Improved Pixel-based Pixel-Value-Ordering High-fidelity Reversible Data Hiding

Haorui Wu, Xiaolong Li, Yao Zhao*, Rongrong Ni

[a]*Institute of Information Science, Beijing Jiaotong University, Beijing 100044, China*
[b]*Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing 100044, China*

Abstract

reversible data hiding (RDH)

Keywords: Reversible data hiding, pixel-value-ordering, adaptive embedding, multiple histograms modification

---

1. Introduction

2. Related Works

In this section, as prior knowledge, we will review the improved PVO-based RDH method of Peng *et al.* [1] and the Pixel-based PVO RDH(PPVO) method of Qu *et al.* [2]. And the data embedding amd extraction procedures are presented followed.

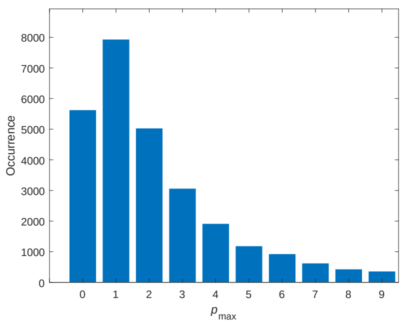
2.1. Improved PVO-based RDH [1]

In original PVO-based method [3] of Li *et al.*, first, the cover image is divided into equal size and non-overlapping blocks with size of $n = a \times b$. And then pixels $(x_1, ..., x_n)$ in the given block is sorted in ascending by the pixel values, to obtained $(x_{\xi(1)}, ..., x_{\xi(n)})$, where $\xi : \{1, ..., n\} \to \{1, ..., n\}$ is the one-to-one mapping which respecting the rules of $\xi(i) < \xi(j)$ if $x_i = x_j$ $(i < j)$. Then, the largest pixel $x_{\xi(n)}$ is predicted by the second largest pixel $x_{\xi(n-1)}$ and the prediction-error of $x_{\xi(n)}$ is defined as $p_{\max} = x_{\xi(n)} - x_{\xi(n-1)}$, where the prediction-error is subject to the interval of $[0, \infty)$ due to the truth $x_{\xi(n)} \geq x_{\xi(n-1)}$. In this method, $p_{\max} = 1$ appears most often so that prediction-errors of $p_{\max} = 1$ are expanded to embed data while others of $p_{\max} > 1$ are shifted to ensure reversibility. However, in original PVO-based RDH, blocks of $p_{\max} = 0$ are not used which are usually smooth and beneficial for reversible embedding. Based on this motivation, Peng *et al.* propose an improved PVO-based RDH method [1].

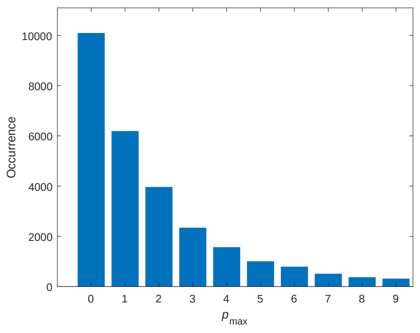
In the improved PVO-based RDH method, for each block, the computation of $p_{\max}$ is redefined by considering the order of $\xi(n-1)$ and $\xi(n)$ as

$$p_{\max} = \begin{cases} x_{\xi(n)} - x_{\xi(n-1)}, & \text{if } \xi(n) > \xi(n-1) \\ x_{\xi(n)} - x_{\xi(n-1)} - 1, & \text{if } \xi(n) < \xi(n-1) \end{cases}, \tag{1}$$

*Corresponding author. Tel./Fax: +86 10 51688667.
Email addresses: `hrwu@bjtu.edu.cn` (Haorui Wu), `lixl@bjtu.edu.cn` (Xiaolong Li), `yzhao@bjtu.edu.cn` (Yao Zhao), `rrni@bjtu.edu.cn` (Rongrong Ni)



(a) Original PVO-based method [3].



(b) Improved PVO-based method [1].

Figure 1: PEH of $p_{\max}$ with block size $n = 3 \times 3$ for the standard gray-scale 512×512 sized image Lena.

where the prediction-errors $p_{\max} \geq 0$ locating in the interval of $[0, \infty)$ as well.

The different of improved PVO-based method and original PVO-based method is that a part of blocks whose prediction-errors large than 1 in original method, i.e., $x_{\xi(n)} - x_{\xi(n-1)} \geq 1$, but with the order $\xi(n) < \xi(n-1)$, are recomputed to $x_{\xi(n)} - x_{\xi(n-1)} - 1$, resulting the PEH peaking at 0 instead of 1 in [3]. For example, PEH of the standard gray-scale image Lena is shown as Fig. 1.

More clearly, prediction-error $p_{\max}$ is modified to derive the marked prediction-error $\hat{p}_{\max}$ by

$$\hat{p}_{\max} = \begin{cases} p_{\max} + b, & \text{if } p_{\max} = 0 \\ p_{\max} + 1, & \text{if } p_{\max} \geq 1 \end{cases}, \tag{2}$$

where $b \in \{0, 1\}$ is one bit secret data to be embedded. And, the marked largest pixel $\hat{x}_{\xi(n)}$ is obtained by

$$\hat{x}_{\xi(n)} = x_{\xi(n-1)} + \hat{p}_{\max}. \tag{3}$$

In the improved PVO-based method, in each block, it is note that $\hat{x}_{\xi(n)} \geq x_{\xi(n)}$, and the PVO of the marked pixels $(\hat{x}_{\xi(1)}, ..., \hat{x}_{\xi(n)})$ is as the same as the original pixels, which guarantees the reversibility.

