

Capacity-Approaching Codes for Reversible Data Hiding

Weiming Zhang^{1,2}, Biao Chen¹, and Nenghai Yu^{1,*}

¹ Department of Electrical Engineering & Information Science, University of Science and Technology of China, Hefei 230026, China

² Department of Information Research, Zhengzhou Information Science and Technology Institute, Zhengzhou 450002, China
zwmshu@gmail.com, chb893@mail.ustc.edu.cn, ynh@ustc.edu.cn

Abstract. By reversible data hiding, the original cover can be losslessly restored after the embedded information is extracted. Kalker and Willems established a rate-distortion model for reversible data hiding, in which they proved the capacity bound and proposed a recursive code construction. In this paper we improve the recursive construction by designing a data embedding method for all-zero covers and a more efficient compression algorithm. We prove that the proposed codes can approach the capacity bound under various distortion constraints. We also apply this coding method to RS method for spatial images, and the experimental results show that the novel codes can significantly reduce the embedding distortion.

Keywords: data hiding, watermark, reversible data hiding, recursive construction, arithmetic coder.

1 Introduction

Data hiding is a technique for embedding information into a cover media such as images, audio and video files, which can be used for the purpose of media notation, copyright protection, integrity authentication and covert communication, etc. Most data hiding methods embed messages into the cover media to generate the marked media by only modifying the least significant part of the cover, and thus keep perceptual transparency. The embedding process will usually introduce permanent distortion to the cover, that is, the original cover can never be reconstructed from the marked cover. However, in some applications, such as medical imagery, military imagery and law forensics, no degradation of the original cover is allowed. In these cases, we need a special kind of data hiding methods, referred to as reversible data hiding or lossless data hiding, by which the original cover can be losslessly restored after the embedded message is extracted.

* Corresponding author.

Many reversible data hiding methods have been proposed since it was introduced. Fridrich et al. [1] presented a universal framework for reversible data hiding, in which the embedding process is divided into three stages. In the first stage, extract losslessly compressible features (or portions) from the original cover. The second stage compresses the features with a lossless compression method, and thus saves space for the payload (message). In the third stage, embed messages into the feature sequence and generate the marked cover. One direct reversible embedding method is to compress the feature sequence and append messages after it to form a modified feature sequence, then replace the original features by the modified features, and thus generate the marked cover. Therefore, after extracting the message, the receiver can restore the original cover by decompressing the features. Fridrich et al. [1] suggested the features that can exploit characteristics of certain image formats, e.g., texture complexity for spatial images and middle frequency coefficients for JPEG images. Celik et al. [2] extended Fridrich's scheme by predicting multiple LSB planes. The same idea proposed in [1] can also be used for reversible data embedding into binary images [3,4] or videos [5,6].

To increase embedding capacity, the researchers desire to construct a longer feature sequence that can be perfectly compressed. One of such constructions is difference expansion (DE), first proposed by Tian [7], in which the features are the differences between two neighboring pixels of pixel pairs. The features are compressed by expansion, i.e., the differences are multiplied by 2, and thus the least significant bits (LSBs) of the differences can be used for embedding messages. The methods proposed in [8] and [9] can achieve better performance by applying DE to the prediction-errors.

Another well-known strategy for reversible data hiding is histogram shift (HS), in which the histogram of the image is used as the compressible feature because the distribution of the pixel values of an image is usually uneven. To compress the histogram, Ni et al. [10] proposed selecting a peak bin and a zero bin, and shifting the bins between them toward the zero bin by one step. Therefore, the bin neighboring to the peak bin is emptied out, which with the peak bin can be used to represent 1 and 0 respectively. It is easy to see that steeper histogram implies better compression rate, and usually the histogram of residuals is quite steep. Thus, most state-of-the-art methods apply histogram shift to residuals of the image [11,12].

According to the stage where distortion happens, we divide reversible data hiding into two types as follows. Type-I: all distortion is introduced in the stage of message embedding. Type-II: both compression stage and embedding stage will introduce some distortion to the cover, and compression stage is responsible for major distortion. The methods in [1,2,3,4,5,6] belong to Type-I; and yet both DE-based methods [7,8,9] and HS-based methods [10,11,12] belong to Type-II. Both types process the feature compression and data embedding in a separate way. In the present paper, we pay our attention to designing joint coding of compression and embedding for a binary feature sequence, which can be directly

used to improve Type-I methods. In fact, schemes in Type-II can be converted to Type-I, and we will discuss how to do such conversion in one of our coming papers.

For the Type-I reversible data hiding, the core problem is how to reversibly embed data into a compressible feature sequences with good performance. The performance is measured by embedding rate versus distortion, which is a special rate-distortion coding problem. A formal model for this problem has been established by Kalker and Willems [13]. In [13], the authors obtained the rate-distortion function, i.e., the upper bound of the embedding rate under a given distortion constraint, and they also proposed a joint compression and embedding code construction, called recursive code construction [13,14], which consists of a non-reversible data embedding code and a conditional compression code.

In this paper, we improve the recursive construction to approach the rate-distortion bound proved in [13]. In the novel construction, we use a data embedding code for all-zero covers and a backward extraction strategy, which enable us to design a very efficient conditional compression algorithm. The proposed codes can be directly applied to Type-I reversible data hiding schemes and significantly reduce the distortion for various given embedding rates.

The rest of this paper is organized as follows. The coding model, theoretical upper bound of embedding rate and recursive construction are briefly introduced in Section 2. The proposed coding method with the analysis of embedding rate versus distortion is elaborated in Section 3. The experiment results on improving RS schemes are given in Section 4. The paper is concluded with a discussion in Section 5.

