applied sciences

MDPI

Article

Enhanced Dual Reversible Data Hiding Using Combined Approaches

Cheonshik Kim [1,*], Ching-Nung Yang [2] and Lu Leng [3,*]

[1] Department of Computer Engineering, Sejong University, Seoul 05006, Republic of Korea
[2] Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien 97401, Taiwan; cnyang@gms.ndhu.edu.tw
[3] School of Software, Nanchang Hangkong University, Nanchang 330063, China
* Correspondence: mipsan@sejong.ac.kr (C.K.); leng@nchu.edu.cn (L.L.)

Abstract: This paper proposes a reversible data hiding technique based on two cover images. The proposed method enhances performance by utilizing Hamming coding (HC), arithmetic coding (AC), and an improved Exploiting Modification Direction (EMD) technique. Since AC provides lossless compression for binary data, it is widely used in image compression and helps maximize the efficiency of data transmission and storage. The EMD technique is recognized as an efficient data hiding method. However, it has a significant limitation: it does not allow for the restoration of the original cover image after data extraction. Additionally, EMD has a data hiding capacity limit of approximately 1.2 bpp. To address these limitations, an improved reversible data hiding technique is proposed. In this study, HC and AC are integrated with an improved EMD technique to enhance data hiding performance, achieving higher embedding capacity while ensuring the complete restoration of the original cover image. In the proposed method, Hamming coding is applied for data encoding and arithmetic coding is used for compression to increase efficiency. The compressed data are then embedded using the improved EMD technique, enabling the receiver to fully restore the original cover image. Experimental results demonstrate that the proposed method achieves an average PSNR of 66 dB and a data embedding capacity of 1.5 bpp, proving to be a promising approach for secure and efficient data hiding applications.

Keywords: data hiding (DH); reversible DH (RDH); dual RDH; Hamming coding; arithmetic coding; exploiting modification direction (EMD)

check for updates

Academic Editor: David Megías

Received: 26 January 2025
Revised: 3 March 2025
Accepted: 12 March 2025
Published: 17 March 2025

Citation: Kim, C.; Yang, C.-N.; Leng, L. Enhanced Dual Reversible Data Hiding Using Combined Approaches. *Appl. Sci.* **2025**, *15*, 3279. https://doi.org/10.3390/app15063279

1. Introduction

Multimedia communication and security technologies are important issues in both academia and industry. The most widely used method for information security is encryption, which encrypts information so that only authorized recipients can decrypt it. However, encrypted images have the problem of potentially attracting the attention of attackers. On the other hand, data hiding (DH) [1–5] is a method of covertly concealing secret data in multimedia media, with the advantage that attackers cannot easily determine the existence of the data. Due to these characteristics, DH is advantageous for covertly transmitting small amounts of confidential information to the receiving end and is therefore being extensively researched.

DH techniques can be broadly divided into non-reversible data hiding (Non-RDH) and reversible data hiding (RDH). In Non-RDH, the cover image cannot be restored to its original state after extracting the hidden information. On the other hand, RDH [6–13] allows for the perfect restoration of the original cover image after extracting the hidden data, making it ideal for applications requiring high fidelity such as medical images, military maps, and art preservation. Over the years, various RDH techniques have been developed to achieve an optimal trade-off between data embedding capacity and visual quality.

A variety of RDH techniques, including difference expansion and histogram shifting, have been developed to balance embedding capacity and image quality. Notable methods include difference expansion (DE), proposed by Tian [8,9], Histogram Shifting (HS), introduced by Ni et al. [10], and Prediction Error Expansion (PEE) [13]. These techniques have improved the performance of RDH by enhancing capacity and image quality. Tian (2003) [8] introduced the difference expansion (DE) technique, which uses the difference between two adjacent pixels to hide data. The DE technique provides high embedding capacity and has the advantage of being able to restore the original image without loss. However, this method may have limited embedding capacity and may result in some degradation of image quality.

Ni et al. (2006) [10] proposed the Histogram Shifting (HS) technique, introducing a method of embedding data by shifting the positions of the peak and zero points in the image histogram. This method has relatively low embedding capacity but shows excellent performance in maintaining image quality. The HS technique became the foundation for RDH research, and various modified techniques were subsequently developed. Alattar (2004) [9] proposed a reversible watermarking technique that generalized the DE technique and applied it to quads. This method provides high embedding capacity and can restore the original image without loss. However, it requires complex calculations and may have limitations in real-time applications.

Li et al. (2013) [14] proposed a high-fidelity RDH technique that combines Pixel Value Ordering (PVO) and prediction-error expansion (PEE). This method simultaneously provides high insertion capacity and excellent image quality, making it effective for use in various application areas. By combining PVO and PEE techniques, they reduced distortions that could occur during the data insertion process and improved the ability to restore the original image.

Dual reversible data hiding (RDH) [15–27] is a technique that simultaneously achieves high embedding capacity and good image quality. This method incorporates the concept of Secret Sharing (SS) [28,29] to ensure data security and recoverability. SS is a technique that divides secret data into multiple pieces and distributes them, allowing the original data to be reconstructed only when pieces meeting specific conditions are gathered. Prominent examples of this method include Shamir's SS [28]. The main purpose of SS is to enhance data security and ensure that even if secret data are leaked, the original data cannot be reconstructed from only a portion of the pieces.

Chang et al. (2007) [15] proposed a dual image-based RDH method using the Exploiting Modification Direction (EMD) [30] technique. The EMD technique optimizes image quality by inserting data using the modification direction of each pixel, which allows for achieving high PSNR (Peak Signal-to-Noise Ratio) values. However, there is a potential issue of quality differences occurring between the two generated images. Kim et al. (2024) [27] proposed a method to optimize the LSB matching revisited [31] technique and the dual RDH method based on EMD by Chang et al. (2006) [15]. Lee and Huang (2013) [16] also converted secret data into base 5 secret symbols. Every two continuous secret symbols were referred as a group for embedding in an identical pixel pair. The reversibility was fulfilled by the orientation combination of pixel pairs in dual stego-images.

Lu et al. (2015) [17] introduced a dual image-based RDH technique using LSB (least significant bit) matching. This technique uses seven rules to conceal pixel modifications and maintains high hiding capacity and image quality. By generating two stego images, it can effectively hide secret data and restore the original image after data extraction.

Huynh et al. (2015) [18] introduced a dual image-based RDH scheme using a Sudoku reference matrix. Huynh et al. took the initiative to take a grayscale image as the secret data. Each pixel in the secret image was first converted into three base 9 numeral symbols, and then three symbols were concealed in a cover pixel pair to generate two stego pixel pairs with the help of the Sudoku reference matrix. Though a higher embedding ratio (ER) of 2 bpp could be achieved in their scheme, the image quality of each stego image was only about 36 dB.

Yao et al. (2017) [19] improved the dual image RDH technique by using a strategy of selecting movable pixel coordinates with minimal distortion. This method generates two stego images and maintains high image quality by minimizing distortion during data insertion. This approach is particularly useful when dealing with sensitive data such as medical images and can achieve high PSNR values. Chen and Guo (2020) [22] used the same idea as Lee and Huang [16], but their embedding ratio reaches to 1.14 bits per pixel.

Matrix encoding (ME), discovered by Crandall [32] in 1998, demonstrates excellent performance in terms of embedding efficiency for DH. Westfeld first implemented ME in his F5 steganography [33] algorithm. To enhance the F5 algorithm's efficiency for lengthy messages, Fridrich et al. (2001) [34] proposed random linear codes. Bose–Chaudhuri–Hocquenghem (BCH) codes [35] were later applied to achieve an optimal balance between embedding complexity and efficiency.

Zhang et al. (2006) [36] proposed an efficient data hiding method utilizing the error detection and correction capabilities of Hamming code (HC). Later, in 2009, they introduced an advanced data hiding technique called Hamming + 1 DH (H1DH) [37], which maximized performance. This technique extends the fundamental principles of HC to allow for the hiding of larger amounts of data. H1DH offers superior embedding efficiency compared to matrix encoding based on Hamming coding. While matrix encoding is efficient, it falls short of theoretically achievable bounds for payloads exceeding 67% of embedding capacity. LSB substitution-based DH schemes can achieve high embedding capacity.

In this study, we propose a method to maximize dual RDH performance by integrating HC with arithmetic coding (AC) compression techniques [38–40]. Our approach utilizes two duplicate grayscale images derived from the original cover image. While HC is effective in data encoding and error correction, it has inherent limitations in embedding large amounts of data and lacks robustness as a standalone RDH solution. To address these issues, we introduce a RDH framework that compresses the HC syndrome representation using AC and subsequently embeds the compressed data through an enhanced EMD method. This combined approach ensures high embedding capacity, efficient data compression, and the reliable recovery of the original cover image.

The key contributions of this work are as follows: First, we present a novel reversible data hiding technique that seamlessly integrates Hamming coding, AC, and an enhanced EMD method to address the limitations of existing RDH approaches. Second, we achieve a significant improvement in DH capacity, reaching an embedding rate of 1.5 bpp, while maintaining exceptional image quality, with an average PSNR of 66 dB. Third, leveraging the unique strengths of HC and AC, our method ensures efficient data compression and accurate recovery, making it a promising solution for secure and high-capacity data hiding applications. Fourth, this work demonstrates the practical potential of combining multiple techniques to overcome the individual limitations of existing methods, paving the way for further advancements in RDH technologies.

This study focuses on a data hiding technique for bitmap (pixel-based) images, aiming to achieve a high embedding rate while preserving image quality. The proposed method utilizes pixel value-based modification techniques, making it difficult to apply directly to geometrically represented data such as vector graphics (e.g., SVG, AI). Furthermore, the proposed approach is designed for two-dimensional (2D) image data and is not directly applicable to one-dimensional (1D) data, such as audio or sensor data. However, with certain modifications, it may be possible to extend the method to audio data by applying a sample block-based approach or to sensor data by utilizing specific frequency bands. These potential extensions will be considered in future research.

The rest of this paper is organized as follows: Section 2 provides a review of Hamming code as a single-error correcting code, AC, and EMD method. Section 3 introduces the proposed dual RDH method, which utilizes Hamming code, AC, and an enhanced EMD method with dual cover images. Section 4 presents the experimental results, and Section 5 concludes the paper with a summary and discussion.

