Signal Processing 92 (2012) 2888–2900

Contents lists available at SciVerse ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Hierarchical reversible data hiding based on statistical information: Preventing embedding unbalance

Kehao Wang [a,b,*], Quan Liu [a], Lin Chen [b]

[a] School of Information Engineering, Wuhan University of Technology, 430070 Hubei, China
[b] Laboratoire de Recherche en Informatique (LRI), University of Paris-Sud XI, 91405 Orsay, France

ARTICLE INFO

Article history:
Received 21 September 2011
Received in revised form
6 April 2012
Accepted 16 May 2012
Available online 27 May 2012

Keywords:
Hierarchical embedding
Embedding unbalance
Statistical information
Reversible data hiding
Difference expansion

ABSTRACT

For reversible data hiding, the histogram-based difference expansion (DE) is one family of the generic methods where secret bits are embedded into an image by DE on prediction error. However, the ratio of the count of bit 0 to that of bit 1, embedded into the pixels with the same prediction error, will vary with the image and the secret bits, which leads to the embedding unbalance and image distortion. Therefore, a novel hierarchical embedding scheme is proposed to remove the unbalance level by level. We only implement it in two levels considering the complexity of constructing reversible transform for the hierarchical embedding. In the first level, the histogram is divided into different groups according to prediction error, and then two group exchange operations are designed to remove the embedding unbalance depending on the statistical information of this level. For further removing the unbalance, two adjacent groups are combined to a composite group in the second level, and another two composite group exchange operations are constructed to eliminate the unbalance by using the statistical information of the second level. Simulations demonstrate that the proposed hierarchical algorithm can achieve better performance in comparison with other existing histogram-based DE methods.

1. Introduction

Reversible data hiding technique is used to hide data in a host signal and to get back the embedded data and losslessly recover the original image as well. This property makes it very useful in the areas where image quality is strictly required, i.e. cover authentication, content integrity verification, covert communication, and image/video coding.

Recently, a number of reversible data hiding methods have been reported. Tian [1] proposed the concept of difference expansion (DE) but his embedding capacity is at best 0.5 bit per pixel. In order to improve the embedding capacity, some variants or extensions based on DE have been proposed. In [2], a new extension of DE-based scheme was proposed to achieve higher embedding capacity by using consecutive, overlapping pairs, instead of the non-overlapping pairs or triads. Ju et al. [3] proposed a block-based reversible data embedding method where the image is first divided into two areas and then the different DE methods are adopted in different areas. Kim et al. [4] utilized a new expandability with a simplified location map to achieve much higher embedding capacity. Hu et al. [5] proposed the histogram-based DE scheme for reversible data hiding in two directions. Hu et al. [6] proposed a DE-based reversible data hiding method in which an efficient payload-dependent overflow location map was constructed depending on stricter constraints. In [7], the author proposed a reversible

* Corresponding author. Tel./fax: +86 27 87665520.
E-mail addresses: kehao.wang@whut.edu.cn (K. Wang), quanliu@whut.edu.cn (Q. Liu), lin.chen@lri.fr (L. Chen).

0165-1684/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sigpro.2012.05.020

watermarking using interpolation technique. In [8], the author presented an adaptive reversible data scheme based on the prediction of DE such that the location map is no more required and the embedding capacity can be adjusted depending on the practical applications. In [9], the author divided the medical image into smooth region and non-smooth region, and then utilized different DE methods for different regions to increase the hiding capacity. In [10], the author proposed a novel cluster-based DE scheme for lossless data hiding. In [11], the author integrated low overhead for decoding with the histogram scheme and high embedding capacity with the DE scheme, and took the characteristics of original image, that is, the smooth and active regions, into account to add the flexibility for reversible data hiding. In [12], the author presented a block-based lossless data hiding schema to utilize the similarity between neighborhood pixels in the block to improve the marked-image quality. In [13], the author adopted a two-dimensional DE technique in order to increase the hiding capacity by minimizing the size of embedding map while keeping a good visual quality of the watermarked image. In [14], a multi-resolution DE extension was proposed to increase the overall embedding capacity.

