J. Vis. Commun. Image R. 41 (2016) 185–199

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Reversible data hiding based on flexible block-partition and adaptive block-modification strategy ☆

Shaowei Weng [a,*], Yijun Liu [a], Jeng-Shyang Pan [b], Nian Cai [a]

[a] School of Information Engineering, Guangdong University of Technology, PR China
[b] Fujian Provincial Key Laboratory of Data Mining and Applications, Fujian University of Technology, Fujian, PR China

ARTICLE INFO

Article history:
Received 15 March 2016
Revised 12 September 2016
Accepted 28 September 2016
Available online 4 October 2016

Keywords:
Reversible data hiding
Flexible block-partition
Adaptive pixel-modification strategy
Two-layer embedding

ABSTRACT

A novel reversible data hiding (RDH) method based on flexible block-partition and adaptive pixel-modification strategy is proposed in this paper, which is implemented from the following two aspects. One aims at partitioning flexibly a smooth block into non-overlapped sub-blocks of arbitrary size according to a local complexity measurement. After partition, since each resulting sub-block can be treated as an independent embedding unit, this block can be embedded with more data bits. The other is that the different pixel modification method is utilized for blocks (or sub-blocks) of different levels. Specifically, for a block, if it is not suitable for further division, only the maximum and minimum are modified at the same time so as to keep the distortion as low as possible. If it is divided into smaller sub-blocks, two pixel-modification schemes are designed for sub-blocks of size 1×3 and other sizes, respectively. One is that a 1×3 block can be embedded with 2 data bits by only modifying the maximum of this block. In this way, the embedding distortion is further decreased. The other is that in order to exploit better redundancy, two largest and two smallest pixels are modified simultaneously so that at most 4 data bits are embedded into a sub-block. Extensive experiments verify that the proposed method outperforms Peng et al.'s, Wang et al.'s, Li et al.'s, Sachnev et al.'s and Hong et al.'s works.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

As the essential of steganography and watermarking, data hiding is a technique that hides secret message into a multimedia carrier, e.g., image, audio, or video for various applications including copyright protection, content authentication, and media asset management. In fact, steganography focuses on undetectability of hidden data, and its counterpart is digital steganalysis which is used for detecting hidden data [1–3], while digital watermarking aims at improving the robustness against various attacks [4–6].

Reversible data hiding is a kind of data hiding technique with reversibility. Reversibility means that the original content can be losslessly recovered after the extraction of hidden data [7]. RDH is an ideal solution for applications, such as remote sensing, military image processing, medical image sharing and multimedia archive management [8] where any permanent distortion to the original content is unacceptable.

The early development of RDH methods (e.g., the method proposed by [9]) were proposed for fragile authentication, and thus, the obtained embedding capacity (EC) is limited. Later on, a large amount of research is done to improve EC. According to the used techniques in RDH, the existing RDH methods can be mainly classified into three classes, i.e., RDH based on lossless compression [10–12], RDH using DE (short for difference expansion), and RDH using HS (short for histogram shifting) [13–15].

RDH using DE was firstly presented by Tian [16]. In his method, the integer Haar wavelet transform is applied to every two neighboring pixels to calculate their difference value and average value of these two pixels. The average value is used to ensure reversibility. The difference value is expanded (i.e., multiplied by 2) to create a vacant least significant bit (LSB), and 1-bit data is appended to this LSB, which is called DE. Therefore, the obtained EC approaches to 0.5 bpp (bit per pixel). A location map is created to record the locations of the pixels which may cause overflows or underflows. The map is compressed losslessly, and the compressed map is embedded into the carrier along with the payload. As we know, the increase of the compressibility helps to the improvement of EC. To this end, Kamstra et al. [12] and Kim et al. [17] designed different embedding strategies to increase the compressibility.

☆ This paper has been recommended for acceptance by M.T. Sun.
* Corresponding author.
E-mail addresses: wswweiwei@126.com (S. Weng), jspan@cc.kuas.edu.tw (J.-S. Pan), cainian@gdut.edu.cn (N. Cai).

http://dx.doi.org/10.1016/j.jvcir.2016.09.016
1047-3203/© 2016 Elsevier Inc. All rights reserved.

Some other extensions of Tian's method were proposed. Inspired by the integer transform [16], the improved ones have been proposed in [18–21]. DE was extended to the pixels blocks containing more than two pixels by Alattar [22], and thus the EC is largely increased. By analyzing the existed problem in Alattar's method [22], Wang et al. [23,24], and Peng et al. [25] respectively presented the improved integer transform to achieve higher embedding performance. In addition, Zhang et al. proposed a dual reversible watermarking algorithm with tamper detection based on the generalized integer transform [22] and the multi-scale decomposition [26].

Besides, another extension was proposed by Thodi et al., who firstly expanded the prediction-errors instead of difference values to carry data [27]. Compared with the histogram of difference values, the histogram of prediction errors is sharper, and therefore, the distortion introduced by prediction-error expansion (PEE) is lower than that of DE for a given EC. Besides, HS is incorporated into Thodi et al.'s method so as to high-efficiently compress the location map. Afterwards, Thodi et al.'s method has been extensively studied, and accordingly some improvement methods [28–36,8,37–44] have been proposed. Among them, Sachnev et al. proposed a sorting technique to priorly select smooth prediction-errors obtained by the Rhombus predictor (a full-enclosed predictor) for carrying data.

In addition to the aforementioned methods, another relatively new research field in RDH focuses on achieving high image fidelity at low payload [45–52]. This new research originally came from the development of pixel value ordering (PVO) technique at Li et al.'s work [45]. In their method, the PVO of each block is the guarantee of lossless recovery of original images and embedded data. To this end, only the maximum and minimum are modified so as to ensure the PVO is unchanged after data embedding. Specifically, the maximum (or minimum) is predicted by the second largest (or smallest) pixel to obtain PVO-based prediction-error histogram. The bin 1 (i.e., the bin with the prediction error equal to 1) is embedded with 1-bit data, while the bins larger than 1 are shifted to ensure reversibility. However, the bin O is excluded in the data embedding process. In fact, the bin 0 is very suitable for data embedding, because it can introduce very low distortion for the same payload.

In order to utilize the blocks with prediction-errors equal to 0 for data embedding, Peng et al.'s work has proposed an improved PVO (IPVO) method [47]. In their method, the new difference is defined considering the pixel locations of the maximum and second largest value (or, the minimum and second smallest value). In this way, the bin 0 can be used for data embedding, and thus, the number of embeddable prediction-errors is largely increased. Therefore, Peng et al.'s work achieves better performance, compared with Li et al.'s method.

Wang et al. have proposed a novel RDH method. In Wang et al.'s method, each 4×4-sized image block is divided into one of three groups: rough, normal and fat groups. Specifically, for a rough block, it is excluded from the data embedding process due to its high complexity. For a normal block, since it is in a moderately smooth area, it is not suitable for further division into 2×2 sub-blocks. Therefore, it can carry at most 2 data bits by using IPVO. When a block is denoted as a flat one, it is further divided into four 2×2 sub-blocks to embed more data bits. Each 2×2-sized sub-block can be embedded with 2 data bits by using IPVO. In this way, flat blocks can embed high embedding capacity while preserve good visual quality. In fact, it is not certain whether 4×4-sized blocks can perform the best. Hence, the host image should be partitioned into various-sized blocks, e.g., 3×3, 5×3 or 5×5. Besides, in Wang et al.'s method, a flat 4×4 block is further partitioned into non-overlapped equal-sized sub-blocks. In fact, it is possible for a 4×4 block that it can be flexibly divided into sub-blocks of unequal blocks, e.g., 1×6-sized and 1×10-sized two sub-blocks according to local complexity. With the above consideration, we argue that Wang et al.'s method can be further improved.

The key idea of our method is to create more embeddalbe spaces with a lower distortion, which is implemented from the following two aspects. One aims at partitioning flexibly a smooth block into non-overlapped sub-blocks of arbitrary size according to a local complexity measurement. After partition, since each resulting sub-block can be treated as an independent embedding unit, this block can be embedded with more data bits. The other is that the different pixel modification method is utilized for blocks (or sub-blocks) of different size. In our method, the host image is firstly divided into non-overlapped $r \times c$-sized blocks, where $r \in \{3, 4, 5\}$ and $c \in \{3, 4, 5\}$. For a $r \times c$-sized block, if it is located in a smooth region, then it may be further divided into sub-blocks, and how to divide this block is determined by its corresponding complexity levels. The lower the complexity level is, the smaller sub-block size and larger to-be-embedded data size in each block. Specifically, the local complexity measurement calculated using pixel context is divided into four levels (very low/low/moderate/high correlation).

Taking a 5×5-sized block for example, if its complexity belongs to the high-correlation level, this block can be further divided into 1×3-sized sub-blocks. Correspondingly, a three-pixel-block-based IPVO (TIPVO) is proposed to reduce the embedding distortion. Specifically, the maximum in a 1×3-sized block is used to predict the other two pixels (i.e., the minimum and second largest pixel), and two differences are obtained. In the data embedding process, there exist two situations in which one difference needs to be subtracted or added by one. One is that when one difference is expandable and the to-be-embedded bit is 1, its value is decreased (or increased) by 1 to carry 1 bit data. The other is that when one difference is shiftable, it is decreased (or increased) by 1 to ensure reversibility. When both two difference values need to be decreased by 1, TIPVO achieves the decrease of two difference values by only increasing the maximum by 1, instead of decreasing both the minimum and second largest pixel by 1 like in some existing methods. In this way, TIPVO preserves as much as possible the image visual quality.

For a moderate-correlation 5×5 sized block, since it is in a moderately smooth area, it is not suitable for further division into 1×3 sub-blocks. Therefore, for this block, it is divided into two sub-blocks of unequal size: e.g., a sub-block with size 1×10 and a sub-block with size 1×15. One can also divide this block into sub-blocks of some other sizes: e.g., a sub-block with size 1×12 and a sub-block with size 1×13. In order to better exploit the redundancy among the pixels in a block, we propose a pixel-modification method, in which two largest and two smallest pixels are modified simultaneously using IPVO so that a moderate-correlation block can be embedded with at most four bits. For a low-correlation block, since it has weak intra-block correlation, it is not suitable for further division, and in order to maintain good visual quality, the number of pixels to be modified should not be large. Based on the above consideration, both maximum and minimum are modified at the same time using IPVO to carry at most 2 data bits.

Two-layer embedding is used in our proposed method to further increase the embedding performance. Two-layer embedding means that the embedding process is performed once again to embed data into an already embedded image. In fact, a single layer embedding is a special two-layer embedding whose the second embedding process is not performed. Since the block is $r \times c$, there exist $r \times c$ different ways of block partition for each embedding process. Therefore, there are $(r \times c)^2$ partition ways for two-layer embedding. In order to reduce the computational complexity, mode 1 is utilized in the first layer embedding, while each of

$r \times c$ modes is selected by experiments for the second layer embedding. Experimental results demonstrate that our method achieves higher PSNRs than the ones [45,47,53] for the same payload. Besides, in comparison with the Sachnev et al.'s work [29], ours also obtains a superior performance.

2. Related works

In this section, Peng et al.'s method [47] and Wang et al.'s method [50] are briefly introduced.

2.1. Peng et al.'s method [47]

In Peng et al.'s method [47], both the maximum and minimum in a block are exploited for data embedding. Here, we will give a brief introduction to the maximum-modification-based and minimum-modification-based embedding phases, respectively (refer to paper [47] for the details). The main idea of the maximum-modification-based embedding phase can be summarized as follows:

- For a n-sized block, the pixels are sorted in an ascending order to obtain $(p_{\sigma(1)}, \dots, p_{\sigma(n)})$, where $n = r \times c, \sigma : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ is the unique one-to-one mapping such that: $p_{\sigma(1)} \leqslant \dots \leqslant p_{\sigma(n)}, \sigma(i) < \sigma(j)$if $p_{\sigma(i)} = p_{\sigma(j)}$ and $i < j$.
- A new difference d_{max} given by Eq. (1), is defined considering the pixel locations of the maximum and second largest value.

$$d_{\max} = p_u - p_v \tag{1}$$

where $u = \min(\sigma(n), \sigma(n-1)), v = \max(\sigma(n), \sigma(n-1))$.
- The maximum $p_{\sigma(n)}$ is modified to get the marked pixel value $p'_{\sigma(n)}$ by taking

$$p'_{\sigma(n)} = \begin{cases} p_{\sigma(n)} + b, & \text{if } d_{\max} = 1 \\ p_{\sigma(n)} + 1, & \text{if } d_{\max} > 1 \\ p_{\sigma(n)} + b, & \text{if } d_{\max} = 0 \\ p_{\sigma(n)} + 1, & \text{if } d_{\max} < 0 \end{cases} \tag{2}$$

where $b \in \{0, 1\}$ is a data bit to be embedded.
At the decoder side, for each block, one can use the marked difference $d'_{\max} = p'_u - p'_v$ to recover the original $p_{\sigma(n)}$ as
  - If $d'_{\max} > 0$, we know that $p'_u > p'_v$. Thus, $\sigma(n) < \sigma(n-1)$, $u = \sigma(n)$ and $v = \sigma(n-1)$:
    * if $d'_{\max} \in \{1, 2\}$, the hidden data bit is $b = d'_{\max} - 1$ and the original maximum is $p_{\sigma(n)} = p'_u - b$;
    * $d'_{\max} > 2$, there is no hidden data in this block and the original maximum is $p_{\sigma(n)} = p'_u - 1$.
  - $d'_{\max} \leqslant 0$, we know that $p_u \leqslant p_v$. Thus, $\sigma(n) > \sigma(n-1)$, $u = \sigma(n-1)$ and $v = \sigma(n)$:
    * if $d'_{\max} \in \{0, -1\}$, the hidden data bit is $b = -d'_{\max}$ and the original maximum is $p_{\sigma(n)} = p'_v - b$;
    * $d'_{\max} < -1$, there is no hidden data in this block and the original maximum is $p_{\sigma(n)} = p'_v - 1$.

