Ijesm
Consulting, help, relaxation

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & MANAGEMENT

Enhancing Reversible Data hiding Schemes for Marked Covers

KANTHI KIRAN KARASLA (M.Tech student),QCET,Nellore,kanthikiran007@gmail.com
SYED AKHTAR , M.Tech Assistant Professor ,QCET,Nellore, akhtar.sab@gmail.com
P.Babu, Associate professor, QCET,Nellore,babu123mca@gmail.com

***Abstract*—In reversible data hiding (RDH), the original cover can be losslessly restored after the embedded information is extracted. Kalker and Willems established a rate–distortion model for RDH, in which they proved out the rate–distortion bound and proposed a recursive code construction. In our previous paper, we improved the recursive construction to approach the rate–distortion bound. In this paper, we generalize the method in our previous paper using a decompression algorithm as the coding scheme for embedding data and prove that the generalized codes can reach the rate–distortion bound as long as the compression algorithm reaches entropy. By the proposed binary codes, we improve three RDH schemes that use binary feature sequence as covers, i.e., an RS scheme for spatial images, one scheme for JPEG images, and a pattern substitution scheme for binary images. The experimental results show that the novel codes can significantly reduce the embedding distortion. Furthermore, by modifying the histogram shift (HS) manner, we also apply this coding method to one scheme that uses HS, showing that the proposed codes can be also exploited to improve integer-operation-based schemes.**

***Index Terms*—Difference expansion (DE), histogram shift (HS), recursive code construction, reversible data hiding (RDH), watermarking.**

I. Introduction

DATA HIDING is a technique for embedding information into covers such as image, audio, and video files, which can be used for media notation, copyright protection, integrity authentication, 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, ensure 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 method, which is referred to as reversible data hiding (RDH) or lossless data hiding, by which the original cover can be losslessly restored after the embedded message is extracted.

Many RDH methods have been proposed since it was introduced. Fridrich and Goljan [1] presented a universal framework for RDH, in which the embedding process is divided into three stages (See Fig. 1). The first stage losslessly extracts 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 payloads (messages). The third stage embeds messages into the feature sequence and generates 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, by which replace the original features to generate the marked cover. Therefore, after extracting the message, the receiver can restore the original cover by decompressing the features. Fridrich and Goljan [1] suggested features obtained by exploiting characteristics of certain image formats, e.g., texture complexity for spatial images and middle-frequency discrete cosine transform (DCT) coefficients for JPEG images. Celik *et al.* [2] extended Fridrich and Goljan's scheme by predicting multiple least significant bit (LSB) planes. The same idea proposed in [1] can be also used for reversible data embedding into binary images [3], [4] or videos [5], [6].

Larger embedding capacity can be achieved by constructing a longer feature sequence that can be perfectly compressed. One of such constructions is difference expansion (DE), which was first proposed by Tian [7], in which the features are the differences between two neighboring pixels. The features are compressed by expansion, i.e., the differences are multiplied by 2, and thus, the LSBs of the differences can be used for embedding messages. Alattar [8] generalized Tian's method by applying DE to a vector of pixels. Kim *et al.* [9] improved the DE method by reducing the size of the location map used to communicate position information of expandable difference values. The methods proposed in [10] and [11] can achieve better performance by applying DE to the prediction errors.

Another well-known strategy for RDH is histogram shift (HS), in which the histogram of the image is used as the compressible features because the distribution of the pixel values of an image is usually uneven. To compress the histogram, Ni *et al.* [12] proposed to select a peak bin and a zero bin and shift the bins between them toward the zero bin by one

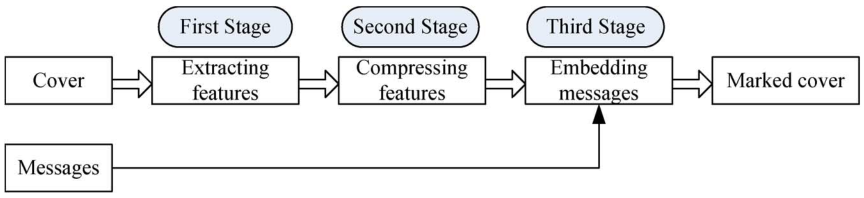

Fig. 1. Diagram for the framework of RDH at the sender side.

Fridrich et al.'s Framework	Kalker et al.'s Framework	Proposed Framework
No side information is used.	The marked cover is used to reconstruct cover.	The marked cover is used to reconstruct the cover. The reconstructed cover is used to extract messages.

Fig. 2. Side information used at the receiver side in three frameworks.

step. Therefore, the peak bin's neighboring bin, which is now emptied out, and the peak bin can be used to represent "1" and "0," respectively. It is easy to see that a steeper histogram implies larger capacity, and, usually, the histogram of residuals is quite steep. Thus, most state-of-the-art methods apply HS to residuals of the image [13], [14].

Both DE- and HS-based schemes use integer features and special methods to compress the features. As for DE, the features (differences) are compressed by expansion operation, and as for HS, the features (histogram) are compressed by shifting operation. There is a common character in both these schemes, that is, the distortion to the original cover is mainly introduced by the special compressing manners. By contrast, Fridrich and Goljan's schemes [1] use a binary feature sequence and a generic compression algorithm, e.g., the arithmetic coder, and no distortion must be introduced by the compression. According to such differences, we divide RDH into two types as follows.

- **Type I**. The features can be formulated as a binary sequence and can be compressed by using a generic compression algorithm. The methods in [1]–[6] belong to Type I.
- **Type II**. The features are nonbinary and compressed in some specific manners. Both DE-based [7]–[11] and HS-based methods [12]–[14] belong to Type II.