2. Preliminaries

2.1. Hamming Codes

We describe the fundamental concepts of Hamming code (HC) [41,42] as applied to cover coding. HC(n,k) is a linear, perfect, single-error-correcting code characterized by a minimum Hamming distance of 3. For an integer $m \geq 2$, the codeword length is given by $n = 2^m - 1$, while the message length is $k = 2^m - m - 1$. However, HC cannot guarantee error correction if multiple errors occur within a single codeword. The parity check matrix $\mathcal{H}$ for HC is defined in Equation (1). Let $\mathcal{H}$ be a $k \times n$ matrix such that $G \cdot \mathcal{H}^T = [0]_{(n-k)\times k}$, where G is the generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$\mathcal{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{1}$$

Let $y \in \mathbb{F}_2^n$ $(n = 2^k - 1)$ represent a codeword generated from an information word $x \in \mathbb{F}_2^{n-k}$ using a $(n-k) \times n$ generator matrix G, such that $y = x \cdot G$. If a single-bit error occurs in the codeword, the error pattern can be determined to correctly decode the received data. Assuming the received codeword is y' and the error pattern is $e = (y \oplus y')$, the position of the error can be identified using Equation (2).

$$y' \cdot \mathcal{H}^T = (e \oplus y) \cdot \mathcal{H}^T = e \cdot \mathcal{H}^T + y \cdot \mathcal{H}^T \tag{2}$$

where $y \cdot \mathcal{H}^T = 0$ by definition. Therefore, the equation simplifies to $y' \cdot \mathcal{H}^T = e \cdot \mathcal{H}^T$. For instance, if an error occurs in the 6th bit of the codeword $\hat{y}$, the error pattern is $e = (e_1, e_2, \ldots, e_7) = (0\ 0\ 0\ 0\ 0\ 1\ 0)$. In this case, the computed syndrome $\eta = (110)$ precisely identifies the error. Thus, the error location within the codeword can be accurately determined using the syndrome.

To hide message bits, the cover image is divided into N/n subsets, each containing n pixels, where n is the length of the codeword. Errors may occur at any of the n positions within each codeword. The HC decoding process is outlined as follows:

Step 1: Calculate the syndrome $\eta(y') = b(y') \cdot \mathcal{H}^T$, where $b(y'_i) = (y'_i \bmod 2)$ and y' is the received codeword.

Step 2: Identify the error pattern e from the computed syndrome $\eta(y')$.

Step 3: Correct the error and decode: modify the cover object so that $y = (y' \oplus e)$, and then recover the original message $x = y \cdot G^{-1}$.

This structured approach ensures accurate decoding and error correction, even in the presence of single-bit errors.

2.2. Arithmetic Coding

AC [38–40] is a sophisticated method of entropy encoding used in lossless data compression. It was first introduced by Jorma Rissanen in the late 1970s and has since been refined and adopted in various applications, particularly where high compression efficiency is required. Unlike traditional Huffman coding [43], which assigns fixed-length codes to symbols, arithmetic coding represents an entire message as a single number, a fraction between 0 and 1. This technique allows for more efficient data representation, especially in cases where symbol probabilities vary significantly.

The primary idea behind arithmetic coding is to represent a sequence of symbols by a subinterval within the range [0, 1). This interval narrows as more symbols are processed, with the width of each subinterval proportional to the probability of the corresponding symbol.

The processes of encoding and decoding for data compression are described in Algorithms 1 and 2, and they are explained step by step as follows:

Step 1: The initial interval is set to $[0, 1)$. The variables `low` and `high` are initialized to 0 and 1, respectively.

Step 2: Encoding

- **a:** For each symbol in the input data, the interval is updated based on the cumulative distribution function (CDF) of the symbol.
- **b:** The `range` is calculated as the difference between `high` and `low`.
- **c:** The `high` value is updated to `low` + `range` × `CDF(symbol)`.
- **d:** The `low` value is updated to `low` + `range` × `CDF(previous_symbol)`.

Step 3: Final Encoded Value

- **a:** After processing all symbols, the midpoint of the final interval is selected as the encoded value.

Step 4: Decoding

- **a:** Decoding is performed by reversing the encoding process. Starting from the encoded value, the interval is iteratively narrowed down.
- **b:** For each symbol, the `range` is calculated, and the value is adjusted to locate the corresponding symbol index using the CDF.
- **c:** The decoded symbol is appended to the output sequence, and the interval is updated accordingly.

Algorithm 1 Arithmetic encoding

```
Require: Data sequence $D = \{d_1, d_2, \ldots, d_n\}$, symbol probabilities $\mathcal{P}$
Ensure: Encoded value $E$
1: Initialize $low \leftarrow 0$
2: Initialize $high \leftarrow 1$
3: Initialize $range \leftarrow high - low$
4: for each symbol $d_i$ in $D$ do
5:     $range \leftarrow high - low$
6:     $high \leftarrow low + range \times \text{CDF}(d_i)$
7:     $low \leftarrow low + range \times \text{CDF}(d_{i-1})$
8: end for
9: $E \leftarrow (low + high)/2$
10: return $E$
```

Algorithm 2 Arithmetic decoding

```
Require: Encoded value $E$, symbol probabilities $\mathcal{P}$, data length $n$
Ensure: Decoded sequence $\{D' = d'_1, d'_2, \ldots, d'_n\}$
 1: Initialize $low \leftarrow 0$
 2: Initialize $high \leftarrow 1$
 3: Initialize $range \leftarrow high - low$
 4: Initialize $encoded\_value \leftarrow E$
 5: for $i \leftarrow 1$ to $n$ do
 6:     $range \leftarrow high - low$
 7:     $value \leftarrow (encoded\_value - low)/range$
 8:     Find the largest $j$ such that $\text{CDF}(d_j) \leq value$
 9:     $d'_i \leftarrow d_j$
10:     $high \leftarrow low + range \times \text{CDF}(d_j)$
11:     $low \leftarrow low + range \times \text{CDF}(d_{j-1})$
12: end for
13: return $S'$
```

2.3. Exploiting Modification Direction Method

The EMD method [30] is a technique that utilizes the modification direction of pixels to conceal data within digital images. EMD employs a $(2n+1)$-ary numeral system to embed secret information into n cover pixels, ensuring that, at most, one pixel is adjusted by ± 1. This approach enhances steganographic efficiency while minimizing distortion during embedding, thereby improving security. Compared to conventional methods, EMD offers higher embedding efficiency and enables the easy retrieval of embedded secret data.

The core concept of this method is to group n pixels together and represent secret information through a specific combination of these pixels. In each group, one pixel at most is incremented or decremented by 1 to embed data, effectively minimizing image distortion.

For instance, using two pixels, data can be embedded based on a quinary numeral system $(2n+1 = 5)$. First, the pixel group's value is calculated using the formula $f(x_1, x_2) = (x_1 \times 1 + x_2 \times 2) \mod 5$. If the computed value does not match the secret value to be embedded, one of the pixels is adjusted to achieve the desired result. This process ensures minimal changes to pixel values, enhancing both the security and recoverability of the embedded data. For example, if the pixel values are $[120, 130]$, the calculation is $(120 \times 1 + 130 \times 2) \mod 5 = 380 \mod 5 = 0$. If the intended secret value is 1, the first pixel value is increased by 1, adjusting the pixel group to $[121, 130]$, thereby achieving the desired embedding result.

3. Proposed Method

In this paper, we propose a dual RDH method based on grayscale images (Figure 1). This method aims to achieve DH while considering efficiency and security aspects. To this end, copies I_1 and I_2 of the original image O are used as cover images. Given arbitrary-length binary data to be hidden, $m = \{b \in \{0,1\}^n\}$, a block $(H \times H)$ is then read from the two cover images and assigned to B_i^1 and B_i^2.

From each block, one pixel is sequentially taken from both blocks to form a codeword P, after which the HC is calculated to obtain the syndrome S. The result of XORing this syndrome S with the binary data m is stored sequentially in M. This process is applied to all pixels in the block, and then AC compression is performed to generate $D_{comp} = \{b_1, b_2, \ldots, b_k\}$. The compressed binary bits D_{comp} are sequentially converted into quinary numbers, and the extended EMD method is applied to each pixel of the two blocks to hide the data. The dual RDH method proposed in this paper introduces an efficient data hiding scheme that ensures security, enhances data hiding performance, and maintains high image quality.

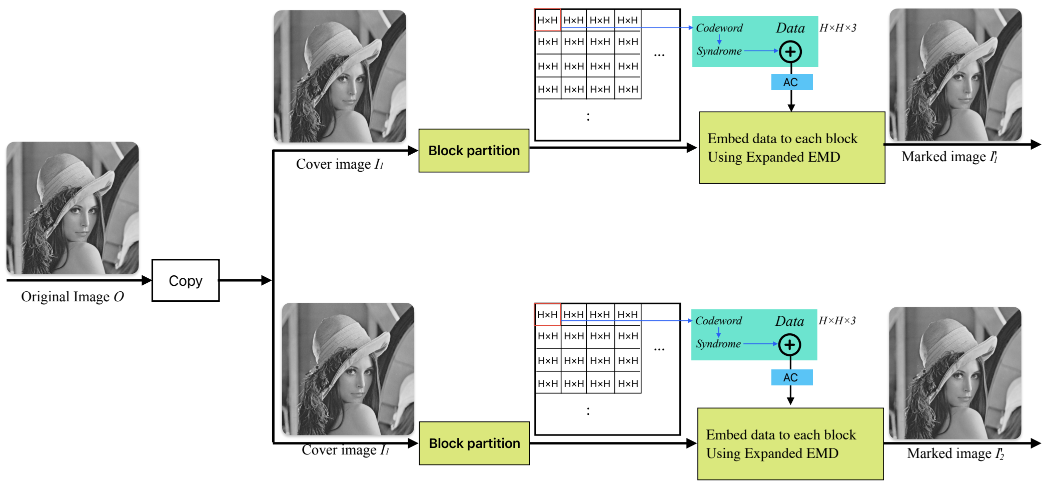


Figure 1. Framework of the proposed scheme.

