Multimed Tools Appl
DOI 10.1007/s11042-013-1493-x

Recursive code construction for reversible data hiding in DCT domain

Biao Chen · Weiming Zhang · Kede Ma · Nenghai Yu

Abstract Reversible data hiding has extensive applications in fields like data authentication, medical data management and error concealment. In this paper, we formulate the model of reversible data hiding over a special ternary cover that is suitable for any transform domain, such as DCT domain, where the probability density function of the transformed coefficients has a Laplacian distribution with a small variance. After deriving rate-distortion function for this model, we propose a code construction that can approach the rate-distortion bound. Based on the code construction, a reversible data hiding method for JPEG images is proposed. Experimental results demonstrate that proposed method has a good balance among image quality, filesize increment and computation time. The excellent performance of proposed method also demonstrate the power of our code construction for reversible data hiding on DCT based media.

Keywords Reversible data hiding · Ternary cover · Recursive code construction · JPEG · DCT domain

This work was supported in part by the Natural Science Foundation of China under Grant 61170234 and Grant 60803155, by the Strategic and Piloted Project of CAS under Grant XDA06030601, and by the funding of Science and Technology on Information Assurance Laboratory under Grant KJ-13-02.

B. Chen
Department of Electronic Engineering and Information Science,
University of Science and Technology of China, Hefei 230027, China
e-mail: chb893@mail.ustc.edu.cn

W. Zhang (✉) · K. Ma · N. Yu
Department of Information Science, University of Science and Technology of China,
Hefei 230027, China
e-mail: weimingzhang@yahoo.cn

K. Ma
e-mail: kdma@mail.ustc.edu.cn

N. Yu
e-mail: ynh@ustc.edu.cn

Published online: 17 May 2013

1 Introduction

As a technique that embeds secret message into cover signal, information hiding has been widely applied in areas such as covert communication, copyright protection and media annotation [19]. Reversible data hiding (RDH) is one kind of information hiding technique with the characteristic that not only the secret message needs to be precisely extracted, but also the cover itself should be losslessly restored. This property is important in some special scenarios such as medical imagery, military imagery. In these applications, the cover is too precious or too important to be damaged [6].

Since first introduced in a Kodak's patent [9], a plenty of reversible data hiding algorithms have been proposed in the past decade. In a technical review [2], Caldelli et al. introduced some most typical RDH algorithms. Classical RDH algorithms roughly fall into three categories. The first kind of algorithms follow the idea of compression-embedding framework, which was first introduced by Fridrich [8]: in these algorithms, the image is segmented into pixel groups and a binary feature value is calculated for each pixel group. These binary feature values form a compressible sequence and message can be embedded in the extra space left by lossless compression. Celik made an extension of this work in [3]. The second category is based on difference expansion (DE) [10, 20, 21], in which the difference of a pair of neighboring pixels is expanded, e.g., multiplied by 2, and thus the least significant bits (LSBs) of the differences are all-zero and can be used for embedding message. The last kind of RDH algorithms are based on histogram shift (HS) [17]. The histogram of one special feature (for example, grayscale value) for natural image is quite uneven, which implies that the histogram can be compressed for embedding data. For instance, some space can be saved for watermark by shifting the bins of histogram. In fact, better performance can be achieved by applying DE or HS to residual part of images, e.g., the predicted errors [16, 22]. Recently, researchers are interested in adaptive RDH methods, which choose the cover image's flat area to embed message and avoid area with much detail, or to embed more bits into flat area and less bits into complex area[14, 18]. Doing like these is for the sake of minimizing the overall distortion under a given payload.

Though many algorithms have been proposed in the past decade, few theoretical analysis about reversible data hiding have been published yet. Kalker and Willems [12] have made a fundamental contribution for this kind of work. In their paper, they established the framework of reversible data hiding as a rate-distortion problem, and in the case of binary cover, they deduced explicit expression for the rate-distortion bound. They also proposed a recursive code construction for RDH in binary covers, but their method can not approach the bound.

In our previous work [26], we proposed capacity-approaching codes for reversible data hiding in binary covers. These codes are proved to be quite effective in preserving stego image's visual quality. Whenever a grayscale RDH problem can be converted to a binary cover model, the codes can be adopted to reduce embedding distortion and to improve stego image's visual quality. In our extended paper [27], the codes are adopted to improve Fridrich's RDH method for JPEG images [7], but both in [7] and in [27] the maximum achievable payload is disappointingly small.

So far, there are only a few RDH methods proposed for JPEG images, or more generally, for DCT (Discrete Cosine Transform) domain cover. Since DCT is used as a core component in mainstream media compression standards (JPEG for image compression and H.264 for video compression), there are a spectrum of applications of RDH in DCT domain.

In [25], Zhang et al. proposed a technique to authenticate the integrity of JPEG image with the help of RDH. In his method, the JPEG image is divided into blocks, and for each block the hashed DCT coefficients are embedded into other blocks, which is used to identify whether one block has been tampered. Because the purpose of this technique is to protect the integrity of images, it is required that the data embedding operation should be harmless to the original image covers.

Recently, it has been found that reversible data hiding can also be quite helpful in video error-concealment coding. Chung et al. [5] proposed an error resilient video coding algorithm for H.264 videos, in which the motion vector of a block is embeded into the DCT coefficients of a neighboring block. To avoid image quality degradation caused by data embedding, they embed the data in a reversible way. It is shown in the experiments that the visual quality of the original video is highly improved since the use of RDH.

In [11], Hwang et al. suggested some mechanisms for secure cloud computing. For example, the cloud server can embed some authentication information into the data, claiming the data's owner, uploading time, etc. However, the server is prohibited to change the user's data in any case, so RDH for DCT domain is suitable to manage JPEG images or compressed videos. In another instance of media management scenerio, Wong and Tanaka [23] combined reversible watermarking with image scrambling. In their method, they embedded data into DC components of a JPEG image and achieved image scrambling at the same time. The embedded data contains basic information about the image to support media management, and the scrambling process helps protect image's content.

In this paper, we extend the idea in our previous work to a special ternary cover model that fits DCT based media, like JPEG images and compressed videos. First we deduce an explicit expression of rate-distortion bound for such cover, and then propose a code construction that can approach the bound. Furthermore, we apply the proposed coding scheme to RDH for JPEG images to improve the performance.

The rest of the paper is organized as follows. Section 2 establishes the model and rate-distortion bound for RDH in ternary cover. An optimal code construction for this model is elaborated in Section 3. Further, a case study is given that applies the code construction to RDH for JPEG images in Section 4, and finally, we conclude this paper with a discussion in Section 5.

2 A ternary-cover model for reversible data hiding

In this section, we will formulate the model of a special ternary cover, for which the rate-distortion bound of RDH will be presented with a proof. We denote the entropy by $H(X)$. Specially, the binary entropy function is denoted by $H_2(p)$ for $0 \leq p \leq 1$, and the ternary entropy function is denoted by $H_3(p_1, p_2, p_3)$ for $0 \leq p_1, p_2, p_3 \leq 1$ and $p_1 + p_2 + p_3 = 1$.

2.1 Model of RDH in a special ternary cover

Assume that the cover is a memoryless ternary sequence $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, with $x_i \in \{-1, 0, 1\}$ and the probability mass function (pmf) $P_X(0) = p_0$, $P_X(1) = p_+$, $P_X(-1) = p_-$. We only consider the case that the percentages of 1 and -1 in $\mathbf{x}$ are quite small, in fact, it is assumed that both p_+ and p_- are less than 1/3 in our formulation, i.e., $p_+ \leq 1/3$ and $p_- \leq 1/3$. Without loss of generality, we assume that $p_+ \leq p_-$. (If $p_+ > p_-$, the conclusion is similar). One typical example of this cover model is the quantized DCT coefficients with values equal to -1, 0 and 1 (see Fig. 7), so RDH on this model can be applied to DCT compression based media, such as JPEG images or compressed videos.

We'll embed messages into $\mathbf{x}$ in a reversible manner. The secret message $\mathbf{m} = (m_1, m_2, \cdots, m_L)$ is a binary random sequence with $m_i \in \{0, 1\}$. $\mathbf{m}$ is embedded into $\mathbf{x}$ and the corresponding stego is $\mathbf{y} = (y_1, y_2, \cdots, y_N)$. Denote the pmf of stego by P_Y, and the transition pmf from cover to stego by $P_{Y|X}$. The embedding distortion is measured by square error distortion, i.e., $d(a, b) = (a - b)^2$. We further define the embedding rate as $\rho = L/N$ and the average embedding distortion as $D = \sum_{x,y} P_X(x) P_{Y|X}(y|x) d(x, y)$.

It's preferable to achieve both high embedding rate and low distortion, but in most circumstances these two targets just conflict with each other. Thus, the question arises that how to maximize the embedding rate under given distortion constraint? Thanks to previous research [12], this rate-distortion problem can be solved with information theory.

2.2 Rate-distortion curve

It has been proved by Kalker et al. [12] that, for any given distortion constraint Δ, the reversible embedding capacity, i.e., the maximum embedding rate, is

$$\rho(\Delta) = \max H(Y) - H(X), \tag{1}$$

in which the maximum is over all test channel $P(Y|X)$ s.t.

$$\sum_{x,y} P_X(x) P_{Y|X}(y|x) d(x, y) \leq \Delta. \tag{2}$$

Now from (1), we will derive the explicit expression of $\rho(\Delta)$ for the special ternary cover model mentioned in Section 2.1.

