Scientia Iranica, Vol. 13, No. 4, pp 404 412

SCIENTIA IRANICA

Research Note

Achieving Higher Stability in Watermarking According to Image Complexity

M. Jamzad* and F. Yaghmaee[1]

One of the main objectives of all watermarking algorithms is to provide a secure method for detecting all or part of the watermark pattern in case of the usual attacks on a watermarked image. In this paper, a method is introduced that is suitable for any spatial domain watermarking algorithm, so that it can provide a measure for the level of robustness when a given watermark is supposed to be embedded in a known host image. In order to increase the robustness of the watermarked image, for a watermark of M bits, it was embedded $N = s \times M$ times, where s is a small integer. Doing this, the entire image is divided into 16 equal size blocks. For each block, the complexity of the sub-image in that block is measured. The amount of repetition of the watermark bits saved in each block is determined, according to the complexity level of that block. The complexity of a sub-image is measured using its quad tree representation. This approach not only secures the watermarked image with respect to usual attacks, but also, enables one to save longer bit patterns of the watermark, while maintaining a good level of similarity between the original image and the watermarked one. For evaluating the performance of this method, it has been tested on 2000 images having low, medium and high levels of complexity and the result have been compared with the same set of images, without considering the complexity of sub-images in blocks. The new method provided 17% higher stability.

INTRODUCTION

Watermarking digital images can be used to insure the legitimacy of the sending side and, also, proof of ownership of the digital images. When a digital image is watermarked, it means that a pattern is hidden in the original image, in such a way that the watermarked image looks identical to the original one when seen. However, the owner of the image can prove the existence of a watermark pattern by analyzing the watermarked image using a decomposition program. One of the major difficulties in watermarking algorithms is that they should be able to resist the usual attacks on images, such as noise addition, filtering, compression, rotation and scaling, etc. By resisting attack, it is meant that if the watermarked image is modified by any of the above mentioned attacks, still, one should be able to extract all or most bits of the watermark pattern. In short, a secure watermarking algorithm is one where different types of degradation applied to the digital images cannot destroy the watermark pattern. Figure 1 shows an example of how an original image (Figure 1a) can be modified by attacks, such as 75% additive noise, the chess board effect, blurring, sharpening and clipping, respectively [1].

Watermarking algorithms can be categorized into two groups. The first group includes those that embed the watermark pattern inside the pixel values, in the spatial domain [2-4]. The second group works in transform domains such as Fourier, DCT, Wavelet, etc. [5-8]. Usually, most watermarking algorithms, in order to extract the watermark pattern, perform the reverse procedure of embedding, such as changing the least significant bits of pixels [3], adding the mean gray level of the image to an image block [2] and modifying the Fourier or Wavelet transform coefficients [7]. Although these methods of extraction seem to work well, they are not robust, with respect to all transforms or attacks. Some good measures for evaluating such robustness are based on calculating the false positive and false negative error rates in detecting the watermark [2].

In this paper, for watermark embedding and

*. *Corresponding Author, Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

1. *Department of Computer Engineering, Sharif University of Technology, Tehran, I.R. Iran.*

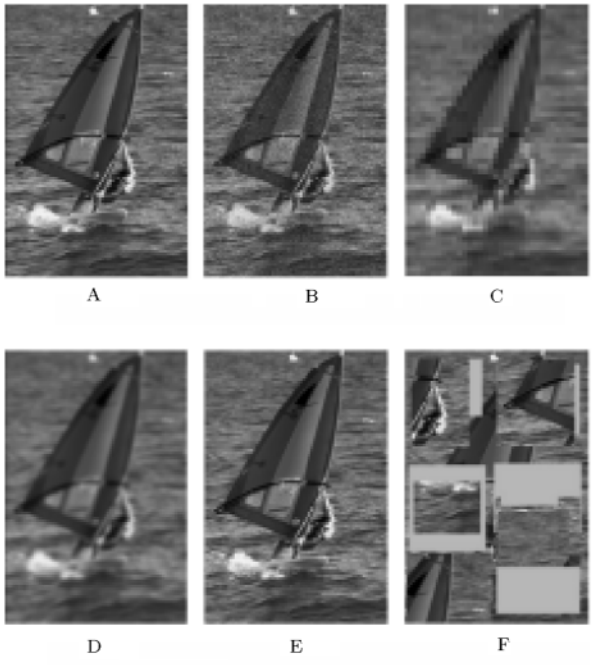

Figure 1. (a) Original image; (b) to (f) The watermarked original image that is attacked by 75% additive noise, chess board effect, blurring, sharpening and clipping, respectively.

detection, the authors' previous work has been used, which is fully described in [1]. In short, the authors in [1] divide the entire host image into 16 blocks. A fixed number of bits, from a repetition of the watermark bits, is saved into each block. The location of bits in each block that should embed a watermark bit is selected randomly. The detection algorithm is based on a Naive-Bayes classifier, that is, instead of using a fixed and pre-defined threshold for watermark detection, it uses the results obtained in a training and learning phase.