2 Coding Model and Recursive Construction

2.1 Coding Model

Throughout this paper, we denote matrices and vectors by boldface fonts, and use the same notation for the random variable and its realization for simplicity. We denote the entropy by $H(X)$ and conditional entropy by $H(X|Y)$.

To do reversible data hiding, a compressible feature sequence should first be extracted from the original cover. For Type-I schemes, the features can usually be represented by a binary sequence. Therefore we directly take the binary feature sequence as the cover to discuss the coding method. A formal setup and theory limit for reversible data hiding into a compressible sequence have been established in [13] by Kalker and Willems, and we follow their notation.

Assume that there is a memoryless source producing binary compressible cover sequence $\mathbf{x} = (x_1, x_2, \dots, x_N)$ such that $x_i \in \{0, 1\}$ with the probability $P(x_i = 0) = p_0$ and $P(x_i = 1) = p_1, 1 \leq i \leq N$. The assumption of \mathbf{x} being compressible implies that the ratios of 0's and 1's are biased. Without loss of generality, we assume that $p_0 > 1/2$. We use Hamming distance to measure the embedding distortion on the cover \mathbf{x} . Because the message \mathbf{m} is usually compressed and encrypted before being embedded, we assume that the

message is a binary random sequence. If we can reversibly embed L -length message $\mathbf{m} = (m_1, m_2, \dots, m_L)$ into \mathbf{x} to get the marked cover $\mathbf{y} = (y_1, y_2, \dots, y_N)$ with d modifications on average, we define the embedding rate as $\rho = L/N$ and the distortion as $\Delta = d/N$. Furthermore, we define the embedding efficiency as $e = \rho/\Delta$, which means the average number of bits embedded per unit distortion. We hope to get high embedding efficiency for various given embedding rates.

A direct construction for reversible data hiding is proposed by Fridrich et al. [1] as follows. First compress the cover \mathbf{x} into a string $Comp(\mathbf{x})$ with a lossless compression algorithm $Comp(\cdot)$. The length of $Comp(\mathbf{x})$ is approximately equal to $NH(p_0)$. Therefore we can append $N(1 - H(p_0))$ bits of message \mathbf{m} after $Comp(\mathbf{x})$ to obtain $\mathbf{y} = Comp(\mathbf{x})||\mathbf{m}$. The recipient can extract the message \mathbf{m} from \mathbf{y} and reconstruct \mathbf{x} by decompressing $Comp(\mathbf{x})$. As the bits of $Comp(\mathbf{x})$ are uncorrelated with those of \mathbf{x} and the message \mathbf{m} is random, the expectation of distortion between \mathbf{x} and \mathbf{y} is 0.5. The embedding rate is equal to $(1 - H(p_0))$, which in fact is the maximum achievable embedding rate. If we only need to embed a shorter message with length equal to $\alpha N(1 - H(p_0))$ for some $\alpha < 1$, we can execute the above-mentioned method on a fraction α of the symbols in \mathbf{x} . In this case, the embedding rate $\rho = \alpha(1 - H(p_0))$ and the distortion $\Delta = \alpha/2$. Therefore, for the distortion constraint Δ , this simple method can achieve a rate-distortion line

$$\rho_{sim}(p_0, \Delta) = 2\Delta(1 - H(p_0)) . \quad (1)$$

Virtually, the simple method above is not optimal. In fact, this method achieves only a lower and fixed embedding efficiency $2(1 - H(p_0))$.

The maximum embedding rate achievable within the distortion constraint Δ is called the capacity under the distortion Δ . The following theorem proved by Kalker et al. [13] gives expression of the capacity.

Theorem 1. ^[13] *The reversible embedding capacity ρ_{rev} for a memoryless binary source with $p_0 \geq 1/2$ is, for $0 \leq \Delta \leq 1/2$, given by*

$$\rho_{rev}(p_0, \Delta) = H(\max(p_0 - \Delta, 1/2)) - H(p_0) \quad (2)$$

Note that the above bound can be increased for non-memoryless sequences, but we assume the binary cover is memoryless throughout this paper and this assumption in fact is suitable for most schemes.

2.2 Recursive Construction

To improve the embedding efficiency, Kalker et al. [13] proposed a recursive embedding method, which consists of a non-reversible embedding algorithm and a conditional compression algorithm. First select a non-reversible embedding code \mathcal{E} with distortion D and embedding rate R . Assume the binary cover sequence $\mathbf{x} = (x_1, x_2, \dots, x_N)$ is sufficiently long. The sequence is segmented into disjoint blocks of length K , such that $\mathbf{x} = \mathbf{x}_1||\mathbf{x}_2||\dots||\mathbf{x}_{N/K}$. Without loss of generality, we assume that N/K is a sufficiently large integer. By using the embedding code \mathcal{E} , KR bits of message \mathbf{m}_1 can be embedded into the first host block \mathbf{x}_1 ,

resulting to the first marked block \mathbf{y}_1 . The recipient can reconstruct \mathbf{x}_1 under the condition of known- \mathbf{y}_1 because she can observe \mathbf{y}_1 . Therefore the amount of information needed to reconstruct \mathbf{x}_1 is equal to $H(\mathbf{x}_1|\mathbf{y}_1)$, which means we can compress \mathbf{x}_1 into a sequence of length $H(\mathbf{x}_1|\mathbf{y}_1)$. This compressed sequence is embedded into the second block \mathbf{x}_2 , leaving room for $KR - H(\mathbf{x}_1|\mathbf{y}_1)$ bits of auxiliary message. Similarly, the information for reconstructing \mathbf{x}_2 is embedded into \mathbf{x}_3 . This process is continued recursively until \mathbf{x}_{K-1} . For the last block \mathbf{x}_K , the simple method described in Subsection 2.1 is used to complete a full reversible data hiding method. When N and N/K are large enough, the distortion of this method is equal to distortion of the code \mathcal{E} , and the embedding rate is equal to $R - H(\mathbf{x}_1|\mathbf{y}_1)/K$.