Theorem 1 *For a ternary memoryless source with* $p_+ \leq p_- \leq 1/3$, *the embedding capacity for* $0 \leq \Delta \leq p_0 - 1/3$ *is given by*

$$\rho(\Delta) = \begin{cases} H_3(p_0 - \Delta, p_+ + \Delta, p_-) - H_3(p_0, p_+, p_-), & \textit{if}\ 0 \leq \Delta \leq p_- - p_+ \\ H_2(p_0 - \Delta) - H_3(p_0, p_+, p_-) + 1 - p_0 + \Delta, & \textit{if}\ p_- - p_+ < \Delta \leq p_0 - 1/3 \end{cases} \tag{3}$$

Proof Equation (1) implies that deriving the rate-distortion function is an optimization problem, which maximizes

$$-\sum_{y=-1}^{1}\left(\sum_{x=-1}^{1}P_X(x)P_{Y|X}(y|x)\right)\log_2\left(\sum_{x=-1}^{1}P_X(x)P_{Y|X}(y|x)\right)$$
$$-H_3\left(p_0,p_+,p_-\right), \tag{4}$$

s.t.:

$$D=\sum_{x,y}P_X(x)P_{Y|X}(y|x)d(x,y)=\Delta, \tag{5}$$

$$\sum_{y=-1}^{1}P_{Y|X}(y|x)=1,\quad \forall x\in\{-1,0,1\}, \tag{6}$$

$$P_{Y|X}(y|x)\geq 0,\quad \forall x,y\in\{-1,0,1\}. \tag{7}$$

We can use the method of Lagrange multipliers to find the optimal solution by setting up the functional

$$J=-\sum_{y=-1}^{1}\left(\sum_{x=-1}^{1}P_X(x)P_{Y|X}(y|x)\right)\log_2\left(\sum_{x=-1}^{1}P_X(x)P_{Y|X}(y|x)\right)$$
$$-H_3\left(p_0,p_+,p_-\right)-\lambda\sum_{x,y}P_X(x)P_{Y|X}(y|x)d(x,y)$$
$$-\sum_{x=-1}^{1}u_x\sum_{y=-1}^{1}P_{Y|X}(y|x)-\sum_{x=-1}^{1}\sum_{y=-1}^{1}v_{x,y}P_{Y|X}(y|x). \tag{8}$$

Differentiating with respect to $P_{Y|X}(y|x)$, we have

$$\frac{\partial J}{\partial P_{Y|X}(y|x)}=-\,P_X(x)\log_2\left(\sum_{i=-1}^{1}P_X(x)P_{Y|X}(y|i)\right)-P_X(x)\log_2 e$$
$$-\lambda P_X(x)d(x,y)-u_x-v_{x,y}=0,\quad \forall x,y\in\{-1,0,1\}. \tag{9}$$

Solving the series of (9) subject to constraints (5)–(7), we can get the optimal conditional pmf $P^*_{Y|X}$ and the optimal pmf P^*_Y, which are formulated as two cases.

□

Case 1 When $\log_2\left(\frac{1}{p_-}-2\right)\leq\lambda\leq\log_2\left(\frac{p_0}{p_+}\right)$,

$$P^*_{Y|X}(y|x)=\begin{cases}\frac{p_0-2^\lambda p_+}{(1+2^\lambda)p_0}, & \text{if } (x,y)=(0,1)\\ \frac{2^\lambda(p_0+p_+)}{(1+2^\lambda)p_0}, & \text{if } (x,y)=(0,0)\\ 1, & \text{if } (x,y)=(1,1) \text{ or } (-1,-1)\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

$$P^*_Y(0)=\frac{2^\lambda(p_++p_0)}{1+2^\lambda},\quad P^*_Y(1)=\frac{(p_++p_0)}{1+2^\lambda},\quad P^*_Y(-1)=p_-, \tag{11}$$

and

$$\Delta = \frac{p_0 - 2^{\lambda} p_+}{1 + 2^{\lambda}}. \tag{12}$$

Case 2 When $0 \le \lambda < \log_2\left(\frac{1}{p_-} - 2\right)$,

$$P^*_{Y|X}(y|x) = \begin{cases} \frac{1}{(2+2^{\lambda})p_0} - \frac{p_+}{p_0}, & \text{if } (x, y) = (0, 1) \\ \frac{1}{(2+2^{\lambda})p_0} - \frac{p_-}{p_0}, & \text{if } (x, y) = (0, -1) \\ \frac{1}{p_0} - \frac{2}{(2+2^{\lambda})p_0}, & \text{if } (x, y) = (0, 0) \\ 1, & \text{if } (x, y) = (1, 1) \text{ or } (-1, -1) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$P^*_Y(0) = \frac{2^{\lambda}}{2 + 2^{\lambda}}, \quad P^*_Y(1) = \frac{1}{2 + 2^{\lambda}}, \quad P^*_Y(-1) = \frac{1}{2 + 2^{\lambda}} \ , \tag{14}$$

$$\Delta = \frac{2}{2 + 2^{\lambda}} - p_+ - p_- \ . \tag{15}$$

By (12) and (15), we can replace the parameter λ with Δ, and then rewrite the optimal solutions as follows.

Case 1 When $0 \le \Delta \le p_- - p_+$,

$$P^*_{Y|X}(y|x) = \begin{cases} \frac{\Delta}{p_0}, & \text{if } (x, y) = (0, 1) \\ 1 - \frac{\Delta}{p_0}, & \text{if } (x, y) = (0, 0) \\ 1, & \text{if } (x, y) = (1, 1) \text{ or } (-1, -1) \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$P^*_Y(0) = p_0 - \Delta, \quad P^*_Y(1) = p_+ + \Delta, \quad P^*_Y(-1) = p_-. \tag{17}$$

In this case, the rate-distortion function is given by

$$\begin{aligned} & H_3\left(P^*_Y(0),\ P^*_Y(1), P^*_Y(-1)\right) - H_3(p_0, p_+, p_-) \\ & = H_3\left(p_0 - \Delta,\ p_+ + \Delta, p_-\right) - H_3(p_0, p_+, p_-). \end{aligned} \tag{18}$$

Case 2 When $p_- - p_+ < \Delta \le p_0 - \frac{1}{3}$,

$$P^*_{Y|X}(y|x) = \begin{cases} \frac{\Delta + p_- - p_+}{2p_0}, & \text{if } (x, y) = (0, 1) \\ \frac{\Delta + p_+ - p_-}{2p_0}, & \text{if } (x, y) = (0, -1) \\ 1 - \frac{\Delta}{p_0}, & \text{if } (x, y) = (0, 0) \\ 1, & \text{if } (x, y) = (1, 1) \text{ or } (-1, -1) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

$$P^*_Y(0) = p_0 - \Delta, \quad P^*_Y(1) = \frac{\Delta + p_- + p_+}{2}, \quad P^*_Y(-1) = \frac{\Delta + p_- + p_+}{2} \tag{20}$$

In this case, the rate-distortion function is given by

$$
\begin{aligned}
&H_3\left(P_Y^*(0),\ P_Y^*(1),\ P_Y^*(-1)\right) - H_3(p_0, p_+, p_-) \\
&\quad = H_3\left(p_0 - \Delta,\ \frac{\Delta + p_- + p_+}{2},\ \frac{\Delta + p_- + p_+}{2}\right) - H_3(p_0, p_+, p_-) \\
&\quad = H_2(p_0 - \Delta) + (\Delta + p_- + p_+) H_2\left(\frac{1}{2}\right) - H_3(p_0, p_+, p_-) \\
&\quad = H_2(p_0 - \Delta) - H_3(p_0, p_+, p_-) + 1 - p_0 + \Delta\ . \qquad (21)
\end{aligned}
$$

When taking $\Delta = p_0 - 1/3$ into (21), we just get the maximum possible capacity, $\rho_{\max}$, because

$$
\begin{aligned}
&H_2\left(\frac{1}{3}\right) - H_3(p_0, p_+, p_-) + \frac{2}{3} \\
&\quad = H_3\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) - H_3(p_0, p_+, p_-) \\
&\quad = \log_2 3 - H_3(p_0, p_+, p_-) \\
&\quad = \rho_{\max} \qquad (22)
\end{aligned}
$$

We can not expect larger capacity than $\rho_{\max}$ even with distortion constraint larger than $p_0 - 1/3$. In other words, the embedding rate keeps constant when $\Delta > p_0 - 1/3$.

Figure 1 shows the rate-distortion curve of a typical cover, with $p_0 = 0.7$, $p_+ = 0.1$, $p_- = 0.2$.

