Journal of Real-Time Image Processing
https://doi.org/10.1007/s11554-019-00855-0

SPECIAL ISSUE PAPER

A fast coding method for distortion-free data hiding in high dynamic range image

Yue Guo[1] · Weiming Zhang[1] · Dongdong Hou[1] · Yuanzhi Yao[1] · Shuangkui Ge[2]

Received: 12 November 2018 / Accepted: 22 January 2019

Abstract
Reversible data hiding (RDH) technique allows the original cover to be lossless restored after the secret message is extracted, and high dynamic range (HDR) images are becoming more and more popular. We found that the existing RDH schemes for HDR image will cause serious stream expansion, which means that the storage size of the cover HDR image will expand. Noticing that we proposed a fast coding method named reverse-Golomb code for message embedding in all-zero cover to reduce the number of the alteration of pixel's status, and thus reduce the stream expansion of cover HDR images. The experimental results show the superiority of our method.

Keywords Reversible data hiding · High dynamic range image · Stream expansion · Arithmetic coder

1 Introduction

Reversible data hiding (RDH) [1] is a peculiar type of data hiding, by which the cover media can be restored from the marked media after extracting embedded message. Since some cover media are so precious that cannot be damaged, RDH technique is widely used in military imagery, medical imagery, and law forensics.

Many reversible data hiding (RDH) methods have been proposed, since it was introduced. Fridrich and Goljan [2] presented a universal framework for reversible data hiding. To achieve larger capacity, Tian [3] proposed a method based on difference expansion (DE), and another well-known reversible data hiding method is histogram shift (HS) [4]. To achieve the better performance, the state-of-the-art RDH methods combine these strategies to the residuals of images [5–11]. The above-mentioned algorithms are presented for gray images, and many RDH methods have been proposed for color images, such as [12–17].

In the past few years, interest in high dynamic range(HDR) images has skyrocketed. HDR images can represent a greater range of luminance levels than that can be achieved using the traditional methods, which is valuable in many real-world scenes containing very bright, direct sunlight to extreme shade or very faint nebulae. HDR images is often achieved by capturing and then combining several different, narrower range, exposures of the same subject matter. Non-HDR cameras take photographs with a limited exposure range, referred as low dynamic range (LDR), resulting in the loss of detail in highlights or shadows. In comparison to the LDR images, HDR images use floating-point numbers to represent luminance for a scene to better represent the wide range of intensity levels found in real scene ranging from direct sunlight to the deepest shadows. Figure 1 displays the visual difference between LDR and HDR image. This scene has high contrast ratio with the bright light in the middle of this scene and the dark background around the building. When we directly display the LDR image, we lose the details of the building and the pool, because the luminance is out of the range supported in an ordinary device. However, when we use the tone-mapping operator to show this HDR image,

✉ Weiming Zhang
zhangwm@ustc.edu.cn

Yue Guo
guoyue@mail.ustc.edu.cn

Dongdong Hou
houdd@mail.ustc.edu.cn

Yuanzhi Yao
yaoyz@mail.ustc.edu.cn

Shuangkui Ge
xsk@beita.cn

[1] School of Information Science and Technology, University of Science and Technology of China, Hefei, China

[2] Beijing Institute of Electronic Technology Application, Beijing, China

Published online: 05 February 2019

Fig. 1 Visual contrast of LDR image and HDR image. **a** LDR image, **b** HDR image

all the details are visualized. Many image processing softwares have been developed to support HDR image, and they are becoming more and more popular in various of fields, such as digital photography, movies, medical imaging, video games, and so on.

Though RDH technique in images with 8-bit pixels is mature, the research in HDR images has not kept up with the pace of it. There has been very few RDH works done on HDR images. Cheng and Wang proposed an adaptive data hiding method for HDR images [18], who classified the pixels into the flat and boundary areas. The classification removes the restrictions of a fixed size of message embedded at each pixel to provide a large embedding capacity with the little visual distortion. Yu et al. [19] proposed the first distortion-free data hiding algorithm for HDR images with radiance RGBE format [20]. The HDR images with radiance RGBE format have some special pixels. If we operate some specific alterations to these pixels, the HDR images will not emerge any visual difference after tone mapping. This is different from the traditional RDH methods in images with 8-bit pixels which have to restore the cover media to achieve the lossless. Thus, we can distortion-freely embed message into it using such character. Subsequent works [21–23] improved the capacity of Yu's work by making better use of the homogeneous representations.

Yu's work and its subsequent works are distortion-free in the level of visual and content, but all of them ignored that it will cause stream expansion after message embedding which means that the storage size of the HDR image will expand after message embedding. In the presented paper, to reduce the stream expansion, we proposed an efficient coding method for message embedding to reduce the number of the alteration of pixel's status. The main contribution of our method is that it will sharply reduce the stream expansion. And it is very fast compared with the minimizing-distortion steganographic coding method syndrome trellis code (STC). The experimental results showed that our method has a better embedding efficiency but much faster than STC.

Because Yu's work and its subsequent work did not take into account the stream expansion and Yu's work is the most representative, to explain our method more concisely, we will introduce our method based on Yu's work rather than its follow-up work.

The following contents are organized as follows. In Sect. 2, we will introduce Yu's work. In Sect. 3.1, we will introduce our coding method, and in Sect. 3.2, the message embedding in all-zero cover will be stated. In Sects. 3.3 and 3.4, it is the analysis of our method and the message embedding in HDR images which we expound. The experimental results will be shown in Sect. 4, and the last section is the conclusion.

2 Related works

In this section, we will first introduce the radiance RGBE format for HDR images. Then, we emphatically introduce Yu's work [19].

Let $P(r, g, b, e)$ represent a pixel with the radiance format in an HDR image, where the r, g, and b are the scales in the three color channels and e represents the scale of the exponent channel, and all the channel scales r, g, b, e are integers ranging from 0 to 255. We can convert the radiance format to floating-point format by the floating-point conversion, as shown in Eq. (1). In like manner, for a given color pixel (R, G, B) with the floating value, we can convert it into the radiance (r, g, b, e) coding using the integer conversion which is shown in Eq. (2), where the $\max(R, G, B)$ represents the maximum scale in the three color components, R, G, and B:

$$\begin{aligned} R &= ((r + 0.5)/256) \times 2^{(e-128)} \\ G &= ((g + 0.5)/256) \times 2^{(e-128)} \\ B &= ((b + 0.5)/256) \times 2^{(e-128)} \end{aligned} \tag{1}$$

$$\begin{aligned} e &= \lfloor \log_2[\max(R, G, B)] + 128 \rfloor \\ r &= \lceil (256 \times R)/(2^{e-128}) \rceil \\ g &= \lceil (256 \times G)/(2^{e-128}) \rceil \\ b &= \lceil (256 \times B)/(2^{e-128}) \rceil . \end{aligned} \tag{2}$$

By Yu et al. [19], for a given pixel with the radiance format $P(r, g, b, e)$, the division operator can be used with the divisor 2 for three color channels and increase 1 to the exponent channel to obtain a representation $A(r/2, g/2, b/2, e+1)$, which would give the completely the same color after tone mapping and give almost the same floating-point color scale. Likewise, we can apply the multiplication operator to produce the representation $B(2r, 2g, 2b, e-1)$ on the condition that scales of three channel, $2r$, $2b$, $2g$, are within the range between 0 and 255. Then, this work defines the homogeneous representation group (HRG) for this pixel to represent a group of pixels where every pixel in this HRG describes the same color with $P(r, g, b, e)$. Then, we denote the HV_p to represent the homogeneity value which means the number of the pixels in this group. The elements are sorted by the values in the exponent channel using the ascending order in HRG. It allows us to define the homogeneity index (HI) for the elements in the HRG range from 0 to $(\mathrm{HV}_P - 1)$. An example is showed in Table 1.