This recursive construction can achieve higher embedding efficiency than the simple method because of two key points: 1) the data is embedded by an efficient non-reversible embedding code; 2) the cover block is compressed under the condition of corresponding marked block. However the recursive construction proposed above cannot still approach the upper bound of embedding efficiency.

3 Improved Recursive Construction

3.1 Motivations and Overall Framework

In this section, we will improve the recursive construction to approach the upper bound of embedding efficiency for various embedding rates. To do that, we first observe the rate-distortion function (2), which shows that the maximum capacity is equal to $1 - H(p_0)$, and it can be achieved when distortion $\Delta = p_0 - 1/2$. In Fig. 1, we draw the rate-distortion lines for $p_0 = 0.7$ and 0.9 , which show that the capacity increases with distortion Δ for $0 \leq \Delta \leq p_0 - 1/2$, but keeps equal to $1 - H(p_0)$ for $p_0 - 1/2 < \Delta \leq 1/2$. Therefore, we only need to consider how to construct codes for $0 \leq \Delta \leq p_0 - 1/2$.

In Corollary 1 of [13], Kalker et al. proved that the optimal embedding manner for $0 \leq \Delta \leq p_0 - 1/2$ is that only the most probable symbol is allowed to be modified. In other words, only zeros are allowed to be changed if $p_0 \geq 1/2$.

Inspired by the observation above, we improve the recursive construction as follows. We only embed messages into zeros of the cover block $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,k})$ to obtain the marked block $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,k})$, and thus only zeros in \mathbf{x}_i will be modified for the i th block such that $1 \leq i \leq N/K - 1$. Therefore, for the position j such that $y_{i,j} = 0$, the corresponding $x_{i,j}$ must also be equal to 0. This property can be used to compress \mathbf{x}_i under the condition of known- \mathbf{y}_i . In fact, we can first delete the symbol $x_{i,j}$ in \mathbf{x}_i at position j such that $y_{i,j} = 0$ and obtain a subsequence of \mathbf{x}_i , denoted by \mathbf{x}'_i , and then compress \mathbf{x}'_i by a lossless compression algorithm $Comp()$. This method will extremely improve the compression rate because most symbols in \mathbf{x}_i have been compressed by deletion. The compressed \mathbf{x}'_i , denoted by $Comp(\mathbf{x}'_i)$, cascaded with an auxiliary message are embedded into the next block \mathbf{x}_{i+1} to get the next marked block \mathbf{y}_{i+1} . To extract the message and reconstruct the cover, the extraction process must be performed via a backward manner. To extract message from \mathbf{y}_i , we first extract

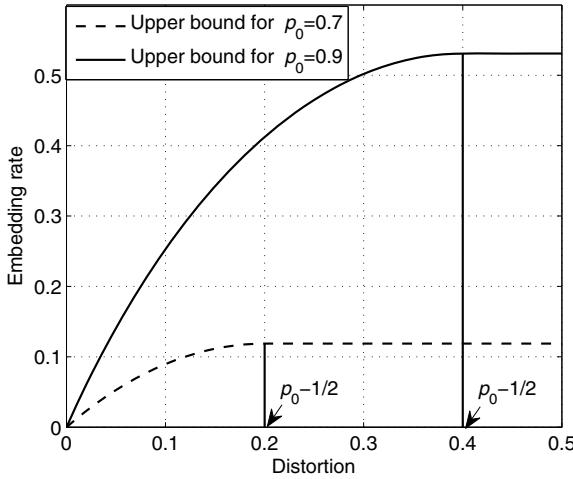


Fig. 1. Maximum capacity lines for $p_0 = 0.7$ and 0.9

message from \mathbf{y}_{i+1} and obtain \mathbf{x}'_i by decompression. Combining \mathbf{x}'_i and \mathbf{y}_i , we can reconstruct \mathbf{x}_i and know the positions of zeros in \mathbf{x}_i . According to these positions of zeros of \mathbf{x}_i , we can extract message from \mathbf{y}_i .

The detailed process and an example for embedding and extraction will be described in Subsection 3.3. We now deal with the first problem, that is, how to construct efficient codes for embedding data into an all-zero cover.

3.2 Data Embedding into All-Zero Covers

Data embedding into all-zero covers is just a special case of the coding model in Section 2 with taking $p_0 = 1$, which can be realized by a decompression algorithm, e.g., the reverse zero-run length (RZL) coding proposed by Wong et al. [6]. In this section, we proposed a more efficient embedding code by improving the RZL method.

Assume the cover sequence $\mathbf{x} = (x_1, x_2, \dots, x_N)$ is an N -length all-zero cover, which means every symbol $x_i = 0$ for $1 \leq i \leq N$. We want to embed a part of the message sequence $\mathbf{m} = (m_1, m_2, \dots, m_L, \dots)$ into \mathbf{x} . Two pointers, $P1$ and $P2$ are needed in the embedding process. $P1$ is used to label the last cover symbol that has been modified, and $P2$ is used to count the number of message bits that have been embedded. The following embedding construction is an rate-variable coding method, in which the embedding rate is determined by a parameter k , $k \geq 1$. First set $P1 = 0$ and $P2 = 0$. The encoder reads the message bit m_{P2+1} , and there are two embedding cases according to the value of m_{P2+1} .

