Signal Processing 111 (2015) 249–260

Contents lists available at ScienceDirect

Signal Processing

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

Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding

Xiaochao Qu, Hyoung Joong Kim *

Graduate School of Information Security and Management, Korea University, Anam dong 5ga, Sungbuk-gu, Seoul 136-713, South Korea

ARTICLE INFO

Article history:
Received 16 September 2014
Received in revised form
30 November 2014
Accepted 4 January 2015
Available online 10 January 2015

Keywords:
Pixel value ordering
Pixel-based pixel value ordering
Reversible data hiding
Pixel selection

ABSTRACT

A pixel-value-ordering (PVO) predictor has been used in reversible data hiding methods to produce high-fidelity marked images when the payload size is moderate. The original PVO generates predictions in a block-by-block manner where only two pixels can be predicted in each block. In this paper, a novel pixel-based PVO (PPVO) is proposed with which each pixel is predicted using its sorted context pixels. The benefit of the PPVO is that the pixels in smooth image regions can be better used to embed data, while most of those pixels are neglected in PVO due to the block constraint. Experimental results demonstrate that the embedding capacity of PPVO is larger than that of PVO, especially for relatively smooth images. The obtained marked image quality of PPVO is also better than that of PVO and other state-of-the-art reversible data hiding methods.

1. Introduction

Reversible data hiding (RDH) embeds secret data into a host image and can recover the original host image perfectly after the secret data have been extracted from the marked image. The reversibility of RDH is highly desired in some special applications such as military and medical image processing, where no distortion is acceptable. Many RDH methods have been proposed in recent years. Among them, difference expansion (DE) [1] and histogram shifting (HS) [2] are the two pivotal approaches.

DE expands the difference between one pair of pixels to embed one bit of data. A location map is needed for distinguishing which pixel pair has been used for expansion. Thus, the true embedding capacity is reduced significantly due to the large size of the location map. Subsequently, DE was generalized by Alattar [3], where $n-1$ bits can be embedded into a vector with n pixels. Kim et al. [4] proposed a novel difference expansion transform with an efficient location map. The location map in their method is smaller than Tian's original method [1]. Thodi and Rodríguez [5] proposed an important extension of DE which is called prediction error expansion (PEE). PEE uses each pixel's prediction error instead of the pixel difference to embed data, so that the embedding capacity is doubled compared with DE. Another advantage of PEE is that complex predictors can generate much smaller prediction errors than the simple pixel difference used in DE, which reduces the embedding distortion. In fact, pixel difference can be regarded as the simplest prediction method. PEE was then improved in many different ways such as using a payload-dependent overflow location map [6], sorting and prediction [7], adaptive embedding [8,9], context embedding [10–12] and PEE for color images [13].

Another popular method, HS, modifies the image histogram to embed data. A histogram is first constructed using all the pixels' gray-scale values. Then, a pair of a peak bin (which has the largest number of pixels) and a zero bin (which does not have pixels) in the histogram is identified. The pixels between the peak bin and zero bin are shifted by one towards the zero bin to generate an empty bin. The pixels in the peak bin are then shifted by one or unchanged to embed one bit. HS can produce high-quality marked images because the maximum pixel value changed is one.

* Corresponding author. Tel.: +82 10 6251 6343.
E-mail addresses: quxiaochao@gmail.com (X. Qu), khj-@korea.ac.kr (H.J. Kim).

http://dx.doi.org/10.1016/j.sigpro.2015.01.002
0165-1684/

However, the embedding capacity of HS is rather limited. Several techniques have been proposed to enhance the performance of the HS based method, where most of them try to construct a sharp histogram. Prediction error [14], pixel difference [15,16], sub-sampled image difference [17], interpolation error [18] and directed-prediction [19] are possible ways to generate histograms with Laplacian-like distributions . Other improvements include adaptive embedding [20], generalized histogram shifting [21], context modeling [22] and two dimensional histogram shifting (TDHS) [23,24].

Recently, a high-fidelity reversible data hiding method was proposed by Li et al. [25]. A new prediction method called pixel value ordering (PVO) was combined with PEE, and they achieved impressive performance for moderate payload size. PVO generates predictions in a block-by-block manner. For each image block, the pixels are reordered into a pixel vector according to their gray-scale values, then the smallest pixel is predicted by the second smallest pixel, and the largest pixel is predicted by the second largest pixel. Clearly, only two pixels of the pixel vector can get prediction values. The prediction error for the smallest pixel is always non-positive and the prediction error for the largest pixel is always non-negative. The proportion of shifted pixels (POSP) is used to evaluate PVO's performance, where smaller POSP indicates that more data can be embedded with less distortion. PVO can generate lower POSP compared with [6] and [7] as shown in [25].

PVO uses prediction errors 1 and -1 to embed data, whereas prediction error 0 is untouched. Peng et al. [26] showed that 0 is usually the peak of the histogram and should be used to embed data. They proposed an improved PVO (IPVO) method in which the prediction error 0 can be used for embedding. The calculation of prediction errors in IPVO considers the relative location relationships of the pixels in the original block and both positive and negative prediction errors can be obtained for the smallest pixel and the largest pixel. IPVO is shown to have larger embedding capacity than PVO, which allows IPVO to use larger blocks compared with PVO. The large blocks usually have small POSP which helps IPVO outperforms PVO as shown in [26]. Another improvement of PVO is proposed in [27] named with PVO-K. PVO-K takes all maximum-valued (or minimum-valued) pixels as a unit to embed data. It is shown that PVO is a special case of PVO-K when K is 1, that is, PVO-1. By combining PVO-1 and PVO-2, PVO-K in [27] achieves better embedding performance than PVO.

Both PVO, IPVO and PVO-K partition the host image into blocks and make predictions in a block-by-block manner. The maximum embedding capacity is therefore constrained to two bits for each block. As a result, the pixels in the smooth image region can not be fully used. In this paper, a novel pixel-based PVO (PPVO) is proposed. Unlike PVO, IPVO and PVO-K, PPVO generates predictions in a pixel-by-pixel manner. The unused pixels in PVO, IPVO and PVO-K in the smooth image regions have a chance to embed data in PPVO. Thus, the embedding capacity of PPVO is increased significantly. To further improve the performance of PPVO, a better pixel selection method that uses more context pixels is incorporated into the proposed reversible data hiding method. In the experiment, PPVO is shown to have better marked image quality and higher embedding capacity than PVO, IPVO and PVO-K.

The outline of this paper is as follows. PVO, IPVO, PVO-K and Algorithm-II in [21] (which has some similarity with PPVO) are introduced in Section 2. In Section 3, the proposed PPVO is introduced. Section 4 presents extensive experiments. Section 5 provides our conclusion.

2. Related works

In this section, PVO, IPVO, PVO-K and Algorithm-II in [21] are briefly introduced.

2.1. PVO

Assume that a pixel block B has $w \times h$ pixels, where w and h are the width and the height of B, respectively. A pixel vector X is generated by scanning B in the raster-scan order and can be written as $(x_1, \ldots, x_{w\times h})$. X is then sorted according to the pixel values in the ascending order and $(x_{\pi(1)}, \ldots, x_{\pi(w\times h)})$ denoted as X_π is obtained. π maps the original X into an ordered X_π such that $x_{\pi(1)} \le x_{\pi(2)} \le \ldots \le x_{\pi(w\times h-1)} \le x_{\pi(w\times h)}, \pi(i) < \pi(j)$ if $x_{\pi(i)} = x_{\pi(j)}$ and $i < j$. The second largest pixel value $x_{\pi(w\times h-1)}$ is used to predict the largest value $x_{\pi(w\times h)}$. Similarly, the second smallest pixel value $x_{\pi(2)}$ is used to predict the smallest pixel value $x_{\pi(1)}$. Two prediction errors can be obtained by

$$\mathrm{PE}_{\max} = x_{\pi(w\times h)} - x_{\pi(w\times h-1)}. \tag{1}$$

$$\mathrm{PE}_{\min} = x_{\pi(1)} - x_{\pi(2)}. \tag{2}$$

Secret data can be embedded according to

$$\tilde{\mathrm{PE}}_{\max} = \begin{cases} \mathrm{PE}_{\max} & \text{if } \mathrm{PE}_{\max} = 0, \\ \mathrm{PE}_{\max} + b & \text{if } \mathrm{PE}_{\max} = 1, \\ \mathrm{PE}_{\max} + 1 & \text{if } \mathrm{PE}_{\max} > 1, \end{cases} \tag{3}$$

