An adaptive reversible data hiding scheme based on histogram shifting using decimal floating signed-digit stream

Reza Ghorbandost Soveiri [a], Maryam Rajabzadeh Asaar [b, *]

Abstract

This paper proposes a novel method based on histogram shifting using signed digits for data hiding. Our proposed method takes the prediction errors obtained from the original image using a 4×4 block-wise prediction. Then, we embed the information in the prediction errors of the image using the histogram shifting technique. A crucial point regarding the embeddable data in this method is that we divide the binary stream into equal parts of two, three, or four bits. For each two, three, or four-bit digit, we consider a numerical equivalent using the approach described in this paper. Subsequently, based on each of the signed digits, assigned floating numbers are used to represent the embeddable information instead of the binary stream. Experimental results for a sample image, "Airplane", with four-bit data segmentation demonstrate an outstanding embedding capacity of 825,080 bits and a PSNR of 33.87 dB, indicating that our proposed scheme achieves a remarkably high embedding capacity while maintaining an acceptable visual quality.

Keywords: Reversible data hiding, Histogram shifting, Prediction error, Signed digit stream, Floating numbers, Embedding capacity, Visual quality

1. Introduction

Authentication and safeguarding digital multimedia content and information have gained immense importance in today's world, and their significance continues to grow. In recent years, the concealment of data in digital multimedia has provided researchers with a vast field of study. With the digitization and networking of various customer services like administrative tasks, medical affairs, and authentication, the secure transfer of confidential information has become crucial. Data hiding can generally be categorized based on different criteria. It can be classified as non-blind, blind, or semi-blind data hiding, depending on the receiver's dependence on the original image, the lack thereof, and the utilization of data and parameters for information extraction [1].

Furthermore, we can base the classification on the type of confidential data stored in digital content, such as audio, images, or text. Additionally, data hiding techniques can be categorized as robust, fragile, or semi-fragile based on the resilience of the watermark or embedded data against intentional or unintentional attacks [1]. Besides the mentioned classifications, other categorizations consider the domain of data hiding in multimedia (spatial or frequency domain) and the reversibility of the original image after extracting the embedded data (reversible or irreversible data hiding). In light of these classifications, we can define our method in this paper as follows: it belongs to the spatial domain reversible data hiding technique, utilizing the histogram shifting technique. We classify it as blind data hiding because it does not rely on data and parameters and original image for data extraction.

Reza Ghorbandost Soveiri Tel.: +989112161637
reza.ghorbandost@srbiau.ac.ir

*. Corresponding author. Tel.: +989123832297
✉ Maryam Rajabzadeh Asaar
asaar@srbiau.ac.ir
m.r.asaar@iau.ir

a Department of Electrical and Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

b Department of Electrical and Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

Additionally, it falls into the category of fragile and non-robust when facing intentional attacks. Considering these characteristics and features, we can effectively employ this method in applications like content integrity verification and identity authentication. Notably, in the spatial domain of digital images, two of the most prominent data-hiding techniques are difference expansion (DE) [2,3,4,5] and histogram shifting (HS) [6-31], known for their high data embeddability and the algorithms' reversibility.

In 2003, the reversible data embedding method based on difference expansion was initially proposed by John Tian [2]. Since then, researchers have extensively utilized this technique and developed various schemes and approaches in recent years. In 2007, Todi and Rodriguez [32] introduced a method for reversible data embedding and concealment using prediction-error expansion (PEE). Since then, researchers have extensively utilized this technique with other base techniques for various schemes in recent years [6,24,31, 33-35]. In 2017, Jung [35] presented a data embedding method employing the PEE technique. Initially, Jung divided the image into non-overlapping 1×3 blocks, predicted the maximum and minimum values within each block, and then performed data embedding and image recovery using the PEE technique. Jung's method embeds two bits of data for every three pixels. Specifically, Jung 's approach exhibits a maximum embedding capacity of 2/3 of the total pixels in the images.

In 2006, Ni et al. [7] introduced the concept of reversible data hiding based on histogram shifting. Subsequently, numerous researchers have developed methods and schemes utilizing this technique to achieve high embedding capacities (EC) [8,13,16]. In the recent years, researchers have also obtained significant results by leveraging prediction error histogram shifting [9-12, 14-31]. In 2009, Tsai et al. utilized linear prediction to increase the embedding capacity [11]. Hu et al. employed cascading block-wise prediction to enhance the embedding capacity. They divided the original image into 4×4 or 5×5 blocks and embedded data in the original image prediction error based on their prediction model [14].

Furthermore, in 2020, Chang et al. achieved high embedding capacity while preserving visual image quality by utilizing cascading block-wise prediction and signed digits [19].

In 2021, Hu et al. [29] proposed a novel Convolutional Neural Network (CNN)-based prediction approach by luminously dividing a grayscale image into two sets and applying one set to predict the other set for data embedding. The proposed CNN predictor is a lightweight and computation-efficient network with the capabilities of multi receptive fields and global optimization. This CNN predictor can be trained quickly and well by using 1000 images randomly selected from ImageNet. Furthermore, they proposed a two stages of embedding scheme for this predictor. Their experimental results showed that the CNN predictor can make full use of more surrounding pixels to promote the prediction performance. Furthermore, in the their experimental way they showed that the CNN predictor with expansion embedding and histogram shifting techniques can provide better embedding performance in comparison with those classical linear predictors.

In 2024, Lu et al. [31] proposed an Optimized Convolutional Neural Network-based Predictors (OCNNP) technique that increased the zero-valued points in predicted images, significantly enhancing the embedding capacity. Additionally, they introduced a novel Lower Surround Background Complexity (LSBC)-based Prediction Error Expansion (PEE) method, which refines the sorting of prediction errors for data embedding, thereby reducing image distortion and improving overall embedding performance.

This paper proposes a data-hiding method using histogram shifting, signed digits, floating numbers, and a 4×4 cascading block-wise prediction. Initially, we use cascading block-wise prediction to calculate the prediction errors from the host image. Then, the binary stream data undergoes a two-step transformation (first transforming binary bits into signed digits and then converting signed digits into floating numbers). We embed these transformed data into the prediction errors of the image using histogram shifting. Experimental results demonstrate that our proposed method achieves an exceptionally high embedding capacity and maintains good image quality, considering the amount of embedded data compared to previous methods. We organize the remaining sections of this paper as follows: Section 2 presents the proposed method, Section 3 provides the experimental results and corresponding discussions, and finally, Section 4 presents the conclusions.

2. Proposed scheme

In this section, we first introduce the proposed encoder and decoder model. Then, we examine the strategy of assigning signed digits to the binary stream. In the following, we describe the proposed prediction scheme, and at the end of this section, we describe the scheme of data embedding, data extraction, and recovery of the original image and end with an example.

2.1. Proposed encoder scheme

According to Fig. 1, the details of our proposed scheme are presented, which we will describe. In the first step in the transmitter or encoder, the prediction (P_r) is taken from the host image (P_x) based on the cascade prediction of the proposed scheme. In the second step, the prediction error (d), (the difference between the original image and the image prediction function) is calculated. Next in the third step, data embedding in the image prediction error is done based on the histogram shifting scheme. Then, in the fourth step, we obtain the stego image from the sum of the prediction function of the original image and the embedded prediction error. The stego image is initially based on floating numbers, Finally in the fifth step, by placing the assigned key, the final stego image is obtained. It is noteworthy that the stego image based on the floating number on the receiver or decoder side helps to recover the original image prediction. For this reason, the image sent to the receiver is a stego image based on floating numbers.

Fig. 1 Proposed encoder flowchart

2.2. Proposed decoder scheme

According to Fig. 2, in the first step on the receiver or decoder side, by placing the assigned key, the main stego image from the stego image based on floating number is obtained. In the second step, we recovery the original image prediction function from the stego image based on floating number. In the third step, the embedded prediction error (d_{new}), from the difference between the main stego image and the original image prediction function is calculated. Then in the fourth step, based on the histogram of the prediction error (d_{new}), we first extract the floating numbers stream and convert it into a binary stream with two conversion steps. In the fifth step, the embedded pixels of prediction error (d_{new}) are returned to their first state. Also, the non-embeddable pixels shifted in the transmitter before the data embedding step to separate the embedded pixels from the non-embedded pixels are returned to the first state. In this step, the prediction error of the original image is recovered. In the final step, we obtain the original image (P_x) from the sum of the prediction function of the original image (P_r) and the recovered prediction error (d_{rec}).

Fig. 2 Proposed decoder flowchart

2.3. Strategy of assigning code to a binary stream

In general, in stego images, the PSNR value of the stego image expresses the quality, clarity and fidelity of the marked image compared to the original image. The larger the PSNR size, the better the quality of the image, and the resolution of the image will be at an acceptable level. In examining the dependence of PSNR in marked images, two factors cause PSNR reduction in images after data embedding, first data embedding in embeddable pixels, second the shift of unembeddable pixels equal to the maximum data size in the stream of embeddable secret data, which is used to avoid overlap between embedded and non-embedded pixels and separate them from each other. For example, in embedding the binary stream data $(101000010)_2$ binary data of "0" will not cause any change in embeddable pixels; even, if their number is huge, they only play a role in occupying the embedding capacity. The binary data of "1" creates change by one unit in the embeddable pixels after embedding, if the number of 1's in the binary stream is high, after embedding all of them in the host image, it will cause extensive changes in embeddable pixels as a result cause distortion in the image visual quality and as a result reduce the amount of PSNR. Also, a one-unit shifting in non-embeddable pixels helps reduce their PSNR. In the method of this paper, which uses the method of signed digit, the effect of this data on reducing or increasing the PSNR of the image has been carefully examined, and the results of relevant experiments prove and confirm this effect in increasing or decreasing PSNR, which we will review and analyze. In the method of the paper, we assume that "S_{bi}" is a stream of numbers of binary and this stream is supposed to be converted from binary numbers to signed digit.

In Fig. 3, n_2, n_3, and n_4 respectively show the size of the division of this stream of binary data in 2-bit binary digit, 3-bit binary digit, 4-bit binary digit format. In general, as the size of n increases, the image embedding capacity increases by the same amount according to the method used in the paper. The interpretation of this can be expressed according to Eq.1. EC_{NEW} is the new embedding capacity (using the proposed signed digit stream), which is directly related to n. EC_{OLD} is the embedding capacity (using a normal binary data stream) using the same prediction function and prediction error used in our proposed method.

