Signal Processing: *Image Communication* 29 (2014) 760–772

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

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

Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion

Bo Ou [a,c], Xiaolong Li [b], Yao Zhao [a,d,*], Rongrong Ni [a,c]

[a] *Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China*
[b] *Institute of Computer Science and Technology, Peking University, Beijing 100871, China*
[c] *Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China*
[d] *State Key Laboratory of Rail Traffic Control and Safety, Beijing 100044, China*

A R T I C L E I N F O

Article history:
Received 24 May 2013
Received in revised form
15 April 2014
Accepted 18 May 2014
Available online 27 May 2014

Keywords:
Reversible data hiding
Pixel-value-ordering
Prediction-error expansion
Performance optimization

A B S T R A C T

Recently, Li et al. proposed a reversible data hiding (RDH) method based on pixel-value-ordering (PVO) and prediction-error expansion. In their method, the maximum and the minimum of a pixel block are predicted and modified to embed data, and the reversibility is guaranteed by keeping PVO of each block invariant after embedding. In this paper, a novel RDH method is proposed by extending Li et al.'s work. Instead of considering only a single pixel with maximum (or minimum) value of a block, all maximum-valued (or minimum-valued) pixels are taken as a unit to embed data. Specifically, the maximum-valued (or minimum-valued) pixels are first predicted and then modified together such that they are either unchanged or increased by 1 (or decreased by 1) in value at the same time. Comparing our method with Li et al.'s, more blocks suitable for RDH are utilized and image redundancy is better exploited. Moreover, a mechanism of advisable payload partition and pixel-block-selection is adopted to optimize the embedding performance in terms of capacity-distortion behavior. Experimental results verify that our method outperforms Li et al.'s and some other state-of-the-art works.

1. Introduction

Nowadays, data hiding has been found useful in various applications such as authentication, ownership protection, annotation, and secret communication. Traditional data hiding techniques mainly concern on the robustness of embedded data against various attacks, but often cause permanent distortion in the cover medium. However, in some sensitive scenarios such as medical imagery, remote sensing and military imagery, misinterpreting digital content is strictly forbidden, and exactly retrieving the original content as well as the embedded data is required. To this end, reversible data hiding (RDH) technique is proposed to recover the original content from the marked medium without any distortion. Usually, for digital images, the capacity versus image fidelity trade-off is used to evaluate the performance of a RDH algorithm.

So far, RDH has been extensively studied. Existing RDH methods are primarily based on the techniques of lossless compression [1–6], difference expansion (DE) [7–10], histogram shifting (HS) [11–17], prediction-error expansion (PEE) [18–33], integer-to-integer transformation [34–40], etc. Among them, an interesting and valuable research [11,12,41,13,19,14,22,15,31] is to achieve high-level marked image fidelity by modifying each pixel value at most by 1. The first high fidelity RDH method is proposed by Ni et al. [11], where the PSNR is guaranteed over 48.13 dB. In their method, the pixels with the most

* Corresponding author at: Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China. Tel./Fax: +86 10 51688667.
E-mail addresses: 09112055@bjtu.edu.cn (B. Ou), lixiaolong@pku.edu.cn (X. Li), yzhao@bjtu.edu.cn (Y. Zhao), rrni@bjtu.edu.cn (R. Ni).

http://dx.doi.org/10.1016/j.image.2014.05.003
0923-5965/© 2014 Elsevier B.V. All rights reserved.

population in the gray-scale histogram are expanded to embed data, while others are shifted by 1 to prevent ambiguity in data extraction. Later on, Lee et al. [12] proposed to use the residual image for histogram modification. Here, the residual image is obtained by computing the difference of two adjacent pixels. In [14], Wang et al. updated the residual value as the difference between the pixel value and the mean of its two neighboring pixels, and obtained an improvement both in capacity and image fidelity compared with [11,12]. In the other works [19,22], the authors proposed to use prediction-errors for data embedding. They employed advanced predictors by taking half-enclosing causal pixels for prediction, and hence improved the prediction accuracy.

Here, it is worth to mention that, as a supplement to DE/HS/PEE techniques, an effective strategy for distortion reduction, namely sorting or pixel-selection, is proposed and employed in recent RDH methods. Such a strategy is verified to be very helpful to the embedding performance, and can be well incorporated into nowadays RDH algorithms. For example, Sachnev et al. [21] proposed a sorting technique to process prediction-errors with small local variance at first. In another work [26], Li et al. utilized the pixel-selection for adaptive embedding, and embed more bits into a smooth pixel. Besides, in a recent work, Hong [30] proposed an energy estimator to estimate the magnitude of prediction-error from its half-enclosing causal pixels, and preferentially use the ones with small magnitude for data embedding.

Recently, Li et al. [31] proposed a novel RDH based on pixel-value-ordering (PVO). For this method, the pixels in a block are first sorted in an ascending order to get $(p_1, \ldots, p_n)$. Then, the maximum p_n is predicted by p_{n-1}. Finally, the pixel with the prediction-error of $p_n - p_{n-1} = 1$ is embedded with one data bit. Besides, in [31], by also considering the minimum p_1, i.e., predict p_1 using p_2, two bits can be embedded into a block at the same time. The experimental results reported in [31] show that the prediction using sorted pixel values is more accurate than the previous methods such as MED (median-edge-detector, [42,20,22]), GAP (gradient-adjusted-prediction, [43,19,26]) and the mean-value-predictor [21]. As a result, compared with the previous works [20,21,23,26], the marked image fidelity can be significantly improved by [31]. In addition, the pixel-selection technique is also utilized in this method to enhance the embedding performance.

The PVO-based embedding [31] has two major advantages. One is that the prediction derived from sorted pixel values is more accurate than the previous ones. The other is that, for low capacity case, larger sized blocks can be utilized to better exploit image redundancy. However, the potential benefit of PVO is not fully exploited by this method since some blocks suitable for RDH are not utilized. Taking the maximum case for example, the block with $p_n - p_{n-1} = 0$ is abandoned in [31]. In fact, these blocks can also be utilized to embed data using a similar mechanism as that of [31]. Based on the above consideration, we argue that the PVO-based embedding can be further improved.

In this paper, we extend the PVO-based embedding [31] into a more general form in which the maximum-valued (minimum-valued) pixels are considered together as a unit for data embedding. As an extension to [31], for each block, we use the second maximum- and minimum-valued pixels to predict maximum- and minimum-valued ones, respectively, and then simultaneously modify them to keep PVO invariant. For example, when the conditions $p_n - p_{n-1} = 0$ and $p_{n-1} > p_{n-2}$ hold, the block is ignored in [31] while it can be exploited in our method. In such a case, for our method, p_{n-2} is employed to predict p_{n-1} and p_n, and the maximum-valued pixels p_{n-1} and p_n will be modified together for data embedding, i.e., they are either unchanged or increased by 1 at the same time. In this way, more blocks can be utilized to embed data and the capacity is thus increased. Moreover, since the numbers of maximum- and minimum-valued pixels for each block remain unchanged, data embedding on different blocks can be classified into categories, and an optimized embedding to obtain the best image fidelity by considering the embedding payoff can be achieved. Experimental results demonstrate that both the capacity and the marked image fidelity can be improved compared with the PVO-based embedding [31]. Besides, in comparison with the state-of-the-art works [21,30], the proposed method also demonstrates a superior performance.

The remainder of this paper is organized as follows. In Section 2, by extending the PVO-based embedding [31], according to a parameter k, a new reversible embedding strategy namely PVO-k is introduced. PVO-k includes the original PVO-based embedding as a special case if taking $k=1$. Here, in Section 2, only the maximum-modification-based embedding is introduced to describe our idea. Then, by considering both the maximum and the minimum and by combining PVO-k with two parameters $k=1$ and $k=2$, a novel RDH scheme is proposed in Section 3. Moreover, to optimize the embedding performance in terms of capacity-distortion behavior, a mechanism of advisable payload partition and pixel-block-selection is also proposed in this section. The experimental results are given in Section 4. Finally, Section 5 concludes this paper.