For the decoder, in each block, the marked prediction-error $\hat{p}_{\max}$ with pixels $(\hat{x}_{\xi(1)}, ..., \hat{x}_{\xi(n)})$ in ascending order is computed as

$$\hat{p}_{\max} = \begin{cases} \hat{x}_{\xi(n)} - \hat{x}_{\xi(n-1)}, & \text{if } \xi(n) > \xi(n-1) \\ \hat{x}_{\xi(n)} - \hat{x}_{\xi(n-1)} - 1, & \text{if } \xi(n) < \xi(n-1) \end{cases}, \tag{4}$$

Next, the original pixel value $x_{\xi(n)}$ is recovered by

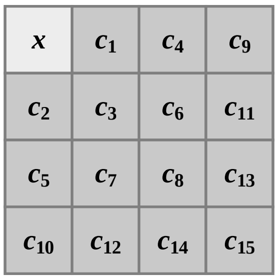
$$x_{\xi(n)} = \begin{cases} \hat{x}_{\xi(n)}, & \text{if } \hat{p}_{\max} = 0 \\ \hat{x}_{\xi(n)} - 1, & \text{if } \hat{p}_{\max} \geq 1 \end{cases}, \tag{5}$$

while other marked pixels $(\hat{x}_{\xi(1)}, ..., \hat{x}_{\xi(n-1)})$ is recovered as themselves. And, the embedded secret data is 0 if $\hat{p}_{\max} = 0$ and 1 if $\hat{p}_{\max} = 1$.

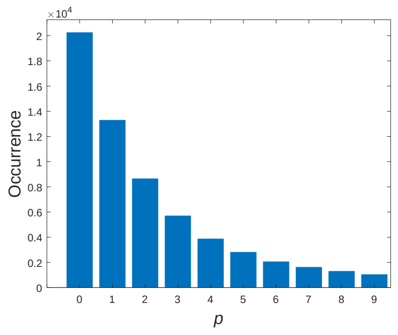
Moreover, for each block, the similar process is applied to the smallest pixel value $x_{\xi(1)}$ by computing the prediction-error $p_{\min}$ as

$$p_{\min} = \begin{cases} x_{\xi(2)} - x_{\xi(1)}, & \text{if } \xi(2) > \xi(1) \\ x_{\xi(2)} - x_{\xi(1)} - 1, & \text{if } \xi(2) < \xi(1) \end{cases}, \tag{6}$$

where pixel $p_{\min} \geq 0$. And the PEH of $p_{\min}$ peaks at 0 as well. The detail embedding and extraction process is described in [1].



(a) Context pixels of x.



(b) PEH of p by (7).

Figure 2: PPVO Context Pixels and show the shaper histogram with $CN = 15$.

2.2. Pixel-based PVO RDH [2]

In the classical PVO-based RDH method such as the original PVO-base method [3] and the improved PVO-based method [1], the cover image is first devided into equal size and non-overlapping blocks. The sorted pixels $(x_{\xi(2)}, ..., x_{\xi(n-1)})$ is not utilized in the embedding process and the total number of prediction-errors is bounded by the block constraint, which causing the difficulty of efficient embedding data into smooth region. In [2], Qu *et al.* propose a pixel-based PVO RDH method, which is called PPVO, to utilized the sorted pixels for data embedding.

PPVO achieves an pixel-by-pixel data embedding process. For current pixel, the context pixels defined as the pixels in the lower right direction and form an vector $C = \{c_1, ..., c_{\text{CN}}\}$ shown in Fig. 2(a), where the number of context pixels is defined as CN.

Before describe the prediction algorithm, to simplify the explanation, pixels participating in the embedding process are grouped into four sets, which are

$$\begin{aligned} S_1 &= \{x|\max(C) \neq \min(C), x \geq \max(C)\} \\ S_2 &= \{x|\max(C) \neq \min(C), x \leq \min(C)\} \\ S_3 &= \{x|\max(C) = \min(C), x \leq \min(C), \min(C) \neq 254\} \\ S_4 &= \{x|\max(C) = \min(C), x \geq \min(C), \min(C) = 254\} \end{aligned} .$$

Then, for each pixel x, the prediction-error p is calculated by

$$p = \begin{cases} x - \max(C), & \text{if } x \in S_1 \\ \min(C) - x, & \text{if } x \in S_2 \cup S_3 \\ 0, & \text{if } x \in S_4 \\ \text{skip}, & \text{otherwise} \end{cases} , \tag{7}$$

where prediction-error p is located in the interval of $[0, \infty)$. More specifically, PEH of p peak at 0 as shown in Fig. 2(b) so that p is modified to derive the marked prediction-error $\hat{p}$ by

$$\hat{p} = \begin{cases} p + b, & \text{if } p = 0 \\ p + 1, & \text{if } p \geq 1 \end{cases} , \tag{8}$$

where $b \in \{0,1\}$ is one bit secret data to be embedded. And the marked pixel $\hat{x}$ is obtained by

$$\hat{x} = \begin{cases} \max(C) + \hat{p}, & \text{if } x \in S_1 \\ \min(C) - \hat{p}, & \text{if } x \in S_2 \cup S_3 \\ \min(C) + \hat{p}, & \text{if } x \in S_4 \\ \text{skip}, & \text{otherwise} \end{cases}. \tag{9}$$

In this method, for each pixel satisfying $x \notin S_4$, marked pixel satisfies $\hat{x} \geq x$ if $x \geq \max(C)$ and $\hat{x} \leq x$ if $x \leq \min(C)$ which ensures reversibility. And reason of processing the pixels $x \in S_4$ specially is that pixels equal 255 are modified to 254 during the location map stage and $x \in S_4$ is increased by 1 or 0 can decrease the distortion of data embedding. It is note that, in the embedding process, the proper NC is selected by choosing the best result of $C = \{1, ..., i\}(i \in 1, ..., 15)$ exhaustively.

For the decoder, to extract data and recover the ordinal pixel, the same marked prediction-error $\hat{p}$ of current pixel x is calculated by

$$\hat{p} = \begin{cases} \hat{x} - \max(C), & \text{if } \hat{x} \in S_1 \\ \min(C) - \hat{x}, & \text{if } \hat{x} \in S_2 \cup S_3 \\ \hat{x} - \min(C), & \text{if } \hat{x} \in S_4 \\ \text{skip}, & \text{otherwise} \end{cases}. \tag{10}$$

In addition, the original pixel value x is recovered by

$$x = \begin{cases} \hat{x}, & \text{if } \hat{p} = 0 \\ \hat{x} - 1, & \text{if } \hat{x} \in S_1 \cup S_4 \text{ and } \hat{p} \geq 1 \\ \hat{x} + 1, & \text{if } \hat{x} \in S_2 \cup S_3 \text{ and } \hat{p} \geq 1 \\ \text{skip}, & \text{otherwise} \end{cases}. \tag{11}$$

With the recover process, the secret data is extracted that $b = 1$ if $\hat{p} = 1$ and $b = 0$ if $\hat{p} = 0$.