In order to obtain higher stability while maintaining image quality, the idea was introduced of distributing the number of repetitions of watermark bits, based on the content of the sub-images, into blocks. The content is measured by determining the image complexity or its detail. The quad-tree representation of a sub-image was used to calculate its complexity. The performance of this method was evaluated by comparing its results with a case when the sub-images complexity measures are not taken into consideration, that is, when a fixed number of pixels in each block is modified. The results of this comparison are given in the next sections. The initial version of this work is presented in [9].

SAVING THE WATERMARK

The watermarking algorithm can be applied to both gray scale and color images. In the case of color images, the watermark is hidden in the blue plane, because the human vision system has a low sensitivity to the color blue [10] and, therefore, modifications in the blue plane are not easily detected by the naked eye.

The watermark can be a character string or a small image (e.g., a trade mark). In both cases, the data to be hidden inside the original image is the bit string representing the ASCII code of the character string, or the bit string representing the small image. The following idea of saving one bit of the watermark and its extraction was initially introduced in [11].

Saving One Bit of Watermark

For simplicity, assume one wants to save one bit, S, of the watermark in the original image. One pixel, $P(i,j)$, is randomly selected from the original image. This random number can be generated by a unique key, which is the private key between sender and receiver. In order to save S into pixel $P(i,j)$, blue component, $B_{i,j}$, of one pixel is modified according to the following equation:

$$B_{i,j} = B_{i,j} + (2S \quad 1) \times q \times Y_{i,j}, \tag{1}$$

$Y_{i,j}$ is the gray level intensity of pixel (i,j). It is defined as $Y_{i,j} = 0.299R_{i,j} + 0.587G_{i,j} + 0.114B_{i,j}$ [10].

The parameter, q, is called the strength of the watermark. It means that higher values of q will result in more changes in the blue component of pixel P. This causes a lowering of quality in the watermarked image, but, a strengthening of the watermarked image. To determine the position of pixels on which a watermark bit should be embedded, a "KEY" was used that is the seed of a random number generator. This key is supposed to be known by the users of the system and is used to locate the watermarked pixels in a detection step.

Detection and Extraction of One Bit of Watermark

The authors algorithm is of the blind type, which assumes that the receiver does not have the original image. On the other hand, to verify if the blue component of a pixel contains a watermark bit or not, one needs to know the original value of the blue component of that pixel for a comparison. Therefore, one must estimate the value of the blue component of the original image from the watermarked one. If $B_{i,j}$ is the value of the blue component at pixel (i,j) in the watermarked image, then, its corresponding value in the original image is estimated by $\hat{B}_{i,j}$, according to the following equation:

$$\hat{B}_{i,j} = \frac{1}{4C} \sum_{k=\ c}^{c} (B_{i,j+k} + B_{i+k,j} \quad 2B_{i,j}), \tag{2}$$

where $c = 1$ for a 3×3 cross mask. Let:

$$\delta = B_{i,j} \quad \hat{B}_{i,j}. \tag{3}$$

Then, the value of a watermark bit at position (i, j) is extracted into bit S, as follows:

if $\delta \geq 0$ then $S = 0$, else $S = 1$.

Multiple Saving of One Bit

In order to increase algorithm stability, each bit of watermark is saved in a few different locations, randomly selected on the original image. Assume that one bit of watermark is saved in K pixels (i.e., $K > 1$) of the original image: In this case, one has, $\delta_k = B_k \quad \hat{B}_k$ where B_k and $\hat{B}_k$ are the actual and estimated values of the blue component at the kth pixel. Let $\hat{\delta}$ be the mean value of δ_k in K pixels. Then, using the sign of $\hat{\delta}$, the value of a watermark bit, S, is estimated as follows:

if $\hat{\delta} \geq 0$ then $\hat{S} = 0$, else $\hat{S} = 1$.

Figure 2a shows the visualization of separations of zero and one bits of the S bits calculated on 32 pixels in a block of an image in which a watermark pattern was saved and the image has not gone through any modification (i.e. attack). A threshold value for δ, as shown by a dashed line in Figure 2a, can separate the zero and one bits correctly. Figure 2b is the same visualization for a noisy image. Note that, in this case only in a few blocks the correct value of the watermark bit can not be distinguished. However, if the number of such errors is small, one might still be able to argue that a good percentage of the watermark bit is extracted and, therefore, conclude the correct extraction of the watermark. But this approach is not reliable, since it is difficult to decide on an upper bound for an acceptable number of wrong detections.

