entropy

MDPI

Article

Generalized Reversible Data Hiding with Content-Adaptive Operation and Fast Histogram Shifting Optimization

Limengnan Zhou [1], Hongyu Han [2] and Hanzhou Wu [3,*]

[1] School of Electronic and Information Engineering, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan 528402, China; dreamzlmn@gmail.com
[2] School of Computer Science, Sichuan Normal University, Chengdu 610000, China; hongyuhanswjtu@163.com
[3] School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China
* Correspondence: h.wu.phd@ieee.org

Abstract: Reversible data hiding (RDH) has become a hot spot in recent years as it allows both the secret data and the raw host to be perfectly reconstructed, which is quite desirable in sensitive applications requiring no degradation of the host. A lot of RDH algorithms have been designed by a sophisticated empirical way. It is not easy to extend them to a general case, which, to a certain extent, may have limited their wide-range applicability. Therefore, it motivates us to revisit the conventional RDH algorithms and present a general framework of RDH in this paper. The proposed framework divides the system design of RDH at the data hider side into four important parts, i.e., binary-map generation, content prediction, content selection, and data embedding, so that the data hider can easily design and implement, as well as improve, an RDH system. For each part, we introduce content-adaptive techniques that can benefit the subsequent data-embedding procedure. We also analyze the relationships between these four parts and present different perspectives. In addition, we introduce a fast histogram shifting optimization (FastHiSO) algorithm for data embedding to keep the payload-distortion performance sufficient while reducing the computational complexity. Two RDH algorithms are presented to show the efficiency and applicability of the proposed framework. It is expected that the proposed framework can benefit the design of an RDH system, and the introduced techniques can be incorporated into the design of advanced RDH algorithms.

Keywords: reversible data hiding; watermarking; dynamic prediction; histogram shifting; optimization

Citation: Zhou, L.; Han, H.; Wu, H. Generalized Reversible Data Hiding with Content-Adaptive Operation and Fast Histogram Shifting Optimization. *Entropy* **2021**, *23*, 917. https://doi.org/10.3390/e23070917

Academic Editor: Raúl Alcaraz

Received: 23 June 2021
Accepted: 16 July 2021
Published: 19 July 2021

1. Introduction

Reversible data hiding (RDH) [1,2], also called reversible watermarking (RW), is referred to as the art of embedding extra data, such as source information and authentication data, into a host signal (also called cover) by slightly modifying the host signal. The embedded information and the original host signal can be fully reconstructed from the marked content by a legal receiver [3,4]. As RDH enables us to perfectly recover the original host content, it is quite desirable and helpful in some sensitive scenarios, such as medical image processing, remote sensing, and military communication.

Up to now, a number of RDH techniques have been reported in the literature. Early methods [5,6] mainly use lossless compression (LC) techniques to substitute a part of the host with the compressed code of the substituted part and the secret message. Since the LC procedure is often applied to the noise-like component of the host, the introduced distortion due to data embedding can be kept low. However, as the entropy of the noise-like component of the host is very high, the compression rate will be very low, which indicates that the (pure) embedding capacity of LC-based RDH is low. More efficient RDH methods are thereafter designed to increase the embedding capacity, e.g., difference expansion (DE) [1,7] and histogram shifting (HS) [2]. Since better payload-distortion performance can be achieved by exploiting the prediction-errors (or differences) of cover

elements, various RDH techniques [8–12] have been developed along this line. It can be said that, today, most RDH methods use prediction-errors (PEs) of cover elements to hide secret data since data embedding in PEs can always provide superior payload-distortion performance. Though these algorithms differ from each other in terms of the working mechanism, they are essentially finding the most compressible component of the host (i.e., the noise-like component with the minimum entropy) so that high capacity or low distortion can be achieved.

In PE-based RDH methods, there are two important steps: content prediction and data embedding. The former can be separated into two stages. First, some cover elements are selected out in advance. Then, the others are orderly predicted to generate a PE histogram (PEH). In the first stage, the pre-selected cover elements are usually unchanged throughout the content prediction, ensuring that both the data hider and the data receiver can find the identical prediction values. In the second stage, a suitable predictor is required to obtain accurate estimation of the cover elements to be embedded. The existing works [10,12] often use fixed content pre-selection rule and predictor, which, actually, is not desirable in applications according to the Kerckhoffs's principle since one may successfully reconstruct the marked PEH.

For data embedding, HS [2] is still the most common operation in today's RDH techniques. Variants, such as prediction-error expansion (PEE) [13], have also been developed. In HS, one hopes to select such PEH bins that the required payload can be carried by shifting the PEH bins while the distortion is as low as possible. The traditional methods often empirically tune the shifting parameters, which is not applicable in practice. And, as a host signal can be utilized to embed the secret data several times, it is actually time consuming for searching parameters and makes a it hard for a reader to reproduce the simulation results due to this non-deterministic empirical operation.

Therefore, on the one hand, it is quite desirable to design a general framework for RDH so that many existing RDH works can be generalized and more advanced RDH schemes (in terms of the payload-distortion performance and the security) can be developed based on the designed framework. On the other hand, by optimizing the data embedding parameters in a deterministic and fast manner, the RDH systems will be more applicable to practice. In order to fill this research gap, in this paper, we revisit the conventional RDH algorithms and propose a general framework of RDH. Meanwhile, we propose a fast and efficient parameter optimization algorithm for HS-based embedding. Two RDH methods based on the proposed framework are further introduced to demonstrate the superiority and applicability of the proposed work. In summary, the main contributions of this paper can be described as follows:

- We propose a general framework dividing the PE-based RDH design into four parts so that we can easily design or improve an RDH system. The four parts are named as binary-map generation, content prediction, content selection, and data embedding. To ensure the security, we use a secret key to control the binary-map generation, and a dynamic predictor for content prediction. For content selection, we use a local-complexity evaluation function to preferentially use smooth elements.
- We propose a fast histogram shifting optimization algorithm to determine the near-optimal embedding parameters for HS-based RDH. A significant advantage is that the embedding performance can be kept sufficient, while the computational cost is low.
- We present two detailed RDH methods to demonstrate the generalization ability of the proposed framework. Extensive experiments are also conducted to verify the superiority and applicability of the proposed work.

The rest of this paper is organized as follows. The proposed framework is first introduced in Section 2. Then, we study each part of the introduced framework in Section 3. Thereafter, in Section 4, two RDH algorithms and experimental results are presented to demonstrate the efficiency and applicability of the proposed work. Finally, we conclude this paper in Section 5. This work extends [14,15] to a general case.

2. Sketch of Proposed Framework

As shown in Figure 1, the proposed framework (at the data hider sider) has four parts: binary-map generation, content prediction, content selection, and data embedding. The binary-map generation produces a binary matrix with the same size of the cover, in which "0"s are elements used for content prediction or storing parameters, and "1"s are elements to be probably embedded. The proposed framework does not specify the type of the cover. It indicates that both the binary matrix and the cover elements could be arbitrary dimensional. Unless mentioned, we use a grayscale image as the cover, i.e., one may imagine that the binary matrix is 2D, and the elements are corresponding to image pixels.

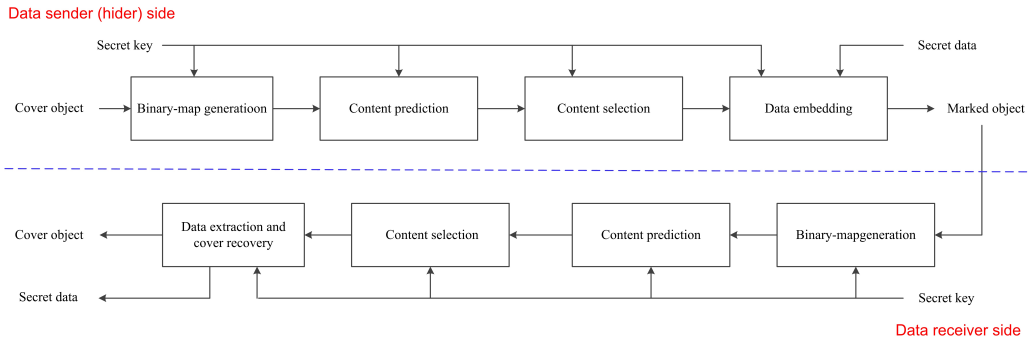


Figure 1. Sketch for the proposed framework. For the data hider, there are four key steps, i.e., binary-map generation, content prediction, content selection, and data embedding. For the data receiver, the key steps include binary-map generation, content prediction, content selection, data extraction, and cover recovery.

The content prediction enables us to use elements marked as "0" to predict those marked as "1". In this way, the PEs of elements marked as "1" can be obtained. The PEs will be used to carry a payload. The PEs are noise-like, which would not introduce obvious artifacts. After content prediction, one may use a content selection method to preferentially use the cover elements that can benefit the payload-distortion performance. For example, smooth pixels in an image will be preferred for RDH since their PEs are often smaller than the PEs of complex pixels. Finally, HS or its variants can be used for data embedding. Though some methods may not use HS directly, they can also be generalized by this framework. In the following, we revisit typical RDH algorithms and point out that they can be generalized by this framework.

2.1. LC-Based RDH

We analyze the simplest case that uses lossless compression. The pixels in an image are divided into two sets S_0 and S_1, where $|S_0| << |S_1|$. The LSBs of pixels in S_0 will carry the system parameters, such as the key. The LSBs of pixels in S_1 are losslessly compressed. The compressed code, original LSBs of S_0, and secret data are embedded into S_1 by LSB substitution. S_0 can be marked as "0" and S_1 for "1". Though there is no intuitive content prediction process, all prediction values can be treated as zero. The data hider can use a key to control the embedding order, which is equivalent to content selection. The LSB substitution is a special case of HS. Clearly, we first empty the LSBs of S_1. The prediction values of S_1 are all zero. Thus, if we only use the LSBs of S_1, the corresponding PEH has only occurrence at zero bin. By shifting "0" to "0/1", the secret bits can be embedded.