Two basic rules make it easy to determine the HRG for a given pixel. One is that the multiplication operator only can be used when all the scales of the given pixel in color channels are less than 128. It means that overflow is not allowed. The other one is that the division operator should be stopped when an odd occurs in color channel. It is not difficult to understand it, because all scales in channel should be represented as the integers. If an odd occurs and we still do division operator, then decimal will occur, which is not allowed. After following such two rules, if we operate multiplication MU times and division DI times, HV_P, the homogeneity value of P is calculated as $\mathrm{HV}_P = \mathrm{MU} + \mathrm{DI} + 1$. For the scales in color channel with the range of 0 to 255, the max $\mathrm{HV}_P = 7$. In addition, the HRG at least has P itself, so the min $\mathrm{HV}_P = 1$. An example for a given pixel $P(96, 56, 68, 128)$, we can only apply the multiplication one time (MU = 1) to get (192, 112, 136, 127). In addition, the division operator can be used for two times (DI = 2) producing (48, 28, 34, 129) and (24, 14, 17, 130). Because of the occurrence of the odd number 17, the division operator stops to prevent decimals from happening. Therefore, the homogeneity valve of P is calculated as $\mathrm{HV}_P = 1 + 2 + 1 = 4$. After the homogeneity group and the homogeneity value of a pixel P have been determined, the pixel capacity in bits, C_P, can be calculated as Eq. (3). The capacity of the pixel means how many bits of messages can be embedded into this pixel:

$$C_P = \lfloor \log_2(\mathrm{HV}_P) \rfloor. \tag{3}$$

Then, we can use the homogeneity index table (HIT) as shown in Table 2 developed by [19] to embed secret messages into the cover pixel $P(r, g, b, e)$. For a given cover pixel P, we can obtain the homogeneity value HV_P. Depending on HV_P, we can determine the number of bits that can be conveyed by cover pixel P by the first column of Table 2. In addition, the third column describes the associate pattern of message that can be embedded with respect to different homogeneity indices. Then, we can alter the cover status $C(\mathrm{HV}_P, \mathrm{HI}_P)$ to stego status $S(\mathrm{HV}_P, \mathrm{HI}'_P)$ by consulting to the HIT.

An example shown below will tersely expound the embedding process. For a given pixel $P(96, 56, 68, 128)$, first, we produce the HRG for P. As shown in Table 3, $\mathrm{HRG}_P = \{(192, 112, 136, 127), (96, 56, 68, 128), (48, 28, 34, 129), (24, 14, 17, 130)\}$. The cover status $C(\mathrm{HV}_P, \mathrm{HI}_P) = C(4, 1)$. And two bits of secret message can be embedded into P, because the homogeneity value of P equals 4. Depending on the two secret bits, the cover status $C(4, 1)$ will be altered to four possible stego status according to Table 3. If the secret message is "11", the stego status will be $S(4, 3)$, and the stego pixel will become $P'(24, 14, 17, 130)$ which has the homogeneity

Table 1 An example of pixel P has four sorted elements in HRG

Pixel value of P	HV_P	Sorted element in HRG_P	HI_p
$P(96, 56, 68, 128)$	4	(192, 112, 136, 127)	0
		(96, 56, 68, 128)	1
		(48, 28, 34, 129)	2
		(24, 14, 17, 130)	3

Table 2 Homogeneity index table use to embed secret message into a cover pixel P with different homogeneity values

Number of bits to conveyed	HV_P	Homogeneity index						
		0	1	2	3	4	5	6
0	1	NP	–	–	–	–	–	–
1	2	"0"	"1"	–	–	–	–	–
1	3	"1"	"0"	NA	–	–	–	–
2	4	"00"	"01"	"10"	"11"	–	–	–
2	5	"01"	"10"	"11"	"00"	NA	–	–
2	6	"10"	"11"	"00"	"01"	NA	NA	–
2	7	"11"	"00"	"01"	"10"	NA	NA	NA

Table 3 An example of embedding 2 bits of secret message into a cover pixel $P(96, 56, 68, 128)$

Sorted elements in HRG_P	HI_P	Status of stego pixel	Conveyed message
(192, 112, 136, 127)	0	$S(4, 0)$	"00"
(96, 56, 68, 128)	1	$S(4, 1)$	"01"
(48, 28, 34, 129)	2	$S(4, 2)$	"10"
(24, 14, 17, 130)	3	$S(4, 3)$	"11"

index $\text{HI}_K = 3$. It is easy to notice that we will change nothing if the secret bits is "01" in this example, and the stego pixel will be exactly the same as the cover pixel.

It is noteworthy that the bit patterns associated with the homogeneity index are different with different homogeneity values of cover pixel. The benefit of this is to avoid coincident alternation of homogeneity index when embedding the same amount of secret messages. Another reason is that it will reduce the variation of histogram distributions in color channel.

The homogeneity index table is necessary to both message embedding and extraction, and we can use a secret key, *key*1, to avoid the attack of eavesdroppers. Now, given an HDR image in RGBE format with size of $M \times N$, the message embedding process is tersely made up of four following steps:

Step 1: Examine every pixel according to a secret key, *key*2, which determines the embedding order of secret message. For an examined pixel, such as K, we determine the corresponding HRG_K and HV_K.

Step 2: For the examined pixel, we calculate the pixel capacity C_K by Eq. (3). If homogeneity valve of K is greater than 1, pixel K is able to carry C_K bits secret message. Otherwise, we go back to step 1 with the next pixel.

Step 3: Compute the current cover pixel status $C(\text{HV}_K, \text{HI}_K)$. According to HIT and the secret message, we can determine the desired stego pixel status $S(\text{HV}_K, \text{HI}'_K)$.

Step 4: Alter the current cover pixel K to the stego pixel K' by selecting a corresponding element in HRG_K that has the homogeneity index of HI'_K. After all this has been done, we go back to step 1 with next pixel.

