Computing (2014) 96:951–973
DOI 10.1007/s00607-013-0357-6

Generalized integer transform based reversible watermarking algorithm using efficient location map encoding and adaptive thresholding

Chi-Man Pun · Ka-Cheng Choi

Received: 25 February 2013 / Accepted: 26 September 2013 / Published online: 11 October 2013

Abstract A novel algorithm that improves a generalized integer transform based reversible watermarking scheme is proposed in this paper. In our proposed algorithm, two main improvements have been achieved: adaptive thresholding and efficient location map encoding. With adaptive thresholding, suitable threshold t is selected adaptively, which ensures enough embedding capacity for the watermark while keeps the distortion introduced as low as possible. This modification is influential as an unsuitable threshold can lead to insufficient space for the watermark or even degrade the visual quality of the image. Moreover, efficient location map encoding helps in reducing the location map size, which down to 0.4 of the one unmodified in average. Therefore, more capacity is available for embedding as there is lesser overhead information. Overall, it provides more embedding capacity whereas improves the visual quality of the embedded image.

Keywords Generalized integer transform · Location map · Reversible watermarking · Threshold

Mathematics Subject Classification 62H35 · 68U10 · 94A08 · 68P25

1 Introduction

Recently, many researches have been carried out on the topic of digital watermarking. Digital watermarking is a process of embedding information into a digital signal, just like audio, pictures or video, which can be used for authentication, attachment of ownership or other descriptive information, or as means of covert communication. Many types of digital watermarking have been proposed so far, among these watermarking

C.-M. Pun (✉) · K.-C. Choi
Department of Computer and Information Science, University of Macau, Macau SAR, China
e-mail: cmpun@umac.mo

techniques, reversible watermarking attracts lots of attention in recent decades for its reversibility characteristic. Reversible watermarking is a type of digital watermarking that allows total recovery of the original image without any distortions after the hidden data is retrieved from the marked content. The characteristic that the signal can be restored is desirable, especially in the fields of law enforcement, medical and military image systems where even a small distortion is intolerable. For reversible watermarking, the original cover signal and marked signal are usually perceptually indistinguishable, and its target is to minimize the distortion introduced by the watermark in the embedding procedure while embed as much data as possible. Moreover, for full recovery proposes, information that helps in recovery is also embedded together with the watermark in embedding procedure. If the marked image is altered, this information will probably be ruined and the image cannot be restored to its original form, therefore this technique is fragile, i.e. it is not robust to any attacks or even transformation, restoration of the signal will fail even a slight modification has been made to the marked signal. For this reason, reversible watermarking is also used for tamper detection. The proposed reversible watermarking scheme is also one that is not robust to modification. Many valuable reversible watermarking algorithms have been published up to today, which can be classified into three categories [1]: data compression [2–7], difference expansion [8–13] and histogram modification [14–17]. Data compression is to losslessly compress the image to be overlaid to leave space for embedding watermark bits. As in [2], the host signal is quantized to obtain the residual, and the residual is then compressed to create capacity for the payload data. For data compression, the compression algorithm influences a lot, the better the compression, the higher the lossless-embedding capacity, due to this, it usually involves complex computation and has limited capacity [18], since it does not fit our purpose of large embedding capacity, no further study is carried out. Difference expansion (DE) is first proposed by Tian [8], many variants are then presented. Like other expansion embedding approaches, it makes use of some decorrelating operators to create features with small magnitudes, and the data embedding is done by expanding these features to create vacancies in which data bits are embedded. DE is an integer wavelet transform where the watermark bits are hidden in the expanded differences. It usually achieves a high embedding capacity and keeps distortion low [19], that's why it attracts a lot of attention including us. Among the variants of Tian's DE, Alattar [10] started extending it in a generalized manner by applying it to a triple or quad of pixels which increases the hiding ability and computation efficiency. Histogram modification usually utilizes the zero or minimum points of the image histogram and modifies the pixel values to leave space for bits to be embedded, and histogram shifting technique is often adopted together to prevent overflow and underflow. In [8], a binary tree structure is used in communicating pairs of peak points, and distribution of pixel differences is used to achieve large hiding capacity and low distortion. Nevertheless, the embedding capacity of histogram modification is usually not as high as difference expansion [20].

In this paper, an algorithm that improves a recently proposed generalized integer transform reversible watermarking method [21] which is based on Tian's DE is proposed, to remove the weakness of [21] that an influential threshold value needs to be preselected, and be able to adapt to the watermark to be embedded. With the proposed method, application of multi-level embedding and the threshold value are determined

according to the watermark size, which achieves the target that the whole watermark can be embedded with the least visual degradation. Moreover, the location map, which occupies a large proportion of the overhead, has been reduced in size in the proposed method, so more space is left for the watermark.

The rest of the paper is organized as follows. In Sect. 2, background information like the generalized reversible watermarking scheme based is briefly reviewed and the general concept of the proposed method is introduced shortly. In Sect. 3, the embedding procedure of the proposed method is introduced in detail. For the extraction procedure, it is described in Sect. 4. The experimental results and comparison with other methods are shown in Sect. 5, and finally, we conclude the paper in Sect. 6.

2 Background information

2.1 Related works

Tian [8] proposed a high capacity, low distortion reversible watermarking method by exploring the redundancy in the digital content. This approach divides the image into pairs of pixels, and then calculates the pixel differences for each pair, some of the pixel pairs that are expected not to cause overflow or underflow after embedding are selected for watermark embedding by DE. In Tian's DE, one bit is embedded into the expanded difference of the pixel pair. For the purpose of indicating the expanded pair locations, a location map is employed in this method, and afterwards, this technique of location map is widely applied by many algorithms. The location map as an overhead is compressed and embedded together with the watermark in the data embedding procedure.

In order to control the embedding capacity and distortion introduced, Tian used a predefined threshold, only those pixel pairs with absolute difference less than or equal to the threshold are expanded for watermark embedding. However, in case of small threshold value, its embedding capacity is usually low since most of the capacity is used up the location map due to its low compressibility.

Weng [13] utilized the invariability of the sum of pixel pairs to devise a novel reversible data embedding method. For each pixel pair, to preserve the sum of pixel pairs to be unchanged, whenever a certain value is added to one pixel, the same value is subtracted from the other pixel. In addition, to minimize the distortion introduced during data embedding, half the difference of the pixel pair plus the watermark bit is chosen as that certain value. Moreover, pairwise difference adjustment (PDA) is proposed in the algorithm to solve the problem in Tian's method, it significantly reduces the capacity consumed by overhead information through largely increases the compressibility of the location map, especially when the threshold value is small. The technique PDA uses is to form a considerable bias between the numbers of 1s and 0a in the location map, so that a higher compression ratio can be achieved, and the number of pixel pairs available for embedding increases.

Alattar [10] generalized Tian's DE to vectors of a triple or a quad of pixels, instead of pairs, this approach allows several bits to be embedded in the difference expansion in a single pass, so a higher embedding ability and more efficient computation can be obtained.

In [21], it suggests that Tian's DE [8] can be reformulated with marked pixel pair (x', y') as:

$$x' = 2x - \mathrm{a}(\mathbf{z}) + b, \quad y' = 2y - \mathrm{a}(\mathbf{z}), \tag{1}$$

where $\mathrm{a}(\mathbf{z})$ is the rounded average of vector $\mathbf{z}$, here $\mathbf{z} = (x, y)$, i.e. the pixel pair, and b is the watermark bit, the bit to be embedded into the pixel pair. Moreover, it also suggests that (1) can be generalized in a general form and applied to vector $\mathbf{x}$ with arbitrary length n,

$$\begin{cases} x'_1 = 2x_1 - 2\mathrm{f}(\mathrm{a}(\mathbf{x})) + b_1, \\ \vdots \\ x'_{n-1} = 2x_{n-1} - 2\mathrm{f}(\mathrm{a}(\mathbf{x})) + b_{n-1}, \\ x'_n = 2x_n - \mathrm{a}(\mathbf{x}). \end{cases} \tag{2}$$

$\mathrm{f}(z)$ is the ceiling of half of z. In the generalized form, there is a small modification, where $\mathrm{a}(\mathbf{x})$ is replaced by $2\mathrm{f}(\mathrm{a}(\mathbf{x}))$, with this, the recovery of the transform can be ensured and a payload-dependent location map can be built, since this minor difference gives the following important property which is also utilized in the proposed method to improve [21],

$$\mathrm{v_h}\left(\mathbf{x}'\right) = \mathrm{v}(\mathbf{x}) \tag{3}$$

where

$$\mathrm{v}(\mathbf{x}) = \sqrt{\sum_{i=1}^{n} (x_i - \mathrm{a}(\mathbf{x}))^2}, \quad \mathrm{v_h}(\mathbf{x}) = \mathrm{v}(\mathrm{h}(\mathbf{x})), \tag{4}$$

and $\mathrm{h}(x)$ is floor of half of x and $\mathrm{h}(\mathbf{x}) = (\mathrm{h}(x_1), \ldots, \mathrm{h}(x_n))$. With this generalized transform, given n pixels, we can embed $n - 1$ bits and return n watermarked pixels back.