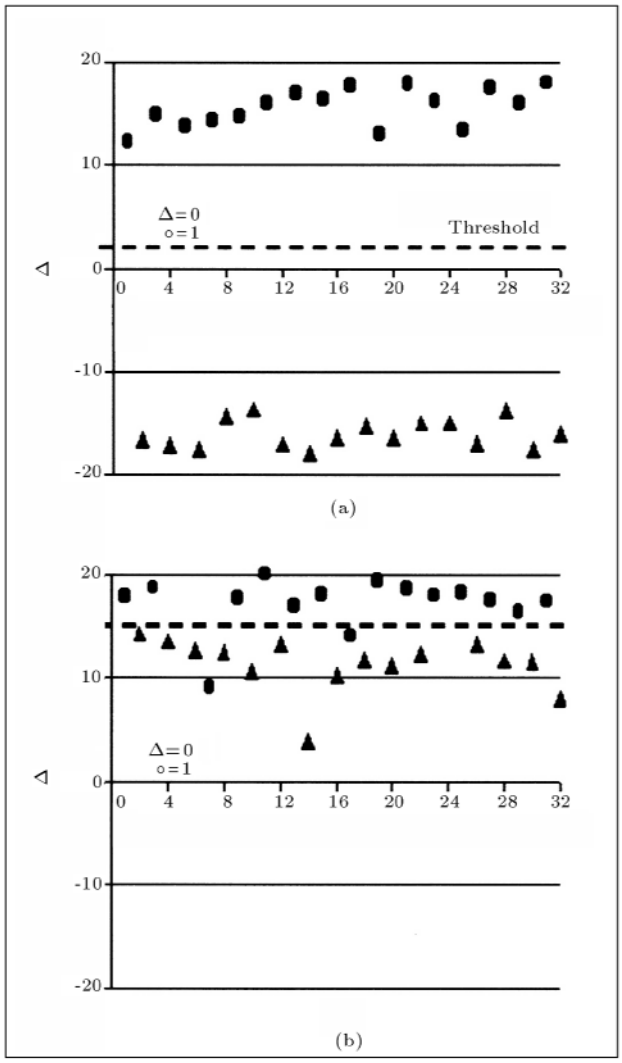

Figure 2. (a) Visualization of a threshold (dashed line) to calculate δ on 32 pixels in a block of image with no noise; (b) The same but for a noisy image. Note: The watermark pattern is 0101(32 bit).

Storing and Retrieving the Watermark Pattern in Blocks

The original image is divided into 16 equal size blocks. One unique watermark pattern is saved in each of these 16 blocks, using the method described previously. In each block, the bits of watermark pattern are saved on random locations. These locations are generated using a random number generator that uses an eight-digit integer number as its seed. This seed is called a key. The same key is used for all 16 blocks. The reason for this redundancy in saving the same watermark pattern in 16 blocks is to increase the robustness of the watermark retrieval algorithm, in case of usual attacks on the watermarked image.

For the watermark extraction, in each block, the sequence of locations in which the bits of the watermark pattern are stored, is determined using the above mentioned key. Since the exact bit pattern of the watermark is known in the receiver side, at each of these locations, a comparison is carried out between the real watermark bit value and the bit value estimated at that location. If there is a match, it means that one bit of pattern has been perfectly retrieved from that location. However, there is a possibility that, in this retrieval, the algorithm might detect some bits in error. In addition, in practice, it is very difficult to define a threshold number, by which an upper bound could be clearly determined for the number of correctly detected bits, sufficient to conclude the existence of a watermark pattern in that block.

Therefore, in order to find a measure for such a conclusion, it was decided to go through a training and learning procedure at the receiving side. This

procedure uses the classification method of the Naive-Bayes algorithm [12].

Using Naive-Bayes Classifier

The Naive-Bayes classifier applies to learning tasks where each instance, x, is described by a conjunction of attribute values and where the target function, $f(x)$, can take on any value from some simple finite set, V [12]. This classifier has two steps, a training step, in which a set of attributes, their probability and the probability of values in target functions, are calculated. The second step is an evaluation step, in which an unknown attribute is given to the system and which will assign the maximum value of a target function for that input.

In this regard, the attributes are defined in the form of a topple, such as $(a_1, a_2, \cdots, a_n)$, where each a_i can take values within a certain boundary, depending on the problem. The target function, V, takes discrete values, V_j, $j \geq 2$, depending on the problem. The Naive-Bayes target function is defined as follows:

$$V_{\mathrm{NB}} = \max \mathrm{Arg} \left(P(V_j) \prod_{i=1}^{n} P(a_i|V_j) \right). \quad (4)$$

In the authors' application, the parameters in the above equation are defined as follows.

$n = 16$ indicates that the image is divided into 16 blocks. V_j is the target function, which takes only two values, V_1 = "yes" and V_2 = "No" (which means the watermark exists or not). a_i represents Error B_1, Error $B_2, \cdots$, Error B_{16}, where Error B_i shows the number of bits that were not correctly detected as watermark bits in block number i. V_{NB} is the value of the target function. It indicates the probability of the existence or not of the watermark pattern, depending on the $P(a_i|V_j)$ term in Equation 4.

In addition, P (Error B_i|Yes), P (Error B_i|No), $i = 1, 2, \cdots, 16$, are defined as the probability of "correctly detecting" and "wrongly detecting" the existence of the watermark, in case only B_i bits of the watermark pattern were wrongly extracted, respectively.