In fact, the key idea mentioned above is directly applied to $p_{\sigma(1)}$ and $p_{\sigma(2)}$, and then we can get the minimum-modification-based embedding phase. The details are given below.

- A new difference $d_{\min}$ is calculated below,

$$d_{\min} = p_s - p_t \tag{3}$$

where $s = \min(\sigma(1), \sigma(2)), t = \max(\sigma(1), \sigma(2))$.
- The minimum $p_{\sigma(1)}$ is modified to get the marked pixel value $p'_{\sigma(1)}$ by taking

$$p'_{\sigma(1)} = \begin{cases} p_{\sigma(1)} - b, & \text{if } d_{\min} = 1 \\ p_{\sigma(1)} - 1, & \text{if } d_{\min} > 1 \\ p_{\sigma(1)} - b, & \text{if } d_{\min} = 0 \\ p_{\sigma(1)} - 1, & \text{if } d_{\min} < 0 \end{cases} \tag{4}$$

At decoder side, for each block, one can use the marked difference $d'_{\min} = p'_s - p'_t$ to recover the original $p_{\sigma(1)}$ as
  - If $d'_{\min} > 0$, we know that $p'_s > p'_t$. Thus, $\sigma(1) > \sigma(2), s = \sigma(2)$ and $t = \sigma(1)$:
    * if $d'_{\min} \in \{1, 2\}$, the hidden data bit is $b = d'_{\min} - 1$ and the original minimum is $p_{\sigma(1)} = p'_t + b$;
    * $d'_{\min} > 2$, there is no hidden data in this block and the original minimum is $p_{\sigma(1)} = p'_t + 1$.
  - $d'_{\min} \leqslant 0$, we know that $p'_s \leqslant p'_t$. Thus, $\sigma(1) < \sigma(2), s = \sigma(1)$ and $t = \sigma(2)$:
    * if $d'_{\min} \in \{0, -1\}$, the hidden data bit is $b = -d'_{\min}$ and the original minimum is $p_{\sigma(1)} = p'_s + b$;
    * $d'_{\min} < -1$, there is no hidden data in this block and the original minimum is $p_{\sigma(1)} = p'_s + 1$.

2.2. Wang et al.'s method [50]

In Wang et al.'s method, the original image is partitioned into non-overlapped 4×4-sized sub-blocks. Each 4×4-sized image sub-block is further divided into one of three groups: rough, normal and fat groups. Specifically, for a block belonging to the rough group, it is excluded from the data embedding process due to its high complexity.

For a block in the normal group, since it is in a moderately smooth area, it can be used for embedding. However, it is not suitable for further division into 2×2 sub-blocks, because the sub-blocks are not smooth enough for embedding. Based on the above consideration, it can carry at most 2 data bits by using IPVO. When a block is denoted as a flat one, it is further divided into four 2×2 sub-blocks to embed more data bits. Each 2×2-sized sub-block can be embedded with 2 data bits by using IPVO. In this way, flat blocks can embed high embedding capacity while preserve good visual quality.

For convenience of description, a flat 4×4-sized block is arranged into a 1×16-sized pixel array. Taking this pixel array for example, as shown in Fig. 1, it is partitioned into four 1×4-sized sub-blocks. For each sub-block, the maximum is predicted by the second largest pixel and modified by taking Eq. (2). Similarly, the minimum in each sub-block is predicted by the second smallest pixel and modified by taking Eq. (4). In this way, at most two bits can be embedded into a sub-block by using IPVO.

3. The proposed method

In this section, we will introduce the proposed method in detail. Specifically, given a $W \times H$-sized image I, it is partitioned into non-overlapped blocks $P_1, \dots, P_N$ such that each block contains $n = r \times c$ pixels. For each block P_k $(k \in \{1, \dots, N\})$, the pixels are sorted in ascending order to obtain $(p_{\sigma(1)}, p_{\sigma(2)}, \dots, p_{\sigma(n)})$. Here, we omit the block-index k in definition of pixel values of P_k for clarity. In addition, for simplicity, P_k is short for P_k $(k \in \{1, \dots, N\})$, and it is used to represent one of N blocks.

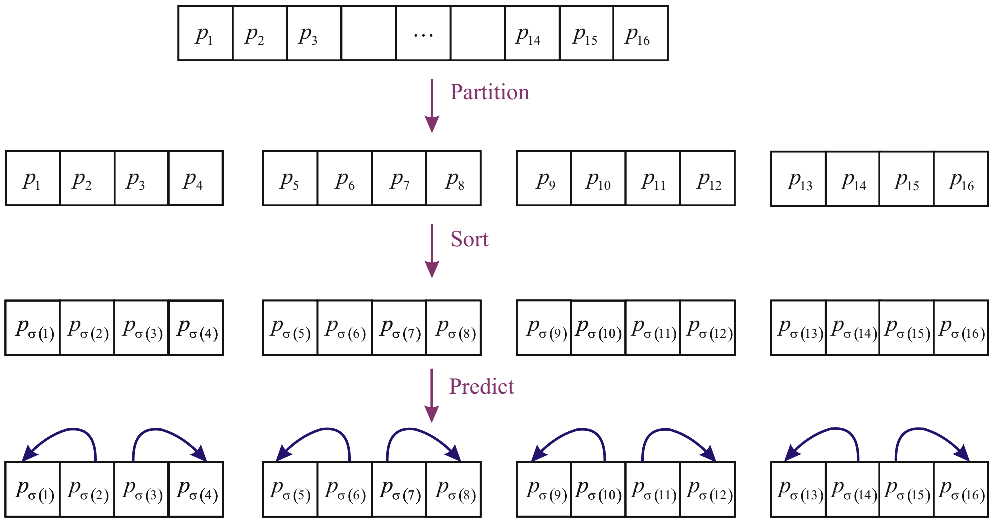


Fig. 1. Diagram of Wang et al.'s method.

All the blocks are divided into two parts (S_P and C_P) according to the local complexity. C_P contains all complex blocks, which are not used for data embedding so as to keep the distortion low. S_P contains the remaining blocks, i.e., all smooth blocks. A smooth block may be further divided into smaller sub-blocks, the size of which is determined by the local complexity measurement. Specifically, the local complexity measurement is divided into three levels (low/-moderate/high correlation). Correspondingly, the smooth blocks having the same correlation level is classified into a group, and thus three groups (i.e., G_1, G_2 and G_3) are generated.

G_1 is composed of the blocks, each of which has high correlation with the neighboring pixels around it. G_2 contains the blocks having moderate correlation with the neighborhood, and G_3 is composed of the blocks with low correlation. Since the pixels in each block of group G_1 have a strong correlation, even if it is further partitioned into 1×3-sized sub-blocks, the pixels in each 1×3-sized sub-block can still keep a high correlation. Correspondingly, TIPVO is proposed to embed at most two bits into a 1×3-sized sub-block. For a block in G_2, since its pixels have a moderate correlation, it can be further divided into smaller sub-blocks. However, the sub-block size needs to be larger than 1×3. In the experiments, for simplification, one block in G_2 is divided into two various-sized sub-blocks, and the size of each sub-block can be selected arbitrarily on the basis that it is ensured to be larger than 4. Accordingly, in order to better exploit redundancy, a pixel modification method is proposed, in which two largest and two smallest pixels in a sub-block are modified so that it can carry at most 4 bits. For a block in G_3, in order to keep the distortion as low as possible, only the maximum and minimum are modified so as to carry at most 2 bits.

3.1. Smoothness classification by pixel neighborhood

For any $P_k(k \in \{1, \ldots, N\})$, if it has a surrounding $(r + c + 1)$-pixel neighborhood (refer to Fig. 2), i.e., $\{I_{1,c+1}, I_{2,c+1}, \ldots, I_{r+1,c+1}, I_{r+1,1}, \ldots, I_{r+1,c}\}$, this neighborhood constitute a new set, denoted by C. The variance of all the pixels in C, denoted by Δ, is used to estimate whether P_k is located in a smooth region or not. Δ is calculated via

$$\Delta = \sqrt{\frac{\sum_{k\in\{1,\ldots,r+1\}}(I_{k,c+1} - \mu)^2 + \sum\limits_{k\in\{1,\ldots,c\}}(I_{r+1,k} - \mu)^2}{r + c + 1}} \tag{5}$$

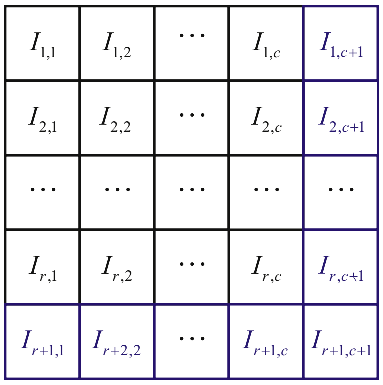


Fig. 2. P_k contains the pixels on r rows and c columns, $I_{1,c+1}, I_{2,c+1}, \ldots, I_{r+1,c+1}, I_{r+1,1}, \ldots, I_{r+1,c}$ are the neighbors of P_k.

where μ is the mean value of set C. When $\Delta < T, P_k$ is regarded to be strongly related to its set C, and therefore, it is classified to be within a smooth region, where T is a predefined threshold which is utilized to distinguish which classification P_k belongs to. Otherwise, P_k is estimated in a complex region. For a smooth block, if $\Delta < \frac{T}{4}$, then it belongs to G_1. If the variance value Δ falls inside the range $(\frac{T}{4}, \frac{T}{2}]$, then it is classified into G_2. If the variance value is larger than $\frac{T}{2}$, i.e., $\Delta > \frac{T}{2}$, then it is in G_3.

3.2. $r \times c$ Embedding modes

There exist $r \times c$ kinds of block partition ways when the block size is $r \times c$. We call a kind of block-partition way as a mode. These $r \times c$ modes are formed by excluding the pixels located in the first i lines and the first j rows from the data embedding process, where $i \in \{0, \ldots, r - 1\}, j \in \{0, \ldots, c - 1\}$. As shown in Fig. 3, we also give an example to illustrate how to obtain $r \times c$ modes when the block size is set to $r \times c$.