In order to restore the original vector $\mathbf{x}$ and extract the watermark $\mathbf{b}$, the inverse of (2) is defined in [21] as follows,

$$x_i = \mathrm{h}\left(x'_i\right) + \mathrm{a}\left(\mathrm{h}\left(\mathbf{x}'\right)\right) + \mathrm{LSB}(x'_n), \quad b_j = \mathrm{LSB}(x'_j) \tag{5}$$

where $\mathrm{LSB}(x)$ is the least significant bit of x, $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n-1\}$.

With (2), an efficient embedding procedure is proposed in [21]. Image is firstly divided into non-overlapping blocks of length n, which can be categorized into embeddable (**E**), changeable (**C**) and others (**O**). Embeddable blocks are the blocks that have the property $\mathrm{v}(\mathbf{x}) \leq t$, for the rest of the blocks, they are divided into changeable and others by the characteristic that changeable block has $\mathrm{v_h}(\mathbf{x}) \leq t$. Location map for $\mathbf{E} \cup \mathbf{C}$ is then built, compressed and embedded into the embeddable and changeable blocks with (2) and least significant bit (LSB) replacement respectively, so as to preserve the property that $\mathrm{v_h}(\mathbf{x}') \leq t$ for $\mathbf{E} \cup \mathbf{C}$ blocks and $\mathrm{v_h}(\mathbf{x}') > t$ for **O** blocks.

With the property that $\mathbf{O}$ can be distinguished from $\mathbf{E} \cup \mathbf{C}$ through $v_h(\mathbf{x}')$, the size of the location map is greatly reduced, however, there are still rooms to be improved. Moreover, t is a threshold that needs to be preselected; unsuitable threshold may cause visual degradation or not enough capacity for the watermark W to be embedded. Therefore, carefully choosing t is an important task that needs to be done.

Finally, the bit string C_{LSB1}, which stores LSBs being replaced in the changeable blocks, and the watermark W are embedded into the remaining embeddable blocks $\mathbf{E}$ by (2). The reason that embeddable blocks which has the property $v(\mathbf{x}) \leq t$ are chosen for watermark embedding is because the difference between the original and embedded block is related to the block variation $v(\mathbf{x})$, $\| \mathbf{x}' - \mathbf{x} \| \approx v(\mathbf{x})$, therefore, choosing embeddable blocks which has property $v(\mathbf{x}) \leq t$ for embedding, distortion introduced can be minimized. The property $\| \mathbf{x}' - \mathbf{x} \| \approx v(\mathbf{x})$ in fact can be known from (2), as $\| \mathbf{x}' - \mathbf{x} \| \approx \| \mathbf{x} - a(\mathbf{x}) \| = v(\mathbf{x})$. With the variation $v(\mathbf{x})$ being minimized through the condition $v(\mathbf{x}) \leq t$ for embeddable blocks, difference between original and embedded images is also minimized, so the visual degradation to human eyes after watermarking can be guaranteed to be the minimal.

For many existing reversible watermarking algorithms, multi-level embedding is applied to increase its embedding capacity. In fact, multi-level embedding is a technique that repeats the embedding procedure again and again into an already embedded image, with the target that more information can be carried by the image.

2.2 General concept

In the next sections, the proposed method will be described in detail, for better illustration, definition of different location map are given in Table 1, and in Fig. 1a and b, the flowchart of the embedding and extraction procedure of the proposed method

Table 1 Definition for different location map

LM	Location map which records the states of all the blocks
LM′	Reduced-size location map, derived from LM in efficient location map encoding procedure, block states of $\mathbf{E} \cup \mathbf{C}$ blocks in LM are recorded in LM′ only when its $v_h(\mathbf{x}') \geq LB$
Compressed LM′	Derived from compressing LM′ by lossless Arithmetic encoder (Arith07)
L	Location map that distinguishes $\mathbf{E} \cup \mathbf{C}$ blocks from other blocks, it can only identify whether the block is $\mathbf{E} \cup \mathbf{C}$ block or other block, in case of $\mathbf{E} \cup \mathbf{C}$ blocks, it can not identify whether the block is embeddable or changeable
L′	Derived from L by identifying the block state (either embeddable or changeable) of the $\mathbf{E} \cup \mathbf{C}$ blocks that stores F_{LSB}

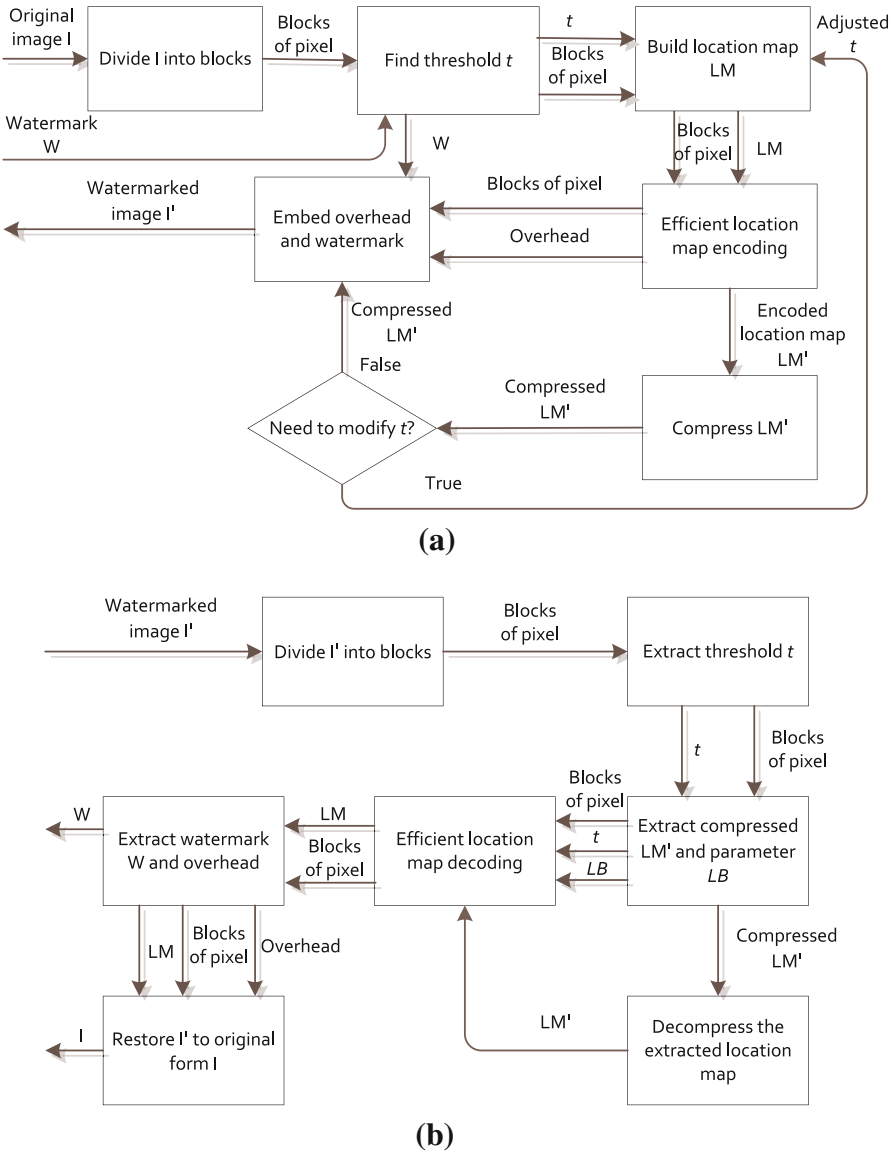


Fig. 1 Flowchart of **a** embedding procedure, **b** extraction procedure

are shown respectively. The embedding procedure of the proposed method improves [21] in two approaches mainly, further reducing the location map size and adaptive thresholding, other steps may refer to the original method. As shown in Fig. 1a, in order to perform the modification, as soon as the location map is built, the efficient location map encoding algorithm is implemented to reduce its size, and the threshold t instead of passing as a parameter, there is a process to adaptively selecting it. The following shows the algorithms of the proposed watermarking embedding and extraction procedures:

Algorithm 1. Watermarking embedding procedure

Input: original image I and watermark W
Output: watermarked image I'
Step1. Divide I into blocks of pixel.
Step2. Find the threshold t adaptive to W, details can be referred to ***Algorithm 3***.
Step3. With the just found t, build the location map LM which records the states of the blocks.
Step4. Efficient location map encoding is then applied to LM to reduce its size, and gives out LM', details are shown in ***Algorithm 4***.
Step5. Compress LM' to further reduce its size, the lossless compression algorithm used in the proposed method is Arith07 [22].
Step6. Check if enough capacity available for the overheads (including t and compressed LM') and W, modify t and back to **Step 3** if necessary.
Step7. Embed the overheads and W into I to produce I', where compressed LM' and some other overheads are embedded into $\mathbf{E} \cup \mathbf{C}$ blocks, t is embedded into the LSBs of the first 7 pixels of I, and W and the rest of the overheads are embedded into the embeddable blocks left over.