$$\tilde{\mathrm{PE}}_{\min} = \begin{cases} \mathrm{PE}_{\min} & \text{if } \mathrm{PE}_{\min} = 0, \\ \mathrm{PE}_{\min} - b & \text{if } \mathrm{PE}_{\min} = -1, \\ \mathrm{PE}_{\min} - 1 & \text{if } \mathrm{PE}_{\min} < -1, \end{cases} \tag{4}$$

where $b \in \{0, 1\}$ is a message bit to be embedded. Then the modified pixel value can be obtained as

$$\tilde{x}_{\pi(w\times h)} = x_{\pi(w\times h-1)} + \tilde{\mathrm{PE}}_{\max}. \tag{5}$$

$$\tilde{x}_{\pi(1)} = x_{\pi(2)} + \tilde{\mathrm{PE}}_{\min}. \tag{6}$$

The data embedding procedure always increases the value of $x_{\pi(w\times h)}$ and decreases the value of $x_{\pi(1)}$ such that the order of the pixels in X_π is maintained. In the data extracting procedure, the pixel vector with the same order can be obtained and data can be extracted from $\tilde{x}_{\pi(w\times h)}$ and $\tilde{x}_{\pi(1)}$.

POSP is used as a metric to evaluate the predictor's performance in PEE-based reversible data hiding in [25]. Assume that N_s is the number of shifted pixels and N_c is the number of pixels where a bit can be embedded. POSP is defined as

$$\mathrm{POSP} = \frac{N_s}{N_s + N_c}. \tag{7}$$

A small POSP implies that more pixels are used for embedding data and fewer pixels are shifted. As shown in

[25], PVO has smaller POSP than the prediction methods in [6,7,18].

PVO tends to have big POSP in rough blocks and small POSP in smooth blocks. In [25], whether to embed data into a certain block is determined by the difference value $x_{\pi(w\times h-1)}-x_{\pi(2)}$, which can be regarded as the noise level of the block. The blocks with noise levels that are smaller than a threshold T will be utilized for embedding, and the other blocks stay unmodified. The threshold T is determined by the payload size. The smallest noise level that can offer enough embedding capacity for the payload is selected as the T.

2.2. IPVO

IPVO first partitions cover image into blocks. For each block, the ordered pixel vector $(x_{\pi(1)},\ldots,x_{\pi(w\times h)})$ is generated by IPVO using the same way as PVO. However, the position order of $\pi(1)$ and $\pi(2)$ and that of $\pi(w\times h)$ and $\pi(w\times h-1)$ in the original block are considered when calculating prediction errors. In detail, the prediction errors for the largest pixel $X_{\pi(w\times h)}$ is calculated as

$$\mathrm{PE}_{\max}=x_u-x_v, \tag{8}$$

where

$$\begin{cases} u=\min(\pi(w\times h), \quad \pi(w\times h-1)),\\ v=\max(\pi(w\times h), \quad \pi(w\times h-1)), \end{cases} \tag{9}$$

where min() and max() return the minimum and the maximum value of a pixel vector, respectively. The histogram of the new $\mathrm{PE}_{\max}$ has a Laplacian-like shape with the peak point 0. Unlike PVO, IPVO uses 0 and 1 of $\mathrm{PE}_{\max}$ to embed data as

$$\tilde{\mathrm{PE}}_{\max}=\begin{cases} \mathrm{PE}_{\max}+b & \text{if } \mathrm{PE}_{\max}=1,\\ \mathrm{PE}_{\max}+1 & \text{if } \mathrm{PE}_{\max}>1,\\ \mathrm{PE}_{\max}-b & \text{if } \mathrm{PE}_{\max}=0,\\ \mathrm{PE}_{\max}-1 & \text{if } \mathrm{PE}_{\max}<0, \end{cases} \tag{10}$$

The marked value of $x_{\pi(w\times h)}$ is calculated as

$$\tilde{x}_{\pi(w\times h)}=\begin{cases} x_{\pi(w\times h)}+b & \text{if } \mathrm{PE}_{\max}=1,\\ x_{\pi(w\times h)}+1 & \text{if } \mathrm{PE}_{\max}>1,\\ x_{\pi(w\times h)}+b & \text{if } \mathrm{PE}_{\max}=0,\\ x_{\pi(w\times h)}+1 & \text{if } \mathrm{PE}_{\max}<0. \end{cases} \tag{11}$$

Note that the largest pixel $x_{\pi(w\times h)}$ is never decreased, therefore, the marked value $\tilde{x}_{\pi(w\times h)}$ remains as the largest value in the marked pixel vector. The data embedding for the smallest pixel is similar with the largest pixel as shown in [26].

2.3. PVO-K

In PVO-K, the maximum-valued (or minimum-valued) pixels are treated as a unit for data embedding. The blocks that produce prediction error of 0 is ignored in PVO, however, those blocks can be used in PVO-K. It is shown that PVO-K has larger embedding capacity and better marked image quality than PVO [27]. The embedding process for the maximum-valued pixels is described in the following.

For a pixel block, PVO-K sorts the pixel vector in the same way as PVO to obtain X_π as $(x_{\pi(1)},\ldots,x_{\pi(w\times h)})$. Assume that the magnitude relationship for the last $K+1$ pixels holds as $x_{\pi(w\times h-K)}<x_{\pi(w\times h-K+1)}=\ldots=x_{\pi(w\times h)}$. The last K pixels have the same value (denoted as p) and is used to predict the last $(K+1)$th pixel. The prediction errors is obtained by $\mathrm{PE}=x_{\pi(w\times h-K+1)}-p$. Based on the value of PE and the to-be-embedded bit b, the values of the last K pixels are modified as

$$\tilde{x}_{\pi(i)}=\begin{cases} x_{\pi(i)}+b & \text{if } \mathrm{PE}=-1,\\ x_{\pi(i)}+1 & \text{if } \mathrm{PE}<-1, \end{cases} \tag{12}$$

where $w\times h-K+1\le i\le w\times h$. Notice that the last K pixels are modified in the same way such that the magnitude relationship of the last K pixels is maintained. The decoder can thus recognize the value of K that is used in the encoder side. It can be noticed that PVO is a special case of PVO-K with the value of K set to be 1. Different values of K can be used for different blocks independently and produce different performances. An optimal combined embedding is thus proposed in [27] to search the best way to combine different PVO-K (e.g, PVO-1 and PVO-2). The obtained optimal combined embedding outperforms PVO with large margin for most cases.

2.4. Algorithm-II in [21]

A general framework to histogram-shifting-based reversible data hiding is proposed in [21]. In this general framework, a reversible data hiding algorithm can be obtained by designing the so-called shifting and embedding functions. Two efficient reversible data hiding methods are proposed based on the general framework. The Algorithm-II in [21] utilizes the ordered two neighboring pixels to predict the current pixel which has some similarity to the proposed PPVO method. In the following, we briefly introduce this method and will show the difference between this method and the proposed PPVO in later sections.

Assume that the current pixel is x and its right and down neighboring pixels are x_1 and x_2, respectively. The prediction of x is obtained by

$$\hat{x}=\begin{cases} p & \text{if } x\ge p,\\ q-1 & \text{if } x<q, \end{cases} \tag{13}$$

where $p=\max(x_1,x_2)$ and $q=\min(x_1,x_2)$ and $p\ge q$ holds. The operation of min() and max() returns the smallest and the largest value of the input vector, respectively. It can be noticed that only the pixels satisfying $x\ge p$ or $x<q$ are predicted. All the other pixels are skipped without embedding data. To embed a bit b, the current pixel is modified as

$$\tilde{x}=\begin{cases} x+b & \text{if } x=p,\\ x+1 & \text{if } x>p,\\ x-b & \text{if } x=q-1,\\ x-1 & \text{if } x<q-1,\\ x & \text{if } q\le x<p, \end{cases} \tag{14}$$

The embedding capacity of this algorithm is low since the maximum modification of each pixel is constrained to be 1. However, the quality of the marked image is shown to be better than the method of [6,18,8] with small payload size.

