Article

Efficient Reversible Data Hiding Based on Connected Component Construction and Prediction Error Adjustment

Limengnan Zhou [1,2], Chongfu Zhang [2], Asad Malik [3] and Hanzhou Wu [4,5,*]

1 School of Electronic and Information Engineering, University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan 528402, China
2 School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
3 Department of Computer Science, Aligarh Muslim University, Aligarh 202002, India
4 School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China
5 Guangdong Provincial Key Laboratory of Information Security Technology, Guangzhou 510006, China
* Correspondence: hanzhou@shu.edu.cn

Abstract: To achieve a good trade-off between the data-embedding payload and the data-embedding distortion, mainstream reversible data hiding (RDH) algorithms perform data embedding on a well-built prediction error histogram. This requires us to design a good predictor to determine the prediction errors of cover elements and find a good strategy to construct an ordered prediction error sequence to be embedded. However, many existing RDH algorithms use a fixed predictor throughout the prediction process, which does not take into account the statistical characteristics of local context. Moreover, during the construction of the prediction error sequence, these algorithms ignore the fact that adjacent cover elements may have the identical priority of data embedding. As a result, there is still room for improving the payload-distortion performance. Motivated by this insight, in this article, we propose a new content prediction and selection strategy for efficient RDH in digital images to provide better payload-distortion performance. The core idea is to construct multiple connected components for a given cover image so that the prediction errors of the cover pixels within a connected component are close to each other. Accordingly, the most suitable connected components can be preferentially used for data embedding. Moreover, the prediction errors of the cover pixels are adaptively adjusted according to their local context, allowing a relatively sharp prediction error histogram to be constructed. Experimental results validate that the proposed method is significantly superior to some advanced works regarding payload-distortion performance, demonstrating the practicality of our method.

Keywords: reversible watermarking; reversible data hiding; graph optimization; prediction error

MSC: 94A08

check for updates

Citation: Zhou, L.; Zhang, C.; Malik, A.; Wu, H. Efficient Reversible Data Hiding Based on Connected Component Construction and Prediction Error Adjustment. *Mathematics* **2022**, *10*, 2804. https://doi.org/10.3390/math10152804

Academic Editors: Ximeng Liu, Yinbin Miao and Zuobin Ying

Received: 11 July 2022
Accepted: 4 August 2022
Published: 7 August 2022

1. Introduction

As an interdisciplinary research field, information hiding (IH) is generally modeled as a covert communication problem involving three participants: *Alice*, *Bob* and *Eve* [1]. Alice plays the role of the data sender. Bob plays the role of the data receiver. However, Eve serves as the attacker. Given a digital media object such as digital video and image, Alice first embeds a secret message into the digital media object (also called *cover*) by modifying the content of the cover without introducing noticeable artifacts. The resulting object concealing the secret message will be sent to Bob via an insecure channel that will be monitored by Eve. Eve will attempt to detect the existence of the secret message within the conveyed object or alter the conveyed object to remove the possibly embedded information. Once Bob receives the probably altered object containing hidden information, he will try to extract the secret message from the received object. Thus, IH is successfully realized if the secret message can be extracted without error. Otherwise, it is deemed failed. Compared with cryptography that may leave noticeable marks on the encrypted data, IH even conceals the existence of the present communication activity, which has good potential in applications and will become increasingly important in modern information security [2,3].

For IH, one of the most important requirements is that the difference between the cover media and the media containing secret information caused by the data embedding operation should be low so that any adversary will not notice the existence of the secret message within the embedded object [4]. Along this line, many IH algorithms have been introduced in the past two decades [5–8]. Given a secret payload to be embedded, most IH algorithms either embed the secret payload by minimizing a well-designed distortion function or embed the secret payload by preserving the selective model of the cover source [9]. On the other hand, for a pre-specified distortion, we expect to embed as many secret bits as possible. This is typically referred to as the payload-distortion problem. Nevertheless, these algorithms inevitably distort the original cover content. In other words, though the secret message can be successfully extracted without any error, the original cover media cannot be perfectly rebuilt from the object containing the secret message, which is not applicable to sensitive application scenarios that require no degradation of the original cover object [10]. This has motivated researchers to study reversible information hiding (RIH) [11] or say reversible data hiding (RDH) [12], reversible watermarking [13] to ensure that both the secret message and the original cover media can be reconstructed at the receiver side. Many RDH algorithms can be found in the literature such as [13–20].

RDH can be applied to various cover sources. For example, due to the popularity over social networks and the ease of handling, digital imagery is still one of the most popular sources for RDH. Other cover sources such as video sequences [21], speech signals [22] and texts [23] are also of increasing interest to researchers recently due to the fast development of multimedia technologies and social networking services. From the viewpoint of the data embedding mechanism, most advanced RDH algorithms use the so-called prediction errors (PEs) of the elements in the cover to be embedded to realize RDH. There are two main reasons [14]. First, the PEs are noise-like, meaning that modifying them will not produce obvious artifacts of the cover since the prediction process significantly reduces the impact caused by the cover content. Second, the PEs are collected to construct a prediction error histogram (PEH), which is a pooled vector that can be easily handled for RDH based on histogram shifting (HS) [12] or its variants [13,24–27]. In terms of data embedding positions, mainstream RDH works could be roughly divided into spatial domain and transform domain. The former uses the values of spatial pixels to carry secret bits, whereas the latter often uses the transformed values as the cover elements such as discrete cosine transform (DCT) coefficients to carry secret bits. In brief summary, regardless of the data embedding domain, exploiting PEs for RDH is efficient.

From the viewpoint of system design, many existing PE-based works mainly consist of five steps, i.e., *content prediction*, *content selection*, *data embedding*, *data extraction* and *cover reconstruction* [14]. Taking a gray-scale image as the cover for example, content prediction aims to predict the pixels and obtain the PEs. For content selection, its target is to sort the collected PEs by a local complexity function so that smooth pixels are embedded preferentially since smooth pixels have a small PE that can result in superior payload-distortion performance. Thereafter, the aforementioned HS or its variants can be applied to the sorted PE sequence to embed secret bits. Once the secret bits are embedded, the resulting new image (or *marked image*) will be sent to the receiver, who will perform secret data extraction and cover image reconstruction. Extracting the embedded secret data and recovering the original image can be roughly viewed as an inverse process of the data hider. Therefore, it is straightforward to draw out that, in order to provide superior payload-distortion performance, the content prediction, content selection and data embedding procedures can be optimized. For content prediction, it is required for us to design a predictor that can accurately estimate the pixels to be embedded. Since it is hard to model all natural images, even a well-designed predictor cannot predict all pixels accurately, e.g., many existing RDH algorithms use a fixed predictor for content prediction, which does not take into account the statistical characteristics of local context. Therefore, a trade-off strategy is to further select the pixels with a small PE out for data embedding preferentially, which is referred to as content selection. However, during the process of constructing an ordered sequence of PEs, many existing algorithms ignore the fact that adjacent PEs may have the identical priority of data embedding. As a result, there is still room for improving the payload-distortion performance. In addition, for data embedding, once the operation is pre-specified, we should further optimize the data embedding parameters so that the distortion will be low for a given payload.