Among the existing work, a representative proposition is that developed in [6], where the prediction error histogram is first divided into inner/embedding region and outer/shifting regions, and then an efficient payload-dependent overflow location map is constructed to achieve an excellent performance. However, the authors do not consider the embedding unbalance resulting from the specific embedding method as well as the content of the secret message. Basically, every method using this type of prediction error histogram faces the same problem.

In this paper, we firstly point out the root of the embedding unbalance in those histogram-based DE methods, and then design hierarchical embedding algorithm to remove the unbalance level by level. We would like to emphasize that the multi-level removing unbalance just reveals the excellent characteristics of wavelet since the DE method is one special wavelet transform in essence. According to the prediction error, the prediction error histogram is divided into different groups in the first level from the hierarchical perspective, and then two group exchange operations are designed to remove the embedding unbalance of each group depending on the statistical information of the group, where the two group exchange operations are realized by two pairs of reversible integer transforms.

For further elimination of the embedding unbalance, two adjacent groups are combined to form a composite group in the second level, and meanwhile, another two composite exchange operations are constructed to remove the embedding unbalance in each composite group. Theoretically, we can carry out the hierarchical embedding level by level until the unbalance is completely removed. However, we do not extend the hierarchical embedding algorithm to the higher levels considering the complexity of constructing reversible integer transform based on DE and the increase of the auxiliary data used to discriminate different operations in different levels. On the other hand, two pairs of optimal integer transforms are delicately designed and integrated into the hierarchical embedding method such that 1 bit data can be embedded in each overflow or underflow pixel without increasing additional data. Finally, two realizations of the hierarchical embedding, corresponding to the first level and the second level, are proposed to remove the embedding unbalance and to increase the embedding capacity as well.

The rest of the paper is organized as follows. First, the embedding unbalance is pointed out in Section 2, and then we propose hierarchical operations—group exchange and composite group exchange—in Section 3. In Section 4, the hierarchical embedding algorithm is realized in the first level and the second level to form group exchange (GX) algorithm and composite group exchange (CGX) algorithm, respectively. Collecting statistical information, embedding and extracting procedures are given in Section 5. GX and CGX are compared with other current methods in Section 6. Finally, we draw the conclusion in Section 7.

2. Embedding unbalance in histogram-based difference expansion

This section briefly reviews the principle of the histogram-based DE using the typical algorithm [6] as an example, and then points out the root of embedding unbalance.

For an 8-bit grayscale image, the reversible data embedding and extracting of prediction-error DE can be described as

$$\tilde{x} = f(\hat{x},(x,b)) \tag{1}$$

$$(x,b) = f^{-1}(\hat{x},\tilde{x}) \tag{2}$$

where $x \in [0,255]$ is the original pixel value, $\hat{x} \in [0,255]$ is the predicted value by a specific predicting method, $\tilde{x} \in [0,255]$ represents the embedded value, $b \in [0,1]$ is the embedded bit, $f(\cdot)$ and $f^{-1}(\cdot)$ constitute a pair of reversible integer transforms.

As proposed in [6], the prediction error histogram was divided into two parts: the inner region for embedding and the outer region for shifting. Assume that two thresholds, T_r and T_l, are used to control the right and left boundaries of the inner region, respectively. Then, in the inner region $[-T_l,T_r-1]$, an embedded value, $\tilde{x}$, is embedded by DE embedding as

$$\tilde{x} = x+d+b \quad \text{if } -T_l \le d < T_r \tag{3}$$

where d represents the difference value, and $\hat{x}$ represents the predicted value,

$$d = x-\hat{x} \tag{4}$$

$$\hat{x} = \begin{cases} \max(u,w) & \text{if } v \le \min(u,w) \\ \min(u,w) & \text{if } v \ge \max(u,w) \\ u+w-v & \text{otherwise} \end{cases} \tag{5}$$

where u, w, and v represent the right, lower, diagonal neighbors of x in the host image, respectively.

In the extracting, for the pixels $\tilde{x}$ in the inner embedded region $[-2T_l,2T_r-1]$, the original pixel x and the embedded bit b are restored as

$$x=\left\lfloor\frac{\tilde{x}+\widehat{x}}{2}\right\rfloor \quad \text{if } -2T_l\le d'<2T_r \tag{6}$$

$$b=LSB(\tilde{x}-\widehat{x}) \quad \text{if } -2T_l\le d'<2T_r \tag{7}$$

where d' represents the expanded difference as follows:

$$d'=x-\widehat{x} \tag{8}$$

For those pixels located in the outer regions, the method of shifting and restoring can be found in [6].