2.2. Ni et al's Method

Ni et al. [2] use the bin-pairs of the histogram directly determined from an image. Though there are not intuitive binary-map generation, content prediction, and content selection, one can mark a part of pixels as "1" and consider their prediction values as zero. To hide secret data, we self-embed some parameters into specific pixels. The LSBs of these pixels are replaced by the parameters, and the original LSBs are recorded as a part of the

secret data. These pixels are unchanged in subsequent procedure. They could be marked as "0" in the binary-map. Thus, Ni et al.'s method can be generalized by the proposed framework.

2.3. Tsai et al.'s Method

Tsai et al. [8] divide an image into disjoint blocks. The central pixel for each block serves as the prediction of adjacent pixels within the block. The central positions of all blocks are marked as "0", and the others are "1". Though there has no intuitive content selection process, one can randomly generate a value for each pixel so that the pixels can be orderly embedded with the HS operation.

2.4. Sachnev et al.'s Method

In Reference [10], the authors divide the pixels into two sets named as dot set and cross set. The pixels in the dot set are used to predict that in the cross set, which are thereafter used for data embedding. In this case, one can set the dot-pixel-positions as "0" and the others as "1". Then, with their introduced local-complexity function, the data hider can select relatively smooth pixels for RDH using the HS. When the dot pixels are used for data embedding, the process is similar. Sachnev et al.'s work exactly matches our framework.

2.5. Transformed Domain-Based RDH

With an image, we may not embed data in the spatial domain, but embed data in the transformed domain. In this case, we may determine the binary-map in the transformed domain. We can adjust the boundary pixels into the reliable range in the spatial domain in advance so as to avoid the underflow/overflow of pixels. Thereafter, the content prediction, content selection, and data embedding are applied to the transformed domain.

2.6. Expansion-Based RDH

With a prediction-error or difference d, we may not shift d to $d+1$ or $d-1$ to carry a message bit, but expand d to $2d$ and replace the LSB with the secret bit. This operation (called expansion) can be treated as a variant of HS, in that we are actually shifting d to $2d$ or $2d+1$, which looks like a "jump". Therefore, some expansion-based RDH algorithms can be described by the proposed framework, as well.

It can be inferred that our framework can generalize many RDH systems. It is quite helpful in practice since one can easily design, implement, and improve an RDH system. We will give detailed descriptions and analysis in the subsequent sections.

3. Details of Proposed Framework

In this section, we analyze each part of proposed framework in detail, and reliable techniques are introduced to achieve better performance. Furthermore, relationships between different parts and other perspectives are provided for better generalization.

3.1. Binary-Map Generation

The conventional methods often use a fixed binary-map, such as first-row-first-column [11], parity-column [16], chessboard [10], and block [8]. For example, Figure 2 shows the chessboard binary-map, where the pixels in the black region are kept unchanged to predict the pixels in the white region. Referring to Figure 2, one can mark the pixels in the black region as "0" and the rest as "1". Since the subsequent procedure relies heavily on the binary-map, different binary-maps result in different payload-distortion performance. Regardless of the embedding performance, using a fixed binary-map may allow an unauthorized decoder to reconstruct the directly embedded data.

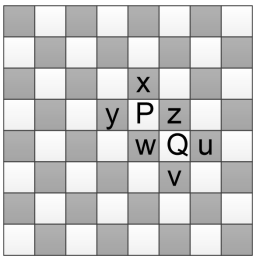


Figure 2. The chessboard binary-map, where pixels are divided to two disjoint sets: a set of pixels marked as white and a set of pixels marked as black.

To this end, we propose to use a *key-controlled* binary-map. It implies that the binary-map will always change due to a key, ensuring that an unauthorized receiver cannot produce the correct binary-pattern. Obviously, the traditional method is a special case of the proposed *key-controlled* binary-map. One may use a content-adaptive operation to generate the binary-map. Namely, with an initialized binary-map, one may further optimize it for improving performance. Algorithm 1 provides the pseudocode. Actually, with an initialized binary-map, one may just randomly select more elements to produce a new binary-map, which also ensures the system security.

Algorithm 1 Binary-map generation procedure

Input: A cover and a secret key.
Output: A binary-map and the side information (if any).
1: Initialize a binary-map
2: **while** need optimization **do**
3: Optimize the binary-map
4: **end while**
5: **return** final binary-map and side information (if any)

3.2. Content Prediction

With a binary-map, the elements marked as "0" are used to predict those marked as "1". The elements to be embedded may be *randomly* or *content-adaptively* distributed in the binary-map. We cannot directly use a fixed predictor depending on fixed neighbors. We propose to use a *dynamic* predictor for prediction. It means that a cover element is predicted from an indefinite number of its neighbors, e.g., a pixel is predicted from its four neighbors, and another one is predicted with its eight neighbors. There are two advantages: the security can be ensured, and the prediction accuracy may be improved.

For the latter, we take Figure 2 for explanation. If P has been predicted from {x, y, z, w}, instead of {z, w, u, v}, Q could be predicted from {P, z, w, u, v}. It is seen that the traditional predictors could be considered as a special dynamic predictor. Accordingly, a most important problem for content prediction is to design the dynamic predictor. It is necessary to find an efficient method to orderly predict the cover elements. One may predict the cover elements by a row-by-row manner, which, however, has limited generality. We propose a method called *degree-first prediction (DFP)*, which consists of *element-wise selection* and *element-wise prediction*, to orderly predict the elements. As shown in Algorithm 2, in each time, the first step selects an element out according to its *degree*. The second step determines its prediction value with a dynamic predictor. The degree of an element is a scalar that denotes the prediction priority. A larger degree means a higher priority. The degree of a cover element relies on its local context.

Algorithm 2 Degree-first prediction (DFP) procedure

Input: A cover, a binary-map and a secret key.
Output: A prediction version of the cover.
1: Initialization
2: **while** exist an element to be processed **do**
3: Choose an unprocessed element that has a largest degree
4: Find the prediction value (with a dynamic predictor)
5: Record the prediction value
6: Mark the element as processed
7: Update the degrees of the rest elements to be processed
8: **end while**
9: **return** the prediction version of the cover

3.3. Content Selection

The content selection aims to identify the embedding order. In image-based algorithms [10,17], the content selection is also named as pixel selection or sorting. For content selection, one has to define a local-complexity function to evaluate the prediction accuracy. Usually, a smaller local-complexity value implies better prediction accuracy. Thus, the cover elements can be orderly collected by sorting their local-complexities and then be orderly embedded. Algorithm 3 shows the pseudocode. In some methods, e.g., in Reference [17], a threshold, rather than sorting, may be used to take advantage of the smooth elements as much as possible. And these methods are in a sense the same as the method using sorting.

Algorithm 3 Local-complexity-based selection procedure

Input: A cover, a binary-map, the corresponding prediction version of the cover and the secret key.
Output: An ordered element sequence.
1: Initialization (e.g., empty the sequence)
2: **while** exist an element to be processed **do**
3: Find/update all required local-complexity values
4: Select an element with a smallest complexity value
5: Append the element to the sequence
6: Mark the element as processed
7: **end while**
8: **return** the ordered sequence

3.4. Data Embedding

After content selection, the data hider would perform data embedding. It is desirable to use HS or its variants to embed secret data since they can lead to superior payload-distortion performance. In detail, after generating an ordered pixel sequence, the data hider can determine out the corresponding PE sequence (PES). Thereafter, by using suitable histogram bin-pairs, the secret data can be easily embedded into the corresponding PEH. The traditional methods often empirically tune the shifting parameters, which is not desirable in applications. For a *single-layer* embedding, one may easily reproduce the reported simulation results. However, when adopting *multi-layer* embedding, it is actually time consuming and makes it hard for a reader to do the simulation due to the large space for searching suitable embedding parameters. To deal with this problem, in this paper, we propose a fast histogram shifting optimization (FastHiSO) algorithm to find *near-optimal* parameters. There are three advantages for the FastHiSO comparing with traditional operations: (1) the time complexity is relatively low; (2) a better payload-distortion performance can be achieved; and (3) the FastHiSO is deterministic, rather than empirical. In the following, we use digital image as the cover for detailing the FastHiSO algorithm.

Mathematically, we define $\mathbf{x}^{(t)}(t \geq 0)$ as the cover image after embedded with t times. For simplicity, let $\mathbf{x}^{(t)} = ({x_1}^{(t)}, {x_2}^{(t)}, \ldots, {x_n}^{(t)}) \in \mathcal{X} = \{\mathcal{I}\}^n$ be an n-pixel cover image with the pixel range $\mathcal{I}$, e.g., $\mathcal{I} = \{0, 1, \ldots, 255\}$ for 8-bit grayscale images. For a given message, we use $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(t)}$ to generate the marked image $\mathbf{x}^{(t+1)}(t \geq 0)$ by HS. Our goal is to find such HS parameters that both the distortion D between $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(t+1)}$ and the computational cost can be kept low at the same time. We limit ourselves to a commonly used *additive-distortion* measure, i.e., mean squared error (MSE):

$$D(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \frac{1}{n} \cdot \sum_{i=1}^{n} ({x_i}^{(0)} - {x_i}^{(t+1)})^2. \tag{1}$$