Motivated by the above analysis, in this article, we are to study the optimization of content prediction and content selection so that the payload-distortion performance can be further improved. Meanwhile, since we use the PEs of image pixels to carry secret data, the optimized HS operation is used. In the proposed work, the pixels are adaptively predicted according to their local context, leading to a sharp prediction error histogram to be embedded. Furthermore, the main idea of content selection is to construct a graph containing multiple connected components for the cover image so that the PEs of the pixels within a connected component are close to each other. Accordingly, the sorted connected components can be used for carrying additional information. Experimental results have demonstrated that the proposed method outperforms a part of advanced works, displaying superiority and applicability of the proposed method.

The remainder of this article is organized as follows. First, we detail the proposed method in Section 2. Experiments and analysis are then provided in Section 3. Finally, we conclude this work and provide discussion in Section 4.

2. Proposed Method

In this section, we first describe the general framework of the proposed method. Then, we will introduce each important part in detail. Before a detailed introduction, we list all the important symbols used in this section and their meaning in Table 1.

Table 1. Important symbols and their meaning.

Symbol	Meaning
$\mathbf{x}$	cover image (gray-scale)
$\mathbf{m}$	secret message
$\mathbf{k}$	secret key
$\mathbf{y}$	marked image
h	the height of the cover image
w	the width of the cover image
$x_{i,j} \in \mathbf{x}$	the pixel at position (i, j) whose value is $x_{i,j}$
D	the adjacent set
$\hat{x}_{i,j}$	the prediction value of $x_{i,j} \in \mathbf{x}$
$e_{i,j}$	the prediction error of $x_{i,j} \in \mathbf{x}$
$G(V, E)$	a non-directed graph whose node-set is V and edge-set is E
T_d	an integer threshold

2.1. General Framework

Throughout this paper, gray-scale images are used to act as the cover image. Figure 1 shows the general technical framework for the proposed algorithm. We describe it as follows. We first pre-process the given cover image adjusting all pixel values into the usable range to avoid the pixel overflow and pixel underflow problem during pixel modification. The modified pixels should be recorded to construct a so-called location map which will be self-embedded into the cover image together with the secret data. Moreover, we self-embed some embedding parameters into the cover image so that the receiver has the ability to extract secret data and recover the original image.

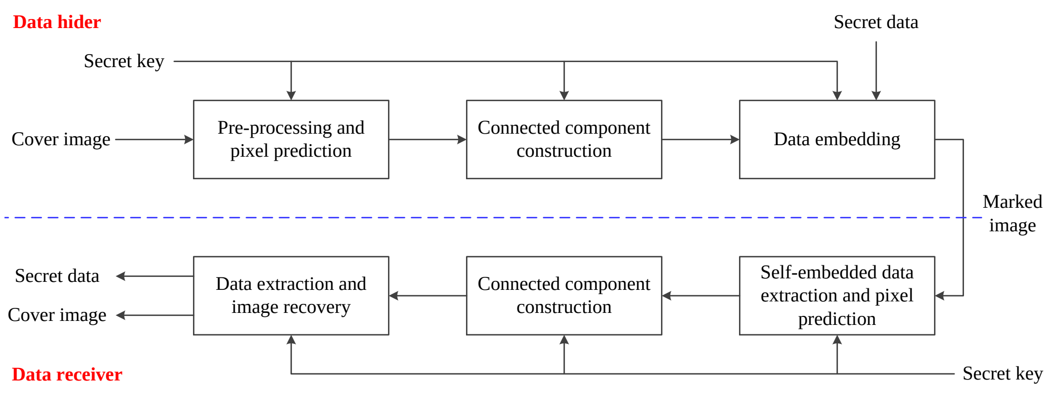
Data hider
Secret data
Secret key
Cover image
Pre-processing and pixel prediction
Connected component construction
Data embedding
Marked image
Secret data
Cover image
Data extraction and image recovery
Connected component construction
Self-embedded data extraction and pixel prediction
Data receiver
Secret key

Figure 1. General framework for the proposed method.

After pre-processing, we predict the cover pixels to be embedded based on the local context, by which we can generate a set of PEs that will be used for carrying secret bits. In order to provide good payload-distortion performance, we construct a graph, whose nodes correspond to the pixels to be embedded and edges represent the adjacent relationship between pixels. We sort the PEs by determining the connected components of the graph and a local complexity function. Therefore, by applying optimized HS, we embed the secret data into the sorted PEs. For the data receiver, they first extract the embedding parameters from the marked image and then perform an inverse process of the data hider to reconstruct the embedded information and the original cover content without error. In this way, efficient RDH can be realized. Below, we provide the details.

2.2. Pre-Processing and Pixel Prediction

Let $\mathbf{x}$ denote a gray-scale image whose size is denoted by $h \times w$, where each pixel $x_{i,j} \in \mathcal{I} = \{0, 1, \ldots, 255\}$. Our mission is to embed a secret binary stream $\mathbf{m} \in \{0, 1\}^l$ into $\mathbf{x}$ to generate a marked image $\mathbf{y} \in \mathcal{I}^{h \times w}$ such that the distortion between $\mathbf{y}$ and $\mathbf{x}$ is low. In addition, both $\mathbf{m}$ and $\mathbf{x}$ can be reconstructed from $\mathbf{y}$ without any error. Mathematically, we have

$$\mathbf{y} = \mathbf{Embed}(\mathbf{x}, \mathbf{m}, \mathbf{k}) \tag{1}$$

and

$$\mathbf{m}, \mathbf{x} = \mathbf{Extract}(\mathbf{y}, \mathbf{k}), \tag{2}$$

where $\mathbf{k}$ represents the secret key shared between the hider and the receiver.

The proposed method modifies the spatial pixels to embed secret data. In order to avoid the above-mentioned pixel underflow and pixel overflow problem, the values of boundary pixels to be embedded should be adjusted into the usable range. Since each pixel is modified by $\{-1, 0, +1\}$ during data embedding, a boundary pixel always has a value of 0 or 255. Therefore, we need to modify each pixel with a value of 0 or 255 as a pixel with a value of 1 or 254 to avoid the pixel underflow and pixel overflow problem. To ensure reversibility, the positions and original values of these boundary pixels are recorded to construct a location map which will be compressed by an efficient lossless compression technique as a binary stream to act as the side information that will be self-embedded into the cover image. This strategy has been applied by many RDH algorithms such as [13,16,17]. The losslessly compressed location map can be considered as a part of the secret message. Since the number of boundary pixels in a natural image is very small, the size of the losslessly compressed location map is also very small. This indicates that the impact of the losslessly compressed location map on the pure embedding payload can be ignored. In addition, the data embedding parameters should be self-embedded into the cover image in advance so that the data receiver is capable of extracting the secret data and reconstructing the original image. This can be achieved by using the least significant bits (LSBs) of some specified pixels to store the parameters. These specified pixels should be unchanged in the subsequent process. The original LSBs of these pixels should be recorded and embedded to ensure reversibility [14].