The total embedding capacity (TMC) of an HDR image can be calculated by the following equation:

$$\text{TMC} = \sum_{i=1}^{M \times N} \lfloor \log_2(\text{HV}_i) \rfloor. \tag{4}$$

The extraction of secret message is the straightforward inverse process, and we will not give unnecessary details here. It is mentioned in [19] that there are two special cases of pixel will not be used to carry the secret message. For a pixel $P(r, g, b, e)$, the first case is when the pixel scales are all zeros in both color and exponent channels. It is called "null" pixel. The second case appears when the pixel scales in the color channels are power of 2, or one or two of pixel scales is/are zeros, i.e., $P(2^j||0, 2^j||0, 2^j||0, e)$, where j is an integer ranged in [0, 7]. We refer to this type of pixel as "neutral" pixel. It's not difficult to understand that such two kinds of pixels can carry relatively more secret message, but it will cause a large pixel difference if we use these two kinds of pixel to carry secret message. The difference is caused by the magnitude of 0.5 added to the floating-point conversion, as shown in Eq. (1), and the floor function is applied for the integer conversion, as shown in Eq. (2). It is proved that if we do not use those two special cases of pixel as mentioned above to carry secret message, the pixel difference will be small enough to ignore. As for the experimental results, the average capacity offered by this method [19] is in range of 0.1256–0.1281 bits per pixel.

3 The proposed works

As we described above, Yu et al. [19] is a distortion-free data hiding scheme for HDR images, in terms of visual quality and content. After some experiments, we found that, though it is truly distortion-free, it will cause stream expansion after secret message is embedded. In another word, it will enlarge the storage size of the HDR images after we embed secret message into it and code the image. It is not difficult to understand why the stream expansion happened for it breaks the correlation of pixels after we alter the pixel scales. For a cover HDR image with storage size 2604 KB, the storage size will grow to 2672 KB with the 0.62 embedding rate using the method of [19]. The core work of the presented paper is to introduce a specific code to reduce the stream expansion.

The stream expansion appears when we alter the pixel's status. Hereby, reducing the number of the alteration of pixel's status is an efficient way to decrease the stream expansion. First, we will introduce our coding method which can be used in all-zero cover to minimize the alteration, and then, we will expound why the redundancy space in HDR images with RGBE format can be regarded as all-zero cover.

3.1 Coding method for secret data

Inspired by a compression algorithm named Golomb Code which is used to compress the sparse sequence, if we can use its decompression algorithm to encode the secret message, we can obtain a sparse message sequence. Thus, we can reduce the alteration in the embedding phase to decrease the stream expansion. Hereby, we proposed a coding algorithm

for all-zero cover called reverse-Golomb (RG) code. And why we can regard the cover as all-zero cover will be explained in Sect. 3.3.

First, we introduce a simple coding method named Unary code as preliminaries. The Unary code of a non-negative integer i is i 0's followed by a 1. For example, the Unary code of 4 is "00001". Now, we will expound our coding scheme using the Unary code. For an all-zero cover sequence $c = (c_1, c_2, \ldots, c_N)$, which means all the symbols $c_i = 0$ for $1 \le i \le N$, we want to embed a massage sequence $x = (x_1, x_2, \ldots, x_L, \ldots)$ into it. Now, we introduce an important positive integer parameters m. The parameter m is a controlled parameter decided by the embedding rate. The discussion of how to determine m appears later.

Once m is decided, we will build a binary tree to encode all the Unary code of the integers from 0 to $m-1$. We use the different prefix codes to build the tree, and details are expounded as follows. First, we choose an integer parameter k, satisfying $2^{k-1} < m \le 2^k$, which is equivalent to $k = \lceil \log_2 m \rceil$. Then, we build a k-step full binary tree which has 2^k leaves. If $m = 2^k$, the building of the tree finished. The leaves are the Unary code of 0 to $m-1$ from left to right. And the branches are left to right by a natural binary code of 0 to $m-1$. If $m < 2^k$, we need to merge some leaves. We encode 0 to $2^k - m - 1$ to the code which constituted with $k-1$ bits which means that the leaves on these branches of the full binary tree will be merged. This is $(k-1)$-bits corresponding natural binary code. Wherein, the code word of integer 0 is $k-1$ 0's, and the rest code words plus 1 orderly until the code word of $2^k - m - 1$; From $2^k - m$ to $m-1$, we use k bits to encode. Wherein, the value of code word of $(2^k - m)$ is $2 \times (2^k - m)$, and the rest code words plus 1 orderly in the same way. Now, we finished the compilation of the coding dictionary of 0 to $m-1$ using two kinds of length ($k-1$ bits and k bits) of code word when $m < 2^k$. A simple example is displayed as follows.

Example 1 Take the parameter $m = 5, k = \lceil \log_2 m \rceil = 3$. The code words of leaves of three-step binary tree, respectively, are 000, 001, 010, 011, 100, 101, 110, 111. There are three code words whose length is 2, so combine 000 with 001 to 00, 010 with 011 to 01 and 100 and 101 to 10. The leaves of this tree is the Unary code of 0 to $m-1$ from left to right orderly. Therefore, the coding of 0 to $m-1$ is 0:00, 1:01, 2:10, 3:110, and 4:111. The code tree is shown in Fig. 2.

The tree which we build above is for the Unary code of the number from 0 to $m-1$. After finishing building this tree, there is one last step to build the tree for encoding of the secret message. Started from the root, branch "0" connects m continuous 0's; Branch "1" connects the tree which we built before. It is all the operations of building the code tree for the secret message. The code tree is shown in Fig. 3,

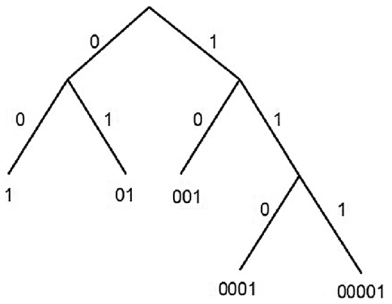

Fig. 2 Code tree for Unary code ($m = 5$)

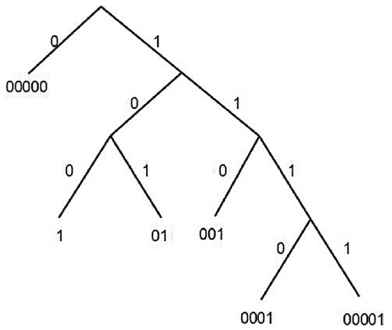

Fig. 3 Code tree for message ($m = 5$)

and the code table which is built based on the code tree is shown in Table 4. It is worth mentioning that we always put the source word "0" at the last of the table and give the index "$m+1$" to it.

3.2 Message embedding in all-zero cover

After the code table is obtained, the message embedding process is visualized. A register R and two pointers $P1$, $P2$ are needed for embedding. $P1$ is used to label the last cover symbol that has been altered, and $P2$ is used to label the number of message bits that have been embedded. R is used to store the current bits which need to be encoded and embedded. First, let $P1 = 0$ and $P2 = 0$ and clear R.