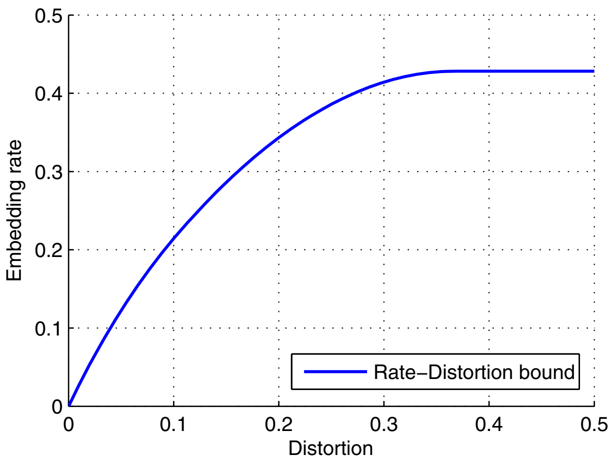


Fig. 1 Rate distortion curve of ternary cover reversible data hiding ($p_0 = 0.7$, $p_+ = 0.1$, $p_- = 0.2$)

3 Proposed recursive code construction

In this section, we will present our code construction to approach the rate-distortion bound (3), which has been motivated by the recursive construction [12]. We will first present an analysis of the methodology; then some implementation details are discussed; finally, a thorough algorithm diagram is presented.

3.1 Proposed code construction for reversible data hiding in ternary cover

Assume that the cover is a memoryless ternary sequence $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, with $x_i \in \{-1, 0, 1\}$. The secret message $\mathbf{m} = (m_1, m_2, \cdots, m_L)$ is a binary random sequence with $m_i \in \{0, 1\}$. We first divide the cover sequence $\mathbf{x}$ into g disjoint blocks, in which the first $g-1$ blocks have the same length K, and the last block has the length L_{last}, and thus $N = K(g-1) + L_{last}$. To finish the embedding, we have to set L_{last} to be larger than K, and we will discuss how to determine L_{last} in Section 3.3. We assume both N and g are large enough. The cover block is denoted by $\mathbf{x}_i$, and the corresponding stego block is denoted by $\mathbf{y}_i$, $i = 1, \ldots, g$.

We embed message into each block sequentially by the embedding function $Emb\,()$, such that $(\mathbf{M}_{i+1}, \mathbf{y}_i) = Emb\,(\mathbf{M}_i, \mathbf{x}_i)$, with $i = 1, \ldots, g$ and $\mathbf{M}_1 = \mathbf{m}$. In other words, the embedding process in the ith block outputs the message embedded in the $(i+1)$th block, $\mathbf{M}_{i+1}$. The length of $\mathbf{M}_{g+1}$ will be zero, meaning that the message $\mathbf{m}$ is completely embedded into $\mathbf{x}$. Figure 2 briefly depicts the data embedding process. The extraction and cover reconstruction are processed in a backwards manner with a extraction function $Ext()$, such that $(\mathbf{M}_i, \mathbf{x}_i)$=$Ext(\mathbf{M}_{i+1}, \mathbf{y}_i)$, with $i = g, \ldots, 1$. The functions, $Emb\,()$ and $Ext()$, will be realized by using the decompression and compression algorithms of an entropy coder, denoted by $Decomp()$ and $Comp()$ respectively. For simplicity, we assume that the entropy coder can reach entropy.

To embed the message with minimal distortion, we select conditional pmf according to the rate-distortion pair (R, Δ) such that $R = L/(K(g-1))$ and $\Delta = \rho^{-1}(R)$ where $\rho^{-1}()$ is the inverse function of (3). Note that (3) is an explicit expression of (1), and it obviously is monotonically increasing with Δ, and thus its inverse function exists. We can estimate the value of $\Delta = \rho^{-1}(R)$ by a numerical solution.

The message $\mathbf{m}$ will be embedded into the first $(g-1)$ blocks, and the last block is used to store the information for reconstructing the $(g-1)$th block and the parameters needed by the recipient. As implied by the proof of rate-distortion function in the previous section, the conditional pmf has different forms in the case $0 \le \Delta \le p_- - p_+$ and the case $p_- - p_+ < \Delta \le p_0 - 1/3$, so we will depict our construction respectively according to these two cases. In both cases, the conditional

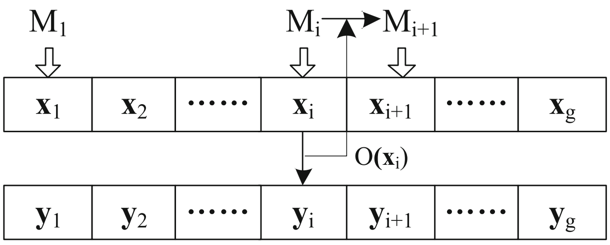


Fig. 2 Sequential blockwise data embedding

pmf (16) and (19) show that the optimal test channel is to only modify 0 to 1 or -1. For simplicity, we denote $d_0 = P^*_{Y|X}(0|0)$, $d_+ = P^*_{Y|X}(1|0)$, and $d_- = P^*_{Y|X}(-1|0)$.

Case 1 $0 \le \Delta \le p_- - p_+$

Data embedding process In the first case, by (16), we get the probability of 0 being modified, (d_0, d_+, d_-), such that

$$d_0 = 1 - \frac{\Delta}{p_0},\ d_+ = \frac{\Delta}{p_0},\ d_- = 0. \tag{23}$$

Denote $\mathbf{M}_1 = \mathbf{m}$, and then we describe how to do $(\mathbf{M}_2, \mathbf{y_1})$=*Emb*$(\mathbf{M}_1, \mathbf{x_1})$. We first extract all zero symbols from the first block $\mathbf{x}_1$, and denoted this all-zero subsequence by $\mathbf{x}'_1$ and its length by $L(\mathbf{x}'_1)$. We will only embed message into $\mathbf{x}'_1$. To do that, we decompress the message sequence $\mathbf{M}_1$ by the decompression algorithm *Decomp*() of the entropy decoder with parameter (d_0, d_+, d_-), such that

$$(\mathbf{y}'_1, b_1) = Decomp(\mathbf{M}_1, (d_0, d_+, d_-), L(\mathbf{x}'_1)). \tag{24}$$

Equation (24) means that the first b_1 bits of $\mathbf{M}_1$, i.e., $(m_1, \cdots, m_{b_1})$, are decompressed into a ternary sequence $\mathbf{y}'_1$ with length $L(\mathbf{x}'_1)$ and the distribution such that $P(y'_{1i} = 0) = d_0$, $P(y'_{1i} = 1) = d_+$, $P(y'_{1i} = -1) = d_-$, $1 \le i \le L(\mathbf{x}'_1)$. In fact, $\mathbf{y}'_1$ is a binary sequence in this case because $d_- = 0$. Second, we substitute the subsequence $\mathbf{x}'_1$ with $\mathbf{y}'_1$, by which we embed b_1 bits of messages into $\mathbf{x}_1$ and get the stego block $\mathbf{y}_1$. At the receiver side, the message bits, $(m_1, \cdots, m_{b_1})$, can be extracted by compressing $\mathbf{y}'_1$ with parameter (d_0, d_+, d_-), such that

$$(m_1, \cdots, m_{b_1}) = Comp(\mathbf{y}'_1, (d_0, d_+, d_-)). \tag{25}$$

Because the average length of $\mathbf{x}'_1$ is Kp_0 and the entropy coder can reach entropy, the average length of the message embedded into $\mathbf{x}_1$, i.e., the expectation of b_1, is equal to

$$Kp_0 H_3(d_0, d_+, d_-) = Kp_0 H_2\left(\frac{\Delta}{p_0}\right). \tag{26}$$

The average amount of distortion introduced by this embedding process is $Kp_0(d_+ + d_-) = K\Delta$, so the average distortion with respect to the length of cover is equal to Δ.

As our data embedding method is reversible, we should be able to restore $\mathbf{x}_1$, which can be fulfilled by embedding some overhead information of $\mathbf{x}_1$ into the subsequent blocks. Now, the question is, what's the overhead of $\mathbf{x}_1$? In [7], Fridrich recommended a strategy that compresses $\mathbf{x}_1$ with an entropy encoder and the compressed sequence is its overhead. Indeed we can implement a conditional compression so that the length of the overhead can be cut down. From the data embedding process we notice that only the 0's in $\mathbf{x}_1$ are used for embedding message, while the 1's and -1's are not changed. Furthermore, in the first case, the 0 will only be modified to 1 as $d_- = 0$. So, if $y_{1i} = 0$, then $x_{1i} = 0$; if $y_{1i} = -1$, then $x_{1i} = -1$; if $y_{1i} = 1$, then $x_{1i} = 1$ or $x_{1i} = 0$. Thus, to losslessly restore $\mathbf{x}_1$, we only need to record the element x_{1i} whose corresponding y_{1i} equal to 1, for $1 \le i \le K$, and obviously such kind of elements include Kp_0d_+ 0's and Kp_+ 1's on average. Therefore, we can compress these elements by using an entropy encoder with parameter

$$\left(\frac{p_0d_+}{p_0d_+ + p_+}, \frac{p_+}{p_0d_+ + p_+}\right), \tag{27}$$

and the compressed sequence is the overhead of $\mathbf{x}_1$. In this way, the average length of the overhead is slashed from $KH_3(p_0, p_+, p_-)$ to

$$K(p_0d_+ + p_+)H_2\left(\frac{p_0d_+}{p_0d_+ + p_+}\right) = K(\Delta + p_+)H_2\left(\frac{\Delta}{\Delta + p_+}\right). \tag{28}$$

We denote the compressed overhead of $\mathbf{x}_1$ by $O(\mathbf{x}_1)$ and concatenate it in the front of the rest bits of $\mathbf{M}_1$ to generate $\mathbf{M}_2$, i.e.,

$$\mathbf{M}_2 = O(\mathbf{x}_1)||(m_{(b_1+1)}, \cdots, m_L) \tag{29}$$

In the same manner as above, some bits in the front of $\mathbf{M}_2$ are embedded into $\mathbf{x}_2$ and the overhead of $\mathbf{x}_2$ is concatenated with the rest bits of $\mathbf{M}_2$ to generate $\mathbf{M}_3$, and then some bits of which is then embedded into $\mathbf{x}_3$...This process is implemented sequentially until the $(g-1)$th block. For the last cover block $\mathbf{x}_g$, we losslessly compress it with parameters (p_0, p_+, p_-) and empty out some space to save $\mathbf{M}_g$ and the parameters needed by the recipient. The parameters should be embedded into the end of $\mathbf{x}_g$ in a backwards manner, and the detail will be described in Section 3.3. Because the last block includes the overhead for reconstructing itself and all bits of $\mathbf{M}_g$, $\mathbf{M}_{g+1}$ will be empty, which implies the finish of the data embedding precess.