Case 1. If $m_{P2+1} = 0$, set $P1 = P1 + 2^k$, $P2 = P2 + 1$ and one bit m_{P2+1} is embedded. In this case, no cover symbol is modified.

Case 2. If $m_{P2+1} = 1$, read the next k bits $(m_{P2+2}, \dots, m_{P2+k+1})$, which can be represented by a decimal integer belonging to $[0, 2^k - 1]$, denoted by $(m_{P2+2}, \dots, m_{P2+k+1})_{\text{int}}$. Set $P1 = P1 + (m_{P2+2}, \dots, m_{P2+k+1})_{\text{int}} + 1$, $P2 = P2 + k + 1$, and flip x_{P1} from “0” to “1”. Thus, $k + 1$ bits $(m_{P2+1}, \dots, m_{P2+k+1})$ are embedded, and only one cover symbol, x_{P1} , is modified.

For both cases, we have embedded the first $P2$ bits of message into the first $P1$ cover symbols. In the same manner, we continue to embed the rest message bits $(m_{P2+1}, m_{P2+2}, \dots)$ into the rest cover symbols (x_{P1+1}, \dots, x_N) , until $N - P1 < 2^k$. The obtained marked cover is denoted by $\mathbf{y} = (y_1, y_2, \dots, y_N)$.

To extract the message from \mathbf{y} , first set pointers $P1 = 0$ and $P2 = 0$. With $P1 + 1$ as the start point, read a 2^k length block from \mathbf{y} , i.e., $(y_{P1+1}, \dots, y_{P1+2^k})$. There are also two cases according to whether the block $(y_{P1+1}, \dots, y_{P1+2^k})$ includes “1”.

Case 1. If all symbols in $(y_{P1+1}, \dots, y_{P1+2^k})$ are equal to “0”, let the $(P2+1)$ th message bit $m_{P2+1} = 0$, $P1 = P1 + 2^k$, and $P2 = P2 + 1$.

Case 2. If there exists symbol “1” in $(y_{P1+1}, \dots, y_{P1+2^k})$, search for the first index i such that $y_{P1+i} = 1$ and $y_{P1+1} = y_{P1+2} = \dots = y_{P1+i-1} = 0$. The integer $i - 1$ can be represented by a binary sequence consisting of k bits, denoted by $(i - 1)_{\text{bin}}$. Let the $(P2+1)$ th message bit $m_{P2+1} = 1$, the next k bits $(m_{P2+2}, \dots, m_{P2+k+1}) = (i - 1)_{\text{bin}}$, and $P1 = P1 + i$, $P2 = P2 + k + 1$.

In the same manner, extract messages from the rest symbols (y_{P1+1}, \dots, y_N) until $N - P1 < 2^k$ 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 process of the method above.

Example 1. Take the parameter $k = 2$. Assume the cover is a 9-length all-zero cover, i.e. $N=9$, and the message consists of 7 bits, as shown in Fig. 2. To embed the message, first set pointers $P1 = 0$ and $P2 = 0$, and then do the following three steps.

Step 1. Read $m_1 = 0$, thus set $P1 = P1 + 2^2 = 4$, and $P2 = P2 + 1 = 1$.

Step 2. Read $m_{P2+1} = m_2$. Because $m_2 = 1$, read the next two message bits $(m_3, m_4) = (0, 1)$, which is interpreted as a decimal integer $(0, 1)_{\text{int}} = 1$. Set $P1 = P1 + (0, 1)_{\text{int}} + 1 = 6$, $P2 = P2 + k + 1 = 4$, and flip $x_{P1} = x_6$ to “1”.

Step 3. Read $m_{P2+1} = m_5$. Because $m_5 = 1$, read the next two message bits $(m_6, m_7) = (1, 0)$, interpreted as a decimal integer $(1, 0)_{\text{int}} = 2$. Set $P1 = P1 + (1, 0)_{\text{int}} + 1 = 9$, $P2 = P2 + k + 1 = 7$, and flip $x_{P1} = x_9$ to “1”. As $N - P1 = 9 - 9 = 0 < 4$, the embedding process stops. The marked cover is denoted by \mathbf{y} that is obtained by modifying the sixth and ninth bits of the cover \mathbf{x} .

To extract the message from \mathbf{y} , we first set $P1 = 0$ and $P2 = 0$, and then do the following three steps.

Index	1	2	3	4	5	6	7	8	9
Message m	0	1	0	1	1	1	0		
Cover x	0	0	0	0	0	0	0	0	0
Marked cover y	0	0	0	0	0	1	0	0	1
Embedding steps	Step 1			Step 2			Step 3		

Fig. 2. Example of data embedding into all-zero covers

Step 1. Read the first four bits $(y_1, y_2, y_3, y_4) = (0, 0, 0, 0)$ from **y**. Because this is an all-zero block, set $m_1 = 0$, $P1 = P1 + 4 = 4$, and $P2 = P2 + 1 = 1$.

Step 2. With $P1 + 1 = 5$ as the start point, read successive four bits $(y_5, y_6, y_7, y_8) = (0, 1, 0, 0)$. Because the first symbol “1” appears at the second position in this block, i.e., the index $i = 2$ and $(i - 1)_{\text{bin}} = (0, 1)$. Let $m_{P2+1} = m_2 = 1$ and $(m_3, m_4) = (i - 1)_{\text{bin}} = (0, 1)$. Set $P1 = P1 + i = 6$ and $P2 = P2 + 1 + k = 4$.