Let $h(v)$ represent the frequency of the PEH bin with a value of v, where $-|\mathcal{I}| < v < |\mathcal{I}|$. To embed data, with the generated PEH, we shift some PEs to vacate empty positions, and then embed secret bits by shifting the peak bins to the empty positions. Let $\mathbf{c}^{(t)} = ({c_1}^{(t)}, {c_2}^{(t)}, \ldots, {c_{n_t}}^{(t)}), n_t \leq n$, be all the pixels to be embedded. We, respectively, denote the prediction of $\mathbf{c}^{(t)}$ and its marked version by $\hat{\mathbf{c}}^{(t)} = (\hat{c}_1^{(t)}, \hat{c}_2^{(t)}, \ldots, \hat{c}_{n_t}^{(t)})$ and $\mathbf{s}^{(t)} = ({s_1}^{(t)}, {s_2}^{(t)}, \ldots, {s_{n_t}}^{(t)})$. Thus, we can determine the PEs $\mathbf{e}^{(t)} = ({e_1}^{(t)}, {e_2}^{(t)}, \ldots, {e_{n_t}}^{(t)})$ as follows:

$${e_i}^{(t)} = {c_i}^{(t)} - \hat{c}_i^{(t)}, (1 \leq i \leq n_t). \tag{2}$$

For data embedding, we first select two peak points $({l_p}^{(t)}, {r_p}^{(t)})$ and two integers $({T_l}^{(t)}, {T_r}^{(t)})$, where ${l_p}^{(t)} < {r_p}^{(t)}$, ${T_l}^{(t)} > 0$, ${T_r}^{(t)} > 0$. Then, secret bits can be embedded by using the HS operation, namely

$${s_i}^{(t)} = \hat{c}_i^{(t)} + \hat{e}_i^{(t)} = \hat{c}_i^{(t)} + {e_i}^{(t)} + \Delta({e_i}^{(t)}), \tag{3}$$

where

$$\Delta({e_i}^{(t)}) = \begin{cases} -{T_l}^{(t)}, & \text{if } {e_i}^{(t)} \leq {l_p}^{(t)} - {T_l}^{(t)}; \\ {T_r}^{(t)}, & \text{if } {e_i}^{(t)} \geq {r_p}^{(t)} + {T_r}^{(t)}; \\ {e_i}^{(t)} - {r_p}^{(t)} + b, & \text{else if } {e_i}^{(t)} \geq {r_p}^{(t)}; \\ {e_i}^{(t)} - {l_p}^{(t)} - b, & \text{else if } {e_i}^{(t)} \leq {l_p}^{(t)}; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Here, $b \in \{0, 1\}$ is the current bit to be embedded.

We use $\mathbf{o}^{(t)} = ({o_1}^{(t)}, {o_2}^{(t)}, \ldots, {o_{n_t}}^{(t)})$ to denote the original pixel values of $\mathbf{c}^{(t)}$ in $\mathbf{x}^{(0)}$. For the pixels not belonging to $\mathbf{c}^{(t)}$, the introduced distortion can be roughly considered as fixed since we will not embed secret data into these pixels (though we may alter some pixels prior to embedding, e.g., we may empty some LSBs to store the secret key). Therefore, for $(t+1)$-layer $(t \geq 0)$ embedding (i.e., to generate $\mathbf{x}^{(t+1)}$), our optimization task is

$$D(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \min_{{l_p}^{(t)}, {r_p}^{(t)}, {T_l}^{(t)}, {T_r}^{(t)}} \frac{1}{n} \cdot \sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2 + \frac{1}{n} \cdot C, \tag{5}$$

where C is a constant. We further have

$$\begin{aligned} \sum_{i=1}^{n_t} ({s_i}^{(t)} - {o_i}^{(t)})^2 &= \sum_{i=1}^{n_t} (\hat{c}_i^{(t)} + \hat{e}_i^{(t)} - {o_i}^{(t)})^2 \\ &= \sum_{i=1}^{n_t} ({c_i}^{(t)} - {o_i}^{(t)} + \hat{e}_i^{(t)} - {e_i}^{(t)})^2 \\ &= \sum_{i=1}^{n_t} ({\alpha_i}^{(t)} + {\beta_i}^{(t)})^2, \end{aligned} \tag{6}$$

where ${\alpha_i}^{(t)} = {c_i}^{(t)} - {o_i}^{(t)}$ and ${\beta_i}^{(t)} = \hat{e}_i^{(t)} - {e_i}^{(t)} = \Delta({e_i}^{(t)})$.

Therefore, it is inferred that our final optimization task is

$$J(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \min_{l_p{}^{(t)}, r_p{}^{(t)}, T_l{}^{(t)}, T_r{}^{(t)}} \sum_{i=1}^{n_t} (\alpha_i{}^{(t)} + \beta_i{}^{(t)})^2. \tag{7}$$

Obviously, all $\alpha_i{}^{(t)}$ $(1 \le i \le n_t)$ are constants before embedding, meaning that they can be determined in advance. It can be seen that $|\beta_i{}^{(t)}| = |\Delta(e_i{}^{(t)})| \le \max\{T_l{}^{(t)}, T_r{}^{(t)}\}$ for all $1 \le i \le n_t$, which gives us the chance to quickly determine $J(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)})$ for fixed $\mathbf{c}^{(t)}$, $\mathbf{o}^{(t)}$ and $\mathbf{e}^{(t)}$. In detail, we use a 2D histogram-matrix $\mathbf{H} = \{H_{u,v} | -|\mathcal{I}| \le u, v \le |\mathcal{I}|\}$ to record the occurrence of every possible integer-pair (u, v) in advance, where u represents the possible value of $\alpha_i{}^{(t)}$, and v shows the possible value of $e_i{}^{(t)}$. In this way, Equation (7) is equivalent to

$$J(\mathbf{x}^{(0)}, \mathbf{x}^{(t+1)}) = \min_{l_p{}^{(t)}, r_p{}^{(t)}, T_l{}^{(t)}, T_r{}^{(t)}} \sum_{u,v} H_{u,v} \cdot (u + \Delta(v))^2. \tag{8}$$

According to Equation (4), we further have

$$\begin{aligned} \sum_{u,v} H_{u,v} \cdot (u + \Delta(v))^2 &= \sum_u \sum_v H_{u,v} \cdot (u + \Delta(v))^2 = \sum_u \sum_{v \le l_p{}^{(t)} - T_l{}^{(t)}} H_{u,v} \cdot (u - T_l)^2 \\ &+ \sum_u \sum_{v \ge r_p{}^{(t)} + T_r{}^{(t)}} H_{u,v} \cdot (u + T_r)^2 + \sum_u \sum_{l_p{}^{(t)} - T_l{}^{(t)} + 1 \le v \le l_p{}^{(t)}} H_{u,v} \cdot (u + v - l_p{}^{(t)} - b)^2, \\ &+ \sum_u \sum_{r_p{}^{(t)} \le v \le r_p{}^{(t)} + T_r - 1} H_{u,v} \cdot (u + v - r_p{}^{(t)} + b)^2 + \sum_u \sum_{l_p{}^{(t)} + 1 \le v \le r_p{}^{(t)} - 1} H_{u,v} \cdot u^2 \end{aligned} \tag{9}$$

where b is the corresponding bit to be embedded.

Since the embedded bits are always encrypted, to keep the computational cost low, we will consider

$$\begin{aligned} &\sum_u \sum_{l_p{}^{(t)} - T_l{}^{(t)} + 1 \le v \le l_p{}^{(t)}} H_{u,v} \cdot (u + v - l_p{}^{(t)} - b)^2 \approx \\ &\qquad \sum_u \sum_{l_p{}^{(t)} - T_l{}^{(t)} + 1 \le v \le l_p{}^{(t)}} \frac{H_{u,v}}{2} \cdot (u + v - l_p{}^{(t)})^2 \\ &\qquad\qquad + \sum_u \sum_{l_p{}^{(t)} - T_l{}^{(t)} + 1 \le v \le l_p{}^{(t)}} \frac{H_{u,v}}{2} \cdot (u + v - l_p{}^{(t)} - 1)^2, \end{aligned} \tag{10}$$

and

$$\begin{aligned} &\sum_u \sum_{r_p{}^{(t)} \le v \le r_p{}^{(t)} + T_r - 1} H_{u,v} \cdot (u + v - r_p{}^{(t)} + b)^2 \approx \\ &\qquad \sum_u \sum_{r_p{}^{(t)} \le v \le r_p{}^{(t)} + T_r - 1} \frac{H_{u,v}}{2} \cdot (u + v - r_p{}^{(t)})^2 \\ &\qquad\qquad + \sum_u \sum_{r_p{}^{(t)} \le v \le r_p{}^{(t)} + T_r - 1} \frac{H_{u,v}}{2} \cdot (u + v - r_p{}^{(t)} + 1)^2. \end{aligned} \tag{11}$$

Algorithm 4 shows the FastHiSO procedure. In Algorithm 4, the 2D histogram-matrix $\mathbf{H}$ can be determined with a time complexity of $O(n_t)$, which is linear with respect to the number of cover elements to be embedded. For Line 6 in Algorithm 4, according to Equation (9), the time complexity is $O(|I|^2)$ at the worst case for the fixed $(l_p{}^{(t)}, r_p{}^{(t)}, T_l{}^{(t)}, T_r{}^{(t)})$. Since the PEH is Gaussian-like [17], the optimal values of $l_p{}^{(t)}$ and $r_p{}^{(t)}$ should be close to zero. It indicates that, from the empirical point of view, the absolute values of $l_p{}^{(t)}$ and $r_p{}^{(t)}$ can be limited to a small range, e.g., $\max\{|l_p{}^{(t)}|, |r_p{}^{(t)}|\} \le 64$. The

values of ${T_l}^{(t)}$ and ${T_r}^{(t)}$ can be also limited to a small range, as well, e.g., $\max\{{T_l}^{(t)}, {T_r}^{(t)}\} \leq 4$. In this way, the near-optimal parameters can be quickly determined, and the corresponding complexity can be approximately expressed as $O(kn_t)$, where $k << n_t$ is a relatively small integer.

With the proposed FastHiSO algorithm, one can easily embed secret data into the cover pixel-sequence. The embedding process is described in Algorithm 5. To avoid the underflow/overflow problem, we need to adjust the pixels with boundary values into the reliable range in advance. The preprocessed pixels should be recorded as side information, which will be self-embedded.