According to the aforementioned description, two-layer embedding contains two embedding processes, which are jointly used for obtaining the required EC. In fact, a single embedding process is a special case of two-layer embedding, i.e., two-layer embedding without the second embedding process. A single embedding process may not be able to achieve the optimal embedding performance. To this end, two-layer embedding is utilized in our

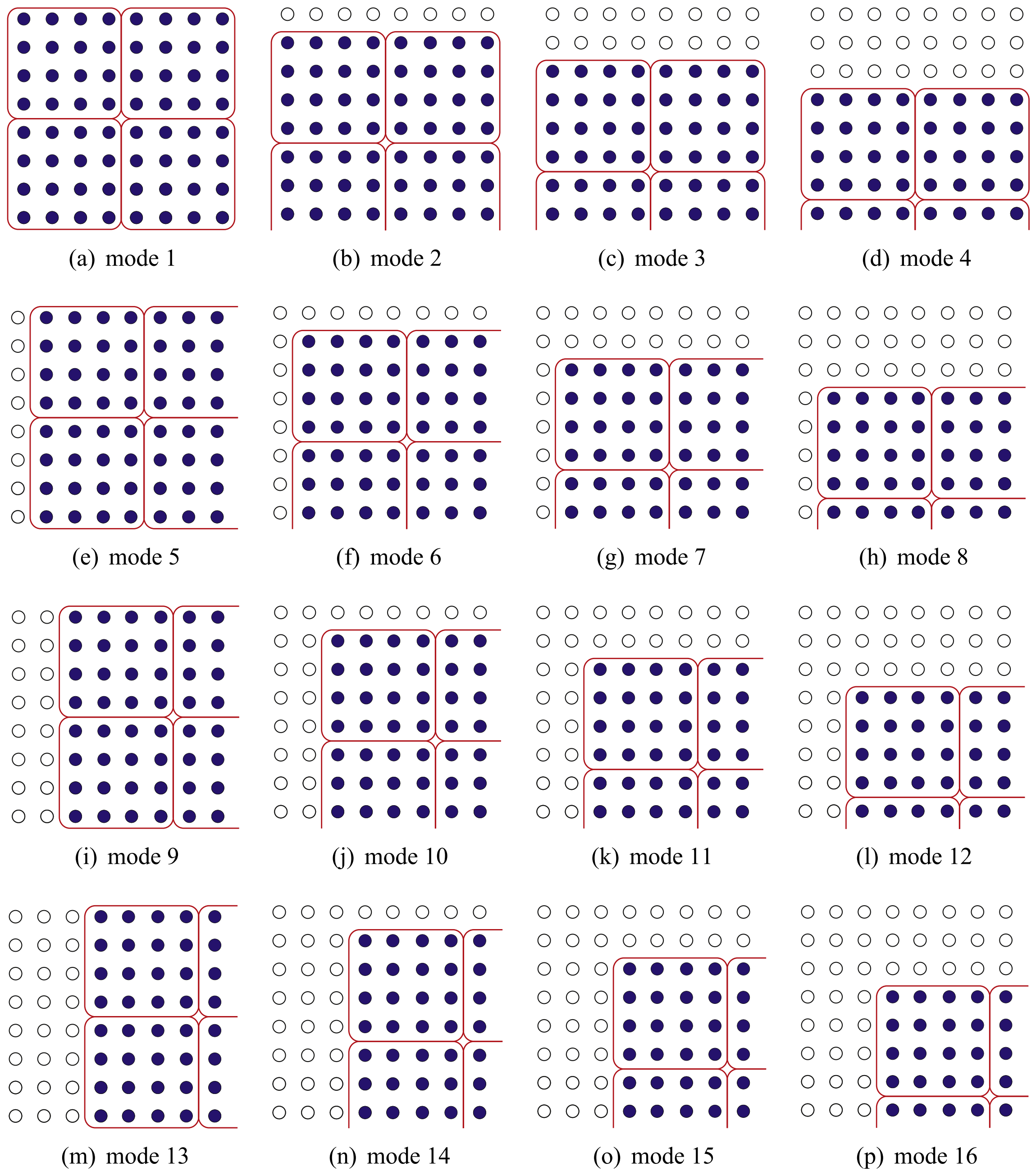


Fig. 3. Sixteen modes, and the pixels marked by white circles are omitted in data embedding for each mode.

method. In general, two-layer embedding can select smoother blocks (i.e., smaller threshold values) for data embedding in each layer embedding process, compared with a single embedding process, and thus the embedding performance is increased.

If each of two processes utilizes the same mode, the obtained performance may be not optimal. In order to further increase embedding performance, we design $r \times c$ modes to search the optimal combinations of two modes. To reduce the computational complexity, the mode utilized in the first embedding process is set to mode 1 in the experiments. In the second embedding process, we traverse each of $r \times c$ modes to find the optimal mode which can achieve the highest PSNR for a given ER. Specially, two threshold values utilized in the first and second layer embedding are denoted as T_1 and T_2, respectively. In the experiments, T_1 is fixed to a value, and then we traverse over all modes, and all possible values of T_2 in each mode, so that the optimal T_2 that can achieve the best image quality is obtained.

3.3. TIPVO: data embedding for blocks in G_1

For each block in G_1, since its pixels has a strong correlation, it often contributes more expandable differences (i.e., difference 0 or 1). Based on this reason, each block in G_1 needs to be further classified into 1×3-sized sub-blocks so that more bits are embedded into a block in G_1. After block-partition, each 1×3-sized sub-block can still remain high intra-block correlation, and therefore each can be processed as one single unit.

A block in G_1 is arranged into a one-dimensional pixel array $\mathbf{p}$ according to a predefined order, i.e., $\mathbf{p} = \{p_1, \ldots, p_n\}$. Specifically, we work the first row from left to right, the second row from right to left, then continue alternating row direction in this manner, so that a one-dimensional pixel array $\mathbf{p}$ is created. Then, the obtained pixel array $\mathbf{p}$ is further classified into 1×3-sized sub-blocks. In this way, $\lfloor \frac{r \times c}{3} \rfloor$ sub-blocks are obtained. We utilize sb to represent a 1×3-sized sub-block, i.e., $sb = (p_1, p_2, p_3)$. sb are sorted in ascending order to obtain $(p_{\sigma(1)}, p_{\sigma(2)}, p_{\sigma(3)})$. Then, two difference values are calculated via Eq. (1), respectively. Specifically, $d_{1\max}$ is the difference between $p_{\sigma(2)}$ and $p_{\sigma(3)}$ considering the order of $\sigma(2)$ and $\sigma(3)$, and $d_{2\max}$ is the difference between $p_{\sigma(3)}$ and $p_{\sigma(1)}$ considering the order of $\sigma(1)$ and $\sigma(3)$. The detailed modification procedure to sb is described in Table 1. In Table 1, $p'_{\sigma(1)}, p'_{\sigma(2)}$ and $p'_{\sigma(3)}$ correspond to the watermarked values of $p_{\sigma(1)}, p_{\sigma(2)}$ and $p_{\sigma(3)}$, respectively, where both b_1 and b_2 represent one data bit, i.e., $b_1 \in \{0, 1\}$ and $b_2 \in \{0, 1\}$.

Referring to Fig. 4, we utilize a 1×15-sized pixel array in G_1 to illustrate the process that it is divided into five 1×3-sized sub-blocks. Taking $(p_{\sigma(1)}, p_{\sigma(2)}, p_{\sigma(3)})$ for example, we utilize $p_{\sigma(3)}$ to predict $p_{\sigma(1)}$ and $p_{\sigma(2)}$, respectively, and thus $d_{1\max}$ and $d_{2\max}$ are obtained. The advantage of TIPVO lies in the fact that $d_{1\max}$ and $d_{2\max}$ can be subtracted by 1 at the same time by only increasing $p_{\sigma(3)}$ by 1. An example is given in Fig. 5 to illustrate the embedding process for three different sub-blocks in G_1. From Fig. 5(a), one can observe that both $d_{1\max}$ and $d_{2\max}$ are subtracted by 1 by only increasing $p_{\sigma(3)}$ by 1. Similarly, referring to Fig. 5(b), when $d_{1\max} = d_{2\max} = 0$, $d_{1\max}$ and $d_{2\max}$ can be embedded respectively with 1 data bit by only modifying $p_{\sigma(3)}$. Fig. 5(c) shows that one bit is embedded into (p_1, p_2, p_3) by embedding $b_1 = 0$ into p_3 and decreasing p_1 by 1 at the same time.

The corresponding data extraction and recovery process of sb are listed in Table 2. In the table, $d'_{1\max}$ is the difference between $p'_{\sigma(2)}$ and $p'_{\sigma(3)}$ considering the order of $\sigma(2)$ and $\sigma(3)$, and $d'_{2\max}$ is the difference between $p'_{\sigma(3)}$ and $p'_{\sigma(1)}$ considering the order of $\sigma(1)$ and $\sigma(3)$.

3.4. Data embedding for blocks in G_2

For P_k $(k \in \{1, \ldots, N\})$ in G_2, since it is in a moderately smooth region, it is not suitable for further division into 1×3-sized sub-blocks. The reason behind this is that for each 1×3-sized sub-block, if it is processed as a single unit, the distortion introduced by data embedding is large due to that its intra-block correlation is not strong. Based on the idea that if the intra-block correlation is not strong, the block size should be set to a larger size, each block in G_2 is partitioned into two non-overlapped sub-blocks. How to set the sub-block size is determined by block size $r \times c$. Taking a 3×5-sized block in G_2 for example, it can be partitioned into 1×6-sized and 1×9-sized sub-blocks.

Specifically, for three largest pixels of each sub-block, $p_{\sigma(n-2)}$ is used to predict each of two largest pixels (i.e., $p_{\sigma(n-1)}$ and $p_{\sigma(n)}$). Then, two difference values are calculated via Eq. (1), respectively. Specifically, $d_{1\max}$ is the difference between the third largest pixel $(p_{\sigma(n-2)})$ and the second largest pixel $(p_{\sigma(n-1)})$ considering the order of $\sigma(n-2)$ and $\sigma(n-1)$, and $d_{2\max}$ is the difference between the third largest pixel $(p_{\sigma(n-2)})$ and the maximum $(p_{\sigma(n)})$. In this way, the blocks with $p_{\sigma(n)} = p_{\sigma(n-1)} = p_{\sigma(n-2)}$ can be exploited by our method to embed data by modifying $d_{1\max}$ and $d_{2\max}$.

After data embedding illustrated in Eq. (2), each of $p'_{\sigma(n-1)}$ and $p'_{\sigma(n)}$ still remains larger than $p_{\sigma(n-2)}$, where $d'_{1\max}, d'_{2\max}, p'_{\sigma(n-1)}$ and $p'_{\sigma(n)}$ stand for the watermarked values of $d_{1\max}, d_{2\max}, p_{\sigma(n-1)}$ and $p_{\sigma(n)}$, respectively. Therefore, the relationship between $p'_{\sigma(n-1)}$ and $p_{\sigma(n-2)}$ remains unchanged after data embedding. Similarly, the relationship between $p'_{\sigma(n-1)}$ and $p_{\sigma(n-2)}$ keeps unchanged. Depending on invariant relationship between $p'_{\sigma(n)}$ and $p_{\sigma(n-2)}, p_{\sigma(n)}$ can be correctly recovered and the embedded bits can be extracted correctly. This is similar for $p'_{\sigma(n-1)}$.

For three smallest pixels of a block in G_2, p_3 is used to predict each of two smallest pixels (i.e., p_2 and p_1), respectively. And, two difference values (i.e., $d_{1\min}$ and $d_{2\min}$) are calculated via Eq. (3). In this way, the blocks with $p_{\sigma(1)} = p_{\sigma(2)} = p_{\sigma(3)}$ can be exploited by our method to embed data by modifying $d_{1\min}$ and $d_{2\min}$. After data embedding in Eq. (4), each of $p'_{\sigma(2)}$ and $p'_{\sigma(1)}$ still remains smaller than $p_{\sigma(3)}$, where $d'_{1\min}, d'_{2\min}, p'_{\sigma(2)}$ and $p'_{\sigma(1)}$ stand for the watermarked values of $d_{1\min}, d_{2\min}, p_{\sigma(2)}$ and $p_{\sigma(1)}$, respectively. Therefore, the relationship between $p'_{\sigma(2)}$ and $p_{\sigma(3)}$ remains unchanged after data embedding. Similarly, the relationship between $p'_{\sigma(2)}$ and $p_{\sigma(3)}$ also keeps unchanged.

Referring to Fig. 6, we utilize a 1×15-sized pixel array in G_2 to illustrate the process that it is divided into one 1×6-sized sub-block and one 1×9-sized sub-block. For each sub-block, we modify two largest and two smallest pixels at the same time so as to embed at most four bits into each sub-block. A detailed example is given in Fig. 7 to illustrate how to modify simultaneously two largest and two smallest pixels for a 1×6-sized sub-block.

Table 1
Watermarked value of a three-pixel block $(p_{\sigma(1)}, p_{\sigma(2)}, p_{\sigma(3)})$.