Data extraction and cover restoration process The data extraction and cover restoration is processed in a backwards manner, such that $(\mathbf{M}_i, \mathbf{x}_i) = Ext(\mathbf{M}_{i+1}, \mathbf{y}_i)$, with $i = g, \ldots, 1$. First, we read the parameters from the end of $\mathbf{y}$. According to these parameters, we could know how to segment $\mathbf{y}$. We also know the parameters of $Decomp()$ and $Comp()$. Second, from the last stego block, we read $\mathbf{M}_g$ and reconstruct the last cover block $\mathbf{x}_g$ by $Decomp(\mathbf{y}_g, (p_0, p_+, p_-), L_{last})$., which is just the process of $(\mathbf{M}_g, \mathbf{x}_g) = Ext(\mathbf{M}_{g+1}, \mathbf{y}_g)$.

Next, we describe how to do $(\mathbf{M}_{g-1}, \mathbf{x}_{g-1}) = Ext(\mathbf{M}_g, \mathbf{y}_{g-1})$. First, determine the number of $1's$ in $\mathbf{y}_{g-1}$, denoted by l_{g-1}. Second, with the parameter (27), decompress $\mathbf{M}_g$ from its front into a l_{g-1}-length binary sequence, denoted by $\mathbf{z}_{g-1}$, such that

$$(\mathbf{z}_{g-1}, c_{g-1}) = Decomp\left(\mathbf{M}_g, \left(\frac{p_0d_+}{p_0d_+ + p_+}, \frac{p_+}{p_0d_+ + p_+}\right), l_{g-1}\right) \tag{30}$$

Herein, the output, c_{g-1}, means that the first c_{g-1} bits of $\mathbf{M}_g$ are decompressed, and the rest bits of it will be used to construct $\mathbf{M}_{g-1}$. Replacing $1's$ of $\mathbf{y}_{g-1}$ with $\mathbf{z}_{g-1}$, we just get the $(g-1)$th cover block $\mathbf{x}_{g-1}$. Therefore, we can determine the indexes of $0's$ in $\mathbf{x}_{g-1}$. According to these indexes, we extract a subsequence from $\mathbf{y}_{g-1}$, which is compressed with the parameter (23), and then the compressed sequence is concatenated in the front of the rest bits of $\mathbf{M}_g$ to generate $\mathbf{M}_{g-1}$. In the same manner, we can do $(\mathbf{M}_i, \mathbf{x}_i) = Ext(\mathbf{M}_{i+1}, \mathbf{y}_i)$, for $i = g-2, \cdots, 1$, which will restore every cover block and outputs $\mathbf{M}_1 = \mathbf{m}$.

Example 1 Figure 3 shows a simple example of our coding method, in which K is set to be 10. First, extract the 0's of $\mathbf{x}_1$, denoted as $\mathbf{x}_1'$ and decompress of $\mathbf{M}_1$ by using an

index	1	2	3	4	5	6	7	8	9	10
M_1	0	0	1	0	1	0	1	0	1	1
$\mathbf{x}_1$	0	-1	0	1	0	0	0	-1	0	-1
$\mathbf{x}'_1$	0		0		0	0	0		0	
$\mathbf{y}'_1$	0		0		0	1	0		0	
$\mathbf{y}_1$	0	-1	0	1	0	1	0	-1	0	-1
overhead				1		0				

Fig. 3 Example of the proposed coding method in Case 1

entropy decoder with parameter (d_0, d_+, d_-), until the length of the decompressed sequence is the same as that of $\mathbf{x}'_1$; in this example, we assume that the first five bits, "0, 0, 1, 0, 1", are decompressed into $\mathbf{y}'_1 =$ "0, 0, 0, 1, 0, 0". Second, substitute $\mathbf{x}'_1$ with $\mathbf{y}'_1$ and get $\mathbf{y}_1$. Third, record the elements of x_{1i} whose corresponding y_{1i} equal to 1, that is "1, 0", denoted by "overhead". Then, the "1, 0" is to be compressed and concatenated in the front of the rest bits of $\mathbf{M}_1$ to generate $\mathbf{M}_2$.

To extract the embedded message and restore $\mathbf{x}_1$, we first extract $\mathbf{M}_2$ from $\mathbf{y}_2$, and decompress it from its front to get the overhead "1, 0". Then, we substitute the 1's in $\mathbf{y}_1$ with "1, 0" and thus $\mathbf{x}_1$ is restored. After that, we can extract elements y_{1i} from $\mathbf{y}_1$ according to the index i such that $x_{1i} = 0$, and get the subsequence $\mathbf{y}'_1$. Finally we compress $\mathbf{y}'_1$ into the embedded message "0, 0, 1, 0, 1" with parameters (d_0, d_+, d_-).

Case 2 $p_- - p_+ < \Delta \le p_0 - 1/3$

Data embedding process The code construction in this case is similar to that of Case 1, but the parameters of entropy encoder and decoder are different. In this case, by (19), the changing probability of 0, (d_0, d_+, d_-), is given as follows

$$d_0 = 1 - \frac{\Delta}{p_0}, d_+ = \frac{\Delta + p_- - p_+}{2p_0}, d_- = \frac{\Delta - p_- + p_+}{2p_0}. \tag{31}$$

The data embedding process can also be depicted with Fig. 2. We denote $\mathbf{M}_1 = \mathbf{m}$, and do $(\mathbf{M}_2, \mathbf{y_1})=Emb\,(\mathbf{M}_1, \mathbf{x_1})$ as follows. First, we extract all 0's from $\mathbf{x}_1$ (denoted by $\mathbf{x}'_1$), and then decompress $\mathbf{M}_1$ until the length of the decompressed sequence (denoted by $\mathbf{y}'_1$) is the same as that of $\mathbf{x}'_1$. The decompression is done by using the decoding algorithm of the entropy decoder with parameter (d_0, d_+, d_-) determined by (31). Second, we substitute $\mathbf{x}'_1$ with $\mathbf{y}'_1$ and obtain the stego block $\mathbf{y}_1$. The average length of the embedded message is equal to

$$\begin{aligned} Kp_0H_3(d_0, d_+, d_-) &= Kp_0H_3\left(1 - \frac{\Delta}{p_0}, \frac{\Delta + p_- - p_+}{2p_0}, \frac{\Delta - p_- + p_+}{2p_0}\right) \\ &= Kp_0H_2\left(\frac{p_0 - \Delta}{p_0}\right) + K\Delta H_2\left(\frac{\Delta + p_- - p_+}{2\Delta}\right). \end{aligned} \tag{32}$$

The average amount of distortion introduced by this embedding process is $Kp_0(d_+ + d_-) = K\Delta$, so the average distortion with respect to the length of cover is equal to Δ.

To restore $\mathbf{x}_1$, we can embed some overhead information of $\mathbf{x}_1$ into $\mathbf{x}_2$. As we only embed message in the 0's of $\mathbf{x}_1$ and leave the 1's and -1's unchanged. If $y_{1i} = 0$, then $x_{1i} = 0$; if $y_{1i} = 1$, then $x_{1i} = 1$ or $x_{1i} = 0$; if $y_{1i} = -1$, then $x_{1i} = -1$ or $x_{1i} = 0$. Thus, to losslessly restore $\mathbf{x}_1$, we only need to record the elements of x_{1i} whose corresponding y_{1i} equal to 1, which is denoted by "overhead1", and the elements of x_{1i} whose corresponding y_{1i} equal to -1, which is denoted by "overhead2". The "overhead1" consists of Kp_0d_+ 0's and Kp_+ 1's on average, which can be compressed by using an entropy encoder with parameter

$$\left(\frac{p_0d_+}{p_0d_+ + p_+}, \frac{p_+}{p_0d_+ + p_+}\right). \tag{33}$$

The "overhead2" consists of Kp_0d_- 0's and Kp_- -1's on average, which can be compressed by using an entropy encoder with parameter

$$\left(\frac{p_0d_-}{p_0d_- + p_-}, \frac{p_-}{p_0d_- + p_-}\right). \tag{34}$$

Therefore, the average length of the compressed overhead is equal to

$$\begin{aligned} &K(p_0d_+ + p_+)H_2\left(\frac{p_0d_+}{p_0d_+ + p_+}\right) + K(p_0d_- + p_-)H_2\left(\frac{p_0d_-}{p_0d_- + p_-}\right) \\ &= K\frac{\Delta + p_- + p_+}{2}\left(H_2\left(\frac{2p_+}{\Delta + p_- + p_+}\right) + H_2\left(\frac{2p_-}{\Delta + p_- + p_+}\right)\right). \end{aligned} \tag{35}$$

After that, the compressed overheads are concatenated in the front of the rest bits of $\mathbf{M}_1$ and generate $\mathbf{M}_2$. In the same manner as above, we do $(\mathbf{M}_{i+1}, \mathbf{y}_i) = Emb\,(\mathbf{M}_i, \mathbf{x}_i)$ for $i = 2, \cdots, g-1$. For the last block, we losslessly compress it with parameter (p_0, p_+, p_-) and save some space to store $\mathbf{M}_g$ and some parameters, which is similar to the process in Case 1.