Given the cover image **x**, we divide the pixels to be embedded in **x** into two disjoint sets S_0 and S_1. The pixels in S_0 will be used to predict the pixels in S_1. The pixels in S_1 will be used for embedding secret data. Since the data embedding operation is reversible, once S_1 has been embedded, we can further use the (modified) pixels in S_1 to predict the pixels in S_0. Thus, the pixels in S_0 can be used thereafter for data embedding. Without the loss of generalization, in the following, we will use S_0 for pixel prediction and S_1 for data embedding unless otherwise specified. It is noted that when we use S_0 to predict S_1, S_0 should be unchanged during the process of embedding secret data into S_1, and vice versa. We are free to determine S_0 and S_1. In this paper, S_0 and S_1 are determined as:

$$S_b = \{x_{i,j} \in \mathbf{x} \mid (i+j) \bmod 2 = b\}. \tag{3}$$

Many existing RDH algorithms use a fixed predictor for pixel prediction, which does not take into account the difference between different local contexts and therefore may not accurately predict the pixels. In this paper, we propose a content-adaptive strategy for pixel prediction. Suppose that the pixels in S_0 are used to predict the pixels in S_1, for each pixel $x_{i,j} \in S_1$, as shown in Figure 2, we determine three candidate prediction values by:

$$\hat{x}_{i,j,0} = \left\lfloor \frac{x_{i,j-1} + x_{i,j+1}}{2} + 0.5 \right\rfloor, \tag{4}$$

$$\hat{x}_{i,j,1} = \left\lfloor \frac{x_{i-1,j} + x_{i+1,j}}{2} + 0.5 \right\rfloor, \tag{5}$$

$$\hat{x}_{i,j,2} = \left\lfloor \frac{x_{i,j-1} + x_{i-1,j} + x_{i,j+1} + x_{i+1,j}}{4} + 0.5 \right\rfloor, \tag{6}$$

where $\lfloor x \rfloor$ returns the largest integer that is no more than x. The operation of "+0.5" is to adjust the prediction value to the corresponding nearest integer. Taking Equation (4) for example, if $(x_{i,j-1} + x_{i,j+1})/2 = 10.7$, then $\hat{x}_{i,j,0}$ will be equal to 11, rather than 10. The final prediction value of $x_{i,j}$, denoted by $\hat{x}_{i,j}$, is selected from the candidate-set $P_{i,j} = \{\hat{x}_{i,j,0}, \hat{x}_{i,j,1}, \hat{x}_{i,j,2}\}$ according to the local context of $x_{i,j}$ which is shown in Figure 3. To this end, we determine a *priority* for each element in $P_{i,j}$. In detail, let $\alpha(\hat{x}_{i,j,k})$ denote the priority of $\hat{x}_{i,j,k}$. We determine $\alpha(\hat{x}_{i,j,k})$ by the following equation:

$$\alpha(\hat{x}_{i,j,k}) = \sum_{(dx,dy)\in D} \delta(|\hat{x}_{i,j,k} - \hat{x}_{i+dx,j+dy,k}|, \min_{r\in\{0,1,2\}} |\hat{x}_{i,j,r} - \hat{x}_{i+dx,j+dy,r}|), \tag{7}$$

where $D = \{(-1,-1), (-1,1), (1,-1), (1,1)\}$, $\delta(x,y) = 1$ if $x = y$, and 0 otherwise. The feasibility of Equation (7) relies on the fact that adjacent pixels in natural images tend to have similar statistical characteristics, which inspires us to use the difference between the prediction values of adjacent pixels to determine which pixels can be embedded preferentially. In this way, $\hat{x}_{i,j}$ can be finally determined by:

$$\hat{x}_{i,j} = \arg\max_{\hat{x}_{i,j,k}} \alpha(\hat{x}_{i,j,k}), \tag{8}$$

where $k \in \{0,1,2\}$. Thus, the PE of $x_{i,j}$ is determined by:

$$e_{i,j} = x_{i,j} - \hat{x}_{i,j}, \tag{9}$$

which will be used for carrying secret data in the subsequent process. It can be concluded that the final prediction value of a pixel is actually selected from multiple candidate values, resulting in that the final PE value of the pixel is essentially selected from multiple candidate PE values as well. It can be said that, as an effective PE adjustment strategy, the proposed method for determining PE is more applicable in practice compared with many existing methods that use a fixed predictor. This is why we term the proposed pixel prediction method as *prediction error adjustment*.

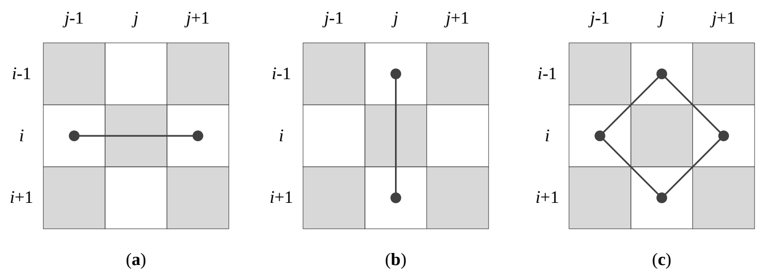

Figure 2. Three prediction modes for the pixel $x_{i,j} \in \mathbf{x}$ to be embedded: (**a**) Horizontal prediction, (**b**) Vertical prediction, (**c**) Four-direction prediction. For example, in (**a**), $x_{i,j}$ will be predicted with $x_{i,j-1}$ and $x_{i,j+1}$, and the prediction equation can be found in Equation (4).

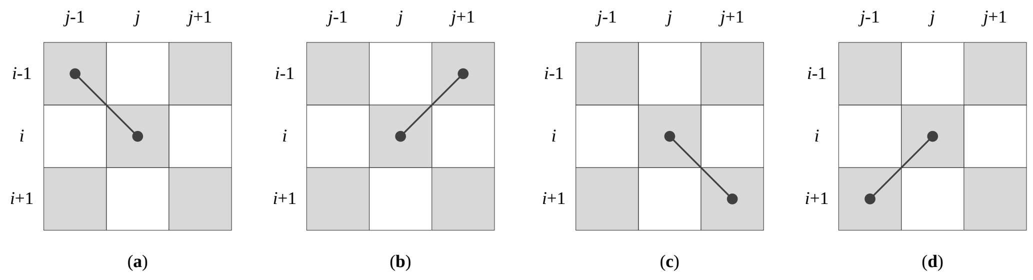

Figure 3. The local context for the pixel $x_{i,j} \in \mathbf{x}$ to be embedded: (**a**) $(dx, dy) = (-1, -1)$, (**b**) $(dx, dy) = (-1, 1)$, (**c**) $(dx, dy) = (1, 1)$, (**d**) $(dx, dy) = (1, -1)$. Namely, four pixels $x_{i-1,j-1}$, $x_{i-1,j+1}$, $x_{i+1,j+1}$ and $x_{i+1,j-1}$ constitute the local context of $x_{i,j}$.