For Type-I RDH, the problem is formulated as how to reversibly embed data into a compressible binary sequence 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 [15]. In [15], the authors obtained the rate–distortion function, i.e., the upper bound of the embedding rate under a given distortion constraint, and, by dividing the cover into disjoint blocks, they proposed a recursive code construction, which consists of a nonreversible data embedding code and a conditional compression code. In fact, Kalker and Willems noted that the receiver can reconstruct the cover with the help of the marked cover, and thus, the sender can compress the cover under the condition of the marked cover. That is why the recursive construction is efficient.

In our previous paper [17], we improved the recursive construction by using not only conditional compression but also conditional embedding, which enables us to design an efficient embedding algorithm and a perfect compressing method to approach the rate–distortion bound. In fact, we noted that the receiver could extract messages from the marked cover with the help of the reconstructed cover because of reversibility. In Fig. 2, the side information exploited at the receiver side in the proposed framework is compared with those used in two previous frameworks.

However, there are still limitations in three aspects in [17]. First, we construct embedding codes by improving the decompression algorithm of run-length coding, by which the recursive code construction is close to but cannot reach the rate–distortion bound. Second, the codes in [17] are restricted to some discrete embedding rates and cannot approach the maximum embedding rate at the least admissible distortion. Third, the codes are restricted to improve Type-I RDH for spatial images, and how to improve Type-II RDH by binary codes is still a problem.

In this paper, we generalize the code construction in [17] by using a general decompression algorithm as the embedding code and extend the applications to Type-II RDH. Compared with our preliminary paper [17], the new contributions of this paper are as follows.

- We prove that the recursive code construction can reach the rate–distortion bound when the decompression/compression algorithms used in the code are optimal, which establishes equivalence between source coding and RDH for binary covers.
- With the decompression of the adaptive arithmetic coder (AAC) as the embedding code, the proposed codes realize continuous embedding rates and reach the maximum embedding rate at the least admissible distortion.
- A method is presented to improve integer-operation-based RDH (Type II) by the proposed binary codes, which are also applied to Type-I RDH for JPEG and binary images.

The rest of this paper is organized as follows. The coding model, rate–distortion function, and recursive code construction are briefly introduced in Section II. The proposed codes with proof of optimality are elaborated in Section III. Some implementation issues and performance comparison are discussed in Section IV. In Section V, we improve Type-I schemes, including schemes for spatial, JPEG, and, binary images. Section VI addresses how to improve Type-II schemes with the proposed binary codes. This paper is concluded with a discussion in Section VII.

II. Coding Model and Recursive Construction

A. 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 entropy by $H(X)$ and conditional entropy by $H(X|Y)$. Particularly, 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 $p_1 + p_2 + p_3 = 1$.

To do RDH, a compressible feature sequence should be first extracted from the original cover. For Type-I schemes, the features can be usually represented by a binary sequence. Therefore, we directly take the binary feature sequence as the cover to discuss the coding method and follow the notation established in [15].

Assume that there is a memoryless source producing binary compressible cover sequence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ 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 an L-bit message $\mathbf{m} = (m_1, m_2, \ldots, m_L)$ into $\mathbf{x}$ to get the marked cover $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ with d modifications on average, we define the embedding rate as $\rho = L/N$ and the distortion as $\Delta = d/N$. For any given distortion constraint, we desire the embedding rate as high as possible.

A direct construction for RDH was proposed by Fridrich and Goljan [1] as follows. First, compress the cover $\mathbf{x}$ into a string $\mathrm{Comp}(\mathbf{x})$ with a lossless compression algorithm $\mathrm{Comp}(\cdot)$. The length of $\mathrm{Comp}(\mathbf{x})$ is approximately equal to $NH_2(p_0)$. Therefore, we can averagely append $N(1 - H_2(p_0))$ bits of message $\mathbf{m}$ after $\mathrm{Comp}(\mathbf{x})$ to obtain $\mathbf{y} = \mathrm{Comp}(\mathbf{x})\|\mathbf{m}$. The recipient can extract the message $\mathbf{m}$ from $\mathbf{y}$ and reconstruct $\mathbf{x}$ by decompressing $\mathrm{Comp}(\mathbf{x})$. As the bits of $\mathrm{Comp}(\mathbf{x})$ 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 $(1 - H_2(p_0))$, which, in fact, is the maximum achievable embedding rate. If we only need to embed a shorter message with length equal to $\alpha N(1 - H_2(p_0))$ for some $\alpha < 1$, we can execute the aforementioned method on a fraction α of the symbols in $\mathbf{x}$. In this case, the embedding rate $\rho = \alpha(1 - H_2(p_0))$ and the distortion $\Delta = \alpha/2$. Therefore, for the distortion constraint Δ, this simple method can achieve a rate–distortion curve, i.e.,

$$\rho_{\mathrm{sim}}(p_0, \Delta) = 2\Delta\left(1 - H_2(p_0)\right). \tag{1}$$

Virtually, the simple method above is not optimal.

The maximum achievable embedding rate within the distortion constraint Δ is called the capacity under the distortion Δ. The following theorem proved by Kalker and Willems [15] gives the expression of capacity.