Data extraction and cover restoration process The data extraction and cover restoration is processed in a backwards manner, such that $(\mathbf{M}_i, \mathbf{x}_i) = Ext(\mathbf{M}_{i+1}, \mathbf{y}_i)$, with $i = g, \ldots, 1$. First, we read parameters from the end of $\mathbf{y}$, and segment $\mathbf{y}$ according to the parameters. From the last stego block, we read $\mathbf{M}_g$ and reconstruct the last cover block by $Decomp(\mathbf{y}_g, (p_0, p_+, p_-), L_{last})$.

Next, we describe how to do $(\mathbf{M}_{g-1}, \mathbf{x}_{g-1}) = Ext(\mathbf{M}_g, \mathbf{y}_{g-1})$. First, determine the number of $1's$ and $-1's$ in $\mathbf{y}_{g-1}$, denoted by l^+_{g-1} and l^-_{g-1} respectively. Second, with parameters (33), we decompress $\mathbf{M}_g$ from its front into a l^+_{g-1}-length binary sequence, denoted by $\mathbf{z}^+_{g-1}$; and with parameters (34), we decompress the rest bits of $\mathbf{M}_g$ from the front into a l^-_{g-1}-length binary sequence, denoted by $\mathbf{z}^-_{g-1}$. Note that we represent non-zero symbols in $\mathbf{z}^+_{g-1}$ by 1, and in $\mathbf{z}^-_{g-1}$ by -1. Third, we replace all $1's$ of $\mathbf{y}_{g-1}$ with $\mathbf{z}^+_{g-1}$ and all $-1's$ of $\mathbf{y}_{g-1}$ with $\mathbf{z}^-_{g-1}$, and get the cover block $\mathbf{x}_{g-1}$. According to the indexes of $0's$ in $\mathbf{x}_{g-1}$, we extract a subsequence from $\mathbf{y}_{g-1}$, which is compressed with parameter (31), and then the compressed sequence is concatenated in the front

index	1	2	3	4	5	6	7	8	9	10
M_1	0	0	1	0	1	0	1	0	1	1
$\mathbf{x}_1$	0	0	0	1	0	0	0	-1	0	-1
$\mathbf{x}'_1$	0	0	0		0	0	0		0	
$\mathbf{y}'_1$	0	1	0		-1	1	0		0	
$\mathbf{y}_1$	0	1	0	1	-1	1	0	-1	0	-1
overhead1		0		1		0				
overhead2					0			-1		-1

Fig. 4 Example of our coding method in Case 2

of the rest bits of $\mathbf{M}_g$ to generate $\mathbf{M}_{g-1}$. In the same manner, we can do $(\mathbf{M}_i, \mathbf{x}_i) = Ext(\mathbf{M}_{i+1}, \mathbf{y}_i)$, for $i = g-2, \cdots, 1$, which will restore every cover block and outputs $\mathbf{M}_1 = \mathbf{m}$.

Example 2 Figure 4 shows an example of our coding method in Case 2, in which K is again set to 10. First, extract the 0's of $\mathbf{x}_1$, denoted as $\mathbf{x}'_1$ and decode a subsequence of $\mathbf{M}_1$ with an entropy decoder of parameter (d_0, d_+, d_-), until the length of the decoded sequence is the same as that of $\mathbf{x}'_1$; in this example, we assume that the first five bits "0, 0, 1, 0, 1" is decompressed into $\mathbf{y}'_1$ = "0, 1, 0, -1, 1, 0, 0". Second, substitute $\mathbf{x}'_1$ with $\mathbf{y}'_1$ and get $\mathbf{y}_1$. Third, record the elements of x_{1i} whose corresponding y_{1i} equal to 1, that is "overhead1" = "0, 1, 0", and the elements of x_{1i} whose corresponding y_{1i} equal to -1, that is "overhead2" = "0, -1, -1". Compress these two overheads respectively. Finally, the compressed overheads are concatenated in the front of the rest bits of $\mathbf{M}_1$ to generate $\mathbf{M}_2$.

To extract the embedded message and restore $\mathbf{x}_1$, we first extract $\mathbf{M}_2$ from the latter block, and decompress it from the front to get "overhead1" and "overhead2". Then, we substitute the 1's in $\mathbf{y}_1$ with "overhead1" = "0, 1, 0", and substitute the -1's in $\mathbf{y}_1$ with "overhead2" = "0, -1, -1", and thus $\mathbf{x}_1$ is restored. After that, $\mathbf{y}'_1$ can be easily extracted from $\mathbf{y}_1$ according to the index i such that $x_{1i} = 0$. Finally we compress $\mathbf{y}'_1$ to obtain the embedded bits of $\mathbf{M}_1$.

Before we end this part, we should declare that in our presentation, we only consider the case $0 \le \Delta \le p_- - p_+$ and the case $p_- - p_+ < \Delta \le p_0 - 1/3$, but our coding method also works for the case $\Delta > p_0 - 1/3$. In this case, it is processed in the same way as $\Delta = p_0 - 1/3$ in Case 2, as the rate-distortion curve shows that the embedding rate will never become larger with increasing Δ in this model.

3.2 Proof of optimality

In this subsection, we will prove that our construction is optimal in the sense that as long as the entropy coder reaches entropy, the proposed codes asymptotically approach the rate-distortion bound (3) when the number of blocks N/K tends to infinity. In fact, when N/K tends to infinity, the influence of the last block is negligible, and thus the average embedding rate and distortion of the code construction can be calculated within one K-length block.

Case 1 $0 \leq \Delta \leq p_- - p_+$

When the average distortion $D = \Delta$, on one hand, the rate-distortion function in this case is given by

$$\rho(\Delta) = H_3(p_0 - \Delta, p_+ + \Delta, p_-) - H_3(p_0, p_+, p_-). \tag{36}$$

On the other hand, for a K-length cover block in the code construction, the average length of embedded messages is given by (26) and the average length of information needed by reconstructing this block is given by (28), so the average embedding rate in one block is given by (26)–(28)/K, that is,

$$R_1(\Delta) = p_0 H_2\left(\frac{\Delta}{p_0}\right) - (\Delta + p_+) H_2\left(\frac{\Delta}{\Delta + p_+}\right). \tag{37}$$

By the expansion of entropy function, it is easy to verify that $R_1(\Delta) = \rho(\Delta)$.

Case 2 $p_- - p_+ < \Delta \leq p_0 - 1/3$

In this case, the rate-distortion function is given by:

$$\rho(\Delta) = H_2(p_0 - \Delta) - H_3(p_0, p_+, p_-) + 1 - p_0 + \Delta. \tag{38}$$

In our code construction, in a K-length cover block, the average length of embedded messages is given by (32) and the average length of information needed by reconstructing this block is given by (35), so the average embedding rate in one block is given by (32)–(35)/K, that is,

$$\begin{aligned} R_2(\Delta) = {} & p_0 H_2\left(\frac{p_0 - \Delta}{p_0}\right) + \Delta H_2\left(\frac{\Delta + p_- - p_+}{2\Delta}\right) \\ & - \frac{\Delta + p_- + p_+}{2}\left(H_2\left(\frac{2p_+}{\Delta + p_- + p_+}\right) + H_2\left(\frac{2p_-}{\Delta + p_- + p_+}\right)\right). \end{aligned} \tag{39}$$

By expansion of entropy functions, it is easy to verify that $R_2(\Delta) = \rho(\Delta)$ also holds in this case.

As a summarization of these two cases, we prove that, for any given distortion constraint Δ such that $0 \leq \Delta \leq p_0 - 1/3$, the proposed codes can reach the rate-distortion bound.