3.1. Embedding Procedure

As preparation for DH, the original image O is used as the first cover image I_1, and a copy of the original image is prepared as the second cover image I_2. By applying our proposed DH method to these two cover images, two marked images are produced as the final result. The data embedding procedure involves encoding the secret data using Hamming code, arithmetic coding, and extended EMD. The embedding process is outlined as follows:

Input: Two cover images I_1 and I_2 and data bits $m = \{b \in \{0,1\}^n\}$.
Output: Two marked images I_1' and I_2'.
Step 1: Divide the grayscale images I_1 and I_2 into non-overlapping blocks of size 16×16.
Step 2: Read corresponding blocks B_i^1 and B_i^2 from I_1 and I_2. The blocks B_i^1 and B_i^2 are composed of pixels $\{x_1^1, x_2^1, \ldots, x_{16\times16}^1\}$ and $\{x_1^2, x_2^2, \ldots, x_{16\times16}^2\}$. The total length of binary bits to be hidden in each block is $16 \times 16 \times 3$.
Step 3: Sequentially read pixels (x_j^1 and x_j^2) from the two blocks B_i^1 and B_i^2 read from the cover image. To embed the binary message m, combine the upper 4 bits and upper 3 bits of the pixel x_j to generate a codeword P (Equation (3)). For the codeword P, calculate the Hamming syndrome S using Equation (4). Perform an XOR operation between the syndrome S and the corresponding secret bit m_k: $S' = S \oplus (m_k, m_{k+1}, m_{k+2})$. Append the resulting syndrome S' sequentially to the array M: $M(k : k+2) = S'$. Continue the process for all pixels in both blocks until each pixel pair is processed.

$$\begin{cases} x_j^1 = \sum_{k=0}^{7} b_{j,k}^1 \cdot 2^k, \quad b_{j,k}^1 = \left\lfloor \dfrac{x_j^1}{2^k} \right\rfloor \bmod 2 \\ x_j^2 = \sum_{k=0}^{7} b_{j,k}^2 \cdot 2^k, \quad b_{j,k}^2 = \left\lfloor \dfrac{x_j^2}{2^k} \right\rfloor \bmod 2 \\ \text{Codeword: } P = \{b_{j,7}^1, b_{j,6}^1, b_{j,5}^1, b_{j,4}^1, b_{j,7}^2, b_{j,6}^2, b_{j,5}^2\} \end{cases} \tag{3}$$

$$S = H \cdot P^T \tag{4}$$

Step 4: If all pixels in the block have been processed, move to step 5. Otherwise, repeat step 3 for the remaining pixels.
Step 5: Compress the encoded message M using the arithmetic encoding function $A(\cdot)$ defined in Algorithm 1. The compressed data D_{comp} and its length L are given by $D_{comp} = A(M)$ and $L = len(D_{comp})$.

Step 6: The compressed data D_{comp} are hidden in the LSB (least significant bit) of the pixels within the blocks B_i^1 and B_i^2. D_{comp} is embedded into pixel pairs (x_k^1 and x_k^2) using Equation (5). The calculation results in pos, which is used to adjust the pixel values according to the rules in Equation (6) to hide the data. This process is repeated for all pixels in the block. The data length L is stored in the pixel positions $\{x_{250}, x_{251}, \ldots, x_{256}\}$ within the blocks (B_i^1) and (B_i^2) using the methods described in Equations (5) and (6). The length information L is an essential payload for the data extraction process.

$$\forall k \in \{1, \ldots, L/2\}$$
$$\begin{cases} f = (x_k^1 + x_k^2 \times 2) \bmod 5 \\ d_{decimal}(k) = D_{comp}^{2r} \cdot 2^1 + D_{comp}^{2r+1} \cdot 2^0 \\ pos = (d_{decimal}(k) - f) \bmod 5 \\ \text{if } (pos < 0)\ pos = 5 - abs(pos) \end{cases} \tag{5}$$

$$\begin{cases} \text{If } pos = 0, & \text{No change needed} \\ \text{If } pos = 1, & x_j'^1 \leftarrow x_j^1 + 1 \\ \text{If } pos = 2, & x_j'^2 \leftarrow x_j^2 + 1 \\ \text{If } pos = 3, & x_j'^1 \leftarrow x_j^1 - 1, \quad x_j'^2 \leftarrow x_j^2 + 2 \\ \text{If } pos = 4, & x_j'^1 \leftarrow x_j^1 + 1, \quad x_j'^2 \leftarrow x_j^2 - 1 \end{cases} \tag{6}$$

Step 7: The block with hidden data is swapped with the corresponding block at its designated location according to Equation (7). This process is repeatedly applied to all blocks. If there are remaining blocks to process, return to step 2. If no more blocks remain to be processed, this procedure is complete.

$$I_i^1 = \begin{bmatrix} B_1'^1 & B_2'^1 & B_3'^1 & \cdots & B_{16}'^1 \\ B_{17}'^1 & B_{18}'^1 & B_{19}'^1 & \cdots & B_{32}'^1 \\ B_{33}'^1 & B_{34}'^1 & B_{35}'^1 & \cdots & B_{48}'^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{241}'^1 & B_{242}'^1 & B_{243}'^1 & \cdots & B_{256}'^1 \end{bmatrix} \quad I_i^2 = \begin{bmatrix} B_1'^2 & B_2'^2 & B_3'^2 & \cdots & B_{16}'^2 \\ B_{17}'^2 & B_{18}'^2 & B_{19}'^2 & \cdots & B_{32}'^2 \\ B_{33}'^2 & B_{34}'^2 & B_{35}'^2 & \cdots & B_{48}'^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{241}'^2 & B_{242}'^2 & B_{243}'^2 & \cdots & B_{256}'^2 \end{bmatrix} \tag{7}$$

Finally, two marked images I_1' and I_2' containing the embedded data are created, and these two marked images are sent to different recipients.

Algorithm 3 represents the detailed description provided above in the form of a pseudocode algorithm. This algorithm performs the process of hiding secret data in two cover images, namely I_1 and I_2. The images are divided into 16×16 non-overlapping blocks, and the upper bits of pixels in each block are extracted to generate codewords. The Hamming code is used to compute the syndrome, which is then XORed with the secret data. The encoded data are compressed using arithmetic coding, and the least significant bits (LSBs) of the pixels are modified to embed the data. The length of the embedded data is stored in each block, and after processing all blocks, the final marked images are generated. As a result, the modified images can be sent to different recipients.

Algorithm 3 Data hiding algorithm

```
Require: Cover images $I_1, I_2$, secret data bits $m = \{b \in \{0,1\}^n\}$
Ensure: Marked images $I_1', I_2'$
1: Initialization:
2: Divide $I_1$ and $I_2$ into non-overlapping blocks of size $16 \times 16$
3: for each block pair $(B_i^1, B_i^2)$ do
```

Algorithm 3 *Cont.*

```
4:      Encoding:
5:      for each pixel pair $(x_k^1, x_k^2)$ do
6:          Extract upper bits to form codeword $P$
7:          Compute syndrome $S \leftarrow H \cdot P^T$
8:          $S' \leftarrow S \oplus (m_k, m_{k+1}, m_{k+2})$
9:          Store $S'$ in encoded message $M$
10:     end for
11:     Compression:
12:     $D_{comp} \leftarrow A(M)$                                   ▷ Apply arithmetic encoding
13:     $L \leftarrow \text{len}(D_{comp})$
14:     Embedding (RDH EMD) method:
15:     for $k = 1$ to $L/2$ do
16:         Compute $f \leftarrow (x_k^1 + x_k^2 \cdot 2) \bmod 5$
17:         $d_{decimal}(k) = D_{comp}^{2r} \cdot 2^1 + D_{comp}^{2r+1} \cdot 2^0$        ▷ binary to decimal value
18:         Compute $pos = (d_{decimal}(k) - f) \bmod 5$
19:         if $pos < 0$ then
20:             $pos \leftarrow 5 - \text{abs}(pos)$
21:         end if
22:         if $pos == 1$ then
23:             $x_k^1 \leftarrow x_k^1 + 1$
24:         else if $pos == 2$ then
25:             $x_k^2 \leftarrow x_k^2 + 1$
26:         else if $pos == 3$ then
27:             $x_k^1 \leftarrow x_k^1 - 1$
28:             $x_k^2 \leftarrow x_k^2 + 2$
29:         else if $pos == 4$ then
30:             $x_k^1 \leftarrow x_k^1 + 1$
31:             $x_k^2 \leftarrow x_k^2 - 1$
32:         end if
33:     end for
34:     Store length $L$ in pixels $\{x_{250}, x_{251}, \ldots, x_{256}\}$ of $B_i^1$
35: end for
36: Finalization:
37: Replace original blocks in $I_1, I_2$ with modified blocks $B_i^1, B_i^2$
38: return Marked images $I_1', I_2'$
```

3.2. Data Extraction and Image Recovering Procedure

The receiver, who has the two marked images I_1' and I_2' sent by the sender, can extract the data from these images using the data extraction and image restoration algorithm that is shared between the sender and receiver. The process of data extraction and cover image restoration is the reverse of the data hiding process, and the detailed procedure is as follows:

Input: Two marked images I_1' and I_2'.

Output: One original image O and hidden bits, m.

Step 1: The marked images I_1' and I_2' are divided into non-overlapping blocks of size 16×16.

Step 2: Read blocks B_i^1 and B_i^2 from the marked image I_1' and I_2'.

Step 3: To extract the length of the binary data embedded in a block, Equations (5) and (6) are applied to the pixel positions $(x_{250}, \ldots, x_{256})$ of blocks B_i^1 and B_i^2 to extract binary digits, which are then converted to a decimal number and assigned to the variable L. Given the length L in the block, the following process is performed: Equation (5) is applied to the pixels x_j^1 and x_j^2 of blocks B_i^1 and B_i^2, respectively, to calculate the value f. The extracted data are assigned to M, where $M(k) = d2b(f)$. Here, the $d2b(\cdot)$ is a function that converts a decimal integer f to a binary integer.

Step 4: The calculation result of Equation (8) restores the original pixel $x_j = p_j$. The pixel x_j is replaced by the corresponding pixel at the same position as the two blocks. This process is repeated for a length of $L/2$ to recover the block. By replacing the pixels x_j^1 and x_j^2 in blocks B_i^1 and B_i^2 that constitute the block, the restored block (as described in Equation (9) can be obtained. Each block is completed through the process explained in Equation (10) along with the original cover image.

$$p = \left\lfloor \frac{x^1{}_j \times x^2{}_j}{2} \right\rfloor \tag{8}$$

$$B_i^1 = B_2^2 = \begin{bmatrix} p_1 & p_2 & p_3 & \cdots & p_{16} \\ p_{17} & p_{18} & p_{19} & \cdots & p_{32} \\ p_{33} & p_{34} & p_{35} & \cdots & p_{48} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{241} & p_{242} & p_{243} & \cdots & p_{256} \end{bmatrix} \tag{9}$$

$$\begin{cases} I_1 = \bigcup_i B_i^1 \\ I_2 = \bigcup_i B_i^2 \end{cases} \tag{10}$$

Step 5: After restoring M, decompress it using the AC (Algorithm 2) function $A_D(\cdot)$; that is, $M_{decompressed} = A_D(M)$.