The experimental results show that PPVO gets a large maximum embedding capacity than the previous PVO-based method [1, 3] by pixel-by-pixel embedding. And it obtains a sharper histogram which is causing a result of better performance as shown in Fig. 2.

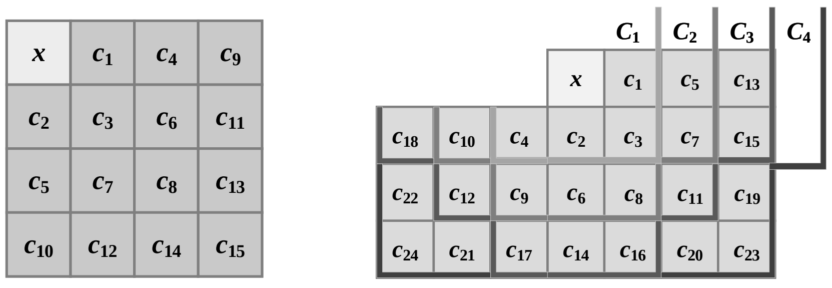
3. Proposed Method

In this section, a extended PPVO predictor is first introduced. Then, a multi-size based embedding method for MHM is proposed to further improve the performance. Finally, the embedding procedure and extraction procedure is described.

sec:2.1

3.1. Extended PPVO predictor

Compared with PVO-based methods such as PVO[3], IPVO[1] and PVO-k[4], in PPVO, every pixel is predicted by context pixels, breaking through the block constraint. In PPVO, to ensure reversibility, pixels in lower right direction are utilized as context region to predict the current pixel which is shown in Fig. 3(a). The prediction is more accurate, however, the surrounding context information is not fully utilized. There are some other pixels are omitted which is beneficial to provide more useful information for prediction.



(a) PPVO. (b) Extended PPVO.

Figure 3: Context pixels of PPVO(left) and Extended PPVO(right).

Clearly, for example, in Fig. 3(b), pixels $\{c_4, c_9, c_{10}, c_{12}, c_{17}, c_{18}, c_{21}, c_{22}, c_{24}\}$ in left lower direction should be used in prediction procedure as one part of context pixels which is omitted in PPVO. Then the extended PPVO predictor is introduced to show how to make better prediction.

Here, the symbol definitions are the same as the definitions in section 2. In Fig. 3, the number of context pixels is $CN = 24$, where context pixel vector is $C = (c_1, ..., c_{24})$. And the prediction procedure is described as follow. Compared with PPVO, except the four sets S_1, S_2, S_3 amd S_4, another set is defined as

$$S_5 = \{x | \max(C) = \min(C), x > \max(C), \max(C) \neq 254\} \ .$$

And, for each pixel x, the prediction-error is calculated by

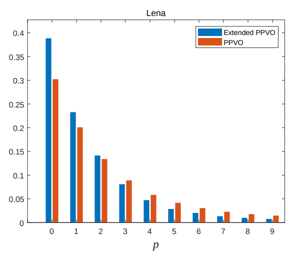
$$p = \begin{cases} x - \max(C), & \text{if } x \in S_1 \\ \min(C) - x, & \text{if } x \in S_2 \cup S_3 \\ 0, & \text{if } x \in S_4 \\ x - \max(C) - 1, & \text{if } x \in S_5 \\ \text{skip}, & \text{otherwise} \end{cases} , \tag{12}$$

where prediction-error p is located in the interval of $[0, \infty)$. By the extended PPVO predictor, with the situation of $\max(C) = \min(C)$, pixel x satisfying $x > \max(C)$ is utilized to obtain prediction-errors. More pixels participate in prediction procedure, leading to an increase in embedding capacity.

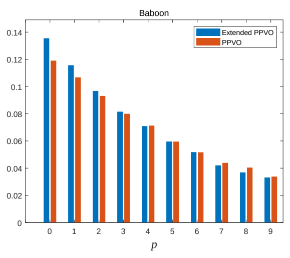
Then, with the obtained prediction-error by (12), secret data $b \in \{0, 1\}$ can be embedded by modifying p to derive $\hat{p}$ by (8). Meanwhile, the marked pixel $\hat{x}$ can be modified by

$$\hat{x} = \begin{cases} \max(C) + \hat{p}, & \text{if } x \in S_1 \\ \min(C) - \hat{p}, & \text{if } x \in S_2 \cup S_3 \\ \min(C) + \hat{p}, & \text{if } x \in S_4 \\ \max(C) + \hat{p} + 1, & \text{if } x \in S_5 \\ \text{skip}, & \text{otherwise} \end{cases} . \tag{13}$$

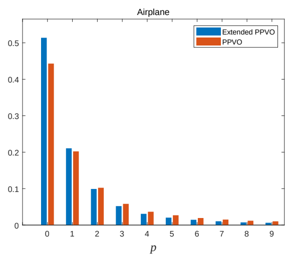
In the above embedding procedure, prediction-error p increased by 1 or remain the same, i.e., $\hat{p} \geq p$. Thus, for original pixel value of $x \in S_1 \cup S_4 \cup S_5$, the marked pixel value $\hat{x}$ is up to 1 larger than x. The both pixel values before and after modification belong to the same collection so that the reversibility is guaranteed for the decoder. In addition, it is note that



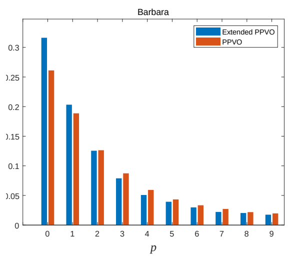
(a) Lena



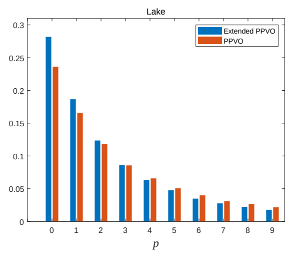
(b) Baboon



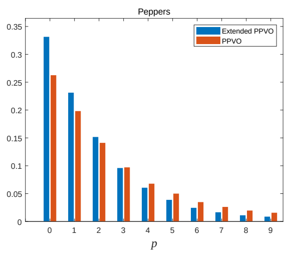
(c) Airplane



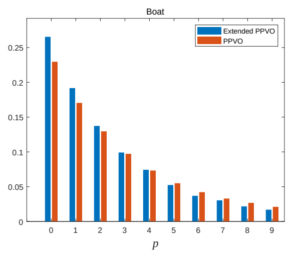
(d) Barbara



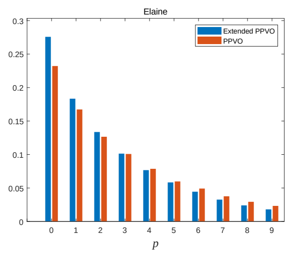
(e) Lake



(f) Peppers



(g) Boat



(h) Eliane

Figure 4: Comparison of normalized prediction-errors p of extended PPVO by (12) in blue and PPVO by (7) in orange.