Conditions on $(d_{1\max}, d_{2\max})$		Watermarked values
$d_{1\max} = 1$ or $d_{1\max} = 0$	$d_{2\max} = 1$ or $d_{2\max} = 0$	if $(b_1 == 0$ and $b_2 == 0)$ $p'_{\sigma(2)} = p_{\sigma(2)}, p'_{\sigma(1)} = p_{\sigma(1)}$ elseif $(b_1 == 0$ and $b_2 == 1)$ $p'_{\sigma(2)} = p_{\sigma(2)}, p'_{\sigma(1)} = p_{\sigma(1)} - 1$ elseif $(b_1 == 1$ and $b_2 == 0)$ $p'_{\sigma(2)} = p_{\sigma(2)} - 1, p'_{\sigma(1)} = p_{\sigma(1)}$ elseif $(b_1 == 1$ and $b_2 == 1)$ $p'_{\sigma(3)} = p_{\sigma(3)} + 1,$
	$d_{2\max} > 1$ or $d_{2\max} < 0$	if $(b_1 == 1)$ $p'_{\sigma(3)} = p_{\sigma(3)} + 1,$ elseif $(b_1 == 0)$ $p'_{\sigma(2)} = p_{\sigma(2)}, p'_{\sigma(1)} = p_{\sigma(1)} - 1$
$d_{1\max} \in (-\infty, -1] \cup [2, \infty)$	$d_{2\max} = 1$ or $d_{2\max} = 0$	if $(b_1 == 1)$ $p'_{\sigma(3)} = p_{\sigma(3)} + 1,$ elseif $(b_1 == 0)$ $p'_{\sigma(2)} = p_{\sigma(2)} - 1, p'_{\sigma(1)} = p_{\sigma(1)}$
	$d_{2\max} \in (-\infty, -1] \cup [2, \infty)$	$p'_{\sigma(3)} = p_{\sigma(3)} + 1$

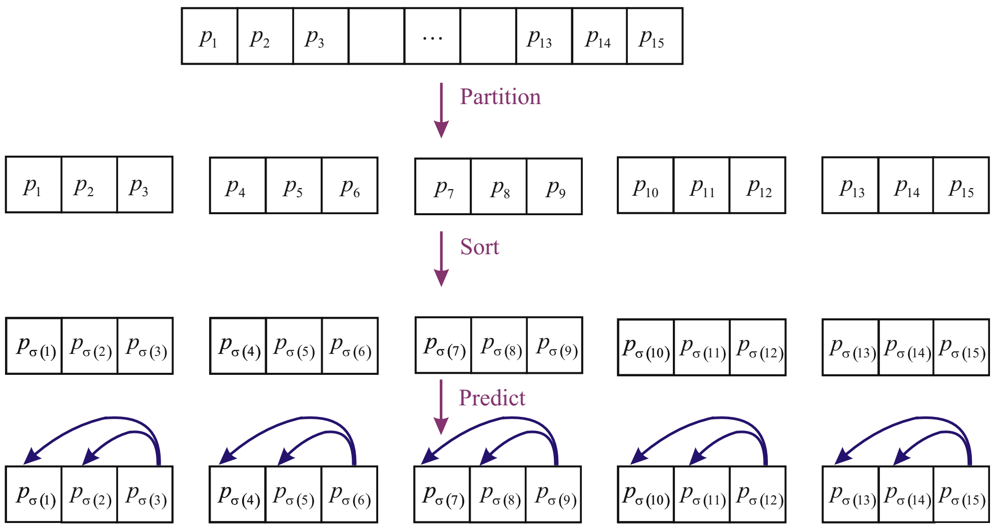


Fig. 4. Illustration of the embedding process for a block in G_1.

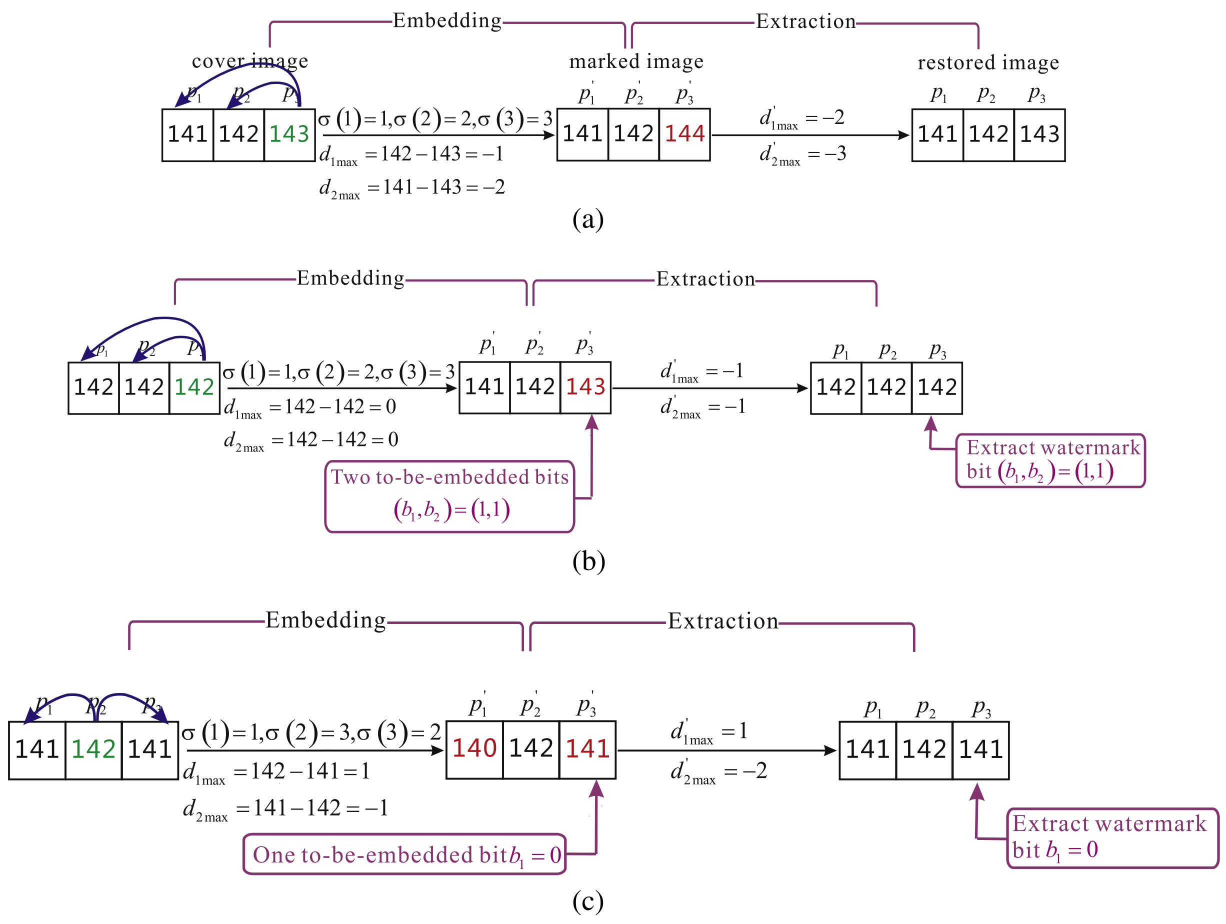

Embedding
Extraction
cover image
marked image
restored image
Two to-be-embedded bits $(b_1,b_2)=(1,1)$
Extract watermark bit $(b_1,b_2)=(1,1)$
One to-be-embedded bit $b_1=0$
Extract watermark bit $b_1=0$
(a)
(b)
(c)

Fig. 5. Illustration of the embedding process of 1 × 3-sized sub-blocks in G_1. The maximum marked in green is used to predict the minimum and second largest pixel, respectively. The pixels marked in red are modified in the embedding process.

3.5. Data embedding for blocks in G_3

For P_k in G_3, since it has weak intra-block correlation, it is not suitable for further division into smaller sub-blocks. Considering high local complexity of blocks in G_3, the pixels as few as possible in a block are modified so as to preserve high visual quality. Based on this reason, only two pixels (i.e., the maximum and minimum) are modified to carry at most two bits. Specifically, $p_{\sigma(n)}$ is predicted by $p_{\sigma(n-1)}$ and are modified via Eq. (2) in the data embedding process. Similarly, $p_{\sigma(2)}$ is used to predict $p_{\sigma(1)}$) and are modified via Eq. (4).

Table 2
Extracted data bit and recovered value from a watermarked three-pixel block $(p'_{\sigma(1)}, p'_{\sigma(2)}, p'_{\sigma(3)})$.

Conditions on $(d'_{1\max}, d'_{2\max})$	Watermarked values	
$d'_{1\max} = 1$ and $d'_{2\max} = 0$	$p_{\sigma(2)} = p'_{\sigma(2)}, p_{\sigma(1)} = p'_{\sigma(1)}$	$b_1 = 0$ and $b_2 = 0$
$d'_{1\max} = 0$ and $d'_{2\max} = 1$		
$d'_{1\max} = 0$ and $d'_{2\max} = 0$		
$d'_{1\max} = 1$ and $d'_{2\max} = 1$		
$d'_{1\max} = 2$ and $d'_{2\max} = 2$	$p_{\sigma(3)} = p'_{\sigma(3)} - 1$	$b_1 = 1$ and $b_2 = 1$
$d'_{1\max} = 2$ and $d'_{2\max} = -1$		
$d'_{1\max} = -1$ and $d'_{2\max} = -1$		
$d'_{1\max} = -1$ and $d'_{2\max} = 2$		
$d'_{1\max} = 2$ and $d'_{2\max} = 1$	$p_{\sigma(2)} = p'_{\sigma(2)} + 1, p_{\sigma(1)} = p'_{\sigma(1)}$	$b_1 = 1$ and $b_2 = 0$
$d'_{1\max} = 2$ and $d'_{1\max} = 0$		
$d'_{1\max} = -1$ and $d'_{2\max} = 1$		
$d'_{1\max} = -1$ and $d'_{2\max} = 0$		
$d'_{1\max} = 1$ and $d'_{2\max} = -1$	$p_{\sigma(2)} = p'_{\sigma(2)}, p_{\sigma(1)} = p'_{\sigma(1)} + 1$	$b_1 = 0$ and $b_2 = 1$
$d'_{1\max} = 1$ and $d'_{2\max} = 2$		
$d'_{1\max} = 0$ and $d'_{2\max} = 2$		
$d'_{1\max} = 0$ and $d'_{2\max} = -1$		
$d'_{1\max} = 1$ and $d'_{2\max} \in (-\infty, -1) \cup (2, \infty)$	$p_{\sigma(2)} = p'_{\sigma(2)}, p_{\sigma(1)} = p'_{\sigma(1)} + 1$	$b_1 = 0$
$d'_{1\max} = 0$ and $d'_{2\max} \in (-\infty, -1) \cup (2, \infty)$		
$d'_{1\max} = 2$ and $d'_{2\max} \in (-\infty, -1) \cup (2, \infty)$	$p_{\sigma(3)} = p'_{\sigma(3)} - 1$	$b_1 = 1$
$d'_{1\max} = -1$ and $d'_{2\max} \in (-\infty, -1) \cup (2, \infty)$		
$d'_{1\max} \in (-\infty, -1) \cup (2, \infty)$ and $d'_{2\max} \in (-\infty, -1) \cup (2, \infty)$		no embedded data bit
$d'_{1\max} \in (-\infty, -1) \cup (2, \infty)$ and $d'_{2\max} = 1$	$p_{\sigma(2)} = p'_{\sigma(2)} + 1, p_{\sigma(1)} = p'_{\sigma(1)}$	$b_1 = 0$
$d'_{1\max} \in (-\infty, -1) \cup (2, \infty)$ and $d'_{2\max} = 0$		
$d'_{1\max} \in (-\infty, -1) \cup (2, \infty)$ and $d'_{2\max} = 2$	$p_{\sigma(3)} = p'_{\sigma(3)} - 1$	$b_1 = 1$
$d'_{1\max} \in (-\infty, -1) \cup (2, \infty)$ and $d'_{2\max} = -1$		

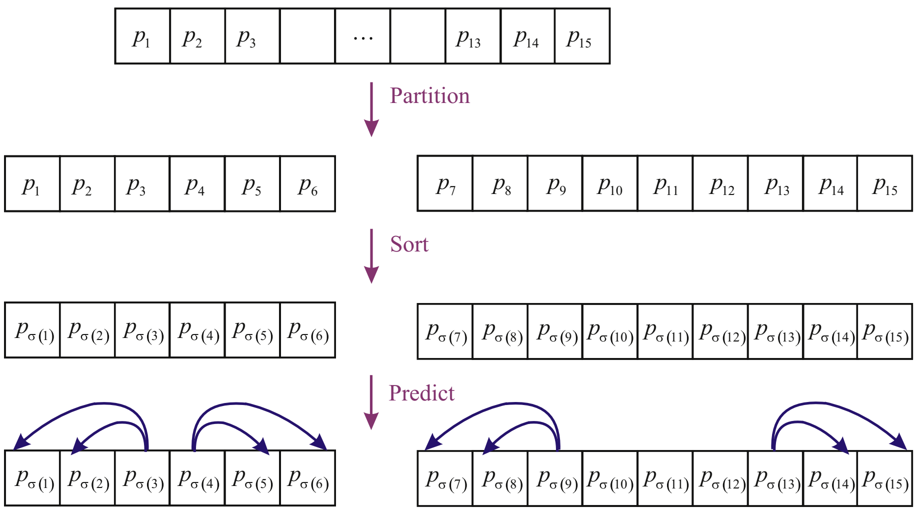


Fig. 6. Illustration of the embedding process of a block in G_2.