3. Proposed method

In this section, PPVO predictor is first introduced. The detailed embedding procedure and extracting procedure of PPVO are then presented.

3.1. Pixel based pixel value ordering (PPVO)

PVO, IPVO and PVO-K utilize a sorted pixel vector inside a block to perform predictions. The obtained prediction is accurate, however, the block constraint makes it difficult to efficiently embed data into smooth regions in the cover image. More specifically, the block constraint allows embedding two bits into one block at most. Therefore, many pixels are ignored even if they are highly predictable in smooth blocks. If the sorted pixel vector can be used for prediction and there is no block constraint at the same time, a better embedding performance can be achieved. In the following, we will show how to use the sorted pixel vector to make predictions in a pixel-by-pixel manner.

For each pixel, the context pixels of that pixel can be defined using the neighboring pixels as in Fig. 1. The number of context pixels is denoted as CN. Assume that CN context pixels form a context pixel vector $C=(c_1,\ldots,c_{CN})$. The method for determining the proper value of CN is described later. The current pixel x can be predicted according to Algorithm 1.

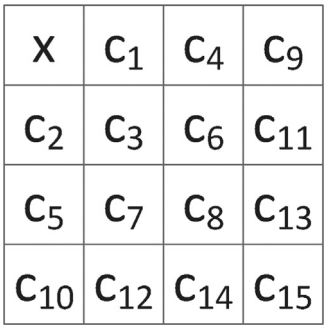

Fig. 1. The context pixels of pixel x (with CN set to 15) in PPVO, which are from c_1 to c_{15}.

Algorithm 1. Prediction.

Input: The current pixel: x, context pixel vector C
Onput: Predicted value: $\hat{x}$
if $\max(C)\neq\min(C)$ **then**
  if $x\leq\min(C)$ **then**
    $\hat{x}=\min(C)$; /*Case1 */
  else if $x\geq\max(C)$ **then**
    $\hat{x}=\max(C)$; /*Case2 */
  else
    Skip; /*Case3 */
else
  if VC = 254 **then** /* The values of pixels in C are denoted as VC since all pixels have the same values. */
    if $x=$VC **then**
      $\hat{x}=$VC; /*Case4 */
    else
      Skip; /*Case5 */
  else
    if $x\leq$VC **then**
      $\hat{x}=$VC; /*Case6 */
    else
      Skip; /*Case7 */
return $\hat{x}$

Notice that the prediction procedure is divided into two different cases based on the values of $\max(C)$ and $\min(C)$. When $\max(C)$ and $\min(C)$ have different values, x can get prediction if $x\leq\min(C)$ or $x\geq\max(C)$. Otherwise, x cannot get prediction and is skipped. When $\max(C)$ equals $\min(C)$, the prediction procedure is further divided into two different cases. If VC is 254, only those pixels that are equal to VC can be predicted. Otherwise, only those pixels that are equal to or smaller than VC can be predicted.

As shown in the previous section, the algorithm-II in [21] also uses min() and max() of neighboring pixels to make predictions. However, there are three differences compared with PPVO. First, only two neighbor pixels are involved in prediction in algorithm-II in [21] and PPVO can use more neighboring pixels. Second, the prediction uses $\min()-1$ instead of min() in algorithm-II [21]. Third, the prediction in algorithm-II [21] does not consider the special situation of those pixels with the value 254. So, the predictions obtained in algorithm-II [21] are different compared with PPVO.

Examples of seven different cases of predictions in PPVO are shown in Fig. 2.

For the pixel that obtains its prediction value, a bit b is tried to be embedded according to Algorithm 2. It is clearly seen that when $\max(C)\neq\min(C)$, x remains the same value or is decreased by 1 if x is predicted by $\min(C)$, and, x

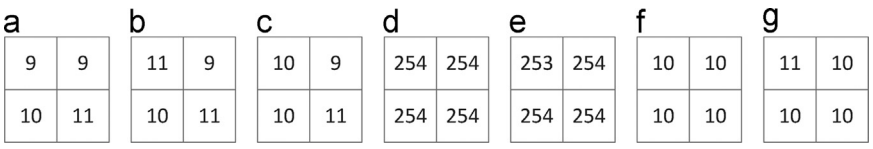

Fig. 2. The top-left pixel is the current pixel x that will be predicted and the number of context pixels used is 3. (a)–(g) Correspond to the seven situations in the prediction algorithm. (a) x is predicted by min(9,10,11). (b) x is predicted by max(9,10,11). (c) x is skipped. (d) x is predicted by 254. (e) x is skipped. (f) x is predicted by 10. (g) x is skipped.

remains the same value or is increased by 1 if x is predicted by max(C). In this way, the order of x and $\tilde{x}$ compared with the context pixels is unchanged, such that the prediction procedure can obtain the same predictions in the extraction side. When $\max(C)=\min(C)$, the way that modifies the value of x is determined by the value of VC. Now, we explain why the prediction and embedding are determined by different values of VC. As we know, the modification of pixel values in the embedding procedure may cause overflow or underflow problem which creates pixels that are bigger than 255 or smaller than 0. To prevent this problem, we modify the pixels with value of 255 to 254 and 0 to 1 as a preprocessing of the cover image, and use a location map to record these modifications. Since the embedding procedure of PPVO modifies pixel values by 1 at most, there will be no overflow and underflow problem now. However, with 255 modified to 254, PPVO may further modify 254 to 253, which causes large distortion because of the value of change for some pixels is 2. When $\max(C)\neq\min(C)$, the maximum value of min(C) is 253, therefore, the value of 254 cannot be decreased. When $\max(C)=\min(C)$, if VC has the value of 254, then the pixel with value 254 will stay same value or be increased. If VC <254, the pixels with value of 254 will be skipped without predicting. In summary, the pixels with value 254 can only be modified to 255 or stay at 254 in all the possible situations. The possible modifications of all pixels in the cover image are thus guaranteed to be equal to or lesser than 1.

Algorithm 2. Embedding.

Input: The current pixel: x, the predicted value: $\hat{x}$, the context pixel vector: C, the to-be-embedded bit: b
Output: The marked pixel: $\tilde{x}$
if $\max(C)\neq\min(C)$ **then**
  if $x\leq\min(C)$ **then**
    if $x=\hat{x}$ **then**
      $\tilde{x}=x-b$;
    else if $x<\hat{x}$ **then**
      $\tilde{x}=x-1$;
  else if $x\geq\max(C)$ **then**
    if $x=\hat{x}$ **then**
      $\tilde{x}=x+b$;
    else if $x>\hat{x}$ **then**
      $\tilde{x}=x+1$;
else
  if VC $=254$ **then**
    $\tilde{x}=x+b$;
  else
    if $x=\hat{x}$ **then**
      $\tilde{x}=x-b$;
    else if $x<\hat{x}$ **then**
      $\tilde{x}=x-1$;
return $\tilde{x}$

To extract the embedded data from $\tilde{x}$ and recover the original pixel x. First, the same prediction $\hat{x}$ is obtained by using Algorithm 1. For the pixels that can be predicted, extract the embedded bit b and recover the original x by using Algorithm 3.

Algorithm 3. Extraction.