Now, I will point out the root of the embedding unbalance resulting from the DE method (3). For the ease of presentation, we assume that:

(1) Two secret bit streams $\mathbf{P}=\{\mathbf{P}_{-T_l}\cdots\mathbf{P}_d\cdots\mathbf{P}_{T_r-1}\}$ and $\mathbf{Q}=\{\mathbf{Q}_{-T_l}\cdots\mathbf{Q}_d\cdots\mathbf{Q}_{T_r-1}\}$, where $\mathbf{P}_d=p_1p_2\cdots p_n$ and $\mathbf{Q}_d=q_1q_2\cdots q_n$ ($p_i,q_i=0$ or 1) are going to be embedded into the pixels satisfying $x-\widehat{x}=d$ in the inner region through the embedding method (3).

(2) The count of 1 is larger than that of 0 in $\mathbf{P}_d$ while the count of 1 is smaller than that of 0 in $\mathbf{Q}_d$.

According to the DE embedding (3), in the positive inner region $d\in[0,T_r]$, the embedding distortion $|\tilde{x}-x|=d$ if $b=0$ while $|\tilde{x}-x|=d+1$ if $b=1$, demonstrating that the distortion of embedding one bit 1 is larger than that of embedding one bit 0. Since the count of 1 in $\mathbf{P}_d$ is larger than that of $\mathbf{Q}_d$, the embedding distortion resulting from embedding $\mathbf{P}_d$ is larger than that from embedding $\mathbf{Q}_d$ in the positive inner region. On the contrary, in the negative inner region $d\in[-T_l,-1]$, the distortion of embedding one bit 1 is smaller than that of embedding one bit 0. Hence the embedding distortion resulting from embedding $\mathbf{Q}_d$ will be larger than that from embedding $\mathbf{P}_d$. The above two cases reveal that the embedding method performs better for $\mathbf{P}_d$ in the negative inner region than in the positive inner region while the conclusion is contrary for $\mathbf{Q}_d$. Therefore, we have:

(1) The DE embedding (3) leads to the embedding unbalance since it cannot perform symmetrically well between the positive and negative inner regions.

(2) The DE embedding (3) achieves the embedding balance only when the ratio of the count of 1 to that of 0 in $\mathbf{P}_d$ is 1. Obviously, it cannot be guaranteed even though the ratio of the input secret bits $\mathbf{P}$ is 1.

(3) In order to reduce the unbalance, we need to adopt adaptive embedding methods according to the ratio of the count of 1 to that of 0 in $\mathbf{P}_d$, which is referred to as *statistical information* in the rest of paper.

We would like to emphasize that even though the ratio of count of bit 1 to that of bit 0 in $\mathbf{P}$ equals 1 in the best case (e.g. approximates 1 when $\mathbf{P}$ is compressed before embedding), the ratio in $\mathbf{P}_d$ is highly impossible to be 1 since $\mathbf{P}_d$ (or d) is tightly related with the host image and the predicting method (e.g. (5)) except its direct connection with $\mathbf{P}$. Thus, the embedding unbalance always does exist in embedding $\mathbf{P}_d$, and furthermore, does exist in embedding $\mathbf{P}$. Therefore, how to eliminate the embedding unbalance is the main motivation for our work in this paper.

3. Hierarchical exchange operations

In this section, we will introduce two kinds of operations, group exchange in the first level and composite group exchange in the second level, and then give their applicable conditions by theoretical analysis, respectively.

3.1. Group exchange in the first level

For the ease of presentation, we first give some definitions and then state the main theorem for group exchange in the first level.

Definition 1. *Embeddable Set* E and *Embedded Set* $\tilde{E}$: E is the set of the original pixels which satisfy (1) and (2) in a host image and $\tilde{E}$ is the set of the embedded pixels corresponding to the pixels in E.