Step 6: Two pixels ($x_j^1 = (b_1^1 b_2^1 \ldots b_8^1)$ and $x_j^2 = (b_1^2 b_2^2 \ldots b_8^2)$) are read from each of the blocks B_i^1 and B_i^2 and assigned to the variables x_j^1 and x_j^2, respectively. The codeword $P_j = \{b_{j,7}^1, b_{j,6}^1, b_{j,5}^1, b_{j,4}^1, b_{j,7}^2, b_{j,6}^2, b_{j,5}^2\}$ is constructed by extracting the top 4 bits of x_j^1 and the top 3 bits of x_j^2, as described in Equation (3). The value of $M_{k:k+2}$ is first converted to a decimal number and stored in the variable l, which identifies the error position in the codeword P_j. The corresponding bit in P_j is then flipped to correct the error, and Equation (11) is applied to compute the syndrome. Here, $b2d(\cdot)$ denotes a function that converts binary numbers to a decimal format.

$$\begin{cases} l = b2d(M_{j:j+2}) = M_{j+2} \cdot 2^2 + M_{j+1} \cdot 2^1 + M_j \cdot 2^0 \\ P^{(flipped)} = P \oplus e_j, \text{ where } e_l = [0, 0, \ldots 1_l, \ldots, 0] \\ m_{k:k+2} = H(P \oplus e_j) \end{cases} \tag{11}$$

Step 7: Repeat step 6 for the length of $L/2$. Afterwards, if there is another block to process, return to step 2. The process terminates when there are no more blocks left to process.

Step 8: The recovered cover images I_1 and I_2 were restored to be identical to the original image O, and the message m was successfully recovered.

Algorithm 4 presents the process described above in the form of a simplified pseudocode algorithm. This algorithm extracts the hidden data from the two marked images I_1' and I_2' and restores the original image. The images are divided into 16×16 blocks, and the length of the hidden data L is extracted from specific pixel positions. The pixel values are used to recover the hidden data, and arithmetic decoding is applied to decompress the extracted data. The recovered data are then used to reconstruct codewords by combining the upper bits of the pixels, and the error positions are corrected to retrieve the hidden secret data. Finally, all the blocks are combined to restore the original image, and the extracted data are the output.

Algorithm 4 Data extraction and image recovery algorithm

```
Require: Marked images $I'_1, I'_2$
Ensure: Recovered original image $O$ and hidden data $m$
 1: Divide $I'_1$ and $I'_2$ into non-overlapping blocks of size $16 \times 16$
 2: for each block pair $(B_i^1, B_i^2)$ do
 3:     Read pixel pairs $\{x_1^1, x_2^1, \ldots, x_{256}^1\}$ and $\{x_1^2, x_2^2, \ldots, x_{256}^2\}$
 4:     Read binary values from pixel positions $(x_{250}, \ldots, x_{256})$ and convert to decimal length $L$
 5:     for $k = 1$ to $L/2$ do                                  ▷ Extract data using RDH EMD
 6:         Compute $f \leftarrow (x_k^1 + x_k^2 \cdot 2) \bmod 5$
 7:         $f_{binary} = \left(\left\lfloor \frac{f}{2} \right\rfloor \cdot 2\right) + (f \mod 2)$
 8:         Assign $M(k) \leftarrow f_{binary}$
 9:     end for
10:     for $j = 1$ to $L/2$ do                                  ▷ Restore original pixel $x_j$
11:         Recover pixel value $p_j \leftarrow \left\lfloor \frac{x_j^1 \times x_j^2}{2} \right\rfloor$
12:         Replace corresponding pixels in blocks $B_i^1, B_i^2$
13:     end for
14:     Combine blocks to form images $I_1, I_2$
15: end for
16: Decompress extracted data using arithmetic decoding: $M_{decompressed} \leftarrow A_D(M)$
17: for each pixel pair $(x_j^1, x_j^2)$ do
18:     Construct codeword $P_j = \{b_{j,7}^1, b_{j,6}^1, b_{j,5}^1, b_{j,4}^1, b_{j,7}^2, b_{j,6}^2, b_{j,5}^2\}$
19:     Convert $M_{j:j+2}$ to decimal $l \leftarrow b2d(M_{decompressed}^{j:j+2})$
20:     Flip bit at position $l$ in $P_j$
21:     Extract hidden bits $m_{k:k+2} \leftarrow H(P_j \oplus e_l)$
22: end for
23: Combine all blocks to restore the original cover image $O$
24: return $O, m$
```

3.3. Underflow and Overflow Management

In the embedding phase of our steganographic method, the careful handling of pixel values at the boundaries of the allowable range (0–255) is essential to prevent underflow and overflow. These safeguards ensure the accuracy of data extraction and maintain the integrity of pixel values throughout the process.

Underflow occurs when a pixel value drops below 0, while overflow results from a pixel value exceeding 255. Such conditions can lead to errors during data extraction, as pixel values must remain within the defined range to ensure proper interpretation. To address this, pixel pairs at the extremes—(0, 0) and (255, 255)—are excluded from both data embedding and extraction. By avoiding these edge cases, the system minimizes the risk of misinterpretation and ensures that the hidden data and pixel values are preserved accurately.

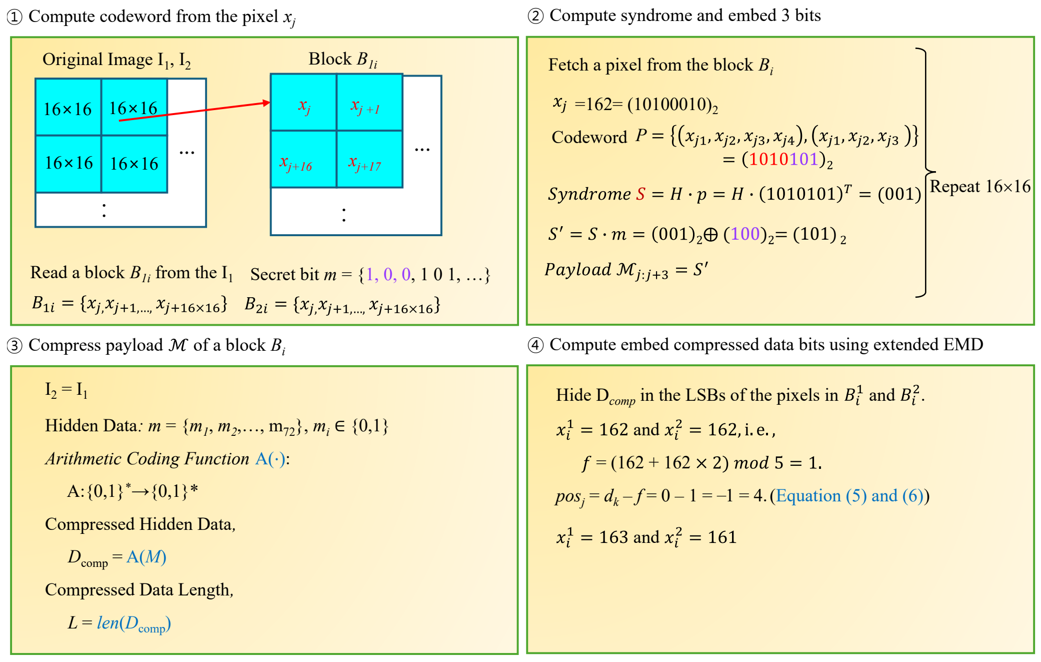
Initially, the complete set of pixel pairs available for data hiding is defined as (0, 0), (1, 1), . . . , (255, 255), where each pair contains identical pixel values. However, to prevent potential issues arising from boundary cases, such as calculated pixel values exceeding the valid range, the pairs (0, 0) and (255, 255) are omitted. The revised set for data embedding is therefore limited to (1, 1), (2, 2), . . . , (254, 254). This adjustment reduces ambiguity and errors, ensuring a smoother and more reliable embedding process.

For example, when using a pixel pair (1, 1) with a data range of 0 to 4, the possible resulting pixel pairs include (1, 2), (1, 0), (0, 1), (1, 1), and (2, 1). By carefully selecting and managing these pairs, decoding challenges are effectively mitigated, resulting in a more secure and robust steganographic procedure. Through the exclusion of extreme pixel values and the precise adjustment of pixel pairs, this method significantly enhances the reliability, efficiency, and security of the steganography system.

3.4. Examples

The process of hiding data for the block given in Equation (12) is briefly explained with an example (Figure 2). Assume the secret data $m = \{1\,0\,0\,1\,0\,0\,\ldots\,0\,1\,1\}$.

$$B_i^1 = B_2^2 = \begin{bmatrix} 162 & 162 & 161 & \cdots & 155 \\ 162 & 162 & 161 & \cdots & 155 \\ 162 & 162 & 161 & \cdots & 155 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 159 & 159 & 160 & \cdots & 157 \end{bmatrix} \tag{12}$$



① Compute codeword from the pixel x_j
Original Image I₁, I₂
16×16
Block B_{1i}
x_j
x_{j+1}
x_{j+16}
x_{j+17}
Read a block B_{1i} from the I₁
Secret bit $m = \{1, 0, 0, 1\ 0\ 1, \ldots\}$
$B_{1i} = \{x_j, x_{j+1}, \ldots, x_{j+16\times16}\}$
$B_{2i} = \{x_j, x_{j+1}, \ldots, x_{j+16\times16}\}$
② Compute syndrome and embed 3 bits
Fetch a pixel from the block B_i
$x_j = 162 = (10100010)_2$
Codeword $P = \{(x_{j1}, x_{j2}, x_{j3}, x_{j4}), (x_{j1}, x_{j2}, x_{j3})\} = (1010101)_2$
Syndrome $S = H \cdot p = H \cdot (1010101)^T = (001)$
$S' = S \cdot m = (001)_2 \oplus (100)_2 = (101)_2$
Payload $\mathcal{M}_{j:j+3} = S'$
Repeat 16×16
③ Compress payload $\mathcal{M}$ of a block B_i
I₂ = I₁
Hidden Data: $m = \{m_1, m_2, \ldots, m_{72}\}$, $m_i \in \{0,1\}$
Arithmetic Coding Function A(·):
A: $\{0,1\}^* \to \{0,1\}^*$
Compressed Hidden Data,
$D_{comp} = A(M)$
Compressed Data Length,
$L = len(D_{comp})$
④ Compute embed compressed data bits using extended EMD
Hide D_{comp} in the LSBs of the pixels in B_i^1 and B_i^2.
$x_i^1 = 162$ and $x_i^2 = 162$, i.e.,
$f = (162 + 162 \times 2)$ mod $5 = 1$.
$pos_j = d_k - f = 0 - 1 = -1 = 4$. (Equation (5) and (6))
$x_i^1 = 163$ and $x_i^2 = 161$

Figure 2. Example illustration of the proposed scheme.