Input: The marked pixel: $\tilde{x}$, the predicted value: $\hat{x}$, the context pixel vector: C
Output: The recovered pixel value: x, the extracted bit: b
if $\max(C)\neq\min(C)$ **then**
  if $\tilde{x}\leq\min(C)$ **then**
    if $\tilde{x}=\hat{x}$ **then**
      $x=\tilde{x};\ b=0$;
    else if $\tilde{x}=\hat{x}-1$ **then**
      $x=\tilde{x}+1;\ b=1$;
    else
      $x=\tilde{x}+1$; /* No bit is extracted */
  else if $\tilde{x}\geq\max(C)$ **then**
    if $\tilde{x}=\hat{x}$ **then**
      $x=\tilde{x};\ b=0$;
    else if $\tilde{x}=\hat{x}+1$ **then**
      $x=\tilde{x}-1;\ b=1$;
    else
      $x=\tilde{x}-1$; /* No bit is extracted */
else
  if VC $=254$ **then**
    if $\tilde{x}=\hat{x}$ **then**
      $x=\tilde{x};\ b=0$;
    else if $\tilde{x}=\hat{x}+1$ **then**
      $x=\tilde{x}-1;\ b=1$;
  else
    if $\tilde{x}=\hat{x}$ **then**
      $x=\tilde{x};\ b=0$;
    else if $\tilde{x}=\hat{x}-1$ **then**
      $x=\tilde{x}+1;\ b=1$;
    else
      $x=\tilde{x}+1$; /* No bit is extracted */
return x, b

The benefit of PPVO is shown in Fig. 3. Let us suppose that both PVO and PPVO try to embed data into the four red pixels in a very smooth image region. PVO obtains two prediction errors 0 out of the four red pixels, where two pixels are neglected even if they are quite easy to be predicted. Worse yet, PVO only embeds data into prediction errors 1 and -1, therefore, no data can be embedded into the four red pixels. PPVO predicts each red pixel using its three context pixels. Because the red pixel and its context pixels have the same value (same situation as in Fig. 2(f)), a prediction error 0 is obtained and one bit can be embedded. Given the four red pixels, four bits can be

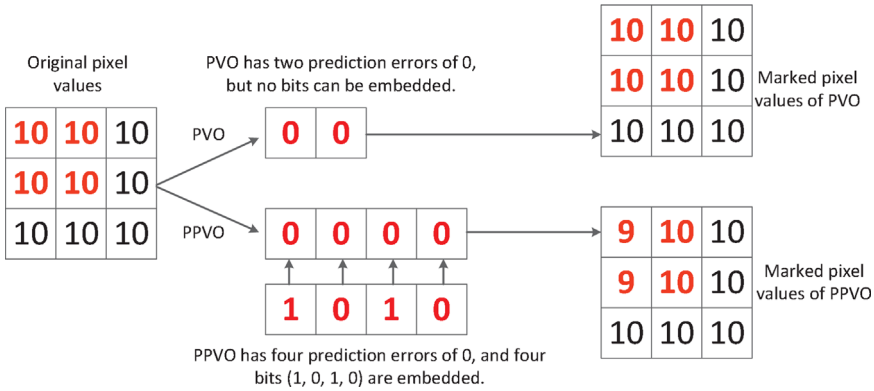

Fig. 3. Both PVO and PPVO try to embed data into the four red pixels. The block size of PVO is 2×2. The context pixel number of PPVO is 3. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

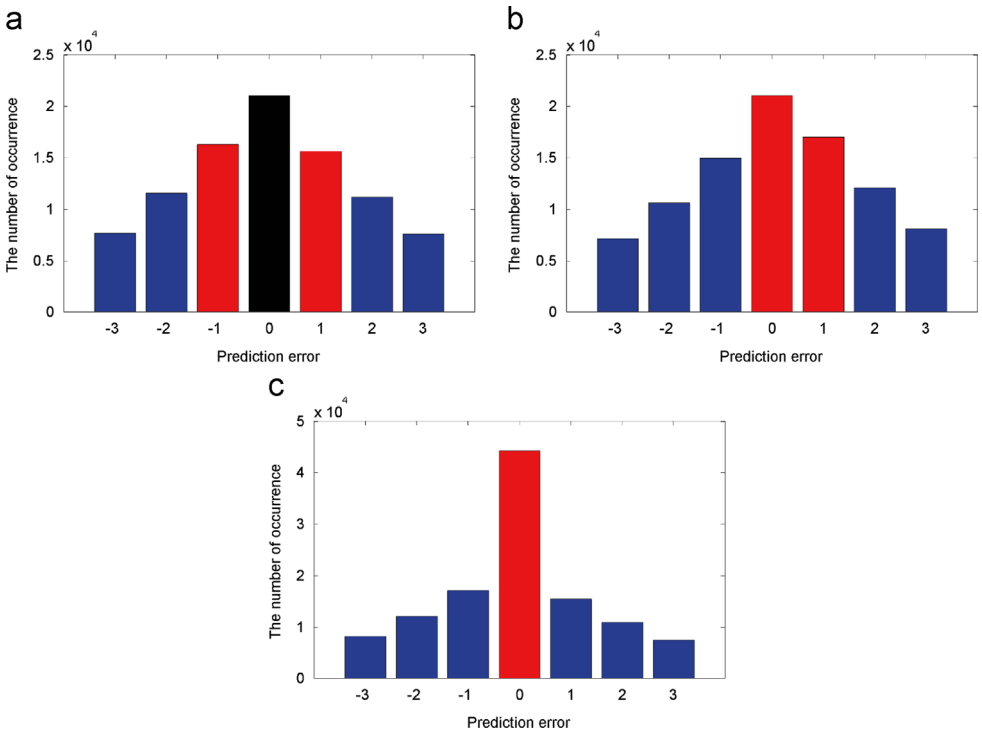

Fig. 4. The prediction error histogram of PVO, IPVO and PPVO for Lena. Black bin is the unused bin. Red bins are used to carry data. Blue bins are shifted. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

embedded. The prediction error distributions of PVO, IPVO and PPVO for Lena and the F16 image are shown in Figs. 4 and 5, respectively. It can be seen that PPVO has a much higher 0 bin than PVO and IPVO. When the image is very smooth, e.g., F16, the embedding capacity of PPVO is much higher than that of PVO and IPVO.

3.2. Pixel selection using a large context

Pixel selection aims to distinguish the smooth and the rough image regions. The smooth image region is utilized with priority. Because prediction errors in the smooth region are usually smaller than those in the rough region, the data embedding procedure can embed more data with less distortion in a smooth region.

In PVO [8] and IPVO [26], noise level (NL) is used to select smooth blocks to embed data. However, the NL calculation is constrained by block size. For a large block, the estimated NL accurately represents the true noise level. The estimated NL is not accurate due to insufficient data when w and h are small. Let us assume that w and h of a block are both 2. There are in total four pixels in this block, and therefore, the estimated NL is the difference between the third pixel and the second pixel, where only two pixels are involved to estimate NL. The estimated NL cannot reflect the true smoothness of the block which will lead rough blocks to be used instead of the smooth blocks.

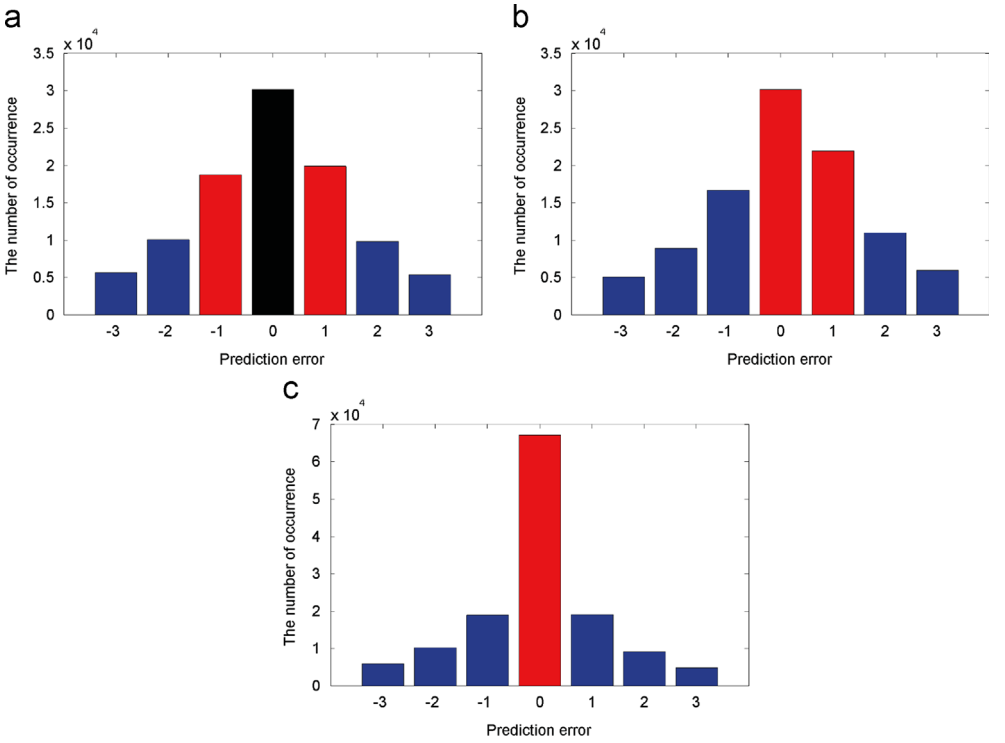

Fig. 5. The prediction error histogram of PVO, IPVO and PPVO for F16. Black bin is the unused bin. Red bins are used to carry data. Blue bins are shifted. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