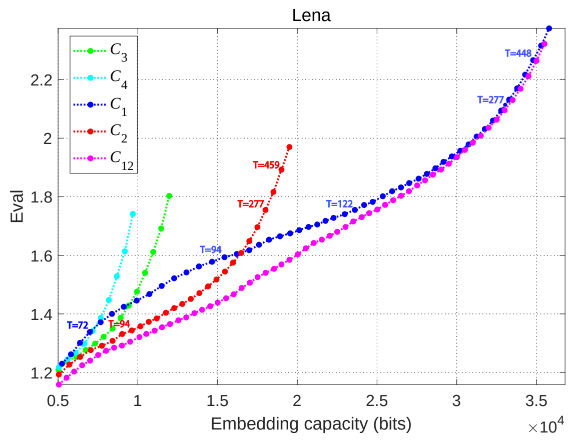
For extended PPVO predictor, more surrounding pixels are fully considered as pixel context to predict current pixel, aiming to get sharper histogram. To show the superiority of extended PPVO predictor, compared with Qu *et al.*' method, an comparison shown in Fig. 4. In the example, eight standard gray-scale 512×512 sized images are used and comparison of normalized PEHs of proposed extended PPVO predictor and PPVO predictor are displayed on the same figures. For extended PPVO predictor, pixels $\{c_1, c_2, c_3, c_4, c_5, c_6, c_9, c_{10}\}$ in Fig. 3(b) are seen as context pixels, and for PPVO predictor, pixels $\{c_1, ..., c_8\}$ in Fig. 3(a) are seen as context pixels. Obviously, both PEHs have high value at small prediction-errors and as the larger the error, the fewer the number of corresponding pixels. It is note that normalized PEHs of extended PPVO predictor have higher peaks at 0, and the distributions are sharper, which is beneficial to the final embedding performance. Moreover, in the next data embedding procedure, pixels of $p = 0$ are used to embed secret data by expanding and others are shifted to ensure reversibility. Thus, we use embedding distortion of one bit data embedding to evaluate the distortion caused by histograms theoretically, numerically expressed as

$$\mathbf{Eval} = \frac{\mathrm{H}(0)}{\frac{1}{2}\mathrm{H}(0) + \sum_{i \geq 1} \mathrm{H}(i)},$$

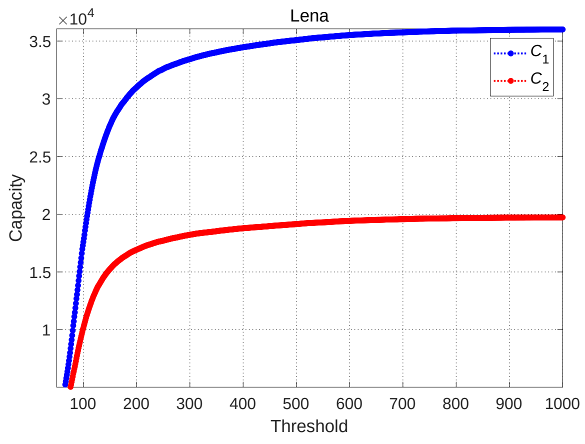
where H is the PEH and H(i) is the frequency of pixels whose prediction-errors $p = i$. This evaluation describes the distortion caused by embedding one bit data on H. In Lena, Baboon and Airplane, **Eval** of extended PPVO predictor are 2.07, 6.88 and 1.44 respectively, while value of evaluation are 2.81, 7.90 and 1.76 of PPVO predictor. It shows that data embedding on PEH of extended PPVO predictor may carry much less distortion so that better performance can be obtained.

3.2. Multi-size based Embedding Method for MHM

Actually, in above embedding procedure, CN of pixel value x to be predicted is set in advance. For a cover image, all the pixels are predicted by the same context pixels. In the proposed method, considering



(a) Lena



(b) Lena

Figure 5: Eval.

the complexity, we consider CN ≤ 24 and to simplify the presentation, as shown in Fig. 3(b), context pixels are group into four sets by the size of the context region, including $C_1 = \{c_1, ..., c_4\}$, $C_2 = \{c_1, ..., c_{10}\}$, $C_3 = \{c_1, ..., c_{18}\}$ and $C_4 = \{c_1, ..., c_{24}\}$. It is expected that the distortion is different by using different size context areas for prediction. In Fig. 5, we use **Eval** to evaluate the data embedding, and X-axis represents the embedding capacity. It is prefer that **Eval** is as small as possible. In Fig. 5(a), the curves show that for the small capacity, distortion by using context pixels with C_2 is lowest but there is a small maximum embedding capacity less than 20,000 bits. And for the large capacity, distortion by using context pixels with C_1 gets the smallest value and it has a large maximum embedding capacity larger than 36,000 bits. In this example, we focus on the two curves of C_1 and C_2. Next, we propose a such scheme called a multi-size based embedding method for MHM, which combines the advantages of both size context, with low distortion and large maximum capacity. .