(1) The values x_j^1 and x_j^2 read from the first block (Figure 2②) are both assigned as 162. Using these values, the codeword P is constructed first. The binary representation of pixels x_j^1 and x_j^2 is (10100010), and the codeword P is generated by concatenating the upper 4 bits of x_j^1 with the upper 3 bits of x_j^2; that is, $P = (b_{j,7}^1, b_{j,6}^1, b_{j,5}^1, b_{j,4}^1, b_{j,7}^2, b_{j,6}^2, b_{j,5}^2)$ = $(\underline{1010101})$.

(2) For the constructed codeword P, calculate the syndrome of the HC, i.e., $S = H \cdot P^T = (0\ 0\ 1)$. Here, H is the HC checker matrix. Perform an XOR operation between this syndrome S and the secret bit $m = (1\ 0\ 0)$ to calculate a new syndrome, i.e., $S' = S \oplus m_k = (0\,0\,1) \oplus (1\,0\,0) = (1\,0\,1)$. Add the syndrome S' to the array M; that is, $M(j : j+2) = (101), j = j+3$.

(3) After completing steps (1) and (2) for all the pixels in the block, the array M will contain data hiding information $16 \times 16 \times 3$ in size. Since the amount of information is too large to be hidden in the LSB without degrading the image quality, this data are compressed using arithmetic coding; that is, $D_{comp} = A(M)$, and the compressed data are hidden in D_{comp}, with their length being L (Figure 2③). For example, if $M = \{101\ldots100\}$, the compression result is $D_{comp} = \{\underline{0011}\ldots1100\}$ and $L = 64$.

(4) Hide D_{comp} in the LSBs of the pixels that make up blocks B_i^1 and B_i^2. In this process, data hiding is performed by applying Equations (5) and (6). Specifically, since $x_i^1 = 162$ and $x_i^2 = 162$, we have $f = (162 + 162 \times 2) \bmod 5 = 1$. Therefore,

$pos_j = d_k - f = 0 - 1 = -1 = 4$. According to Equation (6), the two pixels x_i^1 and x_i^2 become 163 and 161, respectively (Figure 2④).

(5) Data hiding is completed by applying the process iteratively to the pixels within the block for the length L. Subsequently, the next blocks are read in order, and the same process is repeated. As a result, two cover images, I_1' and I_2', are produced.

The data extraction process can be performed by reversing these steps.

3.5. Improvements over Existing Approaches

In this study, we analyzed various data hiding techniques proposed in previous research and introduced a new method that improves upon them by combining Hamming coding, arithmetic coding, and an enhanced EMD technique. Traditional methods typically use LSB (least significant bit) embedding or conventional EMD techniques, which often suffer from limited embedding capacity and difficulties in maintaining high image quality.

Our proposed method enhances data recovery accuracy through Hamming coding and improves compression efficiency using arithmetic coding, allowing for higher data embedding while achieving superior PSNR and SSIM values. While previous studies reported an average PSNR of 45–55 dB, our approach achieves 6–68 dB, minimizing visual quality degradation. Additionally, previous research often limited the embedding rate (ER) to 1.0 bpp or lower, whereas our method allows for up to 1.5 bpp, maximizing embedding efficiency.

Furthermore, many existing methods do not consider error correction, whereas our approach incorporates Hamming coding to enhance data stability and robustness. Additionally, previous techniques often involve high computational complexity, making large-scale data processing inefficient, while our method optimizes encoding and decoding operations using arithmetic coding, thereby improving computational efficiency.

In summary, the proposed method demonstrates significant improvements over previous studies in terms of higher embedding capacity, enhanced image quality (PSNR and SSIM), increased data stability (error correction), and improved computational efficiency (optimized coding operations).

4. Experimental Results

In this section, we evaluate the performance of the proposed dual image-based RDH method by analyzing three key aspects: data hiding capacity, the quality of the marked images, and the ability to restore the original images. To this end, we selected nine test images from the USC-SIPI (University of Southern California—Signal and Image Processing Institute) standard dataset [44], which is widely used as a benchmark in steganography and image processing research. As illustrated in Figure 3, the experiments utilized eight 512×512 images, including Baboon, Barbara, Boat, Goldhill, Airplane, Peppers, Tiffany, Zelda, and Elaine.

The selected images were chosen to reflect diverse texture and frequency characteristics. For example, Baboon, Lena, and Barbara contain rich high-frequency components and complex textures, while Peppers, Goldhill, and Airplane include relatively smooth regions and well-defined edges. This ensures that the proposed data hiding method is evaluated across various image patterns to verify its consistency in performance. Additionally, we used grayscale images instead of color (RGB) images to assess the proposed method, as it operates on bitmap (pixel-based) images. Grayscale images simplify the analysis by using a single intensity channel, whereas color images require independent data embedding for each RGB channel, introducing additional complexity. Therefore, the evaluation focuses on the fundamental performance of the method, while its applicability to color images requires further analysis.

Although the study is based on nine test images, the selected dataset consists of a representative mix of natural images, structured patterns, and complex textures, aligning with the standard benchmarks widely used in steganography and data hiding research. Thus, the experimental results are not limited to specific images but reflect the method's performance across diverse image types.

Furthermore, the proposed method operates on a block-wise basis, making it independent of image resolution. While the experiments were conducted using 512×512 standard-resolution images, the method can be applied to high-resolution images such as 5 MP medical images, 50 MP smartphone images, 100 MP pathological images, and 1000 MP satellite images in the same manner.

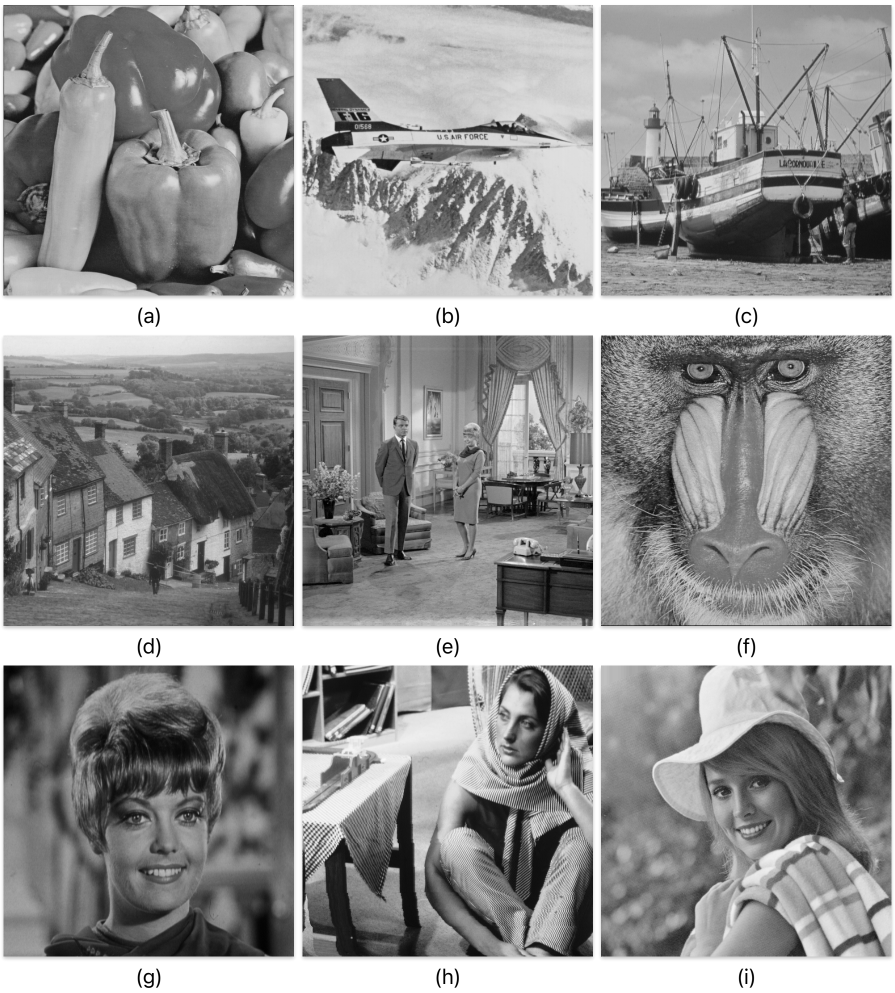


Figure 3. Test images: (**a**) Pepper, (**b**) Airplane, (**c**) Boat, (**d**) Goldhill, (**e**) Couple, (**f**) Baboon, (**g**) Zelda, (**h**) Barbara, and (**i**) Elaine (512×512).

Embedding capacity is typically expressed as the number of bits per pixel (bpp) in an image, denoted as 1 bpp. The term bpp indicates how many bits can be embedded within each pixel. An embedding capacity of 1 bpp signifies that one bit can be hidden in each pixel, making the total amount of data that can be concealed directly proportional to the total number of pixels in the image. For instance, if the data are embedded at 1 bpp in an image with a resolution of 512×512, the total embedding capacity is 262,144 bits (equivalent to 32,768 bytes or 32 KB). The embedding capacity is generally computed using the following formula (Equation (13)):

$$ER(bits) = \frac{\text{embedded bits}}{\text{total number of pixels in the images}} bpp \tag{13}$$

Using the above formula, we can calculate how much data can be hidden depending on the size of the image and the bpp value. For example, if data hiding is performed at 1 bpp and the image resolution is $m \times n$, the embedding capacity will be $m \times n$ bits.

The quality of the marked image, in which data are hidden, is very important in data hiding. In data hiding, the PSNR [27] evaluation of the marked image is a key metric that measures the similarity between the original image and the image after the data have been hidden. The PSNR is used to quantitatively assess the visual quality degradation that occurs in the marked image. The PSNR is generally expressed by the following formula (Equation (14)):

$$PSNR(I, I') = 10 \cdot log_{10}\left(\frac{MAX^2}{\sqrt{MSE}}\right), \tag{14}$$

Here, MAX is the maximum pixel value of the image, which is typically 255 for an 8-bit image. The MSE (Mean Squared Error) represents the average squared error between the original image and the marked image, and it is calculated as follows:

$$MSE = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}[I_{i,j} - I_{i,j}]^2, \tag{15}$$

The symbols $I_{i,j}$ and $I'_{i,j}$ stand for the pixel values of the original grayscale image and the marked image at the respective position, respectively, and $N \times N$ is the width and height of the original image.

In data hiding techniques, the PSNR is used to evaluate the difference between the original image and the marked image; the higher the PSNR value, the less quality degradation and the more similar the marked image is to the original. Generally, a PSNR value of 30 dB or higher is considered to be nearly imperceptible to the human eye. When comparing the performance of various data hiding techniques, the PSNR serves as an important criterion. A higher PSNR value indicates that the data have been effectively hidden while maintaining the image quality. The PSNR objectively measures how well a data hiding technique preserves image quality.