3.6. Algorithm description

3.6.1. Location map construction

To prevent overflow/underflow, for $p_i \in \mathbb{Z}$ $(i \in \{\sigma(1), \ldots, \sigma(n)\})$, its watermarked value p'_i must be included in $[0, 255]$, and $p'_i \in \mathbb{Z}$. We define

$$D = \{P_k \in A : 0 \leqslant p'_i \leqslant 255 \ (i \in \{\sigma(1), \ldots, \sigma(n)\})\}$$

where $A = \{P_k = \{p_{\sigma(1)}, \ldots, p_{\sigma(n)}\} : 0 \leqslant p_i \leqslant 255 \ (i \in \{\sigma(1), \ldots, \sigma(n)\})\}$. We then define a subset of D,

$$S_P = \{P_k \in D : \Delta < T\}$$

Clearly, there is no overflow/underflow if using elements of S_P to embed data. S_P is partitioned into the following three groups: G_1, G_2 and G_3, where $G_1 = \{P_k \in S_P : \Delta \leqslant \frac{T}{4}\}, G_2 = \{P_k \in S_P : \Delta \in (\frac{T}{4}, \frac{T}{2}]\}$, and $G_3 = \{P_k \in S_P : \Delta \in (\frac{T}{2}, T)\}$.

When $\Delta < T$, in order to better differentiate the blocks containing overflow/underflow pixels from the blocks in S_P, we still need to define another set:

$$O_P = \{P_k \notin D : \Delta < T\}.$$

A location map is generated in which $P_k \in S_P$ are marked by '1' while $P_k \in O_P$ are marked by '0'. This map is compressed losslessly by an arithmetic encoder and the resulting binary sequence is

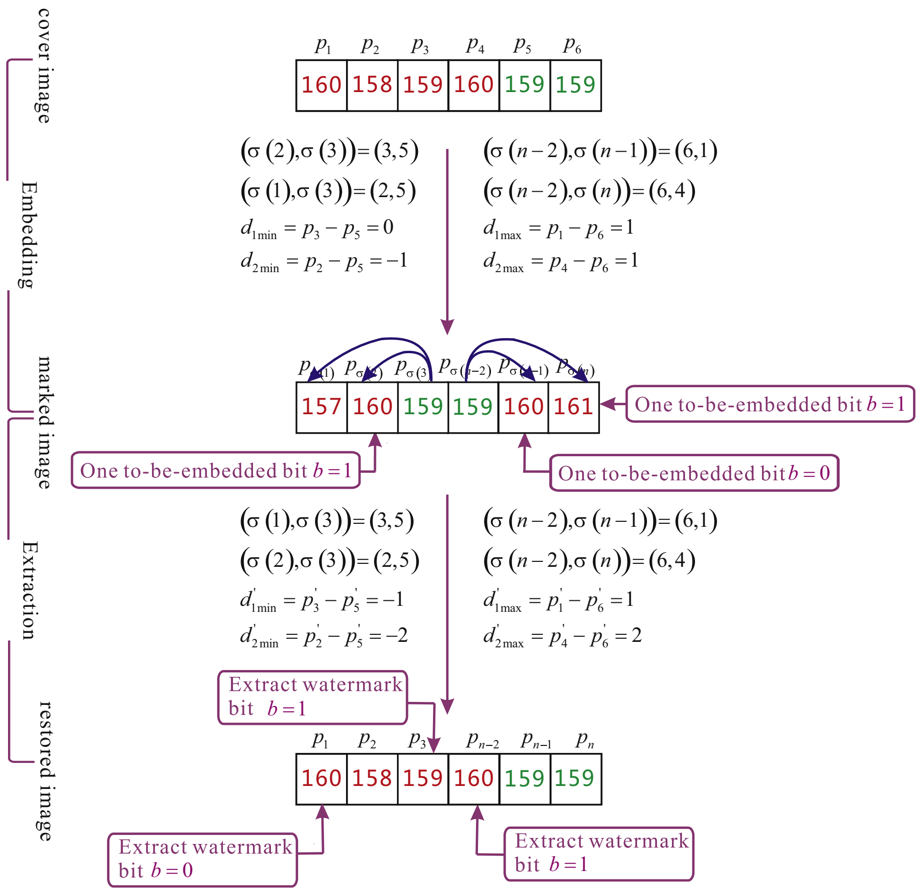

cover image
Embedding
marked image
Extraction
restored image
p_1 p_2 p_3 p_4 p_5 p_6
160 158 159 160 159 159
$(\sigma(2),\sigma(3))=(3,5)$
$(\sigma(1),\sigma(3))=(2,5)$
$d_{1\min}=p_3-p_5=0$
$d_{2\min}=p_2-p_5=-1$
$(\sigma(n-2),\sigma(n-1))=(6,1)$
$(\sigma(n-2),\sigma(n))=(6,4)$
$d_{1\max}=p_1-p_6=1$
$d_{2\max}=p_4-p_6=1$
157 160 159 159 160 161
One to-be-embedded bit $b=1$
One to-be-embedded bit $b=1$
One to-be-embedded bit $b=0$
$(\sigma(1),\sigma(3))=(3,5)$
$(\sigma(2),\sigma(3))=(2,5)$
$d'_{1\min}=p'_3-p'_5=-1$
$d'_{2\min}=p'_2-p'_5=-2$
$(\sigma(n-2),\sigma(n-1))=(6,1)$
$(\sigma(n-2),\sigma(n))=(6,4)$
$d'_{1\max}=p'_1-p'_6=1$
$d'_{2\max}=p'_4-p'_6=2$
Extract watermark bit $b=1$
p_1 p_2 p_3 p_{n-2} p_{n-1} p_n
160 158 159 160 159 159
Extract watermark bit $b=0$
Extract watermark bit $b=1$

Fig. 7. Illustration of the embedding process of a 1×6-sized sub-block in G_2. The third largest pixel marked in green is used to predict the maximum and second largest one, respectively. Similarly, the third smallest pixel is used to predict the minimum and second largest one, respectively. The pixels marked are red are modified in the embedding process.

denoted as $\mathscr{L}$, and we suppose that its length is L_S. Here, arithmetic coding is utilized for lossless compressing.

3.6.2. Overhead information

To guarantee reversibility, some overhead information should be recorded before data embedding. In embedding procedure, it must be embedded into the host image together with the payload so as to ensure reliable image reconstruction on the receiver side.

For each single layer, the corresponding overhead information includes

* the compressed location map (L_S bits)
* T_m ($m \in \{1,2\}$) (8 bits),
* block size parameters r (3 bits) and c (3 bits),
* sub-block size parameters n_2 (5 bits) and n_1 (5 bits),
* the type of mode MT (5 bits),
* E_C (18bits),
* end of symbol (EOS) (8 bits).

where MT is created to distinguish $r \times c$ modes, and $m = 1$ stands for the first layer while $m = 2$ for the second one. According to the aforementioned description, a block in G_2 is divided into one $1 \times n_1$-sized and one $1 \times n_2$-sized sub-blocks. So, n_1 and n_2 need to be recorded so as to ensure reversibility. Since the maximum value of $r \times c$ is 25, the length of MT is set to be 5 bits so as to represent 25 modes. A unique EOS symbol is appended at the end of the overhead information. Considering that the first and second layers may not be fully embedded, an extra information EC is added after the MT. The EC is composed of 18 bits (representing EC in each single layer, specially, $2^{18} = 512 \times 512$) and 1 bit (the number of layers, 0 represents one single layer embedding while 1 represents two-layer embedding).

3.6.3. Optimal-threshold-determination for combined embedding

Suppose $\{P_1, \ldots, P_{Nk}\}$ are all smooth blocks, where $Nk \leqslant N$. Nk_1, Nk_2 and Nk_3 are used to represent numbers of blocks in G_1, G_2 and G_3, respectively. As we have known (see Section II-C of [8] for details), for the histogram-based RDH, if the maximum modification to pixel values is 1 in data embedding, the expected value of the modification (in l^2-norm) to cover image in the mth ($m \in \{1,2\}$) layer embedding, denoted as $E_D(T_m)$, is $\frac{1}{2}N_{exp} + N_{shift}$, where N_{exp} and N_{shift} are numbers of expanded and shifted pixels, respectively. N_{exp} can be formulated as

$$\begin{aligned} N_{exp} = &\#\{1 \leqslant i \leqslant Nk_1 : d_{i,j\max} \in \{0,1\}(1 \leqslant j \leqslant 2)\} + \#\{1 \leqslant i \\ &\leqslant Nk_2 : d_{i,j\max} \in \{0,1\}(1 \leqslant j \leqslant 2)\} + \#\{1 \leqslant i \leqslant Nk_2 \\ &: d_{i,j\min} \in \{0,1\}(1 \leqslant j \leqslant 2)\} + \#\{1 \leqslant i \leqslant Nk_3 : d_{i,1\max} \\ &\in \{0,1\}\} + \#\{1 \leqslant i \leqslant Nk_3 : d_{i,1\min} \in \{0,1\}\} \end{aligned} \quad (6)$$

where $\#$ denotes the cardinal number of a set, $d_{i,j\max}$ denotes the jth $d_{\max}$ of the ith block in one of G_1, G_2 and G_3, e.g., $d_{i,1\max}(1 \leqslant i \leqslant Nk_1)$ denotes the d_{1max} of the ith block in G_1.

And, N_{shift} can be written as

$$\begin{aligned} N_{shift} = \#\{1 \leqslant i \leqslant Nk_1 : d_{i,j\max} > 1 \text{ or } d_{i,j\max} < 0 (1 \leqslant j \leqslant 2)\} \\ + \#\{1 \leqslant i \leqslant Nk_2 : d_{i,j\max} > 1 \text{ or } d_{i,j\max} < 0 (1 \leqslant j \leqslant 2)\} \\ + \#\{1 \leqslant i \leqslant Nk_2 : d_{i,j\min} > 1 \text{ or } d_{i,j\min} < 0 (1 \leqslant j \leqslant 2)\} \\ + \#\{1 \leqslant i \leqslant Nk_3 : d_{i,1\max} > 1 \text{ or } d_{i,1\max} < 0\} \\ + \#\{1 \leqslant i \leqslant Nk_3 : d_{i,1\min} > 1 \text{ or } d_{i,1\min} < 0\} \end{aligned} \quad (7)$$

The capacity in the mth ($m \in \{1,2\}$) layer embedding, denoted as $E_C(T_m)$, equals N_{exp}.

Taking the underflow/overflow problem into account, the optimal thresholds $(T_1{}^*, T_2^*)$ for the combined embedding of two modes are determined as follows:

$$\begin{cases} \underset{0 \leqslant T_1, T_2 \leqslant \Delta_{\max}}{\arg\min} \frac{E_D(T_1)+E_D(T_2)}{E_C(T_1)+E_C(T_2)} \\ \text{subject to } E_C(T_1) + E_C(T_2) \geqslant L_{\sum} \end{cases} \quad (8)$$

where $\Delta_{\max}$ denotes the maximum variance. Furthermore, to verify that Eq. (7) is effective in constructing the optimal combined the first and second embedding, we stimulate all the results by using different (T_1, T_2) as shown in Fig. 8 for the capacity of 10,000 bits. Here, the block size is specified as $r \in \{3,4,5\}$ and $c \in \{3,4,5\}$. The combinations of (T_1, T_2) are sorted in terms of T_1 in an ascending order.

For a given T_1, we increase T_2 gradually from the initial value $T_2 = 1$ to its maximum with a step size of 0.5. For each value of T_2, when the required capacity is just accommodated, stop the embedding process. Meanwhile, calculate the corresponding PSNR, and record the value of T_2 and the corresponding embedding mode. The optimal T_2^* is the one which can yields the highest PSNR. It is clear that $(T_1{}^*, T_2^*)$ derived from Eq. (7) yields the highest PSNR whatever the block size and the image are. From Fig. 8, one can also see that the values of T_1^* and T_2^* obtained by experiments is small. Taking Fig. 8(e) for example, $T_1^* = 1, T_2^* = 5$. Note that, $T_2^* = 0$ means that the highest PSNR is achieved by a single layer embedding at given r and c.

3.7. Embedding procedure

Now, the embedding procedure is described as follows step by step. We use the notation $\mathscr{P}$ to denote the desired payload. $\mathscr{P}_{\mathscr{L}}$ is the length of $\mathscr{P}$.