Theorem 1: 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 [15]

$$\rho_{\mathrm{rev}}(p_0, \Delta) = H_2\left(\max(p_0 - \Delta, 1/2)\right) - H_2(p_0). \tag{2}$$

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

B. Recursive Construction

To approach the rate–distortion curve, Kalker and Willems [15] proposed a recursive embedding method, which consists of a nonreversible embedding code and a conditional compression algorithm. First, select a nonreversible embedding code $\mathcal{E}$ with distortion Δ and embedding rate ρ. Assume the binary cover sequence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is sufficiently long. The sequence is segmented into disjoint blocks of length K, such that $\mathbf{x} = \mathbf{x}_1\|\mathbf{x}_2\|\ldots\|\mathbf{x}_{N/K}$. Without loss of generality, we assume that N/K is a sufficiently large integer. With the embedding code $\mathcal{E}$, $K\rho$ bits of message $\mathbf{m}_1$ can be embedded into the first host block $\mathbf{x}_1$, resulting in the first marked block $\mathbf{y}_1$. The recipient can reconstruct $\mathbf{x}_1$ under the condition of known-$\mathbf{y}_1$ after she receives $\mathbf{y}_1$. 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 $H(\mathbf{x}_1|\mathbf{y}_1)$ on average. This compressed sequence is embedded into the second block $\mathbf{x}_2$, averagely leaving room for $K\rho - H(\mathbf{x}_1|\mathbf{y}_1)$ bits of auxiliary message. Similarly, the information for reconstructing $\mathbf{x}_2$ is embedded into $\mathbf{x}_3$. This process is recursively continued until $\mathbf{x}_{N/K-1}$. For the last block $\mathbf{x}_{N/K}$, the simple method described in Section II-A is used to complete a full RDH method. When N and N/K are large enough, the distortion of this method is equal to the distortion of code $\mathcal{E}$, and the embedding rate is equal to $\rho - H(\mathbf{x}_1|\mathbf{y}_1)/K$.

This recursive construction performs better than the simple method because of two key points: 1) The data is embedded by an efficient nonreversible embedding code, and 2) the cover block is compressed under the condition of the marked block. However, the above recursive construction cannot approach the upper bound (2).

III. Improved Recursive Construction

A. Motivations and Overall Framework

In this section, we will improve the recursive construction to approach the rate–distortion bound for any given distortion constraint. To do that, we first observe the rate–distortion function (2), which shows that the maximum capacity is equal to $1 - H_2(p_0)$, and it can be achieved when distortion $\Delta = p_0 - 1/2$. In Fig. 3, we draw the rate–distortion curves 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 $0 \leq \Delta \leq p_0 - 1/2$. On the other hand, in [15, Corollary 1], Kalker and Willems proved that the optimal

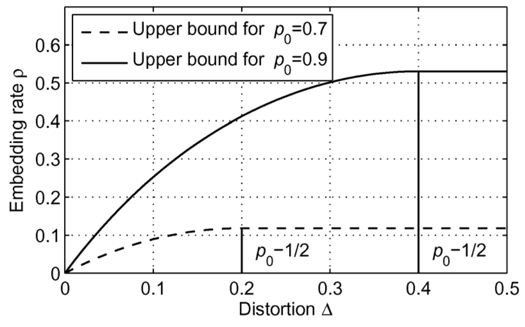

Fig. 3. Maximum capacity for $p_0 = 0.7$ and 0.9.

embedding manner for $0 \le \Delta \le p_0 - 1/2$ is that only the most probable symbol, i.e., "0," is allowed to be modified.

Inspired by the above observation, we improve the recursive construction as follows. We only embed messages into "0's" of the cover block $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,k})$ to obtain the marked block $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,k})$, and thus, only "0's" in $\mathbf{x}_i$ will be modified for the ith block such that $1 \le i \le N/K - 1$. Therefore, for the position j such that $y_{i,j} = 0$, the corresponding $x_{i,j}$ must be also 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 $\mathbf{x}_i$, which is denoted by $\mathbf{x}'_i$, and then compress $\mathbf{x}'_i$ by a lossless compression algorithm Comp$(\cdot)$. This method will greatly improve the compression rate because most symbols in $\mathbf{x}_i$ have been compressed by deletion. The compressed $\mathbf{x}'_i$, which is denoted by Comp$(\mathbf{x}'_i)$, cascaded with an auxiliary message, is 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 in a backward manner. To extract message from $\mathbf{y}_i$, we first extract 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 find the positions of "0's" in $\mathbf{x}_i$. According to the positions of "0's" in $\mathbf{x}_i$, we can extract message from $\mathbf{y}_i$.

The detailed process and an example for embedding and extraction will be described in Section III-B.

B. Improved Recursive Construction

We first need an embedding algorithm for embedding data only into zero symbols, which, in fact, is a special case of the coding model in Section II with taking $p_0 = 1$. By (2), the capacity for $p_0 = 1$ is equal to $H_2(\Delta)$, which implies that the optimal method for embedding data into only "0's" is equivalent to decompression.

For example, assume C is a lossless compression algorithm that has compression rate $C(\Delta)$ for a memoryless binary source with $p_1 = \Delta$, and then, we can use the decompression algorithm of C to embed data into zero symbols. In fact, into an n-bit zero cover, we can embed $nC(\Delta)$ bits of random messages, on average, by decompressing the message into an n-bit sequence by setting $p_1 = \Delta$ as the parameter of decompression. To extract the message, we only need to compress the n-bit sequence back to the $nC(\Delta)$ bits of messages. Obviously, the embedding rate is equal to $C(\Delta)$, and the distortion is equal to Δ because, on average, $n\Delta$ "0's" are changed to "1's" in the embedding process. Therefore, if the compression algorithm C is optimal, i.e., the compression rate $C(\Delta) = H_2(\Delta)$, we just achieve the embedding capacity.

To improve the recursive construction in [15] and [16], we use the decompression algorithm of C as the embedding code and design a corresponding conditional compression algorithm for the cover based on C.