Fig. 3 Strategy of to convert binary stream to signed digit stream

$$EC_{NEW} = EC_{OLD} \times n \tag{1}$$

For example, if the maximum embedding capacity in the cover image is 200,000 bits, if we use signed digit of two-digit, the new capacity will be 400,000 bits. If we use signed digit of three-digit, the embedding capacity will be 600,000 bits. By using signed digit of four-digit, we will achieve a capacity of 800,000 bits. With the same procedure, for larger n, the embedding capacity will increase and this increase will be accompanied by the loss of PSNR. In allocating the signed digit in the method of this paper, we have to say that this general rule is the same for 2-bit, 3-bit, and 4-bit binary numbers or binary numbers with more bits. For brevity, we refer to assigning the code to 3-bit digits of the binary data stream.
$(100,111,011,011,111,110,111,110,000,111,001,001,000)_2$ data stream which is divided into 3 bits to allocate the marking bits. According to the size of n, we separate the above binary data stream as 3 bits. So, based on conversion Table 1, data conversion code allocation is done.
In general, we can prevent the reduction of PSNR in an image after data embedding by using the data transform strategy. In first step, we reduce the size of data stream by converting bits of data stream into 2-bit, 3-bit, 4-bit or n-bit units. Subsequently we convert the 2-bit, 3-bit, 4-bit or n-bit units to signed digit. This data conversion helps to increase the embedding capacity in the image. However, when addressing the reduction of PSNR, we need to address two tasks:
1. Which digit should we use for 2-bit, 3-bit, 4-bit or n-bit units allocation in bit stream?
2. Which mechanism should we employ based on the repetition of these digits in the binary stream?
After conducting experiments and analysis, we have reached a general conclusion. We can interpret the scenarios mentioned in Table 1 as follows: In cases 1 and 2, there is no difference in the assigned codes for signed digits. The maximum changes in embeddable pixels after data embedding is +7 or -7 units. Moreover, to separate the embeddable pixels from the non-embeddable pixels, we will have a shift equal to the Maximum positive and negative numerical size of the allocated code to a binary stream. Specifically, for cases 1 and 2, we will shift +7 units to the right for positive pixels and -7 units to the left for negative pixels. In the conducted experiments on several standard test image for sample, "airplane" image, the data was embedded according to cases 1 and 2, resulting in a PSNR of approximately 35.75 decibels.
Furthermore, in cases 3 and 4, the assigned codes for generating the signed digits are the same. Based on the encoded information, the maximum producible distortion during data embedding in embeddable pixels will be -4 units for case 3 and +4 units for case 4. On the other hand, for distinguishing embeddable pixels from non-embeddable ones, we will have a shift in the image's histogram based on the maximum absolute numerical values of positive and negative digits in the assigned code stream. For case 3, this shift will be +3 units to the right for positive pixels and -4 units to the left for negative pixels, as illustrated in Fig. 4. Similarly, for case 4, this shift will be +4 units to the right for positive pixels and -3 units to the left for negative pixels, similar to Fig. 4.

Fig. 4 illustration of non-embeddable pixel shifting and data embedding in embeddable pixels for (n=3)

In our experiments on a standard image sample "Airplane", we embedded the data using cases 3 and 4, which improved image quality after embedding compared to cases 1 and 2 when using the encoded data arrangement. The PSNR value obtained was 39.5 dB. According to Table 1, the code allocation arrangement for cases 3 and 4 is more suitable than for cases 1 and 2.
By considering cases 3 and 4 as the codes for the signed digit stream and taking into account a statistical parameter, namely the data repetition, we aim to achieve better results in terms of PSNR for the stego image. Additionally, in case 5, the data comprises the same information as in cases 3 and 4, but their arrangement has been modified based on their repetition in the binary bit stream. Let us now delve into the analysis.

Within the 3-bit binary data stream (000, 001, 010, 011, 100, 101, 110, 111)$_2$, we calculated the number of occurrences for each 3-bit code in the binary stream. Subsequently, we assigned the least impactful data code, in terms of its marginal effect on image distortion, based on the descending order of repetition during the embedding process.

Through experiments on a standard image sample "Airplane" with an embedding capacity of 206,270 bits, we determined the repetitions of the binary codes of 000, 001, 010, 011, 100, 101, 110, and 111. We present the repetition counts in Table 2.

As shown in Table 2, binary code 110 has the highest repetition in the binary data stream, while binary code 010 has the lowest. Consequently, we initially assigned code 0 as the least impactful data regarding image distortion to the binary digit 110 and code -4 as the most impactful data to the binary digit 010. The remaining codes are allocated to suitable binary digits based on their influence on image distortion, considering their repetition.

In continuation, using signed-digit pattern of the case 5, we achieved a PSNR of 39.96 dB in the standard "Airplane" image, indicating an improvement compared to the signed-digit pattern of the case 3 or case 4.

Similarly, we define signed digits by n=2 for the bit stream (10, 01, 11, 01, 10, 11, 11, 11, 10, 11, 11, 10, 00, 01, 11, 00, 10, 01, 00, 00)$_2$, as shown in Tables 3 and 4.

Likewise, for the bit stream (1001, 1101, 1011, 1111, 1011, 1110, 0001, 1100, 1001, 0000)$_2$, we define by n=4 signed digits, as presented in Tables 5 and 6.

2.4. Proposed cascading block-wise prediction scheme

Fig.5 and algorithm 1 and algorithm 2 illustrates that we initially divided the original image into 4x4 blocks. In each block, we select the pixel P33 as the reference pixel, and by replacing it with P22, P24, P42, and P44, we obtain the prediction matrix of the proposed method presented in this paper. Finally, the difference of the original image is taken from the prediction matrix, which leads to the prediction error of the original image. The critical point is that according to the characteristics of the prediction function in the method of this paper, in each block of the image, 0.75 of each block finds the ability to embed information.

Fig. 5 Prediction method to calculate prediction error

Algorithm 1. Proposed original image prediction

Input: Original image (P_x) of size (M × N) pixel
Output: Original image prediction (P_r) of size (M × N) pixel
1: **for** i=1:4 to M **do**
2**: for** j=1:4 to N **do**
3: $P_r(i,j) = P_x(i,j)$
4: $P_r(i,j+1) = P_x(i,j+1)$
5: $P_r(i,j+2) = P_x(i,j+2)$
6: $P_r(i,j+3) = P_x(i,j+3)$
7: $P_r(i+1,j) = P_x(i+1,j)$
8: $P_r(i+1,j+1) = P_x(i+2,j+2)$
9: $P_r(i+1,j+2) = P_x(i+1,j+2)$
10: $P_r(i+1,j+3) = P_x(i+2,j+2)$
11: $P_r(i+2,j) = P_x(i+2,j)$
12: $P_r(i+2,j+1) = P_x(i+2,j+1)$
13: $P_r(i+2,j+2) = P_x(i+2,j+2)$
14: $P_r(i+2,j+3) = P_x(i+2,j+3)$
15: $P_r(i+3,j) = P_x(i+3,j)$
16: $P_r(i+3,j+1) = P_x(i+2,j+2)$
17: $P_r(i+3,j+2) = P_x(i+3,j+2)$
18: $P_r(i+3,j+3) = P_x(i+2,j+2)$
end for
end for

Algorithm 2. Proposed original image prediction errors

Input: Original image (P_x) of size (M × N) pixel , Original image prediction (P_r) of size (M × N) pixel
Output: Original image prediction errors (d) of size (M × N) pixel
1: **for** i=1:4 to M **do**
2**: for** j=1:4 to N **do**
3: $d(i,j) = P_x(i,j) - P_r(i,j)$
4: $d(i,j+1) = P_x(i,j+1) - P_r(i,j+1)$
5: $d(i,j+2) = P_x(i,j+2) - P_r(i,j+2)$
6: $d(i,j+3) = P_x(i,j+3) - P_r(i,j+3)$
7: $d(i+1,j) = P_x(i+1,j) - P_r(i+1,j)$
8: $d(i+1,j+1) = P_x(i+1,j+1) - P_r(i+1,j+1)$
9: $d(i+1,j+2) = P_x(i+1,j+2) - P_r(i+1,j+2)$
10: $d(i+1,j+3) = P_x(i+1,j+3) - P_r(i+1,j+3)$
11: $d(i+2,j) = P_x(i+2,j) - P_r(i+2,j)$
12: $d(i+2,j+1) = P_x(i+2,j+1) - P_r(i+2,j+1)$
13: $d(i+2,j+2) = P_x(i+2,j+2) - P_r(i+2,j+2)$
14: $d(i+2,j+3) = P_x(i+2,j+3) - P_r(i+2,j+3)$
15: $d(i+3,j) = P_x(i+3,j) - P_r(i+3,j)$
16: $d(i+3,j+1) = P_x(i+3,j+1) - P_r(i+3,j+1)$
17: $d(i+3,j+2) = P_x(i+3,j+2) - P_r(i+3,j+2)$
18: $d(i+3,j+3) = P_x(i+3,j+3) - P_r(i+3,j+3)$
end for
end for

2.5. Data embedding

According to the proposed method in this paper, the data embedding process in the original image consists of the following steps. Here, the binary data stream is divided into a certain number of bits, considering a value of 3 to create the signed digit stream. The embedding steps follow the same procedure for the binary data stream with n = 2 and n = 4.

Input: The original image (P_x) and the converted secret data (S)
Output: The marked image (P_{xnew})
Step1: Conversion of binary data stream "S" with two conversion steps to floating number data stream
Step2: Calculation of the prediction function of the original image (P_r) based on our Proposed method (Fig. 5)
Step3: Calculation of the prediction error of the original image (d) according to Eq. (2)

$$d = P_x - P_r \tag{2}$$

Step 4: Based on the data in the signed digits (maximum positive and negative values in the signed digits), according to Tables (2, 4, 6), we shift the non-embeddable pixels related to the prediction errors of the original image based on those values. Please refer to equations (3-5).

$$for(n=2)\ d_{new} = \begin{cases} d+1 & if \quad zero\ point > d > peak\ point \\ d-2 & if\ -zero\ point < d < peak\ point \end{cases} \tag{3}$$

$$for(n=3)\ d_{new} = \begin{cases} d+3 & if\ zero\ point > d > peak\ point \\ d-4 & if\ -zero\ point < d < peak\ point \end{cases} \tag{4}$$

$$for(n=4)\ d_{new} = \begin{cases} d+7 & if\ zero\ point > d > peak\ point \\ d-8 & if\ -zero\ point < d < peak\ point \end{cases} \tag{5}$$

Step5: Embedding the floating number stream data in embeddable pixels Eq. (6)

$$for(n=2)\&(n=3)\&(n=4) \quad d_{new} = \begin{cases} d_{new} + S & if\ d_{new} = 0 \\ d_{new} & otherwise \end{cases} \tag{6}$$

Step6: The sum of the embedded prediction error (d_{new}) with the prediction of the original image (P_r)

$$P_{xnew} = d_{new} + P_r \tag{7}$$

Step7: Replacing signed digit instead of the floating number and finally creating the stego image

Fig. 4 shows a representation of the non-embeddable pixel shifting and the embedding of signed digit stream in embeddable pixels.

2.6. Data extraction and original image recovery

According to the method of this paper, the steps of data extraction and original image recovery for n=3 are performed as follows. These steps are the same for n=2 and n=4.

