

Different variants of the above methods are also possible. For example, we may embed several message bits into every subimage.

2.4 Watermark Impact on Image Fidelity

Embedding strength coefficient magnitude directly determines the measure of the image fidelity loss. This impact may be expressed for example with *mean square error*:

$$mse(c_o, c_w) = \frac{1}{mn}\sum_{i=1}^{mn}(c_w(i) - c_o(i))^2 = \frac{1}{mn}\sum_{i=1}^{mn}\alpha^2 r_w(i)^2 = \frac{\alpha^2}{mn}\|r_w\|^2 = \alpha^2.$$

The theme of this paper is determining minimal value of coefficient α that ensures message detectability. Hence, if we wish effective embedding, α and mse are as they are – they cannot be smaller.

3 EMBEDDING EFFECTIVENESS

Watermark embedding is *effective* if immediately after embedding the detector reports the image as watermarked.

In subsection 3.1, we determine the minimum sufficient embedding strength coefficient (α) for effective embedding of one bit message. In subsection 3.2, we determine it for a longer message.

3.1 Effective Embedding of One Bit Message

Embedding strength coefficient α and detection threshold τ directly influence the embedding effectiveness.

If we embed the watermark strongly (with large coefficient α), we will be able to achieve the ideal of 100 % effective embedding. However, that may affect the image quality badly – the image changes caused by watermarking may become noticeable. Therefore, it is important to find the best measure for the coefficient α.

Clearly, if we reduce the detection threshold τ (set it to be close to, or maybe equal, to zero), embedding will be automatically more effective (the detector will report higher percentage of embedding cases for the image as watermarked). But, if the threshold τ is too low, the probability of a false positive error (the case when the detector replies that the image is watermarked if it is not) will increase.

It is very important to find the real measure – to set the threshold τ and the coefficient α, in such a way that false negative probabilities (non-effective embedding) and false positive ones are acceptably low. Also, that watermark must not be embedded too strongly to have the fidelity affected.

3.1.1 Deviation of $lc(c_o, r)$ from Zero – The σ_{lc} Parameter

When setting the α and τ parameters for the image c_o, linear correlation value $lc(c_o, r)$ plays an important role (r is an arbitrary reference pattern).

The σ_{lc} parameter – the standard deviation of a sample (linear correlations between the original image and reference patterns) is the basis for correct setting of parameters α and τ, for effective embedding.

Now we try to determine the dependence of value σ_{lc} on the image characteristics.

All reference pattern $r = (r(1), r(2) \dots r(mn)))$ coordinates are liable to $N(0, 1)$ distribution. The mathematical expectation for $\|r\|$ is $\sqrt{mn}$.

For our image $c = (c(1), c(2) \dots c(mn)))$ and reference pattern r, the linear correlation[3] is

$$lc(c,r) = \frac{c \bullet r}{\|r\| \cdot \|r\|} = \frac{\sum\limits_{i=1}^{mn} c(i)r(i)}{mn}.$$

Therefore, the $lc(c, r)$ value for fixed c and reference pattern r (with coordinates from $N(0, 1)$), also has values from normal distribution, $N(0, \sigma_{lc}^2)$, where the standard deviation, σ_{lc}, is

$$\sigma_{lc} = \frac{\sqrt{(c(1))^2 + (c(2))^2 + \ldots + (c(mn))^2}}{mn} = \frac{\sqrt{E(c)}}{mn}.$$

3.1.2 Setting of τ and α Parameters

In the interval $(-3\sigma_{lc}, 3\sigma_{lc})$ almost all linear correlation values[4] are between the image and reference patterns.

If we set the threshold $\tau = 3\sigma_{lc}$, the linear correlation between the original image and the reference pattern would be in the interval $(-\tau, \tau)$ almost with probability $p = 1$. In other words, it would be almost 100 % sure that the detector would not respond with a false positive error.

We have to consider the possibility of a false positive error when the message is quite short (as here, the message being one bit long). If the message is longer, the false positive error is not a great problem (more about this later – subsection 3.2.1), and τ could be set considerably smaller.

[3] The subject of this text, by its very nature, does not mention “exact”, but “approximate” arithmetic. Based on the value of σ_{lc} we set τ and α , as values that roughly satisfy the mentioned conditions. Nevertheless, in order to simplify, the sign “=” will often appear in formulas. We should be aware that “=” means rather “≈” here.

[4] This comes from the well known [68 – 95 – 99.7] rule for the normal distribution.

The threshold should not be larger than $3\sigma_{lc}$: this is not only needless, but also affects the fidelity (a stronger embedding would be needed, so that the detector might recognize the message).

In order to make the embedding effective, for arbitrary reference pattern r the following should be satisfied (for binary one or binary zero embedding):

$$\begin{array}{lll l} lc(c_o + \alpha \cdot r, r) > \tau & \text{or} & lc(c_o - \alpha \cdot r, r) < -\tau, & \text{i.e.} \\ lc(c_o, r) + \alpha > \tau & \text{or} & lc(c_o, r) - \alpha < -\tau\ , & \text{i.e.} \\ \alpha > \tau - lc(c_o, r) & \text{or} & \alpha > \tau + lc(c_o, r), & \end{array}$$

i.e., for each reference pattern

$$\alpha > \tau + |lc(c_o, r)|.$$

If we choose $\alpha > \tau + 3\sigma_{lc}$, then almost every reference pattern will be effectively embedded with strength α.

Clearly, α has to be as small as possible, in order that the image fidelity stays at a sufficient level.

Taking into account that here approximate arithmetic is used, we should not make a great mistake if we set $\tau = 3\sigma_{lc}$, $\alpha = \tau + 3\sigma_{lc} = 6\sigma_{lc}$.

Such embedding, in which α is set in advance, and the reference pattern is embedded with this strength (without consideration for the value of the linear correlation between the image and the reference pattern *which will be embedded*), is called a *fixed strength embedding.*

The drawback of the fixed strength embedding is that it is bigger than necessary. This is the reason to consider the linear correlation between the image and the pattern that is to be embedded. In this case, we talk about an algorithm with the *embedding strength adjustment.*

Let $lc(c_o, r) = l$ (r is the reference pattern to be embedded). If $l < \tau$ and we embed binary one, we will set $\alpha = \tau - l$. If $l > \tau$, we will set $\alpha = 0$ (not necessary to embed anything). So, if we embed binary one, then $\alpha = max(\tau - l, 0)$. If we embed binary zero, then $\alpha = max(\tau + l, 0)$.

In such a way, we achieve great saving in embedding strength. If $\tau = 3\sigma_{lc}$, it will be (in the case of embedding binary one) $\alpha = max(3\sigma_{lc} - l, 0)$ and (in the case of binary zero) $\alpha = max(3\sigma_{lc} + l, 0)$.

Since $E[l] = 0$ (l is random variable with normal distribution, and $E[l]$ is the mathematical expectation of variable l), then $E[\alpha] \approx 3\sigma_{lc}$. It is smaller than in the case of fixed strength embedding (where $E[\alpha] = \alpha = 6\sigma_{lc}$).

3.2 Effective Embedding of Longer Message

3.2.1 Threshold τ Setting for Longer Message

The threshold τ for a longer message may be smaller than $3\sigma_{lc}$ (the value recommended in case of one bit message). For example, for $\tau = \sigma_{lc}$, for roughly 68 % of possible reference patterns the linear correlation between the image and the reference pattern will be within the interval $(-\tau, \tau)$. So, the false positive may appear in every third case. If we insist that every message bit must be detected in order to confirm the presence of message, it will be almost impossible to detect the message if it is not embedded (for detection of non-existing message, it is necessary for all reference patterns to arise as a false positive, and this is almost impossible in a longer message). So, in the case of longer message, a smaller τ value is permitted.

3.2.2 Embedding of Message Bits One Over Another

We embed k bits into an image, $c_o = (c_o(1), c_o(2), \ldots, c_o(mn))$, in a way that for each bit we add to the image (or subtract from it) the corresponding reference pattern $r_j = (r_j(1), r_j(2), \ldots, r_j(mn))$ $(j = 1, 2, \ldots, k)$, multiplied by embedding strength α_j. The resultant image is $c_w = (c_w(1), c_w(2), \ldots, c_w(mn))$. We obtain each image pixel with

$$c_w(i) = c_o(i) + \sum_{j=1}^{k} (\pm\alpha_j \cdot r_j(i)) \qquad (i = 1, 2, \ldots, mn).$$

Embedding defined in this way is localized in space. Only corresponding coordinates of the original image and reference pattern influence the resultant image pixel value. The reference patterns coordinates take values from normal distribution $N(0, 1)$.