Another performance measure is the SSIM (Structural Similarity Index Measure), a formula (Equation (16)) that measures the similarity between the original image and the marked image. The SSIM is a metric used to evaluate the structural similarity between images, precisely assessing the visual quality between the original image and the one after data embedding. The SSIM mimics the way the human visual system perceives images, providing a more intuitive quality assessment. It calculates the similarity by combining three components: luminance, contrast, and structure. The formula for the SSIM is as follows:

$$SSIM(I, I') = \frac{(2\mu_I\mu_{I'} + C_1)(2\sigma_{II'} + C_2)}{(\mu_I^2 + \mu_{I'}^2 + C_1)(\sigma_I^2 + \sigma_{I'}^2 + C_2)} \tag{16}$$

- μ_I and $\mu_{I'}$ are the mean values of the original image I and the marked image I', respectively.
- σ_I and $\sigma_{I'}$ represent the standard deviations of I and I', respectively.
- σ_I and $\sigma_{I'}$ is the covariance between I and I'.
- C_1 and C_2 are small constants added to ensure numerical stability in the calculations.

The SSIM value is expressed as a number between -1 and 1, where a value closer to 1 indicates a higher structural similarity between two images. By utilizing the SSIM in DH techniques, even subtle structural differences between the original image and the marked image can be accurately assessed. The SSIM is useful for evaluating how well the primary structure and detailed information of an image are preserved during the DH process. In particular, it enables the assessment of high-frequency components (such as edges or textures in the image), allowing for the verification of whether there are significant visual changes after data embedding.

In this study, SSIM, ER, and RS values were measured to evaluate the performance of data hiding. For the SSIM, it is generally considered that values above 0.98 indicate that differences are nearly imperceptible to the human eye. In our study, we achieved an excellent average SSIM value of 0.999 or higher. Therefore, setting the minimum acceptable SSIM threshold at 0.98 or higher is appropriate.

Additionally, an ER of 1.0 bpp or higher is typically regarded as a high-capacity data hiding technique. In our study, we achieved a high ER value of 1.5 bpp. Hence, the minimum acceptable ER threshold in this study is set at 1.0 bpp or higher. Finally, RS analysis can be used to detect traces of data hiding. Previous studies have evaluated RS values within the 0.3 to 0.4 range as a safe level. In our study, the average RS value remained within 0.32 to 0.38, and thus, setting the minimum acceptable threshold at 0.35 or lower is considered appropriate.

Table 1 presents the experimental results evaluating the data hiding performance of the proposed scheme. After embedding 30,000 bits into each image, the PSNR and SSIM values were measured. The results show that the PSNR values of the marked images ranged from 65.33 dB to 68.24 dB, with an average PSNR of approximately 66.7770 dB, demonstrating a robust embedding process with minimal degradation in image quality.

Table 1. Comparisons of the PSNR and SSIM for dual marked images.

Original Image	Measurement	PSNR (I_1)	PSNR (I_2)	Avg. PSNR
Pepper	PSNR	65.2328	68.1832	66.708
	SSIM	0.9999	0.9998	0.99985
Airplane	PSNR	65.2337	68.1782	66.7059
	SSIM	0.9998	0.9997	0.99975
Boat	PSNR	65.2859	68.2662	66.7760
	SSIM	0.9998	0.9996	0.9997
Goldhill	PSNR	65.2116	68.2101	66.71085
	SSIM	0.9999	0.9997	0.9998
Couple	PSNR	65.5378	68.2524	66.8951
	SSIM	0.9999	0.9998	0.99985
Baboon	PSNR	65.2781	68.3875	66.79535
	SSIM	1	1	1
Zelda	PSNR	65.3939	68.3875	66.8907
	SSIM	0.9999	0.9997	0.9998
Barbara	PSNR	65.2593	68.2101	66.7347
	SSIM	0.9999	0.9998	0.99985
Elaine	PSNR	65.5988	68.2221	66.9104
	SSIM	0.9999	0.9998	0.99985
Average	PSNR	65.3368	68.2469	66.7770
	SSIM	0.9999	0.9998	0.99985

The SSIM values consistently remained close to 1 across all test images, confirming that the structural integrity of the images was largely preserved even after data embedding. For the Peppers image, the PSNR values for I_1 and I_2 were measured at 65.2328 dB and 68.1832 dB, respectively, with an average PSNR of 66.7080 dB. These results confirm that the proposed method effectively maintains high image quality despite data hiding. The Elaine image achieved the highest average PSNR of 66.9104 dB, demonstrating strong resilience to distortion even after dual embedding.

In particular, the Baboon image exhibited an SSIM value of 1.0, which can be attributed to its high-frequency components and complex texture characteristics. Since the SSIM measures structural similarity, images with intricate textures tend to experience minimal perceptual changes even with pixel modifications, causing the SSIM value to approach 1.0. Compared to other test images (e.g., Peppers, Goldhill), the Baboon image contains fine patterns and strong variations, making structural changes less detectable. In contrast, images with simpler backgrounds are more susceptible to noticeable differences caused by minor pixel modifications. Therefore, analyzing the SSIM in conjunction with the PSNR provides a more comprehensive evaluation rather than relying solely on the SSIM. This study accounts for both metrics to ensure a balanced assessment of image quality after data embedding.

Overall, the proposed dual RDH scheme demonstrates excellent performance in terms of both the PSNR and SSIM. With an average PSNR of approximately 66.77 dB and SSIM values consistently close to 0.99985, the method achieves RDH with minimal perceptual degradation. These findings validate the effectiveness of the proposed approach as a high-quality and robust RDH solution for applications requiring reliable image reconstruction after data extraction.

The ER measures the amount of data that can be embedded relative to the image size, with higher values indicating better performance while maintaining image quality.

The proposed scheme consistently achieved an embedding ratio of 1.5 across all tested images, which surpasses the performance of existing methods. In contrast, methods by Chang et al. [15], Lee et al. [16], Chen et al. [22], and Liu et al. [20] achieved ER values ranging from 1.0 to 1.14, highlighting the superiority of the proposed approach in terms of data hiding capacity. The consistent results across all images demonstrate the robustness and adaptability of the proposed method to various image types.

The results emphasize that the proposed method significantly increases the embedding capacity while maintaining image fidelity. This improvement suggests that the technique can be effectively applied in applications where high-capacity and reversible data hiding are crucial, such as secure communication, medical imaging, and digital forensics. Additionally, the stability of the achieved ER values across different images further validates the reliability of the proposed approach in practical scenarios.

Overall, the experimental findings confirm that the proposed RDH method offers a competitive advantage over existing approaches [15,16,20,22], making it a promising solution for high-capacity data embedding applications.

Table 2 presents the average PSNR values of nine test images for each shadow generated using the comparative methods when embedding 5000, 10,000, and 20,000 bits of data. The results demonstrate that the proposed scheme achieves a superior PSNR in most shadow images compared to other methods. To ensure a fair evaluation, only dual image RDH methods were selected for comparison. The studies included meet the following criteria:

- Dual image RDH methods were chosen to maintain consistency with the proposed approach and avoid unfair comparisons with Single-Image RDH techniques.
- Well-cited and representative studies in the field were selected to highlight key advancements and distinctions.
- Studies providing PSNR, SSIM, and ER metrics were included to enable an objective performance comparison.

Table 2. Comparative PSNR of each shadow of different schemes with different ER.

Methods	**5000 bits**		**10,000 bits**		**20,000 bits**	
	Shadow$_1$	**Shadow$_2$**	**Shadow$_1$**	**Shadow$_2$**	**Shadow$_1$**	**Shadow$_2$**
Chang et al. [15]	65.63	65.6	62.55	62.54	59.56	59.55
Lee et al. [16]	69.72	69.67	66.72	66.69	63.71	63.7
Huynh et al. [18]	59.35	59.31	56.33	56.36	53.34	53.31
Liu et al. [20]	71.90	65.96	68.93	62.91	65.93	59.87
Lu et al. [23]	61.25	61.36	54.96	55.11	50.39	50.47
Lee et al. [25]	65.33	65.38	62.32	62.49	59.31	59.43
Zhang et al. [26]	71.25	71.33	68.25	68.31	65.35	65.42
Proposed Scheme	78.35	81.17	75.67	78.62	72.57	75.52

Among the existing approaches, Liu et al. [20] and Zhang et al. [26] achieved the highest PSNR values, apart from our proposed method. Zhang et al.'s [26] approach improves the data hiding efficiency by embedding and extracting data using a vector coordinate transformation (TOC) approach without requiring a codebook. This method minimizes unnecessary additional data storage and reduces distortions, leading to higher PSNR values. As a result, it effectively balances high embedding capacity and image quality preservation. On the other hand, the methods proposed by Lu et al. [23] and Lee et al. [25] achieved maximum PSNR values of 61.25 dB and 65.33 dB, respectively, which are relatively lower than those of the methods compared in Table 2.

While our method already demonstrates significant PSNR improvements over previous approaches, further PSNR optimization remains a crucial research objective. Although a PSNR above 30 dB is generally considered to be visually imperceptible, optimizing the PSNR is still essential for data hiding research. A higher PSNR is particularly critical in applications requiring high-fidelity image quality, such as medical imaging, forensic analysis, and security-sensitive environments. Additionally, recent advancements in AI-based steganalysis suggest that higher PSNR values can reduce the likelihood of detection, making data hiding techniques more robust against automated detection systems.

Furthermore, increasing the PSNR while maintaining a high ER presents a technical challenge, and our method demonstrates that high-quality data hiding can be achieved without compromising capacity. Therefore, PSNR optimization in this study is not merely aimed at surpassing 30 dB but rather at maximizing the applicability of the proposed method across different domains while enhancing its resistance to detection.

Regular–Singular (RS) analysis [45] is a statistical method used to evaluate the detectability or quality changes resulting from data hiding in images. This analysis classifies pixel blocks in an image as either "Regular" or "Singular" to assess the impact of data hiding on the statistical properties and block patterns of the image. Typically, in the natural state of an image, "Regular" blocks make up the majority, while the proportion of "Singular" blocks gradually increases as data hiding is performed.

RS analysis proceeds in the following steps:

(1) **Block Generation:** The original image and the marked image are divided into multiple small blocks. Typically, the block size is set to $n \times n$ pixels.