Algorithm 4 Fast histogram shifting optimization (FastHiSO)

Input: The ordered cover sequence $\mathbf{c}^{(t)}$, the original sequence $\mathbf{o}^{(t)}$, the prediction sequence $\hat{\mathbf{c}}^{(t)}$, and the payload size ρ.
Output: $({l_p}^{(t)}, {r_p}^{(t)}, {T_l}^{(t)}, {T_r}^{(t)})$
1: Empty **H**
2: **for** $1 \leq i \leq n_t$ **do**
3: Find $u = {\alpha_i}^{(t)}$ and $v = {e_i}^{(t)}$
4: Set $H_{u,v} \leftarrow H_{u,v} + 1$
5: **end for**
6: Find the near-optimal $({l_p}^{(t)}, {r_p}^{(t)}, {T_l}^{(t)}, {T_r}^{(t)})$ with Equations (8)–(11), subject to the payload size ρ.
7: **return** $({l_p}^{(t)}, {r_p}^{(t)}, {T_l}^{(t)}, {T_r}^{(t)})$

Algorithm 5 Data embedding procedure

Input: The ordered cover sequence $\mathbf{c}^{(t)}$, the original sequence $\mathbf{o}^{(t)}$, the prediction sequence $\hat{\mathbf{c}}^{(t)}$, the required payload $\mathcal{L}$, and the secret key $\mathbf{k}_{\text{emb}}$.
Output: The marked pixel-sequence $\mathbf{s}^{(t)}$.
1: Call FastHiSO to find near-optimal HS parameters
2: Embed $\mathcal{L}$ into the corresponding PEH
3: Construct the marked pixel-sequence $\mathbf{s}^{(t)}$
4: **return** $\mathbf{s}^{(t)}$ {thereafter, we further generate $\mathbf{x}^{(t+1)}$}

3.5. Data Extraction and Cover Recovery

Once secret data is successfully embedded, the resulting marked object will be sent to the desired receiver, who should be able to reconstruct the original cover and extract the embedded data without error according to the secret key. The data extraction and cover recovery procedure can be considered as an inverse process to the data hider.

3.6. Relationships between Different Parts

There may exist interactions between the different parts. We present two different relationships for better generalization.

3.6.1. Relationship between Binary-Map Generation, Content Prediction, and Content Selection

With an initialized binary-map, we can optimize it for realizing better payload-distortion performance. We may use the procedure similar to content prediction and selection, e.g., Algorithm 6 shows an example of optimizing a binary-map using Algorithms 2 and 3.

Algorithm 6 Binary-map generation using Algorithms 2 and 3

Input: A cover and a secret key.
Output: A binary-map.
1: Initialize a binary-map
2: **while** need optimization **do**
3: Call Algorithm 2 to predict the cover
4: Call Algorithm 3 to generate a sequence
5: Choose a certain number of elements (which are marked as "1") out from the ordered sequence
6: Mark them as "0" and update the binary-map
7: **end while**
8: **return** the final binary-map

3.6.2. Relationship between Content Prediction and Content Selection

For content prediction, the data hider has to identify the prediction order. As mentioned above, each time, the data hider chooses an element with a largest degree for prediction. For content selection, the data hider uses a local-complexity function to choose an element with the lowest complexity in each time. It indicates that every to-be-processed element will be associated with a local-complexity value. As the degree and local-complexity are scalars, they may affect each other. e.g., in Algorithm 2, after executing Line 7, one can directly append the processed element to a sequence. Thus, an ordered element sequence presented in Algorithm 3 can be also generated after content prediction.

3.7. Other Perspectives

Next, two different types of dynamic predictors, i.e., raw-content-independent (RCI)-based predictor and raw-content-dependent (RCD)-based predictor are considered. RCI-based predictor uses the marked or predicted values of the neighbors of the present element as the prediction context. It means that the prediction process for an element may not affect the prediction process of another one directly. Thus, the content prediction order for a data receiver is probably identical to the sender. In contrast, RCD-based predictor uses the raw content, which can benefit prediction accuracy. Thus, we should ensure that the raw values of the context have been obtained before prediction.

Intuitively, a dynamic predictor enables us to predict different cover elements from different contexts. Actually, a general dynamic predictor also implies that:

- *Fusion of multiple subpredictors*: The conventional methods use a single predictor. Actually, they can be treated as a fusion of multiple subpredictors. We take median edge detector (MED) [13] for explanation, i.e.,

$$\hat{x} = \begin{cases} \min\{v_1, v_3\}, & \text{if } v_4 \geq \max\{v_1, v_3\}, \\ \max\{v_1, v_3\}, & \text{if } v_4 \leq \min\{v_1, v_3\}, \\ v_1 + v_3 - v_4, & \text{otherwise}, \end{cases} \tag{12}$$

  where v_1, v_3, and v_4 are specific neighbors of x. It can be seen that the MED essentially uses three subpredictors, i.e., $\min\{v_1, v_3\}$, $\max\{v_1, v_3\}$, and $v_1 + v_3 - v_4$. Therefore, it is inferred that a dynamic predictor corresponds to a fusion of multiple subpredictors. A key work is to choose the suitable subpredictor according to the local context.
- *Fusion of multiple subhistograms*: The histogram to be embedded also can be regarded as a fusion of multiple subhistograms as a subpredictor corresponds to a subhistogram. Though we may not directly use the subhistograms separately, it inspires us to divide a histogram into multiple subhistograms for payload-distortion optimization, which has been exploited by Li et al. [18].
- *Fusion of multiple subcovers*: Different subhistograms are corresponding to different subcovers even though the elements belonging to a subcover may be widely or near-randomly distributed in the original cover. In other words, we may divide the cover

into subcovers for payload-distortion optimization since different subcovers may have different texture characteristics. For example, a cover image may be divided into disjoint blocks. Notice that this perspective no longer focuses on only the dynamic predictor, but rather the design of an RDH system, e.g., as in Reference [19].

In addition, by default, we consider the cover element to be embedded as a single value for better understanding. Actually, the "element" can be a vector. For example, in Reference [20], two pixels are grouped as a pair to carry the secret data.

4. Two Examples Based on Proposed Framework

In this section, we will present two novel RDH algorithms based on the proposed framework to show the efficiency and applicability of the proposed framework.

4.1. Prediction-Error of Prediction Error (PPE)-Based RDH

Due to the spatial correlations between neighboring pixels, many existing works use PEs to carry the secret data. Actually, there also exist correlations between neighboring PEs. An evidence can be found in the prediction mechanism of video lossy compression. For example, in intra prediction, the prediction block for an intra 4×4 luma macroblock is generated with 9 possible prediction modes. Then, to improve the coding efficiency, the prediction mode of a luma macroblock is predicted from the prediction modes of neighboring luma macroblocks since correlations also exist between the neighboring prediction modes. The success of steganalysis by modeling the differences between neighboring pixels with low-order Markov chains [21] also reveals that correlations exist between the neighboring PEs if we consider the differences as a kind of PEs. Based on this perspective, we here present a prediction-error of prediction error (PPE)-based RDH algorithm, which is an extension of Reference [14].

With a cover image $\mathbf{x} \in \{0, 1, \ldots, 2^d - 1\}^{n_1 \times n_2}$ and a binary-map $\mathbf{b}_{\text{map}} \in \{0, 1\}^{n_1 \times n_2}$, we define a neighbor-set $\mathcal{D}_0(x_{i,j})$ for each $x_{i,j} \in \mathbf{x}$ corresponding to $b_{i,j} = 1$ ($\in \mathbf{b}_{\text{map}}$). $\mathcal{D}_0(x_{i,j})$ includes the neighboring pixels of $x_{i,j}$ that are marked as "0" in $\mathbf{b}_{\text{map}}$. We first use the pixels in $\mathcal{D}_0(x_{i,j})$ to predict $x_{i,j}$, i.e., $\hat{x}_{i,j} = f_0(\mathcal{D}_0(x_{i,j}))$, from which we can obtain the PE, denoted by $e_{i,j} = x_{i,j} - \hat{x}_{i,j}$. Then, $e_{i,j}$ is further predicted by exploiting the PEs of the pixels in $\mathcal{D}_0(x_{i,j}) \cup \mathcal{D}_1(x_{i,j})$, i.e., $\hat{e}_{i,j} = f_1(\mathcal{D}_0(x_{i,j}) \cup \mathcal{D}_1(x_{i,j}))$. Here, $\mathcal{D}_1(x_{i,j})$ represents the pixels adjacent to at least one pixel in $\mathcal{D}_0(x_{i,j})$. Note that the definition of $\mathcal{D}_0(x_{i,j})$ and $\mathcal{D}_1(x_{i,j})$ may be different from each other. The PPE of $x_{i,j}$ is described as:

$$e^{\circ}{}_{i,j} = e_{i,j} - \hat{e}_{i,j} = x_{i,j} - f_0(\mathcal{D}_0(x_{i,j})) - f_1(\mathcal{D}_0(x_{i,j}) \cup \mathcal{D}_1(x_{i,j})). \tag{13}$$

We use the chessboard pattern to construct $\mathbf{b}_{\text{map}}$. As shown in Figure 3, $\mathcal{D}_0(x_{i,j}) = \{x_{i-1,j}, x_{i,j+1}, x_{i+1,j}, x_{i,j-1}\}$ is first utilized to predict $x_{i,j}$. Then, the PEs of the pixels in $\{x_{i-1,j}, x_{i,j+1}, x_{i+1,j}, x_{i,j-1}\}$ are used to predict the PE of $x_{i,j}$. The prediction process of neighboring pixels is different from $x_{i,j}$, e.g., $x_{i,j+1}$ is predicted from $\{x_{i-1,j}, x_{i-1,j+2}, x_{i+1,j}, x_{i+1,j+2}\}$ since the cross set is to be embedded and the dot set for unchanged.