R reads in the message bits x_{P2+1}.

Case 1: If $R = 0$, let $P1 = P1 + m$, $P2 = P2 + 1$ and one bit x_{P2+1} is embedded. In this case, no cover symbol is altered.

Case 2: If $R = 1$, R reads in next $k-1$ which equals to $\lceil \log_2 m \rceil - 1$ bits message sequentially. Now, try match $R(x_{P2+1}, \ldots, x_{P2+k})$ with the source words in the second row of the code table. If it matches with the column which has the index of "j", let $P1 = P1 + j$, $P2 = P2 + k$, and flip cover symbol c_{P1} form "0" to "1", and clear R. Thus, k bit messages $(x_{P2+1}, \ldots, x_{P2+k})$ are embedded and only one cover symbol is altered. And if match failed, R reads

Table 4 Code table ($m = 5$)

Index	1	2	3	4	5	6
Source word	100	101	110	1110	1111	0
Code word	1	01	001	0001	00001	00000

in one more bit and match $R(x_{P2+1}, ..., x_{P2+k+1})$ with the code words in second row of the code table. If it matches with column which has the index of "j", let $(P1 = P1 + j)$, $(P2 = P2 + k + 1)$, and flip cover symbol c_{P1} form "0" to "1", and clear R. Thus, $k + 1$ bits $(x_{P2+1}, \ldots, x_{P2+k+1})$ are embedded and only one cover symbol is altered.

For both cases, we have embedded the first $P2$ bits of secret message into first $P1$ cover symbol. In the same way, we continue to embed the rest bits into the rest cover, until $N - P1 < m$. We obtain the marked cover $c' = (c'_1, c'_2, \ldots, c'_N)$.

The extraction is performed in the reverse way. Let $P1 = 0, P2 = 0$ and clear R. R reads in c'_{P1+1}, and there are three cases according to c'_{P1+1} and R_{len} which express the length of the current R.

Case 1: If $c'_{P1+1} = 0$ and $R_{\text{len}} < m$. Let $P1 = P1 + 1$, and R reads in one more bit c'_{P1+1}.

Case 2: If $c'_{P1+1} = 0$ and $R_{\text{len}} = m$. Let $P1 = P1 + 1$, $P2 = P2 + 1$, let the $P2$th message bit $x_{P2} = 0$. And clear R.

Case 3: If $c'_{P1+1} = 1$, check the column which has the index of R_{len} of the code table. The message source words of this column are the message fragment that we extract. Assume that the length the message fragment is a ($a = k$ or $k + 1$) and the message fragment is expressed as $A(A_1, \ldots, A_a)$. The message that we extract can be presented as $(x_{p2+1}, \ldots, x_{P2+1+a}) = A$. Let $P1 = P1 + 1$, $P2 = P2 + a$, and clear R.

With the same manner, we extract message from the rest symbol until $N - P1 < m$ and there is no symbol "1" in the rest $N - P1$ symbols of the marked cover. Now, we use a simple example to show the embedding and extraction processes of our method.

Example 2 Take $m = 5$, the code tree and code table have already built in Fig. 3 and Table 4. Assume the message strand that we want to embed is presented as $x = (0, 1, 0, 1, 0, 1, 1, 1, 0)$ and the cover is a 16-length all-zero cover, i.e., $N = 16$, as shown in Table 5. First, let $P1 = 0, P2 = 0$ and clear R.

Step 1: R reads in $x_1 = 0$, so let $P1 = P1 + 5 = 5$, $P2 = P2 + 1 = 1$, and clear R.

Step 2: R reads in $x_{P2+1} = x_2 = 1$, thus R sequentially reads in $k - 1 = 2$ bits message. Match $R(1, 0, 1)$ with the second row in Table 4. Matched with the column 2. Thus, $P1 = P1 + 2 = 7, P2 = P2 + k = 4$, and flips the cover symbol $c_{P1} = c_7$ from "0" to "1". Clear R.

Step 3: R reads in $x_{P2+1} = 0$, so let $P1 = P1 + 5 = 12$, $P2 = P2 + 1 = 5$, and clear R.

Step 4: R reads in $x_{P2+1} = 1$, thus R sequentially reads in $k - 1 = 2$ bits message. Match $R(1, 1, 1)$ with the second row in Table 4, FAILED. Therefore, R reads in one more bit. Match $R(1, 1, 1, 0)$ with code Table 4. Matched with the column 4. Thus $P1 = P1 + 4 = 16, P2 = P2 + k + 1 = 9$ and clear R. Flips the cover symbol c_{P1} from "0" to "1". Clear R. As $N - P1 = 0 < m$, so the embedding process stops. The marked cover x' is obtained by altering the 7th and 16th bits.

To extract the massage from c', set $P1 = 0$, $P2 = 0$, and clear R.

Step 1: $\{c'_{P1+1} = 0$ and $R_{\text{len}} < m$. Let $P1 = P1 + 1$, and R reads in one more bit $c'_{P1+1}\} \times 4$. Now, $c'_{P1+1} = 0$ and $R_{\text{len}} = m = 5$. $P1 = P1 + 1 = 5, P2 = P2 + 1 = 1$, let the $P2$th message bit $x_{P2} = x_1 = 0$. And clear R (the mark "$\{\divideontimes\} \times n$" express the operation "$\divideontimes$" have been repeated for n times).

Step 2: $c'_{P1+1} = 0$ and $R_{\text{len}} < m$. Let $P1 = P1 + 1 = 6$, and R reads in one more bit c'_{P1+1}. Now $c'_{P1+1} = 1$ and $R_{\text{len}} = 2$. Check the column which has index of 2 of Table 4. The message source words is expressed as $A(1, 0, 1)$ and $a = 3$. Thus, message bits $(m_2, m_3, m_4) = A = (1, 0, 1)$. Let $P1 = P1 + 1 = 7, P2 = P2 + a = 4$ and clear R.

Step 3: $\{c'_{P1+1} = 0$ and $R_{\text{len}} < m$. Let $P1 = P1 + 1$, and R reads in one more bit $c'_{P1+1}\} \times 4$. Now, $c'_{P1+1} = 0$ and $R_{\text{len}} = m = 5. P1 = P1 + 1 = 12, P2 = P2 + 1 = 5$, let the $P2$th message bit $x_{P2} = x_5 = 0$. And clear R.

Step 4: Although $N - P1 = 4 < m$, the extraction process continue, because there are symbol "1" in the last four bits. $\{c'_{P1+1} = 0$ and $R_{\text{len}} < m$. Let $P1 = P1 + 1$, and R reads in one more bit $c'_{P1+1}\} \times 3$. Now, $c'_{P1+1} = 1$ and $R_{\text{len}} = 4$.