Assume that the binary cover sequence $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is generated from a memoryless source satisfying $P(x_i = 0) = p_0$ and $P(x_i = 1) = p_1$. To embed messages into $\mathbf{x}$ reversibly with distortion constraint Δ, we first divide $\mathbf{x}$ into N/K disjoint blocks of length K, such that $\mathbf{x} = \mathbf{x}_1 \| \mathbf{x}_2 \| \ldots \| \mathbf{x}_{N/K}$. In each block, we only embed messages into zero symbols via the decompression algorithm of C. Note that, when the distortion on the sequence $\mathbf{x}$ is Δ, the distortion on zero symbols is Δ/p_0 because only "0's" will be modified. Therefore, we use Δ/p_0 as the parameter of the decompression algorithm of C and, on average, decompress the first

$$Kp_0 C\left(\frac{\Delta}{p_0}\right) \tag{3}$$

bits of messages into a Kp_0-bit sequence, which is denoted by $\mathbf{y}'_1$. The sequence $\mathbf{y}'_1$ will include $Kp_0\Delta/p_0 = K\Delta$ "1's," on average. By replacing the Kp_0 "0's" in $\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,K})$ with $\mathbf{y}'_1$, we just obtain the first marked block $\mathbf{y}_1 = (y_{1,1}, \ldots, y_{1,K})$, which introduces $K\Delta$ changes, on average.

Therefore, at the position j, such that $1 \le j \le K$ and $y_{1,j} = 0$, the corresponding $x_{1,j}$ must be also equal to "0" because no "1" in $\mathbf{x}_1$ has been flipped to "0." 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 in a subsequence of $\mathbf{x}_1$. Denote this subsequence by $\mathbf{x}'_1$, and thus, $\mathbf{x}'_1 = \{x_{1j} | 1 \le j \le K, y_{1j} = 1\}$. Note that the proportion of "0's" in $\mathbf{y}_1$ is equal to the ratio of nonmodified "0's" in $\mathbf{x}_1$, that is, $p_0(1 - \Delta/p_0)$, and thus, the ratio of "1's" in $\mathbf{y}_1$ is equal to $1 - p_0(1 - \Delta/p_0) = 1 - p_0 + \Delta$. Therefore, the average length of $\mathbf{x}'_1$ is equal to $K(1 - p_0 + \Delta)$. 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 $1 - p_0 + \Delta$, and we can reconstruct $\mathbf{x}_1$ by replacing "1's" in $\mathbf{y}_1$ by the symbols of $\mathbf{x}'_1$.

Furthermore, we compress $\mathbf{x}'_1$ with the lossless compression algorithm C. Denote the proportion of "0's" and "1's" in $\mathbf{x}'_1$ by q_0 and q_1, respectively, which can be easily computed as follows:

$$q_0 = \frac{\Delta}{1 - p_0 + \Delta}, \quad q_1 = \frac{1 - p_0}{1 - p_0 + \Delta}. \tag{4}$$

Therefore, $\mathbf{x}'_1$ can be compressed with the rate $C(q_0)$. Denote the compressed information by Comp$(\mathbf{x}'_1)$. In summary, when $\mathbf{y}_1$ is known, the amount of information needed to reconstruct $\mathbf{x}_1$, i.e., the length of Comp$(\mathbf{x}'_1)$, is, on average, equal to

$$K(1-p_0+\Delta)C(q_0) = K(1-p_0+\Delta)C\left(\frac{\Delta}{1 - p_0 + \Delta}\right). \tag{5}$$

By using the decompression algorithm of C, we embed Comp$(\mathbf{x}'_1)$ into the "0's" of the next block $\mathbf{x}_2$, leaving room for embedding $K(p_0 C(\Delta/p_0) - (1 - p_0 + \Delta)C(q_0))$ bits of

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

Fig. 4. Example on the improved recursive construction.

auxiliary messages, on average, and resulting in the second marked block $\mathbf{y}_2$. The information for reconstructing $\mathbf{x}_2$, which is denoted by $\mathrm{Comp}(\mathbf{x}_2')$, is embedded into the third block $\mathbf{x}_3$. This process is recursively continued 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 $K(1-C(p_0))$ bits, on average, into which we embed $\mathrm{Comp}(\mathbf{x}'_{N/K-1})$ for reconstructing the second last block $\mathbf{x}_{N/K-1}$. In addition, we also embed the overhead information, such as parameters K, p_0, and Δ, 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 ith block $\mathbf{y}_i$, for $1 \le i \le N/K-1$, 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 find the positions of "0's" in $\mathbf{x}_i$, according to which we extract a subsequence from $\mathbf{y}_i$ and compress this subsequence by C with Δ/p_0 as the parameter to obtain the message hidden in $\mathbf{y}_i$.

When N and N/K are large enough, the distortion of the method above is just Δ, and by (3) and (5), the corresponding embedding rate $\rho(p_0,\Delta)$ can be calculated by

$$\rho(p_0,\Delta) = p_0 C\left(\frac{\Delta}{p_0}\right) - (1-p_0+\Delta) C\left(\frac{\Delta}{1-p_0+\Delta}\right). \quad (6)$$

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

Example 1: As shown in Fig. 4, the first cover block $\mathbf{x}_1$ consists of nine "0's" and one "1," and thus, $p_0 = 0.9$. Set the distortion constraint $\Delta = 0.2$, and therefore, the distortion on zero symbols is equal to $\Delta/p_0 = 2/9$. With 2/9 as the parameter of the decompression algorithm, assume that the first 7 bits of the message are decompressed into a 9-bit sequence $\mathbf{y}_1'$. By replacing "0's" in $\mathbf{x}_1$ with $\mathbf{y}_1'$, we generate the first marked block $\mathbf{y}_1$. Denote the index set of "1's" in $\mathbf{y}_1$ by Ind_1, and thus, $\mathrm{Ind}_1 = \{3,7,10\}$, 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 $\mathrm{Comp}(\mathbf{x}_1')$ and then 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 "1's" 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-bit decompressed sequence, which is just $\mathbf{x}_1'$. Thus, we can reconstruct $\mathbf{x}_1$ by replacing the "1's" in $\mathbf{y}_1$ by $\mathbf{x}_1'$. After that, we find the index set of "0's" in $\mathbf{x}_1$, such that $\mathrm{Ind}_0 = \{1,2,4,5,6,7,8,9,10\}$, 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 by compressing $\mathbf{y}_1'$.