Input: The marked image (P_{xnew})
Output: The extracted secret data (S) and the recovered original image (P_x)

Step1: Calculation of the prediction function of the original image (Pr) and the stego image (P_{xnew}) from the stego image based on the floating number
Step2: Calculation of the prediction error of the stego image (d_{new}) according to Eq. (8)

$$d_{new} = P_{xnew} - P_r \tag{8}$$

Step3: The extraction of embedded data and the two-step conversion of that data into a stream of binary bits for n=2, n=3, and n=4 are the same. For example, the extraction method for the signed digit with n=3 is performed according to Eq. (9). For the signed digit with n=2 and n=4, the extraction method is done in the same way.

$$S = \begin{cases} 0 & if \quad d_{new} = 0 \\ 1 & if \quad d_{new} = 1 \\ 2 & if \quad d_{new} = 2 \\ 3 & if \quad d_{new} = 3 \\ -1 & if \quad d_{new} = -1 \\ -2 & if \quad d_{new} = -2 \\ -3 & if \quad d_{new} = -3 \\ -4 & if \quad d_{new} = -4 \end{cases} \tag{9}$$

Step 4: The prediction error recovery of the original image (d_{rec}) for the signed digit with n=3 is done according to Eqs, (10) and (11).

$$d_{rec} = \begin{cases} 0 & if \quad d_{new} = 0 \\ d_{new} - 1 & if \quad d_{new} = 1 \\ d_{new} - 2 & if \quad d_{new} = 2 \\ d_{new} - 3 & if \quad d_{new} = 3 \\ d_{new} + 1 & if \quad d_{new} = -1 \\ d_{new} + 2 & if \quad d_{new} = -2 \\ d_{new} + 3 & if \quad d_{new} = -3 \\ d_{new} + 4 & if \quad d_{new} = -4 \\ d_{new} & otherwise \end{cases} \tag{10}$$

$$d_{rec} = \begin{cases} d_{rec} - 3 & if \quad zero\ point > d_{rec} > peak\ point \\ d_{rec} + 4 & if \quad -zero\ point < d_{rec} < peak\ point \end{cases} \tag{11}$$

Step5: The sum of the recovered prediction error (d_{rec}) with the prediction of the original image (P_r) according to Eq. (12).

$$P_x = d_{rec} + P_r \tag{12}$$

Fig. 6 shows a representation of data extraction from embedded pixels and the inverse shifting of non-embedded pixels.

Fig. 6 illustration of data extraction from embedded pixels and reverse shifting of non-embedded pixels for (n=3)

2.7. Example of the proposed scheme

This section provides an example of the data embedding process based on the proposed method in this paper. we follow the procedure depicted in Fig. 7 and take the following steps:
Firstly, the procedure begins with obtaining the prediction error from the original image using the suggested cascading block-wise prediction. After shifting the non-embeddable pixels, we embed the secret data into the remaining embeddable pixels. In this example, we set the length of the signed digit stream to n=3. In the first step, we transform the binary bit stream $(100,111,011,011,111,110,111,110,000,111,001,001)_2$ into signed digits $(-3\ -2\ -1\ -1\ -2\ \ 0\ -2\ \ 0\ \ 2\ -2\ \ 1\ 1)_{sd}$ based on n=3 using Table 2. Then, in the second step, the signed digit stream is converted into a floating number stream $(0.401\ \ 0.714\ \ 0.381\ \ 0.366\ \ 0.711\ \ 0.655\ \ 0.761\ \ 0.689\ \ 0.066\ \ 0.715\ \ 0.102\ \ 0.179)_{fn}$ using Eq. (15). The data embedded in the embeddable pixels of the image prediction error is the same stream of floating number data, then the embedded prediction error is added to the prediction of the original image, the result is a marked image with data content of floating numbers. If we replace the embedded floating data with the equivalent of the signed digit according to Eq.(16), we will get the final marked image result.

Fig. 7 Example of data embedding

$$(data)_{sd}\ to\ (data)_{fn} \leftarrow \begin{cases} 0 < \beta_0 < 0.1 & if \quad S_i = 1 \\ 0.1 < \beta_{0.1} < 0.2 & if \quad S_i = -1 \\ 0.2 < \beta_{0.2} < 0.3 & if \quad S_i = -2 \\ 0.3 < \beta_{0.3} < 0.4 & if \quad S_i = 0 \end{cases} \tag{13}$$

$$(data)_{fn}\,to\,(data)_{sd} \leftarrow \begin{cases} S_i = 1 & if \quad 0 < \beta_0 < 0.1 \\ S_i = -1 & if \quad 0.1 < \beta_{0.1} < 0.2 \\ S_i = -2 & if \quad 0.2 < \beta_{0.2} < 0.3 \\ S_i = 0 & if \quad 0.3 < \beta_{0.3} < 0.4 \end{cases} \tag{14}$$

$$(data)_{sd}\,to\,(data)_{fn} \leftarrow \begin{cases} 0 < \beta_0 < 0.1 & if \quad S_i = 2 \\ 0.1 < \beta_{0.1} < 0.2 & if \quad S_i = 1 \\ 0.2 < \beta_{0.2} < 0.3 & if \quad S_i = -4 \\ 0.3 < \beta_{0.3} < 0.4 & if \quad S_i = -1 \\ 0.4 < \beta_{0.4} < 0.5 & if \quad S_i = -3 \\ 0.5 < \beta_{0.5} < 0.6 & if \quad S_i = 3 \\ 0.6 < \beta_{0.6} < 0.7 & if \quad S_i = 0 \\ 0.7 < \beta_{0.7} < 0.8 & if \quad S_i = -2 \end{cases} \tag{15}$$

$$(data)_{fn}\,to\,(data)_{sd} \leftarrow \begin{cases} S_i = 2 & if \quad 0 < \beta_0 < 0.1 \\ S_i = 1 & if \quad 0.1 < \beta_{0.1} < 0.2 \\ S_i = -4 & if \quad 0.2 < \beta_{0.2} < 0.3 \\ S_i = -1 & if \quad 0.3 < \beta_{0.3} < 0.4 \\ S_i = -3 & if \quad 0.4 < \beta_{0.4} < 0.5 \\ S_i = 3 & if \quad 0.5 < \beta_{0.5} < 0.6 \\ S_i = 0 & if \quad 0.6 < \beta_{0.6} < 0.7 \\ S_i = -2 & if \quad 0.7 < \beta_{0.7} < 0.8 \end{cases} \tag{16}$$

$$(data)_{sd}\,to\,(data)_{fn} \leftarrow \begin{cases} 0 < \beta_0 < 0.05 & if\ \ S_i = -2 \\ 0.05 < \beta_{0.05} < 0.1 & if\ \ S_i = -4 \\ 0.1 < \beta_{0.1} < 0.15 & if\ \ S_i = -6 \\ 0.15 < \beta_{0.15} < 0.2 & if\ \ S_i = 2 \\ 0.2 < \beta_{0.2} < 0.25 & if\ \ S_i = -1 \\ 0.25 < \beta_{0.25} < 0.3 & if\ \ S_i = -8 \\ 0.3 < \beta_{0.3} < 0.35 & if\ \ S_i = 7 \\ 0.35 < \beta_{0.35} < 0.4 & if\ \ S_i = 0 \\ 0.4 < \beta_{0.4} < 0.45 & if\ \ S_i = -5 \\ 0.45 < \beta_{0.45} < 0.5 & if\ \ S_i = 3 \\ 0.5 < \beta_{0.5} < 0.55 & if\ \ S_i = -7 \\ 0.55 < \beta_{0.55} < 0.6 & if\ \ S_i = 6 \\ 0.6 < \beta_{0.6} < 0.65 & if\ \ S_i = 4 \\ 0.65 < \beta_{0.65} < 0.7 & if\ \ S_i = 5 \\ 0.7 < \beta_{0.7} < 0.75 & if\ \ S_i = 1 \\ 0.75 < \beta_{0.75} < 0.8 & if\ \ S_i = -3 \end{cases} \tag{17}$$

$$(data)_{fn}\,to\,(data)_{sd} \leftarrow \begin{cases} S_i = -2 & if \quad 0 < \beta_0 < 0.05 \\ S_i = -4 & if \; 0.05 < \beta_{0.05} < 0.1 \\ S_i = -6 & if \quad 0.1 < \beta_{0.1} < 0.15 \\ S_i = 2 & if \; 0.15 < \beta_{0.15} < 0.2 \\ S_i = -1 & if \quad 0.2 < \beta_{0.2} < 0.25 \\ S_i = -8 & if \; 0.25 < \beta_{0.25} < 0.3 \\ S_i = 7 & if \quad 0.3 < \beta_{0.3} < 0.35 \\ S_i = 0 & if \; 0.35 < \beta_{0.35} < 0.4 \\ S_i = -5 & if \quad 0.4 < \beta_{0.4} < 0.45 \\ S_i = 3 & if \; 0.45 < \beta_{0.45} < 0.5 \\ S_i = -7 & if \quad 0.5 < \beta_{0.5} < 0.55 \\ S_i = 6 & if \; 0.55 < \beta_{0.55} < 0.6 \\ S_i = 4 & if \quad 0.6 < \beta_{0.6} < 0.65 \\ S_i = 5 & if \; 0.65 < \beta_{0.65} < 0.7 \\ S_i = 1 & if \quad 0.7 < \beta_{0.7} < 0.75 \\ S_i = -3 & if \; 0.75 < \beta_{0.75} < 0.8 \end{cases} \tag{18}$$

To convert the signed digit stream into a floating number stream or vice versa, Eqs (13-18) are used during the embedding and extraction steps, depending on the length of the n- bit units (n=2 or n=3 or n=4) to create signed digit. To extract data and recover the original image, we follow the procedure depicted in Fig. 8 and take the following steps:

Firstly, we replace the stego image matrix, which contains the embedded data based on the floating number stream, with the equivalent signed digit stream. Subsequently, we receive the marked image at the receiver's side. Then, we obtain the prediction error from the marked image using prediction recovery. Based on the obtained prediction error, which includes the embedded data, we extract the data from the embedded pixels in the first step. We restore the non-embeddable pixels that were shifted at the sender's side to their initial state through an inverse shift in the second step and we obtain the prediction error of the original image. Finally, we recover the original image by adding the recovered prediction error of the image to the image prediction function.

Fig. 8 Example of data extraction and original image recovery

3. Experimental results

All of the experiments were implemented on Matlab R2016a on Windows 10. Standard images with various textures and dimensions of 512×512 with gray scale were selected from [38] and [39] to conduct the experiments, as illustrated in Fig. 9. Also KODAK dataset [40] is considered which contains 12 images in bmp image file format with size of 768×512 were selected as test images, see Fig. 10. Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) and Mean Square Error (MSE) were applied to compare the performance quality of the proposed algorithm with related methods, whose formulas are shown in Eq. (19) and Eq. (20) and Eq. (21) and Eq. (22) and Eq. (23) and Eq. (24).