Table 5 Example of data embedding into all-zero covers

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Message x	0	1	0	1	0	1	1	1	0							
Cover c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Marked cover c'	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Embedding step	Step 1					Step 2		Step 3					Step 4			

Check the column which has index of 4 of Table 4. The massage source words is expressed as $A(1,1,1,0)$ and $a = 4$. Thus, message bits $(m_6, m_7, m_8, m_9) = A = (1,1,1,0)$. Let $P1 = P1 + 1, P2 = P2 + a = 9$ and clear R.

In this example, we embedded 9 bits of secret message into a 16-length all-zero cover with only two alterations. We denote our embedding method by θ. We research two cases to analyze the embedding rate and distortion of θ. In case 1, we embed one bit into a m-length cover without any alteration; in case 2, we embed k or $k+1$ bits of message by costing the cover symbols range from 1 to m and one alteration occurs. Because of the randomness of the message, the probability of being "1" or "0" of the first bit register R reads in within embedding process is both 0.5. Once "0" is first read in, there will be no alteration, and one alteration will happen when "1" is first read in no matter what is to be the next. Therefore, the average number of alteration is equal to the following:

$$N_{\text{alt}} = 0.5. \tag{5}$$

As we discussed above, when $m < 2^k$ which means that m is not the power of 2, there will be $(2^k - m)$ segments of message whose length is k and $(2m - 2^k)$ segments of message whose length is $k+1$ in case 2. Consequently, the average number of embedded bits is equal to the following:

$$\begin{aligned} N_{\text{mes}} &= \frac{1}{2} \times 1 + [\frac{1}{2^k}(2^k - m)k + \frac{1}{2^{k+1}}(2m - 2^k)(k+1)] \\ &= \frac{1}{2}k + \frac{m}{2^k}. \end{aligned} \tag{6}$$

Because the number of cover symbols which we cost is ranging from 1 to m, the average number of expending cover symbols is equal to

$$N_{\text{cov}} = \frac{1}{2}m + \left(\sum_{i=1}^{2^k - m} \frac{i}{2^k} + \sum_{i=2^k-m+1}^{m} \frac{i}{2^{k+1}} \right). \tag{7}$$

Therefore, the embedding rate R_0 and the distortion D_0 of code θ can be calculated by the following:

$$R_0 = \frac{N_{\text{mes}}}{N_{\text{cov}}} = \frac{\frac{1}{2}k + \frac{m}{2^k}}{\frac{1}{2}m + \left(\sum_{i=1}^{2^k-m} \frac{i}{2^k} + \sum_{i=2^k-m+1}^{m} \frac{i}{2^{k+1}} \right)} \tag{8}$$

$$D_0 = \frac{N_{\text{alt}}}{N_{\text{cov}}} = \frac{1}{m + \left(\sum_{i=1}^{2^k-m} \frac{i}{2^{k-1}} + \sum_{i=2^k-m+1}^{m} \frac{i}{2^k} \right)}. \tag{9}$$

And if $m = 2^k$, the calculation will be visualized:

$$\begin{aligned} N_{\text{mes}} &= \frac{1}{2} + \frac{k+1}{2} = \frac{k}{2} + 1 \quad & R_0 &= \frac{N_{\text{mes}}}{N_{\text{cov}}} = \frac{2k+4}{3m+1} \\ N_{\text{cov}} &= \frac{3}{4}m + \frac{1}{4} & D_0 &= \frac{N_{\text{alt}}}{N_{\text{cov}}} = \frac{2}{3m+1}. \end{aligned} \tag{10}$$

3.3 Analysis of pixel characteristic of HDR image and parameter selection

After a survey of the pixel scale of HDR images, we found that, for overwhelming majority of the pixels, the $\max(r,g,b)$ is larger than 127 (except the "null" pixels mentioned in Sect. 2). This is because all the images in RGBE format are converted from images in floating-point format, and we can get $\max(r,g,b) \geq 128$ by Eq. (2). With regard to this feature, the redundancy space in HDR images can be regarded as all-zero cover for the initial HIs of all the pixels are always "0".

As we discussed above, the HV_p of a HRG is ranging from 1 to 7, and when the HV_p is larger than 1, this HRG can be used to carry messages. It is not difficult to understand that the larger the HV_p is, the more message that we can embed into this HRG. However, in our method, we use only the elements whose HI is equal to 0 or 1 in a HRG which means that we have to abandon some capacity of the HDR image. The purpose of this choice is to reduce the stream expansion. For the same one bit message, we want to embed; the stream expansion caused by embedding it using element with large HI will be larger than the that using small HI. And if we regard the pixel scale as random, the probability of pixel which has large HV_p is slim. Every time the HV_p grows 1, the probability of its appearance will be 8 times smaller. Therefore, this choice makes the capacity $(1/8^2) + (1/8^3) + (1/8^4) + (1/8^5) \approx 0.018$ smaller in average calculation, and it is still acceptable. Therefore, for every pixel which HV is larger than 2, we declared this pixel's HV of 2.

The embedding rate R_0 has been obtained by Eq. (8). In addition, for a given HDR image with size of $M \times N$, the total embedding capacity of our method can be easily calculated by examining the homogeneity value of each pixel, as shown in the following equation:

$$\text{TEC} = \sum_{i=1}^{M \times N} \lfloor \log_2(\text{HV}_i) \rfloor. \tag{11}$$

For the message sequence $x = (x_1, x_2, \ldots, x_n)$ to be embedded, the parameter m can be self-adaption computed after the TEC and R_0 are obtained:

$$R_0 = \frac{n}{\mathrm{TEC}} = \frac{N_{\mathrm{mes}}}{N_{\mathrm{cov}}} = \frac{\frac{1}{2}k + \frac{m}{2^k}}{\frac{1}{2}m + \left(\sum_{i=1}^{2^k-m} \frac{i}{2^k} + \sum_{i=2^k-m+1}^{m} \frac{i}{2^{k+1}}\right)}. \quad (12)$$

After all this preparatory works, the redundancy space of HDR image has completely became all-zero cover. The embedding and extraction process will be very simple and intuitive.

3.4 Message embedding and extraction with HDR image

For a given HDR image I with size of $M \times N$ and secret message $x = (x_1, x_2, \ldots, x_n)$, the embedding operation is shown in the following steps:

Step 1: Use a secret key, Key_1, to encrypt the message, and obtain $y = (y_1, y_2, \ldots, y_n)$. Examine every pixel of HDR image I to get the total embedding capacity TEC. After TEC is obtained, we can compute the parameter m using Eq. (12). Then, we use m to encode the encrypted message y to get encoded message y' using our RG code.

Step 2: Examine every pixel to obtain the homogeneous representation group (HRG) and the homogeneity value (HV) of every pixel. For an examined pixel K, if HV_K is 1, examine next pixel. If HV_K is larger than 2, set HV_K to 2.

Step 3: If $\mathrm{HV}_K = 2$, we can embed a bit of message y'_i into it. If $y'_i = 0$, we set the pixel status $C(\mathrm{HV}_K, \mathrm{HI}_K)$ to $C'(2, 0)$. If $y'_i = 1$, we set the pixel status $C(\mathrm{HV}_K, \mathrm{HI}_K)$ to $C'(2, 1)$. And examine next pixel.