C. Optimality

The next theorem shows that the proposed code construction is optimal as long as the compression algorithm C is optimal.

Theorem 2: The proposed codes reach the embedding capacity ρ_{rev} [see eq. (2)] when the compression rate of C is equal to entropy, i.e., $C(p) = H_2(p)$ for $0 \le p \le 1/2$.

Proof: As mentioned in Section III-A, the maximum capacity $\rho_{\max} = 1 - H_2(p_0)$ appears at $\Delta = p_0 - 1/2$, and thus, we only need to consider the distortion range $[0, p_0 - 1/2]$. When $0 \le \Delta \le p_0 - 1/2$, (2) can be rewritten as

$$\rho_{\mathrm{rev}}(p_0,\Delta) = H_2(p_0-\Delta) - H_2(p_0). \quad (7)$$

On the other hand, when the compression rate of C is equal to entropy, the embedding rate ρ [see eq. (6)] of the proposed codes can be rewritten as

$$\rho(p_0,\Delta) = p_0 H_2\left(\frac{\Delta}{p_0}\right) - (1-p_0+\Delta) H_2\left(\frac{\Delta}{1-p_0+\Delta}\right). \quad (8)$$

By the grouping property of entropy [18], on one hand, we have

$$\begin{aligned} &H_3(p_0-\Delta, 1-p_0, \Delta) \\ &\quad = H_2(p_0-\Delta) + (1-p_0+\Delta) H_2\left(\frac{\Delta}{1-p_0+\Delta}\right) \end{aligned} \quad (9)$$

and, on the other hand, we have

$$\begin{aligned} H_3(p_0-\Delta, 1-p_0, \Delta) &= H_3(1-p_0, p_0-\Delta, \Delta) \\ &= H_2(1-p_0) + p_0 H_2\left(\frac{\Delta}{p_0}\right) \\ &= H_2(p_0) + p_0 H_2\left(\frac{\Delta}{p_0}\right). \end{aligned} \quad (10)$$

Subtracting (8) from (7), we have

$$\begin{aligned}
&\rho_{\text{rev}}(p_0, \Delta) - \rho(p_0, \Delta) \\
&= [H_2(p_0 - \Delta) - H_2(p_0)] \\
&\quad - \left[p_0 H_2\left(\frac{\Delta}{p_0}\right) - (1 - p_0 + \Delta) H_2\left(\frac{\Delta}{1 - p_0 + \Delta}\right)\right] \\
&= \left[H_2(p_0 - \Delta) + (1 - p_0 + \Delta) H_2\left(\frac{\Delta}{1 - p_0 + \Delta}\right)\right] \\
&\quad - \left[H_2(p_0) + p_0 H_2\left(\frac{\Delta}{p_0}\right)\right] \\
&= H_3(p_0 - \Delta, 1 - p_0, \Delta) - H_3(p_0 - \Delta, 1 - p_0, \Delta) \\
&= 0 \qquad (11)
\end{aligned}$$

and thus, we obtain the theorem. Herein, the third equality follows from (9) and (10).

IV. Implementation Issues and Performance Comparison

A. Overhead in the Last Block

In this paper, we propose the AAC [19] as the compression algorithm and the decompression algorithm of AAC as the embedding code, which can approximately reach the entropy.

In practice, we should set a proper length for the last block. Denote the estimated length of the last block by K_{last}. After compression, the left room in the last block is about $K_{\text{last}}(1 - H_2(p_0))$ bits, in which we will embed not only the information for reconstructing the second last block but also some overhead information. On one hand, to reconstruct the second last block, we need, at most, $K/2$ bits because the number of "1's" in $\mathbf{y}_{\text{last}-1}$ is not more than $K/2$. On the other hand, the overhead consists of some parameters necessary to the recipient, the length of which is denoted by L_{over}. Thus, the estimated length of the last block K_{last} is enough, if

$$K_{\text{last}}\,(1 - H_2(p_0)) \geq \frac{K}{2} + L_{\text{over}} \qquad (12)$$

which implies that the lower bound of the length of the last block is

$$\underline{K_{\text{last}}} = \left\lceil \frac{K/2 + L_{\text{over}}}{(1 - H_2(p_0))} \right\rceil. \qquad (13)$$

Therefore, for a cover sequence of length N, the estimated number of K-bit blocks, i.e., Num, is equal to

$$Num = \left\lfloor \frac{N - \underline{K_{\text{last}}}}{K} \right\rfloor. \qquad (14)$$

Thus, we get the estimated length of the last block as follows:

$$K_{\text{last}} = N - K \cdot Num. \qquad (15)$$

Note that the estimated length K_{last} may be not enough in practice when the ratio of "0's" in the last block is far from p_0, that is, the overall ratio of "0's." Therefore, we should modify the value of K_{last}. First, take the last K_{last} bits from the N-bit cover and calculate the ratio of "0's" in these bits, which is denoted by p_0'. Second, by using p_0', we modify the lower bound of the length of the last block as

$$\underline{K'_{\text{last}}} = \left\lceil \frac{K/2 + L_{\text{over}}}{(1 - H_2(p_0'))} \right\rceil \qquad (16)$$

and modify the number of K-bit blocks as

$$Num' = \left\lfloor \frac{N - \underline{K'_{\text{last}}}}{K} \right\rfloor. \qquad (17)$$

Thus, the length of the last block is given by

$$K'_{\text{last}} = N - K \cdot Num'. \qquad (18)$$

$Comp(\mathbf{x}_{last})$	Comp($\mathbf{x}'_{last-1}$)	...	10 bits of Δ	10 bits of p_0'	10 bits of p_0

Fig. 5. Structure of the last marked block.