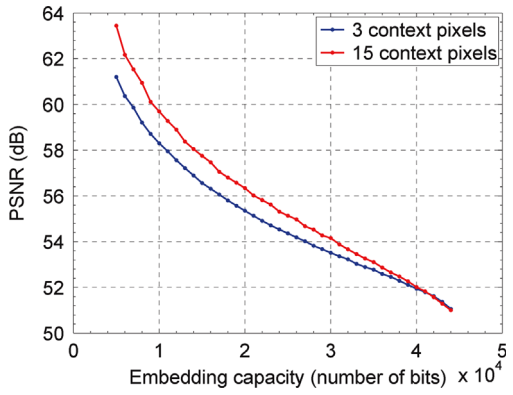

Fig. 6. The performance comparison of PPVO with NL using 3 and 15 context pixels.

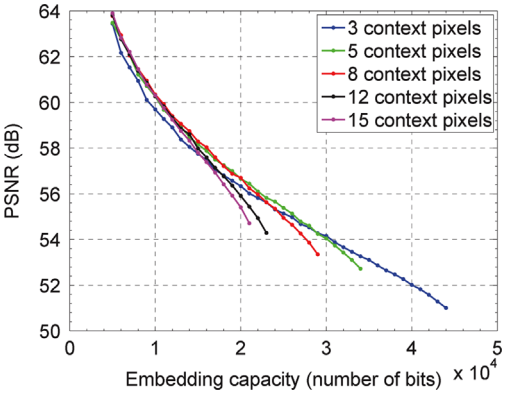

Fig. 7. The performance comparison of different numbers of context pixels used in PPVO for Lena.

It is better to use more pixels into the estimation of the NL. Hence, 15 context pixels, as shown in Fig. 1, are always used in the estimation of NL of the pixel x in our method. The NL is calculated as the difference of the maximum and minimum value of these 15 context pixels. To show the big performance difference when using NL with different number of context pixels, we embed the same data into Lena. Two different NLs with 3 and 15 context pixels are tested, denoted as NL^3 and NL^{15}, respectively. The performance is shown in Fig. 6. When using NL^3, the performance degradation is large for a small payload compared with NL^{15} and the PSNR value converges to that of NL^{15} with the increased payload size. This is because that pixel selection cannot actually select the proper pixels to embed data with a large payload where all pixels should be used.

3.3. Embedding procedure

Before describing the detailed embedding procedure, the overflow and underflow problem is first addressed. The overflow and underflow problem happens when the value of the marked pixel is out of the intensity range of the host image, e.g., [0, 255] for an eight bit gray-scale image. In the proposed reversible data hiding method, the maximum modification of the host image pixel is 1. Thus, if 255 and 0

are modified to 254 and 1 before embedding, the overflow and underflow problem will not occur. A location map (LM) is needed to record those modifications to maintain the reversibility. A bit 1 in LM represents the modification of 255 to 254 or 0 to 1; a bit 0 represents the 254 or 1 is the original pixel value. The other pixels do not need the LM. LM is then losslessly compressed using arithmetic coding into a shorter bit stream denoted as CLM. The length of CLM is denoted as l_{CLM}. The CLM is concatenated with the to-be-embedded data and embedded into the host image.

The detailed embedding procedure is as follows:

1. Assume that the host image is an eight-bit gray-scale image with size $W \times H$. Generate LM for the host image and compress LM to obtain CLM.
2. Some overhead information needs to be transmitted to the extracting side for blind extraction:
   - The length of compressed location map l_{CLM} ($\log_2(W \times H)$ bits).
   - The NL threshold T (8 bits).
   - Context pixel number CN (8 bits).
   - The location of the last embedded pixel L ($\log_2(W \times H)$ bits).

   The first $2 \times \log_2(W \times H)+16$ pixels of the last row of host image are used to transmit the overhead information, where the least significant bits (LSB) of those pixels are replaced with the overhead information and the original LSB are embedded into the host image together with the payload data. Notice that some overhead information (e.g., L) can only be obtained after all data has been embedded.
3. Scan the host image in raster-scan order. For each pixel, calculate the NL. If $NL \leq T$, try to embed data into this pixel. Otherwise, skip this pixel. This step will stop if all payload bits, the CLM and the $2 \times \log_2(W \times H)+16$ LSB bits are embedded.

The NL threshold T is determined iteratively to be the smallest value such that all data can be successfully embedded. The optimal number of context pixels CN is also iteratively determined with values from 1 to a predefined large value (15 is used in all our experiments). The number of context pixels that offer sufficient embedding capacity and give the highest quality marked image is selected. Fig. 7 shows that PPVO with different numbers of context pixels has

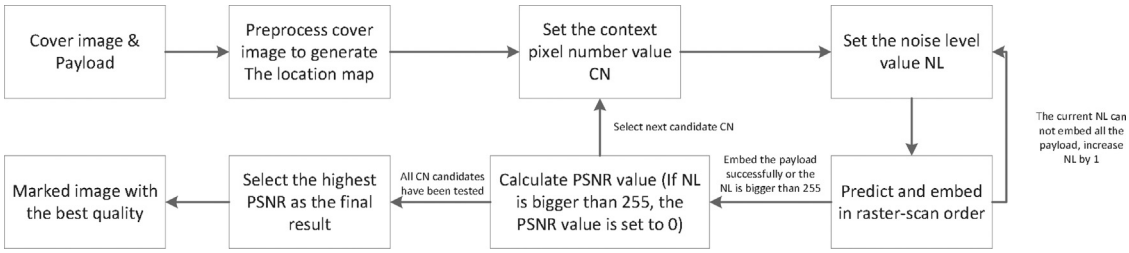

Fig. 8. The flowchart of embedding procedure.

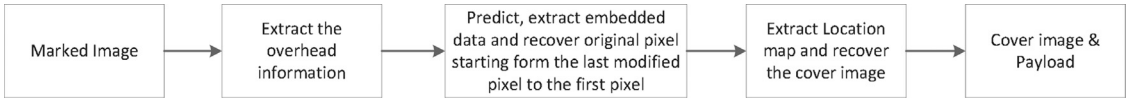

Fig. 9. The flowchart of extraction procedure.

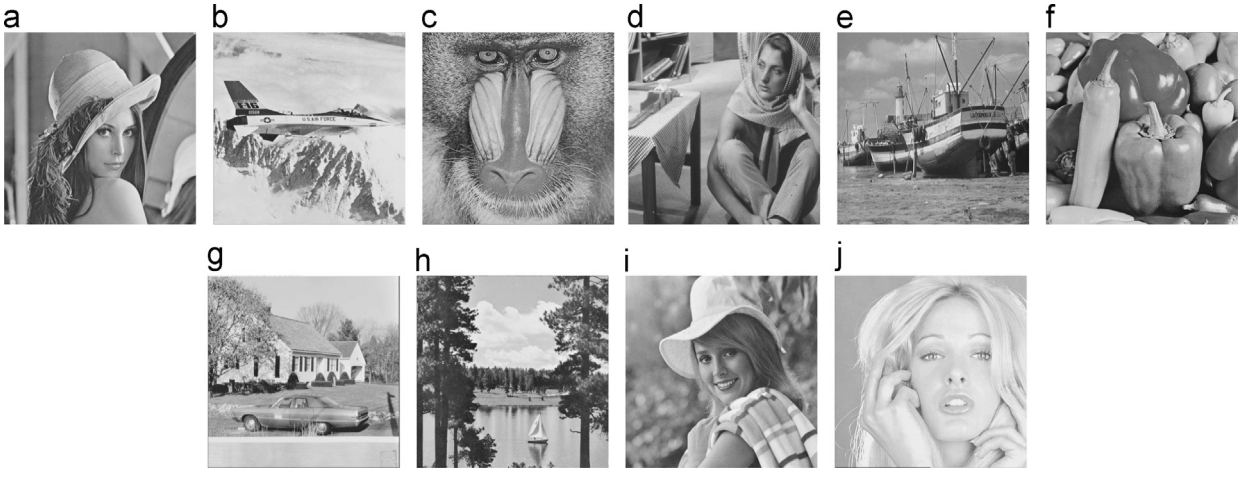

Fig. 10. The test images of SIPI image data set. (a) Lena. (b) F16. (c) Baboon. (d) Barbara. (e) Boat. (f) Peppers. (g) House. (h) Sailboat. (i) Elaine. (j) Tiffany.

different embedding capacities and marked image qualities. A flowchart of the embedding procedure is shown in Fig. 8.