We continue these processes until all messages are embedded.

The extraction of secret message is straightforward reverse. For a given stego HDR image, we examine every pixel. For an examined pixel K if $\mathrm{HV}_K = 1$, we examine next pixel. If $\mathrm{HV}_K = 2$, we compute the pixel status $C(\mathrm{HV}_K, \mathrm{HI}_K)$. If $\mathrm{HI}_K = 0$, the message bit that we extract is "0", and if $\mathrm{HI}_K = 1$, the message bit that we extract is "1". We examine next pixel until all messages are extracted. Then, we use our RG coding to decode the message that we extracted. At last, use the secret key, Key_1, to decrypt the message.

4 Experimental results and discussion

To show the advantage of our method, several experiments will be discussed in this section. We selected 20 images from the HDR image library provided by [24] as test images. In addition, we employed four HDR images as show images including "stairs", "desk", "lobby", and "mountain", as shown in Table 6. The size of these four images is 760×1016, 644×874, 512×768 and 1214×732, respectively. The number of usable pixels ($\mathrm{HV} \geqq 2$) of each image is shown in the third column of Table 6. The next column shows the usable pixel rate of each image and the average usable pixel rate of these four images can be calculated as 0.1289 bpp. The last column shows the rate of $\max(r, g, b) > 127$ of our four test images, and all of them are equal or very close to 1. Figure 4 displays the tone-mapped cover and stego images with different embedding rates. Figure 4a is the tone-mapped image of the "Stairs", and the Fig. 4b is the stego image of it with 0.62 embedding rate ($m = 4$). Figure 4c–h shows the tone-mapped images of the rest three test images and the stego images with the embedding rates of 0.4 ($m = 8$), 0.31 ($m = 12$), and 0.24 ($m = 16$). As for different images with different embedding rates, the generated stego images will not reveal any visual differences contrasted with the cover images.

Table 6 Information of HDR images

Image name	Size	Number of usable pixel	Usable pixel rate (%)	Rate of $\max(r, g, b) > 127$
Stairs	760×1016	100580	13.03	1.00
Desk	644×874	70660	12.55	1.00
Lobby	512×768	52538	13.36	1.00
Mountain	1214×732	112075	12.61	1.00

The main purpose of our method is to reduce the stream expansion caused by the method of [19]. First, we quantify the stream expansion as DS in the following equation:

$$\mathrm{DS} = \frac{S_{\mathrm{s}} - S_{\mathrm{c}}}{S_{\mathrm{c}}} \times 100\%. \quad (13)$$

S_{s} is the storage size of the stego image after secret message is embedded and the S_{c} is the storage size of the original cover image. Table 7 displays the different stream expansions with different embedding rates in our four test images.

In Table 7, the last column shows the embedding rates that we choose to test and the value of the relevant parameter m. We choose four embedding rates 0.62, 0.4, 0.31, and 0.24 to test, and the corresponding parameters m are 4, 8, 12, and 16, respectively. The second and fourth columns of Table 7 show the stream expansion of Yu's work [19] and our proposed method with diverse embedding rates. The third column is the stream expansion generated by Chang's work [23] embedding the message with the same number of bits as the [19] under the corresponding embedding rate. And the fifth column shows the ratio of the stream

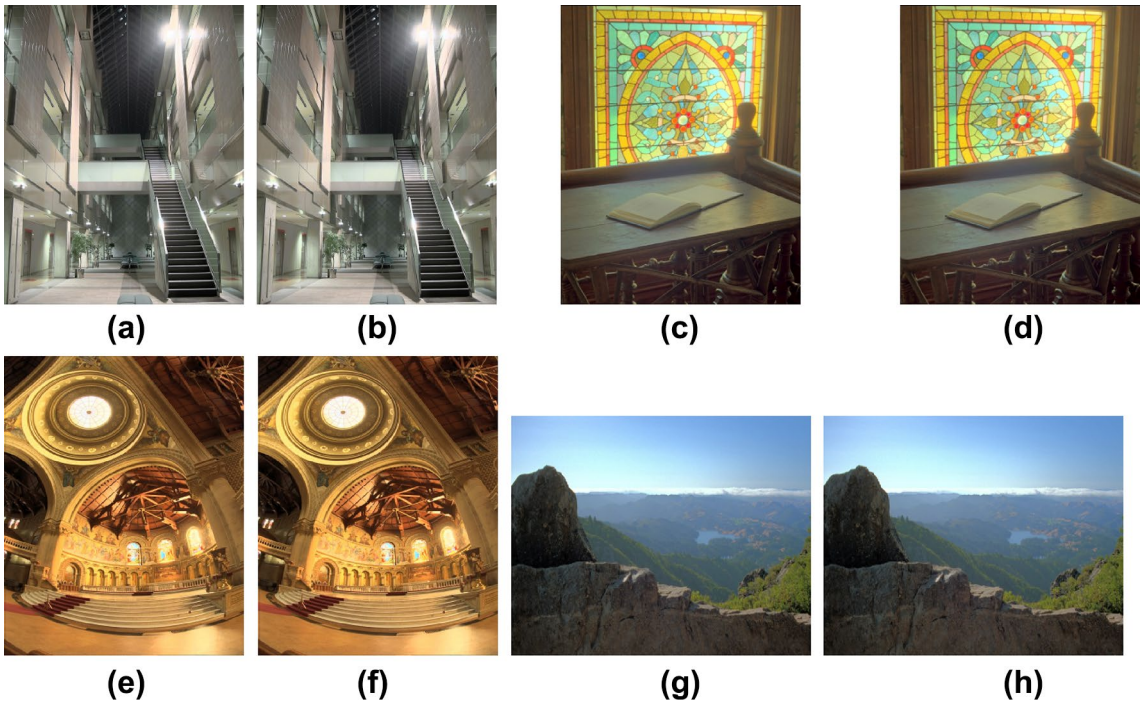

Fig. 4 Cover images and marked images. **a** Cover image, stairs. **b** Marked image, stairs. **c** Cover image, desk. **d** Marked image, desk. **e** Cover image, lobby. **f** Marked image, lobby. **g** Cover image, mountain. **h** Marked image, mountain

expansion of our method and Yu's work. It is worth mentioning that, since [23] is the follow-up work of [19], and [23] only increases the capacity of [19] to a certain extent, the stream expansion generated by [23] is almost equal to [19] when the same number of bits is embedded. Therefore, in the subsequent comparison, we will use [19] as the main