Therefore, their linear combination

$$R(i) = \sum_{j=1}^{k} (\pm\alpha_j \cdot r_j(i)) \qquad (i = 1, 2, \ldots, mn)$$

takes values from $N(0, \sigma^2)$, where $\sigma = \sqrt{\alpha_1^2 + \alpha_2^2 + \ldots, \alpha_k^2}$.

Also, the coordinates of vector $r_s = (r_s(1), r_s(2), \ldots, r_s(mn))$, where

$$r_s(i) = \frac{R(i)}{\sqrt{\alpha_1^2 + \alpha_2^2 + \ldots, \alpha_k^2}} \qquad (i = 1, 2, \ldots, mn)$$

are liable to $N(0, 1)$ distribution, and therefore, r_s is the reference pattern.

Thus, *embedding of k reference patterns r_j one over another, with strengths α_j $(j = 1, 2, \ldots, k)$, is equal with embedding one reference pattern, r_s, with strength*

$$\beta = \sqrt{\alpha_1^2 + \alpha_2^2 + \ldots + \alpha_k^2},$$

or

$$c_w = c_o + \beta \cdot r_s.$$

If all patterns are embedded with the same strength, $\alpha_1 = \alpha_2 = \ldots = \alpha_k = \alpha$, then the total embedding strength is

$$\beta = \sqrt{\alpha_1^2 + \alpha_2^2 + \ldots + \alpha_k^2} = \sqrt{k \cdot \alpha^2} = \sqrt{k} \cdot \alpha.$$

Example 1. If $\alpha = 2$, and we embed the message of $k = 25$ bits (one over another, with fixed strength embedding), it will be $\beta = \sqrt{k} \cdot \alpha = 10$ (embedding of 25 patterns with strength 2 is equivalent to embedding of one pattern with strength 10).

Example 2. If $k = 2$, $\alpha_1 = 3$, $\alpha_2 = 4$, then $r_s = (3 \cdot r_1 + 4 \cdot r_2)/5$.

Embedding effect for two patterns with strengths 3 and 4 is equivalent to embedding of one pattern, with strength $\beta = \sqrt{\alpha_1^2 + \alpha_2^2} = \sqrt{3^2 + 4^2} = 5$.

3.2.3 Embedding Into Subimages

We divide image c_o with dimensions mn pixels into k disjoint subimages $c_o^1, c_o^2, \ldots, c_o^k$, of dimensions $mn^1, mn^2, \ldots, mn^k$, respectively ($mn = \sum_{j=1}^{k} mn^j$). For each subimage, the coefficient σ_{lc}^j $(j = 1, 2, \ldots, k)$ is

$$\sigma_{lc}^j = \frac{\sqrt{E(c_o^j)}}{mn^j}.$$

If it is possible to divide an image into k parts of equal dimensions and energies, then all subimages will have the same value of the σ_{lc}^j parameter:

$$\sigma_{lc}^j = \frac{\sqrt{E(c_o^j)}}{mn^j} = \frac{\sqrt{E(c_o)/k}}{mn/k} = \sqrt{k} \cdot \frac{\sqrt{E(c_o)}}{mn} = \sqrt{k} \cdot \sigma_{lc}.$$

In this case, the overall strength for k bits message, when embedding into k subimages, is equal to the strength for one bit message, multiplied by $\sqrt{k}$. In this case embedding strength is equal as the message is embedded one bit over another.

3.2.4 Dependence of σ_{lc} Parameter on Image Dimension

The σ_{lc} parameter is lower for bigger images. So, the image c_k originated from k equal images, c, of energy $E(c)$, will have energy $E(c_k) = k \cdot E(c)$ and dimension $dim(c_k) = k \cdot dim(c)$. Its σ_{lc} parameter will be

$$\sigma_{lc}(c_k) = \frac{\sqrt{E(c_k)}}{dim(c_k)} = \frac{\sqrt{k \cdot E(c)}}{k \cdot \mathrm{dim}(c)} = \frac{\sqrt{E(c)}}{\sqrt{k} \cdot dim(c)} = \frac{1}{\sqrt{k}} \cdot \sigma_{lc}(c).$$

Therefore, in an image with larger dimensions, we have to embed our message with a lower strength. In the previous case, it will be $\alpha_k = \alpha/\sqrt{k}$ and $\tau_k = \tau/\sqrt{k}$.

If we take care of the image fidelity, it is obvious that in larger image a longer message may be embedded.

The σ_{lc}, α and τ parameters increase with image energy growth. If c_1 and c_2 are images with equal dimensions, and $E(c_1) > E(c_2)$, then in image c_2 we may embed a longer message. Thus, *a longer message may be written in darker than in brighter image.*

4 ROBUSTNESS AGAINST LOSSY COMPRESSION

The following text gives an estimate for the strength coefficient α, in order that the watermark survives the expected lossy compression.

In subsection 4.1 we consider the case of one bit message, and in subsection 4.2 – the case of a longer message. For the purpose of simplicity, we will only consider the case of embedding binary one; what is said here may be easily transferred to the case of binary zero.

4.1 Robustness of One Bit Message

4.1.1 Embedding Strength Parameter After Compression (α')

Now, we try to find which part of the watermark will be destroyed by lossy compression. In this way, we will be able to set the embedding strength coefficient so that it will be adequate to ensure robustness against the expected lossy compression.

In our experiment, first we embed several (k) reference patterns (one by one) into the image with strength α (the number α is the same for all of them). In a way, we get k watermarked images (with binary one embedded).

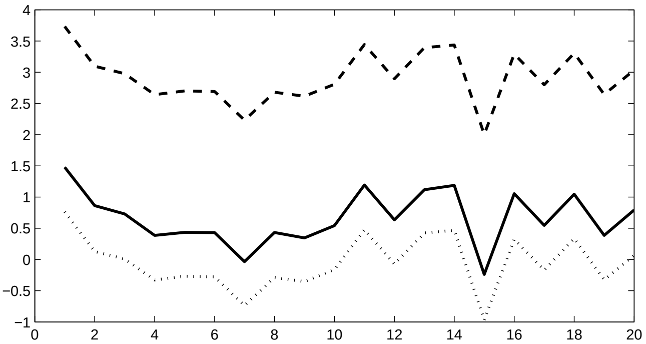
Then we subject each of these k images to expected lossy compression (the same for all k images). So, we get k compressed watermarked images.

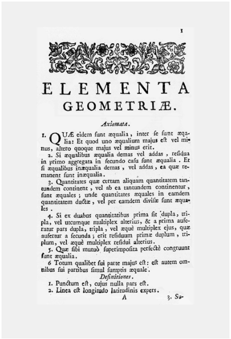
In the detector we have:[5]

- one original image c_o;
- k reference patterns $r(1), r(2), \ldots, r(k)$;
- k watermarked images $c_w(1), c_w(2), \ldots, c_w(k)$, for which $c_w(i) = c_o + \alpha \cdot r(i)$ $(i = 1, 2, \ldots, k)$ applies;
- k compressed watermarked images $c_{wn}(1), c_{wn}(2), \ldots, c_{wn}(k)$ ($c_{wn}(i)$ is the image $c_w(i)$ after lossy compression);
- k values of linear correlation $lc_o(1), lc_o(2), \ldots, lc_o(k)$, where $lc_o(i) = lc(c_o, r(i))$ (in this text they are called LCo array);
- k values of linear correlation $lc_w(1), lc_w(2), \ldots, lc_w(k)$, where $lc_w(i) = lc(c_w(i), r(i))$ (called LCw array);
- k values of linear correlation $lc_{wn}(1), lc_{wn}(2), \ldots, lc_{wn}(k)$, where $lc_{wn}(i) = lc(c_{wn}(i), r(i))$ (called $LCwn$ array).

Figure 1 represents (for $k = 20$) functions graphics for values of arrays of LCo, LCw and $LCwn$. Here we present results of the experiment for the first page of the book "Elementa geometriae". The compression is DjVu Photo (made by the

[5] Here the detector, besides recognizing the message, also compares watermark properties before and after lossy compression. This is why it uses noted images.