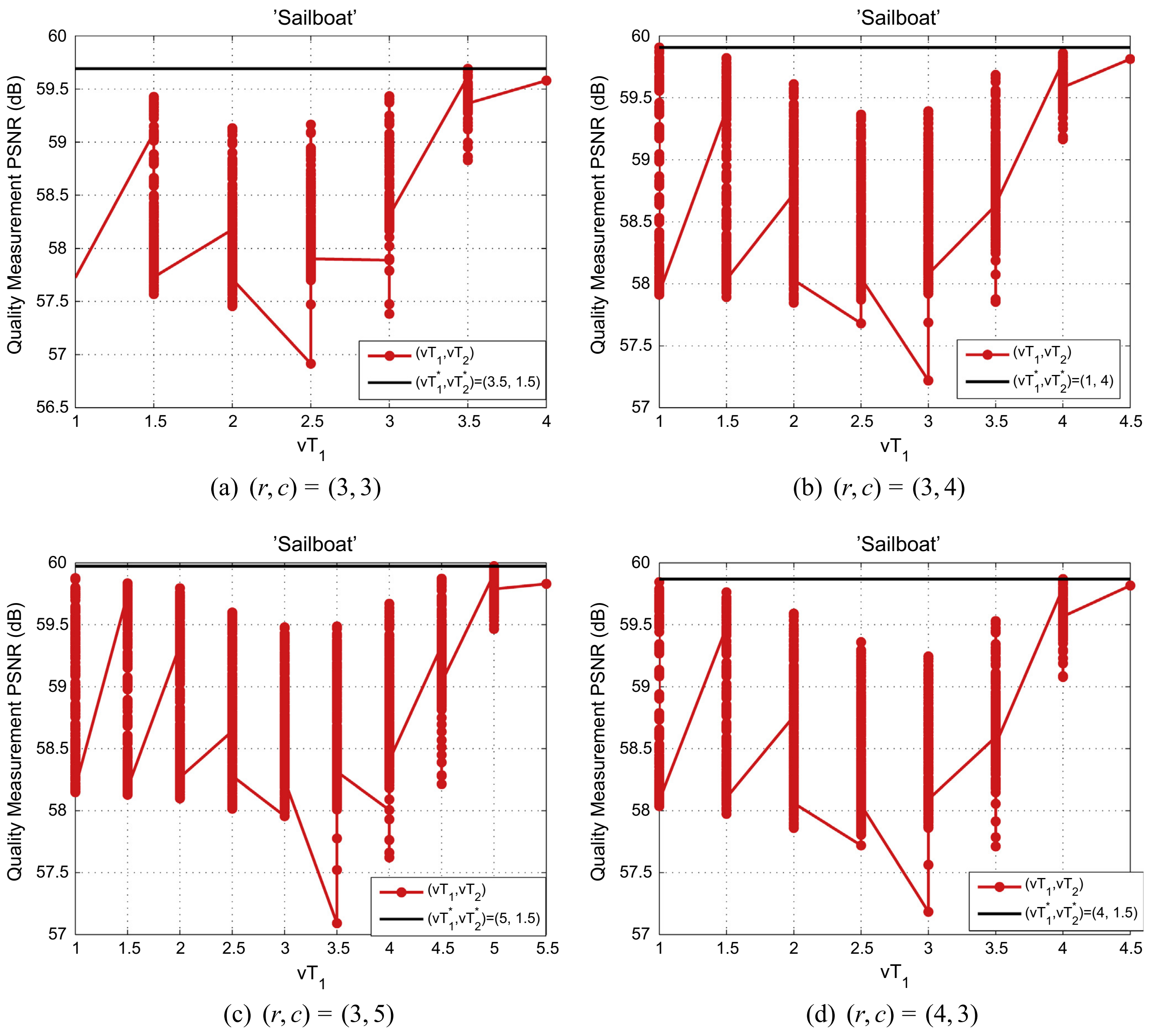
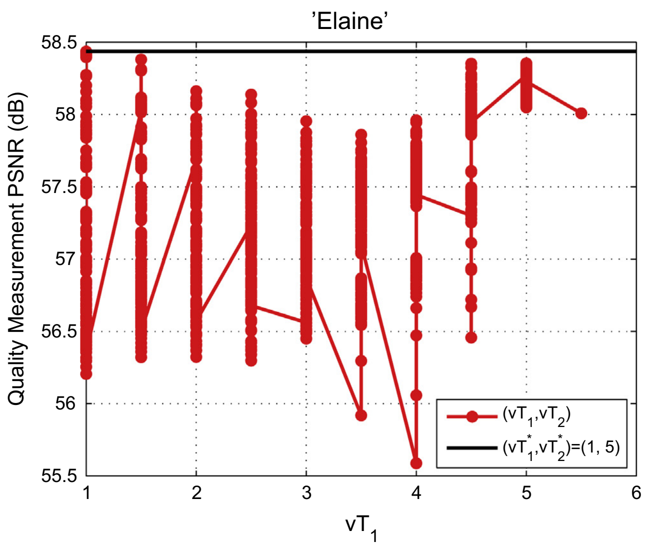
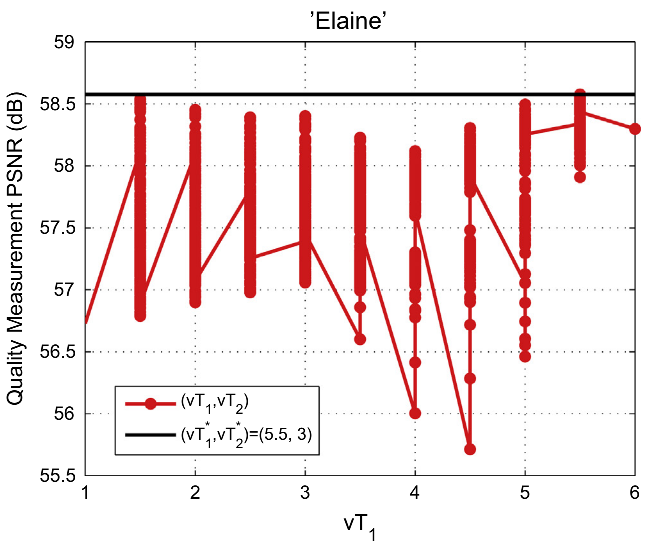


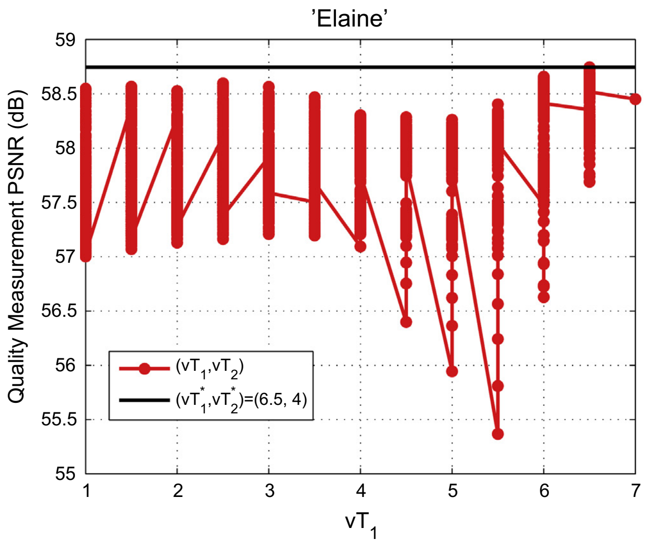
Fig. 8. Performance comparison for a capacity of 10,000 bits on 'Sailboat' and 'Elaine' images by using different (T_1, T_2).



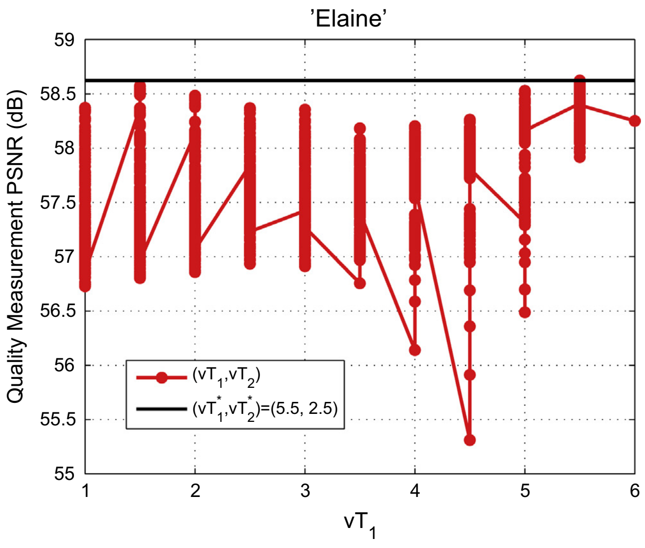
(e) $(r, c) = (3, 3)$



(f) $(r, c) = (3, 4)$



(g) $(r, c) = (3, 5)$



(h) $(r, c) = (4, 3)$

Fig. 8 (*continued*)

Step 1 *Watermark embedding in a single layer*
Since there exists difference in embedding way for the overhead information and the payload, **Step 1** is divided into two parts so that each of two embedding ways can be described clearly.

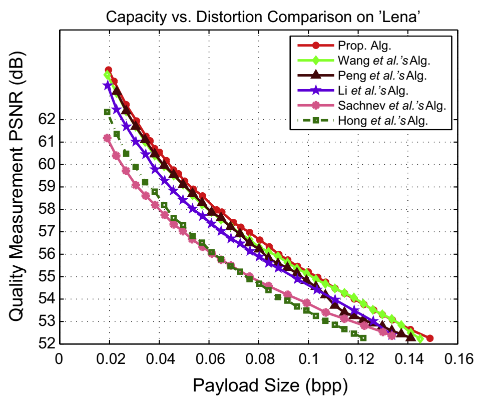
Step 1.1 *Watermark bits embedding*
If P_k $(k \in \{1, \ldots, N\})$ belongs to $(D - S_P)$, we do nothing with it. If P_k $(k \in \{1, \ldots, N\})$ is in S_P, S_P is further divided into three groups: G_1, G_2 and G_3. For one block in G_1, the detailed modification to *sb* is illustrated in Section 3.3. Similarly, the blocks in G_2 and G_3 are modified according to Sections 3.4 and 3.5, respectively.

Step 1.2 *Overhead information embedding*
The overhead information in the current layer is formed. Suppose that the payload to be embedded in the current layer is $\mathscr{P}_{\mathscr{C}}$. We know $|\mathscr{P}_{\mathscr{C}}| \leqslant E_C(T_m)$ $(m \in \{1, 2\})$, where $|\mathscr{P}_{\mathscr{C}}|$ denotes the length of $\mathscr{P}_{\mathscr{C}}$. First, one small part of the payload $\mathscr{P}_{\mathscr{C}}$ is embedded in the blocks in S_P according to Step 1.1. Next, the LSBs of the first $L_{\sum}$ watermarked pixels are appended to the payload $\mathscr{P}_{\mathscr{C}}$ and replaced by the overhead information. Finally, the rest part of $\mathscr{P}_{\mathscr{C}}$ and the bit sequence $\mathscr{C}$ containing $L_{\sum}$ appended bits are embedded into the remaining blocks in S_P using the same method in **Step 1.1**.

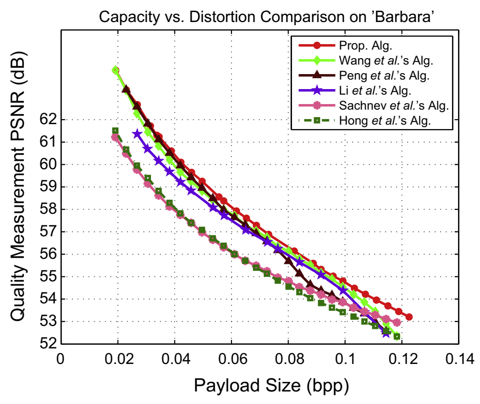
Step 2 *Obtaining the watermarked image I_w*
Step 2 is divided into two parts as follows. $\mathscr{P}_{\mathscr{C}}$ is initialized to $\mathscr{P}$, and $|\mathscr{P}_{\mathscr{C}}| = \mathscr{P}_{\mathscr{L}}$. If $|\mathscr{P}_{\mathscr{C}}|$ is less than $E_C(T_m)$ of this current layer, go to **Step 2.1**. Otherwise, do **Step 2.2**. Note that $m = 1$ for the first layer while $m = 2$ for the second layer.