Given the secret message, after scanning the host image, E is fixed and meanwhile, it is determined that bit 0 or 1 is going to be embedded in each pixel of E, that is to say, *statistical information* is fixed, which would be explained in the following part. Let $x\leftarrow b$ represent the binary value b would be embedded in the pixel x, i.e., $x\leftarrow 0$ represents one bit 0 would be embedded in the pixel x.

Definition 2. Embeddable Group G_d is the subset of E, and each pixel in G_d satisfies the difference between the original pixel value and the predicted value equals d,

$$G_d=\{x|x-\widehat{x}=d,x\in E\}$$

Embeddable bit b Set x_b^d is a subset of G_d, and each pixel in x_b^d will be embedded into the binary value 'b',

$$x_b^d=\{x|x\leftarrow b,x\in G_d\}$$

Given the secret bitstream message, it is obvious to obtain $G_d=x_0^d\cup x_1^d$.

Definition 3. Embedded Group $\tilde{G}_d$ is the set corresponding to Embeddable Group G_d, in which each embedded pixel is obtained from its original pixel by the embedding function $f(\cdot)$,

$$\tilde{G}_d=\{\tilde{x}|\tilde{x}=f(\widehat{x},(x,b)),x\in G_d\}$$

Embedded bit b set $\tilde{x}_b^d$ is the set of the corresponding embedded pixels under $f(\cdot)$,

$$\tilde{x}_b^d=\{\tilde{x}|\tilde{x}=f(\widehat{x},(x,b)),x\in x_d^b\}$$

Given the secret bitstream message, we have $\tilde{G}_d=\tilde{x}_0^d\bigcup\tilde{x}_1^d$.

Definition 4. We say $f(\cdot)$ and $f^{-1}(\cdot)$ are a pair of forward group exchange embedding (FGX) and extracting (FGX^{-1}) if (9) holds for any $x\in x_b^d$ and $\tilde{x}=f(\widehat{x},(x,b))$,

$$\tilde{x}-x-d=\begin{cases}b & \text{if } d\ge 0\\ 1-b & \text{if } d<0\end{cases} \tag{9}$$

$f(\cdot)$ and $f^{-1}(\cdot)$ are a pair of backward group exchange embedding (BGX) and extracting (BGX^{-1}) if (10) holds for

any $x \in x_b^d$ and $\tilde{x} = f(\hat{x},(x,b))$,

$$\tilde{x}-x-d = \begin{cases} 1-b & \text{if } d \geq 0 \\ b & \text{if } d < 0 \end{cases} \tag{10}$$

According to the definition, we design the pairs of FGX vs. FGX^{-1} and BGX vs. BGX^{-1} without proof as follows:

[FGX vs. FGX^{-1}]: Eqs. (11)–(13) constitute a pair of *FGX* and FGX^{-1}:

$$\tilde{x} = \begin{cases} x+d+b & \text{if } 0 \leq d \leq T_r-1 \\ x+d+1-b & \text{if } -T_l \leq d < 0 \end{cases} \tag{11}$$

$$b = \begin{cases} LSB(d') & \text{if } 0 \leq d' \leq 2T_r-1 \\ 1-LSB(|d'|) & \text{if } -2T_l \leq d' < 0 \end{cases} \tag{12}$$

$$x = \hat{x} + \left\lfloor \frac{d'}{2} \right\rfloor, \quad -2T_l \leq d' \leq 2T_r-1 \tag{13}$$

[BGX vs. BGX^{-1}]: Eqs. (14)–(16) constitute a pair of *BGX* and BGX^{-1}.

$$\tilde{x} = \begin{cases} x+d+1-b & \text{if } 0 \leq d \leq T_r-1 \\ x+d+b & \text{if } -T_l \leq d < 0 \end{cases} \tag{14}$$

$$b = \begin{cases} 1-LSB(|d'|) & \text{if } 0 \leq d' \leq 2T_r-1 \\ LSB(d') & \text{if } -2T_l \leq d' < 0 \end{cases} \tag{15}$$

$$x = \hat{x} + \left\lfloor \frac{d'}{2} \right\rfloor, \quad -2T_l \leq d' \leq 2T_r-1 \tag{16}$$

Obviously, (11)–(13) and (14)-(16) constitute a pair of symmetrical operations from the mathematical formulation.