3.4. Extraction procedure

The data extraction procedure is described as follows:

1. Extract the overhead information from the first $2 \times \log_2(W \times H)+16$ pixels' LSB. Determine the value of l_{CLM}, T, CN and L.
2. From the last embedded pixel, extract data and recover the host image in reverse order to the embedding procedure. For each pixel, calculate the NL. If NL $\leq T$, try to extract data and recover the original pixel value. Otherwise, skip this pixel. The extracting procedure is finished when the first pixel has been processed.
3. Extract the CLM from the extracted data and decompress CLM to obtain LM. If the pixel value is 1 or 254 and LM for that pixel is 1, modify the pixel to 0 or 255, respectively. For other cases, the pixels stay unmodified.
4. Recover the LSB of the first $2 \times \log_2(W \times H)+16$ pixels of the last row.

A flowchart of the embedding procedure is shown in Fig. 9.

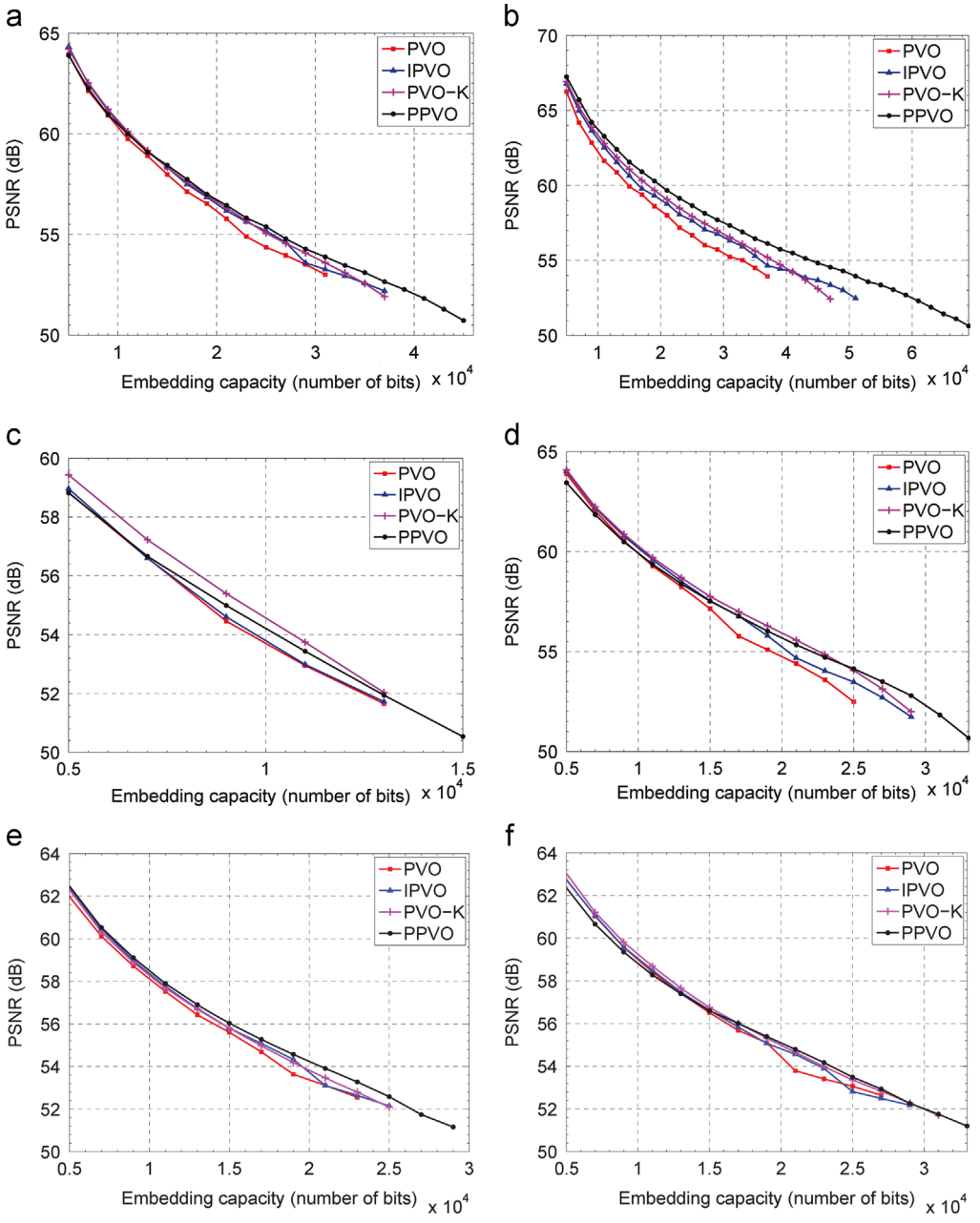

Fig. 11. The performance comparison of PVO, IPVO, PVO-K and PPVO for the test images of SIPI image data set. (a) Lena. (b) F16. (c) Baboon. (d) Barbara. (e) Boat. (f) Peppers. (g) House. (h) Sailboat. (i) Elaine. (j) Tiffany.

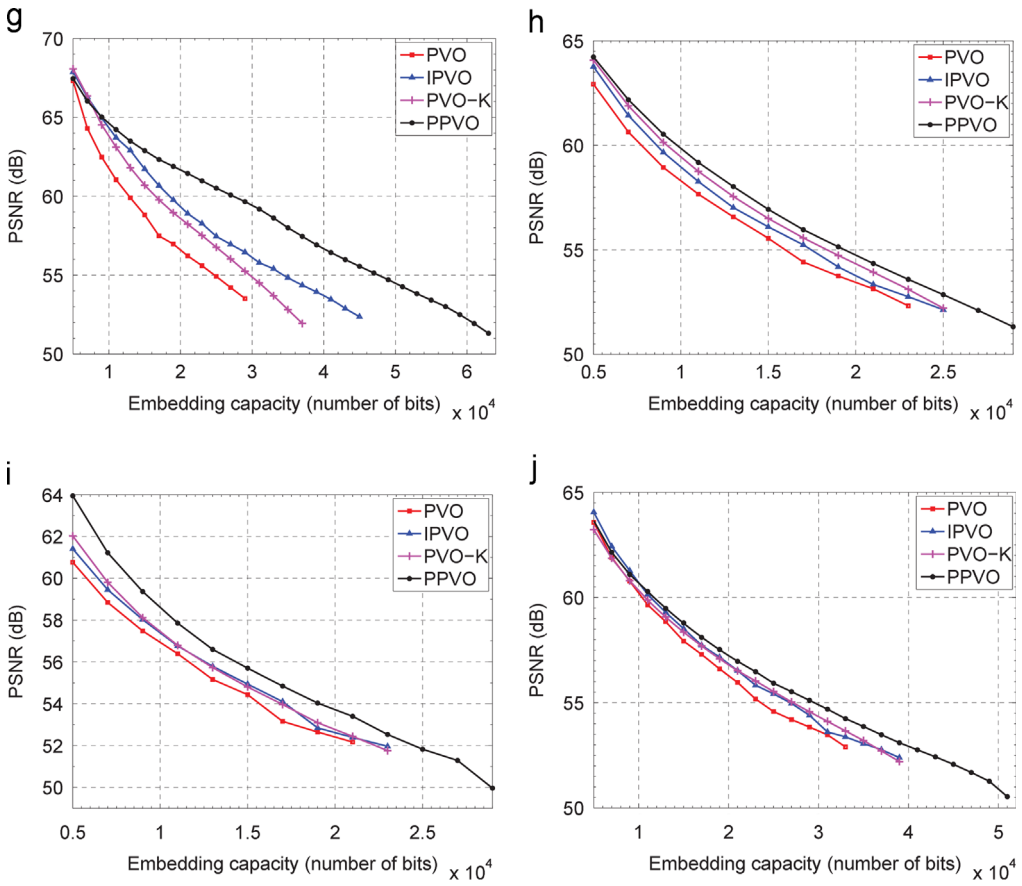

Fig. 11. (*continued*)

Table 1
Comparisons in terms of embedding capacity (number of bits).