2. A new PVO-based reversible embedding strategy

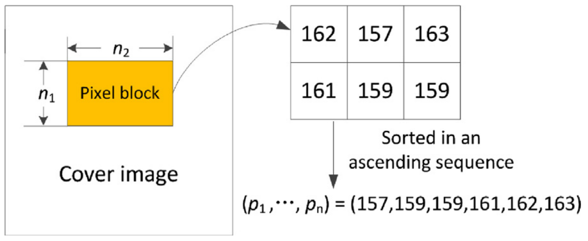
2.1. Related work: PVO-based embedding [31]

In Li et al.'s method [31], both the maximum and minimum pixel values in a block are utilized for data embedding. For the sake of simplicity, we only take the maximum-modification-based embedding phase for illustration. The main idea of this method can be summarized as follows:

- The cover image is first divided into non-overlapped blocks with size $n_1 \times n_2$ (see Fig. 1). For each block, the pixels are sorted in an ascending order to get $(p_1, \ldots, p_n)$, where $n = n_1 \times n_2$.
- Then, the maximum p_n is predicted by p_{n-1}. The derived prediction-error x is

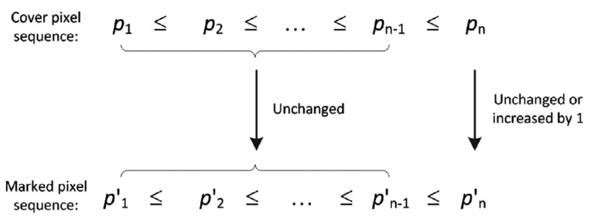
$$x = p_n - p_{n-1} \geq 0. \quad (1)$$

- To keep PVO invariant, as shown in Fig. 2, p_i is unchanged for each $i \in \{1, \ldots, n-1\}$, while the maximum



n_2
n_1
Pixel block
Cover image
162 157 163
161 159 159
Sorted in an ascending sequence
$(p_1, \cdots, p_n)$ = (157,159,159,161,162,163)

Fig. 1. A pixel block with ascending values.



Cover pixel sequence:
$p_1 \le p_2 \le \dots \le p_{n-1} \le p_n$
Unchanged
Unchanged or increased by 1
Marked pixel sequence:
$p'_1 \le p'_2 \le \dots \le p'_{n-1} \le p'_n$

Fig. 2. PVO-based embedding [31] by modifying the maximum.

p_n is modified to get the marked pixel value p'_n by taking

$$p'_n = \begin{cases} p_n & \text{if } x=0 \\ p_n+m & \text{if } x=1 \\ p_n+1 & \text{if } x>1 \end{cases} \tag{2}$$

where $m \in \{0,1\}$ is a to-be-embedded bit. Here, the prediction-errors $x=1$ are used to carry data, $x>1$ are shifted to create vacancy, and $x=0$ are discarded in data embedding. So, the capacity is the number of pixels with prediction-error of 1. It is worth noting that prediction-error 1 is the peak bin in the PVO-based prediction-error histogram (see Fig. 1 of [31]).

At decoder side, for each block, one can use the marked prediction-error $x' = p'_n - p'_{n-1}$ to recover the original maximum p_n as

$$p_n = \begin{cases} p'_n & \text{if } x'=0 \text{ or } 1 \\ p'_n-1 & \text{if } x'>1 \end{cases} \tag{3}$$

and a data bit is extracted as “0” and “1” when x' is 1 and 2, respectively. Notice that here the reversibility and the data extraction are guaranteed by invariant PVO and the fact $p'_{n-1} = p_{n-1}$.

2.2. PVO-k embedding

As mentioned before, in the conventional PVO-based embedding, the blocks with prediction-error of 0, i.e., the ones with $p_n = p_{n-1}$, cannot be utilized. In fact, these blocks can also be exploited using a new prediction by generalizing (1). Specifically, we can take the second largest value in the sorted sequence to predict the maximum-valued ones. For example, for a block with $p_n = p_{n-1}$ and $p_{n-1} > p_{n-2}$, we may use p_{n-2} to predict p_{n-1} and p_n to generate a prediction-error $x = p_{n-1} - p_{n-2} > 0$. The difference in modifying the blocks between the proposed method and the conventional PVO-based embedding is that, to guarantee the PVO invariant, we have to simultaneously modify the maximum-valued pixels p_n and p_{n-1} such that they are either unchanged or increased by 1 at the same time.

Now, based on the aforementioned discussion and by extending the conventional PVO-based embedding, we present a new reversible embedding strategy namely PVO-*k*. Assume that

$$p_1 \le \cdots \le p_{n-k} < p_{n-k+1} = \cdots = p_n \tag{4}$$

where k is the number of maximum-valued pixels. For data embedding, we will modify all the maximum-valued pixels $\{p_{n-k+1}, \dots, p_n\}$ while keeping the others unchanged. That is to say, as shown in Fig. 3, for each $i \in \{n-k+1, \dots, n\}$, the marked value of p_i, p'_i, is obtained as

$$p'_i = \begin{cases} p_i+m & \text{if } x=1 \\ p_i+1 & \text{if } x>1 \end{cases} \tag{5}$$

where the prediction-error x is

$$x = p_{n-k+1} - p_{n-k} \ge 1 \tag{6}$$

and $m \in \{0,1\}$ is a to-be-embedded bit.

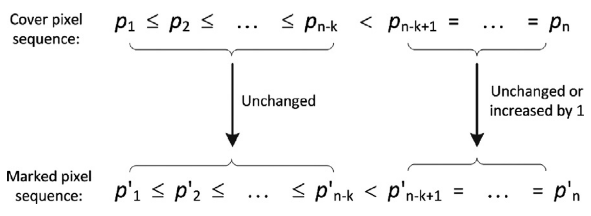
The key issue of PVO-*k* is that the maximum-valued pixels $\{p_{n-k+1}, \dots, p_n\}$ are also the maximum value after data embedding. The extraction process of PVO-*k* is similar to that of the conventional PVO-based embedding and is omitted here.

Clearly, PVO-1 is just the conventional PVO-based embedding since only the blocks with $p_n > p_{n-1}$ are utilized in this case. For $k_1 \ne k_2$, the two embeddings PVO-k_1 and PVO-k_2 can be independently processed as they are applied to different blocks. An example illustrating the independent embedding process is given in Fig. 4 in which PVO-1 and PVO-2 are implemented for 2×2 sized blocks. One can observe that, by identifying the number of maximum-valued pixels, the blocks with $k \in \{1,2\}$ are accordingly processed by PVO-1 and PVO-2, respectively, while the blocks with $k>2$ are skipped without any modification.

In PVO-*k*, the average distortion in terms of mean square error (MSE) for a block is

$$\sum_{i=1}^{n} (p_i - p'_i)^2 = \sum_{i=n-k+1}^{n} (p_i - p'_i)^2 = \begin{cases} k/2 & \text{if } x=1 \\ k & \text{if } x>1 \end{cases} \tag{7}$$

which is closely related to the number of maximum-valued pixels. The larger the *k*, the larger the distortion caused by embedding. In this light, PVO-1 is better than PVO-2 when embedding data into a given block. However, for a block sequence, the trade-off between capacity and distortion is more complex since the distortion of PVO-*k* depends on the specific distribution of the corresponding prediction-error histogram.



Cover pixel sequence:
$p_1 \le p_2 \le \dots \le p_{n-k} < p_{n-k+1} = \dots = p_n$
Unchanged
Unchanged or increased by 1
Marked pixel sequence:
$p'_1 \le p'_2 \le \dots \le p'_{n-k} < p'_{n-k+1} = \dots = p'_n$

Fig. 3. PVO-*k* embedding by modifying the maximum-valued pixels.

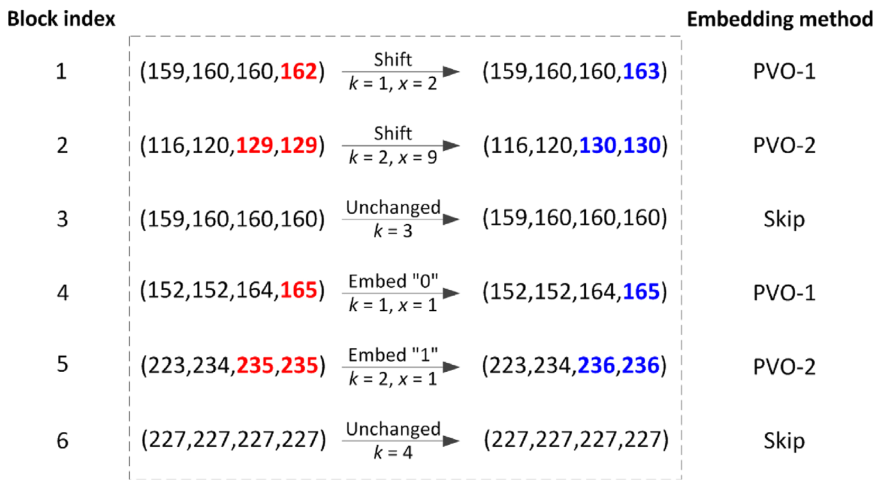


Fig. 4. Combined embedding of PVO-1 and PVO-2 for six 2×2 blocks.

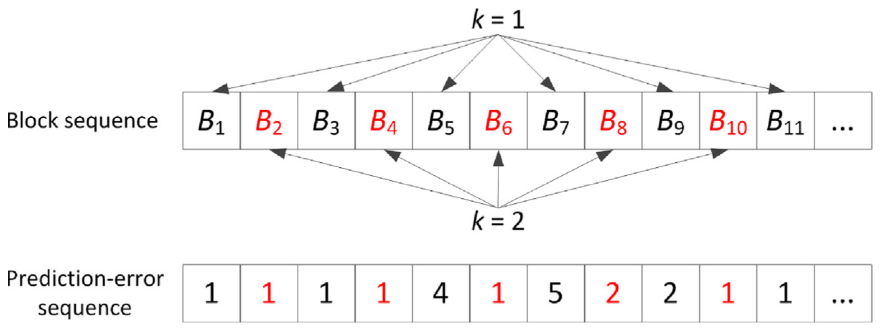

k = 1
Block sequence
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 ...
k = 2
Prediction-error sequence
1 1 1 1 4 1 5 2 2 1 1 ...