1

ELEMENTA
GEOMETRIÆ.

Axiomata.

1. QUÆ eidem ſunt æqualia, inter ſe ſunt æqualia: Et quod uno æqualium majus eſt vel minus, altero quoque majus vel minus erit.

2. Si æqualibus æqualia demas vel addas, reſidua in primo aggregata in ſecundo caſu ſunt æqualia. Et ſi æqualibus inæqualia demas, vel addas, ea quæ remanent ſunt inæqualia.

3. Quantitates quæ certam aliquam quantitatem tantundem continent, vel ab ea tantundem continentur, ſunt æquales; unde quantitates æquales in eamdem quantitatem ductæ, vel per eamdem diviſæ ſunt æquales.

4. Si ex duabus quantitatibus prima ſit dupla, tripla, vel utcumque multiplex alterius, & a prima auferatur pars dupla, tripla, vel æquè multiplex ejus, quæ aufertur a ſecunda; erit reſiduum primæ duplum, triplum, vel æquè multiplex reſidui alterius.

5. Quæ ſibi mutuò ſuperimpoſita perfectè congruunt ſunt æqualia.

6 Totum qualibet ſui parte majus eſt: eſt autem omnibus ſui partibus ſimul ſumptis æquale.

Definitiones.

1. Punctum eſt, cujus nulla pars eſt.

2. Linea eſt longitudo latitudinis expers.

A 3. *Su-*

Fig. 1. Linear correlation between the image (original, watermarked and compressed watermarked) and corresponding reference patterns (left), for the first page of the book "Elementa geometriae" (right). Embedding strength is $\alpha = 3$

program DjVu Solo 3.1 – LizardTech, Inc). Graphic abscissa presents arrays indexes, and ordinate – theirs values.[6]

- The bottom line presents the linear correlation values between the original image and reference patterns (LCo).
- The top line presents the linear correlation values between watermarked images and corresponding reference patterns (LCw).
- The middle line presents the linear correlation values between compressed watermarked images and corresponding reference patterns ($LCwn$).

In our example, embedding with $\alpha = 3$ increased the value of linear correlation roughly by 3. Intensive compression "ate up" the most part of watermark information. We may see from the previous graph that for each reference pattern $r(i)$, ($i = 1, 2, \ldots, k$) and linear correlation of (1) the original image c_o; (2) the watermarked image $c_w(i)$ (binary one embedded); and (3) the compressed watermarked image $c_{wn}(i)$, with this reference pattern ($LCo(i)$, $LCw(i)$, $LCwn(i)$) is

- $LCw(i) - LCo(i) \approx \alpha$ (embedding of binary one raises linear correlation value between the image and the reference pattern by α);
- $LCwn(i) - LCw(i) \approx const$ (lossy compression reduces linear correlation by constant, i.e. it erases the constant part of watermark);
- $LCwn(i) - LCo(i) \approx const$ (after lossy compression, the constant part of watermark remains).

In our example, for the embedded watermark (with $\alpha = 3$), lossy compression (DjVu Photo) of our image ("Elementa geometriae") "ate up" around 2.25 of the

[6] This comment applies also to other similar graphics in this paper.

reference pattern. In other words, the lossy compression effect is the same as if watermark were embedded with strength $\alpha = 0.75$. So, we introduce new concept: *embedding strength coefficient after compression, α'. α' is α that survives after lossy compression.*

For all twenty reference patterns the effect is the same: for our image and $\alpha = 3$, after DjVu Photo compression, $\alpha' = 0.75$ survives.

Conclusion: How much of the watermark will survive after lossy compression, does not depend on the reference pattern choice and on linear correlation of the original image with it. In other words, *for given α, the value of α' does not depend on the watermark key.*

Figure 2 shows this watermark characteristic for JPEG compression. The test is done for the image "Cameraman". Embedding strength is $\alpha = 10$, and JPEG compressions are 5 %, 25 %, 45 %, 65 % and 85 %. The lines in the figure present linear correlation values for images with 30 reference patterns. The bottom line presents them for the original image. The top line is for watermarked images. Five lines in-between are for compressed watermarked images (five compression strengths).

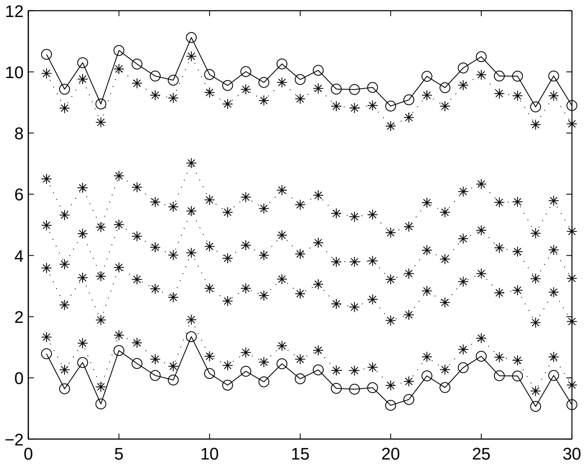


Fig. 2. Linear correlation values for images (original, watermarked and compressed watermarked) with corresponding reference patterns, for different JPEG strengths

So, in the case of JPEG compression, the conclusion is the same: after lossy compression constant part of watermark survives, no matter which key is used as the generator seed for the reference pattern.

4.1.2 Coefficient α Setting for a Robust Watermark

If we wish the watermark to be robust against lossy compression, we need to modify all statements from Section 3 (*"Embedding effectiveness"*) and replace each α with α':

- in the case of fixed embedding strength: $\alpha' = \tau + 3\sigma_{lc}$
- in the case of embedding strength adjustment:
  - $\alpha' = max(\tau - l, 0))$ if we embed binary one,
  - $\alpha' = max(\tau + l, 0))$ if we embed binary zero.

The procedure of deriving α' from α is "natural", because it matches with the events chronology – α and α' are "cause and effect":

- We embed watermark with strength α;
- we compress the image;
- we read α'.

In order to derive α from α', we have to try with different α values. We do this by using watermarks made from *arbitrary (but only one) reference pattern.*

Steps for setting α:

- We set σ_{lc} value for our image. Using σ_{lc}, we set τ and (desired value of) α'.
- We embed in our image the watermark for several α values; in this way we get several watermarked images.
- We expose these images to expected lossy compression.
- The detector reads α' for each compressed watermarked image. Then we compare the obtained α' values with the desired value. The α value which corresponds to the α' value, closest to the desired one, is appropriate.

For example, for the image – the first page of the book "Elementa geometriae", compression DjVu Photo, and eight α values, the results are as follows:

α	3	4	5	6	7	8	9	10
α'	0.72	0.99	1.36	1.90	2.58	3.34	4.14	4.92

In Figure 3, for our example α values for each of eight images can be read on the abscissa. The point of interesection of two lines in this figure shows: To have α' of at least 2.5, α should be ≥ 7.

4.1.3 Dependence of Coefficient α' on Image Content

Two images of different sizes and the same entropy (for example, "Cameraman", of dimensions 128 × 128 and 256 × 256), have the same values of α' for the same α coefficient and compression technique.

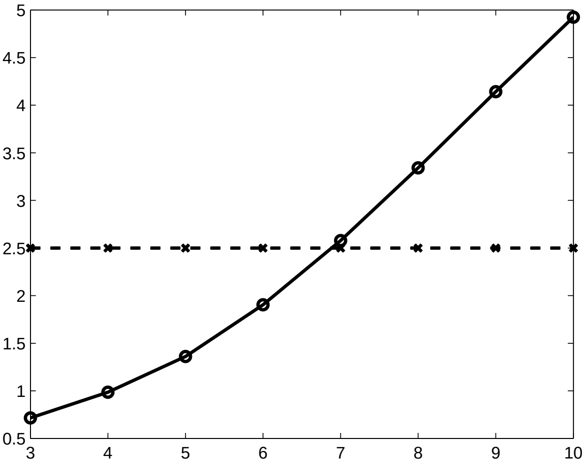


Fig. 3. Dependence α' on α

Conclusion: The coefficient α' does not depend on the image dimension.

In order to examine the value of α' (for given α and compression technique) for different images, we perform the following test: Into one image ("Cameraman") we embed the message with strength α. Then, we subject the image to (DjVu Photo) compression.

The detector reads the image, dividing it into 4×4 subimages. It treats each subimage as separate image. For each subimage it calculates entropy and α' value (for each of them $\alpha = 7$).