(2) **Applying Masking Function:** A masking function is applied to each block to evaluate changes in the block. The masking function modifies the pixel values of the block in a specific manner. The most commonly used masking functions either increase or decrease the difference between pixel values. For example, if the pixel values of a block are $P_1, P_2, \ldots, P_n$, after applying the masking function, the block becomes $Q_1, Q_2, \ldots, Q_n$.

(3) **Calculating RS Value:** The RS value is calculated using the number of Regular and Singular blocks. This can be defined as follows:

$$RS = \frac{R_M - S_M}{R_M + S_M} \tag{17}$$

Here, R_M and S_M represent the number of Regular and Singular blocks, respectively, after the masking function is applied. The same calculation is performed when the masking function is applied in the opposite direction. When two masking functions are applied, the RS value can be obtained using the values R_{-M} and S_{-M} for each case.

(4) **Detecting Data Hiding:** The presence of data hiding is assessed by analyzing the RS value. Generally, if the RS value is close to 0, it can be determined that no data have been hidden in the image. If the RS value deviates significantly, it can be concluded that data hiding has occurred.

This method is particularly useful when compared to the statistical characteristics of the original image and can be used to detect the presence and intensity of hidden data in the image. RS analysis can identify traces of data hiding and assess the integrity of the image.

The two RS analysis results in Tables 3 and 4 were conducted on two different marked images, I_1' and I_2'. Each experiment measured RS values for various images, such as Boat, Barbara, Tiffany, Pepper, and Goldhill, using different message lengths (20,000 bits and 50,000 bits). This evaluation analyzes the differences and consistency of the RS values by examining the measured values of R_M, R_{-M}, S_M, and S_{-M} for each image.

Table 3. Comparison of RS analysis for different marked image I_1' with varying message lengths.

Images	Message (bits)	R_M	R_{-M}	S_M	S_{-M}	RS Value
Boat	20,000	19,147	24,274	9753	4626	0.3250
	50,000	19,140	24,272	9760	4628	0.3245
Barbara	20,000	19,976	26,129	8924	2771	0.3824
	50,000	19,973	26,128	8927	2772	0.3822
Tiffany	20,000	18,899	23,848	10,001	5052	0.3078
	50,000	18,899	23,842	10,001	5058	0.3078
Pepper	20,000	19,231	24,555	9669	4345	0.3308
	50,000	19,227	24,550	9673	4350	0.3305
Goldhill	20,000	19,244	24,559	9668	4341	0.3309
	50,000	19,244	24,765	9656	4135	0.3317

Table 4. Comparison of RS analysis for different marked image I_2' with varying message lengths.

Images	Message (bits)	R_M	R_{-M}	S_M	S_{-M}	RS Value
Boat	20,000	19,152	24,271	9748	4629	0.3253
	50,000	19,143	24,264	9757	4636	0.3247
Barbara	20,000	19,978	26,131	8922	2769	0.3825
	50,000	19,974	26,132	8926	2768	0.3822
Tiffany	20,000	18,899	23,851	10,001	5049	0.3078
	50,000	18,897	23,849	10,003	5051	0.3077
Pepper	20,000	19,232	24,559	9668	4341	0.3309
	50,000	19,232	24,556	9668	4344	0.3309
Goldhill	20,000	19,242	24,762	9658	4138	0.3316
	50,000	19,242	24,767	9658	4133	0.3316

In the Pepper image, the RS values for both I_1' and I_2' are similarly observed at 0.3308 (or 0.3309) for 20,000 bits and 0.3305 (or 0.3309) for 50,000 bits. This suggests that the two hiding methods exhibit almost identical statistical characteristics in the Pepper image. As the message length increases, the RS values show a slight decreasing trend, which consistently appears in both images, indicating that data hiding has minimal impact on the statistical properties of the image.

For the Goldhill image, the RS values show highly consistent results in both experiments. For a 20,000-bit message, the value is measured at 0.3309 (or 0.3316), and for a 50,000-bit message, the value remains nearly the same at 0.3317. The Goldhill image contains natural scenes with complex textures, which helps to reveal the effects of data hiding more clearly.

Overall, the two experiments yielded very similar RS values for the same image, suggesting that the two proposed data hiding methods have a consistent impact on the statistical properties of the image. The consistency of RS values for each image indicates that the data hiding methods can effectively conceal data without significantly compromising the statistical integrity of the image. Additionally, the minimal variation in RS values as the message length increases implies that the proposed methods maintain stable performance across various message lengths.

Tables 3 and 4 summarize the results under specific experimental conditions. Instead of including all nine test images, representative images were selected to effectively illustrate the key trends observed in the experiments. The selection includes images with different texture characteristics to ensure a balanced evaluation. However, all nine images were tested, and the complete results remain consistent with the trends presented in these tables.

Figure 4 shows the results of analyzing the pixel value difference (PVD) histogram to evaluate data hiding performance. The PVD histogram represents the differences between adjacent pixels in the image, playing a crucial role in assessing how the distribution of pixel values changes after data hiding.

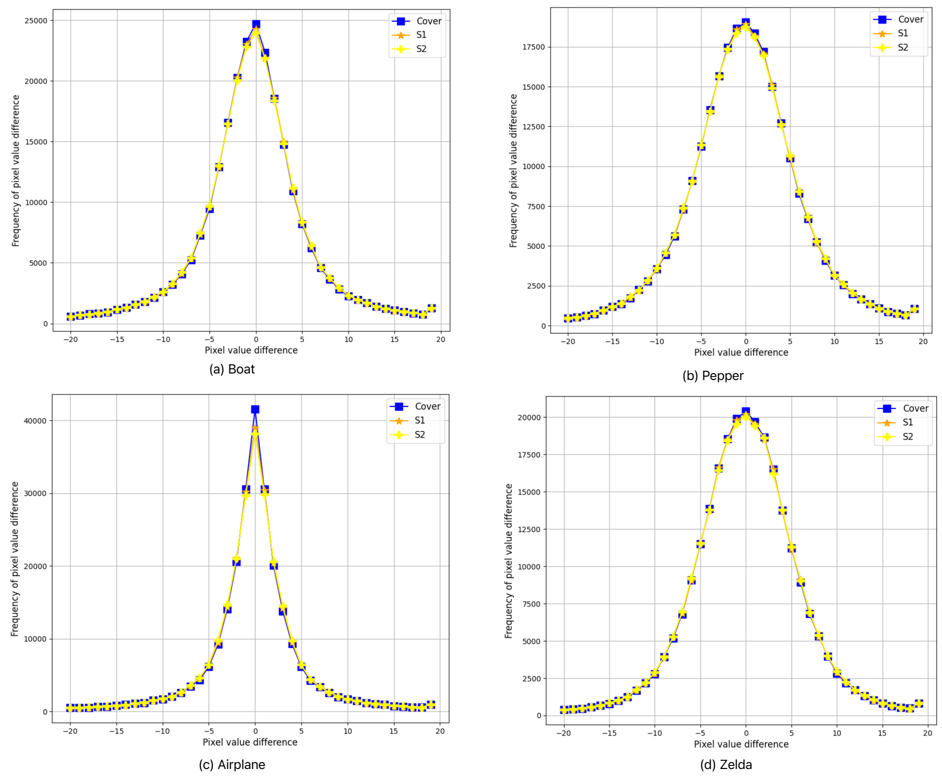


Figure 4. Four PVD histograms of the cover images and their marked images with differences between −20 and 20.

In this study, the PVD histograms of the original image and the stego image (the image with hidden data) were compared. Through histogram analysis, the impact of data hiding on pixel value differences in the image can be visually observed. When data is hidden in the image, subtle changes may occur in the shape of the histogram, which can be more pronounced in regions with small difference values. These changes indicate that some pixel values were adjusted during the data hiding process.

If the histogram does not differ significantly from the original image, it indicates that data hiding has been successfully performed without causing noticeable visual changes. Conversely, if there are significant changes in the histogram, it suggests that the data hiding process may be easily detectable.

The analysis of the four images presented in Figure 4 shows that the histograms of the original image and the stego image are almost identical. This indicates that data hiding was effectively performed, and the hidden data have minimal impact on the visual quality. Therefore, this experiment demonstrates the successful application of the PVD-based data hiding technique.

The selection of test images was carefully made to cover a diverse range of characteristics, ensuring a fair evaluation of the proposed method. The chosen images represent different levels of texture complexity and frequency components, making them well suited for testing the robustness and visual impact of the proposed reversible data hiding technique. Moreover, as in previous studies, we focused on commonly used benchmark images rather than evaluating the entire dataset, as not all images in the USC-SIPI dataset are relevant to reversible data hiding research.

5. Conclusions

This paper proposes a method to achieve optimal dual RDH performance by combining Hamming coding, AC, and an enhanced EMD technique. The proposed method utilizes two cover images to embed secret data while ensuring the complete recovery of the original images after data extraction. By employing dual images instead of a single cover image, the method overcomes the limitations of conventional RDH techniques, achieving higher embedding capacity, improved image quality, and robust security performance. Experimental results demonstrate that the proposed method achieves a data hiding capacity of 1.5 bpp and an average PSNR of 66 dB, while effectively mitigating detection through RS analysis.

Furthermore, the method is independent of image resolution due to its block-wise processing approach, making it applicable to high-resolution images (e.g., medical scans, satellite images, and smartphone photography) in the same manner as standard images. However, high-resolution scenarios introduce additional challenges, such as compression artifacts, storage constraints, and computational efficiency, which warrant further exploration.

The selection of test images was carefully designed to encompass a broad range of texture complexity and frequency components. In particular, we included high-frequency images (e.g., Baboon, Lena, and Barbara) and low-frequency images (e.g., Peppers, Goldhill, and Airplane) to evaluate the robustness of our method across diverse visual patterns. This choice aligns with standard benchmark practices in steganography and RDH research, ensuring a fair and representative assessment of the proposed approach. While a more extensive statistical analysis of image characteristics could provide additional insights, the selected dataset already includes well-established benchmark images widely used in the field. Therefore, the experimental results presented in this study effectively demonstrate the performance of the proposed method without requiring additional dataset-specific statistical evaluations.

While this study primarily focuses on imperceptibility and reversibility, it does not explicitly consider active adversarial attacks such as man-in-the-middle (MITM) attacks.

In real-world applications, an attacker gaining access to both cover images could pose a security risk. To further enhance security during data transmission, future research may explore integrating cryptographic techniques with RDH methods, such as encryption-based authentication mechanisms. This combined approach could strengthen resilience against adversarial attacks while maintaining the reversibility and efficiency of the proposed method. Additionally, future research will investigate the integration of this method into medical imaging to ensure secure and reversible data hiding while complying with healthcare standards and preserving the confidentiality of patient records.