Fig. 5. A block sequence and the corresponding prediction-error sequence.

We point out that PVO-k can introduce a less distorted embedding even though k is larger. For example (see Fig. 5), consider a block sequence $(B_1, B_2, \ldots)$ and the corresponding prediction-error sequence $(1, 1, \ldots)$ computed by (6). Suppose that, for the first 11 blocks, k is 1 for the ones with indices 1, 3, 5, 7, 9 and 11, while k is 2 for the others. We want to embed 3 bits into these blocks. Consider the following three cases:

- If using PVO-1 only, the six blocks with indices 1, 3, 5, 7, 9 and 11 should be used and the distortion is $0.5+0.5+1+1+1+0.5=4.5$.
- If using PVO-2 only, the three blocks with indices 2, 4 and 6 should be used and the distortion is $1+1+1=3$.
- If using PVO-1 and PVO-2 together, the first three blocks are sufficient to embed 3 bits and the distortion is $0.5+1+0.5=2$.

From this simple example, one can see that according to specific distribution of prediction-error sequence, a better result can be obtained by PVO-k with a larger k, and the best performance can be achieved if using PVO-k simultaneously with different k.

Inspirited by the above observation, we propose to consider combined embedding to achieve an optimized performance. Here, for simplicity, only the combination of PVO-1 and PVO-2 is taken into account in this paper.

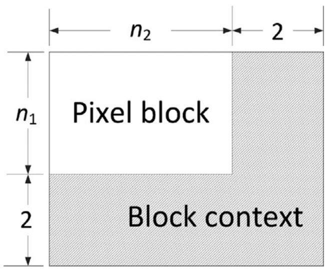


Fig. 6. A $n_1 \times n_2$ sized block with its context (shadow pixels).

2.3. Optimal combined embedding

We now define a notion of embedding payoff to measure PVO-k, and then present the optimized combined embedding using PVO-1 and PVO-2. We first give a complexity measurement as the control parameter to determine whether the smoothness of a block makes it a candidate or not for data embedding. Here, the block context is defined as shown in Fig. 6. Then, the complexity of a block is computed as the sum of the vertical and the horizontal absolute difference of every two adjacent pixels in the context. The block complexity is unchanged after data embedding since the blocks are scanned and processed sequentially at encoder and in the inverse order at decoder.

In the conventional PVO-based embedding, only the blocks with a complexity less than a pre-defined threshold are processed while the others are skipped. This may better exploit the image redundancy to obtain an improved performance. The complexity measurement also plays a key role in our method. We will take two different thresholds for PVO-1 and PVO-2 to generate two prediction-error histograms, and then embed the required payload into the two histograms with an advisable payload partition. The strategy provides a way to optimize the combined embedding.

For convenience, a block with k maximum-valued pixels is called a type-k block. Suppose $B_1, \ldots, B_{N_k}$ are all type-k blocks, $x_i \geq 1$ is the prediction-error of B_i computed using (6), and c_i is the complexity of B_i. We define a function h_k to count the prediction-errors and complexity of type-k blocks as follows:

$$h_k(e,t) = \sharp\{1 \leq i \leq N_k : x_i = e, c_i = t\} \tag{8}$$

where $\sharp$ denotes the cardinal number of a set.

We then take two thresholds t_1 and t_2, and, for $k \in \{1,2\}$, implement PVO-k embedding for each type-k block if its complexity is less than t_k. In this case, ignoring the underflow/overflow problem, the capacity (denoted as E_k) of PVO-k is

$$E_k(t_k) = \sum_{t=0}^{t_k-1} h_k(1,t) \tag{9}$$

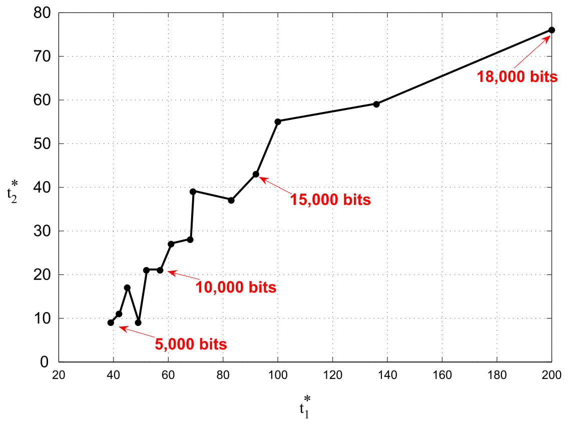


Fig. 7. Optimal thresholds (t_1^*, t_2^*) for different capacities (from 5000 to 18,000 bits, with a step-size of 1000), for Lena image with block size 2×2.

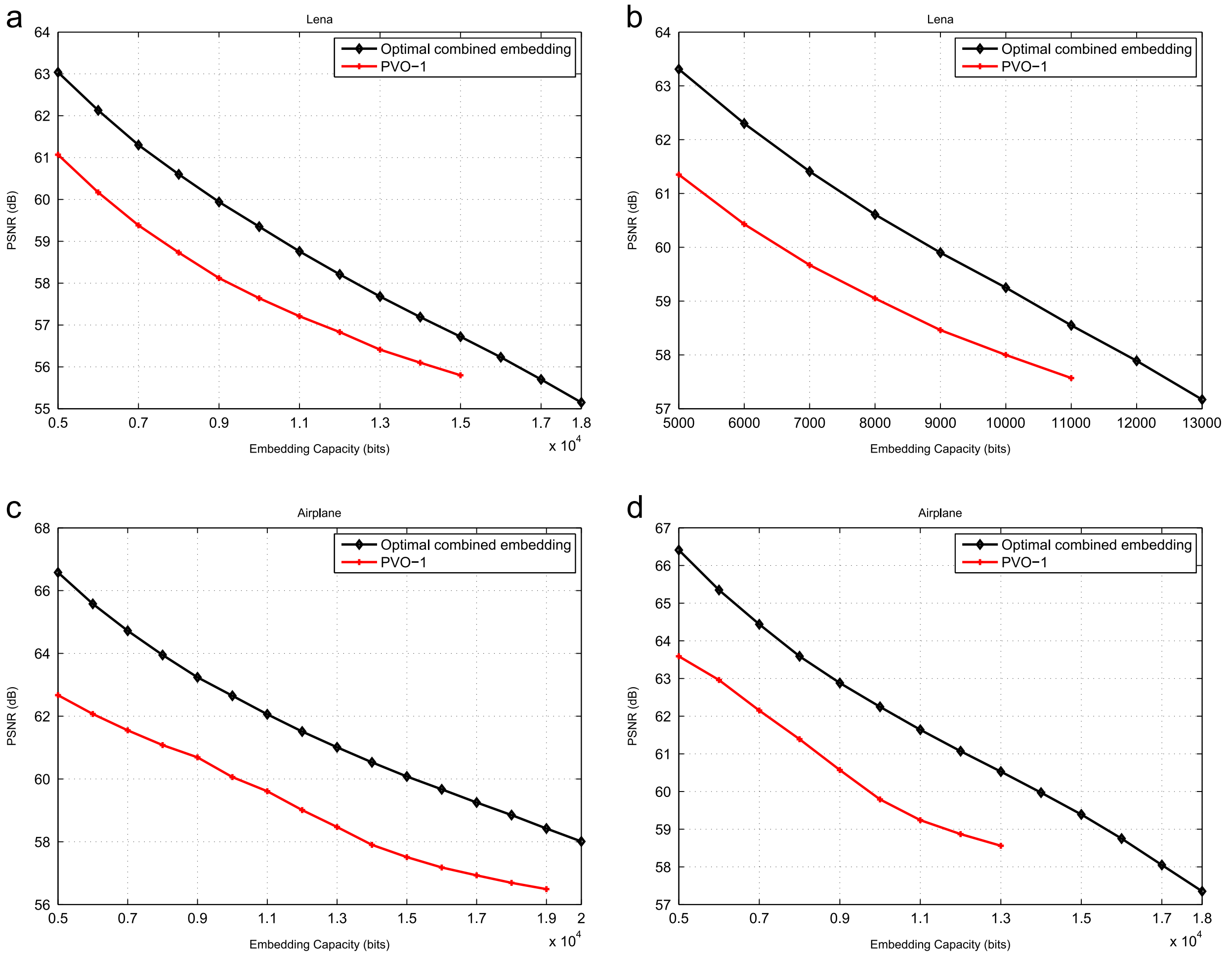


Fig. 8. Performance comparison for the optimal combined embedding and PVO-1. (a) $(n_1, n_2) = (2,2)$; (b) $(n_1, n_2) = (2,3)$; (c) $(n_1, n_2) = (2,2)$ and (d) $(n_1, n_2) = (2,3)$.

and the expected value of the corresponding distortion (denoted as D_k) in terms of MSE can be formulated by

$$D_k(t_k) = \frac{k}{2}\sum_{t=0}^{t_k-1} h_k(1,t) + k\sum_{t=0}^{t_k-1}\sum_{e>1} h_k(e,t). \tag{10}$$

With (9) and (10), the payoff of combined embedding is defined by

$$P(t_1,t_2) = \frac{D_1(t_1)+D_2(t_2)}{E_1(t_1)+E_2(t_2)}. \tag{11}$$

It means the MSE payoff for embedding 1 bit into the cover image using PVO-1 and PVO-2 with thresholds (t_1,t_2), respectively. Finally, for a given capacity EC, the optimal thresholds (t_1^*,t_2^*) are selected as follows to minimize the embedding payoff:

$$\begin{cases} \arg\min\limits_{0\le t_1,t_2\le c_{max}} \; P(t_1,t_2) \\ \text{subject to } E_1(t_1)+E_2(t_2)\ge EC \end{cases} \tag{12}$$

where c_{max} is the maximum of block complexity.