Now, we give the definition of the *statistical* variable in the first level, corresponding to group exchange, as follows:

Definition 5. Group balance factor $\alpha(d)$ represents the ratio of cardinality of two sets belonging to G_d as follows:

$$\alpha(d) = \frac{|x_1^d|}{|x_0^d|} \tag{17}$$

where $|\bullet|$ stands for the cardinality of the set $\bullet$.

Definition 6. Group distortion deviation (*GDD*) is defined as

$$GDD_f(\tilde{G}_d,G_d) = \sum_{\tilde{x}\in\tilde{G}_d, x\in G_d} |\tilde{x}-x|^2$$

The following theorem states the optimal condition of choosing *FGX* or *BGX* in the embedding procedure.

Theorem 1. *For any G_d, if $\alpha(d) < 1$, GDD resulting from FGX is less than that resulting from BGX; if $\alpha(d) > 1$, GDD resulting from BGX is less than that resulting from FGX; if $\alpha(d) = 1$, GDD resulting from FGX equals that from BGX.*

Proof. (1) We first consider the case $d \geq 0$. According to the definition of FGX, we have

$$GDD_{FGX}(\tilde{G}_d,G_d) = |x_0^d| \times d^2 + |x_1^d| \times (d+1)^2$$

For BGX, we have

$$GDD_{BGX}(\tilde{G}_d,G_d) = |x_0^d| \times (d+1)^2 + |x_1^d| \times d^2$$

Thus,

$$GDD_{FGX}(\tilde{G}_d,G_d) - GDD_{BGX}(\tilde{G}_d,G_d) = (|x_1^d| - |x_0^d|)(2d+1) = \begin{cases} >0 & \text{if } \alpha(d) > 1 \\ =0 & \text{if } \alpha(d) = 1 \\ <0 & \text{if } \alpha(d) < 1 \end{cases}$$

(2) if $d \leq -1$, we have

$$GDD_{FGX}(\tilde{G}_d,G_d) - GDD_{BGX}(\tilde{G}_d,G_d) = (|x_0^d| - |x_1^d|)(2d+1) = \begin{cases} >0 & \text{if } \alpha(d) > 1 \\ =0 & \text{if } \alpha(d) = 1 \\ <0 & \text{if } \alpha(d) < 1 \end{cases}$$

Combining the two cases, we conclude the theorem. □

Lemma 1. *For any G_d, if $\alpha(d) \leq 1$, FGX and FGX^{-1} (11)–(13) should be used to embed secret data in G_d, to extract secret data and to recover the original pixel from the corresponding $\tilde{G}_d$; otherwise, BGX and BGX^{-1} (14)–(16) should be used.*

Proof. It is obvious according to Theorem 1. □

3.2. Composite group exchange in the second level

In the previous part, the embedding unbalance of the first level is eliminated by FGX or BGX depending on statistical information $\alpha(d)$ of each group G_d, whereas the embedding unbalance between groups (the second level) is ignored. Therefore, in this part, we will introduce two operations to furthermore reduce the embedding unbalance between the adjacent groups, and moreover, two pairs of integer transform are delicately designed to achieve this purpose.

Definition 7. Embeddable composite group C_d is composed of two embeddable groups G_d and G_{d+1} as follows:

$$C_d = \{(G_d,G_{d+1}) | d = x-\hat{x} = 2k, -127 \leq k \leq 127\}$$

Embedded composite group $\tilde{C}_d$ is composed of two embedded group $\tilde{G}_d$ and $\tilde{G}_{d+1}$ as follows:

$$\tilde{C}_d = \{(\tilde{G}_d,\tilde{G}_{d+1}) | d = x-\hat{x} = 2k, -127 \leq k \leq 127\}$$

Definition 8. Assuming two embedded groups $\tilde{G}_{2k} = (\tilde{x}_0^{2k},\tilde{x}_1^{2k})$ and $\tilde{G}_{2k+1} = (\tilde{x}_0^{2k+1},\tilde{x}_1^{2k+1})$, we say Ξ and Θ are Group concatenation and Group exchange, if Ξ and Θ satisfy (18) and (19), respectively.