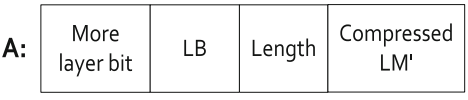
Algorithm 2. Watermarking extraction procedure

Input: watermarked image I'
Output: original image I and watermark W
Step1. Divide I' into blocks of pixel.
Step2. Extract the threshold t by reading the LSBs of the first 7 pixels of I'.
Step3. Extract the compressed, encoded location map, LM' and relative information by reading the LSBs of the $\mathbf{E} \cup \mathbf{C}$ blocks, i.e. blocks that satisfy$v_h(\mathbf{x}') \leq t$ where $v_h(\mathbf{x}')$ is defined in (4).
Step4. Decompress the just extracted location map with Arith07 [22] to give out LM'.
Step5. Apply the efficient location map decoding to LM' to restore the whole location map LM, details referred to ***Algorithm 5***.
Step6. Extract W and overhead that is needed in restoring I by reading the LSBs of embeddable blocks with the help of LM.
Step7. Restore I' to the original form I with the help of the overhead extracted.

3 Embedding procedure of proposed method

3.1 Overview

In Fig. 2 general visualization of the embedding procedure for given image is shown for the introduction of the proposed method. In embedding procedure, four strings of bits are built for embedding:



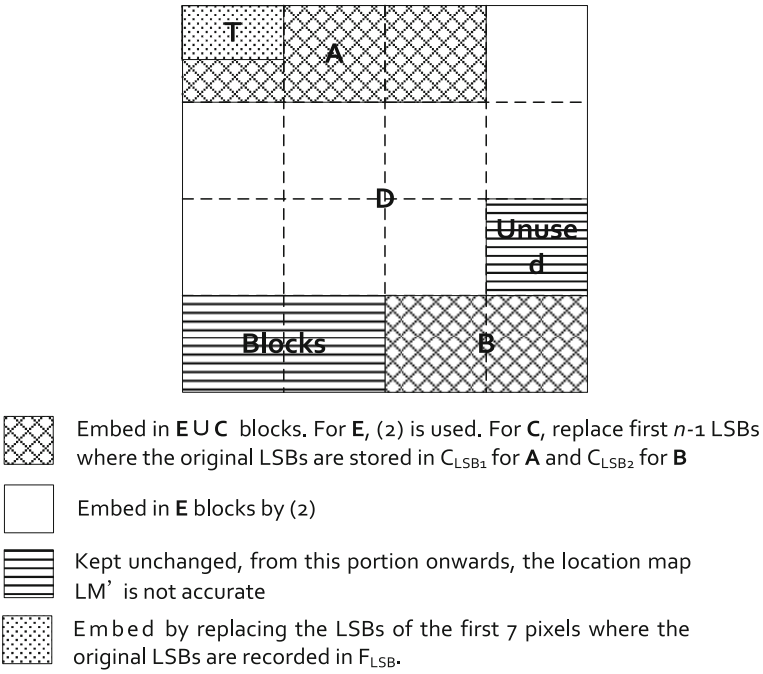
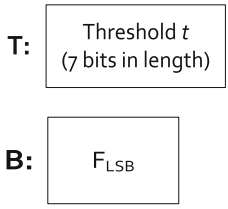
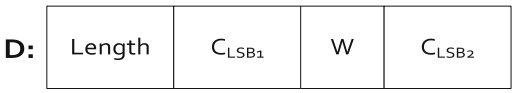


Fig. 2 Embedding in the given image

More layer bit: indicates whether it is the last layer to extract in multi-layer embedding.
LB: parameter needed for location map decoding.
Length: length of compressed LM′.
Compressed LM′: compressed encoded location map.



F_{LSB}: original LSBs of the first 7 pixels in the image before replaced by the threshold.



Length: total length of C_{LSB1}, W and C_{LSB2}.

C_{LSB1}: original LSBs of changeable blocks before replaced by **A**.
C_{LSB2}: original LSBs of changeable blocks before replaced by **B**.

With the length portion, we know the length of the component, so get exactly what we want, neither more nor less. In addition, it is suggested to add padding to the bit streams to be embedded to a multiple of $n - 1$, so that recovery of the image is more convenient as no tests for checking the end is needed when restoring a block, test is only carried out on every block, but not every pixel.

3.2 Adaptive thresholding

As mentioned before, correctly choosing the threshold (t) is crucial, since it may help to raise peak signal-to-noise ratio (PSNR), an objective evaluation standard for image quality, of the watermarked image. In order to do this, at first, the number of bits can be embedded for different values of t need to be known, so that enough space for the watermark (W) can be forecasted. Since t affects whether the blocks belong to embeddable group through the requirement that embeddable blocks need to have $v(\mathbf{x}) \leq t$, therefore, the larger t is, the more embeddable blocks are available for embedding, and more bits can be hidden in the image. However, as explained in [21], the distortion introduced to the embeddable block is close to its variance $v(\mathbf{x})$, so the value of t should not be too large so that the visual quality is guaranteed. To compromise between the embedding capacity and visual quality, a small t that can accommodate W should be chosen.

In fact, it is easy to find the embedding capacity of an image for different values of t, what is needed is to find the number of embeddable blocks for different t and multiplies it by $n - 1$, where n is the size of the block. However, in order to know the number of bits available for the watermark, the information that capacity used by the overhead is necessary. Actually, in the embedding procedure, the size of the overhead except the location map portion is fixed; therefore, the main task is to estimate the compressed encoded location map size (**LS**) as accurate as possible. The way to do this is to utilize the number of embeddable blocks (**ENO**) as a parameter to estimate **LS** as shown in (6), with different types of image, different constant is applied in the equation, and the range of the constant is from 0.45 to 1.5. The image with more patterns or more textured would have a larger constant value, since the result of efficient encoding and lossless compression of the location map for this type of image are not satisfactory comparatively, due to lesser embeddable blocks are available in textured image, which is caused by larger block variation. Therefore, we can know how textured an image by observing the embeddable block percentage, so the image with smaller embeddable block percentage will have a larger constant value in the equation. Since t is not determined in this state, the embeddable block percentage of the image is calculated with $t = 100$ for comparison. With **ENO** and estimated **LS**, bits available for the watermark (**CAP**) can roughly be estimated; finally, value for t can be suggested. Due to **LS** is estimated, t suggested is not accurate, it needs to be adjusted to increase its accuracy with (8), which depends on the range of t. The reason that t is adjusted in this way is because the efficient location map encoding algorithm performs better with increasing t, so more capacity than what have been estimated is left for the watermark

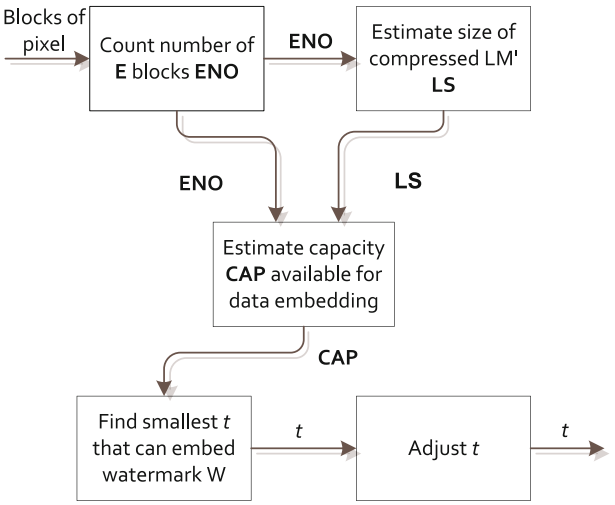


Fig. 3 Flowchart of adaptive threholding

when t is large enough, i.e. for large t, **LS** is overestimated, and actual **CAP** is more than estimated. In this case, a smaller t is usually efficient for W. Nevertheless, this is still a rough estimation, there can be cases that when the actual location map is built, find out that the estimated location map size is overestimated or underestimated, so modification may need to be made to adjust it. When the location map size is overestimated, reduces t by 1 until it finds the smallest t that is capable for W to be embedded, in contrast, when the size of location map is underestimated, repeats modification through increasing t by 1 until W can be wholly embedded. Normally, this can be done in a few trials as the first estimation for t is usually quite good. With this technique, t that can embed W and gives the best visual quality can be found. For better illustration, the flowchart of adaptive thresholding is shown in Fig. 3 and its algorithm is given below:

Algorithm 3. Adaptive thresholding

Input: blocks of pixel **PB**
Output: threshold t

Step1. In **PB**, find the blocks that will not cause overflow/underflow, and from these blocks, for different value of t, count number of blocks that satisfies the statement $\mathrm{v}(\mathbf{x}) \leq t$ to find the number of embeddable bocks **ENO**, where $\mathrm{v}(\mathbf{x})$ is the variance of the block and defined in (4).

Step2. With **ENO** and depends on the type of the given image, for all t, estimate the size of location map to be embedded **LS** through

$$\mathbf{LS} = \begin{cases} \mathbf{ENO} \times 0.45 & \textbf{if } \frac{\mathbf{ENO}}{number\_of\_blocks} \geq 0.85, \\ \mathbf{ENO} \times 0.75 & \textbf{if } 0.85 > \frac{\mathbf{ENO}}{number\_of\_blocks} \geq 0.70, \\ \mathbf{ENO} \times 1.5 & \textbf{if } \frac{\mathbf{ENO}}{number\_of\_blocks} < 0.70. \end{cases} \tag{6}$$

Step3. With **ENO** and **LS**, estimate the capacity available for the watermark **CAP** with

$$\mathbf{CAP} = \mathbf{ENO} \times (n-1) - \mathbf{LS} - other\_overhead. \tag{7}$$

Step4. Find the smallest t that is able to embed the given watermark by comparing the size of watermark with **CAP**.

Step5. Due to the fact that **LS** is estimated, adjustment needs to be made to t according to the range of t to increase its accuracy,

$$t = \begin{cases} t+1 & \textbf{if } 0 \geq t > 30, \\ t & \textbf{if } 30 \geq t > 40, \\ t-1 & \textbf{if } 40 \geq t > 70, \\ t-2 & \textbf{if } 70 \geq t > 100. \end{cases} \tag{8}$$

Finally, output the determined t.

Now as t is not a parameter anymore, but is computed in the embedding procedure, it is better to embed it into the image, as in multiple-layer embedding, t can be different values, and it would be confusing and inconvenient in passing this information. Since t is needed to distinguish $\mathbf{E} \cup \mathbf{C}$ blocks from $\mathbf{O}$ blocks, it should be embedded in a place where can still be extracted even a single information is not available. Here we choose to embed it by replacing the LSBs of the first 7 pixels of the image after the compressed LM$'$ is embedded, where the original 7 LSBs are stored in $\mathrm{F_{LSB}}$, and is then embedded into the $\mathbf{E} \cup \mathbf{C}$ blocks starting from the last block in the same way as embedding compressed LM$'$. In case of embedding into changeable blocks, the LSBs being replaced are recorded in $\mathrm{C_{LSB2}}$, which is then attached to the end of W and embedded with it. The reason that $\mathrm{F_{LSB}}$ is embedded in the this way just like compressed LM$'$ is because before we can extract compressed LM$'$, $\mathrm{F_{LSB}}$ is needed to restore the first 7 pixels of the image to the state that compressed LM$'$ is just embedded, so the only information available in this state is the value t which helps in identifying the $\mathbf{E} \cup \mathbf{C}$ blocks, just in the same situation as compressed LM$'$, this is why they have the same embedding method.

Now, with the proposed method, given W, according to its length, a suitable t that can fully accommodate W while keeps distortion low can be chosen. With experiments conducted, we found out that t should not be larger than 100, otherwise the PSNR value drops below 30, therefore, the range of t in our experiments are kept between 0 and 100.

In the case of very large watermark, there is opportunity that not enough space available for the bits to be completely embedded into the image for all the values of t. In this situation, multiple-layer embedding may be used which will be described in detail in later subsection.

3.3 Efficient location map encoding

In the proposed method, modification has been made to the location map before it is compressed and embedded to the image to further reduce its size, so that for the same PSNR value, more bits can be embedded to the image. With this modification, for "Airplane" (F-16), the encoded location map (LM′) is reduced down to 0.2 of the size of unmodified one in the best case and down to 0.4 in average. The unmodified location map originally records all the $\mathbf{E} \cup \mathbf{C}$ blocks, where all of them has the characteristic $v_h(\mathbf{x}') \leq t$, therefore, usually $v_h(\mathbf{x}')$ of both **E** and **C** blocks have the same or very near upper bounds, however, their lower bounds usually are not the same, which can be utilized in our proposed location map encoding procedure to reduce the location map size. As mentioned before, embeddable blocks have the feature $v_h(\mathbf{x}') = v(\mathbf{x})$, and for changeable blocks, $v_h(\mathbf{x})$ keeps unchanged, so we can easily get $v_h(\mathbf{x}')$ for $\mathbf{E} \cup \mathbf{C}$ blocks. With this information and the whole location map (LM), we can find LB, the lower bound of $v_h(\mathbf{x}')$ of changeable blocks which have been rounded down for storage reason, with this value being recorded, during extraction, any blocks of the watermarked image I′ with $v_h(\mathbf{x}') < LB$, we can ensure that it is embeddable, as $v_h(\mathbf{x}')$ of changeable blocks is always larger than or equal to LB, there is no need to record these blocks in LM′, therefore, in the location map encoding procedure, only those blocks with $v_h(\mathbf{x}') \geq LB$ need to be recorded so as to distinguish its identity in the extraction procedure. The value LB must be embedded together with compressed LM′ in order to recover LM for the image; it should never be embedded with the watermark, as the retrieval of the watermark needs the help of LM. In Fig. 4a, it illustrates this efficient location map encoding procedure in detail and the algorithm is given below.

Algorithm 4. Efficient location map encoding

Input: Pixel blocks **PB** and location map *LM*
Output: reduced-size location map *LM*′

Step1. Calculate the expected $v_h(\mathbf{x}')$ of **PB** with the help of *LM* where **E** blocks has $v_h(\mathbf{x}') = v(\mathbf{x})$ and **C** blocks has $v_h(\mathbf{x}') = v_h(\mathbf{x})$, in which $v(\mathbf{x})$ and $v_h(\mathbf{x})$ are defined in (4) and $\mathbf{x}'$ is the watermarked block.

Step2. Find *LB* where $LB = \lfloor lower\ bound \rfloor$, *lower bound* is the smallest $v_h(\mathbf{x}')$ of changeable blocks in *LM*.

Step3. Record the *LM* state of the $\mathbf{E} \cup \mathbf{C}$ blocks that has $v_h(\mathbf{x})' \geq LB$ to construct *LM*′.