For a better illustration, Fig. 7 shows the optimal thresholds (t_1^*,t_2^*) determined using (12) for different capacities. One can observe that both t_1^* and t_2^* are strictly positive integers. This means that both PVO-1 and PVO-2 contribute to the optimal combined embedding. On the other hand, in most cases, both t_1^* and t_2^* increase when capacity increases, and $t_1^* > t_2^*$ holds for every tested case. However, it is difficult to derive an analytical relationship between t_1^* and t_2^*, and we prefer to solve the optimization problem (12) with naive exhaustive search.

Finally, we show in Fig. 8 the performance comparison for the optimal combined embedding and PVO-1. Recall here that by using PVO-1 only, it is just the conventional PVO-based embedding. According to this figure, it is clear that the optimal combined embedding is better than PVO-1 with a higher PSNR and a larger maximum capacity.

3. Proposed RDH scheme

In this section, a new RDH scheme based on the optimal combined embedding by using both maximum and minimum is presented. The section is organized as follows: we first describe in Section 3.1 the extension of PVO-*k* by modifying both maximum- and minimum-valued pixels. Then, for the extended PVO-*k*, a mechanism of optimal-threshold-determination for the combined embedding of PVO-1 and PVO-2 is given in Section 3.2. Finally, Section 3.3 gives the detailed embedding and extracting procedures of the proposed scheme.

3.1. Extended PVO-*k* embedding by modifying both maximum- and minimum-valued pixels

By considering both maximum- and minimum-valued pixels of a block, we extend the PVO-*k* embedding introduced in Section 2.2. First, for a block with ascending pixel values $(p_1,\ldots,p_n)$, if $p_n > p_1$, we define

$$p_1 = \cdots = p_b < p_{b+1} \le \cdots \le p_{n-a} < p_{n-a+1} = \cdots = p_n \tag{13}$$

where a and b are numbers of maximum- and minimum-valued pixels, respectively. Notice that the most flat blocks with identical values (i.e., $p_1 = \cdots = p_n$) will be ignored in our method. In this way, we have $a > 0$, $b > 0$ and $a+b \le n$. Then, we use p_{n-a} to predict p_{n-a+1}, and p_{b+1} to predict p_b. The corresponding prediction-errors are

$$x = p_{n-a+1} - p_{n-a} \ge 1 \quad \text{and} \quad y = p_b - p_{b+1} \le -1. \tag{14}$$

If $a=k$ or $b=k$, the block will be used for data embedding according to the following four cases (see Fig. 9 for an illustration):

- *Case* 1: $a=k, b\ne k$. In this case, only the maximum-valued pixels will be modified, i.e., for each $i\in\{1,\ldots,n\}$, the marked value of p_i is determined as

$$p_i' = \begin{cases} p_i & \text{if } 1\le i\le n-a \\ p_i+m & \text{if } n-a+1\le i\le n \text{ and } x=1 \\ p_i+1 & \text{if } n-a+1\le i\le n \text{ and } x>1 \end{cases} \tag{15}$$

  where x is the prediction-error computed according to (14) and $m\in\{0,1\}$ is a to-be-embedded bit. Here, one bit will be embedded into the block if the prediction-error of maximum is 1.
- *Case* 2: $a\ne k, b=k$. In this case, only the minimum-valued pixels will be modified, i.e., p_i' is determined as

$$p_i' = \begin{cases} p_i-m & \text{if } 1\le i\le b \text{ and } y=-1 \\ p_i-1 & \text{if } 1\le i\le b \text{ and } y<-1 \\ p_i & \text{if } b+1\le i\le n \end{cases} \tag{16}$$

  where y is the prediction-error computed according to (14). Here, one bit will be embedded into the block if the prediction-error of minimum is -1.

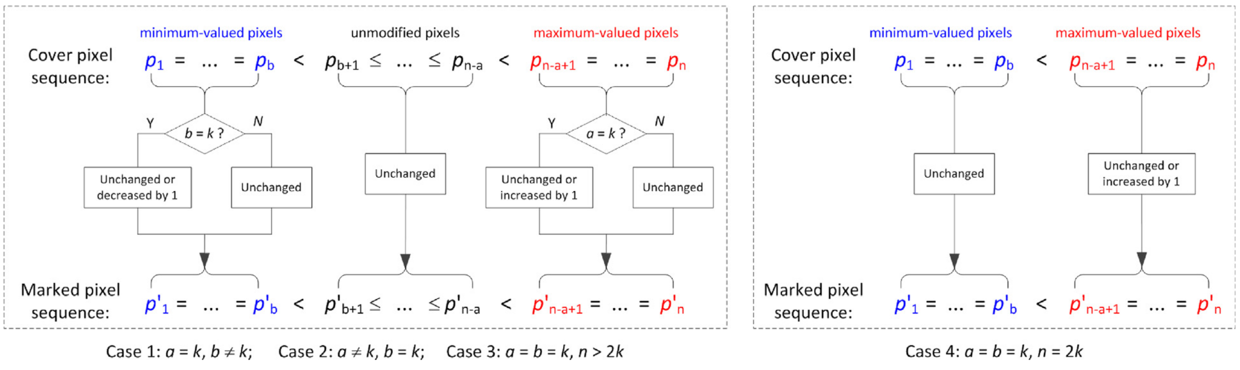

minimum-valued pixels
unmodified pixels
maximum-valued pixels
Cover pixel sequence:
p_1 = … = p_b < p_{b+1} ≤ … ≤ p_{n-a} < p_{n-a+1} = … = p_n
b = k ?
a = k ?
Y
N
Unchanged or decreased by 1
Unchanged
Unchanged or increased by 1
Marked pixel sequence:
p'_1 = … = p'_b < p'_{b+1} ≤ … ≤ p'_{n-a} < p'_{n-a+1} = … = p'_n
Case 1: a = k, b ≠ k; Case 2: a ≠ k, b = k; Case 3: a = b = k, n > 2k
Case 4: a = b = k, n = 2k

Fig. 9. Extended PVO-*k* by using both maximum- and minimum-valued pixels.

- *Case* 3: $a=b=k$ and $n>2k$. Here, both maximum- and minimum-valued pixels will be modified, i.e., p'_i is determined as

$$p'_i = \begin{cases} p_i - m_1 & \text{if } 1 \le i \le b \text{ and } y = -1 \\ p_i - 1 & \text{if } 1 \le i \le b \text{ and } y < -1 \\ p_i & \text{if } b+1 \le i \le n-a \\ p_i + m_2 & \text{if } n-a+1 \le i \le n \text{ and } x = 1 \\ p_i + 1 & \text{if } n-a+1 \le i \le n \text{ and } x > 1 \end{cases} \tag{17}$$

  where $m_1, m_2 \in \{0,1\}$ are two to-be-embedded bits. In this case, at most two bits will be embedded into the block.
- *Case* 4: $a=b=k$ and $n=2k$. In this case, each pixel in the block is either a maximum or a minimum, and the maximum is in fact predicted by the minimum (and vice versa). For this special case, since the prediction should be the same for both encoder and decoder, both the maximum- and minimum-valued pixels cannot be modified. Then, in our method, only the maximum-valued pixels are modified while the minimum-valued pixels are used for prediction, and we use (15) to determine marked values.