Table 7 Stream expansion comparison with different embedding rates

DS	Yu's work (%)	Chang's work (%)	Our method (%)	Stream expansion ratio	R_0 (m)
Stairs	2.61	2.64	1.42	0.54	0.62 (4)
	1.65	1.67	0.77	0.47	0.4 (8)
	1.27	1.29	0.54	0.43	0.31 (12)
	0.96	0.97	0.38	0.39	0.24 (16)
Desk	2.02	2.08	1.25	0.62	0.62 (4)
	1.14	1.16	0.67	0.59	0.4 (8)
	0.93	0.94	0.47	0.51	0.31 (12)
	0.73	0.73	0.31	0.42	0.24 (16)
Lobby	2.29	2.33	1.40	0.61	0.62 (4)
	1.48	1.50	0.74	0.5	0.4 (8)
	1.11	1.12	0.52	0.47	0.31 (12)
	0.88	0.87	0.37	0.42	0.24 (16)
Mountain	2.78	2.81	1.34	0.48	0.62 (4)
	1.88	1.90	0.70	0.37	0.4 (8)
	1.44	1.42	0.50	0.35	0.31 (12)
	1.14	1.15	0.37	0.32	0.24 (16)
Average results of 20 test images	2.45	2.48	1.33	0.54	0.62 (4)
	1.47	1.49	0.75	0.51	0.4 (8)
	1.18	1.19	0.51	0.43	0.31 (12)
	0.99	0.99	0.36	0.36	0.24 (16)

comparison method. In Table 7, the second-to-fifth rows show the experimental results of the test image "stairs". As for the 0.62 embedding rate ($m = 4$), the stream expansion of Yu's work is 2.61%, and the stream expansion of our proposed method is 1.42%. The ratio of the stream expansion of our work to Yu's work is 0.54 which means that our method reduces 46% stream expansion of Yu's work with the 0.62 embedding rate. When the embedding rate is equal to 0.4 ($m = 8$), the stream expansion of Yu's work and our work are 1.65% and 0.77% severally. And the ratio of the stream expansion of our work to Yu's work is 0.47. With the embedding rate reducing to 0.31 ($m = 12$), the stream expansion of Yu's work and our method reduce to 1.27% and 0.54% respectively. And the ratio of the stream expansion of our work to Yu's work becomes 0.43. The stream expansion of Yu's work and our work is 0.96% and 0.38% with the 0.24 embedding rate ($m = 16$). The ratio of the stream expansion of our work to Yu's work is 0.39 correspondingly. The experimental results of the last tree test images is shown in the sixth-to-seventeenth rows in Table 7. The last four rows of Table 7 show the average results of the 20 test pictures. It is not hard to understand that when the embedding rate is less than or equal to 0.75 ($m \geqq 2$), our proposed method will be able to reduce the stream expansion. And observing the fifth column of Table 7, we can discover that with the embedding rate reducing (m growing) the ratio of the stream expansion of our work to Yu's work is reducing. This means that the smaller embedding rate is, the more efficient our method is.

We have discussed before why we use only the homogeneity value of 2 in Sect. 3.3, now, we have done some experiments to prove that the total capacity [19] and the capacity of our choice of the four test images are shown in Table 8.

The second column of Table 8 shows the capacity of our method (denoted as $C1$) and the next column shows the total capacity of Yu's [19] method (denoted as $C2$). The last column shows the capacity loss rate which equals to $(C2 - C1)/C2$. For a simple example, the second row shows the experimental result of the test image "stairs". The capacity of our method is 100588 bits, and the total capacity is 103,250 bits. Therefore, the capacity loss rate can be calculated as $(103{,}250 - 100{,}588)/103{,}250 \approx 0.025$. Though the capacity loss rate of the test image "stairs" is a little bit higher than what we expect (0.018), the capacity loss rates of the last three test images which are 0.018, 0.017, and 0.017, respectively, are conform to the expectation.

Table 8 Capacities of our method and Yu's work

Capacity	Our method (bit)	Yu's work (bit)	Capacity loss rate
Stairs	100,588	103,250	0.025
Desk	70,660	71,937	0.018
Lobby	52,538	53,459	0.017
Mountain	112,075	114,040	0.017

Table 9 Maximal stream expansion of our method and Yu's work

Maximal stream expansion	Our method (%)	Yu's work (%)	Decrement of stream expansion (%)
Stairs	7.72	7.88	2.03
Desk	6.85	6.99	2.00
Lobby	7.45	7.64	2.48
Mountain	7.65	7.81	2.04

The next experiment that we did is to show the biggest stream expansion of Yu's work and our method, which means that we have to utilize the division operator as much as possible. In this experiment, the division operator will be done once the pixel having the homogeneity value no less than 2 for our method. And for Yu's method, the division operator will be tautologically utilized till an add occurs in color channel. Table 9 displays the experimental result. The second column of Table 9 shows the maximal stream expansion of possible of our method (denoted as DS1), and the next column shows the maximal stream expansion of possible of Yu's work (denoted as DS2). The last column is the decrement of the maximal stream expansion of possible which equals to $(DS2 - DS1)/DS2$. In the second cow of Table 9, the maximal stream expansion of possible of our method is 7.72% and the maximal stream expansion of possible of Yu's work is 7.88%. Therefore, the decrement of the maximal stream expansion of possible of test image "stairs" is 2.03%. The experimental results of last three test images are shown in the next three rows. Therefore, the average decrement of the four test images is 2.14%. And the two experiments discussed above proved that we sacrificed 1.8% capacity to reduce 2.14% stream expansion.

In this experiment, we compared our coding method with the syndrome trellis code (STC). In STC's method,

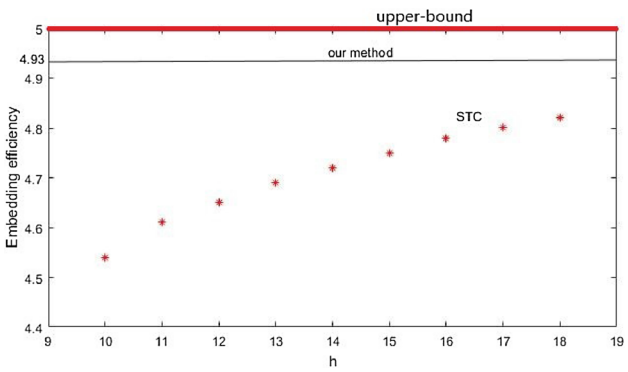

Fig. 5 Embedding efficiency of our method and STC

Table 10 Velocity of embedding of STC and our method

Method	h								
	VoE (kbit/s)								
	10	11	12	13	14	15	16	17	18
STC	798.6	423.3	220.5	121.9	60.6	31.0	15.9	7.8	4.0
Our method	16,683.8								

with the parameter h increasing the embedding efficiency (which is equal to the number of embedded message bits divided by the number of the modified cover bits) is reaching the theoretical boundary, but the algorithm complexity and execution time are exponential growth. In our method, the embedding efficiency is approaching the theoretical boundary, and the execution time is very short. We use MATLAB R2016a to implement these two algorithms. And the experimental results are shown in Fig. 5 and Table 10. In Fig. 5, the vertical axis is the embedding efficiency, and the abscissa axis is the parameter h of STC. We choose 0.4 embedding rate in this experiment, and the parameter m of our method will remain as 8. The embedding efficiency of our method will not change with the h as it stays at 4.93. With h increasing from 10 to 18, the embedding efficiency of STC grows from 4.54 to 4.82, which is still under our method but near. And in Table 10, the velocity of embedding (VoE) is shown. The velocity of embedding reduces from 798.6 kbit/s to 4kbit/s with the parameter h change from 10 to 18 in STC. And velocity of embedding of our method is much faster as 16,683.8 kbit/s. And it is worth mentioning that STC is not only suitable for all-zero cover, it is also applicable to none-all-zero cover. Therefore, we can claim that, in the all-zero cover, the embedding efficiency of our coding method is slightly higher than that of STC, but the embedding speed is much higher than that of STC.