Firstly, to embed data into the cover image, it is prefer to select pixels in smooth context region for PEH generation. And the complexity NL of one pixel is evaluated by the sum of the absolution differences of the horizontal and vertical of every two adjacent pixels belong to the corresponding context. In Fig. 5, for each pixel, 24 context pixels of C_4 are used to compute NL for all curves. A threshold T is set to choose the smooth pixels whose complexity less than the threshold to generate PEH so that all secret data can just be embedded. Some thresholds are marked on the curves of C_1 and C_2. It shows that larger size context has a bigger threshold for a given capacity. This is because the large size context must skip much more pixels, and bigger threshold provides more available pixels for prediction. It is shown in Fig. 5(b) as well. Specifically, let $X_T = \{x| \text{ NL}(x) < T\}$ indicates the set of pixels with complexity less than T. The **Eval** of X_{72} with C_1 is 1.34, which is approximately equal to the X_{94} with C_2. The **Eval** of the X_{94} with C_1 equals 1.59 and it is much larger than 1.34. It means that pixels in $X_{94} \setminus X_{72} = \{x| \ 72 \leq \text{NL}(x) < 94\}$ with C_1 for prediction bring much more distortion than C_2. In other words, it will effectively reduce the distortion if pixels $x \in X_{72}$ are predicted by the context pixels with C_1 and pixels $x \in X_{94} \setminus X_{72}$ are predicted by the context pixels with C_2. The similar conclusion can be obtained on sets X_{122} and $X_{277} \setminus X_{122}$. Based on the above consideration, a detailed description of the multi-size based embedding method is given.

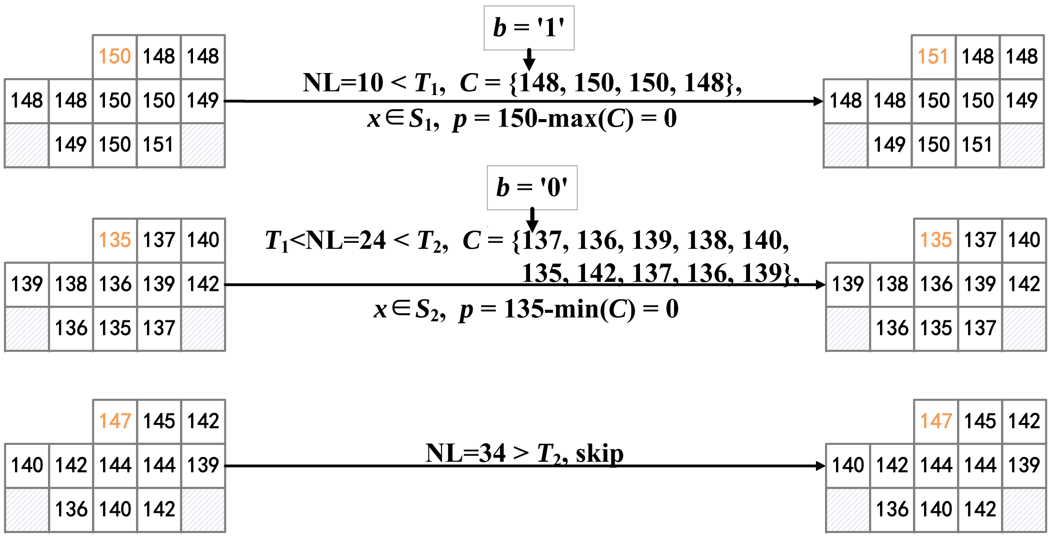


Figure 6: Example.

In the proposed method, if two size of C_1 and C_2 are utilized for prediction, pixels on the cover images are divided into three parts by two thresholds T_1, T_2 $(0 \leq T_1 \leq T2)$. The context of pixels of X_{T_1} are seen as smooth region, and these pixels are predicted by the context pixels of C_1. Pixels of $X_{T2} \setminus X_{T_2}$ are predicted by the context pixels of C_2, and other pixels are all omitted. It is reasonable to use as many pixels as possible in the smoothing area during the embedding procedure. Therefore, as shown in Fig. 5(b), the pixels in smooth regions X_{T_1} is predicted by the context pixels of C_1. This provides much more embedding capacity. And for the pixels in the normal regions X_{T_2}, pixels of C_2 are chosen as context pixels to predict, making the distortion as small as possible. Fig. 6 shows examples of the the embedding procedures for the three types of pixels, where $(T_1, T_2) = (15, 25)$. For the first pixel value $x = 150$, because the complexity $\mathrm{NL} = 10$ of the pixel less than the small threshold T_1, the pixel is in a smooth region and context pixels $C = \{148, 150, 150, 148\}$ are chosen. The prediction-error of the pixel value is $p = 0$. One bit secret data $b = 1$ can be embedded by increasing the x by 1 to $\hat{x} = 151$. However, if context pixels of C_2 are chosen for prediction, the pixel wound be skipped because of $\min(C) < x < \max(C)$. For the second pixel $x = 135$, it is in normal region, where the complexity value $\mathrm{NL} = 24$ larger than T_1 but less than T_2. The context pixels $C = \{137, 136, 139, 138, 140, 135, 142, 137, 136, 139\}$ are chosen and prediction-error is $p = 0$. One bit data $b = 0$ can be embedded by remaining the pixel value $\hat{x} = x = 135$. If context pixels of C_1 are chosen, the prediction-error wound be $p = 1$ and no one bit data wound be embedded. The Third pixel $x = 147$ is in the rough region whose complexity $\mathrm{NL} = 34$ larger than T_2 and it is skipped.

And for the purpose of finding suitable thresholds, let $\mathrm{H}_{(T_1,T_2)}$ as the histogram of prediction-errors derived from the pixels in X_{T_1} with C_1 and pixels in $X_{T2} \setminus X_{T_2}$ with C_2. For a given embedding capacity EC, the optimal thresholds (T_1^*, T_2^*) are obtained by

$$\begin{array}{ll} \underset{0 \leq T_1 \leq T_2}{\arg\min} & \frac{\frac{1}{2}\mathrm{H}_{(T_1,T_2)}(0)+\sum_{i\geq 1}\mathrm{H}_{(T_1,T_2)}(i)}{\mathrm{H}_{(T_1,T_2)}(0)} \\ s.t. & \mathrm{H}_{(T_1,T_2)}(0) \geq EC \end{array} . \tag{14}$$

In order to better illustrate, Fig. 7 shows the optimal thresholds (T_1^*, T_2^*) by (14) for some capacities. It