Step 3. Although $N - P1 = 9 - 6 = 3 < 4$, the extraction process will continue because the last three symbols of **y** includes a “1”. The “1” appears at the third position in $(y_7, y_8, y_9) = (0, 0, 1)$, so let the index $i = 3$ and thus $(i - 1)_{\text{bin}} = (1, 0)$. Thus, we extract the last three bits of messages such that $m_{P2+1} = m_5 = 1$ and $(m_6, m_7) = (i - 1)_{\text{bin}} = (1, 0)$.

In this example, we embed 7 bits of message into a 9-length cover with only 2 modifications. We denote this embedding method by \mathcal{E}_0 . To analyze the embedding rate and distortion of \mathcal{E}_0 , we investigate the two cases in the embedding process. In Case 1, we embed one bit into a 2^k -length cover without making any modification; in Case 2, we embed $k + 1$ bits of messages by expending $n = (m_{P2+2}, \dots, m_{P2+k+1})_{\text{int}} + 1$ cover symbols and one modification. Because the message block $(m_{P2+2}, \dots, m_{P2+k+1})$ is random, the probability $P(n = j) = 1/2^k$ for any $j \in \{1, 2, \dots, 2^k\}$, and thus the expectation of n is equal to

$$\frac{1}{2^k}(1 + 2 + \dots + 2^k) = \frac{2^k + 1}{2} . \quad (3)$$

Therefore in Case 2, we on average embed $k + 1$ bits into $(2^k + 1)/2$ cover symbols with one modification. Because $P(m_{P2+1} = 0) = P(m_{P2+1} = 1) = 1/2$, the two cases occur with equal probability. In summary, for one embedding step, the average number of embedded bits is equal to

$$N_{\text{mess}} = \frac{1}{2} \times 1 + \frac{1}{2} \times (k + 1) = \frac{k + 2}{2} . \quad (4)$$

The average number of expended cover symbols is equal to

$$N_{\text{cover}} = \frac{1}{2} \times 2^k + \frac{1}{2} \times \frac{2^k + 1}{2} = \frac{2^{k+1} + 2^k + 1}{4} , \quad (5)$$

and the average number of modifications is equal to

$$N_{\text{modi}} = \frac{1}{2} \times 0 + \frac{1}{2} \times 1 = \frac{1}{2} . \quad (6)$$

Therefore the embedding rate R and distortion D of code \mathcal{E}_0 can be calculated as follows

$$R_0 = \frac{N_{\text{mess}}}{N_{\text{cover}}} = \frac{2k+4}{2^{k+1} + 2^k + 1}, \quad D_0 = \frac{N_{\text{modi}}}{N_{\text{cover}}} = \frac{2}{2^{k+1} + 2^k + 1} . \quad (7)$$

3.3 Improved Recursive Construction

We use \mathcal{E}_0 as the non-reversible embedding code and design a corresponding compression algorithm to improve the recursive construction in [13,14]. Assume that the binary cover sequence $\mathbf{x} = (x_1, x_2, \dots, x_N)$ is generated from a memoryless source satisfying $P(x_i = 0) = p_0$ and $P(x_i = 1) = p_1$. Firstly we divide \mathbf{x} into N/K disjoint blocks of length K , such that $\mathbf{x} = \mathbf{x}_1 || \mathbf{x}_2 || \dots || \mathbf{x}_{N/K}$. In every block we only embed messages into zero symbols via the embedding code \mathcal{E}_0 . Therefore, we can on average embed Kp_0R_0 bits into the Kp_0 “0’s” of $\mathbf{x}_1 = (x_{1,1}, \dots, x_{1,K})$ by flipping Kp_0D_0 “0’s” to “1’s” because the code \mathcal{E}_0 has embedding rate R_0 and distortion D_0 , yielding the first marked block $\mathbf{y}_1 = (y_{1,1}, \dots, y_{1,K})$. Therefore, at the position j , such that $1 \leq j \leq K$ and $y_{1,j} = 0$, the corresponding $x_{1,j}$ must also be equal to “0”, because no “1” in \mathbf{x}_1 has been flipped to “0” in \mathbf{y}_1 . We use this property to compress \mathbf{x}_1 under the condition of known- \mathbf{y}_1 by deleting all symbols in \mathbf{x}_1 at position j ’s such that $y_{1,j} = 0$, resulting to a subsequence of \mathbf{x}_1 . Denote this subsequence by \mathbf{x}'_1 , and thus $\mathbf{x}'_1 = \{x_{1j} | 1 \leq j \leq K, y_{1j} = 1\}$. The ratio of zeros in \mathbf{y}_1 is equal to the ratio of non-modified zeros of \mathbf{x}_1 , that is $p_0(1 - D_0)$. Thus, the ratio of ones in \mathbf{y}_1 is equal to $1 - p_0(1 - D_0) = p_1 + p_0D_0$, and the average length of \mathbf{x}'_1 is equal to $K(p_1 + p_0D_0)$. In other words, under the condition of known- \mathbf{y}_1 , the block \mathbf{x}_1 is compressed to \mathbf{x}'_1 with compression rate $p_1 + p_0D_0$, and we can reconstruct \mathbf{x}_1 by replacing ones of \mathbf{y}_1 by the symbols of \mathbf{x}'_1 .