The results of the experiment for subimages (presented by the order defined in Figure 4a)) are presented by the graph in Figure 4b).

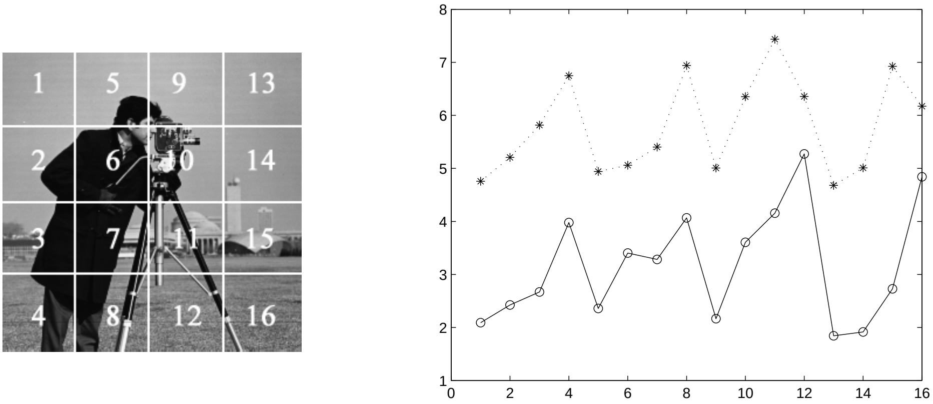
Principally, where there is bigger entropy, there is also bigger α' value. Hence, more of the watermark will remain in active than in flat regions.

Conclusion: α' depends on the image content. More of watermark remains (α' is bigger for the same α) in more active images (or image parts).

4.1.4 Dependence of Coefficient α' on Compression Technique

Into the image we embed a message bit with strength α. The image is then subjected to different compression techniques, as JPEG (based on the image division into 8×8 pixels blocks and on discrete cosine transform), as DjVu (a wavelet compression, without an image division – the transform is performed on the whole image).

In each lossy compression technique we have tested, for the given image and α coefficient, α' remains the same after compression, independently of the watermark key and linear correlation value between the original image and the embedded



a) b)

Fig. 4. a) Subimages of the image “Cameraman”. b) Upper line presents entropy values flow, and the lower one – α' for 16 image parts

reference pattern.

For fixed α, α' value is smaller in more intensive compression. For example, α' is bigger for JPEG 70 % compression than for JPEG 40 %. DjVu Clean compression will have still smaller α' value.

4.2 Robustness of a Longer Message

If we embed k patterns into the image, each with own embedding strength coefficient α_i $(i = 1, 2, \ldots, k)$, then:

$$c_w \approx c_o + \sqrt{\alpha_1^2 + \alpha_2^2 + \ldots + \alpha_k^2} \cdot r_s = c_o + \beta \cdot r_s,$$

where

$$r_s = \frac{\pm\alpha_1 r_1 \pm \alpha_2 r_2 \pm \ldots \pm \alpha_k r_k}{\beta}.$$

After compression:

$$c_{wn} \approx c_o + \beta' r_s = c_o + \sqrt{(\alpha_1')^2 + (\alpha_2')^2 + \ldots + (\alpha_k')^2} \cdot r_s.$$

After compression, the overall strength coefficient that remains will be β' (the coefficient that corresponds to β for our image and compression technique). The procedure for setting the coefficients α_i $(i = 1, 2, \ldots, k)$ as follows:

- for our image, we set parameter σ_{lc} ;
- based on σ_{lc}, we set parameters τ and α' $(i = 1, 2, \ldots, k)$;

- $\beta' = \sqrt{(\alpha'_1)^2 + (\alpha'_2)^2 + \ldots + (\alpha'_k)^2}$
- we obtain β from β' and $p = \beta'/\beta$
- $\alpha_i = \alpha'_i/p$ $(i = 1, 2, \ldots, k)$.

Example 3. Into the image c_o = "Fishingboat" (512 × 512 pixels), we embed $k = 16$ message bits (binary ones).

$$\sigma_{lc} = \sqrt{E(c_o)}/(512 \cdot 512) = 0.25 \qquad \tau = 0.25$$
$$\alpha' = [0, 0.18, 0, 0.17, 0.41, 0.54, 0.19, 0.58, 0.42, 0.58, 0.64, 0.66, 0.14, 0.57, 0.31, 0.05]$$
$$\beta' = 1.65, \qquad \beta = 5.4, \qquad p = 0.3, \qquad \alpha_i = \alpha'_i/p \quad (i = 1, 2, \ldots, k)$$

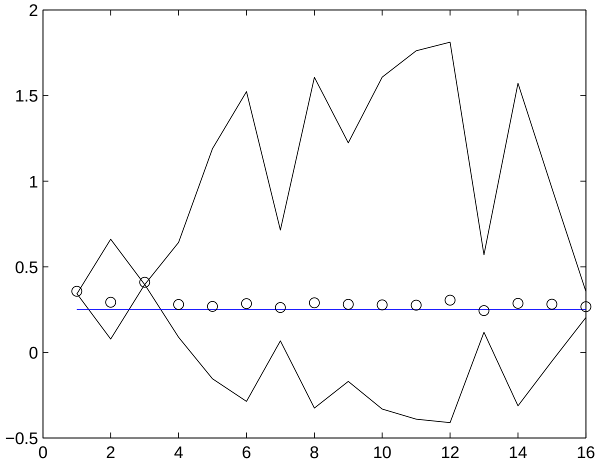


Fig. 5. Linear correlation between the image (original, watermarked and compressed watermarked) and embedded reference patterns

Figure 5 shows the linear correlation values between the image and each of the embedded patterns. The bottom line presents linear correlation values for the original image; the top line is for the watermarked image.

Roundels present linear correlation values between the compressed watermarked image and embedded reference patterns. It may be seen that their ordinates are near the threshold τ value. So, they are the smallest values needed that the detector after compression reports the image as watermarked.

Example 4. The same test is done for 64 bits message. The watermark also survives DjVu Photo compression. Overall embedding strength is $\beta = 6.94$. In Figure 6, the compressed original (without watermark) and compressed watermarked image may be seen.

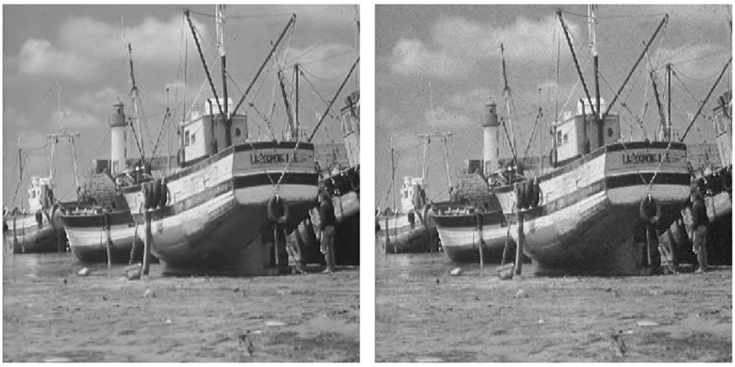


Fig. 6. Compressed original and compressed watermarked image ("Fishingboat", 512×512 pixels, $k = 64$ message bits, DjVu Photo compression)

5 AWGN WATERMARK EMBEDDING IN TRANSFORM DOMAIN

In this section, we apply the described watermarking algorithm in transform domain (embedding and detection occur in transform, instead of spatial domain). We show that there is no difference from the viewpoint of effective embedding of AWGN watermark between watermarking in spatial and in transform domains (DCT, block DCT, Fourier or arbitrary orthogonal wavelet domain).