Fig. 9 Test images: (a) Lena, (b) Boat, (c) Barbara, (d) Pepper in [38, 39]

Fig. 10 Kodak image dataset in [40]

$$PSNR = 10 \times log_{10} \frac{255^2}{MSE} (dB) \tag{19}$$

$$MSE = \frac{1}{H \times W} \times \sum_{i=1}^{H} \sum_{j=1}^{W} (P(i,j) - P'(i,j))^2 \tag{20}$$

$$SSIM(P,P') = [l(P,P')]^{\alpha} \times [C(P,P')]^{\beta} \times [S(P,P')]^{\gamma} \tag{21}$$
$$where\ \alpha = \beta = \gamma = 1$$

$$l(p,p') = \frac{2 \times \bar{p} \times \bar{p}' + c}{\bar{p}^2 \times (\bar{p}')^2 + c} \tag{22}$$

$$c(p,p') = \frac{2\delta_p \delta_{p'} + c}{\delta_p^2 + \delta_{p'}^2 + c} \tag{23}$$

$$s(p,p') = \frac{\delta_{pp'} + c}{\delta_p \delta_{p'} + c} \tag{24}$$

3.1. Performance of the proposed scheme

Based on the analysis of the curves depicted in Fig. 11, the bar chart in Fig. 12 and Fig. 13, and the data presented in Table 7 and Table 8, we observe a direct relation between the embedding Capacity and the number of bits n used for creating the signed digit. Generally, for n=2, the embedding capacity is twice that of the typical case using an embeddable binary bit stream. For n=3, it is three times; for n=4, it is four times the typical embeddable binary bit stream. The PSNR value of the image, which indicates the quality and level of distortion caused by data embedding, is influenced by two factors. The first factor is the distortion resulting from the shift of non-embeddable pixels, differentiating between embeddable and non-embeddable pixels. The second factor is the embedding of signed digits in the embeddable pixels. For instance, in the case of the "Lena" image, when the data is in the form of a binary stream, the embedding capacity is 202,990 bits, and the corresponding PSNR value is 51.28 dB. When embedding the signed digit stream with n=2 in the image, the embedding capacity doubles compared to the usual case. This embedding capacity reaches 405,980 bits, and the corresponding PSNR is 45.89 dB. Similarly, when embedding the signed digit stream with n=3 in the image, the embedding capacity becomes three times the usual case. In this case, the embedding capacity is 608,970 bits, and the related PSNR is 39.90 dB. Finally, when embedding the signed digit stream with n=4 in the image, the embedding capacity increases fourfold compared to the usual case. The embedding capacity reaches 811,960 bits in this instance, and the corresponding PSNR is 33.79 dB.

Fig. 11 Comparison curve between embedding capacity and PSNR values for four standard images with different n

Fig. 12 Comparison of embedding capacity of standard images with different n

Fig. 13 Comparison of embedding capacity of Kodak images with different n

3.2. Security analysis of the proposed scheme

One of the crucial aspects of data hiding in multimedia is accessing the embedded data securely. Based on the discussions in sections 2.3, we employ a two-step transformation process for embedding binary stream data. In the first step, we convert the binary data into a signed digit stream, and in the second step, we transform it into a decimal floating number stream. Both transformations utilize the numbers' statistical properties and floating nature, acting as a key provided by the sender and securely transmitted through a secure channel to the trusted receiver. For example, for n=3, we assigned a code to each 3-bit binary digit in binary stream based on their calculated the number of repetitions properties. For instance, $(000)_2$ corresponds to the number 2, $(001)_2$ corresponds to the number 1, and $(111)_2$ corresponds to the number -2. This digit assignment serves as a key and code for each 3-bit binary digit in the binary stream. If unauthorized individuals gain access to the signed digit stream, which includes the assigned codes, on the receiver's side, the data becomes practically untranslatable without possessing the key.
Additionally, each signed digit is associated with a floating number during the embedding step. In the first step on the receiver's side, we need to convert this floating number back to a signed digit. In the second step, with the help of the key, we transform the signed digit into a binary bit stream representing the original data.

3.3. Performance comparison of the proposed method with other related methods

This section compares the performance and effectiveness of the proposed method with three other methods [6, 29, 31] based on two criteria: Embedding Capacity (EC) and PSNR. The results are illustrated in the curves Fig. 14. It is important to note that this comparison is between three related methods. Fig.14 shows that the proposed method exhibits a significantly increased PSNR compared to the three other methods. Additionally, Fig. 14 presents the ratio of embedding capacity to PSNR for four sample images "Lena", "Boat" and "Barbara" and "Pepper". for instance, on the curve of the "Lena" image, when we embed the binary data size of 10,000 bits, the PSNR is 64.36 dB and when we embed the binary data size of 40,000 bits, the PSNR is 58.32 dB. in the same way after embedding the binary data size of 80,000 bits, the PSNR is 55.33 dB and after embedding the binary data size of 120,000 bits, the PSNR is 53.55 dB. Furthermore, we observe that the other methods achieve lower PSNR values in the same embedding capacities than our method. In other words, our proposed method maintains a suitable PSNR. Moving on, we perform a detailed analysis of the PSNR of the proposed method compared to the three other methods. In this comparison, we consider the signed digit stream with n=1. For the "Lena" image, our method achieves an PSNR of 64.36 dB in the EC of 10,000 bits , while the other three methods [6, 29, 31] achieve PSNRs of 45 dB, 55 dB, and 58 dB, respectively. Compared to these three methods, our method shows improvements of 43%, 17%, and 10.9%. Also our method achieves an PSNR of 58.32 dB in the EC of 40,000 bits , while the other three methods [6, 29, 31] achieve PSNRs of 38 dB, 48 dB, and 52 dB, respectively. Compared to these three methods, our method shows improvements of 53.47%, 21.5%, and 12%. our method achieves an PSNR of 55.33 dB in the EC of 80,000 bits , while the other three methods [6, 29, 31] achieve PSNRs of 34 dB, 44 dB, and 49 dB, respectively. Compared to these three methods, our method shows improvements of 62. 7%, 25.7%, and 12.9%. Finally, our method achieves an PSNR of 53.55 dB in the EC of 120,000 bits , while the other three methods [6, 29, 31] achieve PSNRs of 32 dB, 42 dB, and 47.5 dB, respectively. Compared to these three methods, our method shows improvements of 67. 3%, 27.5%, and 12.7%. Also for other standard test images, "Boat" and "Barbara" and "Pepper" the trend of increase of PSNR is almost the same.

Based on the obtained results from comparing our proposed scheme with three other schemes, it is evident that our method exhibits a significantly increased PSNR or visual quality. We can attribute this improvement to utilizing the proposed signed digits presented in this paper.

Fig. 14 Comparison between the proposed method (n=1) and the three related methods [6, 29, 31] in terms of embedding rates and PSNR values

3.4. solution for pixel underflow or overflow problems

Since the length of the pixels values in grayscale images is 8 bits and in the range [0 255], when embedding binary data, data with the value of 1 bit is lost after embedding in pixels of 255. Also, data with a value of -1 bit is lost after embedding in pixels 0. Ultimately, the data is not extracted correctly after being embedded in the image. To prevent overflow and underflow problems when embedding data, some researchers reduced the pixels values of 255 by 1 unit and increased the pixels values of zero by 1 unit [18,20,22,24,28,30,31]. But their method reduced the visual quality of the stego image. Some also used extra data (flag bits) to marked the pixels that would experience overflow and underflow problems after data embedding [21]. Their method also resulted in the space consumption of embedding extra data in the cover image and reduced image quality. In our proposed scheme, considering the use of a matrix of decimal digits to create the final stego image in the transmitter and the receiver and recovering the prediction function and the original image in the receiver, With access to auxiliary data (the decimal part of the matrix), despite encountering a alot of number of pixels of 255 and 0 in the cover image and creation numerous overflow and underflow problems when embedding data in the cover image, After receiving the decimal matrix image at the receiver side and removing the decimal part of the matrix and adding the equivalent signed digits to the whole number part of the matrix, the final stego image is obtained. In addition, with the help of the decimal matrix, the original image and the prediction function are recovered. Finally, using Eq. 14 and Eq. 16 and Eq. 18, the data is extracted based on the signed digits, and after replacing it with the equivalent binary data in the form of 2-bit, 3-bit, or 4-bit digits, the main data (binary data stream) is extracted correctly, see Table 2 and Table 4 and Table 6. Therefore, based on the proposed algorithm, the effects of the overflow and underflow problems do not have any disruption in the performance of extracting embedded data and recovering the original image from the stego image. Performance of the proposed scheme in data embedding and original image recovery against overflow and underflow problems on 12 images of Kodak image dataset is shown in Table 9.

4. Conclusions

This paper introduces a reversible data hiding method based on prediction error histogram shifting, utilizing signed digits and cascading block-wise prediction. The achievements of the proposed scheme can be summarized as follows:

1- Using signed digits stream increases the embedding capacity up to four times compared to binary bit stream.

2- The two-step transformation process, involving the conversion of binary bit stream to signed digit stream and then to floating number stream, followed by embedding them in the prediction error of the image, dramatically limits data access and effectively enhances algorithm security.

3- Using the matrix of decimal values eliminates overflow and underflow problems after data embedding.

Considering the vulnerability of spatial domain data hiding methods to deliberate attacks, our future research aims to leverage the high embedding capacity of the proposed method and employ data strategies to enhance the resistance of data embedding algorithms against intentional attacks. We aim to further improve the performance of the proposed method in terms of robustness against intentional attacks.

In the future work, we intend to increase the resistance of the stego image against all kinds of the intentional attacks by using the method of this paper and using the proposed schemes in [36] and [37].