Furthermore we compress \mathbf{x}'_1 with a lossless compression algorithm $\text{Comp}(\cdot)$, e.g., an arithmetic coder. Denote the ratio of zeros and ones in \mathbf{x}'_1 by q_0 and q_1 respectively, which can be easily computed as follows.

$$q_0 = \frac{p_0D_0}{p_1 + p_0D_0}, \quad q_1 = \frac{p_1}{p_1 + p_0D_0} . \quad (8)$$

Therefore \mathbf{x}'_1 can be compressed with the rate about equal to $H(q_0)$. In summary, when \mathbf{y}_1 is known, the amount of information needed to reconstruct \mathbf{x}_1 is equal to

$$K(p_1 + p_0D_0)H(q_0) . \quad (9)$$

By using the code \mathcal{E}_0 , the compressed information $\text{Comp}(\mathbf{x}'_1)$ is embedded into the zeros of the next block \mathbf{x}_2 , leaving room for $K(p_0R_0 - (p_1 + p_0D_0)H(q_0))$ bits of auxiliary message and resulting to the second marked block \mathbf{y}_2 . The information for reconstructing \mathbf{x}_2 , denoted by $\text{Comp}(\mathbf{x}'_2)$ is embedded into the third block \mathbf{x}_3 . This process is continued recursively until the one but the last block

$\mathbf{x}_{N/K}$. For the last block $\mathbf{x}_{N/K}$, we directly compress the block and make room for $K(1 - H(p_0))$ bits, into which we embed $Comp(\mathbf{x}'_{N/K-1})$ for reconstructing the second last block $\mathbf{x}_{N/K-1}$. In addition, we also embed the overhead information, such as the value of p_0 , the block length K , and the parameter k used by code \mathcal{E}_0 , into the last block.

To extract the message and reconstruct the cover, the extraction process must be performed in a backward manner. To extract messages from the i th block \mathbf{y}_i , for $1 \leq i \leq N/K - 1$, we must first extract messages from \mathbf{y}_{i+1} and obtain \mathbf{x}'_i by decompression. Combining \mathbf{x}'_i and \mathbf{y}_i , we can reconstruct \mathbf{x}_i and know the positions of zeros in \mathbf{x}_i , according to which we can extract messages from \mathbf{y}_i by using the code \mathcal{E}_0 .

When N and N/K are large enough, the embedding rate and distortion of the method above can be estimated by ρ_{rec} and D_{rec} such that

$$\rho_{rec} = p_0 R_0 - (p_1 + p_0 D_0) H(q_0), \quad D_{rec} = p_0 D_0 . \quad (10)$$

When varying the parameter k in the code \mathcal{E}_0 , we can get variable rate (R_0, D_0) by Eq. (7), and thus yield rate-variable recursive construction with embedding rate $\rho_{rec}(k)$ and distortion $D_{rec}(k)$ as follows.

$$\rho_{rec}(k) = p_0 \frac{2k+4}{2^{k+1} + 2^k + 1} - \left(p_1 + p_0 \frac{2}{2^{k+1} + 2^k + 1} \right) H(q_0), \quad (11)$$

$$D_{rec}(k) = p_0 \frac{2}{2^{k+1} + 2^k + 1}, \quad k \geq 1 . \quad (12)$$

Now we use an example with only two blocks to illustrate the embedding and extraction process of the method described above.

Example 2. This example is based on Example 1. As shown in Fig.3, the first cover block \mathbf{x}_1 consists of ten symbols with only one “1”. The first seven message bits are the same as in Example 1, which are embedded into the zeros of \mathbf{x}_1 and generate a nine-length marked block as we have obtained in Example 1. We denote this marked block in interim step by \mathbf{y}'_1 . Replace zeros of \mathbf{x}_1 by \mathbf{y}'_1 and generate the ultimate marked block \mathbf{y}_1 . Denote the index set of 1's in \mathbf{y}_1 by Ind_1 , and thus $Ind_1 = \{3, 7, 10\}$, according to which we extract bits from \mathbf{x}_1 and get $\mathbf{x}'_1 = (x_3, x_7, x_{10}) = (1, 0, 0)$. The sequence \mathbf{x}'_1 is compressed to $Comp(\mathbf{x}'_1)$ and then is embedded into the second block.

To reconstruct the cover block and extract messages from the marked block \mathbf{y}_1 , we first count the number of ones in \mathbf{y}_1 that is equal to 3. Second, we extract messages from the second marked block and decompress the extracted messages successively until we get a 3-length decompressed sequence which is just \mathbf{x}'_1 . Thus, we can reconstruct \mathbf{x}_1 by replacing the ones of \mathbf{y}_1 by \mathbf{x}'_1 . After that we know the index set of zeros in \mathbf{x}_1 such that $Ind_0 = \{1, 2, 4, 5, 6, 7, 8, 9, 10\}$, according to which we extract bits from \mathbf{y}_1 and get the sequence \mathbf{y}'_1 . Finally, we can extract the seven message bits from \mathbf{y}'_1 by using the code \mathcal{E}_0 .

Index	1	2	3	4	5	6	7	8	9	10	Second Block
message	0	1	0	1	1	1	0
\mathbf{y}'_1	0	0	0	0	0	0	1	0	0	1	
\mathbf{x}'_1	0	0	1	0	0	0	0	0	0	0	...
\mathbf{y}_1	0	0	1	0	0	0	1	0	0	1	$\text{Comp}(\mathbf{x}'_1)...$
\mathbf{x}'_1		1				0			0		
$\text{Comp}(\mathbf{x}'_1)$											