2.3. Connected Component Construction

After pre-processing and pixel prediction, we need to generate an ordered PE sequence that will be used for data embedding. We propose a novel method to construct the PE sequence. Suppose that S_0 is used for pixel prediction and S_1 is used for data embedding, our goal is to sort the pixels in S_1 so that a pixel sequence and the corresponding PE sequence can be determined. We achieve this goal by applying connected component construction. Clearly, for any two different pixels $x_{i,j} \in S_1$ and $x_{i',j'} \in S_1$, they are adjacent to each other if we have

$$(i' - i, j' - j) \in D \text{ and } |\hat{x}_{i,j} - \hat{x}_{i',j'}| \leq T_d, \tag{10}$$

where T_d is a small integer threshold that needs to be pre-determined. If we model every pixel in S_1 as a graph node and the adjacent relationship between two pixels as a graph edge connecting the corresponding two graph nodes, we are able to construct a graph. Without loss of generality, let $G(V, E)$ be the constructed graph, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ denotes the node set and $E = \{(u_i, v_i) | u_i \in V, v_i \in V, u_i \neq v_i, 1 \leq i \leq |E|\}$ represents the edge set. Clearly, G is a non-directed graph, which means that two edges $(u, v) \in E$ and $(v, u) \in E$ are equivalent to each other. Figure 4 provides an example for constructing $G(V, E)$, from which it is easily inferred that G may contain multiple connected components. A connected component of G is defined as such a sub-graph $G'(V', E'), V' \subset V, E' \subset E$ that for any two different nodes $u' \in V'$ and $v' \in V'$, there is at least one *path* between u' and v'. The detailed pseudo-code to collect all the connected components of G is shown in Algorithm 1, which is based on the classical graph search technique called depth-first search (DFS). It can be inferred from Algorithm 1 that the computational complexity of collecting all the connected components given G is $O(|V| + |E|)$, which is very efficient. Clearly, the number of connected components and the number of nodes of a connected component are both affected by T_d. Specifically, a larger T_d

allows more pixels to be adjacent to each other in the graph, resulting in that the number of nodes in a connected component becomes larger, but the number of connected components becomes smaller. A smaller T_d makes the adjacent condition become more strict. As a result, the number of nodes in a connected component becomes smaller, but the number of connected components becomes larger.

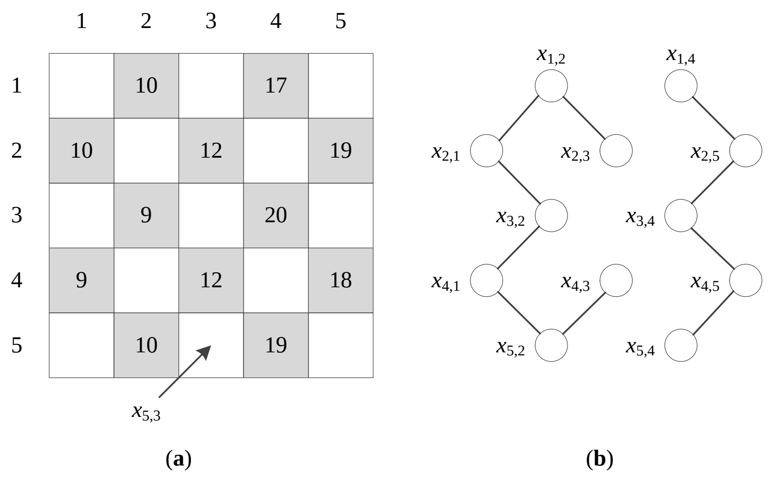

Figure 4. An example for constructing the graph: (**a**) Cover image, (**b**) Constructed graph. The pixels in the gray region are used for data embedding. The displayed values in the gray grids are the prediction values of the corresponding pixels. By setting $T_d = 2$, the corresponding graph is constructed as shown in (**b**).

Algorithm 1 The pseudo-code to collect all the connected components

```
Input: $G(V, E)$: a non-directed graph.
Output: $t$: the number of connected components, $G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_t(V_t, E_t)$: all
   the connected components.
 1: Initialize $t = 0$
 2: for each node $v \in V$ do
 3:    if $v$ has been previously processed then
 4:       Continue
 5:    end if
 6:    Set $t = t + 1$
 7:    Initialize $G_t(V_t, E_t)$ by $V_t = \{v\}$ and $E_t = \emptyset$
 8:    Call DFS($v$, $G(V, E)$, $G_t(V_t, E_t)$)
 9: end for
10: Sub-procedure DFS($v$, $G(V, E)$, $G_t(V_t, E_t)$)
11: Mark $v$ as processed
12: for each $(v, v') \in E$ do
13:    if $v'$ has been previously processed then
14:       Continue
15:    end if
16:    Update $V_t = V_t \cup \{v'\}$ and $E_t = E_t \cup \{(v, v')\}$
17:    DFS($v'$, $G(V, E)$, $G_t(V_t, E_t)$)
18: end for
19: End sub-procedure
20: return $t, G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_t(V_t, E_t)$
```

By setting a small T_d, the prediction values of most pixels in the same connected component will be close to each other. Based on this, it is reasonable to further assume that the original values of most pixels in the same connected component are close to each other. This indicates that the PEs of most pixels within a connected component are close to each other. Therefore, in order to generate an order PE sequence, it is natural to treat the pixels in the same connected component as equally important. For two different connected components, the one containing more nodes (i.e., pixels) has a higher priority for data embedding since the connected component containing more nodes is likely to have more smooth pixels which is more helpful for data embedding [13,14].

Based on the above analysis, we determine all the connected components of $G(V, E)$, denoted by $\{G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_t(V_t, E_t)\}$, where t is the total number of connected components. It is required that

$$V = \cup_{i=1}^{t} V_i \text{ and } E = \cup_{i=1}^{t} E_i \tag{11}$$

and

$$V_i \cap V_j = \emptyset \text{ and } E_i \cap E_j = \emptyset, \forall 1 \leq i < j \leq t. \tag{12}$$

We sort the connected components according to the number of nodes of a graph. To this end, we determine a permutation of $\{1, 2, \ldots, t\}$ as $\{r_1, r_2, \ldots, r_t\}$ so that

$$|V_{r_1}| \geq |V_{r_2}| \geq \cdots \geq |V_{r_t}|, \tag{13}$$

where the r_1-th connected component has the maximum number of nodes, whereas the r_t-th connected component has the minimum number of nodes. It is noted that $\{r_1, r_2, \ldots, r_t\}$ can be easily determined by sorting $\{|V_1|, |V_2|, \ldots, |V_t|\}$, whose computational complexity is $O(t \cdot \log_2 t)$. In order to generate the ordered PE sequence, we process each of the sorted connected components in an orderly manner. First of all, we initialize the PE sequence as an empty sequence. Then, for each $1 \leq i \leq t$, we sort all the pixels corresponding to $G_{r_i}(V_{r_i}, E_{r_i})$ according to the local complexity function defined in [13]. After sorting the pixels, we orderly append the corresponding PEs to the end of the above PE sequence. By processing all connected components, we can finally generate an ordered PE sequence. Clearly, during the construction of the PE sequence, for each PE in the sequence, we can easily identify the position of the corresponding pixel, which enables us to modify the corresponding pixel value in the subsequent data embedding procedure. It is worth mentioning that one may not use the local complexity function defined in [13] for sorting the PEs of a connected component. It is free to define other local complexity functions to order the PEs. In summary, sorting the PE sequence in this paper requires us to build a graph and find all connected components. Therefore, we term this process as *connected component construction*.