References

1. Cox, I., Miller, M., Fridrich, J. and Kalker, T. “Digital Watermarking and Steganography”, Morgan Kaufmann Publishers Inc., San Francisco. (2007).
2. Tian, J. “Reversible data embedding using a difference expansion”, IEEE Trans. Circuits Syst, 13(8), pp. 890-896 (2003). https://doi.org/10.1109/TCSVT.2003.815962.
3. Alattar, AM. “Reversible watermark using the difference expansion of a generalized integer transform”, IEEE Transactions on Image Processing, 13(8), pp. 1147–1156 (2004). https://doi.org/10.1109/TIP.2004.828418.
4. Qu, X., Kim, S., Kim, HJ. “Reversible watermarking based on compensation”, J Electr Eng Technol, 10(1), pp. 422–428(2015). https://doi.org/10.5370/JEET.2015.10.1.422.
5. Chang, C., Huang, Y. & Lu, T. “A difference expansion based reversible information hiding scheme with high stego image visual quality”, Multimed Tools Appl, 76(10), pp. 12659–12681 (2017). https://doi.org/10.1007/s11042-016-3689-3.
6. Coltuc, D. “Improved embedding for prediction-based reversible watermarking”. IEEE Trans Inf Forensics Secur 6(3), pp. 873–882 (2011). https://doi.org/10.1109/TIFS.2011.2145372.
7. Ni, Z., Shi, Y.Q., Ansari, N., Su, W. “Reversible Data Hiding”, IEEE Transactions on Circuits and Systems for Video Technology, 16(3), pp. 354–362 (2006). https://doi.org/10.1109/TCSVT.2006.869964.
8. Fallahpour M, Sedaaghi MH. “High capacity lossless data hiding based on histogram modification”, IEICE Electron Express, 4(7), pp. 205–210 (2007). https://doi.org/10.1587/elex.4.205.
9. Hong ,W., Chen, TS., Shiu, CW. “Reversible data hiding based on histogram shifting of prediction errors”, Proceedings of the International Symposium on Intelligent Information Technology Application Workshop, pp. 292–295 (2008). https://doi.org/10.1109/ETTandGRS.2008.263.
10. Hong, W., Chen, TS., Shiu, CW. “Reversible data hiding for high quality images using modification of prediction errors”, J Syst Softw, 82(11), pp.1833–1842 (2009). https://doi.org/10.1016/j.jss.2009.05.051.
11. Tsai, PY., Hu, YC., Yeh, HL. “Reversible image hiding scheme using predictive coding and histogram shifting”, Signal Process, 89(6), pp.1129–1143 (2009). https://doi.org/10.1016/j.sigpro.2008.12.017.
12. Fallahpour, M., Megias, D., Ghanbari M. “Subjectively adapted high capacity lossless image data hiding based on prediction errors”, Multimed Tools Appl, 52(2-3), pp.513–527 (2011). https://doi.org/10.1007/s11042-010-0486-2.
13. Pan, Z., Hu, S., Ma, X., Wang, L. “Reversible data hiding based on local histogram shifting with multilayer embedding”, J. Vis. Commun. Image Represent, 31, pp.64–74 (2015). https://doi.org/10.1016/j.jvcir.2015.05.005.
14. Hu, YC., Tsai, PY., Yeh, JS., Chen, WL. “Residual histogram shifting technique based on cascading prediction for reversible data hiding”, Advanced multimedia and ubiquitous engineering, Berlin, Heidelberg, pp. 105–110 (2015). https://doi.org/10.1007/978-3-662-47487-7_16.
15. Tseng, CC., Chiu, YH., Chou, YC. “A histogram shifting-based reversible data hiding scheme using multi pattern strategy”, Proceedings - 2015 international conference on intelligent information hiding and multimedia signal processing, pp. 125–128 (2016). https://doi.org/10.1109/IIH-MSP.2015.11.
16. He, W., Xiong, G., Zhou, K., et al. “Reversible data hiding based on multilevel histogram modification and pixel value grouping”, J.Vis. Commun. Image Represent, 40, pp.459–469 (2016). https://doi.org/10.1016/j.jvcir.2016.07.014.
17. Yu, C., Zhang, X., Tang, Z., Xie, X. “Separable and error-free reversible data hiding in encrypted image based on two-layer pixel errors”, IEEE Access, 6, pp.76956–76969 (2018). https://doi.org/10.1109/ACCESS.2018.2882563.
18. Tang, Z., Xu, S., Ye, D., Wang, J., Zhang, X., Yu, C. “Real-time reversible data hiding with shifting block histogram of pixel differences in encrypted image”, J Real-Time Image Proc, 16(3), pp.709–724 (2019). https://doi.org/10.1007/s11554-018-0838-0.
19. Xie, X.Z., Chang, C.C., Hu, Y.C. “An adaptive reversible data hiding scheme based on prediction error histogram shifting by exploiting signed-digit representation”, Multimed. Tools Appl, 79, pp. 24329–24346 (2020). https://doi.org/10.1007/s11042-019-08402-6.
20. Jia, YJ., Yin, ZX., Zhang, XP., Luo, YL. “Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting”, Signal Process, 163, pp. 238–246 (2019). https://doi.org/10.1016/j.sigpro.2019.05.020.
21. Faragallah, O.S., Elaskily, M.A., Alenezi, A.F., El-Sayed, H.S., Kelash, H.M. “Quadruple histogram shifting based reversible information hiding approach for digital images”, Multimed. Tools Appl, 80, pp. 26297–26317 (2021). https://doi.org/10.1007/s11042-021-10956-3.
22. He, W., Xiong, G., Wang, Y. “Reversible Data hiding based on adaptive multiple histograms modification”, IEEE Trans. Inf. Forensics Secur, 16, pp. 3000–3012 (2021). https://doi.org/10.1109/TIFS.2021.3069173.
23. Yang, C.Y., Wu, J.L. “Two-bit embedding histogram-prediction-error based reversible data hiding for medical images with smooth area”, Computers, 10 (11), pp. 152 (2021). https://doi.org/10.3390/computers10110152.
24. Kouhi, A., Sedaaghi, M.H. “Prediction error distribution with dynamic asymmetry for reversible data hiding”. Expert Syst. Appl, 184, pp. 115475.1–115475.13 (2021). https://doi.org/10.1016/j.eswa.2021.115475.
25. Huang, L. C., Chiou, S. F., & Hwang, M. S. “A Reversible Data Hiding Based on Histogram Shifting of Prediction Errors for Two-Tier Medical Images”, Informatica, 32(1), pp. 69-84 (2021), https://doi.org/10.15388/20-INFOR422.

26. Padmaja, B., Manikandan, V.M. “A novel prediction error histogram shifting-based reversible data hiding scheme for medical image transmission”, 2021 4th International Conference on Security and Privacy (ISEA-ISAP), pp. 1–6 (2021). https://doi.org/10.1109/ISEA-ISAP54304.2021.9688572.
27. Myakal, S., Pal, R., Naveen, N. “Reversible data hiding technique using multi-layer perceptron based prediction and adaptive histogram bin shifting”, In: Proceedings of the 10th International Conference on Soft Computing for Problem Solving, pp. 231–243 (2020). https://doi.org/10.1007/978-981-16-2712-5_20.
28. Mohammadi, A., Nakhkash, M. “Sorting methods and adaptive thresholding for histogram based reversible data hiding”, Multimed Tools Appl, 80, pp. 3307–3325 (2021). https://doi.org/10.1007/s11042-020-09719-3.
29. Hu, R., Xiang, S. “CNN prediction based reversible data hiding”, IEEE Signal Process Lett, 28, pp. 464–8 (2021). https://doi.org/10.1109/LSP.2021.3059202.
30. Fu, Z., Gong, M., Long, G., Gan, Z., Chai, X., Lu, Y. “Efficient capacity-distortion reversible data hiding based on combining multipeak embedding with local complexity”, Appl. Intell, 52, pp. 13006–13026 (2022). https://doi.org/10.1007/s10489-022-03323-8.
31. Luo, Y., Qiu, Y., Lu, B., Qin, S., Fu, Q., Zhang, S., Huang, Y., Su, Y. “Reversible Data Hiding based on optimized CNN predictor and Prediction Error Expansion with Lower Surround Background Complexity”, Computers and Electrical Engineering, 119, 109472 (2024). https://doi.org/10.1016/j.compeleceng.2024.109472.
32. Thodi, DM., Rodriguez, JJ. “Expansion embedding techniques for reversible watermarking”, IEEE Transactions on Image Processing, 16(3), pp. 721–30 (2007). https://doi.org/10.1109/TIP.2006.891046.
33. Chang, C.C., Huang, Y.H., Tsai, H.Y., Qin, C. “Prediction-based reversible data hiding using the difference of neighboring pixels”, Int. J. Electron. Commun. (AEÜ), 66, PP. 758 – 766 (2012). https://doi.org/10.1016/j.aeue.2012.01.008.
34. Qin, C., Chang, C,C., Liao, L.T. “An adaptive prediction-error expansion oriented reversible information hiding scheme”, Pattern Recogn Lett, 33(16), pp. 2166–2172 (2012). https://doi.org/10.1016/j.patrec.2012.08.004.
35. Jung, KH. “A high-capacity reversible data hiding scheme based on sorting and prediction in digital images”, Multimed Tools Appl, 76(11), pp. 13127–13137 (2017). https://doi.org/10.1007/s11042-016-3739-x.
36. Milani, M. “A Novel Image Encryption using Improved Chaotic Maps and Multiple Encryption Tables”, Scientia Iranica, 31(21), pp. 2041-2055 (2022). https://doi.org/10.24200/sci.2022.59249.6138.
37. Rasouli, F., Taheri, M and Sarvestani, R.R. “A Fragile Watermarking by Hamming Code on Distributed Pixels with Perfect Recovery for Small Tampers ” , ISeCure ,15.2 (2023). https://doi.org/10.22042/isecure.2023.321411.740.
38. Standard dataset images available at https://ccia.ugr.es/cvg/CG/base.htm.
39. The USC-SIPI Image Database https://sipi.usc.edu/database/.
40. The Kodak Color Image Dataset, Image Available [Online]. Available: <http://r0k.us/graphics/kodak/>.

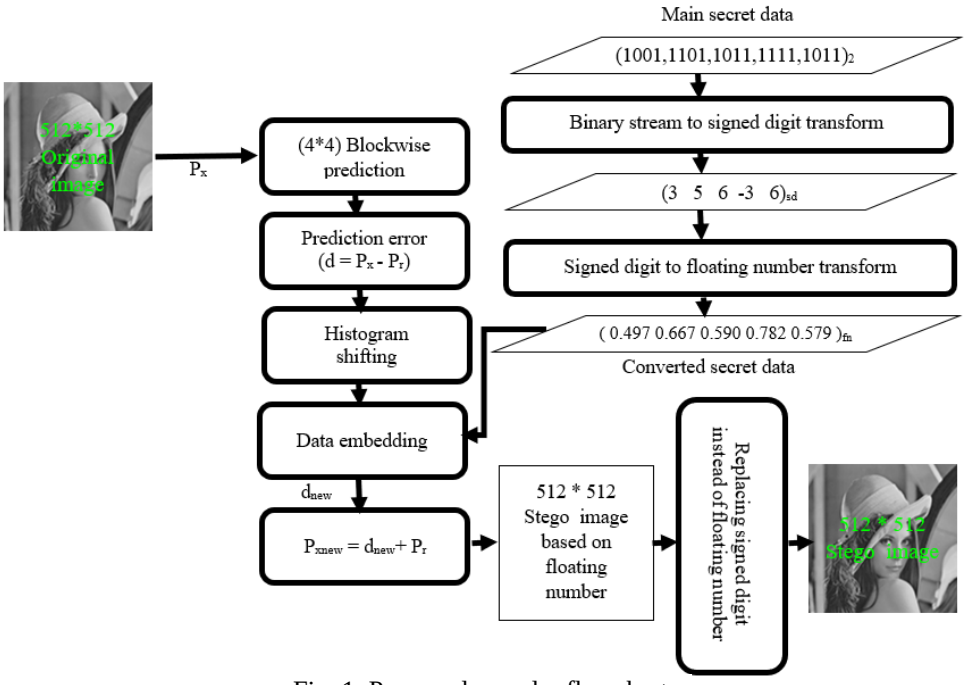

Fig. 1 Proposed encoder flowchart

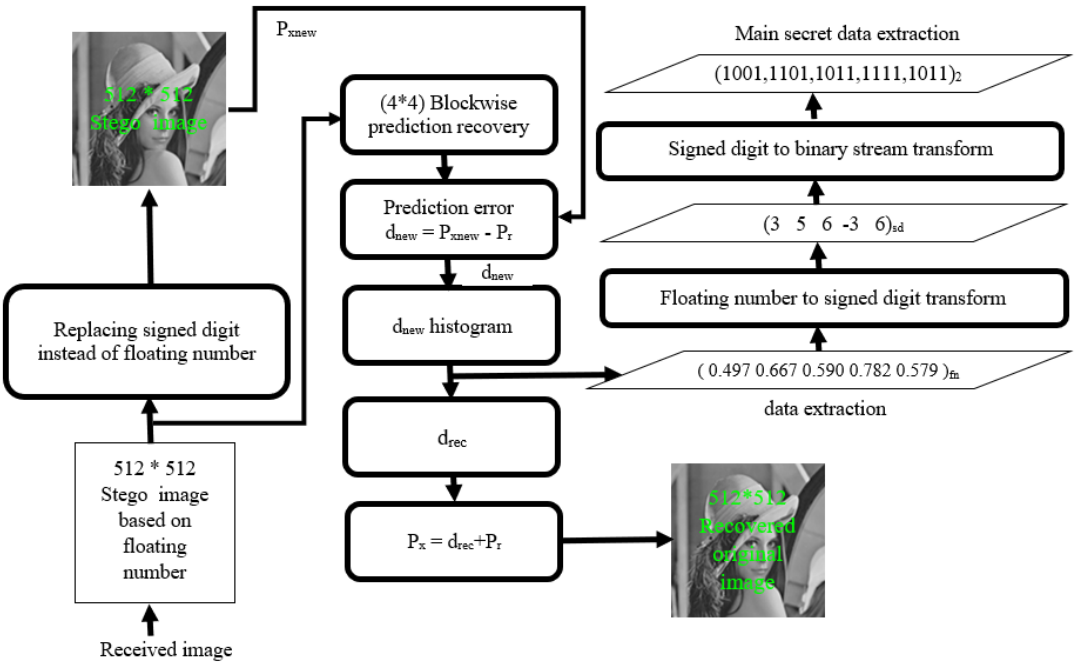

Fig. 2 Proposed decoder flowchart

$$S_{bi} = (b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \ldots\ldots\ldots\ldots\ldots\ldots)_2$$