To extract messages exactly, the recipient should know the length K of previous blocks and the parameters p_0, p_0', and Δ needed by the compression/decompression algorithm. Block length K can be fixed in advance by the sender and the recipient. Thus, the overhead consists of p_0, p_0', and Δ. In practice, it is enough to take three decimal for each parameter, and thus, 30 bits are enough for the overhead, i.e., $L_{\text{over}} = 30$. We embed the overhead into the end of the last block. The structure of the last block is shown in Fig. 5.

When extracting messages from the N-bit marked sequence, the recipient first reads the parameters p_0, p_0', and Δ from the last 30 bits of the sequence and then computes q_0 by (4). Furthermore, the recipient obtains K'_{last} by (13) (14) (15) (16) (17) (18) and determines the start point of the last block and then reads bits successively and decompresses them with p_0' as the parameter. When the length of the decompressed sequence is equal to K'_{last}, stop the process of decompression because the last cover block $\mathbf{x}_{\text{last}}$ has been obtained. Continue reading bits from the last marked block and decompress them with q_0 as the parameter until the length of the decompressed sequence is equal to the number of "1's" in the second last marked block $\mathbf{y}_{\text{last}-1}$. This decompressed sequence is just $\mathbf{x}'_{\text{last}-1}$, which will be used to reconstruct the second last cover block $\mathbf{x}_{\text{last}-1}$. Next, with Δ/p_0 as the parameter, the extraction process, as described in Section III-B, will be continued until the first marked block.

B. Determining the Parameter Δ

In practice, when we need to embed an M-bit message into an N-bit cover, we should determine the minimal distortion Δ as the parameter for realizing the embedding rate $\rho = M/N$. We first calculate the length of the last block K'_{last} by (13) (14) (15) (16) (17) (18) and get the modified embedding rate $\rho' = M/(N - K'_{\text{last}})$. Theorem 2 implies that the rate–distortion bound (2) can be used to estimate the embedding rate of the proposed codes when the last block is neglected. Therefore, we

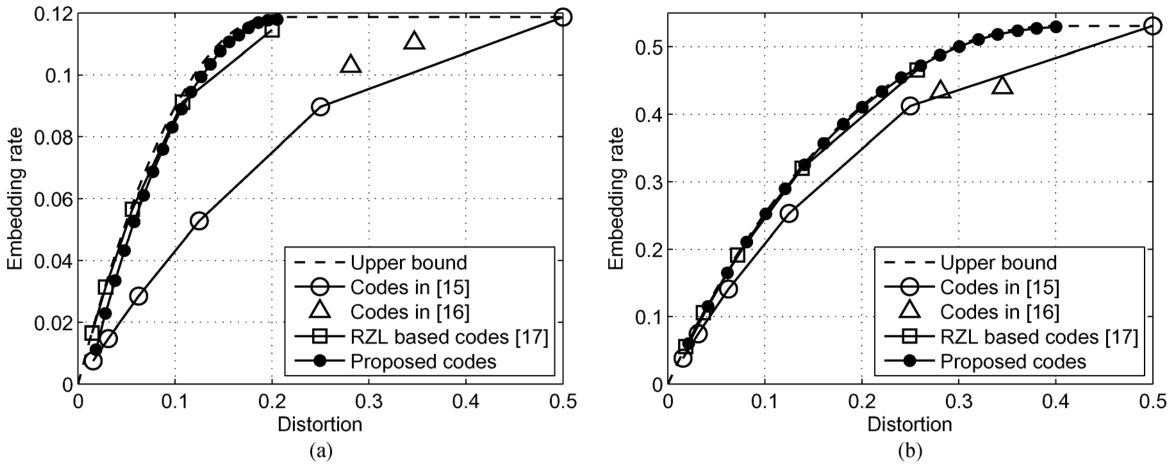

Fig. 6. Comparison of embedding rate versus distortion between the proposed codes and codes in [15]–[17]. (a) $p_0 = 0.7$. (b) $p_0 = 0.9$.

use the reverse function of (2) to estimate the needed parameter Δ, that is

$$\Delta = \rho_{\mathrm{rev}}^{-1}(p_0, \rho') \tag{19}$$

which will be embedded into the tail of the cover, as shown in Fig. 5. Note that the Δ obtained by (19) is not distortion on the original N-bit cover but on its first $N - K'_{\mathrm{last}}$ bits.

C. Performance Comparison

In this paper, we propose the decompression algorithm of the AAC [19] as the embedding algorithm, whereas in our previous paper [17], we used the improved reverse zero-run length (RZL) coding [6] as the embedding algorithm. In fact, the improved RZL can be viewed as a modified version of the decompression algorithm of run-length coding. When we compare these two methods, the AAC is used as the compression algorithm in both cases.

We also compare these two methods with the codes proposed in [15] and [16]. In the original recursive construction [15], Kalker and Willems used Hamming matrix embedding [20] as the nonreversible data embedding code, by which they can embed k bits of message into $2^k - 1$ cover bits with, at most, one modification. The Hamming codes modify "0's" and "1's" with equal probability. Maas *et al.* [16] improved the original recursive construction by adjusting Hamming code to change more "0's" than "1's" for the case $k = 2$.