$$(\tilde{x}_0^{2k},\tilde{x}_1^{2k})\Xi(\tilde{x}_0^{2k+1},\tilde{x}_1^{2k+1}) = (\tilde{x}_0^{2k},\tilde{x}_1^{2k},\tilde{x}_0^{2k+1},\tilde{x}_1^{2k+1}) \tag{18}$$

$$(\tilde{x}_0^{2k},\tilde{x}_1^{2k})\Theta(\tilde{x}_0^{2k+1},\tilde{x}_1^{2k+1}) = (\tilde{x}_0^{2k},\tilde{x}_{-1}^{2k+1},\tilde{x}_2^{2k},\tilde{x}_1^{2k+1}) \tag{19}$$

where $|\tilde{x}_2^{2k}| = |\tilde{x}_1^{2k}|$ and $|\tilde{x}_{-1}^{2k+1}| = |\tilde{x}_0^{2k+1}|$.

Now, we define the *statistical* variable of the second level, corresponding to composite group exchange, as follows:

Definition 9. Composite group balance factor $\beta(k)$ is the ratio of cardinality of two sets located in different embeddable groups of an embeddable composite group

as follows:

$$\beta(k)=\begin{cases}\dfrac{\max\{|x_0^{2k+1}|,|x_1^{2k+1}|\}}{\min\{|x_0^{2k}|,|x_1^{2k}|\}} & \text{if } k\geq 0\\ \dfrac{\min\{|x_0^{2k}|,|x_1^{2k}|\}}{\max\{|x_0^{2k+1}|,|x_1^{2k+1}|\}} & \text{if } k<0\end{cases} \tag{20}$$

The optimal condition of choosing Θ or Ξ is stated in the theorem as follows:

Theorem 2. *Group exchange operation Θ can be used to reduce GDD if* (21) *holds*; *otherwise, group concatenation operation Ξ should be used*:

$$\begin{cases}\beta(k)>\dfrac{4k+3}{4k+1} & \text{if } k\geq 0\\ \beta(k)<\dfrac{4k+3}{4k+1} & \text{if } k\leq -1\end{cases} \tag{21}$$

Proof. (1) We first consider the case $k\geq 0$. So we have $|\tilde{x}_0^{2k+1}|=\max\{|x_0^{2k+1}|,|x_1^{2k+1}|\}$ and $|\tilde{x}_1^{2k}|=\min\{|x_0^{2k}|,|x_1^{2k}|\}$ according to Lemma 1:

$$\begin{aligned}GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_1^{2k}|\times(2k+1)^2+|\tilde{x}_0^{2k+1}|\\&\quad\times(2k+1)^2+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\end{aligned}$$

$$\begin{aligned}GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_2^{2k}|\times(2k+2)^2\\&\quad+|\tilde{x}_{-1}^{2k+1}|\times(2k)^2+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\\&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_1^{2k}|\times(2k+2)^2+|\tilde{x}_0^{2k+1}|\times(2k)^2\\&\quad+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\end{aligned}$$

Therefore,

$$\begin{aligned}&GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})-GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})\\&=|\tilde{x}_0^{2k+1}|\times(4k+1)-|\tilde{x}_1^{2k}|\times(4k+3)\\&=\max\{|x_0^{2k+1}|,|x_1^{2k+1}|\}\times(4k+1)-\min\{|x_0^{2k}|,|x_1^{2k}|\}(4k+3)\\&=\min\{|x_0^{2k}|,|x_1^{2k}|\}\times(4k+1)\times\left(\beta-\frac{4k+3}{4k+1}\right)\end{aligned}$$

Furthermore, we have

$$GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})-GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})=\begin{cases}>0 & \text{if } \beta>\dfrac{4k+3}{4k+1}\\ =0 & \text{if } \beta=\dfrac{4k+3}{4k+1}\\ <0 & \text{if } \beta<\dfrac{4k+3}{4k+1}\end{cases}$$