Therefore, in Case 1, the maximum embedding rate, $R_{1\max}$, achievable by the proposed codes can be expressed by the rate-distortion bound at $\Delta = p_- - p_+$, that is

$$R_{1\max} = H_3(p_0 - p_- + p_+, p_-, p_-) - H_3(p_0, p_+, p_-). \tag{40}$$

In Case 2, the maximum embedding rate, $R_{2\max}$, achievable by the proposed codes is equal to $\rho_{\max}$, that is,

$$R_{2\max} = \rho_{\max} = \log_2 3 - H_3(p_0, p_+, p_-). \tag{41}$$

3.3 Transferring parameters in the last block

It has been declared above that our data extraction method is implemented in a backwards manner. For a recipient, in order to commence the data extraction process, he needs to know clearly about where and how to start the decoding. This requires negotiation between the transmitter and the recipient before the communication begins. One direct idea is to embed some negotiation information into the cover itself, and the recipient can easily extract the information without ambiguity, then with the negotiation information the recipient can start data extraction process. But there are two questions: what's the negotiation information, and how to embed it into the cover in a reversible manner?

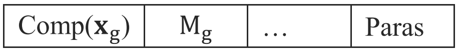
We find that there are a few parameters that the recipient have to know for successful decoding, these parameters include p_0, p_+, L (length of the message), K (length of each block except for the last one), L_{last} (length of the last block) and $L_{\mathbf{M}_g}$ (length of the message embedded into the last block). The recipient can figure out other information from these parameters.

To answer the second question, let's reconsider the composition of the last stego block, $\mathbf{y}_g$. As mentioned above, we will losslessly compress $\mathbf{x}_g$ with (p_0, p_+, p_-) and save some space to store $\mathbf{M}_g$ and the parameters. As depicted in Fig. 5, Comp($\mathbf{x}_g$) is the losslessly compressed edition of $\mathbf{x}_g$, and Paras stands for the parameters. The format of Paras can be negotiated by the transmitter and the recipient beforehand, but notice that it should be embedded in a backward manner, from the last element $\mathbf{y}_g$ to preceding elements.

After receiving the stego sequence $\mathbf{y}$, the recipient can determine its length N, and read parameters from the end of the stego sequence. Then, as both K and L_{last} are known, the decoder understands how to segment $\mathbf{y}$. On the other hand, the decoder can calculate other important parameters with the extracted parameters: $p_- = 1 - p_0 - p_+$; the embedding rate $R = L/(N - L_{last})$. Furthermore, calculate $R_{1\max}$ and $R_{2\max}$ by (40) and (41) respectively, and estimate $\Delta = \rho^{-1}(R)$ by the inverse function of (3). Then, the recipient calculates parameters for compression/decompression with (27) if $R \le R_{1\max}$, or with (33) and (34) if $R_{1\max} < R \le R_{2\max}$. After that, the recipient can implement the backwards decoding process, as discussed in Section 3.1.

There is another important issue that relates to the length of the last block, L_{last}. As mentioned above, after being compressed, the saved space in $\mathbf{x}_g$ is about $(\log_2 3 - H_3(p_0, p_+, p_-))L_{last}$ bits, which should be long enough to contain $\mathbf{M}_g$ and the parameters. Note that the embedding rate $R = L/(N - L_{last})$ is respect to the length of the first $g-1$ blocks, so the proof of optimality implies that the message can be completely embedded into the first $g-1$ blocks as g is large enough. In other words, the first $g-1$ blocks can carry the message and the overhead for restore the first $g-2$ cover blocks, and thus we can expect that the length of $\mathbf{M}_g$ is close to the average length of compressed overhead of a K-length block. In Case 1, the length of the overhead before being compressed is equal to the number of $1's$ in the stego block, which is not larger than $K/3$ as implied by the conditional pmf (16); in Case 2, the length of the overhead before being compressed is equal to the sum of the number of $1's$ and $-1's$ in the stego block, which is not larger than $(2K)/3$ as implied by the conditional pmf (19). Therefore, we reserve L_{oh} bits for $\mathbf{M}_g$ and set $L_{oh} = K/3$ for Case 1 and $L_{oh} = (2K)/3$ for Case 2. We also denote L_{Paras} as the length of Paras

Fig. 5 Composition of $\mathbf{y}_{last}$



(in practice, the parameters can be represented with no more than 100 bits, thus $L_{Paras} \leq 100$). Then, the following inequality should be satisfied:

$$(\log_2 3 - H_3(p_0, p_+, p_-))L_{last} \geq L_{oh} + L_{Paras}, \tag{42}$$

and thus,

$$L_{last} \geq \frac{L_{oh} + L_{Paras}}{\log_2 3 - H_3(p_0, p_+, p_-)}. \tag{43}$$

We first set L_{last} equal to the the lower bound (43) and then determine the number of K-length blocks by

$$g - 1 = \left\lfloor \frac{N - L_{last}}{K} \right\rfloor. \tag{44}$$

The ultimate length of the last block is set as

$$L_{last} = N - (g-1)K. \tag{45}$$

In practice, to ensure enough space in the last block, we can use a somewhat larger L_{last}.

3.4 Diagram of proposed code construction

As a conclusion of above discussion, we'd present our code construction within an algorithm diagram (Algorithm 1):

We also ran a simulation of proposed code construction. We selected adaptive arithmetic coder as the entropy coder, and set the cover length $N = 100000$, and the block length $K = 300$. For each embedding rate, we run Algorithm 1 one hundred times by randomly generating covers and messages, and estimated the distortion as an average over the 100 samples. The test was performed at 2.8 GHz with 2GB RAM. The algorithm was implemented in Matlab R2009a. For each embedding rate, the

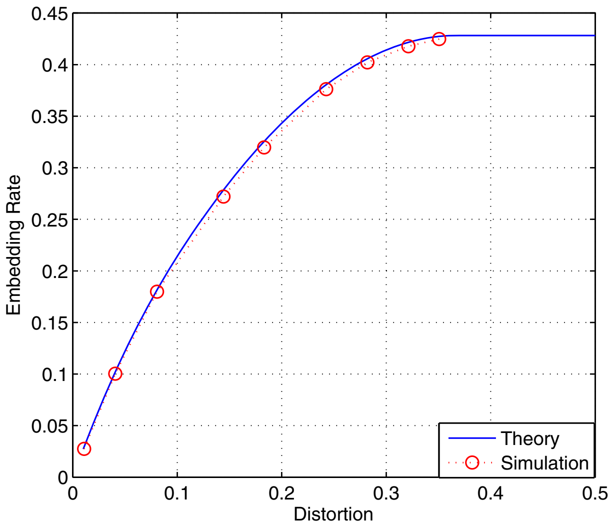


Fig. 6 Simulation of the proposed code construction ($p_0 = 0.7$, $p_+ = 0.1$, $p_- = 0.2$)

Algorithm 1 Ternary recursive code construction

```
Data Embedding

– 0. Input the cover sequence $\mathbf{x}$ with length $N$, and the message sequence $\mathbf{m}$ with length $L$.
– 1. Estimate $p_0$, $p_+$ and $p_-$ from $\mathbf{x}$. Set the initial value of $L_{last}$ by the lower bound of (43).
– 2. Calculate number of blocks $g$ by (44) and update the value of $L_{last}$ by (45) and then calculate the embedding rate $R = L/(N - L_{last})$.
– 3. Calculate $R_{1\max}$, $R_{2\max}$ by (40) and (41) respectively. If $R \le R_{1\max}$, do Step 4; if $R_{1\max} < R \le R_{2\max}$, do Step 5; else, if $R > R_{2\max}$, stop the embedding process and return "The message is too long to be embedded successfully."
– 4. Calculate $\Delta$ via inverse function of (36). Calculate parameters for entropy encoding and decoding by (23) and (27). Implement the data embedding procedure as described in Section 3.1-Case 1.
– 5. Calculate $\Delta$ via inverse function of (38). Calculate parameters for entropy encoding and decoding by (31), (33) and (34). Implement the data embedding procedure as described in Section 3.1-Case 2.
– 6. If $\mathbf{M}_g$ and the parameters, $p_0$, $p_+$, $L$, $K$, $L_{last}$, $L_{\mathbf{M}_g}$, can be completely embedded into the last block, output the stego sequence $\mathbf{y}$; otherwise, set $L_{last} = L_{last} + K$, and redo Step 2–Step 6.

Data Extraction and Cover Restoration

– 0. Input the stego sequence $\mathbf{y}$ with length $N$.
– 1. Extract the parameters $p_0$, $p_+$, $L$, $K$, $L_{last}$, $L_{\mathbf{M}_g}$ from the end of $\mathbf{y}$; calculate other parameters such that $p_- = 1 - p_0 - p_+$, $R = L/(N - L_{last})$; if $R \le R_{1\max}$, do 2–3; if $R_{1\max} < R \le R_{2\max}$, do 4–5.
– 2. Calculate $\Delta$ via inverse function of (36). Calculate parameters for entropy encoding and decoding by (23) and (27). Implement the data extraction and cover restoration procedure as described in Section 3.1-Case 1. Output the message $\mathbf{m}$ and the cover $\mathbf{x}$.
– 3. Calculate $\Delta$ via inverse function of (38). Calculate parameters for entropy encoding and decoding by (31), (33) and (34). Implement the data extraction and cover restoration procedure as described in Section 3.1-Case 2. Output the message $\mathbf{m}$ and the cover $\mathbf{x}$.
```

average running time of one turn data embedding is 86.4 ms in this setting. As shown in Fig. 6, the simulation result is quite close to the theoretic upper bound. This also proves the correctness and feasibility of our coding method.