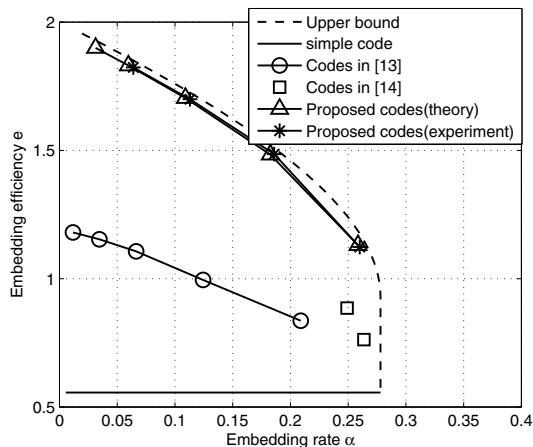
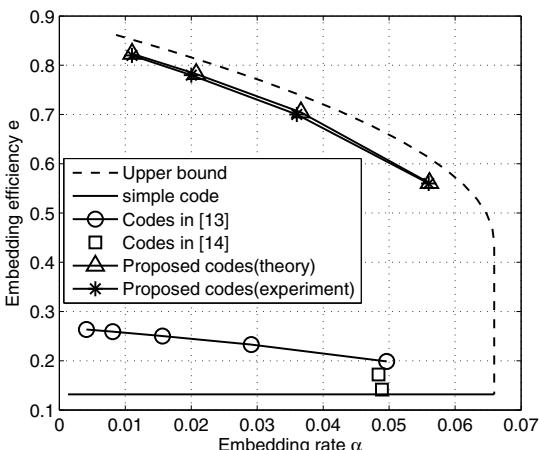
Fig. 3. Example of improved recursive construction

3.4 Performance Comparison

We compare the coding method above with the codes proposed in [13] and [14]. In the original recursive construction [13], Kalker and Willems used Hamming matrix embedding [16] as the non-reversible data embedding code, by which we can embed k bits of message into $2^k - 1$ cover bits by at most one modification. The Hamming codes modify zeros and ones with equal probability. Maas et al. [14] improved the original recursive construction by adjusting Hamming code to change more zeros than ones for the case $k = 2$.

Both theoretical and simulation results of the proposed method are compared with the codes in [13,14] for $p_0 = 0.8$ and 0.65 . The simulation results are obtained by embedding random messages into a 2^{16} -length cover. In the experiments, we set the length of cover blocks $K=200$, and an adaptive arithmetic coder [15] is used as the compression algorithm $\text{Comp}()$. We compare the codes by using the measurement of embedding efficiency versus embedding rate. As shown in Fig.4, the proposed codes significantly outperform the codes presented in [13,14]. For small p_0 ($p_0 = 0.6$), the embedding efficiency of codes in [13,14] is close to that of simple codes, while the embedding efficiency of proposed codes is still close to the upper bound. However, we note that the proposed codes with small parameter k will perform poor when the value of p_0 decreases. Therefore we generate codes for $p_0 = 0.8$ by using $k = 1, 2, 3, 4, 5$, while for $p_0 = 0.65$ we only use $k = 2, 3, 4, 5$. The suitable coding parameters for $p_0 \in [0.54, 1]$ are proposed in Table 1. When p_0 is smaller than 0.54, the minimal parameter k should further increase, but the capacity for such cases usually is too small for practical applications.

Note that this construction can not reach the upper bound of embedding efficiency because \mathcal{E}_0 is not optimal. If we use the decompressor or an optimal compression algorithm as coding method for all-zero covers, we have $R_0 = H(D_0)$ and then it is easy to prove that Eq. (10) is equivalent to Eq. (2). Therefore, the proposed construction in fact is optimal for reversible data hiding.

(a) $p_0 = 0.8$ (b) $p_0 = 0.65$ **Fig. 4.** Comparison of embedding efficiency vs. embedding rate between the proposed codes and codes in [13, 14]**Table 1.** Coding parameter k according to p_0

p_0	$[0.66, 1]$	$[0.58, 0.66)$	$[0.54, 0.58)$...
k	≥ 1	≥ 2	≥ 3	...

4 Applications in Type-I Schemes

The coding method above can be directly applied to data hiding schemes that belong to Type-I, such as the schemes in [1,2,3,4,5,6]. Taking the regular-singular (RS) method in [1] as an example, we illustrate the ability of the proposed codes for reducing embedding distortion.

The RS method [1] is proposed for spatial images by constructing compressible features based on texture complexity. Assume the cover is a 8-bit grayscale image. The image first is divided into small groups, e.g., n pixels per group. A permutation F is used to flip the gray values, the amplitude of which is controlled by a positive inter A . For instance, when $A = 1$, the flipping function is as follows:

$$F : 0 \leftrightarrow 1, 2 \leftrightarrow 3, 4 \leftrightarrow 5, \dots, 254 \leftrightarrow 255 . \quad (13)$$

For a pixel group $G = (x_1, \dots, x_n)$, $F(G) = (F(x_1), \dots, F(x_n))$. A distinguishing function f is used to detect the changing direction of the variation of the group.

$$f(G) = \sum_{i=1}^{n-1} |x_{i+1} - x_i| . \quad (14)$$

By using the functions F and f , the pixel group can be defined as regular (R) , singular (S), or unusable (U) such that

$$\begin{aligned} G \in R &\Leftrightarrow f(F(G)) > f(G) \\ G \in S &\Leftrightarrow f(F(G)) < f(G) . \\ G \in U &\Leftrightarrow f(F(G)) = f(G) \end{aligned} \quad (15)$$

For typical picture, adding some noise will lead to an increase of the variation, so we expect a bias between the number of regular groups and the number of singular groups. By assigning a “0” to a regular group and a “1” to a singular group, we can generate a binary cover sequence satisfying $p_0 > 1/2$. Flipping between “0” to “1” can be realized by applying F to the corresponding pixel group.