Author Contributions: Each author discussed the details of the manuscript. C.K. designed and wrote the manuscript. C.K. implemented the proposed technique and provided the experimental results. C.-N.Y. and L.L. reviewed and revised the article. C.K. drafted and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by National Science and Technology Council (NSTC), under Grant No. 112-2221-E-259-007-MY2. (C.-N.Y.) National Natural Science Foundation of China (No. 62466038), Technology Innovation Guidance Program Project (Special Project of Technology Cooperation, Science and Technology Department of Jiangxi Province) (No. 20212BDH81003), Open Foundation of Jiangxi Provincial Key Laboratory of Image Processing and Pattern Recognition (ET202404437) (L.L.).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: We thank the anonymous reviewers for their valuable suggestions that improved the quality of this article.

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

Abbreviations

The following abbreviations are used in this manuscript:

DH	data hiding
RDH	reversible data hiding
Dual RDH	dual reversible data hiding
arithmetic coding	AC
HC	Hamming code
I_1 and I_2	cover image 1 and 2
I_1' and I_2'	marked image 1 and 2
m	$\{b \in \{0,1\}^n\}$ and data bits
$N \times N$	image size
$H \times H$	block size
B	read block—composed of pixels
x_i	pixel
P	codeword
$\mathcal{H}$	a parity check matrix
S	Hamming syndrome
$A(\cdot)$	arithmetic compression encoding
M	1-D array
D_{comp}	compressed binary bits, $\{b_1, b_2, \ldots, b_k\}$
f	EMD, $f = (x_j^1 + x_j^2 \times 2)\ mod\ 5$
pos	used to adjust the pixel values—Equations (5) and (6)

References

1. Petitcolas, F.A.; Anderson, R.J.; Kuhn, M.G. Information hiding—A survey. *Proc. IEEE* **1999**, *87*, 1062–1078. [CrossRef]
2. Provos, N.; Honeyman, P. Hide and seek: An introduction to steganography. *IEEE Secur. Priv.* **2003**, *1*, 32–44. [CrossRef]
3. Johnson, N.F.; Jajodia, S. Exploring steganography: Seeing the unseen. *Computer* **1998**, *31*, 26–34. [CrossRef]
4. Cheddad, A.; Condell, J.; Curran, K.; Kevitt, P.M. Digital image steganography: Survey and analysis of current methods. *Signal Process.* **2010**, *90*, 727–752. [CrossRef]
5. Bender, W.; Gruhl, D.; Morimote, N.; Lu, A. Techniques for data hiding. *IBM Syst. J.* **1996**, *35*, 313–336. [CrossRef]
6. Shi, Y.Q.; Li, X.; Zhang, X.; Wu, H.-T.; Ma, B. Reversible data hiding: Advances in the past two decades. *IEEE Access* **2016**, *4*, 3210–3237. [CrossRef]
7. Fridrich, J.; Goljan, M.; Du, R. Lossless data embedding—New paradigm in digital watermarking. *EURASIP J. Adv. Signal Process.* **2002**, *2002*, 986842. [CrossRef]
8. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
9. Alattar, A.M. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Process.* **2004**, *13*, 1147–1156. [CrossRef]
10. Ni, Z.; Shi, Y.-Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
11. Dragoi, I.-C.; Coltuc, D. Local-Prediction-Based Difference Expansion Reversible Watermarking. *IEEE Trans. Image Process.* **2014**, *23*, 1779–1790. [CrossRef] [PubMed]
12. Yang, C.N.;Wu, S.Y.; Chou, Y.S.; Kim, C. Enhanced stego-image quality and embedding capacity for the partial reversible data hiding scheme. *Multimed. Tools Appl.* **2019**, *78*, 18595–18616. [CrossRef]
13. Ou, B.; Zhao, Y.; Ni, R.; Shi, Y.Q. Pairwise Prediction Error Expansion for Efficient Reversible Data Hiding. *IEEE Trans. Image Process.* **2010**, *19*, 3156–3166. [CrossRef] [PubMed]
14. Li, X.; Li, J.; Li, B.; Yang, B. High-fidelity reversible data hiding scheme based on pixel-value-ordering and prediction-error expansion. *Signal Process.* **2013**, *93*, 198–205. [CrossRef]
15. Chang, C.C.; Kieu, T.D.; Chou, Y.C. Reversible data hiding scheme using two steganographic images. In Proceedings of the IEEE Region 10 International Conference (TENCON), Taipei, Taiwan, 30 October–2 November 2007; pp. 1–4.
16. Lee, C.F.; Huang, Y.L. Reversible data hiding scheme based on dual stegano-images using orientation combinations. *Telecommun. Syst.* **2013**, *52*, 2237–2247. [CrossRef]
17. Lu, T.C.; Tseng, C.Y.; Wu, J.H. Dual imaging-based reversible hiding technique using LSB matching. *Signal Process.* **2015**, *108*, 77–89. [CrossRef]
18. Huynh, N.T.; Bharanitharan, K.; Chang, C.C. Quadri-directional searching algorithm for secret image sharing using meaningful shadows. *J. Vis. Commun. Image Represent.* **2015**, *28*, 105–112. [CrossRef]
19. Yao, H.; Qin, C.; Tang, Z.; Tian, Y. Improved dual-image reversible data hiding method using the selection strategy of shiftable pixel's coordinates with minimum distortion. *Signal Proc.* **2017**, *135*, 26–35. [CrossRef]
20. Liu, Y.J.; Chang, C.C. A turtle shell-based visual secret sharing scheme with reversibility and authentication. *Multimed. Tools Appl.* **2018**, *77*, 25295–25310. [CrossRef]
21. Lin, J.Y.; Chen, Y.; Chang, C.C.; Hu, Y.C. Dual-image-based reversible data hiding scheme with integrity verification using exploiting modification direction. *Multimed. Tools Appl.* **2019**, *78*, 25855–25872. [CrossRef]
22. Chen, X.; Guo, W. Reversible data hiding scheme based on fully exploiting the orientation combinations of dual stego-images. *Int. J. Netw. Secur.* **2020**, *22*, 126–135.
23. Lu, T.C.; Vo, T.N.; Jana, B. Dual-image reversible data hiding based on encoding the numeral system of concealed information. In Proceedings of the 2023 15th International Conference on Advanced Computational Intelligence (ICACI), Seoul, Republic of Korea, 6–9 May 2023; pp. 1–7
24. Kumar, S.; Gupta, A.; Walia, G.S. Dual Image Reversible Data Hiding: A Survey. In Proceedings of the 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 26–28 May 2023; pp. 190–195.
25. Lee, C.-F.; Chan, K.-C. A Novel Dual Image Reversible Data Hiding Scheme Based on Vector Coordinate with Triangular Order Coding. *IEEE Access* **2024**, *12*, 90794–90814. [CrossRef]
26. Zhang, H.; Peng, Z.; Meng, F. Dual-image Reversible Data Hiding Based on Pixel Value Parity and Multiple Embedding Strategy. *Signal Process.* **2025**, *228*, 109764. [CrossRef]
27. Kim, C.; Cavazos Quero, L.; Jung, K.-H.; Leng, L. Advanced Dual Reversible Data Hiding: A Focus on Modification Direction and Enhanced Least Significant Bit (LSB) Approaches. *Appl. Sci.* **2024**, *14*, 2437. [CrossRef]
28. Shamir, A. How to share a secret. *Commun. Assoc. Comput. Mach.* **1979**, *22*, 612–613. [CrossRef]
29. Naor, M.; Shamir, A. Visual cryptography. In Proceedings of the Advances in Cryptology—EUROCRYPT'94, Perugia, Italy, 9–12 May 1994; De Santis, A., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1995; Volume 950.

30. Zhang, X.; Wang, S. Efficient Data Hiding with the Exploiting Modification Direction (EMD) Method. *IEEE Trans. Inf. Forensics Secur.* **2006**, *1*, 391–399.
31. Tseng, H.W.; Leng, H.S. A reversible modified least significant bit (LSB) matching revisited method. *Signal Process. Image Commun.* **2022**, *101*, 116556. [CrossRef]
32. Crandall, R. Some Notes on Steganography. 1998. Available online: https://dde.binghamton.edu/download/Crandall_matrix.pdf (accessed on 1 January 2025).
33. Westfeld, A. F5—A Steganographic Algorithm: High Capacity Despite Better Steganalysis; In *Information Hiding;* Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2137, pp. 289–302.
34. Fridrich, J.; Goljan, M.; Du, R. Reliable Detection of LSB Steganography in Color and Grayscale Images. In Proceedings of the 2001 Workshop on Multimedia and Security: New Challenges, Ottawa, ON, Canada, 5 October 2001; pp. 27–30.
35. Zhang, R.; Sachnev, V.; Botnan, M.B.; Kim, H.J.; Heo, J. An Efficient Embedder for BCH Coding for Steganography. *IEEE Trans. Inf. Theory* **2012**, *58*, 7272–7279. [CrossRef]
36. Zhang, X.; Wang, S. Hamming Code Embedding for Reversible Data Hiding. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME), Toronto, ON, Canada, 9–12 July 2006; pp. 1321–1324.
37. Zhang, X.; Wang, S. Hamming + 1 for Hiding Data in High-Payload Reversible Steganography. *Signal Process.* **2009**, *89*, 2198–2205.
38. Rissanen, J.; Langdon, G.G. Arithmetic Coding. *IBM J. Res. Dev.* **1979**, *23*, 149–162. [CrossRef]
39. Langdon, G.G. An Introduction to Arithmetic Coding. *IBM J. Res. Dev.* **1984**, *28*, 135–149. [CrossRef]
40. Witten, I.H.; Neal, R.M.; Cleary, J.G. Arithmetic Coding for Data Compression. *Commun. ACM* **1987**, *30*, 520–540. [CrossRef]
41. Rurik, W.; Mazumdar, A. Hamming codes as error-reducing codes. In Proceedings of the 2016 IEEE Information Theory Workshop (ITW), Cambridge, UK, 11–14 September 2016; pp. 404–408.
42. Moon, T.K. *Error Correction Coding—Mathematical Methods and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2005; pp. 2001–2006.
43. Hashemian, R. Direct Huffman coding and decoding using the table of code-lengths. In Proceedings of the ITCC 2003. International Conference on Information Technology: Coding and Computing, Las Vegas, NV, USA, 28–30 April 2003; pp. 237–241.
44. Image Databases. Available online: https://www.imageprocessingplace.com/root_files_V3/image_databases.htm (accessed on 5 January 2025).
45. Manoharan, S. An Empirical Analysis of RS Steganalysis. In Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection, Bucharest, Romania, 29 June–5 July 2008; pp. 172–177. [CrossRef]