2.4. Data Embedding

We are now able to embed secret data into the sorted PE sequence by applying HS. Mathematically, we express the sorted PE sequence to be embedded as $\mathbf{e} = \{e_i\}_{i=1}^{n_e}$. Two pairs of peak-zero bins, denoted by (l_p, l_z) and (r_p, r_z), where $l_z < l_p < r_p < r_z$, are used to embed the secret data $\mathbf{m}$ into $\mathbf{e}$ by applying HS as mentioned above. Here, the peak bins l_p and r_p are used to carry secret bits. The bins in range $(l_z, l_p) \cup (r_p, r_z)$ will be shifted to ensure reversibility. The remaining bins will be unchanged. For a given bit $b \in \{0, 1\}$ to be embedded and a PE $e_i \in \mathbf{e}$, the PE carrying secret information $\hat{e}_i$ (also called marked PE) is determined by [28]:

$$\hat{e}_i = \begin{cases} e_i + b, & \text{if } e_i = r_p, \\ e_i - b, & \text{if } e_i = l_p, \\ e_i + 1, & \text{if } r_p < e_i < r_z, \\ e_i - 1, & \text{if } l_z < e_i < l_p, \\ e_i, & \text{otherwise.} \end{cases} \tag{14}$$

The sum of $\hat{e}_i$ and the prediction value of the corresponding pixel will be used as the final value of the marked pixel. We terminate the procedure of embedding secret bits when the secret data $\mathbf{m}$ is entirely embedded. In other words, there must be a PE position $t_s \leq n_e$ where all the PEs $\{e_i \mid i > t_s\}$ are unchanged. It is necessary to optimize the two pairs of peak-zero bins so that the distortion introduced by embedding $\mathbf{m}$ into $\{e_i\}_{i=1}^{t_s}$ can be kept low. To this end, we apply the optimized method introduced in [16] for determining the near-optimal (l_p, l_z) and (r_p, r_z). It is highlighted that one may exhaust all possible (l_p, l_z) and (r_p, r_z) and find the optimal solution, given sufficient computational resources. Nevertheless, suppose that we have found (l_p, l_z) and (r_p, r_z), as mentioned previously, we should self-embed (l_p, l_z) and (r_p, r_z), as the data embedding parameters, into the cover image so that the data receiver can extract them before extracting secret data and recovering the cover image. In addition, the parameter T_d in Equation (10) should be self-embedded as well.

2.5. Data Extraction and Image Recovery

By extracting the data embedding parameters from the marked image, the data receiver can successfully extract secret data from the marked image and meanwhile recover the original cover image. First of all, the receiver performs pixel prediction and connected component construction in the same way as the data hider, by which a sorted PE sequence carrying the secret information can be obtained. Then, with the data embedding parameters, the secret data can be fully extracted by processing the marked PEs in an orderly manner. In this way, the original secret information and the side information can be retrieved. With the side information, the cover image can be reconstructed without error since the embedding operation is reversible.

2.6. Effectiveness and Complexity Analysis

The technical motivation behind many existing RDH algorithms is that embedding secret bits into smoother pixels will result in better payload-distortion performance. This is based on the fact that smoother pixels are likely to be predicted with a higher prediction accuracy based on their local context. As a result, the prediction error histogram follows a Gaussian-like distribution centered at zero, which is very helpful for data embedding. In this paper, instead of directly exploiting smoother pixels for data embedding, we propose a connected-component-based method to collect the pixels with close PEs for data embedding preferentially. Though the PE values of the pixels in a connected component may not be closer to zero compared with many existing works, the resultant prediction error histogram is still Gaussian-like distributed, meaning that by optimizing the embedding parameters, superior payload-distortion performance can be achieved. In other words, the proposed method has better applicability and generalization ability.

The main contributions of the proposed method include two aspects. One is optimization of the pixel predictor. Unlike many existing methods which use a fixed pixel predictor, the proposed method predicts a pixel in such a way that the final prediction value of a pixel is adaptively adjusted to the most suitable value according to the local context. As a result, the prediction can be more accurate. Since only three prediction modes are used in the proposed method and the local context of a pixel only consists of four neighboring pixels, the complexity to determine the final prediction value for a single pixel is very low. In other words, the overall complexity to determine all the prediction values is linearly proportional to the number of pixels to be predicted, which is very suitable for practice. On the other hand, as mentioned previously, given the graph $G(V, E)$, the procedure of constructing all the connected components requires a complexity of $O(|V| + |E|)$, which is linearly proportional to the size of the node set and the size of the edge set. By using a small T_d, $O(|V| + |E|)$ can be reduced to $O(k \cdot |V|)$, where k is a small coefficient. In other words, the complexity to determine all the connected components is also linearly proportional to the number of pixels to be predicted. Therefore, based on the above analysis, it can be concluded that the time complexity of the proposed method is low.

3. Performance Evaluation and Analysis

In this section, we conduct experiments for evaluating the performance of the proposed method. To this end, we take six standard test images *Airplane*, *Lena*, *Tiffany*, *Peppers*, *Baboon*, and *Sailboat* shown in Figure 5 varying from smooth to complex for simulation. The test images are all sized at 512×512. Furthermore, all values of the pixels are in the range $[0, 255]$. As described in the previous section, the proposed method divides the cover pixels into two disjoint subsets S_0 and S_1. Both subsets can be used for data embedding. That is, after data embedding with S_1, S_0 can be used for data embedding as well. Therefore, given the secret data $\mathbf{m}$ (in the form of binary stream), we can use S_1 to carry $|\mathbf{m}|/2$ secret bits. The remaining secret bits can be embedded into S_0. This type of payload partition strategy has been used in existing methods [13,16].

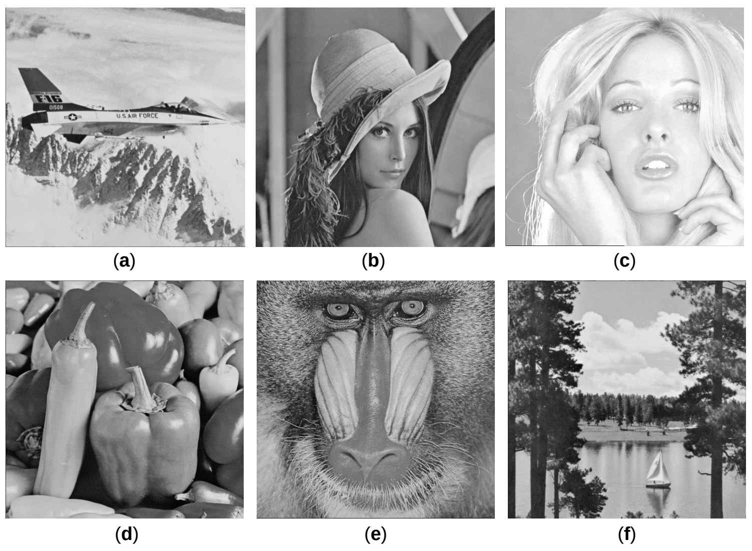

Figure 5. Six standard test images with a size of 512×512: (**a**) *Airplane*, (**b**) *Lena*, (**c**) *Tiffany*, (**d**) *Peppers*, (**e**) *Baboon*, and (**f**) *Sailboat*.

In order to demonstrate the superiority of the proposed method, we first show some visual examples for the proposed method. Figure 6 shows the marked images of the test images with an embedding rate of 10,000 bits and 20,000 bits. It can be seen that the proposed method does not introduce noticeable visual artifacts. The reason lies in that the proposed method either keeps the pixels unchanged or modifies the pixels by ± 1, which will not introduce significant distortion to the cover image and therefore provides very good visual quality of the marked images. To quantitatively evaluate the payload-distortion performance of the proposed method, we use peak signal-to-noise ratio (PSNR, in dB) to measure the visual quality of the marked image, i.e.,

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right), \tag{15}$$

$$\mathrm{MSE} = \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} |x_{i,j} - y_{i,j}|^2, \tag{16}$$

where $x_{i,j}$ and $y_{i,j}$ represent the original value and the marked value of the pixel at position (i, j). It can be seen that PSNR evaluates the difference between the original cover image and the corresponding marked image. A higher PSNR indicates that the difference between the original cover image and the corresponding marked image is lower, accordingly indicating that the marked image is visually better. When the size of the payload is specified, we expect to achieve as high a PSNR value as possible.