ALGORITHM CONSIDERING IMAGE COMPLEXITY

As described in the previous sections, the host image was partitioned into 16 blocks for the distributing watermark in the embedding process. However, if a constant number of bits is saved in each block, when the number of bits increase, the degradation of the original image, in blocks with less complexity, will become evident. This is a clear limitation on the number of bits that can be embedded in the original image, which, on the other hand, affects the watermark stability. This problem could be solved by taking advantage of the fact that the content (i.e., the level of complexity or detail) of each block may be different from others and, thus, the distribution of the watermark bits in each block should be related to the sub-image content of the block. Therefore, the image complexity is, first, computed in each block and, then, the number of bits to be saved into each block is related to its complexity measure.

Determining Image Complexity

The image complexity can be computed according to its spatial or frequency features. There are several methods for this computation, such as [13,14]. In this paper, since the spatial domain statistical distribution of the pixels is used for embedding the watermark bits, it was found that a measure obtained by quad-tree representation of an image is most suitable for the authors' purpose, because it gives them a good measure with which to determine the capacity of the image for modification without losing image quality.

Some other researchers have used the quad-tree for complexity applications like [15,16], but, their focus is on quad-tree estimation and simplification for the purpose of compression. Originally, quad-tree representation is introduced for binary images, but, in the application of watermarking the color images, the blue plate of the image is taken as a gray scale image and the quad tree is obtained for it.

For gray scale images, the difference between maximum and minimum gray level at each sub-block is used as a measure of contrast. This difference is calculated in each 16 blocks of the image, if it is lower than a predefined threshold, it means that there is not much detail in that block (i.e., there is too much similarity among the pixels of the block), thus, that block is not divided any more. Otherwise, the division of a block into 4 blocks is continued until either a block cannot be divided any further or it reaches a block size of one pixel.

Figure 3 shows the blue plane of a color image. Figure 4 is a binary image that is a visual representation of the final quad-tree constructed for Figure 3. In this representation, all four corners of each square, corresponding to a node in the quad tree, is shown with black pixels. The areas with a high density of black pixels are those in which more divisions of quad-tree nodes have occurred (this is obvious at the edges). It means that the areas with a high density of black pixels correspond to areas with high complexity. On the contrary, the white areas correspond to areas with low complexity (i.e., the sky region in this figure).

Figure 5 shows a quad tree representation of two gray-scale images. Figure 5b has an unbalanced tree with low depth and near balance, but Figure 5a has a tree with high depth.

Figure 3. The gray scale representation of the blue component of a color image.

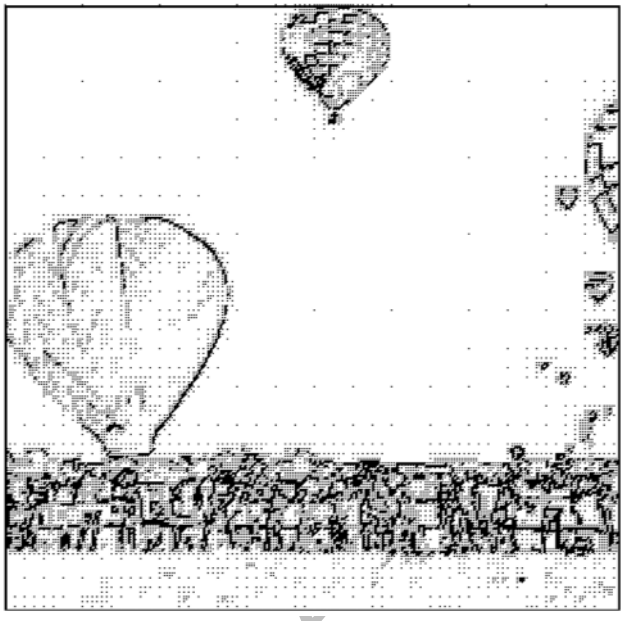

Figure 4. Visualization of quad tree representation of Figure 3.

It is clear that if the number of nodes in the quad tree is taken as a measure of complexity, then, both images will have similar complexity, since there is not a significant difference between the number of nodes in their quad trees (i.e., 29 and 21). For calculating the complexity, the sum of multiplication of the number of nodes at each level by 2 to the power of that level number was used. More description on the reason for this computation is given later.