With this technique, the location map size is further reduced, however, in case of W is not long enough to use up all the embeddable blocks, it causes problem. As for visual quality reason, the embeddable blocks left over are kept unchanged, however, in building LM′, for all embeddable blocks, we assume bits are embedded into them and $v_h(\mathbf{x}') = v(\mathbf{x})$, but now some of them (the embeddable blocks left behind) are kept unchanged which has $v_h(\mathbf{x}') = v_h(\mathbf{x})$ rather, so the treat to record $\mathbf{E} \cup \mathbf{C}$ block state of LM only when $v_h(\mathbf{x}') \geq LB$ will cause error starting from the part that embeddable blocks are kept unchanged, i.e. after W is fully embedded, LM′ afterward

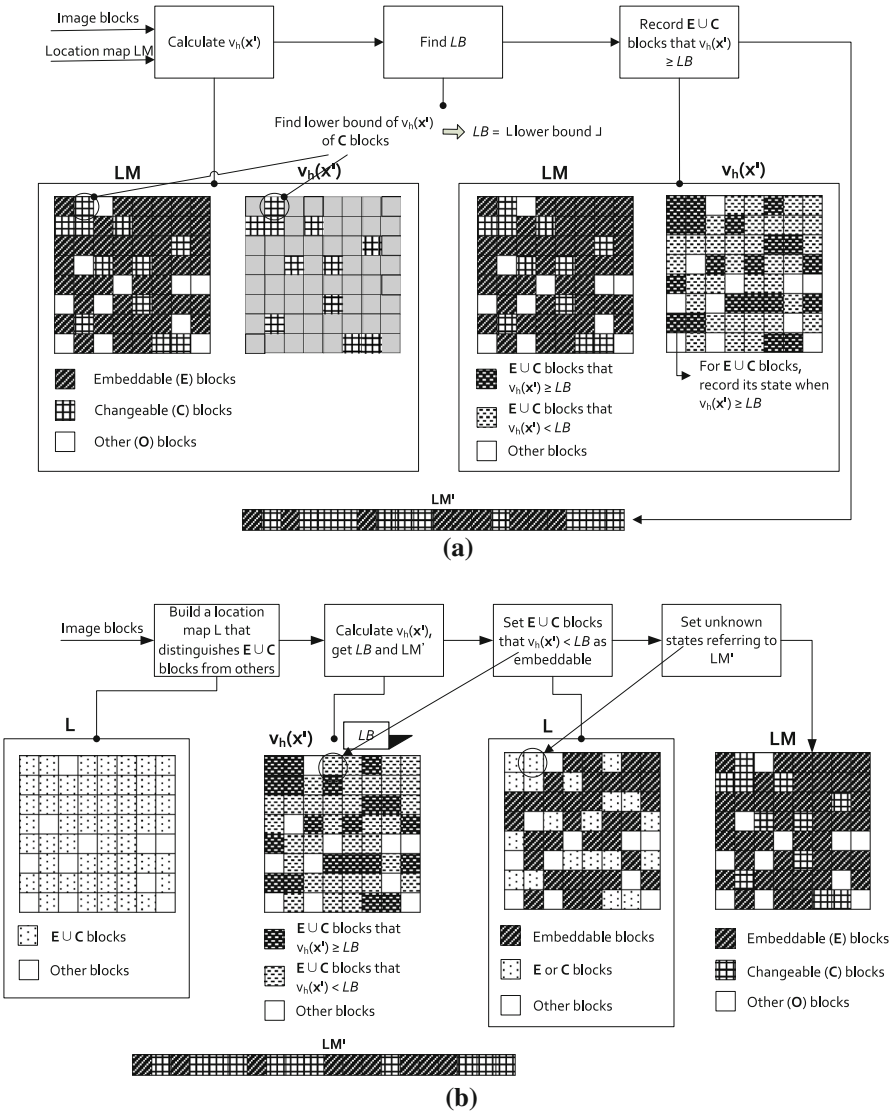


Fig. 4 **a** Efficient location map encoding procedure. **b** Efficient location map decoding procedure

is not accurate, this is because blocks that need to be recorded may have not been recorded, blocks that do not need to be recorded may have been recorded. Luckily, that part of the location map is not so important; the only block states needed are those of the blocks that are used to embed F_{LSB}, so that they can be restored to its original form. To record those states, instead of $n-1$ bits are embedded each block, here only $n-2$ bits can be embedded, where the first bit to be embedded to the block is used to record its state, by this way, LM′ can be corrected to have right block states in necessary part.

3.4 Multi-level embedding

Multi-level embedding is a technique that repeats the embedding procedure once again into an already embedded image, so that more information can be carried by the image. In multi-level embedding, instead of embedding the watermark into the image in a single pass, the watermark is partitioned into segments, with one segment being embedded in each pass, and repeats the embedding procedure until the whole watermark is embedded.

Multiple embedding can also be applied to the proposed method, and it is encouraging to be used together. As with multiple layers, usually more bits can be embedded into the same image, and with the same number of bits to be embedded, the visual quality of the one using multiple layers is usually better. The reason is that instead of embedding into the blocks that will cause more distortion, multi-level will choose to embed twice in blocks that cause less distortion. This can be done through the following steps:

- When the embedding capacity is not big enough even t has been set to 100, multiple embedding is used.
- In order to identify whether multiple embedding is being used, header information "more layer bit" is embedded before the location map. The bit '0' represents it is the first embedding layer or multiple-layers has not been used, generally speaking, this is the last layer to be extracted, the bit '1' is the opposite. For the watermark, it is embedded as long as possible; the remaining portion of the watermark will be embedded in the next round/layer. This procedure will continue until the watermark is fully embedded.
- For the extraction procedure, extraction continues until the header information that records the multi-level state indicates that it is the last layer need to be extracted. Finally, the watermark is concatenated with the newest extracted watermark portion attached at the front.

With this method, given a large watermark, the embedding procedure would choose multiple embedding rather than a large t that use single embedding. As described before, a large t usually provides more blocks for embedding which results a larger capacity, however, blocks introduce more distortion would also be used, so large t should be prevented. Therefore, it is natural to have the decision to use multi-level embedding which uses smaller thresholds instead, with the principle to embed multiple times in blocks introduce less distortion.

4 Extraction procedure of proposed method

At first, watermarked image $(I)'$ is divided into blocks in the same way as embedding and t is extracted which is needed to distinguish $\mathbf{E} \cup \mathbf{C}$ blocks from $\mathbf{O}$ blocks, the way that it is extracted is to get the LSBs of the first 7 pixels of the image. With t, location map (L) that distinguishes $\mathbf{E} \cup \mathbf{C}$ and $\mathbf{O}$ can be built, followed by, we need to extract the original LSBs of those 7 pixels, F_{LSB} (String $\mathbf{B}$), in order to restore those blocks to the state that the compressed LM' is just embedded into them. As those bits are embedded in the $\mathbf{E} \cup \mathbf{C}$ blocks starting from the end, therefore, what have to do is continue to

extract the first $n-1$ LSBs of $\mathbf{E} \cup \mathbf{C}$ blocks starting from the end which can be known from L until that 7 LSBs are got. Notice that, here the first bit of the extracted LSBs of every block is its state, indicates that it is embeddable or changeable, which is then used to modify the location map L to L′ by replacing with the respective states got from previous step. The 7 LSBs extracted just now are then used to restore the LSBs of the first 7 pixels.

Now, the efficient location map decoding procedure can be done, first extract the compressed LM′ and related information (String **A**), the way that it is extracted is the same as [21], the difference is that a few bits extracted is the overhead including *LB*. Compressed LM′ is then decompressed, then with LM′, *LB* and L′ built previously, LM that identifies the embeddable blocks can be built as shown in Fig. 4b. To change L′ to LM, i.e. location map that only identifies $\mathbf{E} \cup \mathbf{C}$ to distinguish **E**, the only thing need to do is to check the state of every $\mathbf{E} \cup \mathbf{C}$ blocks to see if it is embeddable. This is done with the help of *LB* and LM′, for every $\mathbf{E} \cup \mathbf{C}$ blocks that with $v_h(\mathbf{x}') < LB$, where $\mathbf{x}'$ is a block vector of I′, set its state as embeddable. At last, for the $\mathbf{E} \cup \mathbf{C}$ blocks with unknown state, just change its state referring to LM′. Below shows the algorithm of this procedure:

Algorithm 5. Efficient location map decoding

Input: Pixel blocks **PB**

Output: location map *LM*

Step1. Build a location map *L* for **PB** that distinguishes $\mathbf{E} \cup \mathbf{C}$ blocks from other blocks by $v_h(\mathbf{x}') \leq t$, where t is extracted from the first 7 LSBs of **PB** and $v_h(\mathbf{x}')$ is defined in (4).

Step2. Extract the reduced-size location map *LM*′ together with *LB* from **PB**.

Step3. Set $\mathbf{E} \cup \mathbf{C}$ blocks with $v_h(\mathbf{x}') < LB$ in *L* as embeddable.

Step4. Set state of unknown blocks in *L* correspond to *LM*′ to build the whole location map *LM*.

Then, extract the watermark portion (String **D**) in the same way as mentioned in [21], C_{LSB1} and C_{LSB2} are extracted altogether with the watermark, and the block index of the end of string **D** is recorded. Finally, what is left is to restore the image to its original form by (5) as described in [21] until it reaches the block index recorded before, as the blocks afterward are unchanged, so nothing have to be done, except the blocks that are used to embed F_{LSB} (String **B**) which is needed to be recovered, it is recovered using the same way as recovering the blocks that are used to embed the compressed LM′, for changeable blocks, the LSBs used in recovering are referred to C_{LSB2} which is attached at end of W. Therefore, after recovery, the attached LSBs C_{LSB1} and C_{LSB2} will be removed, and the extracted string **D** will leave over with the watermark only which is what is needed.