As shown in Figure 3, we first predict $x_{i,j}$ along the horizontal and vertical directions. The two directional predictors are:

$$x_{i,j}{}' = \frac{x_{i,j-1} + x_{i,j+1}}{2}, \; x_{i,j}{}'' = \frac{x_{i-1,j} + x_{i+1,j}}{2}. \tag{14}$$

Then, $\hat{x}_{i,j}$ is determined by:

$$\hat{x}_{i,j} = \text{Round}(w_{i,j} \cdot x_{i,j}{}' + (1 - w_{i,j}) \cdot x_{i,j}{}''). \tag{15}$$

Here, Round$(\cdot)$ returns the nearest integer, $w_{i,j}$ is defined as:

$$w_{i,j} = \frac{\sigma_{i,j}{}''}{\sigma_{i,j}{}' + \sigma_{i,j}{}''}, \tag{16}$$

where

$$\sigma_{i,j}{}' = \frac{1}{3} \cdot \sum_{v \in \{x_{i,j-1}, x_{i,j}{}', x_{i,j+1}\}} (v - \frac{x_{i,j}{}' + x_{i,j}{}''}{2})^2. \tag{17}$$

$$\sigma_{i,j}{}'' = \frac{1}{3} \cdot \sum_{v \in \{x_{i-1,j}, x_{i,j}{}'', x_{i+1,j}\}} (v - \frac{x_{i,j}{}' + x_{i,j}{}''}{2})^2. \tag{18}$$

It is straightforward to process other pixels in the cross set and the dot set with the similar procedure, e.g., $x_{i,j+1}$ will be predicted from the two diagonal directions. Thereafter, we use the average value of the PEs of pixels in $\mathcal{D}_0(x_{i,j})$ as the prediction of $e_{i,j}$, i.e., $\hat{e}_{i,j}$. And, the corresponding PPE can be determined according to Equation (13). Therefore, all PPEs of pixels in the cross set can be determined. By defining a local-complexity function, the PPEs can be sorted in a decreasing order of the prediction accuracy, which can benefit embedding performance. Here, the local-complexity function is defined as:

$$\epsilon_{i,j} = \left[\frac{1}{6} \cdot \sum_{s=1}^{6} ({\varrho_{i,j}}^{(s)} - \sum_{t=1}^{6} {\varrho_{i,j}}^{(t)}/6)^2 \right]^{1/2}, \tag{19}$$

where ${\varrho_{i,j}}^{(1)} = |x_{i-1,j} - x_{i,j+1}|$, ${\varrho_{i,j}}^{(2)} = |x_{i-1,j} - x_{i+1,j}|$, ${\varrho_{i,j}}^{(3)} = |x_{i-1,j} - x_{i,j-1}|$, ${\varrho_{i,j}}^{(4)} = |x_{i,j+1} - x_{i+1,j}|$, ${\varrho_{i,j}}^{(5)} = |x_{i,j+1} - x_{i,j-1}|$, and ${\varrho_{i,j}}^{(6)} = |x_{i+1,j} - x_{i,j-1}|$. Accordingly, an ordered PPE sequence can be generated. We sincerely refer the readers to References [14,22] for more details. Note that one may use other efficient binary-maps, pixel prediction procedures, and local-complexity functions. Finally, for a payload, by applying the proposed FastHiSO algorithm, one can quickly find the near-optimal parameters and embed the secret bits into the PPE histogram according to the operation similar to Equations (3) and (4). For a receiver, data extraction and image recovery correspond to a reverse operation.

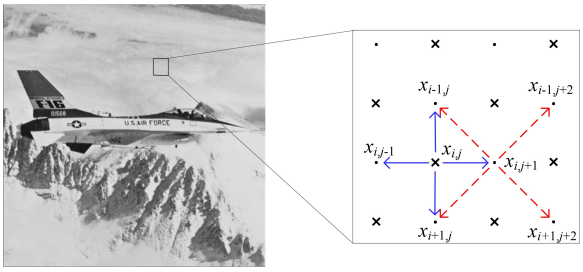


Figure 3. The pixel prediction pattern. $x_{i,j}$ is predicted from four neighbors in the dot set, and $x_{i,j+1}$ is predicted from four neighbors still in the dot set.

We present some experimental results to show the efficiency of PPE-based RDH method. Six standard test images, from smooth to complex (http://sipi.usc.edu/database/, accessed on 10 January 2021): *Airplane*, *Lena*, *Baboon*, *Tiffany*, *Peppers*, and *Sailboat* (i.e., *Fishing boat*); all are grayscaled with a size of 512×512 used. During data embedding, we set $\max\{|{l_p}^{(t)}|, |{r_p}^{(t)}|\} \leq 255$ and $\max\{{T_l}^{(t)}, {T_r}^{(t)}\} \leq 1$. Figure 4 shows the payload-distortion performance comparison between some prediction-based RDH works and the proposed method. It is seen that, for relatively low embedding rates, the proposed method significantly outperforms the related works, meaning that the data embedding performance can benefit from the cover PPEs and the FastHiSO algorithm. On the other hand, in Figure 4, when the embedding payload increases, the PSNR improvement is not significant, even slightly bad (e.g., *Airplane*). It means that the generated PPE histogram (PPEH) has its limitation. Overall, the proposed method can still provide better payload-distortion performance compared to a part of related works and maintain satisfactory trade-off between the embedding payload and the distortion.

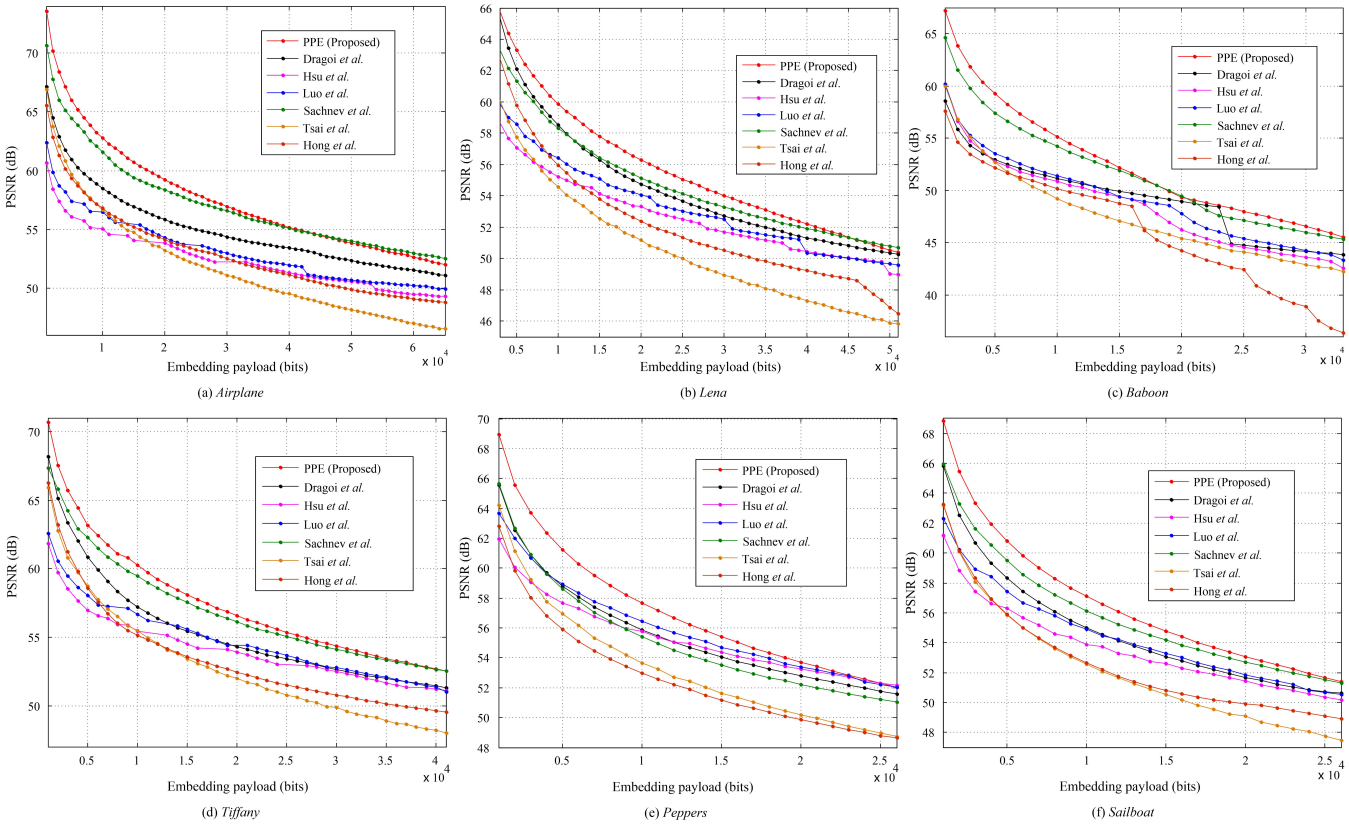


Figure 4. The payload-distortion performance comparison between the state-of-the-art methods of Tsai et al. [8], Sachnev et al. [10], Hong et al. [11], Luo et al. [12], Hsu et al. [23], Dragoi et al. [24], and the proposed method (PPE).

In this subsection, we use the PPEs to carry a payload. It is straightforward to apply higher-order PEs to hide the secret data. It can be inferred that both the PPEs and the higher-order PEs essentially correspond to a kind of *calibration* operation to improve the prediction accuracy so as to provide better performance. Any predictor can be written as a form of predicting a pixel itself, so that PE-based RDH and PPE-based RDH, as well as the higher-order PE-based RDH, are all generalized by the proposed framework.

Therefore, a core work for prediction-based RDH system is to keep the *calibration* operation as accurate as possible. In this paper, we will not study the prediction accuracy in depth since it is not the main interest of this paper. The traditional methods usually have no *calibration* term (or are considered as constant).