4 A case study: reversible data hiding in JPEG images

Quantized DCT Coefficients are the typical scenario fitting the cover model described in Section 2, and thus the proposed construction can be used to design RDH schemes in DCT based multimedia such as JPEG images and H.264 videos. In this section, we take JPEG as an example to show how the proposed code construction can improve the performance of reversible data hiding in DCT domain.

JPEG is the most popular image format currently, but so far there are not that many publications on RDH for JPEG images, compared to works on RDH for grayscale images. Fridrich [7] first made such an attempt. She first selected some mid-frequency quantized DCT coefficients (QDCTCs) which are equal to 0 and 1, the 0's and 1's thus form a compressible binary sequence. The sequence is then losslessly compressed and some spare space is emptied out. The spare space is finally used to embed secret message. In one of our previous papers [27], we increased the PSNR of Fridrich et al.'s method by binary codes, but both in [7] and in [27] the achievable maximum payloads are quite small. Chang [4] proposed a RDH algorithm that searches special "zero" pattern in each block. Then message can be embedded by directly substituting the zeros with message bits. Lin et al. [15] combined Chang's method with Difference Expansion Technique and improved the payload considerably. Li [13] and Xuan [24] adopted the idea of histogram shift in their methods. Li's [13] method combines simple histogram shift with quantization table modification and has shown a satisfactory performance. Xuan [24] took several factors into consideration and tried to find best PSNR adaptively for any given payload.

4.1 RDH algorithm for JPEG images with proposed code construction

It's well known that the quantized DCT coefficients (QDCTCs) in a 8×8 block approximately follow a Laplacian distribution. Fig.7 is the histogram of mid-frequency QDCTCs of lenna.jpg image, with Q-factor equal to 80. By mid-frequency coefficients, we mean the QDCTCs in each block whose zig-zag order is between 11 and 21, which are in bold font in Table 1. The quantization step is very close in these 11 entries. The reason to choose mid-frequency coefficients is that

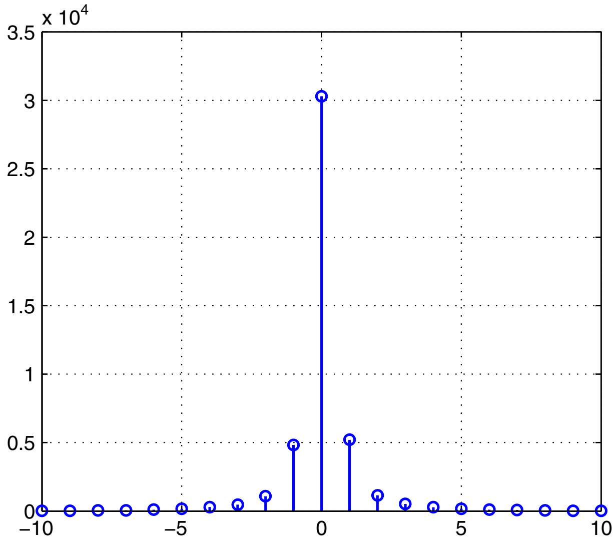


Fig. 7 Histogram of mid-frequency QDCTCs (lenna.jpg, $Q = 80$)

Table 1 Selected coefficient entries in each block

1	2	6	7	**15**	**16**	28	29
3	5	8	**14**	**17**	27	30	43
4	9	**13**	**18**	26	31	42	44
10	**12**	**19**	25	32	41	45	54
11	**20**	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

modification to low frequency coefficients is harmful to image's subjective quality, while modification to high frequency coefficients affects the objective quality and filesize considerably. As shown in Fig. 7, more than 90 % QDCTCs fall into the set $\{0, 1, -1\}$. Specially, the percentage of 0 is about 70 % of all the coefficients. Therefore, typical images, e.g., the lenna.jpg, with Q-factor less than or equal to 80, provide an ideal example of ternary cover if we focus on the mid-frequency coefficients that belong to the set $\{0, 1, -1\}$ and skip all other coefficients. Then, we can embed message into the selected coefficients in a reversible way with our coding method over ternary cover, which has been detailed in Section 3. In this way, we can embed message into JPEG image reversibly.

The following is the diagram of our reversible data hiding algorithm for JPEG images (Algorithm 2).

Algorithm 2 Reversible data hiding in JPEG images

Data Embedding

- **1**. For the cover JPEG image, first extract all mid-frequency coefficients in each block in order and form a sequence; extract the subsequence of coefficients whose value belong to $\{0, 1, -1\}$, denoted as **x_ter**;
- **2**. Implement RDH over **x_ter** via our code construction, and the corresponding stego is denoted as **y_ter**;
- **3**. Substitute **x_ter** with **y_ter**; then put the modified coefficients back to their original locations; then form a new JPEG image.

Data Extraction and Cover Restoration

- **1**. Extract **y_ter** in the same way as **x_ter** is extracted in the Data Embedding Process;
- **2**. Implement data extraction and cover restoration over **y_ter**, which has been presented in Section 3; thus **x_ter** is restored; meanwhile, the embedded message is extracted;
- **3**. Substitute **y_ter** with **x_ter**; then put the modified coefficients back to their original locations to form a new JPEG image. The new JPEG image is indeed the original cover JPEG image.

4.2 Experimental results

The data embedding process will lead to the deterioration of image's quality and increase of JPEG filesize, which are not appreciated. Indeed, it is a latent

Fig. 8 Comparison of original and stego Lenna images

requirement that the RDH algorithm should not cause severe image quality degradation or serious filesize increment. In our experiments, we choose some commonly adopted measurements to measure the performance of proposed method these measurements include payload (the number of message bits embedded), PSNR (to measure stego image's objective quality against cover image) and filesize increment. Generally, if a RDH method can achieve high PSNR value and small filesize increment for a large payload range, we say the method is good.

We implement our RDH algorithm on a very popular image, Lenna, with Q-factor equal to 80. Figure 8 shows the original and stego Lenna images. (a) is the original image, and (b) to (d) are the stego images with payload equal to 2000, 5000, 20000, respectively. From the figure, we can see that even with as many as 20000 message bits embedded, the image's quality is still quite good. Table 2 shows the stego images' PSNRs and filesize increments under various payloads. The filesize increment is about 0.25 byte/bit, which means on average, per bit payload embedding will give rise to 2 bits increment in the JPEG file length. With as many as 2000 bits to embed, the filesize just increase 1.3 %, which is acceptable in most applications.

We also compare our method with four previous arts: [4, 13, 15, 24]. Out of fairness, we do not modify the quantization table in any of these experiments. We randomly selected 200 grayscale images from website [1], and compress into JPEG images with normal JPEG coder. Then we run before-mentioned RDH algorithms on those JPEG images, and average the results. Regarding the fact that different compression scales will lead to difference in distribution of QDCTCs and thus difference in performance, we also implement the experiments on JPEG images of different Q-factors.

Figures 9 and 10 show the performance comparison of these methods. Figure 9 shows the average PSNR against payload curves for images with Q-factor equal to 50 and 80, while Fig. 10 demonstrates the filesize increment comparison.

Table 2 PSNR and filesize increment under different payloads

Test image	Lenna						
Payload (bits)	0	2,000	3,000	5,000	10,000	15,000	20,000
PSNR (dB)	–	52.69	50.72	48.53	45.19	42.97	40.82
Filesize (Bytes)	36,131	36,617	36,888	37,360	38,586	39,871	41,617

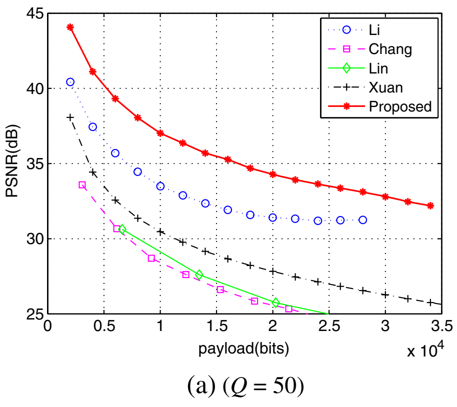


Fig. 9 PSNR comparison of different methods

From the above figures, we can draw the conclusion that our RDH algorithm shows great advantage over previous arts as regards to image quality, which is measured by PSNR. Throughout payload axis, when embedding the same number of message bits, our method has a PSNR improvement of at least 3 dB compared to other works. The excellent performance comes from several reasons. First we only embed data into mid-frequency coefficients, other than Chang's, Lin's work; then, we only modify the zero coefficients by 1 in DCT domain, so the distortion is restricted. The power of proposed code construction also contributes to the good performance.