5 Experimental results

As explained in [21], 4×4 ($n = 16$) is the size of the blocks that gets the best performance, therefore, 4×4 sized blocks will be used in the experiments conducted. Moreover, four 512×512 grayscale standard images which are shown in Fig. 5 are used in our experiments.

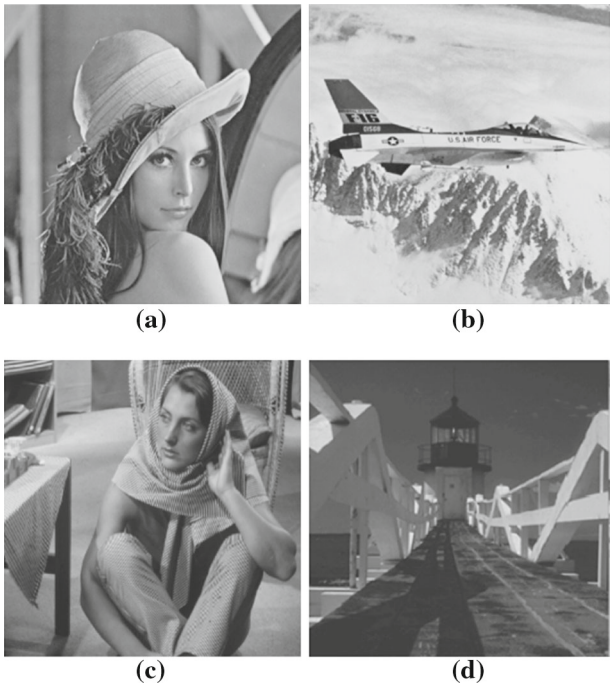


Fig. 5 Test images from Computer Vision Group (http://decsai.ugr.es/cvg/index2.php). **a** Lena, **b** Airplane, **c** Barbara and **d** Lighthouse

In the proposed method, the size of the location map has further been reduced; Fig. 6 shows the location map size of the original method and the proposed method of the four standard images for comparison, as we can see, the size reduced is quite significant and satisfactory. The location map size is reduced obviously and dramatically from $t \approx 10$ onwards, i.e. when the data to be embedded is not too small, the effect of this modification is quite effective. In the best case, the reduced location map is just around one-eighth of the original size; this reduction of the overhead provides more capacity for embedding the payload, therefore, allows larger embedding capacity with the same PSNR value.

In order to test the effect of adaptive thresholding, a watermark with 210,000 bits is embedded into the given image, "Lena" and "Lighthouse" are used for this experiment. By the original method, as threshold t is not adaptively selected and is needed to be preselected, let's try $t = 30$, in case of "Lena", it is only capable to embed 166 230 bits, there is not enough space for the whole watermark, so the performance of PSNR value is not applicable, and we need to choose a larger number, let $t = 70$, in this case, whole watermark can be embedded with 32.6885 dB. With proposed method, adaptive threshold is used, 63 is determined to be the threshold, where whole watermark can be embedded with better PNSR 33.3402 dB. Moreover, the capacity available for watermark embedding is also closer to the actual watermark size with the proposed method. In case of the original method, the number of embeddable blocks is much more than needed comparatively, which is due to unsuitable higher threshold

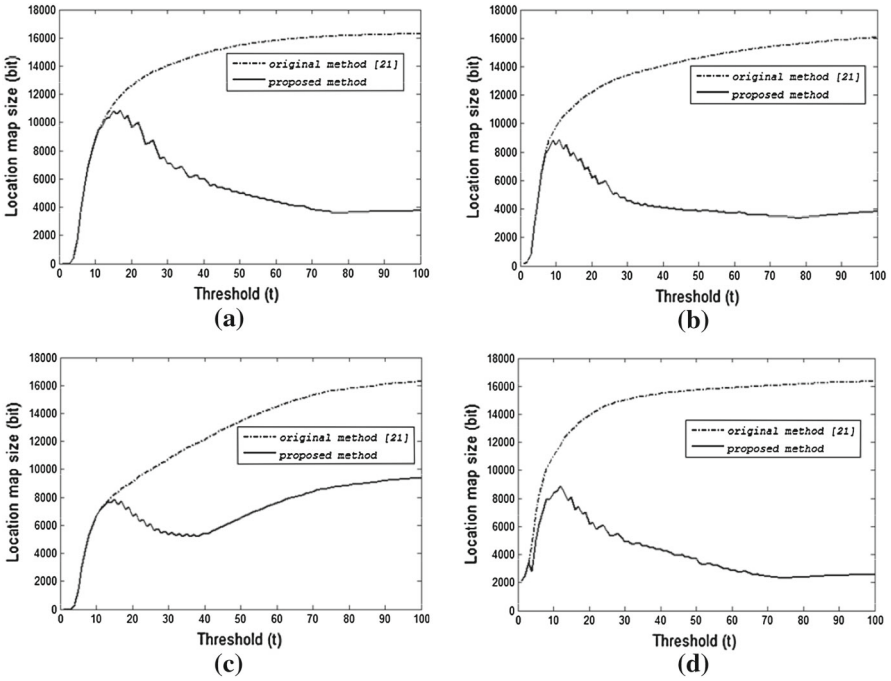


Fig. 6 Comparison of location map size between original and proposed modified method of **a** Lena, **b** Airplane, **c** Barbara and **d** Lighthouse

Table 2 Effects of adaptive thresholding of the proposed method

	Results of embedding 210,000 bits watermark into: (*i*) Lena			(ii) Lighthouse		
	Original method		Proposed method	Original method		Proposed method
t	30	70	63	30	70	42
Capacity (bits)	166,230	212,550	210,132	192,555	226,395	210,552
PSNR	N/A	32.6885	33.3402	N/A	34.6019	36.9687

value t with the condition $v(\mathbf{x}) \leq t$ as the embeddable blocks. Therefore, in the original method, blocks that with larger variation may be chosen for embedding instead, and the one with smaller variation may leave behind, and causes more distortion. In case of "Lighthouse", the same situation happens, when $t = 30$, only 192 555 bits can be embedded, when $t = 70$, whole watermark is embedded with 34.6019 dB, whereas with proposed method, better PSNR 36.9687 dB is got with $t = 42$. For better visualization, the above results are summarized in Table 2.

With suitable threshold being determined in adaptive thresholding algorithm, experiment about the additional time consumed by this algorithm is also conducted. To measure the time consumed by adaptive thresholding algorithm in the watermark embedding procedure, we use the tic and toc command in MATLAB 2009b on a com-

puter with a 2.0-GHz Intel(R) Core(TM) 2 Duo CPU and 4 GB memory to carry out this experiment. For the four test images, in average, 3.5 s are used for the algorithm, rounding to one decimal place.

Moreover, as the proposed method adaptively use multiple-layered embedding, better visual results can be achieved which can be shown by embedding same payload (220,000 bits) into the given image, "Airplane" and "Barbara" are used here, by the algorithm with and without multiple-layered embedding. The resulting information is listed in Table 3. By the original method, for "Airplane", with a larger preselected threshold 125, whole watermark can be embedded with 29.7973dB in a single embedding, however, with proposed method, as multiple-layered embedding is used, the payload is embedded in double embedding instead with smaller thresholds 100 and 6 in the two passes respectively, therefore, less distortion is introduced in the resulting image which has a higher PNSR value 31.2247 dB. For "Barbara", similar results are obtained which can refer to Table 3.