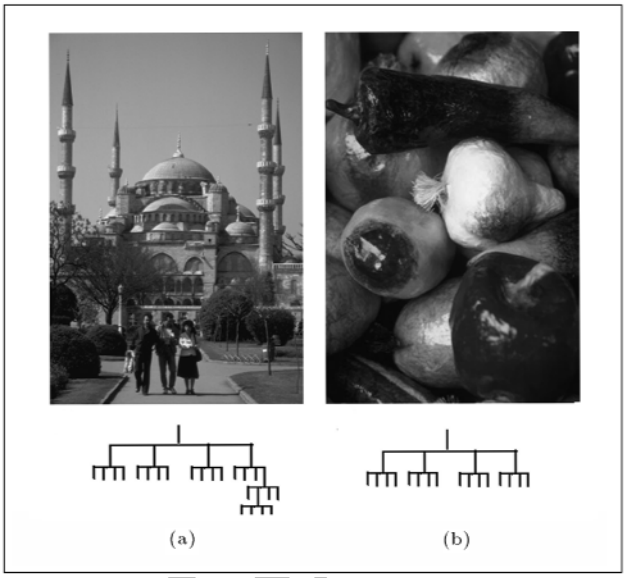

Figure 5. A quad-tree representation of two images. (a) An unbalance quad tree with high depth with 29 nodes; (b) A balanced quad tree with low depth with 21 nodes.

Distributing Watermark Bits on Image Blocks According to Their Complexity

The original images are assumed to be of size 512×512. Initially, an image is divided into 16 equal size blocks (i.e., sub-images). Let $C_1, C_2, \cdots, C_{16}$ be the image complexity calculated for blocks $1, 2, \cdots, 16$, and $C_{\min}$ and $C_{\max}$ be their minimum and maximum. Now, a complexity division factor, d, is defined as follows:

$$d = \frac{C_{\max} \quad C_{\min}}{K}, \tag{5}$$

where K is a scale factor for d. It means that the complexity is being quantized into K different classes. For simplicity, one assumes $K = 2^m$. The ranges of $[C_{\min}, C_{\min} + d], [C_{\min} + d, C_{\min} + 2d], \cdots$ and $[C_{\min} + (K \quad 1)d, C_{\min} + Kd]$ are assigned to classes $1, 2, \cdots$ and K, respectively. For each block $i, i = 1, 2, \cdots, 16$, the variable, QC_i, is referred to be the level of complexity of that block.

Now, assume one wants to change N bits of the original image to embed M bits of the watermark pattern (e.g., $N = s \times M$ where s is an integer). Let n_i, which is the number of bits that can be modified in a block, i, be determined, according to the following equation:

$$n_i = \frac{QC_i}{\sum_{j=1}^{16} QC_j} \times N. \tag{6}$$

In this way, by choosing an appropriate value for s, which is the number of repetitions that a watermark bit has embedded in an image, one can save the total, $N = s \times M$, bits in all 16 blocks, such that the number of bits saved in each block is directly related to its class of complexity.

Determining the Complexity of Watermarked Image

Because many bits of the original image are modified in its watermarked version, the complexity of the original image and that of the watermarked one are no longer the same. It means that one cannot carry out the quad tree calculation on the watermarked image to determine the complexity values of the original image blocks. Another reason is that the watermarked image might have been attacked and modified by geometrical transformation, compression, etc.

As a solution to this problem, the number related to the complexity of the original image is considered as part of the key. Since it is assumed that $K = 2^m$ and $QC_i \prec K$, therefore, m bits are needed to show the complexity number of each block. The complexity numbers of all 16 blocks can be shown with $m \times 16$ bits. Therefore, the key in the algorithm was made to have two parts. The first part relates to the seed of the random number generator and the 2nd part is the 16 m bits corresponding to complexity numbers of all 16 blocks. Now, by having the key, the complexity values of blocks in the original image can be determined.

STABILITY COMPARISON

In this section, the stability of the new method is compared with the case when a fixed number of bits is saved in each block. A set of 1000 images (fair mixture of low, medium and high level of complexities) was used to train both algorithms, using the Naive Bayes classifier [1] and $N = 40,000$ for watermark bits repetition.

For the new algorithm, the number of complexity classes was assumed to be $K = 4$. Note that setting $K = 1$ changes the new algorithm to the algorithm which assumes only one level of complexity for all blocks. For simplicity of discussion, in the following, these two algorithms are named the K1-algorithm (e.g., fixed capacity) and the K4-algorithm (e.g., variable capacity), respectively.

For comparing the capacity of images in embedding watermark bits, 2000 images that were different from those used in the training stage were watermarked. The watermarked images from both K1 and K4-algorithms were randomly selected and attacks, such as random noise addition with different percentages (200 images with 70% and 200 images with 80% noise addition), blurring with different filter size (100 images with a 3*3 window and 100 images with a 5*5 window), JPEG compression with different quality factors (200 images with a 65% quality factor and 200 images with a 75% quality factor), chessboard effect (200 images), clipping and cropping with different percentages (totally 400 images), histogram equalization (200 images) and geometrical transformation, such as rotation (200 images) were applied on them. In fact, most of the attacks were simulated that were considered in Stirmark [12]. Stability is defined as the percentage of images that an algorithm can classify correctly concerning the existence of a watermark pattern, after the attacks mentioned before.

For calculating the complexity, the sum of multiplication of the number of nodes at each level by 2 to the power of that level number was used, as described later.