First of all, almost all transforms which are used in image processing and compression nowadays are orthogonal (or at least unitary) linear transformations. Linear transform f respects addition and scaling. Therefore,

$$f(c_o + \alpha \cdot r) = f(c_o) + \alpha \cdot f(r).$$

Orthogonal transform f preserves the inner product and, consequently, preserves vectors lengths and angles between vectors. Consequently,

$$lc(c_o, r) = lc(f(c_o), f(r))$$

$$\|c_o\| = \|f(c_o)\| \quad \text{and} \quad \|r\| = \|f(r)\|$$

$$\cos(c_o, r) = \cos(f(c_o), f(r)).$$

Also (Parseval's, energy theorem): If $b_o = f(c_o)$, then

$$E(b_o) = \sum_{i=1}^{mn} (b_o(i))^2 = \sum_{i=1}^{mn} (c_o(i))^2 = E(c_o)$$

Orthogonal transform maps reference pattern $r = (r(1), r(2), \ldots, r(mn))$ (i.e. vector whose coordinates were taken from distribution $N(0,1)$) into vector $f(r)$ from distribution $N(0,1)$, i.e. *orthogonal transform maps reference pattern to reference pattern.*

Standard deviations from zero of linear correlation for vector c_o and reference patterns in spatial and transform domain are equal:

$$\sigma_{lc}(c_o) = \sigma_{lc}(f(c_o)).$$

It is all the same if we embed AWGN watermark into an image with strength α in spatial or in orthogonal transform domain. Correlation values as well as errors values will be the same in both domains.

Also, there is no difference in AWGN watermark robustness against lossy compression between spatial and transform domains.

6 AWGN WATERMARK EMBEDDING INTO SUBIMAGE IN TRANSFORM DOMAIN

As in the spatial domain, it is possible to embed watermark in some part of coefficients in transform domain (here also, we call it *embedding into subimage*).

Technically, there is not any difference in embedding strength setting, for effective and robust against lossy compression, speaking of the part of coefficients (subimage) and of the whole image. Effective embedding strength is determined by dimension and energy of image (or image part) in which AWGN watermark will be embedded. Robustness against lossy compression is determined by these coefficients properties with regard to expected compression. However, robustness examining when embedding in part of image coefficients in transform domain is a considerably more complex problem, compared with embedding over the whole image. The domain in which we embed watermark is not often equal to the domain used in the lossy compression. Also, during embedding, it is usually not known to which compression our image will be subjected afterwards. Therefore, in this case it is not possible to give general advice concerning embedding strength.

We may analyze only some particular cases here.

We will give some examples for such embedding in blockwise DCT domain. This domain, which is the basis of JPEG compression, is often used in digital watermarks embedding. When describing these embeddings, we will use the term *image subchannel*. Eggers and Girod have introduced this term in [9]. *Subchannel* is the vector which in blockwise DCT has coordinates – elements, with the same index in blocks. Subchannels are ordered according to zigzag order. Thus, subchannel 1 consists of all DC blocks elements; subchannel 10 consists of all elements which are on position 10 in the block (zigzag order).

So, the image of dimension $m \times n$ in DCT domain may be presented with 64 subchannels[7]:

$$s_j = (s_j(1), s_j(2), \ldots, s_j(nbl)) \quad (j = 1, 2, \ldots, 64), \quad wherenbl = mn/64.$$

[7] Here, for the purpose of simplicity, we will assume that image dimensions are multiples of 8, in order to make the partition in 8×8 blocks possible.

In the next two subsections, by several examples we illustrate the procedure for determining the strength coefficient, when embedding in part of coefficients in transform domain. Subsection 6.1 shows how we determine the strength coefficient for effective embedding in the case of embedding in image subchannels. In subsection 6.2 we examine robustness of watermark embedded in some subchannels.

In these examples we show the results for the image "Cameraman", dimension 256 × 256 pixels.

6.1 Effectiveness of Embedding Into Image Subchannels

6.1.1 Watermark Embedding Into First 32 Subchannels

In this sub-subsection we will find effective embedding measure for the first 32 subchannels

$$c_{ws} = c_o + \alpha_s \cdot r_s,$$

and the given results will be compared with embedding over the whole image

$$c_w = c_o + \alpha \cdot r.$$

We obtain the r_s pattern from the reference pattern r, by data nullifying in subchannels 33–64 (clearly, r_s pattern is no more a reference pattern).

When embedding over the whole image, we calculate the σ_{lc} parameter by the formula

$$\sigma_{lc}(c_o) = \sqrt{E(c_o)}/dim(c_o).$$

We may observe embedding into the first 32 subchannels as embedding into subimage

$$s = \{s_i, i = 1, 2, \ldots, 32\},$$

and

$$\sigma_{lc}(s) = \sqrt{E(s)}/dim(s) \approx \sqrt{E(c_o)}/(dim(c_o)/2) = 2 \cdot \sigma_{lc}(c_o),$$

because almost all image energy is concentrated in the first 32 subchannels (for example, with the image "Cameraman", this is even 99.76 % of total image energy).

Therefore, in the case of effective embedding into the first 32 subchannels, a double strength is needed compared to embedding over the whole image:

$$\alpha_s = 2 \cdot \alpha.$$

To evaluate real impact of watermark embedding on the image fidelity, we will notice that $\|r_s\| = \|r\|/\sqrt{2}$. If we "standardize" the r_s pattern, i.e. lead it to the reference pattern r norm, it will be $\|r_s^{nor}\| = \sqrt{2} \cdot \|r_s\|$. Therefore, effective embedding strength is

$$\alpha_s^{nor} = \sqrt{2} \cdot \alpha.$$

Embedding into the first 32 subchannels, in order to be effective, must be done with $\sqrt{2}$ times bigger strength, comparing with embedding across the whole image.

Also, $mse(c_o, c_{ws}) = 2 \cdot mse(c_o, c_w)$ (it is necessary, in order to effective embedding, to make the mean square error twice as large, compared to the original algorithm).

Also, embedding in this way is more noticeable – because embedding is done in lower frequencies.

Now we examine capacity and robustness for some subchannels from the first half of the zigzag scan. We will, as in [9], concentrate on representatives. Those are, traditionally, subchannels 1, 10 and 22.

6.1.2 Embedding into Subchannel 1

For the image "Cameraman" (dimension 256 × 256), the value of parameter σ_{lc} value for the first subchannel is

$$\sigma_{lc\_1} = \frac{\sqrt{E(s_1)}}{nbl} = \frac{\sqrt{1\,130.41 \cdot 10^6}}{1\,024} = 32.83.$$

On the basis of Section 3, embedding strength must be (for one message bit, and fixed strength embedding algorithm), at least $\alpha = \tau + 3 \cdot \sigma_{lc\_1}$, i.e. the minimal α value, needed for effective embedding of one bit message (if we set $\tau = \sigma_{lc\_1}$) must be even 131.33!

If we standardize embedded pattern r_1 (r_1 appears when in reference pattern r we nullify all elements except in subchannel 1), it will be $\|r_1^{nor}\| = 8 \cdot \|r_1\|$. Therefore, standardized embedded strength coefficient is $\alpha^{nor} = \alpha/8 = 16.42$. The mean square error value for this embedding is $mse(c_o, c_w) = 268.92$.

Clearly, watermark embedding with blind detection into these coefficients is out of question due to extremely high DC coefficients.

6.1.3 Embedding into Subchannel 10

$$\sigma_{lc\_10} = \frac{\sqrt{E(s_{10})}}{nbl} = \frac{\sqrt{1.14 \cdot 10^6}}{1\,024} = 1.04$$

For effective embedding of one bit message into subchannel 10 (if we set $\tau = \sigma_{lc\_10}$), it is adequate to use embedding strength of $\alpha = 4 \cdot \sigma_{lc\_10} = 4.17$, $\alpha^{nor} = \alpha/8 = 0.52$. We may embed 64 message bits with $\beta = 8 \cdot \alpha = 33.35$ ($\beta^{nor} = \beta/8 = 4.17$, $mse = 16.22$). Seemingly, this mse value is not too large; however, image changes are fairly noticeable (Figure 7). Subchannel 10 contains data of the low-frequency image component. HVS is sensitive to such data. Therefore, we should be careful with embedding strength for this component.