n_1

n_2

n_3

n_4

Fig. 3 Strategy of to convert binary stream to signed digit stream

Table 1 conversion and assignment of signed digit code for (n=3)

3-bit binary digits	states of signed-digit				
	case 1	case 2	case 3	case 4	case 5
0 0 0	0	0	0	0	2
0 0 1	1	-1	1	1	1
0 1 0	2	-2	2	2	-4
0 1 1	3	-3	3	3	-1
1 0 0	4	-4	-1	4	-3
1 0 1	5	-5	-2	-1	3
1 1 0	6	-6	-3	-2	0
1 1 1	7	-7	-4	-3	-2

Table 2 For example the number of repetitions of 3-bit binary digits in binary stream and assigning a signed digit

3-bit binary digits	The number of repetitions in the binary stream	signed-digit
0 0 0	22625	2
0 0 1	26840	1
0 1 0	19782 (Min)	-4
0 1 1	29496	-1
1 0 0	26900	-3
1 0 1	22373	3
1 1 0	29568 (Max)	0
1 1 1	28686	-2

Table 3 conversion and assignment of signed digit code for (n=2)

2-bit binary digits	states of signed-digit				
	case 1	case 2	case 3	case 4	case 5
0 0	0	0	0	0	1
0 1	1	-1	1	1	-1
1 0	2	-2	-1	2	-2
1 1	3	-3	-2	-1	0

Table 4 For example the number of repetitions of 2-bit binary digits in binary stream and assigning a signed digit

2-bit binary digits	The number of repetitions in the binary stream	signed-digit
0 0	49488	1 OR-1
0 1	49327	1 OR-1
1 0	49254 (Min)	-2
1 1	58201 (Max)	0

Table 5 conversion and assignment of signed digit code for (n=4)

4-bit binary digits	states of signed-digit				
	case 1	case 2	case 3	case 4	case 5
0 0 0 0	0	0	0	0	-2
0 0 0 1	1	-1	1	1	-4
0 0 1 0	2	-2	2	2	-6
0 0 1 1	3	-3	3	3	2
0 1 0 0	4	-4	4	4	-1
0 1 0 1	5	-5	5	5	-8
0 1 1 0	6	-6	6	6	7
0 1 1 1	7	-7	7	7	0
1 0 0 0	8	-8	-1	8	-5
1 0 0 1	9	-9	-2	-1	3
1 0 1 0	10	-10	-3	-2	-7
1 0 1 1	11	-11	-4	-3	6
1 1 0 0	12	-12	-5	-4	4
1 1 0 1	13	-13	-6	-5	5
1 1 1 0	14	-14	-7	-6	1
1 1 1 1	15	-15	-8	-7	-3

Table 6 For example the number of repetitions of 4-bit binary digits in binary stream and assigning a signed digit

4-bit binary digits	The number of repetitions in the binary stream	signed-digit
0 0 0 0	11816	-2
0 0 0 1	11009	-4
0 0 1 0	9819	-6
0 0 1 1	16730	2
0 1 0 0	12179	-1
0 1 0 1	7933 (Min)	-8
0 1 1 0	12399	7
0 1 1 1	17430 (Max)	0
1 0 0 0	10625	-5
1 0 0 1	16177	3
1 0 1 0	9625	-7
1 0 1 1	12450	6
1 1 0 0	14783	4
1 1 0 1	14766	5
1 1 1 0	17039	1
1 1 1 1	11486	-3

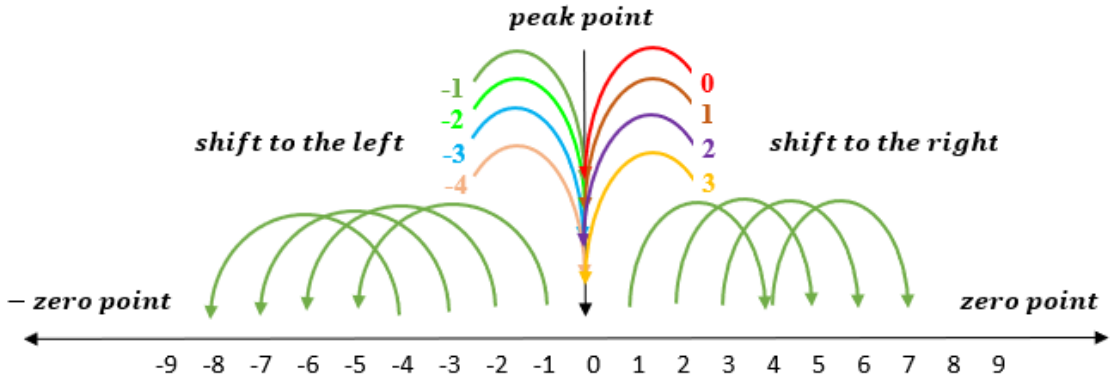

Fig. 4 illustration of non-embeddable pixel shifting and data embedding in embeddable pixels for (n=3)

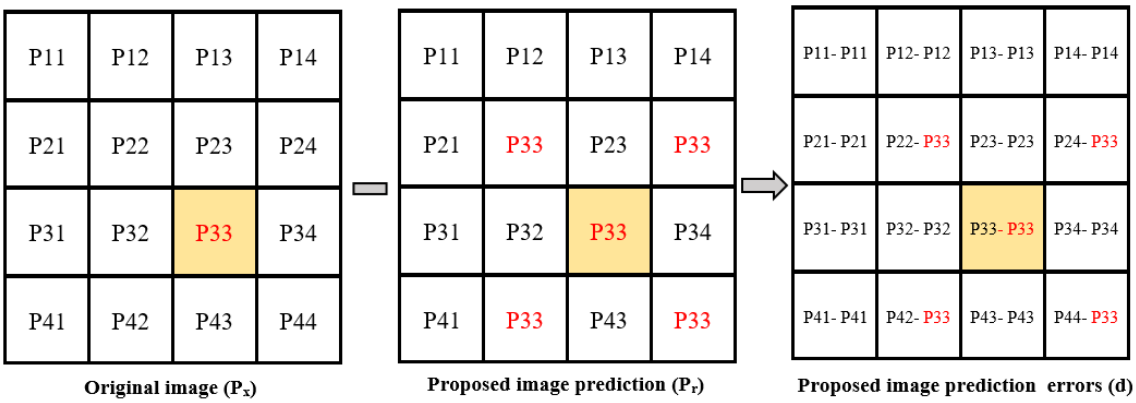

Fig. 5 Prediction method to calculate prediction error

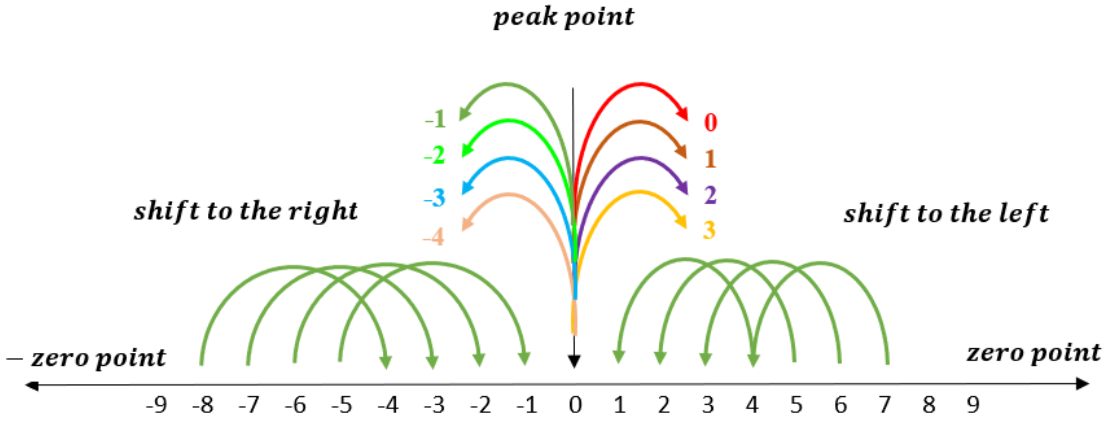

Fig. 6 illustration of data extraction from embedded pixels and reverse shifting of non-embedded pixels for (n=3)