Since PVO of each block is invariant after data embedding, at decoder, the marked values can be sorted in the same ascending order as $(p'_1, \dots, p'_n)$, and we have, the same as (13),

$$p'_1 = \cdots = p'_b < p'_{b+1} \le \cdots \le p'_{n-a} < p'_{n-a+1} = \cdots = p'_n. \tag{18}$$

Thus (a,b) can be utilized to determine whether the block is processed or not in the recovery process. On the other hand, as the causal pixels used for prediction are unmodified, the same prediction can be obtained as well. In data extraction of extended PVO-k, like the embedding process, it also has four cases:

- *Case* 1: $a=k, b\ne k$. In this case, the original value of p'_i can be recovered as

$$p_i = \begin{cases} p'_i & \text{if } 1 \le i \le n-a \\ p'_i & \text{if } n-a+1 \le i \le n \text{ and } x' = 1 \\ p'_i - 1 & \text{if } n-a+1 \le i \le n \text{ and } x' > 1 \end{cases} \tag{19}$$

  where the marked prediction-error of maximum is computed as $x' = p'_{n-a+1} - p'_{n-a}$, and the embedded data bit is “0” and “1” if x' is 1 and 2, respectively.
- *Case* 2: $a \ne k, b=k$. In this case, the original value of p'_i is

$$p_i = \begin{cases} p'_i & \text{if } 1 \le i \le b \text{ and } y' = -1 \\ p'_i + 1 & \text{if } 1 \le i \le b \text{ and } y' < -1 \\ p'_i & \text{if } b+1 \le i \le n \end{cases} \tag{20}$$

  where $y' = p'_b - p'_{b+1}$ is the marked prediction-error of minimum. The embedded data bit is “0” and “1” if y' is -1 and -2, respectively.
- *Case* 3: $a=b=k$ and $n>2k$. Here, the original value of p'_i is

$$p_i = \begin{cases} p'_i & \text{if } 1 \le i \le b \text{ and } y' = -1 \\ p'_i + 1 & \text{if } 1 \le i \le b \text{ and } y' < -1 \\ p'_i & \text{if } b+1 \le i \le n-a \\ p'_i & \text{if } n-a+1 \le i \le n \text{ and } x' = 1 \\ p'_i - 1 & \text{if } n-a+1 \le i \le n \text{ and } x' > 1 \end{cases} \tag{21}$$

  and the embedded data (two bits) is “00”, “01”, “10” and “11” if (x', y') is $(1,-1)$, $(1,-2)$, $(2,-1)$ and $(2,-2)$, respectively.
- *Case* 4: $a=b=k$ and $n=2k$. In this case, the block recovery and the data extraction are exactly the same as Case 1.

3.2. Optimal-threshold-determination for combined embedding

As mentioned before, to optimize the combined embedding, we have to first classify the block types and then estimate the capacity and the expected distortion. Here, since maximum- and minimum-valued pixels in a block are involved, a block may be utilized both in extended PVO-1 and PVO-2. Although we cannot specify the block type for a given block, it does not affect embedding and extraction on the pixels since each pixel would be modified once at most. In this situation, the estimation of the capacity $E_k(t)$ and the expected distortion $D_k(t)$ for extended PVO-k is given in Algorithm 1, where t is the block-selection-threshold.

Algorithm 1. Calculation of capacity $E_k(t)$ and expected distortion $D_k(t)$ for extended PVO-k, where t is the block-selection-threshold.

```
Data Divide image into $N$ non-overlapped blocks with size $n_1 \times n_2$,
    set $n = n_1 \times n_2$. Initialize $E_k(t)=0, D_k(t)=0$.
for $i \leftarrow 1$ to $N$ then
  Sort the values in the $i$–th block to get $(p_1, \dots, p_n)$,
  and count the numbers of maximum– and minimum – valued
  pixels $a$ and $b$. Compute its complexity $c$ and
  prediction – errors $x$ and $y$ according to (14).
  if $c < t$ and $p_n - p_1 > 0$ then
    if $a = k, b \ne k$ then
      if $x = 1$ then $E_k(t) = E_k(t)+1, D_k(t) = D_k(t)+k/2$.
      else $D_k(t) = D_k(t)+k$
    end
    if $a \ne k, b = k$ then
      if $y = 1$then $E_k(t) = E_k(t)+1, D_k(t) = D_k(t)+k/2$.
      else $D_k(t) = D_k(t)+k$
    end
    if $a = b = k, n > 2k$ then
      if $x = 1$ then $E_k(t) = E_k(t)+1, D_k(t) = D_k(t)+k/2$.
      else $D_k(t) = D_k(t)+k$
      if $y = 1$ then $E_k(t) = E_k(t)+1, D_k(t) = D_k(t)+k/2$.
      else $D_k(t) = D_k(t)+k$
    end
    if $a = b = k$ and $n = 2k$ then
      if $x = 1$ then $E_k(t) = E_k(t)+1, D_k(t) = D_k(t)+k/2$.
      else $D_k(t) = D_k(t)+k$
    end
  end
end
```

Then, similar to (12), take the underflow/overflow problem into account, the optimal thresholds (t_1^*, t_2^*) for the combined embedding of extended PVO-1 and PVO-2 are determined as follows:

$$\begin{cases} \arg\min\limits_{0 \le t_1, t_2 \le c_{max}} \dfrac{D_1(t_1)+D_2(t_2)}{E_1(t_1)+E_2(t_2)} \\ \text{subject to } E_1(t_1)+E_2(t_2) \ge EC+L+78 \end{cases} \tag{22}$$

where $EC+L+78$ denotes the total payload size including the required capacity EC, the compressed location map of size L, and the auxiliary information size 78 contains (suppose for simplicity that the image size is 512×512)

- value of EC (18 bits),
- values of block size n_1 and n_2 (4 bits),
- size of the compressed location map L (18 bits)
- values of optimal thresholds t_1^* and t_2^* (20 bits),
- the block index where data embedding ends (18 bits).

Notice that the location map construction is straightforward in our method. We adjust pixels with value 0 and 255 to 1 and 254, respectively, and then mark the locations of these pixels with “1” in the location map. Accordingly, the other pixels are marked with “0”. After construction, the location map is losslessly compressed to reduce its size.

Furthermore, to verify that (22) is effective in constructing the optimal combined PVO-1 and PVO-2 embedding, we stimulate all the results by using different (t_1, t_2) as shown in Fig. 10 for the capacity of 10,000 bits. Here, the block size is specified as 2×2 and 2×3. The combinations of (t_1, t_2) are sorted in terms of t_1 in an ascending order. For a given t_1, we traverse over all values of t_2, and report the optimal one that yields the highest PSNR among them. It is clear that (t_1^*, t_2^*) derived from (22) yields the highest PSNR whatever the block size and the image are. In fact, based on our experiments, the optimal performance can be obtained by using (22) in most of the cases.

3.3. Embedding and extracting procedures

For a given block size $n_1 \times n_2$, after the optimal thresholds (t_1^*, t_2^*) are determined and the location map constructed, the detailed data embedding procedure is given below. First, divide the cover image into non-overlapped blocks sized $n_1 \times n_2$. For each block, count the numbers of maximum- and minimum-valued pixels a and b, and compute its complexity c. Then, we embed the secret message into the cover image. If $a=1$ or $b=1$, embed data into the block using extended PVO-1 if $c<t_1^*$; if $a=2$ or $b=2$, embed data into the block using extended PVO-2 if $c<t_2^*$; otherwise, skip the block and move to the next one. After the message is embedded, embed the least significant bits (LSB) of the first $L+78$ pixels into the rest part of image. Finally, replace the LSB of the first $L+78$ pixels by