Image	PVO	IPVO	PVO-K	PPVO
Lena	31,000	38,000	37,000	46,000
F16	38,000	52,000	48,000	69,000
Baboon	13,000	13,000	13,000	15,000
Barbara	25,000	29,000	29,000	33,000
Boat	24,000	26,000	26,000	30,000
Peppers	27,000	30,000	31,000	33,000
House	30,000	46,000	37,000	64,000
Sailboat	23,000	26,000	26,000	29,000
Elaine	21,000	24,000	23,000	29,000
Tiffany	33,000	40,000	40,000	52,000
Average	**26,500**	**32,400**	**31,000**	**40,000**

Table 2
Comparisons in terms of PSNR (dB) for [7,8,24] and PVO [25], IPVO [26], PVO-K [27] and PPVO with the payload of 10,000 bits.

Image	[7]	[8]	[24]	[25]	[26]	[27]	PPVO
Lena	58.4	58.1	59.6	60.3	60.5	60.6	60.3
F16	61.1	61.2	63.7	62.0	62.9	63.3	63.7
Baboon	52.5	54.0	55.2	53.5	53.6	54.5	54.2
Barbara	58.3	58.6	59.4	59.8	60.1	60.2	59.8
Boat	55.9	55.5	57.4	58.1	58.3	58.2	58.4
Peppers	55.4	56.1	56.1	58.9	59.0	59.2	58.8
House	63.6	64.0	64.8	61.8	64.4	63.7	64.6
Sailboat	56.2	58.2	58.6	58.2	58.9	59.4	59.8
Elaine	55.4	56.7	58.0	56.8	57.3	57.4	58.7
Tiffany	59.5	56.0	59.4	60.1	60.6	60.3	60.6
Average	**57.6**	**57.8**	**59.2**	**58.9**	**59.6**	**59.7**	**59.9**

4. Experimental results

In this section, we validate the performance of the proposed PPVO compared with PVO, IPVO and PVO-K using two criteria: the embedding capacity and the peak signal-to-noise ratio (PSNR) value. The test images (except Barbara) in the SIPI image database[1] are all eight-bit gray-scale images with size 512×512 as shown in Fig. 10. The other reversible data hiding methods involved in the comparison are from [7,8,24]. All payloads used in the experiment were random bit stream and all experiments were performed with Matlab.

For PVO, IPVO and PVO-K, the block sizes w and h used in our experiment were from 5 to 2 such that for each given embedding capacity, 16 times embedding with those 16 different block sizes should be performed. The block size that generates the highest PSNR value was used. For PPVO, the candidate number of context pixels is from 1 to 15, therefore, the embedding should be conducted 15 times for each

[1] http://sipi.usc.edu/database

Table 3
Comparisons in terms of PSNR (dB) for [7,8,24] and PVO [25], IPVO [26], PVO-K [27] and PPVO with the payload of 20,000 bits.

Image	[7]	[8]	[24]	[25]	[26]	[27]	PPVO
Lena	55.1	54.8	56.2	56.2	56.5	56.6	56.7
F16	58.5	56.8	60.1	58.1	59.0	59.3	59.9
Barbara	54.7	55.2	56.1	54.7	55.2	55.9	55.6
Boat	52.2	52.4	53.3	53.3	53.9	53.7	54.2
Peppers	52.1	52.5	52.8	54.7	54.7	54.9	55.0
House	60.0	61.0	61.6	5.4	59.3	58.5	61.6
Sailboat	53.0	53.2	53.6	53.4	53.6	54.3	54.7
Elaine	52.3	51.6	52.9	52.4	52.6	52.7	53.7
Tiffany	56.0	53.9	56.4	56.3	56.7	56.7	57.2
Average	**54.9**	**54.6**	**55.9**	**55.0**	**55.7**	**55.9**	**56.5**

Table 4
Comparisons of Kodak image data set in terms of PSNR (dB) for [24], PVO [25], IPVO [26], PVO-K [27] and PPVO with the payload of 10,000 bits.

Image	[24]	[25]	[26]	[27]	PPVO
kodim01	63.64	57.98	61.59	61.44	64.23
kodim02	64.23	62.83	64.06	61.97	64.49
kodim03	65.04	63.84	65.37	63.57	65.11
kodim04	63.79	62.00	63.62	63.46	64.17
kodim05	63.38	59.64	61.61	54.62	62.51
kodim06	66.70	61.36	65.02	60.62	66.05
kodim07	64.42	63.64	65.12	63.06	64.83
kodim08	57.04	52.73	56.23	63.50	57.83
kodim09	61.73	62.79	63.72	62.68	63.34
kodim10	60.98	61.90	63.11	60.32	62.68
kodim11	64.83	62.46	65.19	62.66	65.42
kodim12	64.45	63.00	64.50	62.96	64.75
kodim13	58.46	52.16	54.53	53.95	58.20
kodim14	62.83	59.51	61.35	62.86	62.62
kodim15	61.10	62.35	64.46	62.06	62.86
kodim16	64.81	63.03	64.98	63.42	65.06
kodim17	62.91	62.03	63.76	60.36	64.52
kodim18	59.36	59.63	60.74	63.96	61.02
kodim19	62.03	62.53	63.36	62.57	63.19
kodim20	52.26	61.14	65.54	60.74	57.76
kodim21	61.78	62.54	63.71	63.22	63.58
kodim22	61.43	61.36	62.59	63.74	63.00
kodim23	63.16	63.21	64.58	56.47	64.35
kodim24	56.93	58.80	62.20	62.88	62.22
Average	**61.97**	**60.34**	**62.96**	**61.55**	**63.08**

payload. Fig. 11 shows the performance comparisons of PVO, IPVO, PVO-K and PPVO. The embedding capacity is from 5000 bits to their maximum embedding capacity with a step size of 2000 bits. It can be seen that PPVO has larger embedding capacity than PVO, IPVO and PVO-K for all the test images. For example, the embedding capacities of PVO, IPVO, PVO-K and PPVO for F16 are 38,000 bits, 52,000 bits, 48,000 bits and 69,000 bits, respectively. The complete comparison of embedding capacity of PVO, IPVO, PVO-K and PPVO is shown in Table 1.

In terms of PSNR value, PPVO has higher PSNR values in most cases. The superiority of PPVO is clearly demonstrated with large payload size. For very smooth images, e.g., House, the PSNR of PPVO is more than 5 dB higher than that of PVO and more than 2 dB higher than that of IPVO and PVO-K. To compare with other state-of-the-art methods, 10,000 and 20,000 bits are embedded using PVO, IPVO, PVO-K, PPVO, [7,8,24]. The results are shown in Tables 2 and 3, respectively. Baboon is eliminated in Table 3 because PVO, IPVO, PVO-K and PPVO cannot achieve an embedding capacity of 20,000 bits. The average PSNR value of PPVO is 1.0 dB, 0.3 dB, 0.2 dB, 2.3 dB, 2.1 dB and 0.7 dB higher than that of PVO, IPVO, PVO-K, [7,8,24], respectively, when the payload is 10,000 bits. The average PSNR value of PPVO is 1.5 dB, 0.8 dB, 0.6 dB, 1.6 dB, 1.9 dB and 0.6 dB higher than that of PVO, IPVO, PVO-K, [7,8,24], respectively, when the payload is 20,000 bits. Note that although the recent proposed pair-wise PEE in [24] has impressive performance for small payloads, however, PPVO still has better performance.