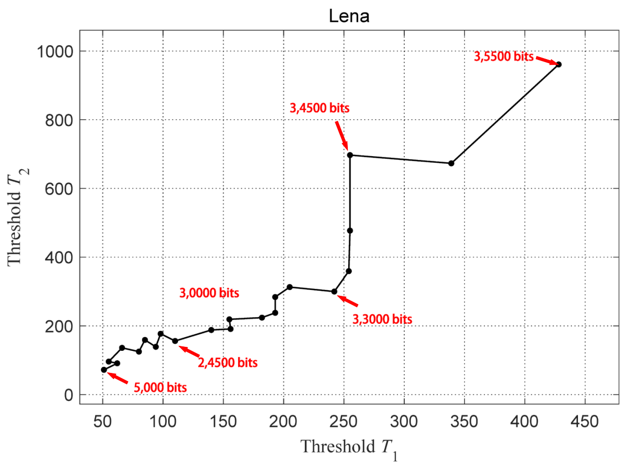


Figure 7: Thresholds.

is note that the two optimal thresholds are positively correlated. However, it is difficult to find the exact relationship between them. So, in the optimization process, we find the optimal parameters by traversing all the values. The pink curve of C_{12} in Fig. 3(a) shows the better result is achieved by combining two size context with two complexity thresholds.

3.3. Data Embedding and Extraction Procedures

The data embedding procedure is divided into the following steps to introduce as shown in Fig. 8. Firstly, a location map LM is constructed to solve overflow/underflow problem. For i-th pixel x_i, if $x_i = 0$ or $x_i = 255$, we assign $\mathrm{LM}(i) = 1$. Otherwise, we assign $\mathrm{LM}(i) = 0$. All pixels equals 255 are modified to 254 and 0 are modified to 1. Because LM records all the locations where the overflow/underflow may occur, original pixels can be recovered. Next, for the given several context sizes $\{C_1, ..., C_N\}$ (N is the number of context sizes), the complexity $\mathrm{NL}(i)$ of current pixel value x_i is calculated the sum of the absolution differences of the horizontal and vertical of every two adjacent pixels belong to the largest size context C_N. And context pixels to predict x_i are selected by comparing $\mathrm{NL}(i)$ with the given thresholds $\{T_1, ..., T_N\}(0 \leq T_1 < T_2 < ... < T_N\ )$, which is similar to the procedure described in section 3.2. Then, x_i is predicted by the chosen context pixels by (12) and one bit data is embedded by modifying x_i to $\hat{x}_i$ by (13). The steps of prediction and embedding are performed pixel-by=pixel until all secret data is embedded and current pixel position is recorded as k_{end}. Finally, for reversibility, some auxiliary information should be used for blind decoding such as

- number of context sizes N (2 bits),
- the complexity thresholds $\{T_1, ..., T_N\}$ ($11 \times N$ bits),
- the pixel position where data embedding ends (18 bits),
- length of compressed location map L (18 bits),
- the compressed location map LM (L bits).

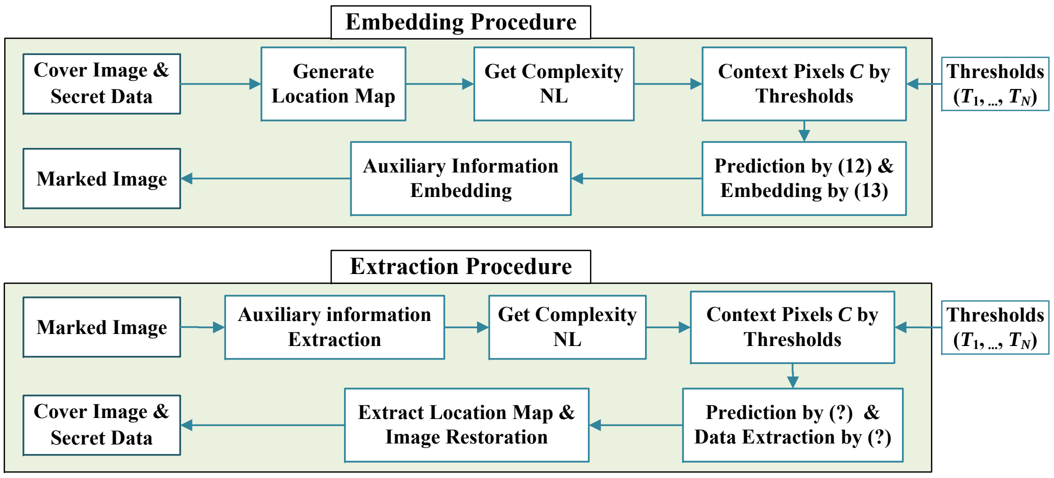

Embedding Procedure
Cover Image & Secret Data
Generate Location Map
Get Complexity NL
Context Pixels C by Thresholds
Thresholds (T1, ..., TN)
Prediction by (12) & Embedding by (13)
Auxiliary Information Embedding
Marked Image
Extraction Procedure
Marked Image
Auxiliary information Extraction
Get Complexity NL
Context Pixels C by Thresholds
Thresholds (T1, ..., TN)
Prediction by (?) & Data Extraction by (?)
Extract Location Map & Image Restoration
Cover Image & Secret Data

Figure 8: Procedures.

To embed the auxiliary information, LSB of first $38 + 11 \times N + L$ bits pixels of the image is recorded to obtain a binary vector V_{LSB} and then replaced by the auxiliary information. V_{LSB} is embedded into pixels after $x_{k_{\text{end}}}$ by the same method by (13) at last. During the whole embedding procedure, the threshold T_1 is selected from 1 and increased by 1 iteratively and j-th threshold T_j is selected from $T_{j-1} + 1$ and increased by 1 iteratively. The optimal thresholds are obtained by traversing all values of pixel complexities.