6.1.4 Embedding into Subchannel 22

$$\sigma_{lc\_22} = \frac{\sqrt{E(s_{22})}}{nbl} = \frac{\sqrt{0.23 \cdot 10^6}}{1\,024} = 0.47$$

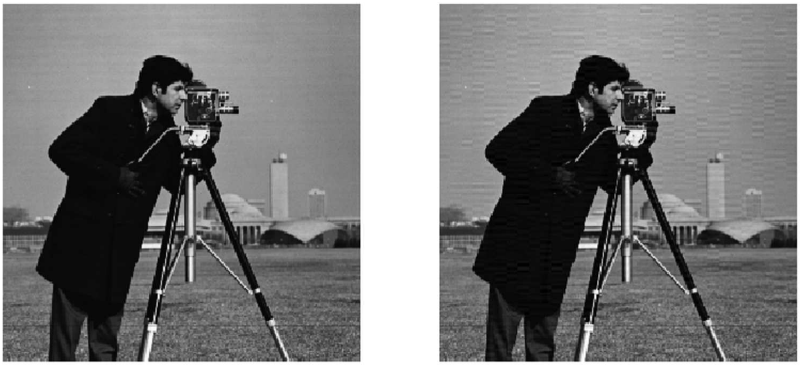


Fig. 7. Original image and image after message embedding with $\beta = 33.35$ in subchannel 10

For our image, embedding strength (in the case of one bit message, fixed strength embedding, and $\tau = \sigma_{lc\_22}$) is $\alpha = 4 \cdot 0.47 = 1.89$. Therefore, with embedding strength $\beta = 20$, in this subchannel we could embed somewhat more than 100 bits, and with embedding strength 40 – even 400. Thus, the embedding capacity in this subchannel is not a problem.

For $\beta = 20$, $mse = 6.14$; for $\beta = 30$, $mse = 13.80$; for $\beta = 40$, $mse = 24.54$. Even the last mean square error value is not too big. It corresponds to embedding strength 5 in the spatial domain. Nevertheless, the embedding in subchannel 22 is in the low frequency region, compared to the embedding across the whole image. Hence, the image subjective quality is not so good here.

6.2 Robustness Against Lossy Compression of Embedding into Image Subchannels

Now, for some of the recently presented embedding examples in block DCT domain, we analyze watermark robustness against some compression techniques. We examine the effect of embedding into some of block DCT domain coefficients – both in the case that image undergoes JPEG, and for some different lossy compression. Among these “different” compression techniques, we will also present the effect for wavelet DjVu compression.

6.2.1 Subchannel 10

In the next experiment, our watermarked image with embedded strengths $\beta = 10$, $\beta = 20$ and $\beta = 30$ is subjected to many different compressions: JPEG (from 0 % to 100 %, with step 5 %), and DjVu (Photo and Clean). Watermark robustness results for this subchannel are illustrated in Figure 8. JPEG compression intensities are presented on the abscissa. In order to present the results for DjVu compression in the same figure too, we use the abscissa value −5 for DjVu Photo, and −10 for DjVu Clean.

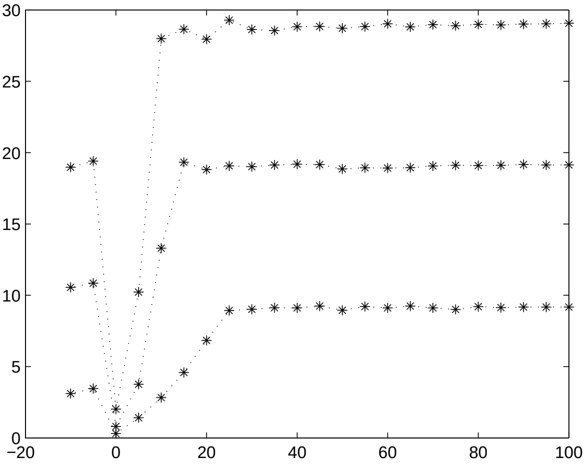


Fig. 8. Subchannel 10 robustness versus different compression qualities

We can see from the graph that embedding into this subchannel is robust against any reasonable JPEG compression strength[8], and the embedded watermark remains practically completely undamaged. Hence, a sufficiently strong embedding in these coefficients is practically immune against each sensible JPEG compression. As to DjVu compression, it destroys a part of the data embedded into the subchannel. Therefore we should take this fact into consideration when we choose embedding strength: if we want to detect our message after DjVu compression, we need to set adequate embedding strength.

6.2.2 Subchannel 22

In Figure 9 we can see subchannel robustness against three embedding strengths: 20, 30 and 40. For weak embedding strength in subchannel 22, robustness against compression is not good. If the whole embedding strength is high enough, the watermark will survive compression in the highest part. It can be seen on the graph that for robustness against somewhat stronger compression the watermark would be embedded somewhat stronger. Also, for DjVu compression we can see that, if we embed watermark strongly enough, after the compression the watermark will be detectable.

[8] With compression quality $< 15\,\%$, the image is practically worthless. Therefore, it is not a problem if watermark is not preserved.

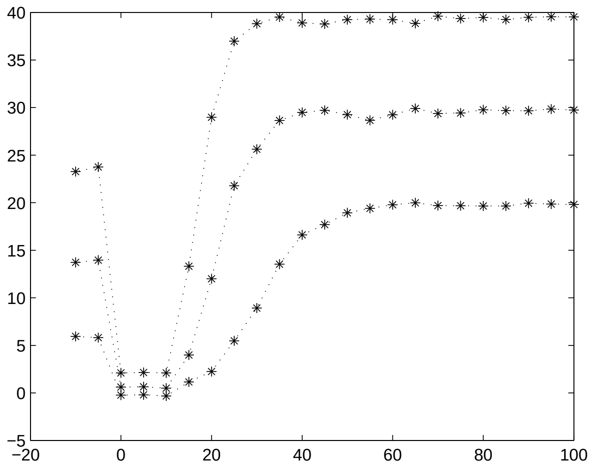


Fig. 9. Subchannel 22 robustness versus compression

Important note: The values obtained in this section are more or less similar for most natural images. Hence, the preceding two figures are similar for them. For example, in Figure 10 corresponding values are presented in subchannel 22 for embedding strength $\beta = 30$, for images "Cameraman" (red), first page of book "Elementa geometriae" (black) and "flat" image (all pixels have the same grayscale nuance) (blue line).

7 AWGN WATERMARK AND OTHER ATTACKS

In [29], habitual image operations are organized in two classes – valumetric and geometric distortions.

7.1 Valumetric Distortions

Valumetric distortions are simpler than geometric ones. They change the values of individual pixels. These attacks include additive noise, amplitude changes, linear filtering and lossy compression.

7.1.1 Additive Noise

This attack has the effect of adding a random signal. For watermarked image, additive noise adding is defined by the formula:

$$c_{w1} = c_w + s,$$

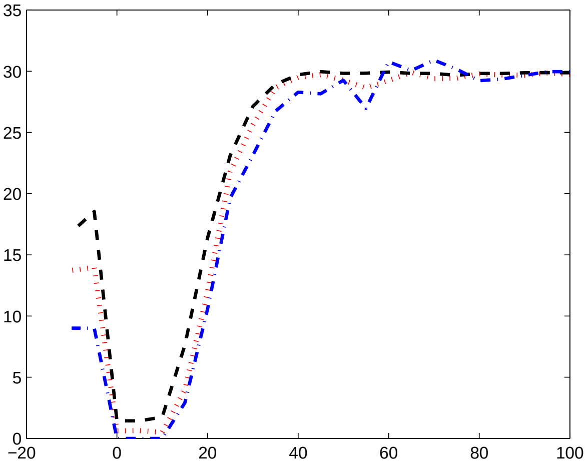


Fig. 10. Illustration of equal behavior of white Gaussian noise message in subchannel 22, for three different images

where $c_w = c_0 + \alpha \cdot r$ is watermarked image and s is a random vector chosen from some distribution, independently of c_w and reference pattern r.

Clearly,

$$lc(c_{w1}, r) = lc(c_w, r) + lc(s, r) \approx lc(c_w, r)$$

(because s is uncorrelated with r).

Therefore, an additive noise does not affect AWGN watermark (AWGN watermarking algorithm with detection based on linear correlation is robust against this attack).

AWGN watermark is also robust against *brightness changing* ($c_{w1} = c_w + n \cdot J$, where J is matrix of ones and n is an integer).

7.1.2 Amplitude Changes

They may be modeled by help of the formula

$$c_{w1} = c_w \cdot \nu,$$

where $\nu > 0$ is a scaling factor. Such operations cause *brightness and contrast changing.*

Linear correlation value will be equal to starting, multiplied by factor ν. Depending on factor ν, the linear correlation value (watermark detectability) will increase (if $\nu > 1$), or diminish (if $\nu < 1$).