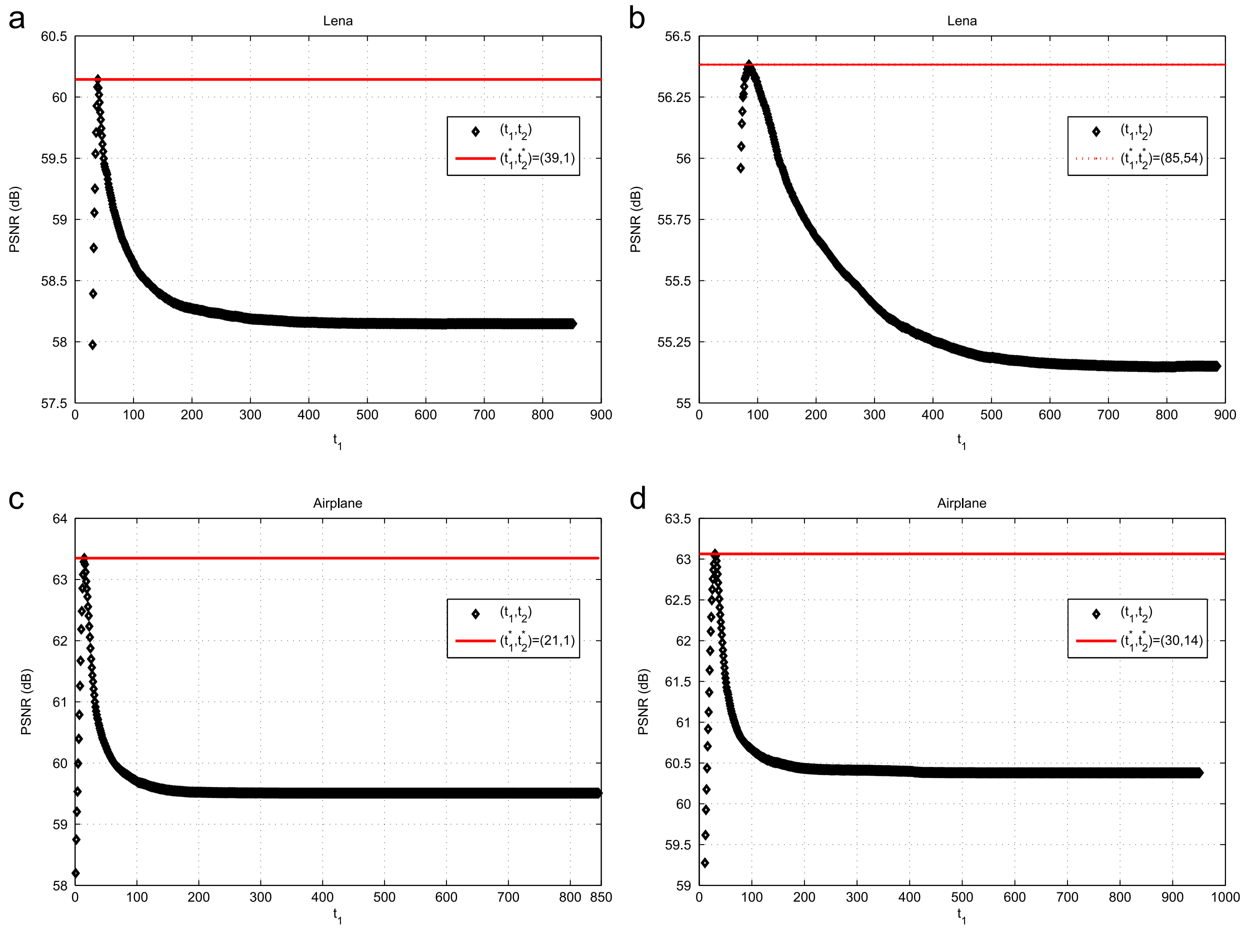


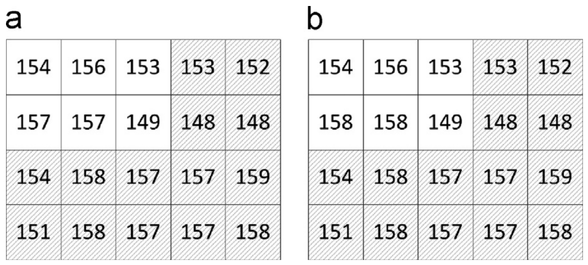
Fig. 10. Performance comparison for a capacity of 10,000 bits on Lena and Airplane images by using different t_1, t_2, where the block sizes are 2×2 and 2×3, respectively. (a) $(n_1, n_2) = (2, 2)$; (b) $(n_1, n_2) = (2, 3)$; (c) $(n_1, n_2) = (2, 2)$ and (d) $(n_1, n_2) = (2, 3)$.

the auxiliary information and the compressed location map. The marked image is then generated.

The corresponding data extraction procedure is described as follows. First, extract the auxiliary information by reading the LSB of the first 78 pixels. Then, extract the compressed location map by reading the LSB of next L pixels. Next, from the end position, in a reverse order, extract the embedded LSB sequence of length $L+78$. Specifically, for each block, count the numbers of maximum- and minimum-valued pixels a and b, and compute its complexity c. If $a=1$ or $b=1$, extract data and restore the original block using the inverse of extended PVO-1 if $c<t_1^*$; if $a=2$ or $b=2$, extract data and restore the original block using the inverse of extended PVO-2 if $c<t_2^*$; otherwise, there is no hidden data and the block is recovered as itself. After that, replace the LSB of the first $L+78$ pixels by the extracted LSB sequence, and then extract the embedded message and realize the image restoration for the unprocessed pixels. Finally, according to the decompressed location map, revise the pixel with value 1 and 254 to 0 and 255, respectively, if it is marked with "1" in the location map. In this way, both the embedded message and the original image are recovered.

We remark that, for a better performance, the block size for optimal combined embedding can also be adaptively determined. The same as Li et al.'s work [31], 16 cases for $n_1, n_2 \in \{2,3,4,5\}$ are taken as candidates, and the one yielding the highest PSNR is selected as the final block size for implementing the data embedding procedure.

Finally, for better illustration, an embedding/extraction example of pixel block is given in Fig. 11. Here, suppose that the block size is 2×3, the to-be-embedded message sequence is "10…", and the optimal thresholds using (22) are $(t_1^*, t_2^*)=(35,60)$. The cover block and its context are shown in Fig. 11(a). For PVO-k embedding, the pixels are first sorted into an ascending order as $(p_1,\ldots,p_6)=(149,153,154,156,157,157)$. The numbers of maximum- and minimum-valued pixels a and b are 2 and 1, respectively, and the complexity c derived from its context is 50. The prediction-errors x and y for maximum- and minimum-valued pixels are $x=157-156=1$ and $y=149-153=-4$, respectively. Since $c>t_1^*$ and $c<t_2^*$, the block will be processed in PVO-2, but skipped in PVO-1. Referring to the first case of PVO-k embedding in Section 3.1 with $k=2$, a message bit "1" can be embedded into this block by modifying maximum-valued pixels. So, according to (15), the pixels are modified as $(p_1', ..., p_6')=$


a

154	156	153	153	152
157	157	149	148	148
154	158	157	157	159
151	158	157	157	158

b

154	156	153	153	152
158	158	149	148	148
154	158	157	157	159
151	158	157	157	158

Fig. 11. A cover (left) and its marked (right) block with its context. The block size is 2×3, and the 14 shadow pixels are the context of the block. (a) A 2×3 cover block and its context , (b) a 2×3 marked block and its context.

$(149,153,154,156,158,158)$. The marked block is shown in Fig. 11(b).

At decoder, the parameters $(t_1^*, t_2^*)=(35,60)$ are first extracted from the LSB of first 78 pixels. Then, the data extraction and recovery are processed in an inverse order as in the embedding phase, i.e., from bottom to top, and right to left. So, the context of each block can be guaranteed the same as the original one. When the marked block shown in Fig. 11(b) is processed, its context has been already recovered, and the same complexity $c=50$ is obtained. Then, the marked pixels are sorted as $(p_1',\ldots,p_6')=(149,153,154,156,158,158)$. So $a=2$, $b=1$, and the prediction-errors x' and y' are $x'=158-156=2$ and $y'=149-153=-4$, respectively. Here, the fact $c>t_1^*$ and $c<t_2^*$ indicates that the block is only processed by PVO-2. Referring to the first case of PVO-k extraction in Section 3.1 with $k=2$, the original pixels are restored as $(p_1,\ldots,p_6)=(149,153,154,156,157,157)$, and one data bit "1" is extracted from the block.

4. Experimental results

In this section, the proposed method is evaluated by comparing it with Li et al.'s PVO-based embedding [31] and two state-of-the-art methods of Sachnev et al. [21] and Hong [30].

Fig. 12 shows the comparison of capacity versus image fidelity trade-off for eight 512×512 sized standard gray-scale images including Lena, Baboon, Airplane, Barbara, Elaine, Lake, Boat and Peppers. Since our method focuses on high fidelity cases by only modifying each pixel value at most by 1, the capacity range in comparison is limited from an initial capacity of 5000 bits to its maximum with a step-size of 1000 bits. For the eight images, the maximum capacity is 37,000, 13,000, 47,000, 31,000, 23,000, 26,000, 26,000 and 31,000 bits. From Fig. 12, one can see that the proposed method outperforms these state-of-the-art works in most cases since our method provides usually a larger PSNR for a given capacity.

For Li et al.'s PVO-based embedding, our method is a direct extension of their work. According to Fig. 12, our method can provide a larger PSNR for every test image whatever the capacity is. Referring to Tables 1 and 2, our method improves Li et al.'s by 0.87 dB on average for a capacity of 10,000 bits, and 0.89 dB for 20,000 bits. Moreover, the proposed method can increase Li et al.'s maximum capacity by 12% on average for the eight test images. The superiority over Li et al.'s work is thus verified.

Sachnev et al.'s method is a hybrid RDH scheme by using rhombus prediction and sorting technique. This method performs well and has been verified better than many state-of-the-art works. Compared with this well performed method, the proposed one yields a better result in most cases, but has a comparable performance when the capacity approaches the maximum. This is due to the fact that options of combined embedding for high capacity is insufficient, and thus our gain in optimized embedding becomes smaller. However, for moderate capacity, our method is better. Referring to Tables 1 and 2, our method improves Sachnev et al.'s by 2.25 dB on average for a capacity of 10,000 bits, and 1.6 dB for 20,000 bits.