As distortion (differences between marked image and original image) introduced to the watermarked image is proportional to v(**x**) of the embeddable blocks (blocks used for embedding), for better visual results, blocks with small v(**x**) should be chosen as the embeddable blocks, with the requirement that embeddable blocks have v(**x**) $\leq t$, smaller threshold probably introduces less distortion. With multi-layer embedding algorithm, the watermark is partitioned into two or more segments, and only one segment is embedded into the image in each pass, the next watermark segment is embedded again into the already embedded image in next pass until whole watermark is embedded. By this way, the whole watermark is embedded into the image by multiple times of embedding, and in each pass, the bits need to be embedded is lesser comparing with the whole watermark. Therefore, smaller thresholds are enough to accommodate the watermark segments in the image, so less distortion is introduced. The reason that smaller threshold introduces less distortion is due to small threshold only embeds data to blocks that will cause smaller differences comparatively. With large threshold, data is embedded to blocks even it knows the embedding will cause large differences from the original blocks. In multiple-layered embedding, data embedding can be carried out on the blocks with small v(**x**) in second layer and so on instead of the blocks with large v(**x**) in a single layer. Since there are blocks which have smaller v(**x**) than other blocks even they have been chosen as embeddable blocks in the previous layer, therefore, with multi-layer algorithm, watermark bits can repeatedly embed to the blocks that have smaller v(**x**), and the marked image will have

Table 3 Effects of adaptive multiple-layered embedding

	Results of embedding 220,000 bits payload into: (*i*) Airplane		(ii) Barbara	
	Original method	Proposed method	Original method	Proposed method
Embedding	Single-layered	Double-layered	Single-layered	Double-layered
t	125	100/6	160	100/17
PSNR	29.7973	31.2247	25.6873	28.5157

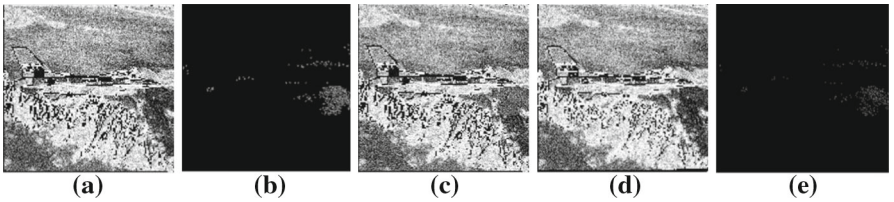


Fig. 7 Embeddable blocks (shown in *white dots*) in multi-layered and single-layered embedding for Airplane. **a** First layer, **b** Second layer in multi-layered algorithm, **c** $(a)^{\vee}(b)$, **d** Single-layered algorithm and **e** $(a)^{\wedge}(b)$

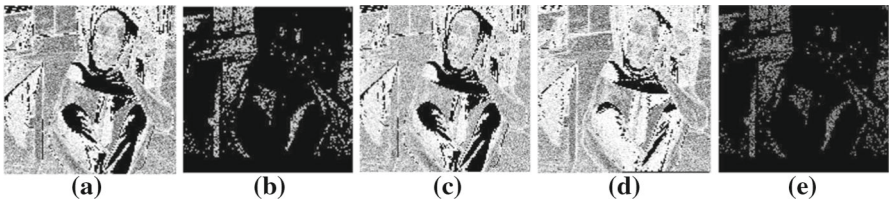


Fig. 8 Embeddable blocks (shown in *white dots*) in mulit-layered and single-layered embedding for Barbara. **a** First layer, **b** second layer in multi-layered algorithm, **c** $(a)^{\vee}(b)$, **d** single-layered algorithm and **e** $(a)^{\wedge}(b)$

smaller differences from the original image and introduces lesser distortion. While in single-layer algorithm, data embedding can only be executed in a single pass, once for every block. Therefore, a larger threshold which causes more distortion is needed to accommodate the watermark. Thus, in single layer embedding, blocks that have larger v(**x**) will also be chosen for embedding if necessary, and causes more distortion than multi-layer algorithm, as multiple small distortions is better than one large distortion that distorts the image seriously. For better clarification, embeddable blocks chosen for watermark embedding for each layer in multi-layered and single-layered algorithm are shown in Figs. 7 and 8. In the figures, logical conjunction and logical disjunction of the embeddable block locations in multi-layered algorithm are also given, so that locations selected between single-layered and multi-layered embedding can be compared, and locations that have been reselected in multi-layered algorithm can be seen clearly. As shown in the figures, in multi-layered algorithm, many blocks have been chosen repeatedly for embedding, while in single-layered algorithm, most of the blocks including the ones with large v(**x**) are chosen as the embeddable blocks.

Followed by, the performance of the original method, proposed method and other three algorithms: (1) Tian's method, (2) Alattar's method and (3) Weng's method are compared, in Fig. 9, the embedding capacity versus image visual quality data for these methods of "Lena" and "Airplane" are shown, for the original method, the preselected threshold is set to 70. From the figure, we can see that our method is better than all the other methods. This is due to generalization is applied in data embedding which increases the hiding ability and the computation efficiency of the algorithm. Moreover, the encoded location map in the proposed method is much smaller than the location map in other methods, so more

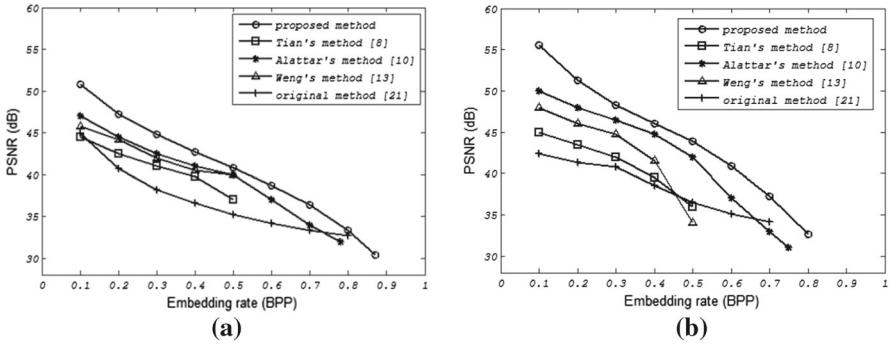


Fig. 9 Performance comparison of our method with original and other methods for **a** Lena and **b** Airplane

embedding capacity is available for the watermark, Furthermore, in comparison with the original method, we can see that unsuitable threshold affects the performance significantly, with adaptive thresholding in the proposed method, least distortion can be ensured, and this is why the proposed method outperforms the other methods.

Finally, the embedded images of different test images by the proposed method having payloads of 131,072 bits with 0.5 bit per pixel (bpp), 262,144 bits with 1.0 bpp, and 512,000 bits with 1.95 bpp are given in Fig. 10, detailed information of the embedded images is listed in Table 4. As shown in the figure, there is not much difference between the original and watermarked images to human eyes; and for images with a high embedding rate of 1.95 bpp, the image quality is still quite good. The result is as expected, since reversible watermarking algorithm has the characteristic that no significant difference between the original and marked images, human being usually can not distinguish whether the given image is watermarked or not. Regardless of the watermark size, details of the images usually are preserved quite well after reversible watermarking, however, if the watermark to be embedded is too large, blocking effect will appear.

In Table 4, detailed information including the embedding rate, the embedding type, adaptively selected thresholds and PSNR value of the embedded image are shown. For the thresholds, in case of multi-layered embedding, threshold of each layer are given, in the format of 1st layer/2nd layer/and so on. From the results, we can see that adaptive multi-layered algorithm will be applied when the watermark size increases, so that low variation blocks can be reused for embedding repeatedly to reduce distortion introduced. Overall, the results are quite satisfactory; PSNR values of the embedded images are quite high, even for the images with high embedding rate of 1.95 bpp, PSNR values are still greater than 23 dB.

Besides the objective evaluation experiments, subjective evaluation have also been carried out on the embedded images shown in Fig. 10. These images are presented to the testers, and every tester is asked to tell whether the given image carries watermark. For embedded images with embedding rate of 0.5 bpp and 1.0 bpp, most testers tell that there is no watermark, which indicates that only images with high embedding rate can be identified by human eyes. Followed by, the original images are given to