(2) For the case $k\leq -1$, we have $|\tilde{x}_0^{2k+1}|=\min\{|x_0^{2k+1}|,|x_1^{2k+1}|\}$ and $|\tilde{x}_1^{2k}|=\max\{|x_0^{2k}|,|x_1^{2k}|\}$ according to Lemma 1:

$$\begin{aligned}GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_1^{2k}|\times(2k+1)^2\\&\quad+|\tilde{x}_0^{2k+1}|\times(2k+1)^2+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\end{aligned}$$

$$\begin{aligned}GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_2^{2k}|\times(2k+2)^2\\&\quad+|\tilde{x}_{-1}^{2k+1}|\times(2k)^2+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\\&=|\tilde{x}_0^{2k}|\times(2k)^2+|\tilde{x}_1^{2k}|\times(2k+2)^2+|\tilde{x}_0^{2k+1}|\times(2k)^2\\&\quad+|\tilde{x}_1^{2k+1}|\times(2k+2)^2\end{aligned}$$

Therefore,

$$\begin{aligned}&GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})-GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})\\&=|\tilde{x}_0^{2k+1}|\times(4k+1)-|\tilde{x}_1^{2k}|\times(4k+3)\\&=\min\{|x_0^{2k+1}|,|x_1^{2k+1}|\}\times(4k+1)-\max\{|x_0^{2k}|,\\&\quad|x_1^{2k}|\}\times(4k+3)\\&=\max\{|x_0^{2k}|,|x_1^{2k}|\}\times(4k+1)\times\left(\beta-\frac{4k+3}{4k+1}\right)\end{aligned}$$

Furthermore, we have

$$GDD_{\Xi}(\tilde{G}_{2k},\tilde{G}_{2k+1})-GDD_{\Theta}(\tilde{G}_{2k},\tilde{G}_{2k+1})=\begin{cases}>0 & \text{if } \beta<\dfrac{4k+3}{4k+1}\\ =0 & \text{if } \beta=\dfrac{4k+3}{4k+1}\\ <0 & \text{if } \beta>\dfrac{4k+3}{4k+1}\end{cases}$$

Combining cases 1 and 2, we obtain the conclusion. □

To achieve better performance, we always assume that the priority of FGX (BGX) is higher than that of Ξ (Θ) because the former is used in each group while the latter is used between groups. From the technical perspective, this assumption reveals the hierarchical characteristics of the proposed embedding method where the groups belong to the first level while the composite groups belong to the second level.

In the embedding process, we firstly operate FGX (or BGX) in G_{2k} and G_{2k+1}, obtain $\tilde{G}_{2k}$ and $\tilde{G}_{2k+1}$ respectively. Secondly, we operate Ξ (or Θ) between $\tilde{G}_{2k}$ and $\tilde{G}_{2k+1}$, and obtain $\tilde{C}_{2k}$. Therefore, embedding one bit for each $x\in C_{2k}$ needs two steps. However, the two-step approach will destroy the reversibility, which is prohibited in the methodology of data hiding. Then an alternative way is to obtain the same embedded pixels by one-step reversible integer transform rather than the two-step approach. Obviously, the design of one-step reversible integer transform to achieve the purpose of two-step embedding is the key point for the realization of the hierarchical embedding in the second level.

To this end, for each $x\in G_{2k}$, it exists four cases: (1) first FGX, then Ξ; (2) first BGX, then Ξ; (3) first FGX, then Θ; (4) first BGX, then Θ.

In order to discriminate the four cases, two bits auxiliary data is needed for G_{2k}. For the same reason, another two bits data is needed for G_{2k+1}. However, for each C_{2k}, Ξ or Θ operated in $\tilde{G}_{2k}$ is the same as that in $\tilde{G}_{2k+1}$. Thus, 4 bits data can be reduced to 3 bits, denoted as $O_2O_1O_0$ as shown in Table 1. The least significant bit O_0 is used to indicate FGX or BGX operation in G_{2k}. The middle bit O_1 is used to indicate FGX or BGX operation in G_{2k+1}. The most significant bit O_2 represents Ξ or Θ between G_{2k} and G_{2k+1}.

For the cases 1 and 2, the embedding and extracting are easily implemented by *FGX* (FGX^{-1}) or *BGX* (BGX^{-1}) because group exchange Θ is not used in the two cases. For cases 3 and 4, we need to design two reversible integer transforms such that the embedding or extracting can be carried out in one step. Eqs. (22)–(27) are two pairs of reversible integer transforms delicately designed to

Table 1
Coding of operation.