7.1.3 Linear Filtering

It is given by the formula

$$c_{w1} = c_w \star f,$$

where f is a filter, and $\star$ designates convolution. Many common image operations are performed using linear filters. These operation examples are *blurring* and *sharpening effects.*

7.2 Geometric Distortions

This class of attacks includes many image distortions, such as *rotation*, *spatial scaling*, *translation*, *skew* or *shear*, *cropping*, *perspective transformation*, and *changes in aspect ratio.*

These attacks are more complicated than valumetric ones, because they displace information about image pixels in image matrix. Also, image matrix dimensions change usually. Therefore, the watermark can not be detected readily here. Yet, with these attacks, information about embedded message will not be lost, but only "*masked*".

For each geometric attack, we need to perform one *correction procedure* before detection. This procedure is not uniquely determined, and it is beyond the scope of this paper.

To illustrate this, we will depict one geometric attack – image rotation by an angle. Many of this analyze inferences could be applied to other attacks.

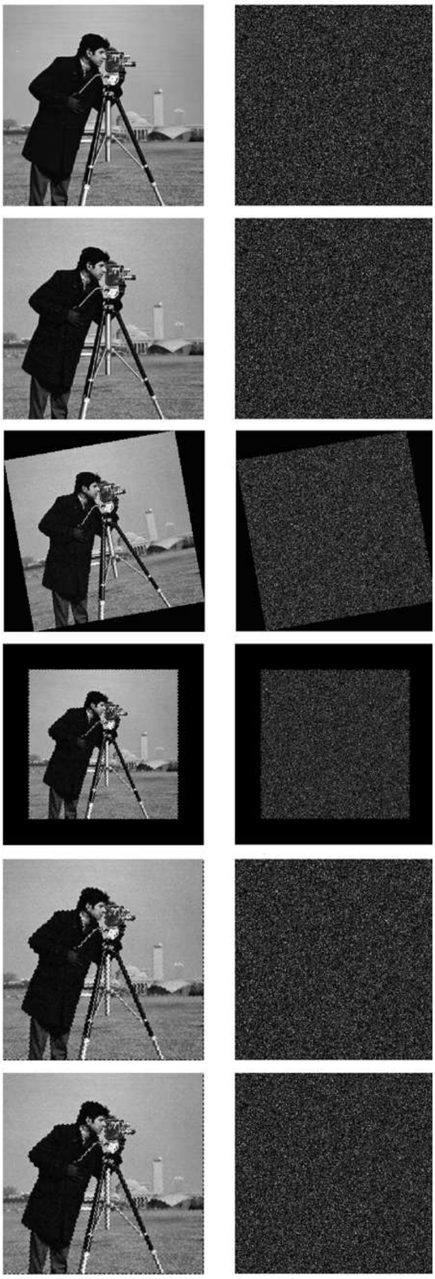
7.2.1 Robustness Against Rotation

In the case of image rotation, one of several different procedures may be used prior to the detection. If the watermarked image is rotated by angle φ, we may do one of the following to make watermark detection possible:

- we may compare (rotated) image (using linear correlation) with the reference pattern which is also rotated by φ
- before detection, we may rotate the rotated image by angle $-\varphi$; and then compare it with reference pattern, which is also rotated by angles φ and $-\varphi$
- we may compare the image rotated by φ and then by $-\varphi$ (and then cropped to original image dimensions) with the original reference pattern.

In Figure 11 linear correlation values for image and reference pattern (unrotated and rotated) are presented. In each row we present image and pattern for which the correlation is calculated. So, it is specified for each example which image and pattern (and their dimensions) we deal with, and also the linear correlation value for them.

Reference pattern is not an image: its matrix elements are real (positive or negative) numbers. For its presentation we use the solution proposed in [35].



c_0 (256,256) - original image ('Cameraman')
r (256,256) - reference pattern
$lc_0 = lc(c_0, r) = 0.787$

c_w (256,256) - watermarked image (embedding strength $\alpha = 5$)
r (256,256) - reference pattern
$lc_w = lc(c_w, r) = 5.787$

c_{wr} (297,297) - watermarked image, rotated by $\varphi (= 10^o)$
r_r (297,297) - reference pattern, rotated by φ
$lc_r = lc(c_{wr}, r_r) = 5.815$

c_{wr1} (345,345) - watermarked image, rotated by φ, and then by $-\varphi$
r_{r1}(345,345) - reference pattern, rotated by φ, and $-\varphi$
$lc_{r1} = lc(c_{wr1}, r_{r1}) = 5.591$

c_{wr2} (256,256) - watermarked image, rotated by φ, and $-\varphi$, after cropping to original image dimensions
r_{r2} (256,256) - reference pattern, rotated by φ, and $-\varphi$, after cropping to original dimensions
$lc_{r2} = lc(c_{wr2}, r_{r2}) = 5.592$

c_{wr2} (256,256) - watermarked image, rotated by φ, and $-\varphi$, cropped to original image dimensions
r (256,256) - reference pattern
$lc_{r2p} = lc(c_{wr2}, r_r) = 1.953$

Fig. 11. Linear correlation values for image and reference pattern (unrotated and rotated)

Rotation practically does not reduce the linear correlation value, if it is calculated for image rotated in the same manner and for the reference pattern. However, if we compare the rotated image (after restoration to original position by rotation in opposite direction and cropping to original dimension) with the original reference pattern, a part of the watermark will be lost (α' value will be less than embedding strength value α).

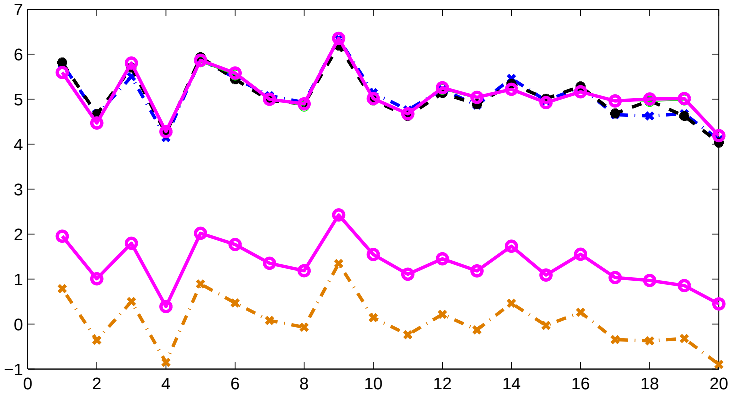


Fig. 12. Values lc_0 (bottom line), lc_w, lc_r, lc_{r1}, lc_{r2} (top lines) and lc_{r2p} (middle line) for 20 reference patterns

In Figure 12 we present the above-mentioned linear correlation values, for 20 different reference patterns, for image 'Cameraman', $\varphi = 10\,^\circ$ and $\alpha = 5$. We may see that (similarly as earlier analyzed lossy compression attack), for given image and embedding strength α after rotation by given angle φ, strength α' remains, that *does not depend on the reference pattern nor on its correlation with the original image.* Hence, for deciding with what strength α the watermark needs to be embedded so that after rotation it strength α' remains, *it is sufficient to experiment with only one reference pattern.*

The above inference remains also for other (valumetric and geometric) image distortions. The procedure of determining the necessary embedding strength α may be performed in the next steps:

1. Using the formula given in Section 3 (Embedding Effectiveness) we determine the embedding strength needed for watermark detectability.
2. In order that after attack the watermark remains detectable, the minimal strength that remains (α') must be that determined in the preceding point.
3. Experimenting with only one reference pattern we determine the embedding strength α needed that after the expected attack, the remaining strength would be at least α'.

8 CONCLUSION

In this paper we considered effectiveness and robustness against lossy compression for AWGN watermark embedded in grayscale image. We found the "minimum" embedding strength value for effective embedding and robustness against compression.

For effective embedding of a longer message, we performed the following steps:

- For our original image, we calculated the parameter σ_{lc};
- Based on σ_{lc}, we set the α parameter for each message bit; total embedding strength β was calculated using the Pythagoras theorem (embedded reference patterns are mutually orthogonal).

For each lossy compression (JPEG or wavelet), for every image and the chosen coefficient α the same part α' remains after the compression.

In order to embed the watermark robust against lossy compression, in the previous procedure we have to replace each α with α' (and β with β'). The calculating sequence is as follows:

$$\sigma_{lc} \longrightarrow \alpha'_1, \alpha'_2, \ldots, \alpha'_k \longrightarrow \beta' \longrightarrow \beta \longrightarrow \alpha_1, \alpha_2, \ldots, \alpha_k$$

There is no essential difference between the AWGN watermark embedding in spatial and in transform domain. Nonetheless, there is the difference in information amount that may be embedded in some coefficients, and robustness of such messages against lossy compression.