Step 2.1 *Creating the watermarked image I_w*
$|\mathscr{P}_C| \leqslant E_C(T_m)$ means that only performing the current layer can produce $\mathscr{P}$. That is to say, this is the last layer. After **Step 1** is performed, a new watermarked image I_w is obtained.

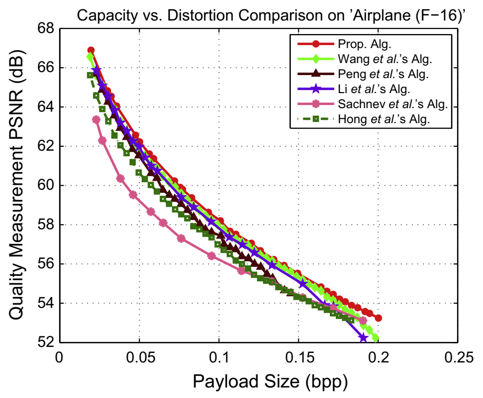
Step 2.2 *Performing the second layer embedding*
Due to $|\mathscr{P}_{\mathscr{C}}| > E_C(T_m)$, we must perform two-layer embedding so as to obtain the payload $\mathscr{P}$. First, for this layer, perform **Step 1** for the payload of length $E_C(T_m)$



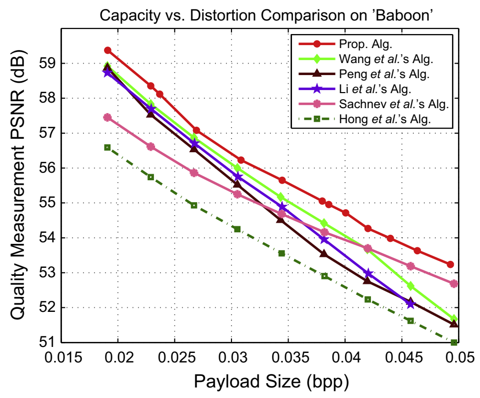
(a) 'Lena'



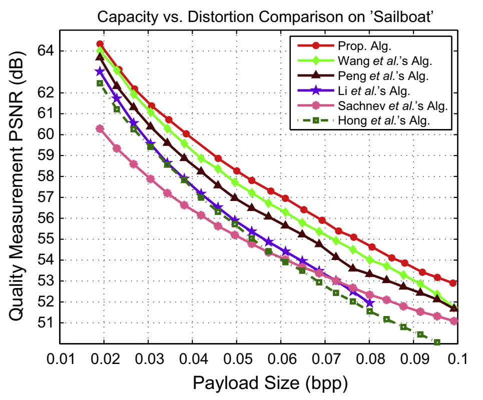
(b) 'Barbara'



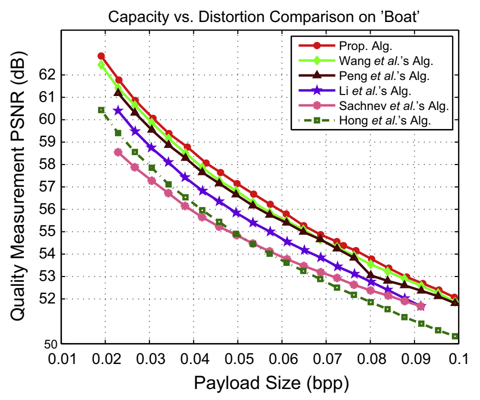
(c) 'Airplane'



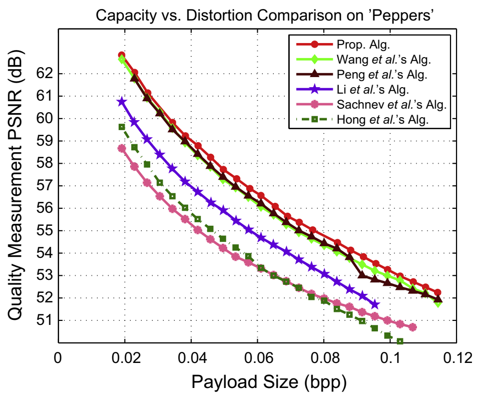
(d) 'Baboon'



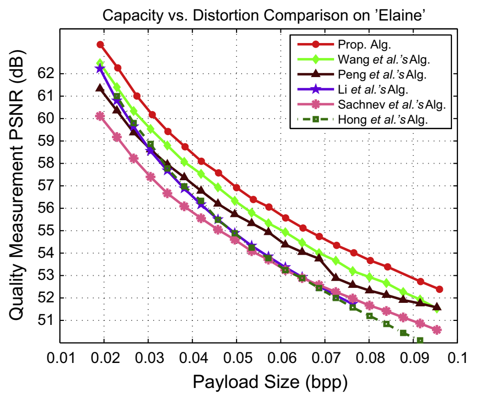
(e) 'Sailboat'



(f) 'Boat'



(g) 'Peppers'



(h) 'Elaine'

Fig. 9. Performance comparisons between the proposed method and following five methods: Wang et al. [50], Peng et al. [47], Li et al. [46], Sachnev et al. [29], Hong et al. [54].

and the corresponding overhead information. Note that the size of the residual payload is $|\mathscr{P}_L| - E_C(T_m)$. Set $|\mathscr{P}_C| = |\mathscr{P}_C| - E_C(T_m)$. Next, go to **Step 1** for next layer embedding.

3.8. Data extraction and image restoration

The detailed process consists of the following three steps.

Step 1 *Extraction of the overhead information*

The LSBs of the watermarked pixels in $S_P \cup O_P$ are collected into a bitstream $\mathscr{B}$ according to the same order as in embedding. By identifying the EOS symbol in $\mathscr{B}$, the bits from the start to the EOS, which comprise the compressed map $\mathscr{L}$, are decompressed by an arithmetic decoder to retrieve the location map. The location map is compressed losslessly by an arithmetic encoder so as to obtain its length, i.e., L_S. Once L_S is known, T_m $(m \in \{1,2\})$, r, c and MT will be extracted one by one according to their fixed lengths. We will easily know the number of embedding layers and the embedded bits for the last layer by extracting EC, respectively.

Note that the data extraction and recovery are processed in an inverse order as in the data embedding process, i.e., $\{P'_N, \ldots, P'_1\}$.

Step 2 *Extraction of the payload* $\mathscr{P}$

For each watermarked block P'_k $(k \in \{N, \ldots, 1\})$, if its location is associated with '0' in the location map, then it is ignored. If its location is associated with '1' and $\Delta < T$, according to the decoding methods described in Sections 3.3, 3.4 and 3.5, extract data and recover the original values from P'_k $(k \in \{N, \ldots, 1\})$.

Step 3 *Retrieve of the original image*

If the number of extracted data bits is equal to the capacity calculated by EC, stop extraction and form $\mathscr{P}$; else go the next layer extraction and repeat steps above. When the current layer is fully extracted, classify the extracted bitstream into data bits and restore the pixels by simple LSB replacement.

4. Experimental results

The 'Lena', 'Baboon' 'Barbara' 'Boat', 'Peppers', 'Airplane', 'Elaine' and 'Sailboat' images with size 512×512 are provided by the authors of paper [46]. Fig. 9 shows performance comparisons between the proposed method and the following five methods: Wang et al. [50], Peng et al. [47], Li et al. [46], Sachnev et al. [29], Hong et al. [54]. The source code of Sachnev et al.'s method is provided by the authors of paper [55]. The source codes of Li et al.'s and Peng et al.'s methods are provided by the authors of paper [46].

In Fig. 9, the capacity range in comparison is limited from an initial capacity of 5000 (or 6000) bits to the maximum capacity of Wang et al.'s method. From Fig. 9, one can observe that our method significantly outperforms Wang et al.'s for every image whatever the ER is. Besides, the proposed method can increase Wang et al.'s ER for the eight test images. Our advantage lies in the utilization of the flexible block-partition and adaptive block-modification strategy. In Wang et al.'s method, the block size is set to 4×4, while in our method, 9 block size $(r, c \in \{3,4,5\})$ candidates are considered, and for one smooth block at a given size, it may be further partitioned into variance-sized sub-blocks. Besides, our method classifies a smooth block into one of three groups: G_1, G_2 and G_3, and more importantly, designs different pixel-modification methods for each group according to the local complexity measurement. The experiments results show our performance is superior to Wang et al's.

By incorporating a sorting technique into the rhombus predictor, Sachnev et al.'s method achieves better performance than some typical schemes such as [27]. However, their method utilizes four nearest neighbors of each pixel to be embedded to estimate the local complexity, and thus the estimation accuracy is weak, compared with that of our method. In addition, our utilized predictor is superior to the rhombus one when the required EC is not high. By means of these two aspects, our method achieves higher performance than Sachnev et al.'s method (see Fig. 9 for details).

Li et al.'s method partitions the host image into pairs and generates predictions in a pair-by-pair manner. In Li et al.'s method, a pixel pair (p_1, p_2) and the prediction of p_2 (i.e., p_3) constitute a three-pixel set $p = (p_1, p_2, p_3)$, so that a difference-pair (d_1, d_2) is obtained, where $d_1 = p_1 - p_2$ and $d_2 = p_2 - p_3$. In this way, a pixel-pair (p_1, p_2) can carry 1-bit data for six of the total ten cases of difference-pair (d_1, d_2), i.e., $d_1 = 0, d_2 = 0, d_1 = 1$ and $d_2 \geqslant -1$, and $d_1 = -1$ and $d_2 < 0$. For the rest of the total ten cases: $d_1 > 1$ and $d_2 \neq 0, d_1 < -1$ and $d_2 < 0, d_1 < 0$ and $d_2 > 0$, and $d_1 = -1$ and $d_2 < -1, (p_1, p_2)$ is shifted so as to avoid ambiguous conduction in data extraction. Two difference values d_1 and d_2 can provide more cases for carrying data, compared with a single difference value d_1 or d_2. Based on this reason above, Li et al.'s method achieve good embedding performance.

Since the maximum (or) minimum is related strongly to the second largest (or smallest) pixel, the predictor used in our method is more accurate than the nearest predictor. When the EC is low, our used predictor can achieve better prediction performance than the rhombus predictor, and thus Peng et al.'s method and ours yield better results than Li et al.'s for most of ten test images. However, for 'Airplane', Li et al.'s method outperforms Peng et al's. That is to say, for 'Airplane', the prediction performance of Li et al.'s method is superior to that of Peng et al's. From Fig. 9(c), one can see that our method obtains a better performance than Li et al's. This is due to the utilizations of the flexible block-partition and the adaptive pixel-modification strategy.

The detailed information for the proposed method, including the numbers of blocks for the groups $(G_1, G_2$ and $G_3)$ in each image and the execution time for embedding and extracting, has been

Table 3
The numbers of blocks in the groups $(G_1, G_2$ and $G_3)$ and the execution time for embedding and extracting on eight test images.

	'Lena'	'Baboon'	'Barbara'	'Airplane'	'Sailboat'	'Boat'	'Peppers'	'Elaine'
G_1^{1st}	0	614	0	258	663	724	0	610
G_1^{2nd}	138	3	215	1	0	0	209	65
G_2^{1st}	2	2400	0	1245	2463	3668	0	2337
G_2^{2nd}	2768	308	3187	65	14	0	3838	146
G_3^{1st}	55	3049	22	1245	2617	3056	10	5038
G_3^{2nd}	3292	1015	1013	288	131	0	4117	126
T_1	1	12	1	1.5	5	6.5	1	6.5
T_2	3.5	6	4	1	1.5	0	5	4
Capacity (bpp)	0.038147	0.022888	0.038147	0.038147	0.038147	0.038147	0.038147	0.038147
PSNR (dB)	60.7729	50.0365	60.7052	63.4796	59.963	58.6884	59.1580	58.6587
Execution time of embedding (seconds)	1.238697	1.393481	1.261608	1.140800	1.263064	0.944198	1.386384	1.374029
Execution time of extraction (seconds)	1.224167	1.382328	1.252328	1.134944	1.256336	0.93960	1.325269	1.352622

Table 4
Comparison of the average time cost on eight test images among Peng et al.'s, Li et al.'s, Sachnev et al.'s and our methods, where the unit of runtime is second.

	'Lena'	'Baboon'	'Barbara'	'Airplane'	'Sailboat'	'Boat'	'Peppers'	'Elaine'
Prop.	1.3051	2.52775	1.7915	1.195203	1.340579	1.020366	1.340757	1.357625
Peng	1.0882	2.9023	1.2680	0.954061	1.346517	1.0709	1.3487	1.217674
Li	5.523377	15.680320	5.988035	4.245556	7.455190	12.355615	10.116945	8.745322
Sachnev	8.559598	8.799692	8.494688	8.290126	8.506015	8.476624	8.414477	8.573356

listed in Table 3, where the block size is 3×5, and a block in G_2 is divided into one 1×6-sized sub-block and one 1×9-sized sub-block. The execution time for embedding (or extraction) refers to the computation time which is spent in the embedding (or extraction) process at the given two thresholds (i.e., T_1 and T_2).