The extraction procedure is an opposite process as shown in Fig. 8. Firstly, auxiliary information is gotten by reading LSB of first $38 + 11 \times N + L$ bits pixels and replaced by LSB of pixels after $x_{k_{\text{end}}}$. Next, for marked pixel value $\hat{x}$, complexity is computed as the same as embedding procedure by extracting data pixel-by-pixel in reverse order. And, with extracted N complexity thresholds $\{T_1, ..., T_N\}$, the same context pixels C of $\hat{x}$ as original pixel value x can be obtained. Then, prediction-error $\hat{p}$ of the marked pixel value $\hat{x}$ is computed by

$$\hat{p} = \begin{cases} \hat{x} - \max(C), & \text{if } x \in S_1 \\ \min(C) - \hat{x}, & \text{if } x \in S_2 \cup S_3 \\ 0, & \text{if } x \in S_4 \\ \hat{x} - \max(C) - 1, & \text{if } x \in S_5 \\ \text{skip}, & \text{otherwise} \end{cases}. \tag{15}$$

The original pixel value x is recovered by

$$x = \begin{cases} \hat{x}, & \text{if } \hat{p} = 0 \\ \hat{x} - 1, & \text{if } \hat{x} \in S_1 \cup S_4 \cup S_5 \text{ and } \hat{p} \geq 1 \\ \hat{x} + 1, & \text{if } \hat{x} \in S_2 \cup S_3 \text{ and } \hat{p} \geq 1 \\ \text{skip}, & \text{otherwise} \end{cases}. \tag{16}$$

Meanwhile, the secret data b equals 1 if $\hat{p} = 1$ or 0 if $\hat{p} = 0$. Finally, original image is recovered by LM referring the embedding procedure. The flowchart of extraction procedure is shown in Fig. 8.

Table 1: Comparison of PSNR (dB) among the proposed method and the PVO-based methods of Peng *et al.* [1], Ou *et al.* [4] and Qu *et al.* [2]. The embedding capacity is 10,000 bits.

Images	Qu *et al.*	Peng *et al.*	Ou *et al.*	Proposed
Lena	60.23	60.49	60.59	**61.19**
Baboon	53.76	53.58	54.48	**54.29**
Airplane	63.79	62.97	63.29	**64.11**
Barbara	60.02	60.48	60.59	**60.60**
Lake	59.89	58.81	59.36	**60.47**
Boat	58.34	58.26	58.23	**58.69**
Peppers	58.75	58.97	59.18	**59.37**
Elaine	58.24	57.37	57.37	**59.61**
Average	59.13	58.86	59.13	**59.79**

Table 2: Comparison of PSNR (dB) among the proposed method and the PVO-based methods of Peng *et al.* [1], Ou *et al.* [4] and Qu *et al.* [2]. The embedding capacity is 20,000 bits.

Images	Qu *et al.*	Peng *et al.*	Ou *et al.*	Proposed
Lena	56.61	56.56	56.58	**57.24**
Airplane	59.99	59.07	59.33	**60.39**
Barbara	56.26	56.20	56.50	**56.74**
Lake	54.74	53.53	54.29	**55.32**
Boat	54.14	53.83	53.76	**54.24**
Peppers	55.02	54.77	54.93	**55.37**
Elaine	53.55	52.65	52.71	**54.30**
Average	55.76	55.23	55.44	**56.23**

4. Experimental Results

In this section, experimental results of proposed method are presented. The performance is compared with three typical PVO-based methods of Peng *et al.* [1], Ou *et al.* [4] and Qu *et al.* [2]. The proposed method is implemented on MatLab version 2018a on a tower server (SUPERMICRO LT-7038AX). The experimental results are evaluated on eight standard gray-scale 512 $\times$ 512-sized images including Lena, Baboon, Airplane, Barabra, Lake, Boat, Peppers and Eliane.

Firstly, we consider the impact of number of context sizes N to the embedding capacity. As shown in Fig. 9, for each given capacity, N is from 2 to 4, corresponding the sizes $C_1 \in \{C_1\}$, $C_{12} \in \{C_1, C_2\}$, $C_{123} \in \{C_1, C_2, C_3\}$ and $C_{1234} \in \{C_1, C_2, C_3, C_4\}$, respectively. It shows that although the embedding performance increases with the increase of N, it is already very small when $N = 4$. Therefore, balancing the embedding performance and embedding time complexity, we set $N = 4$ as a suitable size number. There are four complexity thresholds $\{T_1, T_2, T_3, T_4\}$ to context pixels for the current pixel, and The smoother the area where the pixel is, the less context pixels are used to predict. For a given embedding capacity, embedding procedure can be implemented less than four seconds.