4.2. Dynamic Selection-and-Prediction (DSP)-Based RDH

The aforementioned algorithm uses a fixed pattern to construct $\mathbf{b}_{\text{map}}$, providing better embedding performance compared to the related works. According to Kerckhoffs's principle, this previously specified pattern may allow one to successfully reconstruct the marked histogram, and extract the directly embedded information, which is not desirable for applications. To overcome this drawback, we here use a key to initialize the binary-map generation such that the final $\mathbf{b}_{\text{map}}$ is always changing due to the key.

For RDH, smooth regions often correspond to better payload-distortion performance. To improve the performance, we can optimize the initialized $\mathbf{b}_{\text{map}}$ to select smooth regions out for data embedding as long as the smooth pixels can carry the payload. However, since all "1"s in $\mathbf{b}_{\text{map}}$ will be randomly or content-adaptively distributed due to the key, we cannot directly use such predictors that rely on the specified neighbors. To deal with this problem, we propose to use a *dynamic* predictor for pixel prediction. That is why we call it as *dynamic selection-and-prediction (DSP)*-based RDH method. Thereafter, with a well-defined local-complexity function and the FastHiSO algorithm, the secret data can be embedded into the corresponding PEH. It can be seen that DSP-based RDH meets

the requirement of our framework. In the following, we will describe the details of the proposed DSP-based RDH, which is an extension version of Reference [15].

For self-contained, let $\mathcal{X}$ be an image with $n = h \times w$ pixels; for compactness, we sometimes consider $\mathcal{X}$ as the set including all pixels and say "pixel $x_{i,j}$" meaning a pixel located at position (i, j), whose grayscale value is $x_{i,j}$. We first use a secret key to initialize $\mathbf{b}_{\text{map}}$, where "0"s are pseudo-randomly distributed. For simplicity, we use $\mathcal{S}_0$ to denote the pixel-set containing all pixels marked as "0" in the initialized $\mathbf{b}_{\text{map}}$. Then, we optimize $\mathbf{b}_{\text{map}}$ by selecting more complex pixels out, denoted by $\mathcal{S}_1$. Here, $\mathcal{S}_0 \cap \mathcal{S}_1 = \emptyset$ and $\mathcal{S}_0 \cup \mathcal{S}_1 \subset \mathcal{X}$. Note that the pixels in $\mathcal{S}_0 \cup \mathcal{S}_1$ will be marked as "0", and that in $\mathcal{X} \setminus (\mathcal{S}_0 \cup \mathcal{S}_1)$ correspond to "1". The generation of $\mathcal{S}_1$ involves two steps, namely the *degree-first prediction (DFP)* and the *complexity-first selection (CFS)*.

DFP Procedure: We collect all pixels in $\mathcal{S}_0$, and only exploit these pixels to predict the pixels in $\mathcal{X} \setminus \mathcal{S}_0$. The pixels are orderly predicted according to the associated degrees. The *degree* of a pixel is defined as the size of its *degree-set*, which is a subset of its *neighbor-set*. The neighbor-set of a pixel $x_{i,j}$ is defined as:

$$\mathcal{N}_{i,j} = \left\{ x_{i+u,j+v} | 1 \leq u^2 + v^2 \leq r^2; u, v \in \mathbf{Z} \right\}. \tag{20}$$

By default, $r = \sqrt{2}$. Thus, except for boundary positions, the neighbor-set of a pixel consists of eight pixels. The degree-set of $x_{i,j}$ is then determined by:

$$\mathcal{D}_{i,j} = \mathcal{N}_{i,j} \cap (\mathcal{S}_0 \cup \mathcal{A}), \tag{21}$$

where $\mathcal{A}$ represents the pixel-set consisting of the pixels that have been previously predicted. We always have $\mathcal{S}_0 \cap \mathcal{A} = \emptyset$.

A pixel with a larger degree will be predicted prior to that with a smaller degree. The reason is, only pixels in the degree-set are utilized to predict a pixel. Thus, a pixel with a larger degree can be predicted from more pixels, meaning that the pixel can be well predicted as more context are provided. That is why we consider the prediction as "degree-first prediction". In this paper, the prediction of $x_{i,j}$ is defined as:

$$\hat{x}_{i,j} = \frac{\sum\limits_{x_{u,v} \in (\mathcal{N}_{i,j} \cap \mathcal{A})} \hat{x}_{u,v} + \sum\limits_{x_{u,v} \in (\mathcal{N}_{i,j} \cap \mathcal{S}_0)} x_{u,v}}{|\mathcal{D}_{i,j}|}. \tag{22}$$

Based on the above description, we describe the proposed DFP procedure as follows.

(Step 1) Set $\mathcal{A} = \emptyset$. For all $x_{i,j} \in \mathcal{X} \setminus \mathcal{S}_0$, compute $\mathcal{D}_{i,j}$ with Equation (21). Mark all $x_{i,j} \in \mathcal{X} \setminus \mathcal{S}_0$ as unprocessed.

(Step 2) Select such a *unprocessed* pixel $x_{i,j} \in \mathcal{X} \setminus \mathcal{S}_0$ that has the largest degree in $\mathcal{X} \setminus (\mathcal{S}_0 \cup \mathcal{A})$. If there are multiple pixels that have the largest degree, choose one according to a key or a specified rule. Find $\hat{x}_{i,j}$ with Equation (22).

(Step 3) Mark $x_{i,j}$ as processed and update $\mathcal{A}$ as $\mathcal{A} \cup \{x_{i,j}\}$, and further update $\mathcal{D}_{i,j}$ with Equation (21).

(Step 4) Terminate the procedure if all pixels in $\mathcal{X} \setminus \mathcal{S}_0$ are processed; otherwise, go to **(Step 2)**.

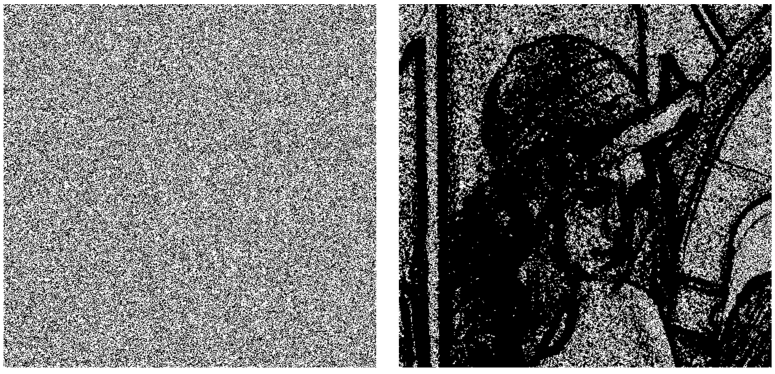
CFS Procedure: After all required pixels are predicted, we are to select a part of the predicted pixels out to constitute $\mathcal{S}_1$. It relies on the local complexities of the pixels. Here, we define the local complexity of a predicted pixel $x_{i,j}$ as:

$$\rho_{i,j} = \frac{\sum\limits_{x_{u,v} \in \mathcal{P}_{i,j}} (x_{u,v} - \hat{x}_{i,j})^2 + \sum\limits_{x_{u,v} \in \mathcal{Q}_{i,j}} (\hat{x}_{u,v} - \hat{x}_{i,j})^2}{|\mathcal{N}_{i,j}|}, \tag{23}$$

where $\mathcal{P}_{i,j} = \mathcal{S}_0 \cap \mathcal{N}_{i,j}$, $\mathcal{Q}_{i,j} = \mathcal{N}_{i,j} \setminus \mathcal{S}_0$.

A larger local complexity indicates that the pixel is likely to be located at a more complex region. Every predicted pixel is associated with its local complexity. We sort the pixels by their local complexities in an increasing order. In this way, the $|\mathcal{S}_1|$ pixels with largest complexity-values are chosen to constitute $\mathcal{S}_1$, and the selected pixels are likely located at relatively complex regions. Since we only use the original values of pixels in $\mathcal{S}_0$, both the data hider and receiver should be able to construct the identical $\mathcal{S}_0 \cup \mathcal{S}_1$ with the key. Thereafter, we use the pixels in $\mathcal{S}_0 \cup \mathcal{S}_1$ to predict the pixels in $\mathcal{X} \setminus (\mathcal{S}_0 \cup \mathcal{S}_1)$ by applying the proposed DFP procedure. An ordered pixel-sequence together with the corresponding PEs can be generated according to the proposed CFS procedure. With the resulting PEH and the FastHiSO, the secret data can be sucessfully embedded.

We present some experimental results to show the embedding performance. The system parameters mainly include $|\mathcal{S}_0|$, $|\mathcal{S}_1|$ and the secret key. For a payload, it is free to choose $|\mathcal{S}_0|$ and $|\mathcal{S}_1|$. For convenience, we consider $|\mathcal{S}_0| = |\mathcal{S}_1|$ in default, which may be not optimal. Figure 5 shows the distribution of $\mathcal{S}_0$ and $\mathcal{S}_0 \cup \mathcal{S}_1$ (black regions) tested on *Lena* image due to proposed binary-map generation procedure. It is seen that the proposed binary-map generation procedure can *auto-capture* the relatively smooth pixels (*white regions*) for data hiding, which is quite desirable for practice.



(a) Initialized binary-map (b) Optimized binary-map

Figure 5. An example of the proposed binary-map generation procedure tested on *Lena*: $r = \frac{|\mathcal{S}_0 \cup \mathcal{S}_1|}{|\mathcal{X}|} \times 100\% = 80\%$ (*black* regions in optimized version).