In Table 4, we calculate the average time cost of the proposed method, Peng et al.'s, Sachnev et al.'s and Li et al.'s for eight test images, respectively. Since we do not obtain the source code of Wang et al.'s method, we can not provide the average time of Wang et al.'s method in Table 4. The proposed and the compared algorithms are implemented on Matlab R2013a, and the experiments are run on a personal PC. It can be seen that the runtime of the proposed method varies with images. For a given block size, the time cost of data embedding in our proposed method is mainly composed of the optimal-threshold-determination. Once the optimal thresholds are determined, the followed data embedding is fast. From Table 4, one can observe that our time cost is smaller than those of Li et al.'s and Sachnev et al.'s methods. For some images, although our time cost is slightly larger than that of Peng et al.'s method, it is still acceptable in this age as it can be reduced by a high-performance PC or an effective programming language such as C++.

5. Conclusions

In this paper, a novel RDH method based on flexible block-partition and adaptive pixel-modification strategy is proposed. In our method, more data bits can be embedded into smooth blocks by arbitrarily dividing smooth blocks into smaller sub-blocks. Specifically, for a high-correlation block, it is further divided into several 1×3-sized sub-blocks, and each can be embedded with at most 2 data bits by using TIPVO. For a moderate-correlation block, it is divided into two various-sized sub-blocks, and each can be embedded with at most four data bits by using IPVO. For a low-correlation block, at most two bits can be embedded into it by using IPVO. Extensive experiments verify that the proposed method outperforms Peng et al.'s, Wang et al.'s, Li et al.'s, Sachnev et al.'s and Hong et al.'s works.

Acknowledgment

This work was supported in part by National NSF of China (Nos. 61201393, 61571139), New Star of Pearl River on Science and Technology of Guangzhou (No. 2014J2200085), the Project of Collaborative Innovation and Platform Environment Construction in Guangdong Province (No. 2015B090903017), funds from Department of Science and Technology at Guangdong Province and Guangzhou City (Nos. 2014B090901061, 2015B090901060, 2015B090908001, 2014Y2-00211).

References

[1] Z.H. Xia, X.H. Wang, X.M. Sun, Q.S. Liu, N.X. Xiong, Steganalysis of lsb matching using differences between nonadjacent pixels, Multimed. Tools Appl. 75 (4) (2016) 1947–1962.
[2] Z.H. Xia, X.H. Wang, X.M. Sun, B.W. Wang, Steganalysis of least significant bit matching using multi-order differences, Security Commun. Netw. 7 (8) (2014) 1283–1291.
[3] B.J. Chen, H.Z. Shu, G. Coatrieux, G. Chen, X.M. Sun, J.-L. Coatrieux, Color image analysis by quaternion-type moments, J. Math. Imaging Vision 51 (1) (2015) 124–144.
[4] J. Li, X.L. Li, B. Yang, X.M. Sun, Segmentation-based image copy-move forgery detection scheme, IEEE Trans. Inf. Forensic Secur. 10 (3) (2015) 507–518.
[5] Y.H. Zheng, B. Jeon, D.H. Xu, Q.M. Jonathan Wu, H. Zhang, Image segmentation by generalized hierarchical fuzzy c-means algorithm, J. Intell. Fuzzy Syst. 28 (2) (2015) 961–973.
[6] Z.L. Zhou, Y.L. Wang, Q.M.J. Wu, C.-N. Yang, X.M. Sun, Effective and efficient global context verification for image copy detection, IEEE Trans. Inf. Forensic Secur., <http://dx.doi.org/10.1109/TIFS.2016.2601065>.
[7] Y.Q. Shi, Z. Ni, D. Zou, C. Liang, G. Xuan, Lossless data hiding: fundamentals, algorithms and applications, in: Proceedings of IEEE International Symposium on Circuits and Systems II, 2004, pp. 33–36.
[8] X.L. Li, B. Yang, T.Y. Zeng, Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection, IEEE Trans. Image Process. 20 (12) (2011) 3524–3533.
[9] C.W. Honsinger, P. Jones P., M. Rabbani, J.C. Stoffe, Lossless recovery of an original image containing embedded data, US patent: 6278791W, 2001.
[10] J. Fridrich, M. Goljan, R. Du, Lossless data embedding-new paradigm in digital watermarking, EURASIP J. Appl. Signal Process., vol. 2002, 2002, pp. 185–196.
[11] M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Lossless generalized-lsb data embedding, IEEE Trans. Image Process. 12 (2) (2005) 157–160.
[12] L. Kamstra, H.J.A.M. Heijmans, Reversible data embedding into images using wavelet technique and sorting, IEEE Trans. Image Process. 14 (12) (2005) 2082–2090.
[13] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circ. Syst. Video Technol. 16 (2006) 354–362.
[14] G.R. Xuan, C.Y. Yang, Y.Z. Zhen, Y.Q. Shi, Reversible data hiding using integer wavelet transform and companding technique, Proceedings of IWDW, vol. 5, 2004, pp. 23–26.
[15] X.L. Li, B. Li, B. Yang, T.Y. Zeng, General framework to histogram-shifting-based reversible data hiding, IEEE Trans. Image Process. 22 (6) (2013) 2181–2191.
[16] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circ. Syst. Video Technol. 13 (8) (2003) 890–896.
[17] H.J. Kim, V. Sachnev, Y.Q. Shi, J. Nam, H.G. Choo, A novel difference expansion transform for reversible data embedding, IEEE Trans. Inf. Forensic Secur. 4 (3) (2008) 456–465.
[18] D. Coltuc, J.M. Chassery, Very fast watermarking by reversible contrast mapping, IEEE Signal Process. Lett. 14 (4) (2007) 255–258.
[19] S.W. Weng, Y. Zhao, J.S. Pan, R.R. Ni, Reversible watermarking based on invariability and adjustment on pixel pairs, IEEE Signal Process. Lett. 45 (20) (2008) 1022–1023.
[20] S.W. Weng, Y. Zhao, R.R. Ni, J.S. Pan, Parity-invariability-based reversible watermarking, IET Electron. Lett. 1 (2) (2009) 91–95.
[21] D. Coltuc, Low distortion transform for reversible watermarking, IEEE Trans. Image Process. 21 (1) (2012) 412–417.
[22] A.M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, IEEE Trans. Image Process. 13 (8) (2004) 1147–1156.
[23] X. Wang, X.L. Li, B. Yang, Z.M. Guo, Efficient generalized integer transform for reversible watermarking, IEEE Signal Process. Lett. 17 (6) (2010) 567–570.
[24] X. Wang, X.L. Li, B. Yang, High capacity reversible image watermarking based on integer transform, in: Proceedings of ICIP, 2010, pp. 217–220.
[25] F. Peng, X. Li, B. Yang, Adaptive reversible data hiding scheme based on integer transform, Signal Process. 92 (1) (2012) 54–62.
[26] Z.W. Zhang, L.F. Wu, H.G. Lai, H.B. Li, C.H. Zheng, Double reversible watermarking algorithm for image tamper detection, J. Inform. Hiding Multimedia Signal Process. 7 (3) (2016) 530–542.
[27] D.M. Thodi, J.J. Rodrłguez, Expansion embedding techniques for reversible watermarking, IEEE Trans. Image Process. 16 (3) (2007) 721–730.
[28] Y. Hu, H.K. Lee, J. Li, DE-based reversible data hiding with improved overflow location map, IEEE Trans. Circuits Syst. Video Technol. 19 (2) (2009) 250–260.
[29] V. Sachnev, H.J. Kim, J. Nam, S. Suresh, Y.Q. Shi, Reversible watermarking algorithm using sorting and prediction, IEEE Trans. Circuits Syst. Video Technol. 19 (7) (2009) 989–999.
[30] P.Y. Tsai, Y.C. Hu, H.L. Yeh, Reversible image hiding scheme using predictive coding and histogram shifting, Signal Process. 89 (6) (2009) 1129–1143.
[31] W.L. Tai, C.M. Yeh, C.C. Chang, Reversible data hiding based on histogram modification of pixel differences, IEEE Trans. Circ. Syst. Video Technol. 19 (6) (2009) 906–910.
[32] W. Hong, T.S. Chen, C.W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, J. Syst. Softw. 82 (11) (2009) 1833–1842.

[33] L. Luo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, Reversible image watermarking using interpolation technique, IEEE Trans. Inf. Forensic Secur. 5 (1) (2010) 187–193.
[34] W. Hong, An efficient prediction-and-shifting embedding technique for high quality reversible data hiding, EURASIP J. Adv. Signal Process. (2010).
[35] D. Coltuc, Improved embedding for prediction-based reversible watermarking, IEEE Trans. Inf. Forensic Secur. 6 (3) (2011) 873–882.
[36] X. Gao, L. An, Y. Yuan, D. Tao, X. Li, Lossless data embedding using eneralized statistical quantity histogram, IEEE Trans. Circ. Syst. Video Technol. 21 (8) (2011) 1061–1070.
[37] H.-T. Wu, J.W. Huang, Reversible image watermarking on prediction errors by efficient histogram modification, Signal Process. 92 (12) (2012) 3000–3009.
[38] G. Coatrieux, W. Pan, N. Cuppens-Boulahia, F. Cuppens, C. Roux, Reversible watermarking based on invariant image classification and dynamic histogram shifting, IEEE Trans. Inf. Forensic Secur. 8 (1) (2013) 111–120.
[39] J. Li, X.L. Li, B. Yang, Reversible data hiding scheme for color image based on prediction-error expansion and cross-channel correlation, Signal Process. 93 (9) (2013) 2748–2758.
[40] S. Jung, L. Ha, S. Ko, A new histogram modification based reversible data hiding algorithm considering the human visual system, IEEE Signal Process. Lett. 18 (2) (2011) 95–98.
[41] W. Hong, T. Chen, M. Wu, An improved human visual system based reversible data hiding method using adaptive histogram modification, Opt. Commun. 291 (2013) 87–97.
[42] S.W. Weng, J.S. Pan, Reversible watermarking based on multiple prediction modes and adaptive watermark embedding, Multimed. Tools Appl. 72 (3) (2013) 3063–3083.
[43] Y.H. Chen, H.C. Huang, C.C. Lin, Block-based reversible data hiding with multi-round estimation and difference alteration, Multimed. Tools Appl. (2015) 1–6.
[44] H.C. Huang, Y.Y. Lu, J. Lin, Ownership protection for progressive image transmission with reversible data hiding and visual secret sharing, Optik – Int. J. Light Electron Opt. 127 (15) (2016) 5950–5960.
[45] X.L. Li, J. Li, B. Li, B. Yang, High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion, Signal Process. 93 (1) (2013) 198–205.
[46] X.L. Li, W.M. Zhang, X.L. Gui, B. Yang, A novel reversible data hiding scheme based on two-dimensional difference-histogram modification, IEEE Trans. Inf. Forensic Secur. 8 (7) (2013) 1091–1100.
[47] F. Peng, X.L. Li, B. Yang, Improved pvo-based reversible data hiding, Digit. Signal Process. 25 (2014) 255–265.
[48] X.C. Qu, H.J. Kim, Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding, Signal Process. 111 (2015) 249–260.
[49] W. Hong, T.S. Chen, J. Chen, Reversible data hiding using delaunay triangulation and selective embedment, Inform. Sci. 308 (2015) 140–154.
[50] X. Wang, J. Ding, Q.Q. Pei, A novel reversible image data hiding scheme based on pixel value ordering and dynamic pixel block partition, Inform. Sci. 310 (2015) 16–35.
[51] S.W. Weng, J.S. Pan, Reversible watermarking based on two embedding schemes, Multimed. Tools Appl. (2015).
[52] X.L. Li, W.M. Zhang, X.L. Gui, B. Yang, Efficient reversible data hiding based on multiple histograms modification, IEEE Trans. Inf. Forensic Secur. 10 (9) (2015) 2016–2027.
[53] B. Ou, X.L. Li, Y. Zhao, R.R. Ni, Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion, Signal Process.: Image Commun. 29 (7) (2014) 198–205.
[54] W. Hong, Adaptive reversible data hiding method based on error energy control and histogram shifting, Opt. Commun. 285 (2) (2012) 101–108.
[55] B. Ou, X.L. Li, Y. Zhao, R.R. Ni, Reversible data hiding based on pde predictor, J. Syst. Softw. 86 (10) (2013) 2700–2709.