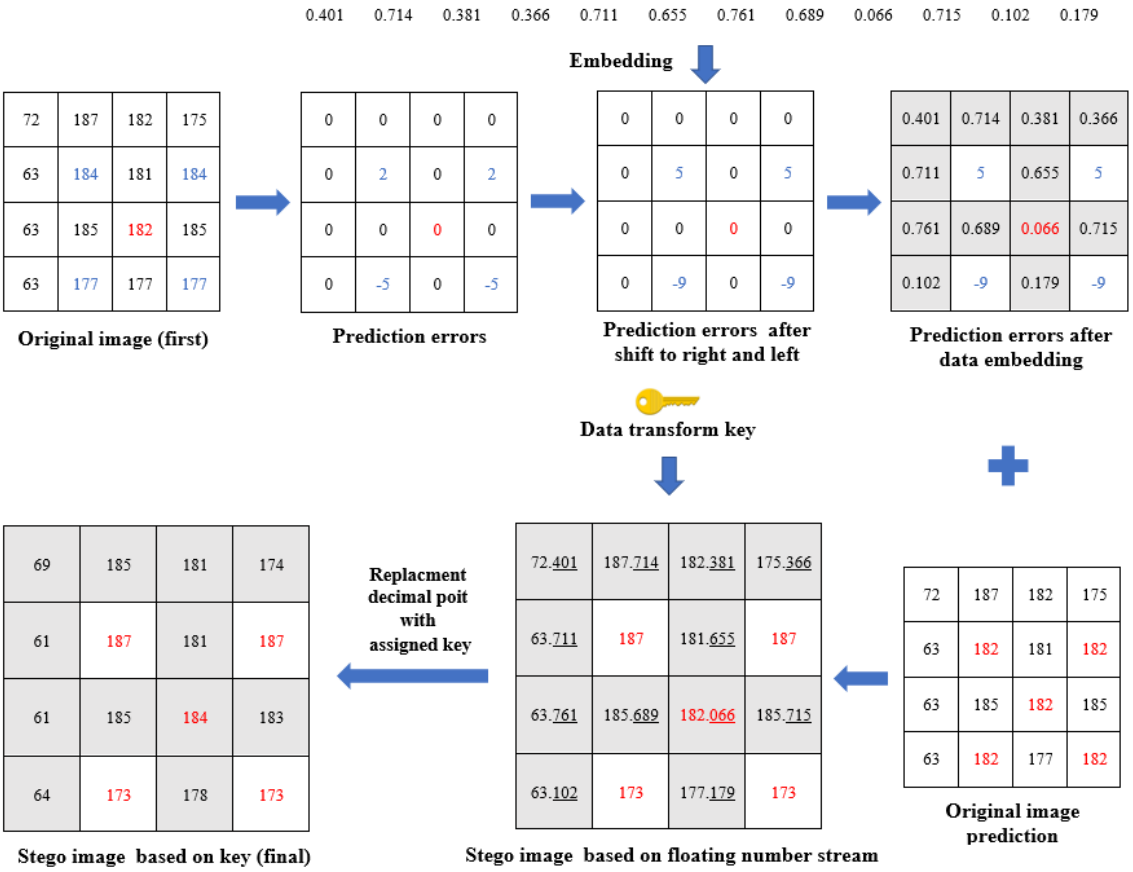

Fig. 7 Example of data embedding

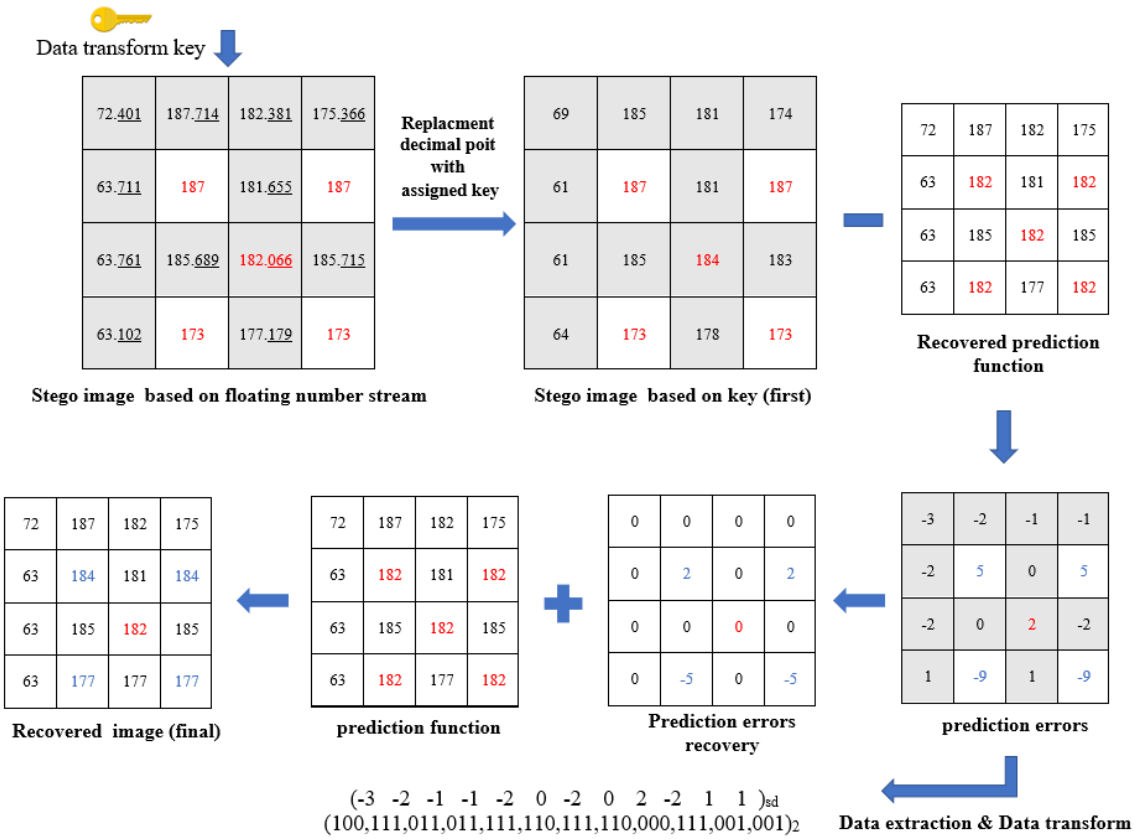

Fig. 8 Example of data extraction and original image recovery

(a) Lena (b) Boat (c) Barbara (d) Pepper

Fig. 9 Test images: (a) Lena, (b) Boat, (c) Barbara, (d) Pepper in [38, 39]

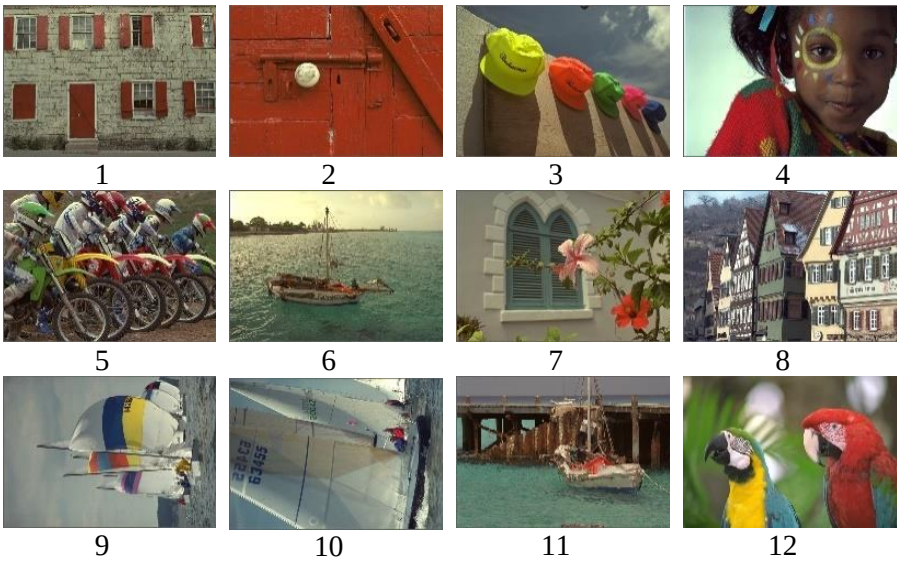

Fig. 10 Kodak image dataset in [40]

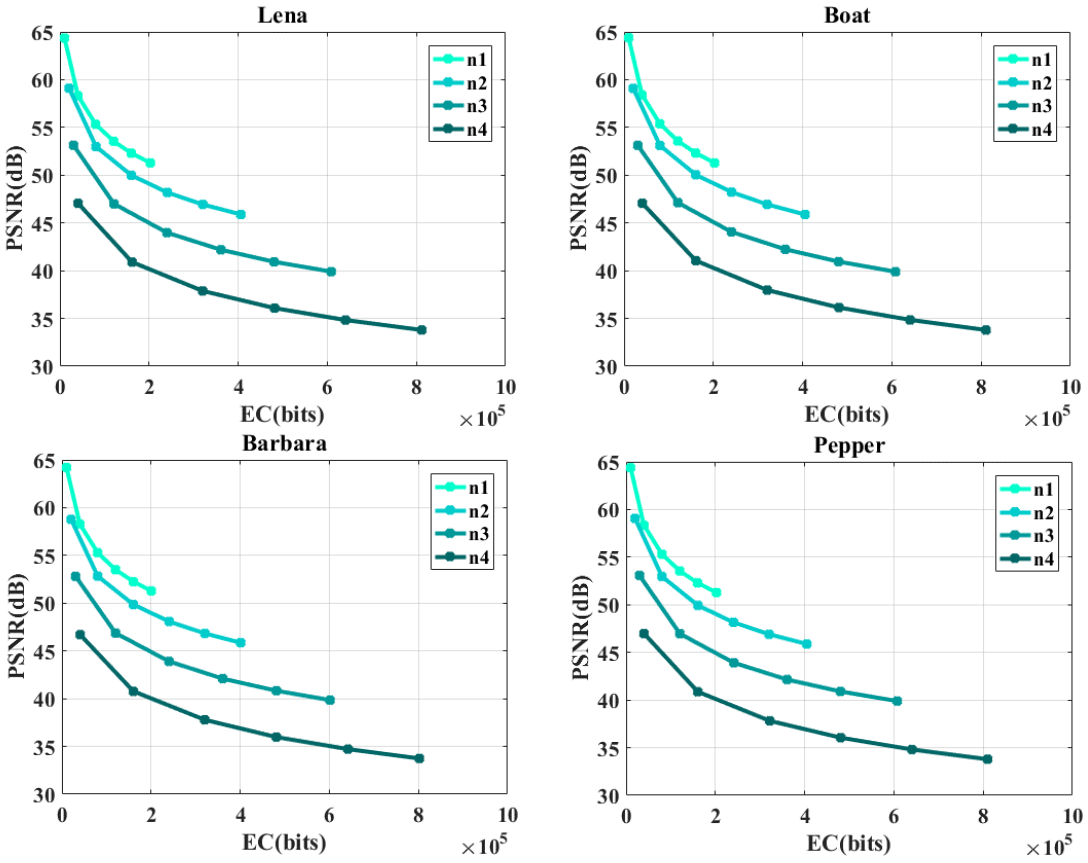

Fig. 11 Comparison curve between embedding capacity and PSNR values for four standard images with different n