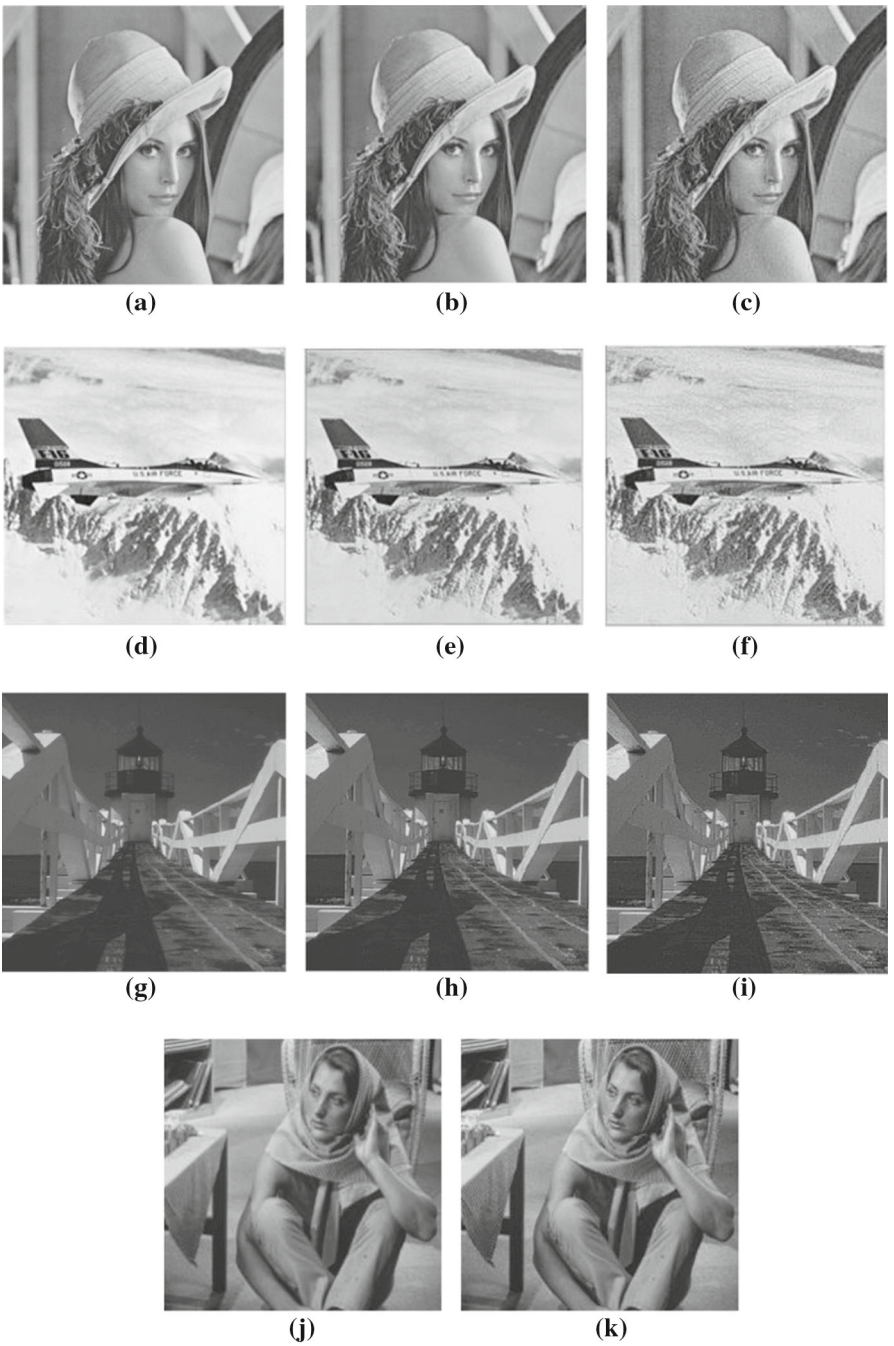


Fig. 10 Embedded images. **a–c** Lena (0.5, 1.0 and 1.95 bpp), **d–f** Airplane (0.5, 1.0 and 1.95 bpp), **g–i** Lighthouse (0.5, 1.0 and 1.95 bpp) and **i–k** Barbara (0.5 and 1.0 bpp)

Table 4 Results of the proposed method

(i) Lena	Payload size	131,072 bits (0.5 bbp)	262,144 bits (1.0 bbp)	512,000 bits (1.95 bbp)
	Embedding	Single-layered	Double-layered	Triple-layered
	t	20	100/18	100/100/56
	PSNR	40.80	30.10	23.17
(ii) Airplane	Payload size	131 072 bits (0.5 bbp)	262,144 bits (1.0 bbp)	512,000 bits (1.95 bbp)
	Embedding	Single-layered	Double-layered	Triple-layered
	t	15	100/13	100/100/47
	PSNR	43.84	31.03	24.53
(iii) Lighthouse	Payload size	131 072 bits (0.5 bbp)	262 144 bits (1.0 bbp)	512,000 bits (1.95 bbp)
	Embedding	Single-layered	Double-layered	Triple-layered
	t	12	100/4	100/100/25
	PSNR	45.68	32.63	26.18
(iv) Barbara	Payload size	131,072 bits (0.5 bbp)	262,144 bits (1.0 bbp)	512,000 bits (1.95 bbp)
	Embedding	Single-layered	Double-layered	N/A
	t	37	100/26	N/A
	PSNR	38.20	28.08	N/A

the testers for comparison with the embedded images. For embedded images with embedding rate of 0.5 bpp, most testers tell that there is no difference between the original and embedded images. With embedding rate of 1.0 bpp, only some testers notice the difference, and for high embedding rate of 1.95 bpp, due to obvious blocking effect, all testers identify the difference. Overall, human eyes mostly can not identify the differences between the original and marked images if the embedding rate is not too high.

6 Conclusion

In this paper, we proposed an algorithm that improves a recently proposed generalized integer transform reversible watermarking scheme in two aspects. One is an improved distortion control through adaptively chosen threshold in the algorithm. The other is further reducing the size of the location map to be embedded, thus less capacity is used for storing overhead so as to increase its embedding capacity. Overall, it provides high embedding capacity whereas maintains good visual quality for the embedded image. Experimental results also indicate that the proposed method has improved performance compared with the existing methods.

Acknowledgments The authors would like to thank the referees for their valuable comments. This research was supported in part by the Research Committee of the University of Macau and the Science and Technology Development Fund of Macau SAR (Project No. 034/2010/A2).

References

1. Agrawal R, Srikant R (2000) Privacy-preserving data mining. SIGMOD Rec 29:439–450
2. Fridrich J et al (2001) Invertible authentication watermark for JPEG images. In: Proceedings of the international conference on information technology: coding and computing, 2001, pp 223–227
3. Fridrich J et al (2002) Lossless data embedding-new paradigm in digital watermarking. EURASIP J Appl Signal Process 2002:185–196
4. Kalker T, Willems FMJ (2002) Capacity bounds and constructions for reversible data-hiding. In: 14th international conference on digital signal processing, 2002. DSP 2002, vol 1, pp 71–76
5. Kamstra et al (2004) Wavelet techniques for reversible data embedding into images. Centrum voor Wiskunde en Informatica, Amsterdam, PAYS-BAS
6. Celik MU et al (2005) Lossless generalized-LSB data embedding. In: IEEE transactions on image processing, vol 14, pp 253–266
7. Kamstra L, Heijmans HJAM (2005) Reversible data embedding into images using wavelet techniques and sorting. In: IEEE transactions on image processing, vol 14, pp 2082–2090
8. Jun T (2003) Reversible data embedding using a difference expansion. In: IEEE transactions on circuits and systems for video technology, vol 13, pp 890–896
9. Alattar AM (2003) Reversible watermark using difference expansion of triplets. In: Proceedings of international conference on image processing, ICIP 2003, vol 1, pp I-501–504
10. Alattar AM (2004) Reversible watermark using the difference expansion of a generalized integer transform. In: IEEE transactions on image processing, vol 13, pp 1147–1156
11. Alattar AM (2004) Reversible watermark using difference expansion of quads. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing, 2004 (ICASSP '04), vol 3, pp iii-377–380
12. Thodi DM, Rodriguez JJ (2007) Expansion embedding techniques for reversible watermarking. In: IEEE transactions on image processing, vol 16, pp 721–730
13. Shaowei W et al (2008) Reversible watermarking based on invariability and adjustment on pixel pairs. IEEE Signal Process Lett 15:721–724
14. Zhicheng N et al (2006) Reversible data hiding. In: IEEE transactions on circuits and systems for video technology, vol 16, pp 354–362
15. Xuan GS, Yao YQ, Ni Q, Yang Z, Gao J, Chai P (2006) Lossless data hiding using histogram shifting method based on integer wavelets. In: Presented at the LNCS. International workshop digital watermarking, Jeju Island, Korea
16. Fallahpour M, Sedaaghi MH (2007) High capacity lossless data hiding based on histogram modification. In: Presented at the IEICE electron express
17. Wei-Liang T et al (2009) Reversible data hiding based on histogram modification of pixel differences. In: IEEE transactions on circuits and systems for video technology, vol 19, pp 906–910
18. Lixin L et al (2010) Reversible image watermarking using interpolation technique. In: IEEE transactions on information forensics and security, vol 5, pp 187–193
19. Shi YQ et al (2004) Lossless data hiding: fundamentals, algorithms and applications. In: Proceedings of the 2004 international symposium on circuits and systems, 2004. ISCAS '04, vol 2, pp II-33–36
20. Bin Z et al (2010) A near reversible image watermarking algorithm. In: International conference on machine learning and cybernetics (ICMLC), vol 2010, pp 2824–2828
21. Xiang W et al (2010) Efficient generalized integer transform for reversible watermarking. IEEE Signal Process Lett 17:567–570
22. Skretting K et al (1999) Improved Huffman coding using recursive splitting. In: NORSIG 1999. Proceedings of Norwegian signal processing symposium, Asker, Norway