Figure 6 shows the result of this comparison for $N = 40000$ and $N = 80000$ pixels. As seen, in a case of $N = 40000$, the K4-algorithm gives a 13% increase in capacity. This is an increase in stability for these images, but, for $K = 8$, the classes reach to 83%, which shows a 17% increase compared to the K1-algorithm.

However, for $N = 80000$, only a 4% increase is achieved. This bad performance of the later case is mainly due to the fact that, because of the large number of watermark bits, all blocks, independent of their complexity level, have gone through the maximum possible changes (i.e., the host image is fully saturated by repetition of watermark bits).

In addition, as seen in Figure 6, the stability of the watermark has reached 78% for $N = 40000$ in the K4-

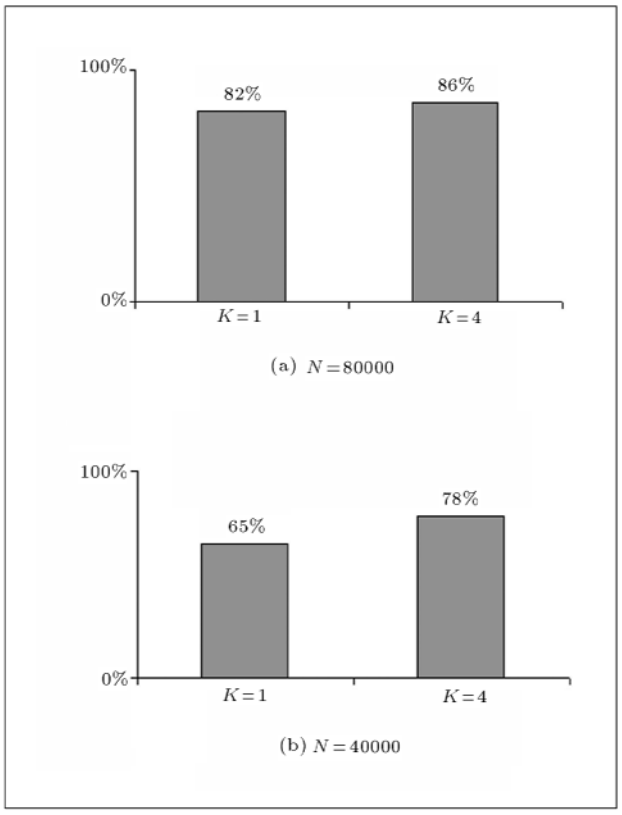

Figure 6. (a) Performance evaluation of K1- and K4-algorithms for $N = 40000$ bits and (b) for $N = 80000$. The Y axis shows the percentage of stability.

algorithm compared to 82% for $N = 80000$ in the K1-algorithm. Moreover, 78% of stability for $N = 40000$ in the K4-algorithm is very close to 82% for $N = 80000$ in the K1-algorithm. It means that by using the K4-algorithm in an image of size 512×512 and by changing 15% of the pixels (i.e., 40000 pixels) instead of 30% (i.e., 80000), one can reach the same stability expected by the K1-algorithm.

How the Number of Classes of Complexity Can Affect the Stability

The stability of watermarked images were examined by saving $N = 40000$ bits of watermark in each of 2000 images described in the previous section with a different number of classes of complexity. Figure 7 shows the stability for the number of classes of $K = 2, 4, 8, \cdots$ and 32. As seen in this figure, the stability increases for $K = 2, 4$ and 8, but, a further increase in the number of classes of complexity, $K = 16$ and 32, reduces the stability.

The reason is that, by increasing K, the distribution of watermark bits in blocks with low complexity is reduced in an unpredictable way. In other words, the distribution is mainly done in certain blocks. For example, for $K = 32$, the number of bits distributed in a class with maximum complexity is 32 times that of a block with lowest complexity. This means that in some blocks only a few repetitions of watermark bits are embedded, therefore, the watermark may not be extracted correctly. This is the main reason for the decrease in stability. The experimental results showed that defining $K = 4$ or $K = 8$ classes of complexity gives the highest stability.

HOW THE SHAPE OF QUAD-TREE IS RELATED TO STABILITY

A good measure of complexity should be based on the overall structure of the quad tree. Therefore, the two most important factors in this regard will be the number of nodes and the depth of the quad tree. Having the quad tree representation of an image, the following four measures were measured for complexity:

1. The number of nodes in the quad tree,
2. A long decimal number, whose digits, from the lowest level to the root, are the number of nodes at each level,
3. The sum of multiplication of the number of nodes at each level by that level number,
4. The sum of multiplication of the number of nodes at each level by 2 to the power of that level number.

Figure 8 shows the results of stability measurements for over 1000 images using each of the above four methods. As seen in this figure, Method 4 gives the best stability. The reason is that the overall structure of the quad tree is determined by the number of nodes and the depth of the quad tree. Therefore, this method was selected to measure the image complexity.