On the other hand, the performance of filesize increment is not that impressive, from Fig. 10, the filesize increment of our algorithm is smaller than Li's, Chang's and Lin's method, but larger than Xuan's work. Our method is based on modifying 0 to 1 and -1 with certain probability, this will break up consecutive zeros and thus disturbs the run-level coding. This explains most filesize increment. In Chang's and Lin's work, it's quite possible to modify high-frequency zeros and thus filesize increment

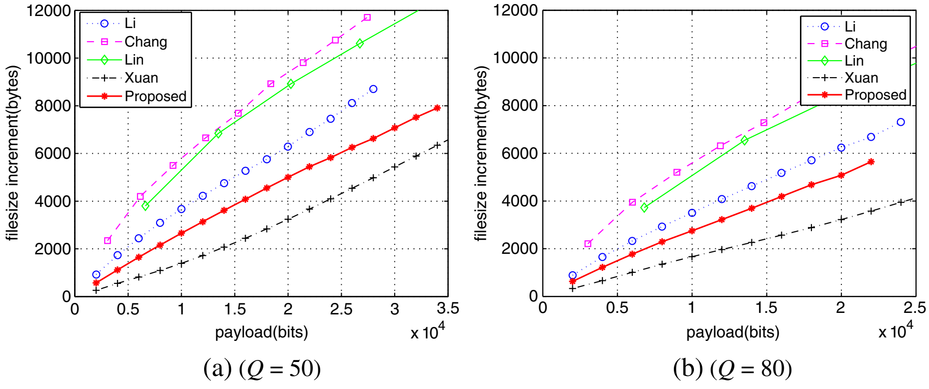


Fig. 10 Filesize increment comparison of different methods

Table 3 Computation time comparison of different methods

Payload (bits)	2,000	5,000	10,000	15,000	20,000
Li	0.010	0.010	0.012	0.013	0.013
Xuan	0.119	0.142	0.162	0.178	0.185
Chang	0.050	0.122	0.246	0.353	0.467
Lin	0.030	0.064	0.134	0.181	0.234
Proposed	0.062	0.068	0.075	0.077	0.083

Time unit: second

problem is very serious in their methods. Li's method modifies mid-frequency zeros like our work, but simple histogram shift method cannot provide efficient restriction to increase of filesize. Xuan's method modifies the largest DCT coefficients, which are often low-frequency coefficients, modifying them has little influence on filesize. This is the reason why Xuan's method has a very small filesize increment.

Our method doesnot work very well on filesize benchmark. How to improve the performance? Our ternary code construction is based on the assumption that cover symbols are independent, thus we have not considered avoiding breaking up consecutive zeros, so it seems hard to restrict filesize increment from the code construction's viewpoint. On the other hand, our method modifies coefficients equal to 0, while Xuan's method modifies the largest coefficients. If we combine our method with Xuan's method (i.e., a part payload is embedded with our method, and the other part of payload is embedded with Xuans method), we can make a better trade-off between image quality and filesize increment.

Besides comparing the performance of different methods, we also compare the computation time. Table 3 shows the average computation time of those methods when the payload varies from 2000 to 20000 bits, in which the time unit is second. From the table, we can find Li's method is quite fast. The speed of proposed method is also satisfactory. Meanwhile, in Chang's and Lin's work, computation time is almost linear to payload, but in both Li's and proposed method the relation between computation time and payload is not that strong. Our method is based on ternary recursive code and entropy coder, so computation time keeps almost the same.

As a whole, the figures and Table 3 imply that the proposed method makes a better trade-off than previous works, as regards to image quality, filesize increment and computation time.

5 Conclusion

Though many reversible data hiding techniques have been reported since it first appeared, there are only a few theoretical works in this area. In this paper, we formulate the theoretical model of reversible data hiding over a special ternary cover, in which the probability of 0 is prominent. We also propose a code construction to realize RDH over the ternary cover. The performance of the code construction is based on the performance of the coding and decoding algorithms of an entropy coder. We prove that our code construction can asymptotically approach the rate-distortion bound with increasing length of cover sequence as long as the adopted entropy coder reaches entropy. Finally, a case study is presented that applies the code construction to RDH for JPEG images. The experimental results demonstrate the outstanding performance of our RDH method. Our code construction is also applicable to other DCT-based multimedia signals as digital audio and digital video.

However, the model considered in this paper is not that universal. One of our further work is to extend the model and the optimal code construction to the multi-ary covers.

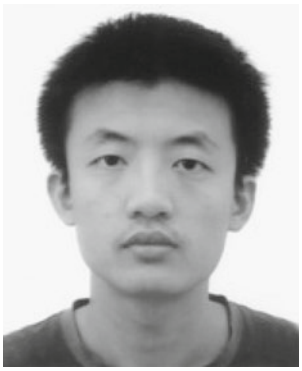
References

1. Uncompressed Colour Image Database [Online]. Available: http://homepages.lboro.ac.uk/~cogs/datasets/ucid/ucid.html. Link is still valid on May 4th, 2013
2. Caldelli R, Filippini F, Becarelli R (2010) Reversible watermarking techniques: an overview and a classification. EURASIP J Inf Secur 2010:1–19
3. Celik M, Sharma G, Tekalp A, Saber E (2005) Lossless generalized-LSB data embedding. IEEE Trans Image Process 14(2):253–266
4. Chang CC, Lin CC, Tseng CS, Tai WL (2007) Reversible hiding in DCT-based compressed images. Inf Sci 177(13):2768–2786
5. Chung K, Huang Y, Chang P et al (2010) Reversible data hiding-based approach for intra-frame error concealment in H.264/AVC. IEEE Trans Circuits Syst Video Technol 20(11):1643–1647
6. Feng J, Lin I, Tsai C et al (2006) Reversible watermarking: current status and key issues. Int J Netw Secur 2(3):161–171
7. Fridrich J, Goljan M, Du R (2001) Invertible authentication watermark for JPEG images. In: Proc. of IEEE conference on information technology: computing and coding, pp 223–227
8. Fridrich J, Goljan M (2002) Lossless data embedding for all image formats. In: SPIE proceedings of photonics west, electronic imaging, security and watermarking of multimedia contents, vol 4675. San Jose, pp 572–583
9. Honsinger C, Jones P, Rabbani M, Stoffel J (2001) Lossless recovery of an original image containing embedded data. US Patent 6278791
10. Hu Y, Lee H, Li J (2009) DE-based reversible data hiding with improved overflow location map. IEEE Trans Circuits Syst Video Technol 19(2):250–260
11. Hwang K, Li D (2010) Trusted cloud computing with secure resources and data coloring. IEEE Internet Comput 14(5):14–22
12. Kalker T, Willems F (2003) Capacity bounds and constructions for reversible data hiding. In: Proc. Of EI SPIE, security and watermarking of multimedia contents V, vol 5020. Santa Clara, pp 604–611
13. Li Q, Wu Y, Bao F (2010) A reversible data hiding scheme for JPEG images. In: PCM 2010, LNCS 6297, pp 653–664
14. Li X, Yang B, Zeng T (2011) Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. IEEE Trans Image Process 20(12):3524–3533
15. Lin CC, Shiu PF (2010) DCT-based reversible data hiding scheme. J Softw 5(2):214–224
16. Luo L, Chen Z, Chen M, Zeng X, Xiong Z (2010) Reversible image watermarking using interpolation technique. IEEE Trans Inf Forensics Secur 5(1):187–193
17. Ni Z, Shi Y, Ansari N, Wei S (2006) Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3):354–362
18. Peng F, Li X, Yang B (2012) Adaptive reversible data hiding scheme based on integer transform. Signal Process 92(1):54–62
19. Petitcolas F, Anderson R, Kuhn M (1999) Information hiding: a survey. Proc IEEE 87:1062–1078
20. Thodi D, Rodriguez J (2007) Expansion embedding techniques for reversible watermarking. IEEE Trans Image Process 16(3):721–730
21. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Trans Circuits Syst Video Technol 13(8):890–896
22. Tsai P, Hu YC, Yeh HL (2009) Reversible image hiding scheme using predictive coding and histogram shifting. Signal Process 89:1129–1143
23. Wong K, Tanaka K (2010) DCT based scalable scrambling method with reversible data hiding functionality. In: Proceedings of 2010 4th International symposium on communications, control and signal processing (ISCCSP), pp 1–4
24. Xuan G, Shi Q, Ni Z et al (2007) Reversible data hiding for JPEG images based on histogram pairs. In: ICIAR 2007, LNCS 4633, pp 715–727
25. Zhang X, Wang S, Qian Z, Feng G (2010) Reversible fragile watermarking for locating tampered blocks in JPEG images. Signal Process 90(12):3026–3036

26. Zhang W, Chen B, Yu N (2011) Capacity-approaching codes for reversible data hiding. In: Proc. of 13th information hiding conference, LNCS 6958. Prague, pp 255–269
27. Zhang W, Chen B, Yu N (2012) Improving various reversible data hiding schemes via optimal codes for binary covers. IEEE Trans Image Process 21(6):2991–3003

Biao Chen received the Bachelor's degree from Hefei University of Technology, Hefei, China, in 2010. He is currently seeking for his master's degree in Department of Electrical Engineering and Information Science, University of Science and Technology of China. His research interests include data hiding and image analysis.

Weiming Zhang received his M.S. degree and PH.D. degree in 2002 and 2005 respectively from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China. Currently, he is an associate professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include information hiding and cryptography.



Kede Ma received the B.S. degree in electronic engineering and information science from University of Science and Technology of China. He is currently pursuing the Master degree with the University of Waterloo. He's research interests include information hiding and image quality assessment.

Nenghai Yu received his B.S. degree in 1987 from Nanjing University of Posts and Telecommunications, M.E. degree in 1992 from Tsinghua University and Ph.D. degree in 2004 from University of Science and Technology of China, where he is currently a professor. His research interests include multimedia security, multimedia information retrieval, video processing and information hiding.