The simulation results of these coding methods are compared for $p_0 = 0.7$ and 0.9. The simulation results are obtained by embedding random messages into a 2^{16}-bit cover. In the experiments, we set the length of cover blocks $K = 200$ for the proposed method and the RZL-based method in [17]. As shown in Fig. 6, the RZL-based method is only somewhat better than the proposed method at small embedding rates for $p_0 = 0.7$, but it generates codes with only sparse embedding rates and cannot reach the maximum capacity for $p_0 = 0.9$. The proposed method generates codes with continuous embedding rates and reaches the maximum capacity at distortion $p_0 - 1/2$. Although the codes proposed in [16] are close to the maximum capacity for $p_0 = 0.7$, they need larger distortion than $p_0 - 1/2$.

V. Applications in Type-I Schemes

The coding method above can be directly applied to data hiding schemes that belong to Type I, such as those in [1]–[6]. Next, we use the proposed codes to improve RDH schemes for spatial, JPEG, and binary images, respectively.

A. Improving the RS Scheme for Spatial Images

The RS method [1] is proposed for spatial images by constructing compressible features based on texture complexity. Assume the covers are 8-bit gray scale images. The image is first 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 integer A. For instance, when $A = 1$, the flipping function is as follows:

$$F:\ 0 \leftrightarrow 1,\ 2 \leftrightarrow 3,\ 4 \leftrightarrow 5, \ldots,\ 254 \leftrightarrow 255. \tag{20}$$

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

$$f(G) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|. \tag{21}$$

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

$$\begin{aligned} G \in R &\Leftrightarrow f\left(F(G)\right) > f(G) \\ G \in S &\Leftrightarrow f\left(F(G)\right) < f(G) \\ G \in U &\Leftrightarrow f\left(F(G)\right) = f(G). \end{aligned} \tag{22}$$

For typical pictures, adding some noise will lead to an increase in the variation; hence, 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 $p_0 > 1/2$.

Fig. 7. Test images sized 512 × 512. (a) Lenna.pgm. (b) Baboom.pgm. (c) Boat.pgm. (d) Barbara.pgm. (e) Goldrill.pgm. (f) Peppers.pgm.

TABLE I
PATTERNS EXTRACTED FROM TEST IMAGES

Images	English text		Pig		Boy		Butterfly	
Patterns	(1 0 1 0)	(1 1 1 0)	(0 0 1 0)	(0 1 1 1)	(0 1 0 0)	(1 1 1 0)	(1 0 0 0)	(1 1 1 0)
Numbers	1914	14	1381	23	1841	3	788	1

Flipping between "0" and "1" can be realized by applying F to the corresponding pixel group.

Usually, larger amplitude A implies larger capacity, but it also implies 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 six 8-bit gray scale images sized 512 × 512 [21] (see Fig. 7). We observe that the ratio of "0's" p_0 in the RS sequence varied from 0.54 to 0.87. In our coding method, we embed messages with the maximum embedding rate, i.e., taking $\Delta = p_0 - 1/2$ as the parameter. As shown in Fig. 8, the proposed codes increase the peak signal-to-noise ratios (PSNRs) for various kinds of test images. Herein, the embedding rate is defined as bits carried by per pixel (bpp).

B. Improving the Scheme for JPEG Images

In this subsection, we apply the codes to the reversible embedding scheme for JPEG images proposed by Firdrich and Goljan [1]. In the method in [1], quantized DCT coefficients that are equal to 0 and 1 at middle or high frequency are selected to form a compressible binary sequence. In our experiments, the test images are generated by compressing test images in Fig. 7 into a JPEG format with quality factor 80. We construct the binary cover by extract 0–1 coefficients from 11 positions, such as (3, 3), (2, 4), (4, 2), (1, 5), (5, 1), (3, 4), (4, 3), (2, 5), (5, 2), (1, 6), and (6, 1), from every 8 × 8 block of quantized DCT coefficients. Random messages are embedded into the binary cover by using Fridrich and Goljan's method [1] and the proposed codes with several kinds of embedding rates. As shown in Fig. 9, the proposed codes can improve the method in [1] by increasing PSNRs from 1 to 4 dB.

C. Improving PS Scheme for Binary Images

As another example of applications, we use the proposed codes to improve the pattern substitution (PS) method for RDH in binary images [4]. Denote an $m \times n$ binary image by $\mathbf{I}$, such that $\mathbf{I}(i, j) \in \{0, 1\}, 1 \le i \le m, 1 \le j \le n$. When using $\mathbf{I}$ as the cover image, the PS method first computes the image differencing $\mathbf{D}$ by EXCLUSIVE-OR operation between neighbor pixels of $\mathbf{I}$, such that

$$\mathbf{D}(i,j) = \begin{cases} \mathbf{I}(i,j), & \text{if } i = 1 \text{ and } j = 1 \\ \mathbf{I}(i,j) \oplus \mathbf{I}(i-1,j), & \text{if } i \ne 1 \text{ and } j = 1 \\ \mathbf{I}(i,j) \oplus \mathbf{I}(i,j-1), & \text{otherwise.} \end{cases} \quad (23)$$

TABLE II
NUMBER OF MODIFICATIONS FOR DIFFERENT LENGTHS OF MESSAGES EMBEDDED BY THE PS METHOD AND THE IMPROVED PS METHOD

Test image	English text				
Message length(bits)	1567	1461	1249	720	386
PS method	1818	1734	1484	985	623
Improved PS	1324	1117	779	343	168
Test image	Pig				
Message length(bits)	1060	994	735	505	257
PS method	–	1396	1108	891	653
Improved PS	1032	774	559	297	131
Test image	Boy				
Message length(bits)	1556	1453	1221	733	363
PS method	1626	1541	1305	806	411
Improved PS	1275	1059	766	394	161
Test image	Butterfly				
Message length(bits)	680	642	550	408	281
PS method	362	370	293	234	154
Improved PS	326	285	183	121	70