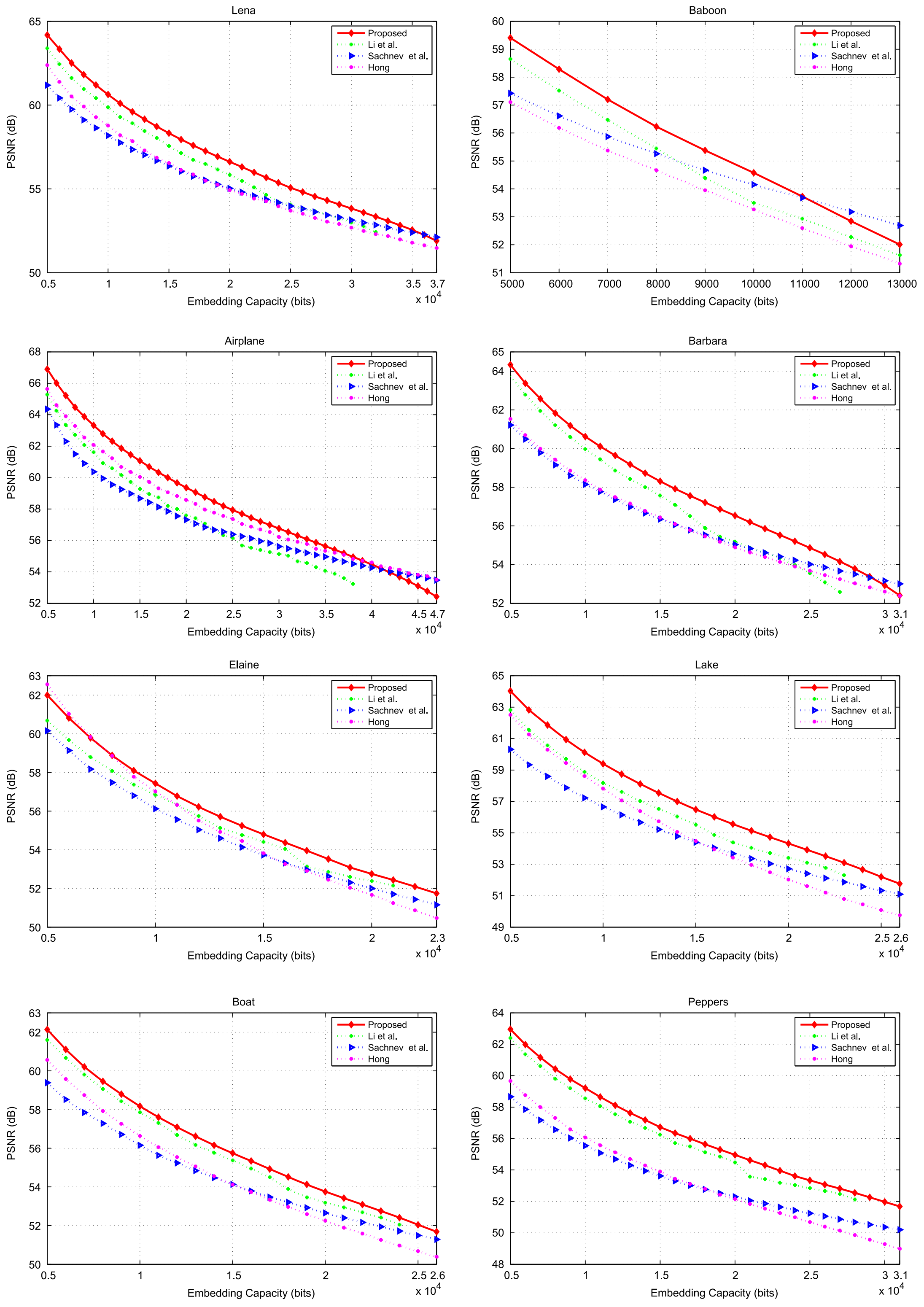


Fig. 12. Performance comparison between our method and three methods of Li et al. [31], Sachnev et al. [21] and Hong [30].

Like Sachnev et al.'s, Hong's method [30] is also based on PEE and an embedding-position-selection strategy in which MED is used for prediction and an error energy estimator for selecting profitable pixels. According to Fig. 12, Hong's method performs similar to Sachnev et al.'s and it is better than ours only for a few cases.

Referring to Tables 1 and 2, our method improves Hong's by 1.67 dB on average for a capacity of 10,000 bits, and 1.68 dB for 20,000 bits.

In PVO-k embedding, only maximum- or minimum-valued pixels in a block are modified, while others are kept unchanged. The embedding using large sized blocks gives a better performance than the one based on small sized blocks, but yields a lower capacity. For low and moderate capacities, encoders can choose large block size and only use smooth blocks for data embedding. Hence, in this case, the combined PVO-k embedding outperforms the compared methods [21,30,31]. However, for high capacity cases, one has to not only divide image into blocks with small size, but also employ rough blocks for data embedding. Therefore, as the embedding capacity goes higher, the performance is weakened. For this reason, for high capacity, the proposed method performs worse than the conventional PEE methods [21,30] in some cases (e.g., for Airplane image with capacity larger than 41,000 bits).

For a given block size, the time cost of PVO-k embedding mainly consists of the optimal-threshold-determination. Once the optimal parameters are determined, the followed data embedding is fast. In our method, considering 16 block size ($n_1, n_2 \in \{2,3,4,5\}$) candidates, the optimal combined embedding would cost longer time than the one using a specified block size. However, the time cost is still acceptable, and the average time cost for data embedding is about one second in our experiments. Here, our method is implemented by C++ and ran on a personal PC.

Obviously, rather than $k \in \{1,2\}$, the combined PVO-k embedding could include more values of k for further improvements, but the resulted computational complexity would dramatically increase. Fig. 13 gives the performance comparison between the combined PVO-k with $k \in \{1,2\}$ and $k \in \{1,2,3\}$ on Boat image. In the figure, one can see that the combined PVO-k with $k \in \{1,2,3\}$ has a slight improvement on performance compared with the one of $k \in \{1,2\}$. It should be noted that for the other seven images, the two combined embeddings yield almost the same performance, and therefore the corresponding comparisons are not plotted. The tiny gains in PSNR by combining PVO-3 for data embedding is due to that PVO-3 introduces much more alterations than PVO-1 and PVO-2 (see the average distortion for PVO-k in (7)), and meantime the number of PVO-3 blocks is few. So, the benefit in using smooth blocks of PVO-3 is limited. On the other hand, the runtime of the combined PVO-k with $k \in \{1,2,3\}$ reaches up to about 2 min since one more loop is required for the exhaustive search of (t_1^*, t_2^*, t_3^*). Such a time cost is unsatisfied for the practical application. Moreover, it can be inferred that combining more than three values of k will be more time consuming. Obviously, compared with the case of $k \in \{1,2\}$, there is not an obvious benefit by including more values of k, but a burden of high computational complexity. From this point of view, the combined PVO-1 and PVO-2 would be the

Table 1
Comparison of PSNR (in dB) between our method and three methods of Li et al. [31], Sachnev et al. [21] and Hong [30], for a capacity of 10,000 bits.

Image	Li et al.	Sachnev et al.	Hong	Proposed
Lena	59.85	58.18	58.78	**60.63**
Baboon	53.50	54.15	53.26	**54.58**
Airplane	61.61	60.37	62.07	**63.33**
Barbara	59.98	58.15	58.36	**60.62**
Elaine	56.84	56.12	57.01	**57.43**
Lake	58.18	56.65	57.82	**59.40**
Boat	57.85	56.15	56.64	**58.17**
Peppers	58.55	55.55	56.07	**59.21**
Average	58.30	56.92	57.50	**59.17**

Table 2
Comparison of PSNR (in dB) between our method and three methods of Li et al. [31], Sachnev et al. [21] and Hong [30], for a capacity of 20,000 bits. The results for Baboon are not presented here since both our method and Li et al.'s cannot provide such a payload.

Image	Li et al.	Sachnev et al.	Hong	Proposed
Lena	55.84	55.03	54.92	**56.62**
Airplane	57.59	57.32	58.58	**59.36**
Barbara	55.19	55.04	54.89	**56.54**
Elaine	52.39	52.01	51.67	**52.75**
Lake	53.41	52.72	52.26	**54.32**
Boat	53.19	52.65	52.03	**53.75**
Peppers	54.48	52.30	52.16	**54.96**
Average	54.58	53.87	53.79	**55.47**

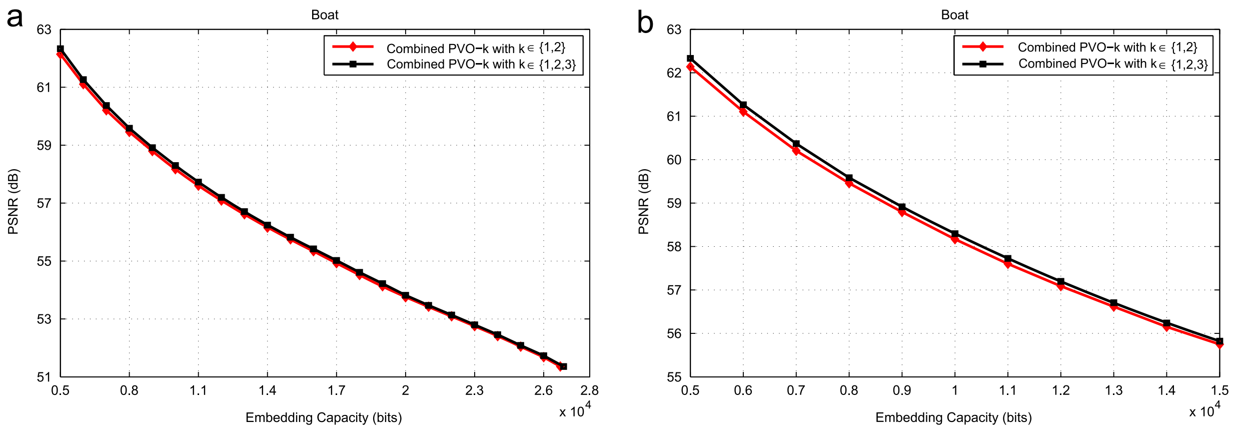


Fig. 13. Performance comparison between the combined PVO-k with $k \in \{1,2\}$ and $k \in \{1,2,3\}$ on Boat image, where the sub-figure (b) is the zoom in of (a).

good choice for the trade-off between the performance and the time cost.