Figure 5a has an unbalanced quad tree with high depth, but, Figure 5b has a well balanced quad tree with low depth. Lower stability in Figure 5a is because more nodes are assigned to only a few blocks of the image and, as a result, most of the watermark bits have had to be saved in these few blocks. For Figure 5b, the nodes are evenly assigned to blocks of the image (i.e., a more balanced tree) and, as a result, the watermark bits are evenly distributed over the entire image.

RELATION BETWEEN STABILITY AND THE TOTAL NUMBER OF PIXEL MODIFICATIONS

To see how the total number of pixel modifications in the watermarked image can affect its stability, 200 images were tested, such that 5% to 45% of image

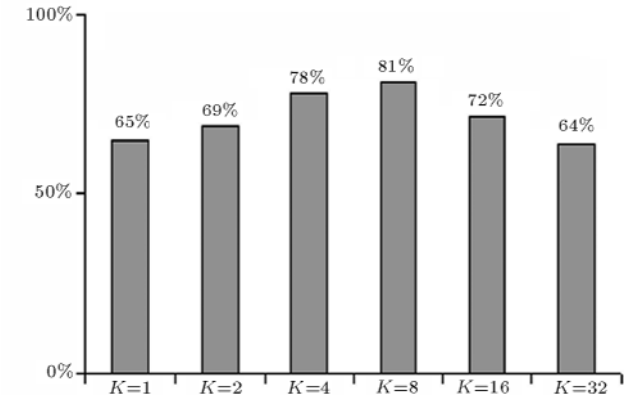

Figure 7. The relation between stability and the number of classes of complexity.

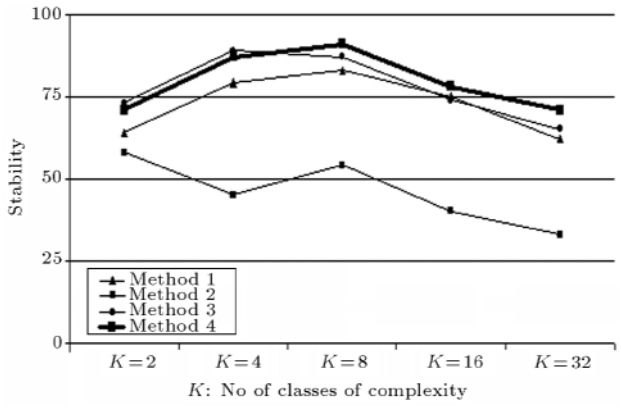

Figure 8. Comparison of four methods of complexity measurement and the number of classes of complexity with watermark stability. The results were obtained by applying the methods on 1000 images.

pixels were modified to embed the watermark bits. The stability was calculated for each case for the $K = 4$ class of complexity (note that $K = 4$ has given the best stability, as described before). Figure 9 shows that the stability increases with an increase in the percentage of the number of pixels modified, but, reaches a stable state after a 25% modification. The reason for this behavior is that the sub-image blocks in the host image become saturated by watermark bits. Therefore, further modification can, not only increase the stability, but, decrease the quality of the watermarked image.

Voloshynovsky and others [17] proposed a Noise Visibility Function (NVF) that can be used for computing maximum allowable distortion for any pixel in image. Briefly, in a Gaussian distribution, the NVF can be calculated by the following equation:

$$\text{NVF} = \frac{1}{1 + \sigma_x^2(i,j)}, \quad (7)$$

where σ_x^2 is the local variance of the image in a window centered on the pixel (i, j). According to the above equation, by calculating NVF, one can obtain the maximum allowable distortions of each pixel in the following equation:

$$\Delta(i,j) = (1 \quad \text{NVF}(i,j)) \times S_0 + \text{NVF}(i,j) \times S_1, \quad (8)$$

where S_0 and S_1 are the maximum allowable pixel distortions in textured and flat regions, respectively. These parameters completely depend on a human visual system. Typically, S_0 and S_1 are considered as 30 and 3 for natural images [17].

In the author's algorithm, if any of the above mentioned watermark distribution methods are used, the allowable threshold given by NVF is satisfied. In other words, the invisibility of the watermark is guaranteed.

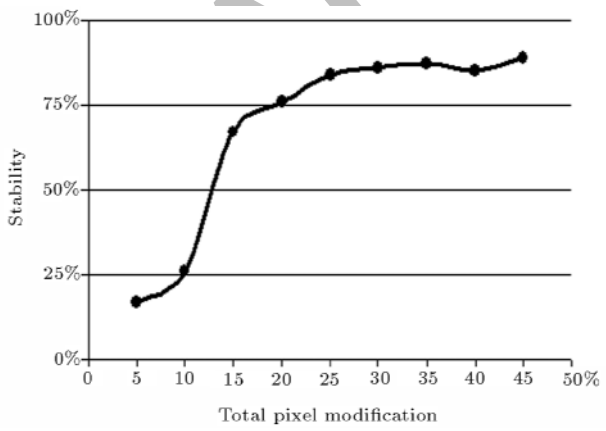

Figure 9. The relation between the percentage of total number of pixels modified in host image and the stability.