4.2.1. Evaluation on Standard Images

We test the above algorithm on the standard images mentioned previously. Since it is free to set r (please refer to Figure 5), for a payload, we generate the marked image with the highest PSNR by varying r from 0.01 to 0.99 with a step value of 0.01 since the data hider always has the freedom to generate a marked image with a better quality. Figure 6 shows the corresponding payload-distortion performance comparison. It is inferred that the used data hiding operation can benefit from the pixel selection and prediction procedure and, therefore, provide relatively good embedding performance. In Figure 6, with relatively low embedding rates, the proposed scheme significantly outperforms the related works. For example, for *Airplane* embedded with 0.1×10^3 bits, the PSNR value of proposed method is 74.97 dB, which is extremely close to the theoretical uniform-embedding bound 75.33 dB (= $10 \times \log_{10} \frac{255^2}{0.5 \times 1000/512/512}$). It indicates that the proposed algorithm indeed has the ability to well-capture the smooth pixels out for data hiding. Note that, here, "*uniform-embedding*" means all PEs are shifted to carry a message that "0/1" are evenly distributed.

It can be also observed that, when the embedding payload increases, the PSNRs are likely to be relatively lower than some of the related works. For example, the proposed RDH scheme has a weaker performance for the *Baboon* image after embedded with more than 0.7×10^4 bits. The reason is that the proposed algorithm aims to select the complex pixels for prediction and smooth pixels for data hiding, while the amount of smooth pixels within an image is actually limited due to the image content. The *Baboon* image is full of complex content so that many complex pixels are finally selected out for data hiding.

However, the complex pixels are likely to be predicted with a larger PE, which, therefore, cannot keep a good payload-distortion performance. It also implies that the proposed pixel predictor should be further improved when a larger payload is required or the number of smooth pixels is limited.

In addition, to provide better performance, it is necessary to apply the optimization operation for binary-map generation (i.e., further using $\mathcal{S}_0$ to generate $\mathcal{S}_1$). The reason is that, for an embeddable payload, $\mathcal{S}_0 \cup \mathcal{S}_1$ provides more original image context for pixel prediction than only $\mathcal{S}_0$, which could benefit prediction accuracy. Meanwhile, the optimization operation can select regions of interest (RoI) for RDH, which can result in better payload-distortion performance. Figure 7 shows the content degradation comparison (with MSE measure) between only using $\mathcal{S}_0$ and using $\mathcal{S}_0 \cup \mathcal{S}_1$ for the *Lena* and *Airplane* image. It can be seen that an optimized binary-map would be better than a completely random binary-map in terms of payload-distortion performance.

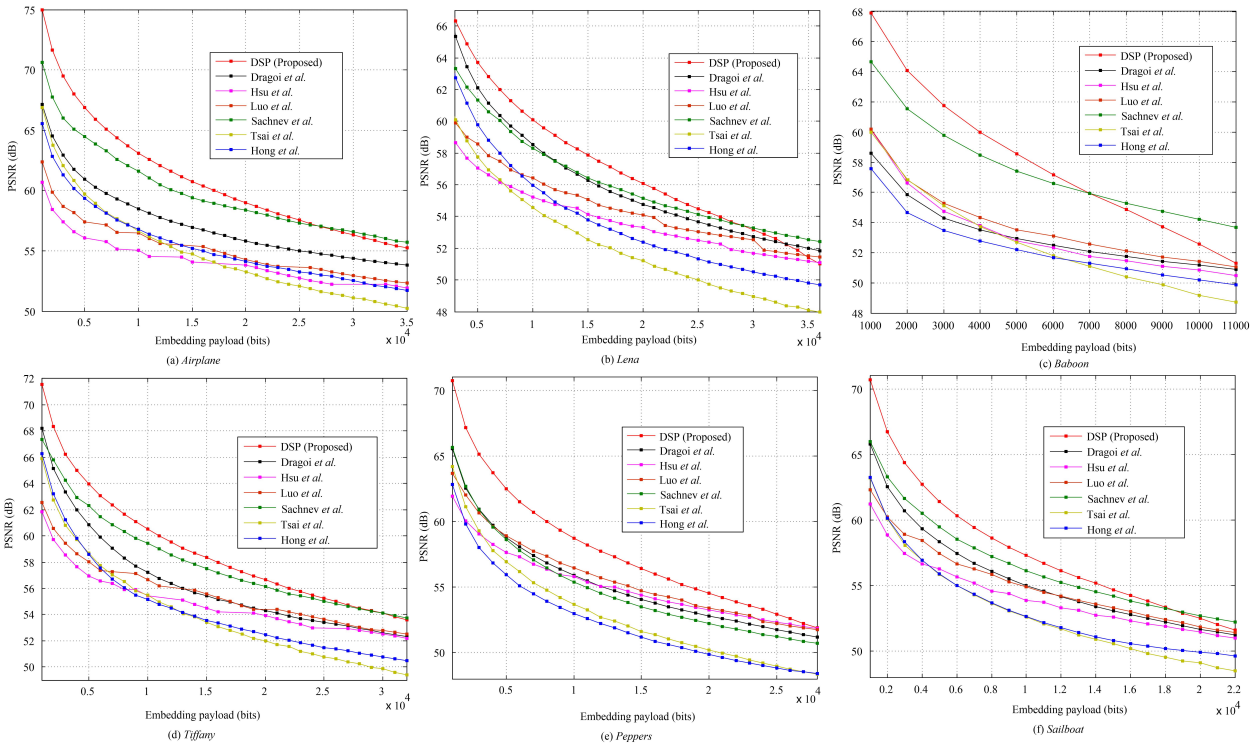


Figure 6. The payload-distortion performance comparison between the state-of-the-art methods of Tsai et al. [8], Sachnev et al. [10], Hong et al. [11], Luo et al. [12], Hsu et al. [23], Dragoi et al. [24], and the proposed method (DSP).

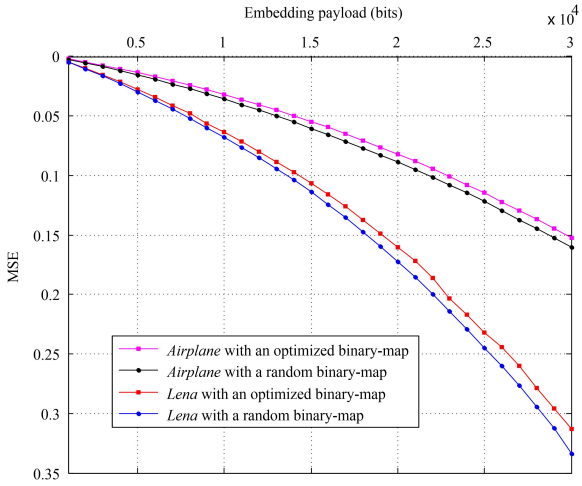


Figure 7. The MSE comparison between using a random binary-map (namely, using only $\mathcal{S}_0$ for prediction) and using an optimized binary-map (namely, using $\mathcal{S}_0 \cup \mathcal{S}_1$ for prediction) for the *Lena* and *Airplane* image.

4.2.2. Evaluation on Special Images

Unlike the traditional methods that often embed secret data according to a specified order or fully controlled by the local-complexity, the proposed DSP-based method first pre-selects relatively complex pixels out for pixel prediction and smooth pixels for data hiding. To further show its efficiency, we provide some experimental results on special images. Figure 8 shows three test images. Figure 9 shows the generated binary-maps due to the proposed binary-map generation procedure. It can be observed that the proposed method indeed can well-capture the smooth regions out for data embedding.

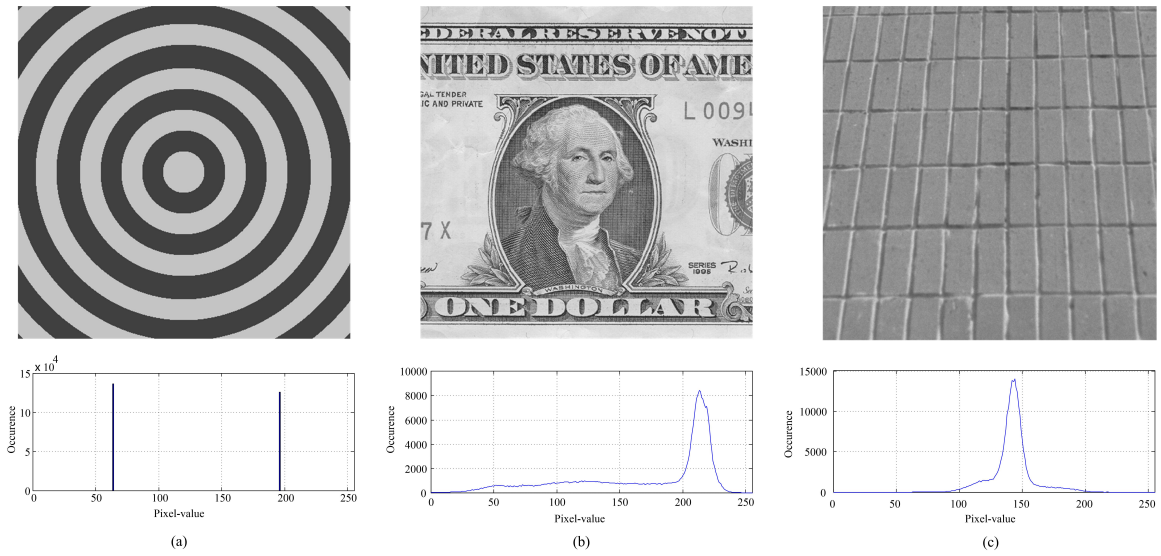


Figure 8. Three special test images and their histograms (all grayscaled with a size of 512 × 512): (**a**) *Circle*, (**b**) *Dollar*, (**c**) *Wall*.

(a) (b) (c)

Figure 9. The corresponding binary-maps ($r = 60\%$) due to proposed binary-map generation procedure: (**a**) *Circle*, (**b**) *Dollar*, (**c**) *Wall*.