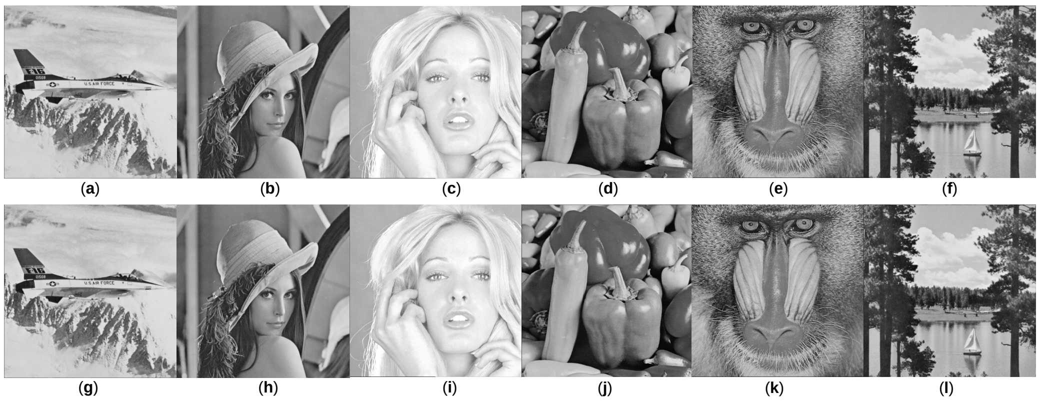

Figure 6. Examples for the marked images: (**a**) *Airplane* with 10,000 bits, (**b**) *Lena* with 10,000 bits, (**c**) *Tiffany* with 10,000 bits, (**d**) *Peppers* with 10,000 bits, (**e**) *Baboon* with 10,000 bits, (**f**) *Sailboat* with 10,000 bits, (**g**) *Airplane* with 20,000 bits, (**h**) *Lena* with 20,000 bits, (**i**) *Tiffany* with 20,000 bits, (**j**) *Peppers* with 20,000 bits, (**k**) *Baboon* with 20,000 bits, and (**l**) *Sailboat* with 20,000 bits.

We use a threshold T_d to control the number of connected components. Since the construction of the sorted PE sequence is dependent on the constructed connected components, we need to analyze the impact of the threshold T_d on the payload-distortion performance. To this end, we take two representative test images *Lena* (smooth image) and *Baboon* (complex image) for necessary analysis. Figure 7 shows the payload-distortion performance for the image *Lena* and the image *Baboon* due to different T_d. The abscissa represents the embedding rate, i.e., the size of the embedded payload, in bits. The ordinate represents the PSNR value. It can be inferred from Figure 7 that different images have different payload-distortion performance and different T_d also result in different payload-distortion performance. For the smooth image *Lena*, a smaller T_d is superior to a larger T_d at a low embedding rate, which is due to the reason that a smoother image enables us to collect more smooth pixels for data embedding by setting a small T_d. In contrast, for the complex image *Baboon*, a relatively smaller T_d is not a good choice for data embedding since the difference between adjacent pixels in *Baboon* is significantly larger than that in *Lena*. As a result, using a relatively larger T_d is a better strategy at a low embedding rate so that the number of relatively smooth pixels in a connected component can be increased, thereby benefiting data embedding. From the overall trend, as the embedding rate increases, the performance difference due to different T_d can be controlled within a small range. This indicates that, from the viewpoint of payload-distortion optimization, we can optimize T_d in a small range so that the payload-distortion performance is superior and the computational complexity is acceptable for practice. To this end, in the following experiments, for all test images, we limit T_d to the range $[0, 8)$ and use the integer resulting in the highest PSNR (for a fixed embedding rate) as the final value of T_d. It is worth mentioning that one may use a very large T_d for RDH. However, this will reduce the performance, as shown in Figure 8. The reason is that, a larger T_d will relax the constraint on the adjacent relationship between pixels, which can be easily inferred from Equation (10). For example, in the extreme case that $T_d = 256$, all the pixels to be embedded will be in the same connected component. In other words, there is only one connected component for $G(V, E)$ and the only one connected component is the graph itself. In this case, the proposed method degenerates to the traditional method. Figure 8 further shows the payload-distortion performance for a large T_d, from which we can infer that it is desirable to use a small T_d. This is why we optimize T_d in range $[0, 8)$ in this paper, which not only provides good payload-distortion performance but also keeps the computational complexity low.

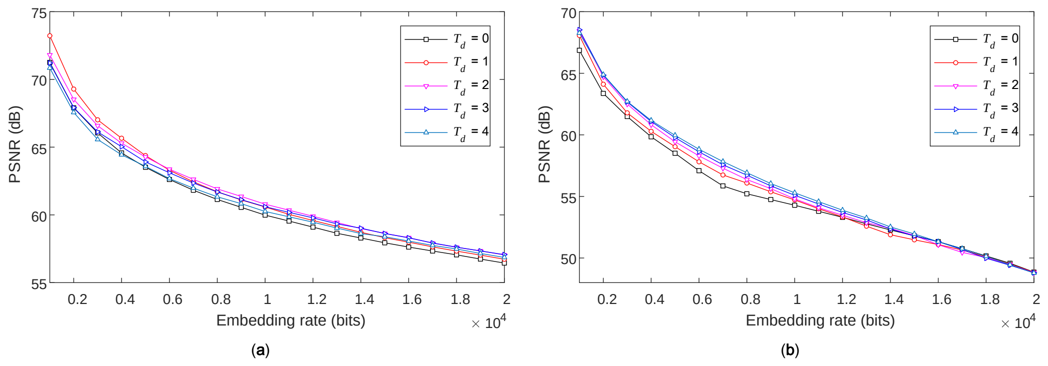

Figure 7. The payload-distortion performance due to small T_d for the images *Lena* and *Baboon*: (**a**) *Lena*, (**b**) *Baboon*.

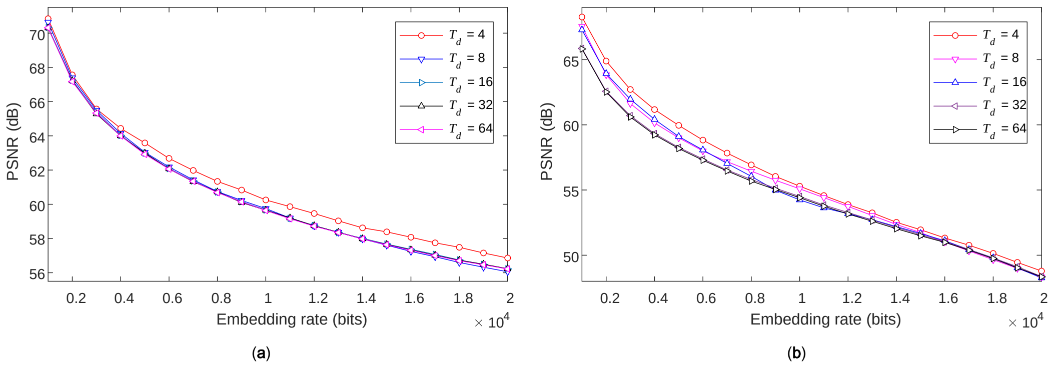

Figure 8. The payload-distortion performance due to large T_d for the images *Lena* and *Baboon*: (**a**) *Lena*, (**b**) *Baboon*.