Table 5
Comparisons of Kodak image data set in terms of PSNR (dB) for [24], PVO [25], IPVO [26], PVO-K [27] and PPVO with the payload of 20,000 bits. The average PSNR value is calculated without kodim05, kodim08 and kodim13 due to incomplete data.

Image	[24]	[25]	[26]	[27]	PPVO
kodim01	59.87	53.36	56.37	55.52	60.47
kodim02	60.82	58.76	60.52	58.57	61.37
kodim03	61.88	60.16	62.00	59.04	62.42
kodim04	60.33	58.02	59.99	59.89	60.85
kodim05	59.96	55.10	57.53	0	58.94
kodim06	63.32	56.47	61.45	56.27	63.45
kodim07	61.20	60.05	61.81	59.90	62.03
kodim08	54.63	0	51.98	59.91	55.10
kodim09	58.36	59.09	60.23	58.89	60.50
kodim10	58.13	58.24	59.71	56.24	59.98
kodim11	61.48	57.38	61.23	58.93	62.63
kodim12	61.09	58.98	61.03	59.47	61.80
kodim13	53.58	0	0	0	53.76
kodim14	58.99	55.28	57.27	59.09	58.68
kodim15	59.51	58.30	61.32	58.22	61.99
kodim16	61.42	58.55	61.52	60.10	62.22
kodim17	59.63	58.07	59.99	56.87	61.17
kodim18	56.50	54.82	56.47	60.56	57.48
kodim19	58.87	58.71	59.82	58.90	60.14
kodim20	51.94	56.95	62.40	56.42	58.15
kodim21	58.59	58.66	60.17	58.50	60.87
kodim22	58.40	57.15	58.66	60.23	59.56
kodim23	60.35	59.68	61.12	53.06	61.61
kodim24	56.14	54.81	58.59	59.23	60.49
Average	**59.15**	**57.69**	**60.08**	**58.28**	**60.85**

It is pointed out that for many practical applications, the embedding capacity of PVO [25] is likely sufficient. However, a higher embedding capacity is sometimes desired. The proposed PPVO has higher embedding capacity without reducing the PSNR value, so it can be applied in more applications.

The methods in [7,8] are capable of embedding large payload and are not optimized for the moderate size payload. PVO, IPVO, PVO-K and [24] targets the moderate size payload, therefore, we further compare PPVO with these four methods with a larger image data set: the Kodak.[2]

The Kodak image data set contains 24 color images with a size of 512×768 or 768×512. All test images are transformed into gray-scale and embedded with 10,000 and 20,000 random bits. The comparison is shown in Tables 4 and 5. The average PSNR value of PPVO is

[2] http://r0k.us/graphics/kodak/

1.11 dB, 2.74 dB, 0.12 dB and 1.53 db higher than that of [24], PVO, IPVO and PVO-K, respectively, when the payload is 10,000 bits. The average PSNR value of PPVO is 1.7 dB, 3.16 dB, 0.77 dB and 2.57 db higher than that of [24], PVO and IPVO and PVO-K, respectively, when the payload is 20,000 bits. The proposed PPVO achieved very impressive embedding performance and is suitable for applications that require high-quality marked images.

5. Conclusion

In this paper, a novel pixel-based PVO reversible data hiding method is proposed. PPVO was demonstrated to have a larger embedding capacity and higher marked image quality than PVO , IPVO and PVO-K. The experiments also showed that PPVO outperformed three other state-of-the-art methods with moderate payloads.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MEST) (No. 2012015587). This work was supported by the ICT R&D program of MSIP/IITP. [2014 (I5501-14-1007), 3D Smart Media/Augmented Reality Technology, Korea-China-Japan-Russia Cooperation International Standardization]. This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1001-0004). This research was supported by Korea University.

References

[1] J. Tian, Reversible data embedding using a difference expansion, IEEE Trans. Circuits Syst. Video Technol. 13 (2003) 890–896.
[2] Z. Ni, Y.-Q. Shi, N. Ansari, W. Su, Reversible data hiding, IEEE Trans. Circuits Syst. Video Technol. 16 (2006) 354–362.
[3] A.M. Alattar, Reversible watermark using the difference expansion of a generalized integer transform, IEEE Trans. Image Process. 13 (2004) 1147–1156.
[4] 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. Forensics Security 3 (2008) 456–465.
[5] D.M. Thodi, J.J. Rodríguez, Expansion embedding techniques for reversible watermarking, IEEE Trans. Image Process. 16 (2007) 721–730.
[6] Y. Hu, H.-K. Lee, J. Li, De-based reversible data hiding with improved overflow location map, IEEE Trans. Circuits Syst. Video Technol. 19 (2009) 250–260.
[7] 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 (2009) 989–999.
[8] X. Li, B. Yang, T. Zeng, Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection, IEEE Trans. Image Process. 20 (2011) 3524–3533.
[9] X. Gui, X. Li, B. Yang, A high capacity reversible data hiding scheme based on generalized prediction-error expansion and adaptive embedding, Signal Process. 98 (2014) 370–380.
[10] D. Coltuc, Improved embedding for prediction-based reversible watermarking, IEEE Trans. Inf. Forensics Security 6 (2011) 873–882.
[11] D. Coltuc, Low distortion transform for reversible watermarking, IEEE Trans. Image Process. 21 (2012) 412–417.
[12] D. Coltuc, I.-C. Dragoi, Context embedding for raster-scan rhombus based reversible water marking, in: Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security (ACM IHMMSEC), ACM , Montpellier, France, June 17–19, 2013, pp. 215–220.
[13] J. Li, X. Li, B. Yang, Reversible data hiding scheme for color image based on prediction-error expansion and cross-channel correlation, Signal Process. 93 (2013) 2748–2758.
[14] W. Hong, T.-S. Chen, C.-W. Shiu, Reversible data hiding for high quality images using modification of prediction errors, J. Syst. Softw. 82 (2009) 1833–1842.
[15] 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 (2009) 906–910.
[16] Y.-Y. Tsai, D.-S. Tsai, C.-L. Liu, Reversible data hiding scheme based on neighboring pixel differences, Digital Signal Process. 23 (2013) 919–927.
[17] K.-S. Kim, M.-J. Lee, H.-Y. Lee, H.-K. Lee, Reversible data hiding exploiting spatial correlation between sub-sampled images, Pattern Recognit. 42 (2009) 3083–3096.
[18] L. Luo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, Reversible image watermarking using interpolation technique, IEEE Trans. Inf. Forensics Security 5 (2010) 187–193.
[19] X. Chen, X. Sun, H. Sun, L. Xiang, B. Yang, Histogram shifting based reversible data hiding method using directed-prediction scheme, Multimed. Tools Appl. (2014) 1–19.
[20] W. Hong, Adaptive reversible data hiding method based on error energy control and histogram shifting, Opt. Commun. 285 (2012) 101–108.
[21] X. Li, B. Li, B. Yang, T. Zeng, General framework to histogram-shifting-based reversible data hiding, IEEE Trans. Image Process. 22 (2013) 2181–2191.
[22] W. Fan, Z. Chen, M. Chen, L. Luo, Z. Xiong, Reversible data hiding with context modeling, generalized expansion and boundary map, Multimed. Tools Appl. 57 (2012) 477–499.
[23] X. Li, W. Zhang, X. Gui, B. Yang, A novel reversible data hiding scheme based on two-dimensional difference-histogram modification, IEEE Trans. Inf. Forensics Security 8 (2013) 1091–1100.
[24] B. Ou, X. Li, Y. Zhao, R. Ni, Y. Shi, Pairwise prediction-error expansion for efficient reversible data hiding, IEEE Trans. Image Process. 22 (2013) 5010–5021.
[25] 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 (2013) 198–205.
[26] F. Peng, X. Li, B. Yang, Improved pvo-based reversible data hiding, Digital Signal Process. 25 (2014) 255–265.
[27] Bo Ou, Xiaolong Li, Yao Zhao, Rongrong Ni, Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion, Signal Process.: Image Commun. 29 (7) (2014) 760–772, http://dx.doi.org/10.1016/j.image.2014.05.003.