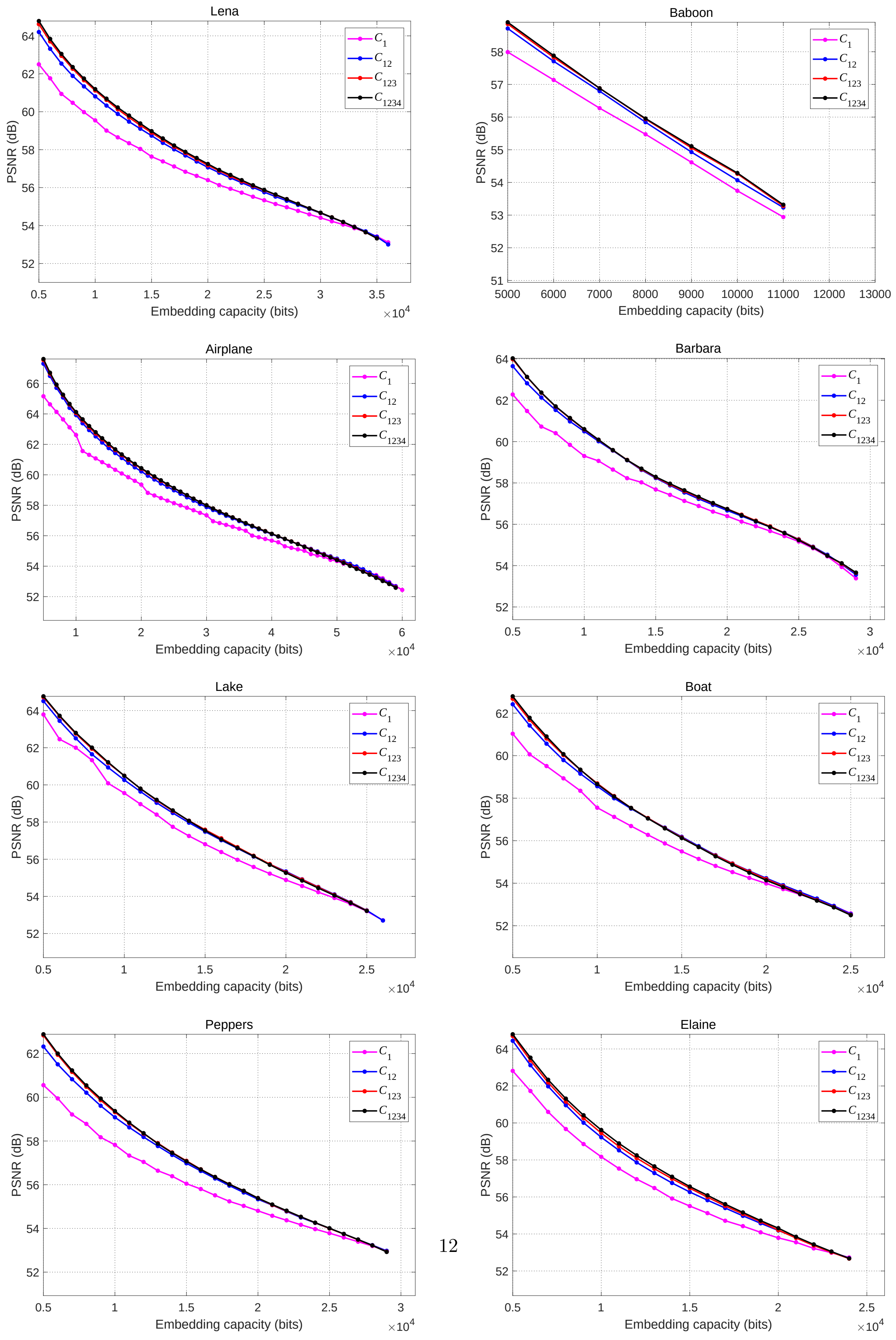


Figure 9: size.

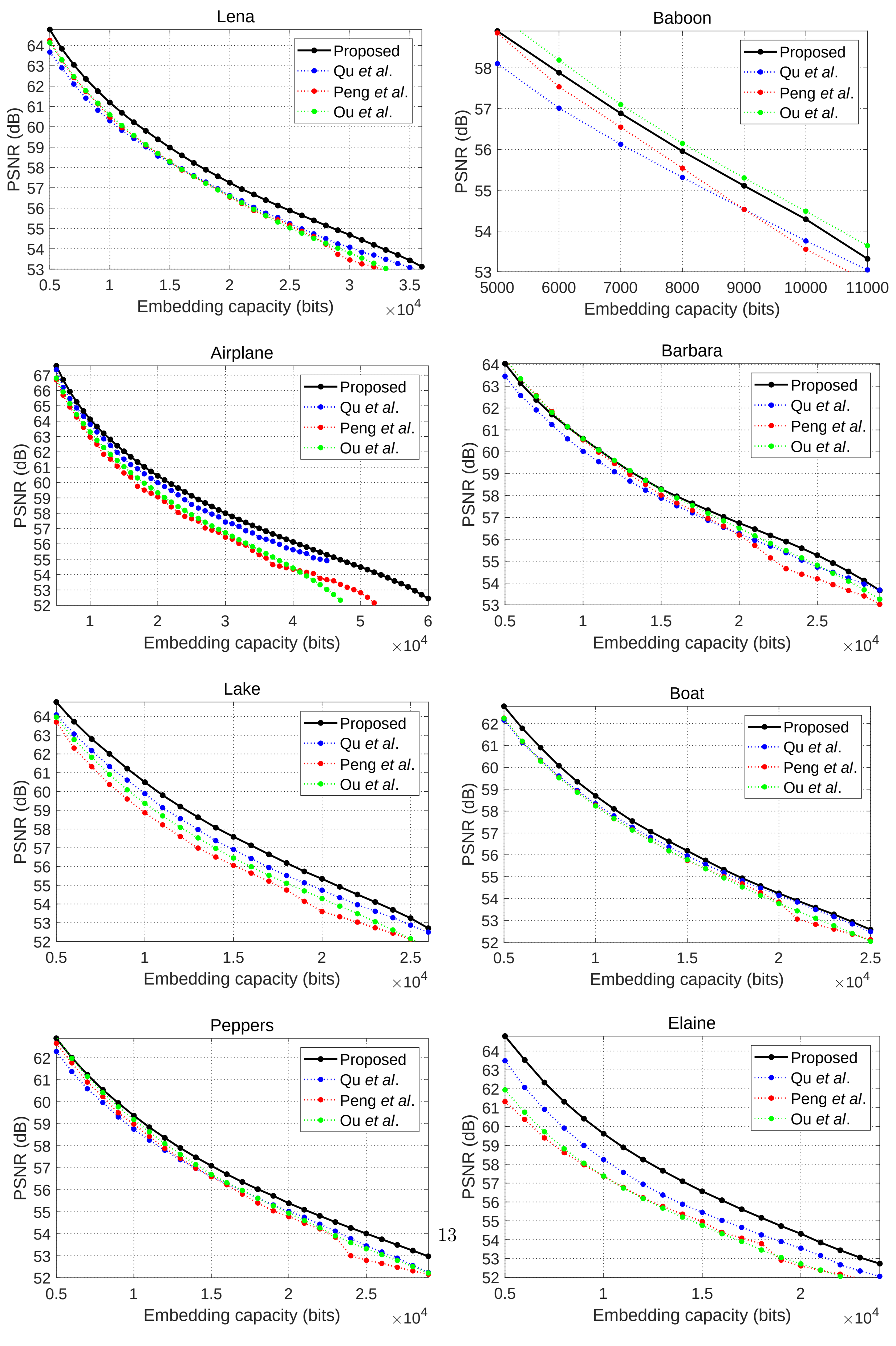


Figure 10: capacity.

5. Conclusion

Acknowledgement

References

[1] F. Peng, X. Li, B. Yang, Improved pvo-based reversible data hiding, Digit. Signal Process. 25 (2) (2014) 255–265.

[2] X. Qu, H. J. Kim, Pixel-based pixel value ordering predictor for high-fidelity reversible data hiding, Signal Process. 111 (2015) 249–260.

[3] 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.

[4] B. Ou, X. Li, Y. Zhao, R. Ni, Reversible data hiding using invariant pixel-value-ordering and prediction-error expansion, Signal Process.: Image Commun. 29 (7) (2014) 760–772.