The above analysis shows that the proposed method has the potential to provide superior payload-distortion performance. To evaluate the payload-distortion performance of the proposed method, we compare the proposed method with some advanced PE-based RDH methods, i.e., DSP [14], PPE [14], GP [29], SP [13], MPE [15] and BFS [28]. Both DSP and PPE are introduced in [14]. It is fair to compare the proposed method with these related works since they all focus on either improving the prediction accuracy or improving the pixel sorting procedure. For comparison, in our experiments, we use a randomly generated binary stream to represent the secret data to be embedded. Figure 9 shows the payload-distortion performance for the proposed method and the related works. It can be seen that different images have different payload-distortion performance, which is reasonable because different images have different statistical characteristics. As the embedding rate increases, the PSNR will decline, which is also reasonable since a larger embedding rate means that more modifications to the pixels will be performed, thereby resulting in a larger distortion between the marked image and the original image. It can be inferred from Figure 9 that the proposed method significantly outperforms the related works in terms of payload-distortion performance for all test images. We explain the reason as follows. DSP [14] uses a dynamic predictor for pixel prediction, which is efficient for smooth images but not suitable for complex images. Moreover, since the prediction mode of one pixel is affected by the prediction mode of the adjacent pixels, the prediction procedure for DSP is time-consuming. PPE [14] exploits the PE of a PE for data embedding, which is efficient for complex images. However, since the predictor is fixed for all pixels to be embedded, there is large room for further performance improvement. GP [29] uses a gradient-based prediction strategy for pixel prediction, which is efficient for RDH. However, processing the pixels from top to bottom and from left to right may result in many pixels not carrying secret bits being modified, accordingly introducing a high distortion of the marked image. Though SP [13] sorts the pixels to be embedded according to a well-designed local complexity function, the used data embedding parameters are not optimal for low embedding rates. Thus, it will introduce large distortion at low embedding rates. Similarly, for MPE [15], the data embedding parameters need to be optimized so as to provide better payload-distortion performance. In BFS [28], the prediction value of a pixel is determined as that of the adjacent pixel, which is not suitable for complex images. Moreover, the BFS method is time-consuming since it needs to optimize many parameters. Therefore, based on our experiments and analysis, it is true that the proposed method achieves better trade-off between embedding payload and embedding distortion compared with related works.

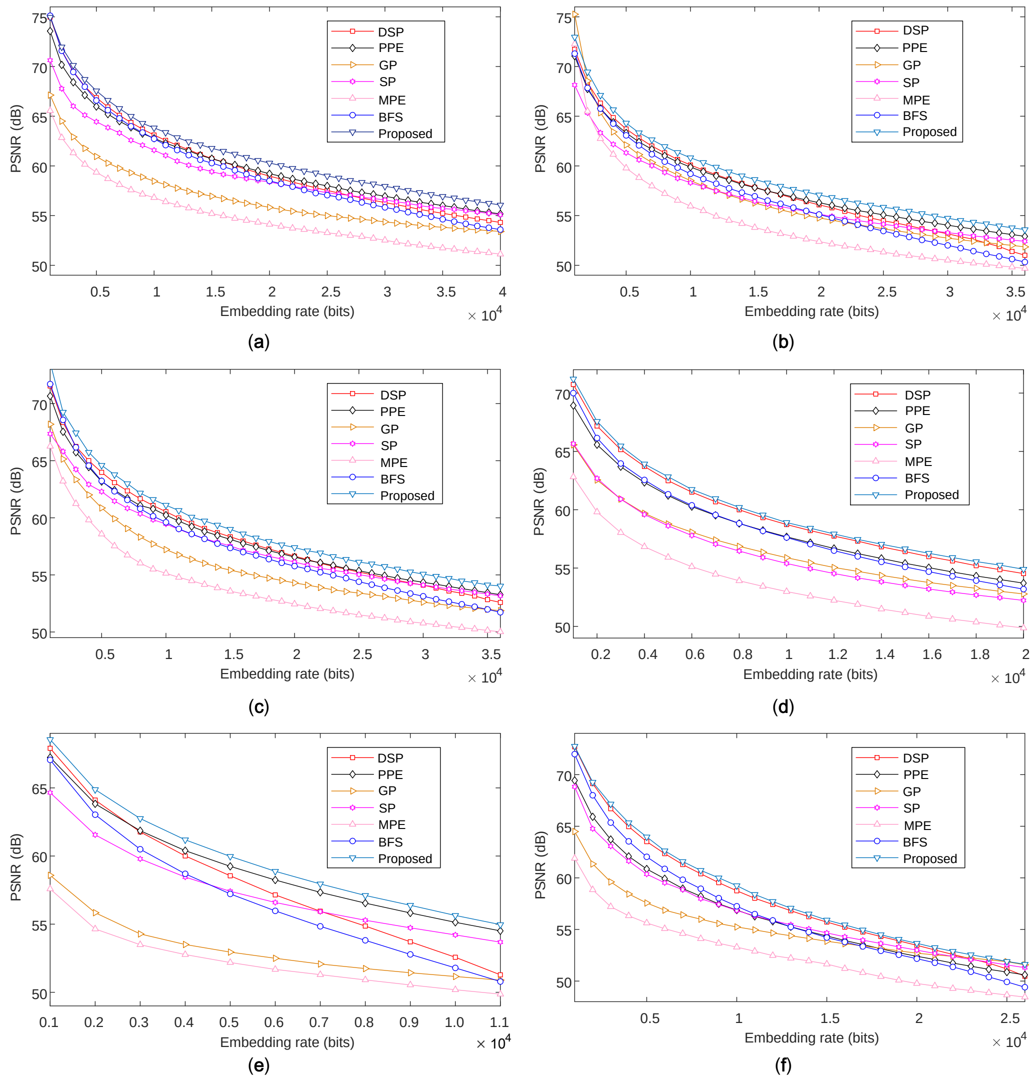

Figure 9. Performance comparison between the proposed method and the methods introduced by DSP [14], PPE [14], GP [29], SP [13], MPE [15] and BFS [28]: (**a**) *Airplane*, (**b**) *Lena*, (**c**) *Tiffany*, (**d**) *Peppers*, (**e**) *Baboon*, (**f**) *Sailboat*.

4. Conclusions

Improving the prediction accuracy of pixels and the embedding order of PEs represents a core problem in prediction-based RDH. This has motivated the authors in this paper to propose a novel strategy based on connected component construction and prediction error adjustment to further improve the prediction accuracy and optimize the embedding order of PEs. On the one hand, compared with previous works, a significantly different insight is that we model the cover pixels on a graph, where the nodes correspond to pixels and the edges represent the adjacent relationship between pixels. Since adjacent pixels have a strong correlation, the PEs of two pixels are close to each other when there is a path between the two pixels. Therefore, it is quite desirable to preferentially use pixels belonging to the same connected component (whose size is large enough) for data embedding. To this end, by determining the connected components of the graph, we propose to sort both the connected components of the graph and the pixels within a connected component for constructing the PE sequence. On the other hand, unlike mainstream methods that use a fixed predictor, we propose to adaptively determine the prediction value of a pixel based on its local context. As a result, the predictors for different pixels can be distinct from each other, which is more helpful for real-world scenarios. Experimental results demonstrate that the payload-distortion performance is significantly improved compared with some advanced methods that use PEs for data embedding. In the future, we will further optimize the proposed method through investigation on the prediction accuracy and the embedding order.