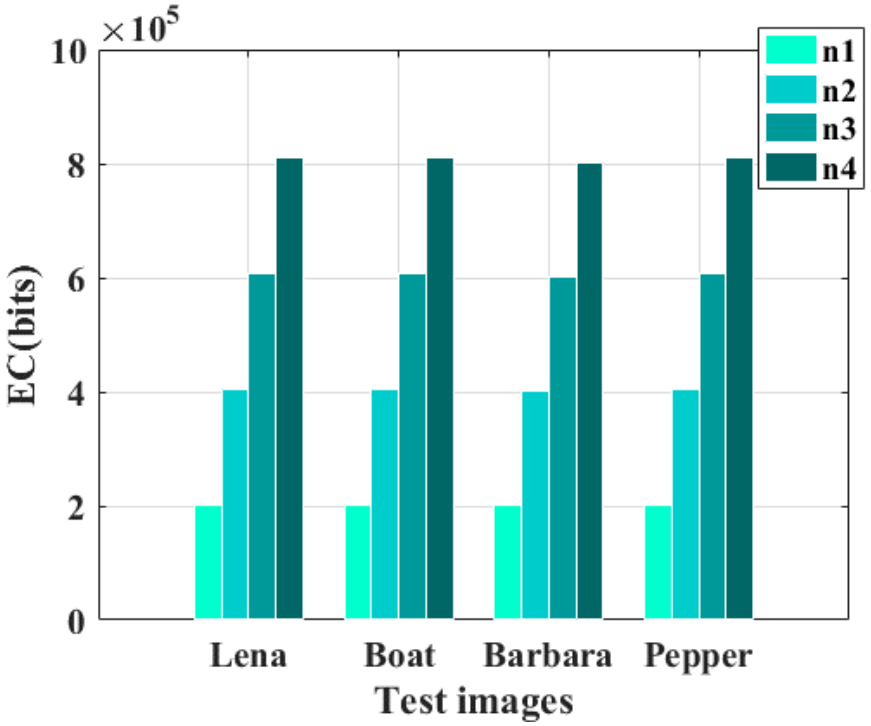

Fig. 12 Comparison of embedding capacity of standard images with different n

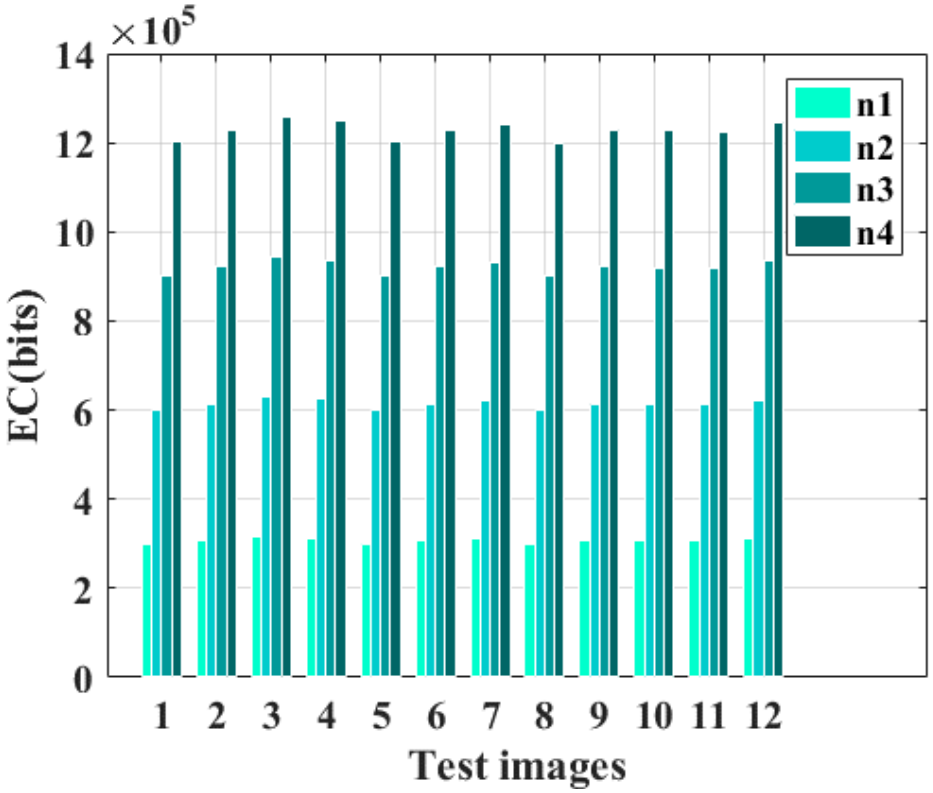

Fig. 13 Comparison of embedding capacity of Kodak images with different n

Table 7 Measured PSNR and EC values for eight standard images of 512×512 sizes along whit different n

Test images	binary digit stream **(n=1)**		signed-digit stream **(n=2)**		signed-digit stream **(n=3)**		signed-digit stream **(n=4)**	
	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)
Lena	202,990	51.28	405,980	45.89	608,970	39.90	811,960	33.79
Baboon	198,911	51.32	397.822	45.81	596,733	39.79	795,644	33.69
Airplane	206,270	51.28	412,540	45.93	618,810	39.96	825,080	33.87
Boat	202,692	51.28	405,384	45.88	608,076	39.89	810,768	33.79
Elaine	200,260	51.28	400,520	45.86	600,780	39.84	801,040	33.73
Man	201,933	51.28	403,866	45.88	605,799	39.88	807,732	33.77
Barbara	200,541	51.28	401,082	45.86	601,623	39.85	802,164	33.74
Pepper	202,472	51.28	404,944	45.88	607,416	39.89	809,888	33.79
Average	**202,008**	**51.28**	**404,017**	**45.87**	**606,025**	**39.87**	**808,034**	**33.77**

Table 8 Measured PSNR and EC values for twelve Kodak image of 512*768 sizes along whit different n

Test images	binary digit stream **(n=1)**		signed-digit stream **(n=2)**		signed-digit stream **(n=3)**		signed-digit stream **(n=4)**	
	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)	EC (**bit**)	PSNR(**dB**)
1	300,329	51.26	600,658	42.53	900,987	38.72	1,201,316	33.60
2	307,191	51.26	614.382	42.54	921,573	38.77	1,228,764	33.69
3	314,186	51.26	628,372	42.56	942,558	38.83	1,256,744	33.80
4	311,966	51.26	623,932	42.56	935,898	38.82	1,247,864	33.81
5	301,189	50.32	602,378	43.63	903,567	40.66	1,204,756	36.39
6	306,972	50.38	613,944	43.57	920,916	40.59	1,227,885	36.31
7	310,481	51.26	620,962	42.55	931,443	38.80	1,241,924	33.74
8	299,867	50.31	599,734	43.64	899,601	40.68	1,199,468	36.43
9	307,448	50.38	614,896	43.57	922,344	40.58	1,229,792	36.29
10	306,669	51.26	613,338	42.55	920,007	38.78	1,226,676	33.70
11	305,610	51.26	611,220	42.54	916,830	38.76	1,222,440	33.67
12	311,567	51.26	623,134	42.56	934,701	38.82	1,246,268	33.77
Average	**306,956**	**50.95**	**613,912**	**42.90**	**920,868**	**39.40**	**1,227,825**	**34.60**

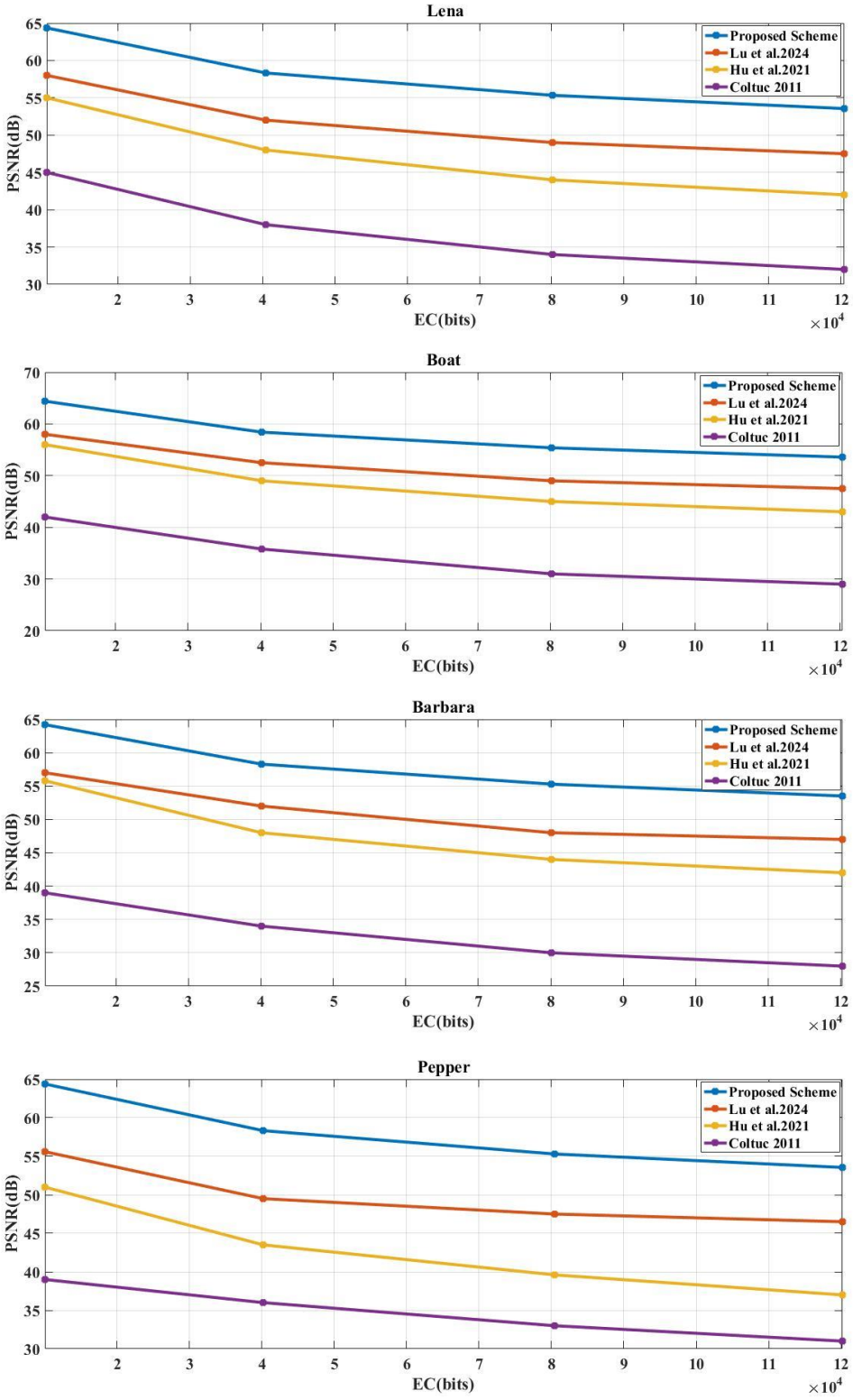

Fig. 14 Comparison between the proposed method (n=1) and the three related methods [6, 29, 31] in terms of embedding rates and PSNR values

Table 9 Performance of the proposed scheme against overflow and underflow problems

Kodak images dataset	Overflow (n)	Underflow (n)	Original image recovery (Complete)	Data extraction (Complete and correct)
1	548	768	✓	✓
2	1	768	✓	✓
3	3	768	✓	✓
4	4803	768	✓	✓
5	176	825	✓	✓
6	173	768	✓	✓
7	3	768	✓	✓
8	7542	768	✓	✓
9	48	768	✓	✓
10	1273	768	✓	✓
11	12	768	✓	✓
12	1191	768	✓	✓

Biographies

Reza Ghorbandost Soveiri received his B.S. degree in Electrical Engineering from Yazd, Islamic Azad University in 2004, and received his M.Sc. degree in Telecommunications Engineering from Science and Research Branch in 2021, Islamic Azad University. His research interests include multimedia security, data hiding, steganography and image processing.

Maryam Rajabzadeh Asaar received her M.Sc. and Ph.D. degrees in Electrical Engineering from Sharif University of Technology. She is an assistant professor at Department of Electrical and Computer Engineering, Science and Research Branch, Islamic Azad University. Her research interests include cryptographic protocols, steganography and network security.