CONCLUSION

There are a great number of watermarking methods that work in spatial and frequency domains. All these methods try to provide a secure way to embed a watermark in an image, in such a way that it would be robust, with respect to some normal attacks on digital images. However, the performance of these algorithms differs with respect to the type of attack and the amount of its robustness with respect to those attacks. In this paper, it is shown that, regardless of the method of watermarking used, one can increase its robustness by choosing an intelligent method of distributing watermark bits over the entire image. In this regard, the entire host image was divided into 16 blocks and, for each block, by using a quad tree representation, a measure of complexity for the sub-image of that block was introduced. The experiments showed that the sum of multiplication of the number of nodes at each level of the quad tree by 2 to the power of that level number, was the best measure for the complexity.

In this paper, the distribution strategy was examined by dividing the complexity measure of blocks into $1, 4, 8, 16, \cdots$ classes. Then, the performance of the watermarking algorithm was evaluated with respect to its stability towards some usual attacks, such as, noise addition, smoothing, compression, clipping, rotation, shuffling and the chess-board effect. The best result was obtained with 4 classes of complexity. It is noted that the algorithm described could be broken by attacks not mentioned previously. However, it is very difficult (if not impossible) to design a watermarking algorithm that is robust with respect to all possible attacks.

Moreover, as a result of this work, it is concluded that there is a limit for the stability when one is forced to embed a particular watermark pattern into a given host image. It is suggested that it is possible to get an optimum robustness for a given host image, if one can limit the number of bits in the watermark to a certain number. Such a watermark can be selected from a watermark pattern data bank.

In addition, dividing the entire image into 16 blocks is another limitation that forces one to limit the maximum number of bits that can be embedded into a block, while proving a certain level of stability. However, increasing the block size allows a larger number of watermark bits to be saved, but, on the other hand, one may face less stability, with respect to attacks such as clipping.

REFERENCES

1. Yaghmaee, F. and Jamzad, M. "A robust watermarking method for color images using Naive-Bayes classi-

fier", *IASTED Con. on Signal and Image Processing (SIP 2003)* (Aug. 2003).

2. Chen, B. "Design and analysis of digital water marking, information embedding and data hiding systems", PhD Thesis, MIT University, USA (June 2000).
3. Celik, M., Sharma, G., Saber, E. and Tekalp, A. "Hierarchical watermarking for secure image authentication with localization", *IEEE Transaction on Image Processing*, **11**(6), (June 2002).
4. Podilchuk, C.I. and Delp, E.J. "Digital watermarking: Algorithms and applications", *IEEE Signal Processing Magazine* (July 2003).
5. Shen, M., Zhang, X., Sun, L., Beadle, P.J. and Chan, F.H. "A method for digital image watermarking using ICA", *ICA Conference* (Apr. 2003).
6. Cox, I.J., Kilian, J., Leighton, T. and Shamoon, T. "Secure spread spectrum watermarking for images, audio and video", *IEEE Con. on Image Processing*, pp 243-246 (1996).
7. Cao, J., Fowler, J. and Younan, N. "An image-adaptive watermark based on redundant wavelet transform", *IEEE Con. on Image Processing*, pp 277-280 (Oct. 2001).
8. Dugad, R., Ratakonda, K. and Ahuja, N. "A new wavelet-based scheme for watermarking images", *Proc. IEEE Int. Conf. on Image Proc. (ICIP98)*, Chicago, IL, USA, pp 419-423 (1998).
9. Jamzad, M. and Yaghmaee, F. "Distribution of watermark according to image complexity for higher stability", *Proc. Int. Conf. Image Analysis and Recognition, ICIAR, Porto, Portugal* (Sept. 29- Oct. 1 2004).
10. Gonzales, R.C. and Woods, R.E. "Digital image processing", Addison Wesley, USA (2002).
11. Kutter, M., Jordan, F. and Bossen, F. "Digital watermarking of color image using amplitude modulation", *Journal of Electron Imaging*, **7**(2), pp 326-332 (1998).
12. Michel, T., *Machine Learning*, Prentice Hall (1997).
13. Franco, R. and Malah, D. "Adaptive image partitioning for fractal coding achieving designated rates under a complexity constraint", *IEEE 2001 International Conference on Image Processing* (2001).
14. Chandramouli, N., Memon, "How many pixels to watermark", *The International Conference on Information Technology: Coding and Computing (ITCC'00)*, Nevada, USA (2000).
15. Mobasseri, B.G. "A new quad tree complexity theorem", *Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on Pattern Recognition* (1992).
16. Knipe, J. and LI, X. "A new quad tree decomposition reconstruction method", *The International Conference on Pattern Recognition (ICPR1996)*.
17. Voloshynovsky, S., Herrigel, A. and Baum Gartner, N. "A stochastic approach to content adaptive digital image watermarking", *Lecture Notes in Computer Science*, **1768**, *Archive Proceedings of the Third International Workshop on Information Hiding* (1999).