We also use the special images to compare the performance between related works and the proposed method. As shown in Figure 10, the proposed method shows superior performance compared to the related works. It is worth noting that the PSNR values shown in Figure 10 of the proposed method for the *Circle* image are optimal from the viewpoint of "uniform-embedding". For example, for a payload of 1.5×10^4 bits, the PSNR value is 63.60 dB, while the theoretical uniform-embedding bound is $10 \times \log_{10} \frac{255^2}{0.5 \times 15000/512/512}$ = 63.57 dB. The reason is that the proposed binary-map generation procedure can select the circle-edges of the *Circle* image out such that the pixels within a connected area are all with the same value. It results in that all the prediction values are all the same as the original ones. Figure 11 shows the generated PEHs due to the different r for the *Circle* image. It can be observed that both the two PEHs are very sharp. Moreover, Figure 11a indicates that all the PEs of the pixels to be embedded are all with a value of zero, meaning that, for an embeddable payload, the proposed method can achieve the corresponding theoretical bound. Our experiments have shown that, even for a payload of 1.79×10^5 bits, the PSNR of the proposed method for the *Circle* image is 52.41 dB, which is rather close to the theoretical bound $10 \times \log_{10} \frac{255^2}{0.5 \times 179000/512/512}$ = 52.80 dB.

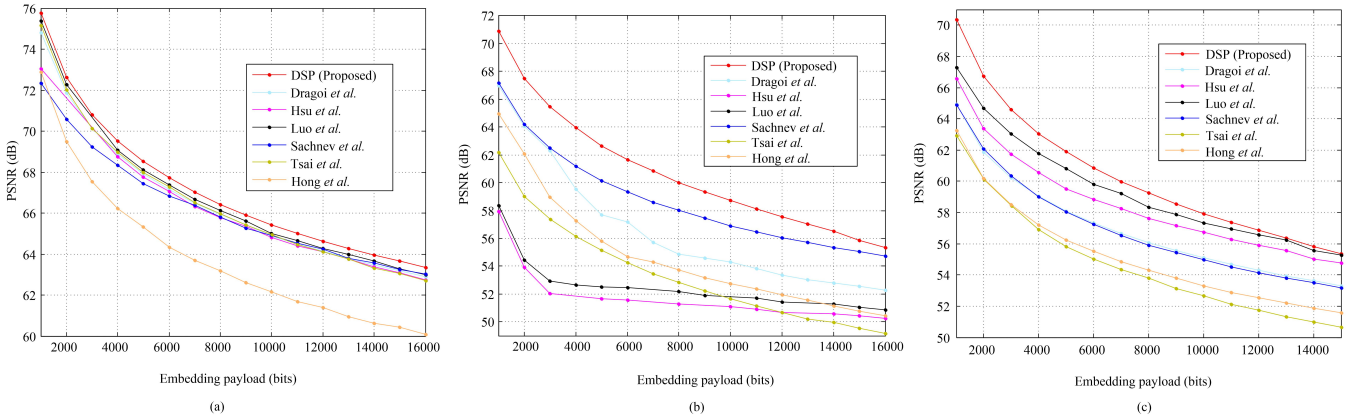


Figure 10. The payload-distortion performance comparison between the state-of-the-art methods of Tsai et al. [8], Sachnev et al. [10], Hong et al. [11], Luo et al. [12], Hsu et al. [23], Dragoi et al. [24], and the proposed method (DSP): (**a**) *Circle*, (**b**) *Dollar*, and (**c**) *Wall*.

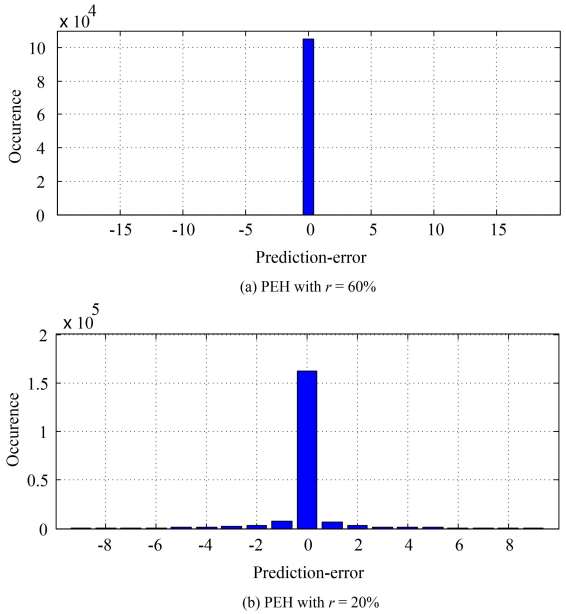


Figure 11. The generated PEH due to the different r for the *Circle* image.

5. Conclusions and Discussion

In this paper, we present a framework for prediction-based RDH technologies by revisiting a part of reported works. The proposed framework divides an RDH system at the data hider side into four parts so that one can design or improve an RDH system easily. We propose to use a key to generate a binary-map to improve the security, which is required in practice. Since the binary-map is always changing due to the key and the pixel prediction relies on the used binary-map, we propose to use a dynamic predictor for prediction. We also introduce a fast and efficient optimization algorithm, which can be equipped into the design of RDH or the existing works, to find the suitable HS parameters. Two novel RDH algorithms based on the proposed framework are also presented. Experimental results have shown that both the two novel RDH algorithms outperform a part of state-of-the-art works in terms of payload-distortion performance. Moreover, the proposed DSP-based algorithm can even achieve the theoretical bound of the uniform-embedding on special images. In the future, based on the proposed framework, we will focus on designing new RDH algorithms and also on improving the payload-distortion performance and the security of the existing works.

Author Contributions: Conceptualization, L.Z.; methodology, L.Z. and H.W.; software, L.Z. and H.W.; validation, H.H. and H.W.; supervision, H.W.; project administration, H.H. and H.W.; funding acquisition, L.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), and the Science and Technology Foundation of Guangdong Province (Grant No. 2021A0101180005).

Data Availability Statement: Not applicable.

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

References

1. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
2. Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
3. Wu, H.T.; Wu, Y.; Guan, Z.H.; Cheung, Y.M. Lossless contrast enhancement of color images with reversible data hiding. *Entropy* **2019**, *21*, 910. [CrossRef]

4. Lu, T.C.; Yang, P.C.; Jana, B. Improving the reversible LSB matching scheme based on the likelihood re-encoding strategy. *Entropy* **2021**, *23*, 577. [CrossRef] [PubMed]
5. Fridrich, J.; Goljan, M.; Du, R. Invertible authentication. In Proceedings of the SPIE Security Watermarking Multimed Contents, San Jose, CA, USA, 20–26 January 2001; pp. 197–208.
6. Celik, M.U.; Sharma, G.; Tekalp, A.M. Lossless watermarking for image authentication: A new framework and an implementation. *IEEE Trans. Image Process.* **2006**, *15*, 1042–1049. [CrossRef] [PubMed]
7. Dragoi, I.; Coltuc, D. Local-prediction-based difference expansion reversible watermarking. *IEEE Trans. Image Process.* **2014**, *23*, 1779–1790. [CrossRef] [PubMed]
8. Tsai, P.; Hu, Y.; Yeh, H. Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Process.* **2009**, *89*, 1129–1143. [CrossRef]
9. Thodi, D.M.; Rodríguez, J.J. Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Process.* **2007**, *16*, 721–730. [CrossRef] [PubMed]
10. Sachnev, V.; Kim, H.J.; Nam, J.; Suresh, S.; Shi, Y.Q. Reversible watermarking algorithm using sorting and prediction. *IEEE Trans. Circuits Syst. Video Technol.* **2009**, *19*, 989–999. [CrossRef]
11. Hong, W.; Chen, T.S.; Shiu, C.W. Reversible data hiding for high quality images using modification of prediction errors. *J. Syst. Softw.* **2009**, *82*, 1833–1842. [CrossRef]
12. Luo, L.; Chen, Z.; Chen, M.; Zeng, X.; Xiong, Z. Reversible image watermarking using interpolation technique. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 187–193.
13. Li, X.; Yang, B.; Zeng, T. Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. *IEEE Trans. Image Process.* **2011**, *20*, 3524–3533. [PubMed]
14. Wu, H.Z.; Wang, H.X.; Shi, Y.Q. PPE-Based Reversible Data Hiding. In Proceedings of the ACM Workshop Inf. Hiding Multimed. Security (Two-Page Summary, On-Going Work), Vigo Galicia, Spain, 20–22 June 2016; pp. 187–188.
15. Wu, H.Z.; Wang, H.X.; Shi, Y.Q. 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.
16. Yang, C.; Tsai, M. Improving histogram-based reversible data hiding by interleaving prediction. *IET Image Process.* **2010**, *4*, 223–234. [CrossRef]
17. Li, X.; Li, B.; Yang, B.; Zeng, T. General framework to histogram-shifting-based reversible data hiding. *IEEE Trans. Image Process.* **2013**, *22*, 2181–2191. [CrossRef] [PubMed]
18. Li, X.; Zhang, W.; Gui, X.; Yang, B. Efficient reversible data hiding based on multiple histogram modification. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2016–2027.
19. Wu, H.Z.; Wang, W.; Dong, J.; Wang, H.X. Ensemble reversible data hiding. In Proceedings of the International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 2676–2681.
20. Ou, B.; Li, X.; Zhao, Y.; Ni, R.; Shi, Y. Pairwise prediction-error expansion for efficient reversible data hiding. *IEEE Trans. Image Process.* **2013**, *22*, 5010–5021. [CrossRef]
21. Pevny, T.; Bas, P.; Fridrich, J. Steganalysis by subtractive pixel adjacency matrix. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 215–224. [CrossRef]
22. Wu, H.Z.; Wang, H.X.; Shi, Y.Q. Prediction-error of prediction error (PPE)-based reversible data hiding. *arXiv* **2016**, arXiv:1604.04984.
23. Hsu, F.; Wu, M.; Wang, S. Reversible data hiding using side-match predictions on steganographic images. *Multimed. Tools Appl.* **2013**, *67*, 571–591. [CrossRef]
24. Dragoi, I.; Coltuc, D.; Caciula, I. Gradient based prediction for reversible watermarking by difference expansion. In Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security, Salzburg, Austria, 11–13 June 2014; pp. 35–40.