Usually larger amplitude A implies larger capacity but also larger embedding distortion. In our experiments, we set A from 1 to 4, and the group size $n = 4$. For each value of A , we embed messages with the original RS method and the proposed codes into 10 test images [17] with size of 512×512 (see Fig.5), and calculate the average embedding rate and average PSNR. We observed that the ratio of zeros p_0 in the RS sequence varies from 0.54 to 0.87. In our coding method, we use the minimal parameter k according to p_0 as proposed in Table



Fig. 5. Text images with size of 512×512

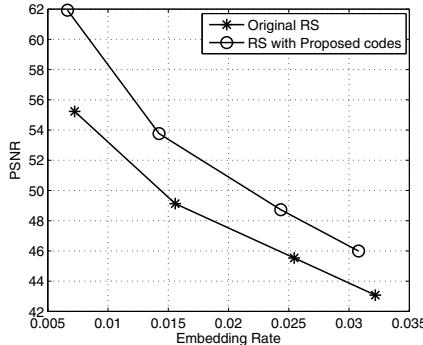


Fig. 6. Experimental results on improving RS method

- As shown in Fig.6, the proposed codes can significantly increase the PSNRs for various embedding rates. Herein, the embedding rate is defined as bits carried by per pixel (bpp).

5 Conclusion

Most state-of-the-art reversible data hiding schemes use a strategy with separate processes of feature compression and message embedding. Kalker and Willems noted that higher embedding rate under given distortion constraint may be achieved by using joint encoding of feature compression and message embedding and thus proposed the recursive code construction. In this paper we improve the recursive construction by using not only the joint encoding above but also a joint decoding of feature decompression and message extraction. The improved codes can approach the capacity bound and significantly outperform previous codes [13,14] by embedding efficiency.

The current codes are designed for embedding data into a biased 0-1sequence. These kinds of codes cannot be directly used for the Type-II schemes such as DE-based schemes and HS-based schemes, because the Type-II schemes generate a binary feature sequence in a special compression manner that accounts for the majority of the distortion. In one of our subsequent papers, we will discuss how to convert the Type-II schemes to fit the coding model established by Kalker and Willems [13] and apply the codes propose in the present paper to improve these schemes.

Acknowledgments. This work was supported by the Natural Science Foundation of China (60803155) and the National Science and Technology Major Project (No.2010ZX03004-003). The authors would also like to sincerely thank Dr. Tomáš Filler and anonymous reviewers for their valuable comments.

References

1. Fridrich, J., Goljan, M.: Lossless Data Embedding for All Image Formats. In: Proc. of EI SPIE, Security and Watermarking of Multimedia Contents IV, San Jose, vol. 4675, pp. 572–583 (2002)
2. Celik, M.U., Sharma, G., Tekalp, A.M., Saber, E.: Lossless Generalized-LSB Data Embedding. IEEE Trans. on Image Processing 14(2), 253–266 (2005)
3. Xuan, G., Shi, V., Chai, P., et al.: Reversible Binary Image Data Hiding by Run-Length Histogram Modification. In: 19th International Conference on Pattern Recognition, ICPR 2008 (2008)
4. Li, S., Kot, A.C.: Privacy Protection of Fingerprint Database Using Lossless Data Hiding. In: Proceedings of the 2010 IEEE International Conference on Multimedia and Expo., pp. 1293–1298 (2010)
5. Du, R., Fridrich, J.: Lossless Authentication of MPEG-2 Video. In: Proc. of IEEE International Conference on Image Processing, vol. 2, pp. 893–896 (2002)
6. Wong, K., Tanaka, K., Takagi, K., Nakajima, Y.: Complete Video Quality-Preserving Data Hiding. IEEE Trans. on Circuits and Systems for Video Technology 19(10), 1499–1512 (2009)
7. Tian, J.: Reversible Data Embedding Using a Difference Expansion. IEEE Trans. Circuits Syst. Video Technol. 13(8), 890–896 (2003)
8. Thodi, D.M., Rodriguez, J.J.: Expansion Embedding Techniques for Reversible Watermarking. IEEE Trans. Image Process. 16(3), 721–730 (2007)
9. Hu, Y., Lee, H.-K., Li, J.: DE-based Reversible Data Hiding with Improved Overflow Location Map. IEEE Trans. Circuits Syst. Video Technol. 19(2), 250–260 (2009)
10. Ni, Z., Shi, Y.Q., Ansari, N., Wei, S.: Reversible Data Hiding. IEEE Trans. Circuits Syst. Video Technol. 16(3), 354–362 (2006)
11. Tsai, P., Hu, Y.C., Yeh, H.L.: Reversible Image Hiding Scheme Using Predictive Coding and Histogram Shifting. Signal Process. 89, 1129–1143 (2009)
12. Luo, L.X., Chen, Z.Y., Chen, M., et al.: Reversible Image Watermarking Using Interpolation Technique. IEEE Trans. Inf. Forensics and Security 5(1), 187–193 (2010)
13. Kalker, T., Willem, F.M.: Capacity Bounds and Constructions for Reversible Data-Hiding. In: Proc. of 14th International Conference on Digital Signal Processing, DSP 2002, pp. 71–76 (2002)
14. Maas, D., Kalker, T., Willem, F.M.: A Code Construction for Recursive Reversible Data-Hiding. In: Proc. Multimedia and Security Workshop at ACM Multimedia, Juan-les-Pins, France (December 6, 2002)
15. Sayood, K.: Introduction to Data Compression, pp. 87–94. Morgan Kaufmann Publishers, San Francisco (1996)
16. Crandall, R.: Some Notes on Steganography. Posted on steganography mailing list (1998), <http://os.inf.tu-dresden.de/~westfeld/crandall.pdf>
17. Miscellaneous gray level images,
<http://decsai.ugr.es/cvg/dbimagenes/g512.php>