In the image differencing $\mathbf{D}$, the symbol "1's" indicate the edges of the original image $\mathbf{I}$, around which modifications are usually insensible. Scan the image $\mathbf{D}$ from left to right and from top to bottom, and divide it into disjoint blocks containing four pixels. Search all these 4-bit patterns to find a pair of patterns, such that there is a large gap between their occurrence frequencies. Denote the pattern with large numbers by PM and the pattern with small numbers by PF. By assigning "0" to PM and "1" to PF, we just get a compressible binary cover. To keep visual quality, neither PM nor PF can be all-zero vectors, and the number of modifications for PS should be as small as possible. In the PS method, a location map is used to record the positions of PFs, and then, the message bits combining with the location map are embedded into the PM–PF (0–1) sequence. Modifications are realized by substitution between patterns PM and PF. Denote the number of PFs by N_{PF}, and the location map will cost $(m+n)N_{\mathrm{PF}}$ bits of capacity. Therefore, the number of PFs must be very small.

We improve the PS method by applying the proposed codes to embed messages into the PM–PF sequence with five kinds of message lengths, which does not need a location map. Because the cover sequence is short, we take the length of the cover blocks $K = 100$ in the experiments. For each 256 × 256 test images in Fig. 10, the selected pattern pairs with frequencies are listed in Table I. Note that, in one PS, we need to flip one pixel in "Butterfly" and two pixels in other images according to the

VII. Conclusion

Most state-of-the-art RDH schemes use a strategy with separate processes of feature compression and message embedding. Kalker and Willems [15] noted that a higher embedding rate under a 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 proposed code construction significantly outperforms previous codes [15], [16] and is proved to be optimal when the compression algorithm reaches entropy.

The current codes are designed for binary covers and, thus, can significantly improve Type-I schemes based on binary feature sequences. By slightly modifying the HS manner, we found that the proposed binary codes can be also partly applied to Type-II schemes and improve their performance, but the improvement is not so significant as that for Type-I schemes. Note that we only use two simple methods to modify HS, and therefore, one interesting problem is whether there exists other more effective modifying methods or not. Another problem is how to design recursive codes for gray scale covers. We will pay our attention to these problems in further works.

References

[1] J. Fridrich and M. Goljan, "Lossless data embedding for all image formats," in *Proc. EI SPIE, Security Watermarking Multimedia Contents IV*, San Jose, CA, 2002, vol. 4675, pp. 572–583.

[2] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Trans. Image Process.*, vol. 14, no. 2, pp. 253–266, Feb. 2005.

[3] S. Li and A. C. Kot, "Privacy protection of fingerprint database using lossless data hiding," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2010, pp. 1293–1298.

[4] Y.-A. Ho, Y.-K. Chan, H.-C. Wu, and Y.-P. Chu, "High-capacity reversible data hiding in binary images using pattern substitution," *Comput. Standards Interfaces*, vol. 31, no. 4, pp. 787–794, Jun. 2009.

[5] R. Du and J. Fridrich, "Lossless authentication of MPEG-2 video," in *Proc. IEEE Int. Conf. Image Process.*, 2002, vol. 2, pp. II-893–II-896.

[6] K. Wong, K. Tanaka, K. Takagi, and Y. Nakajima, "Complete video quality-preserving data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 10, pp. 1499–1512, Oct. 2009.

[7] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[8] A. M. Alattar, "Reversible watermark using difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.

[9] H.-J. Kim, V. Sachnev, Y. Q. Shi, J. Nam, and H.-G. Choo, "A novel difference expansion transform for reversible data embedding," *IEEE Trans. Inf. Forensic Security*, vol. 3, no. 3, pp. 456–465, Sep. 2008.

[10] D. M. Thodi and J. J. Rodriguez, "Expansion embedding techniques for reversible watermarking," *IEEE Trans. Image Process*, vol. 16, no. 3, pp. 721–730, Mar. 2007.

[11] Y. Hu, H.-K. Lee, and J. Li, "DE-based reversible data hiding with improved overflow location map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 250–260, Feb. 2009.

[12] Z. Ni, Y. Q. Shi, N. Ansari, and S. Wei, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[13] P. Tsai, Y. C. Hu, and H. L. Yeh, "Reversible image hiding scheme using predictive coding and histogram shifting," *Signal Process.*, vol. 89, no. 6, pp. 1129–1143, Jun. 2009.

[14] L. X. Luo, Z. Y. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 187–193, Mar. 2010.

[15] T. Kalker and F. M. Willems, "Capacity bounds and code constructions for reversible data-hiding," in *Proc. 14th Int. Conf. DSP*, 2002, pp. 71–76.

[16] D. Maas, T. Kalker, and F. M. Willems, "A code construction for recursive reversible data-hiding," in *Proc. Multimedia Security Workshop ACM Multimedia*, Juan-les-Pins, France, Dec. 6, 2002.

[17] W. Zhang, B. Chen, and N. Yu, "Capacity-approaching codes for reversible data hiding," in *Proc 13th IH*, 2011, vol. 6958, LNCS, pp. 255–269, Springer-Verlag.

[18] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[19] K. Sayood, *Introduction to Data Compression*. San Francisco, CA: Morgan Kaufmann, 1996, pp. 87–94.

[20] R. Crandall, Some Notes on Steganography 1998 [Online]. Available: http://os.inf.tu-dresden.de/westfeld/crandall.pdf, Posted on steganography mailing list

[21] Miscellaneous gray level images [Online]. Available: http://decsai.ugr.es/cvg/dbimagenes/g512.php