Author Contributions: Conceptualization, L.Z.; methodology, L.Z., A.M. and H.W.; software, L.Z. and H.W.; validation, C.Z. and A.M.; supervision, H.W.; project administration, C.Z. and H.W.; funding acquisition, L.Z., C.Z. and H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the National Natural Science Foundation of China (Grant Nos. 61901096 and 61902235), the Shanghai "Chen Guang" Project (Grant No. 19CG46), the National Key Research and Development Program of China (Grant No. 2018YFB1801302), the Science and Technology Foundation of Guangdong Province (Grant No. 2021A0101180005), the Opening Project of Guangdong Province Key Laboratory of Information Security Technology (Grant No. 2020B1212060078), and the CCF-Tencent Rhino-Bird Young Faculty Open Research Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Petitcolas, F.A.P.; Anderson, R.J.; Kuhn, M.G. Information hiding-a survey. *Proc. IEEE* **1999**, *87*, 1062–1078. [CrossRef]
2. Ke, Y.; Zhang, M.Q.; Liu, J.; Su, T.T.; Yang, X.Y. Fully Homomorphic Encryption Encapsulated Difference Expansion for Reversible Data Hiding in Encrypted Domain. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2353–2365. [CrossRef]
3. Faheem, Z.B.; Ali, M.; Raza, M.A.; Arslan, F.; Ali, J.; Masud, M.; Shorfuzzaman, M. Image Watermarking Scheme Using LSB and Image Gradient. *Appl. Sci.* **2022**, *12*, 4202. [CrossRef]
4. Qin, J.; Wang, J.; Tan, Y.; Huang, H.; Xiang, X.; He, Z. Coverless Image Steganography Based on Generative Adversarial Network. *Mathematics* **2020**, *8*, 1394. [CrossRef]
5. Atta, R.; Ghanbari, M. A high payload data hiding scheme based on dual tree complex wavelet transform. *Optik* **2021**, *226*, 165786. [CrossRef]
6. Chang, C.C.; Liu, Y.; Nguyen, T.S. A Novel Turtle Shell Based Scheme for Data Hiding. In Proceedings of the Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, Japan, 27–29 August 2014; pp. 89–93.
7. Sun, Y.; Lu, Y.; Chen, J.; Zhang, W.; Yan, X. Meaningful Secret Image Sharing Scheme with High Visual Quality Based on Natural Steganography. *Mathematics* **2020**, *8*, 1452. [CrossRef]
8. Chen, Y.; Wang, H.; Wu, H. Data hiding-based video error concealment method using compressed sensing. In Proceedings of the International Conference on Cloud Computing and Security, Nanjing, China, 16–18 June 2017; pp. 28–38.
9. Cox, I.; Miller, M.; Bloom, J.; Fridrich, J.; Kalker, T. *Digital Watermarking and Steganography*, 2nd ed; Morgan Kaufmann: Burlington, MA, USA, 2007.
10. Wu, H.; Shi, Y.; Wang, H.; Zhou, L. Separable reversible data hiding for encrypted palette images with color partitioning and flipping verification. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *27*, 1620–1631. [CrossRef]
11. Huang, C.-T.; Wang, W.-J.; Yang, C.-H.; Wang, S.-J. A scheme of reversible information hiding based on SMVQ. *Imaging Sci. J.* **2013**, *61*, 195–203. [CrossRef]
12. Ni, Z.; Shi, Y.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
13. Sachnev, V.; Kim, H.J.; Nam, J.; Suresh, S.; Shi, Y. Reversible watermarking algorithm using sorting and prediction. *IEEE Trans. Circuits Syst. Video Technol.* **2009**, *19*, 989–999. [CrossRef]
14. Zhou, L.; Han, H.; Wu, H. Generalized reversible data hiding with content-adaptive operation and fast histogram shifting optimization. *Entropy* **2021**, *23*, 917. [CrossRef] [PubMed]

15. Hong, W.; Chen, T.; Shiu, C. Reversible data hiding for high quality images using modification of prediction errors. *J. Syst. Softw.* **2009**, *82*, 1833–1842. [CrossRef]
16. Wu, H.; Wang, H.; Shi, Y. PPE-based reversible data hiding. In Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, Vigo Galicia, Spain, 20–22 June 2016; pp. 187–188.
17. Wu, H.; Wang, H.; Shi, Y. Dynamic content selection-and-prediction framework applied to reversible data hiding. In Proceedings of the IEEE International Workshop on Information Forensics and Security, Abu Dhabi, United Arab Emirates, 4–7 December 2016; pp. 1–6.
18. Chen, K.-M. High Capacity Reversible Data Hiding Based on the Compression of Pixel Differences. *Mathematics* **2020**, *8*, 1435. [CrossRef]
19. Kaur, G.; Singh, S.; Rani, R. PVO based reversible data hiding technique for roughly textured images. *Multidimens. Syst. Signal Process.* **2021**, *32*, 533–558. [CrossRef]
20. Fragoso-Navarro, E.; Cedillo-Hernandez, M.; Garcia-Ugalde, F.; Morelos-Zaragoza, R. Reversible Data Hiding with a New Local Contrast Enhancement Approach. *Mathematics* **2022**, *10*, 841. [CrossRef]
21. Chen, Y.; Wang, H.; Wu, H.; Liu, Y. Reversible video data hiding using zero QDCT coefficient-pairs. *Multimed. Tools Appl.* **2019**, *78*, 23097–23115. [CrossRef]
22. Nishimura, A. Reversible audio data hiding based on variable error-expansion of linear prediction for segmental audio and G. 711 speech. *IEICE Trans. Inf. Syst.* **2016**, *99*, 83–91. [CrossRef]
23. Zheng, X.; Fang, Y.; Wu, H. General framework for reversible data hiding in texts based on masked language modeling. *arXiv* **2022**, arXiv:2206.10112.
24. Tsai, P.; Hu, Y.-C.; Yeh, H.-L. Reversible Image Hiding Scheme Using Predictive Coding and Histogram Shifting. *Signal Process.* **2009**, *89*, 1129–1143. [CrossRef]
25. Coatrieux, G.; Pan, W.; Cuppens-Boulahia, N.; Cuppens, F.; Roux, C. Reversible Watermarking Based on Invariant Image Classification and Dynamic Histogram Shifting. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 111–120. [CrossRef]
26. Chen, H.; Ni, J.; Hong, W.; Chen, T.S. Reversible Data Hiding with Contrast Enhancement Using Adaptive Histogram Shifting and Pixel Value Ordering. Signal Process. *Image Commun.* **2016**, *46*, 1–16.
27. Ying, Q.; Qian, Z.; Zhang, X.; Ye, D. Reversible Data Hiding with Image Enhancement Using Histogram Shifting. *IEEE Access* **2019**, *7*, 46506–46521. [CrossRef]
28. Wu, H. Efficient reversible data hiding simultaneously exploiting adjacent pixels. *IEEE Access* **2020**, *8*, 119501–119510. [CrossRef]
29. Dragoi, I.-C.; Coltuc, D.; Caciula, I.; Gradient based prediction for reversible watermarking by difference expansion. In Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, Salzburg, Austria, 11–13 June 2014; pp. 35–40.