5. Conclusions

In this paper, a new RDH scheme based on generalized PVO-based embedding, namely PVO-k, is proposed. PVO-k is an extension of Li et al.'s work [31], and includes it as a special case when $k=1$. In contrast to [31], the maximum-valued (minimum-valued) pixels in a block are taken as a unit for data embedding, and are modified together to keep PVO invariant. Consequently, more blocks suitable for RDH can be utilized and an increase in capacity is obtained. Moreover, considering combined embedding of PVO-1 and PVO-2, an optimal embedding strategy to discriminately embed data into different image blocks according to the embedding payoff is presented. In this way, the payload is advisably partitioned and the distortion is further reduced. Experimental results demonstrate that the proposed method yields a superior performance than the state-of-the-art works [31,21,30].

However, there is still a lot of room for improvement of the combined PVO-k embedding. One is that the block complexity is not computed by itself, and the way employed in our method to estimate the smoothness of a block is not the best. It is expected that designing a more accurate estimation for block complexity would further enhance the PVO-based embedding. Besides, although more blocks are exploited in our method compared with [31], the smoothest blocks with identical values, i.e., $p_1 = \cdots = p_n$, have not been utilized to carry data yet. We argue that exploitation of these blocks to improve performance of PVO-based embedding would be valuable. These issues would be interesting directions for future works.

References

[1] J. Fridrich, M. Goljan, R. Du, Invertible authentication, in: Security and Watermarking of Multimedia Contents III, SPIE, vol. 4314, 2001, pp. 197–208.
[2] J. Fridrich, M. Goljan, R. Du, Lossless data embedding—new paradigm in digital watermarking, EURASIP J. Appl. Signal Process. 2002 (2) (2002) 185–196.
[3] M.U. Celik, G. Sharma, A.M. Tekalp, E. Saber, Lossless generalized-LSB data embedding, IEEE Trans. Image Process. 14 (2) (2005) 253–266.
[4] M.U. Celik, G. Sharma, A.M. Tekalp, Lossless watermarking for image authentication: a new framework and an implementation, IEEE Trans. Image Process. 15 (4) (2006) 1042–1049.
[5] W. Zhang, B. Chen, N. Yu, Improving various reversible data hiding schemes via optimal codes for binary cover, IEEE Trans. Image Process. 21 (6) (2012) 2991–3003.
[6] W. Zhang, X. Hu, X. Li, N. Yu, Recursive histogram modification: establishing equivalency between reversible data hiding and lossless data compression, IEEE Trans. Image Process. 22 (7) (2013) 2775–2785.
[7] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circuits Syst. Video Technol. 13 (8) (2003) 890–896.
[8] A.M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, IEEE Trans. Image Process. 13 (8) (2004) 1147–1156.
[9] L. Kamstra, H.J.A.M. Heijmans, Reversible data embedding into images using wavelet techniques and sorting, IEEE Trans. Image Process. 14 (12) (2005) 2082–2090.
[10] W.L. Tai, C.M. Yeh, C.C. Chang, Reversible data hiding based on histogram modification of pixel differences, IEEE Trans. Circuits Syst. Video Technol. 19 (6) (2009) 906–910.
[11] Z. Ni, Y.Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circuits Syst. Video Technol. 16 (3) (2006) 354–362.
[12] S.K. Lee, Y.H. Suh, Y.S. Ho, Reversible image authentication based on watermarking, in: Proceedings of IEEE ICME, 2006, pp. 1321–1324.
[13] M. Fallahpour, M.H. Sedaaghi, High capacity lossless data hiding based on histogram modification, IEICE Electron. Express 4 (7) (2007) 205–210.
[14] X. Wang, X. Li, B. Yang, Z. Guo, A reversible watermarking scheme for high-fidelity applications, in: Proceedings of PCM, Lecture Notes in Computer Science, vol. 5879, Springer, Berlin Heidelberg, 2009, pp. 613–624.
[15] Y.-C. Li, C.-M. Yeh, C.-C. Chang, Data hiding based on the similarity between neighboring pixels with reversibility, Digit. Signal Process. 20 (4) (2010) 1116–1128.
[16] Y.-Y. Tsai, D.-S. Tsai, C.-L. Liu, Reversible data hiding scheme based on neighboring pixel differences, Digit. Signal Process. 23 (3) (2013) 919–927.
[17] X. Li, B. Li, B. Yang, T. Zeng, General framework to histogram-shifting-based reversible data hiding, IEEE Trans. Image Process. 22 (6) (2013) 2181–2191.
[18] D.M. Thodi, J.J. Rodriguez, Expansion embedding techniques for reversible watermarking, IEEE Trans. Image Process. 16 (3) (2007) 721–730.
[19] M. Fallahpour, Reversible image data hiding based on gradient adjusted prediction, IEICE Electron. Express 5 (20) (2008) 870–876.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] W. Hong, An efficient prediction-and-shifting embedding technique for high quality reversible data hiding, EURASIP J. Adv. Signal Process. (2010), http://dx.doi.org/0.1155/2010/104835.
[25] X. Gao, L. An, Y. Yuan, D. Tao, X. Li, Lossless data embedding using generalized statistical quantity histogram, IEEE Trans. Circuits Syst. Video Technol. 21 (8) (2011) 1061–1070.
[26] X. Li, B. Yang, T. Zeng, Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection, IEEE Trans. Image Process. 20 (12) (2011) 3524–3533.
[27] D. Coltuc, Improved embedding for prediction-based reversible watermarking, IEEE Trans. Inf. Forensic Secur. 6 (3) (2011) 873–882.
[28] Y.-C. Lin, Reversible data-hiding for progressive image transmission, Signal Process.: Image Commun. 26 (10) (2011) 628–645.
[29] H.-T. Wu, J. Huang, Reversible image watermarking on prediction errors by efficient histogram modification, Signal Process. 92 (12) (2012) 3000–3009.
[30] W. Hong, Adaptive reversible data hiding method based on error energy control and histogram shifting, Opt. Commun. 285 (2) (2012) 101–108.
[31] X. 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.
[32] W.-J. Yang, K.-L. Chung, H.-Y.M. Liao, W.-K. Yu, Efficient reversible data hiding algorithm based on gradient-based edge direction prediction, J. Syst. Softw. 86 (2) (2013) 567–580.
[33] 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.
[34] S. Lee, C.D. Yoo, T. Kalker, Reversible image watermarking based on integer-to-integer wavelet transform, IEEE Trans. Inf. Forensic Secur. 2 (3) (2007) 321–330.
[35] D. Coltuc, J.M. Chassery, Very fast watermarking by reversible contrast mapping, IEEE Signal Process. Lett. 14 (4) (2007) 255–258.
[36] S. Weng, Y. Zhao, J.S. Pan, R. Ni, Reversible watermarking based on invariability and adjustment on pixel pairs, IEEE Signal Process. Lett. 15 (2008) 721–724.
[37] X. Wang, X. Li, B. Yang, Z. Guo, Efficient generalized integer transform for reversible watermarking, IEEE Signal Process. Lett. 17 (6) (2010) 567–570.
[38] D. Coltuc, Low distortion transform for reversible watermarking, IEEE Trans. Image Process. 21 (1) (2012) 412–417.

[39] F. Peng, X. Li, B. Yang, Adaptive reversible data hiding scheme based on integer transform, Signal Process. 92 (1) (2012) 54–62.
[40] X. Gui, X. Li, B. Yang, A novel integer transform for efficient reversible watermarking, in: Proceedings of ICPR, 2012, pp. 947–950.
[41] J. Hwang, J. Kim, J. Choi, A reversible watermarking based on histogram shifting, in: Proceedings of IWDW, Lecture Notes in Computer Science, vol. 4283, Springer, Germany, 2006, pp. 348–361.
[42] M.J. Weinberger, G. Seroussi, G. Sapiro, The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS, IEEE Trans. Image Process. 9 (8) (2000) 1309–1324.
[43] X. Wu, N. Memon, Context-based, adaptive, lossless image coding, IEEE Trans. Commun. 45 (4) (1997) 437–444.