With other attacks, procedures for determining strength α are similar with lossy compression. With geometric attacks, however, it is necessary to first perform the proposed interventions to make that detection possible.

Acknowledgment

The author thanks Miljan Knežević and Milan Dražić (Faculty of Mathematics, University of Belgrade) and Goran Nenadić (School of Informatics, University of Manchester), for useful tips.

REFERENCES

[1] Nikolaidis, N.—Pitas, I.: Robust Image Watermarking in the Spatial Domain. Signal Processing 66, 1998, pp. 385-403.

[2] Lin, P.-L.: Oblivious Digital Watermarking Scheme with BlobOriented and ModularArithmeticBased SpatialDomain Mechanism. Journal of Visual Communication and Image Representation 12, 2001, pp. 136–151.

[3] Feng, G. R.–Jiang, L. G.–Wang, D. J.—He, C.: Quickly Tracing Detection for Spread Spectrum Watermark Based on Effect Estimation of the Affine Transform. Pattern Recognition 38, 2005, pp. 2530–2536.

[4] Koch, E.—Zhao, J.: Towards Robust and Hidden Image Copyright Labeling. IEEE Workshop on Nonlinear Signal and Image Processing, June 1995, pp. pp. 452–455.

[5] Cox, I. J.—Kilian, J.—Leighton, T.—Shamoon, T.: Secure Spread Spectrum Watermarking for Images. Audio and Video, Proceedings, International Conference on Image Processing 1996, Vol. III, pp. 243–246.

[6] Cox, I. J.—Kilian, J.—Leighton, T.—Shamoon, T.: A Secure, Robust Watermark for Multimedia. Workshop on Information Hiding, Newton Institute, Univ. of Cambridge, May 1996.

[7] Cox, I. J.–Kilian, J.–Leighton, T.—Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia.IEEE Trans. on Image Processing, Vol. 6, 1997, No. 12, pp. 1673–1687.

[8] Barni, M.—Bartolini, F.—Cappellini, V.—Piva, A.: DCTDomain System for Robust Image Watermarking. Signal Processing 66, 1998.

[9] Eggers, J. J.—Girod, B.: Quantization Effects on Digital Watermarks. Signal Processing 81, 2001, pp. 239–263.

[10] Miller, M. L.—Doerr, G. J.—Cox, I. J.: Applying Informed Coding and Embedding to Design a Robust, High Capacity Watermark. IEEE Trans. on Image Processing, Vol. 13, 2004, No. 6, pp. 792–807.

[11] Ni, R.—Ruan, Q.—Cheng, H. D.: Secure SemiBlind Watermarking Based on Iteration Mapping and Image Features. Pattern Recognition 38, 2005, pp. 357–368.

[12] Eyadat, M.—Vasikarla, S.: Performance Evaluation of an Incorporated DCT BlockBased Watermarking Algorithm With Human Visual System Model. Pattern Recognition Letters 26, 2005, pp. 1405–1411.

[13] Fan, K.—Wang, M.—Mo, W.—Zhao, X.: Novel Copyright Protection Scheme for Digital Content. Journal of Systems Engineering and Electronics, Vol. 17, 2006, No. 2, pp. 423–429.

[14] Wua, Y.-T.—Shih, F. Y.: Digital Watermarking Based on Chaotic Map and Reference Register. Pattern Recognition 40, 2007, pp. 3753–3763.

[15] Lu, W.—Lu, H.—Chung, F. L.: Novel Robust Image Watermarking Using Difference Correlation Detector. Computer Standards & Interfaces 29, 2007, pp. 132–137.

[16] Ruanaidh, J. K. O.—Csurka, G.: A Bayesian Approach to Spread Spectrum Watermark Detection and Secure Copyright Protection for Digital Image Libraries. cvpr, p. 1207, 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99), Volume 1, 1999.

[17] Qi, X.—Qi, J.: A Robust ContentBased Digital Image Watermarking Scheme. Signal Processing 87, 2007, pp. 1264–1280.

[18] Autrusseau, F.—le Callet, P.: A Robust Image Watermarking Technique Based on Quantization Noise Visibility Thresholds. Signal Processing 87, 2007, pp. 1363–1383.

[19] Wang, H. J.–Jay Kuo, C. C.:: Image Protection via Watermarking on Perceptually Significant Wavelet Coefficients. IEEE Signal Processing Society 1998 Workshop on Multimedia Signal Processing, December 7–9, 1998, Los Angeles, pp. 279–284.

[20] Joo, S.—Suh, Y.—Shin, J.—Kikuchi, H.: A New Robust Watermark Embedding into Wavelet DC Components. ETRl Journal, Vol. 24, 2002, No. 5.

[21] Zhao, D.—Chen, G.—Liu, W.: A ChaosBased Robust WaveletDomain Watermarking Algorithm. Chaos, Solitons and Fractals 22, 2004, pp. 47–54.

[22] Liu, J. L.–Lou, D. C.–Chang, M. C.—Tso, H. K.: A Robust Watermarking Scheme Using SelfReference Image. Computer Standards & Interfaces 28, 2006, pp. 356–367.

[23] Shih, F.—Scott, Y.—Wu, X.: Combinational Image Watermarking in the Spatial and Frequency Domains. Pattern Recognition 36, 2003, pp. 969–975.

[24] Wang, Y.—Pearmain, A.: Blind Image Data Hiding Based on Self Reference. Pattern Recognition Letters 25, 2004, pp. 1681–1689.

[25] Wu, X.—Guan, Z. H.: A Novel Digital Watermark Algorithm Based on Chaotic Maps. Physics Letters A 365, 2007, pp. 403–406.

[26] Qi, H.—Zheng, D.—Zhao, J.: Human Visual System Based Adaptive Digital Image Watermarking. Signal Processing 88, 2008, pp. 174–188.

[27] Su, J. K.—Hartung, F.—Girod, B.: Digital Watermarking of Text, Image, and Video Documents. Comput. & Graphics, Vol. 22, 1998, No. 6, pp. 687–695.

[28] Ruanaidh, J. J. K.—Pun, T.: Rotation, Scale and Translation Invariant Spread Spectrum Digital Image Watermarking. Signal Processing 66, 1998, pp. 303–317.

[29] Cox, I. J.—Miller, M. J.—Bloom, J. A.—Fridrich, J.—Kalker, T.: Digital Watermarking and Steganography. Morgan Kaufmann Publishers 2008.

[30] Baudry, S.–Nguyen, P.—Maitre, H.: Optimal Decoding for Watermarks Subject to Geometrical Attacks. Signal Processing: Image Communication 18, 2003, pp. 297–307.

[31] Feng, G.—Jiang, L.—He, C.—Xue, Y.: Chaotic Spread Spectrum Watermark of Optimal SpaceFilling Curves. Chaos, Solitons and Fractals 27, 2006, pp. 580–587.

[32] Craver, S.—Memon, N.—Yeo, B.-L.—Yeung, M.: Can Invisible Watermarks Resolve Rightful Ownerships? IBM Research Technical Report RC 20509, July 1996.

[33] Craver, S.—Memon, N.—Yeo, B.L.—Yeung, M. M.: Resolving Rightful Ownerships With Invisible Watermarking Techniques: Limitations, Attacks, and Implications. IEEE Journal on Selected Areas in Communications, Vol. 16, 1998, No. 4, pp. 573–586.

[34] Craver, S.—Memon, N.—BoonLock Y.—Yeung, M. M.: On the Invertibility of Invisible Watermarking Techniques. In Proceedings of the International Conference on Image Processing (ICIP '97), Volume 1, 26–29 Oct. 1997, pp. 540–543.

[35] Vučković, V.: Image and Its Matrix, Matrix and Its Image. Review of the National Center for Digitization, Issue 12, 2008, http://www.ncd.matf.bg.ac.yu/casopis/12/NCD12017.pdf.

[36] Marvin, K. S.: Probability Distributions Involving Gaussian Random Variables: A Handbook for Engineers and Scientists. Kluwer Academic Publishers 2002.

Vesna Vučković received her B. Sc. degree in mathematics and M. Sc. in computer science from Faculty of Mathematics, University of Belgrade, Serbia. She works as a research assistant in the Computing Laboratory at Faculty of Mathematics. Her research interests include image processing and digital watermarking.