5 Conclusions

Yu proposed the first distortion-free data hiding scheme for HDR images in terms of visual quality and content, but he ignored the stream expansion which is caused by the message embedding process. Therefore, we presented a specific coding method named reverse-Golomb code to reduce the stream expansion. Our method will sharply decrease the stream expansion with a short execution time compared with the minimizing-distortion steganographic coding method STC. And the coding method that we proposed can be used not only in HDR images but also in all the circumstances where the cover can be regarded as all-zero cover. However, the corresponding limitation of our work is that the coding method that we proposed only can be used in all-zero cover.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61572452, U1636201, and U1536104.

References

1. Khan, A., Siddiqa, A., Munib, S., Malik, S.A.: A recent survey of reversible watermarking techniques. Inf. Sci. **279**, 251–272 (2014)
2. Fridrich, J., Goljan, M., Du, R.: Lossless data embedding for all image formats. In: Proceedings of EI SPIE, Security and Watermarking of Multimedia Contents IV, vol. 4675, San Jose, pp. 572–583 (2002)
3. Tian, J.: Reversible data embedding using a difference expansion. IEEE Trans. Circuits Syst. Video Technol. **13**(8), 890–896 (2003)
4. Ni, Z., Shi, Y.Q., Ansari, N., Wei, S.: Reversible data hiding. IEEE Trans. Circuits Syst. Video Technol. **16**(3), 354–362 (2006)
5. Tsai, P., Hu, Y.C., Yeh, H.L.: Reversible image hiding scheme using predictive coding and histogram shifting. Signal Process. **89**, 1129–1143 (2009)
6. Sachnev, V., Kim, H.J., Nam, J., Suresh, S., Shi, Y.: Reversible watermarking algorithm using sorting and prediction. IEEE Trans. Circuits Syst. Video Technol. **19**(7), 989–999 (2009)
7. Yang, C.H., Tsai, M.H.: Improving histogram-based reversible data hiding by interleaving predictions. IET Image Process. **4**(4), 223–234 (2010)
8. Li, X., Yang, B., Zeng, T.: Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. IEEE Trans. Image Process. **20**(12), 3524–3533 (2011)
9. Wang, S.Y., Li, C.Y., Kuo, W.C.: Reversible data hiding based on two-dimensional prediction errors. IET Image Process. **7**(9), 805–816 (2013)
10. Wang, J., Ni, J., Zhang, X., Shi, Y.: Rate and distortion optimization for reversible data hiding using multiple histogram shifting. IEEE Trans. Cybern. **47**(2), 315–326 (2017)
11. Qin, C., Chang, C.C., Huang, Y.H., et al.: An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism. IEEE Trans. Circuits Syst. Video Technol. **23**(7), 1109–1118 (2013)
12. Qin, C., Chang, C.C., Chiu, Y.P.: A novel joint data-hiding and compression scheme based on SMVQ and image inpainting. IEEE Trans. Image Process. **23**(3), 969–978 (2014)
13. Chang, C., Lin, C., Fan, Y.: Lossless data hiding for color images based on block truncation coding. Pattern Recognit. **41**(7), 2347–2357 (2008)
14. Asikuzzaman, M., Alam, M.J., Lambert, A.J., Pickering, M.R.: A blind and robust video watermarking scheme using chrominance embedding. In: International conference on digital image computing: techniques and applications, pp. 1–6 (2014)

15. Ou, B., Li, X., Zhao, Y., Ni, R.: Efficient color image reversible data hiding based on channel-dependent payload partition and adaptive embedding. Signal Process. **108**, 642–657 (2015)
16. Li, J., Li, X., Yang, B.: Reversible data hiding scheme for color image based on prediction-error expansion and cross-channel correlation. Signal Process. **93**(9), 2748–2758 (2013)
17. Hou, D., Zhang, W., Chen, K., Lin, S., Yu, N.: Reversible data hiding in color image with grayscale invariance. In: IEEE Transactions on Circuits and Systems for Video Technology (2018)
18. Cheng, Y.M., Wang, C.M.: A novel approach to steganography in high-dynamic range images. IEEE MultiMedia **16**(3), 70–80 (2009)
19. Yu, C.M., Wu, K.C., Wang, C.M.: A distortion-free data hiding scheme for high dynamic range images. Displays **32**(1), 225–236 (2011)
20. Ward, G.: Real pixel, Graphic Gem II, Chapter 15. pp. 80–83 (1991)
21. Wang, Z.H., Lin, T.Y., Chang, C.C, Lin, C.C.: A novel distortion-free data hiding scheme for high dynamic range images. In: 2012 Fourth International Conference on Digital Home (ICDH). IEEE (2012)
22. Chang, C.C., Nguyen, T.S., Lin, C.C.: Distortion-free data embedding scheme for high dynamic range images. J. Electron. Sci. Technol. **11**(1), 20–26 (2013)
23. Chang, C.C., Nguyen, T.S., Lin, C.C.: A new distortion-free data embedding scheme for high-dynamic range images. Multimedia Tools Appl. **75**(1), 145–163 (2016)
24. http://www.anyhere.com/gward/hdrenc/pages/originals.html

Yue Guo received his B.S. degree in 2015 from University of Science and Technology of China (USTC). He is currently pursuing the M.S. degree in information security in University of Science and Technology of China (USTC). His research interests include image watermarking and information hiding.

Weiming Zhang received his M.S. degree and Ph.D. degree in 2002 and 2005, respectively, from the Zhengzhou Information Science and Technology Institute, People's Republic of China. Currently, he is a professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include information hiding and multimedia security.

Dongdong Hou received his B.S. degree in 2014 from Hefei University of Technology, Hefei, China. He is now pursuing the Ph.D. degree in University of Science and Technology of China. His research interests include multimedia security, image processing, and deep learning.

Yuanzhi Yao received his Ph.D. degree in electronic engineering from the University of Science and Technology of China in 2017, where he is currently a postdoctoral researcher. His research interests include information hiding and video coding.

Shuangkui Ge graduated from Electronic Information School, Wuhan University in 2003. Currently, he is an Associate Professor in Beijing Institute of Electronic Technology Application. He is now engaged in multimedia processing and information security research.