Bit	0	1
O_0	FGX	BGX
O_1	FGX	BGX
O_2	Ξ	Θ

Table 2
Coding of embedding and extracting.

O_2O_1 or O_2O_0	Embedding, Extracting, Restoring
00	(11)–(13)
01	(14)–(16)
10	(22)–(24)
11	(25)–(27)

realize the cases 3 and 4, respectively. The proof of the reversibility for the two pairs of integer transforms is ignored in this paper:

$$\tilde{x}=\begin{cases} x+d+2b-LSB(d) & \text{if } 0\le d\le T_r-1 \\ x+d+2(1-b)-LSB(|d|) & \text{if } -T_l\le d<0 \end{cases} \tag{22}$$

$$b=\begin{cases} LSB(d')\oplus LSB\left(\dfrac{d'+LSB(d')}{2}\right) & \text{if } 0\le d'+LSB(d')\le 2T_r \\ 1-LSB(d')\oplus LSB\left(\dfrac{d'+LSB(|d'|)}{2}\right) & \text{if } -2T_l\le d'+LSB(|d'|)<0 \end{cases} \tag{23}$$

$$x=\hat{x}+\frac{d'+LSB(|d'|)}{2}-b \quad \text{if } -2T_l\le d'+LSB(|d'|)\le 2T_r \tag{24}$$

$$\tilde{x}=\begin{cases} x+d+2(1-b)-LSB(d) & \text{if } 0\le d\le T_r-1 \\ x+d+2b-LSB(|d|) & \text{if } -T_l\le d<0 \end{cases} \tag{25}$$

$$b=\begin{cases} 1-LSB(d')\oplus LSB\left(\dfrac{d'+LSB(d')}{2}\right) & \text{if } 0\le d'+LSB(d')\le 2T_r \\ LSB(|d'|)\oplus LSB\left(\dfrac{d'+LSB(|d'|)}{2}\right) & \text{if } -2T_l\le d'+LSB(|d'|)<0 \end{cases} \tag{26}$$

$$x=\hat{x}+\frac{d'+LSB(|d'|)}{2}-b \quad \text{if } -2T_l\le d'+LSB(|d'|)\le 2T_r \tag{27}$$

Obviously, (22)–(24) and (25)–(27) constitute another pair of symmetrical operations.

After obtaining the two pairs of integer transforms, we give the main theorem as follows:

Theorem 3. *For any pixel in C_k, the embedding and extracting rules are shown in Table 2.*

Proof. It is obvious according to Lemma 1 and Theorem 2. □

4. Hierarchical embedding algorithm

In the previous section, four pairs of reversible integer transforms are proposed to reduce the embedding unbalance in the first level as well as the second level. In this section, based on these four pairs of integer transforms, the hierarchical embedding algorithm will be implemented in the first level and the second level to form GX and CGX, respectively. In addition, another two pairs of optimal reversible integer transforms are designed and integrated into GX (CGX) such that one bit can be embedded in each overflow/underflow pixel without introducing any auxiliary data. As a sequence, GX (CGX) can improve the embedding capacity compared to those algorithms where the overflow/underflow pixels are simply discarded in the embedding process.

4.1. GX algorithm for the first level

Two thresholds T_r and T_l as in [6] are adopted in our paper, and the prediction error histogram is divided into two parts: the inner region and the outer region. Then we embed data in the pixels located in the inner region, and shift the pixels located in the outer region to evacuate place in order to avoid overlapping. Considering the embedding unbalance in the inner region, it is necessary to utilize an adaptive embedding method for every group G_d according to Lemma 1. In the receiving end, to exactly recover the original image and to extract secret data need to know which operation of FGX and BGX is used in a group. So one bit auxiliary data should be provided to indicate FGX or BGX for each group in the embedding process, and then all these bits constitute a group exchange table (*GET*) embedded into the host image as a part of auxiliary data. However, the introduction of *GET* will increase the size of auxiliary data and moreover, lead to the additional embedding distortion. In order to lessen the negative effect of overlong *GET* information, two thresholds T_{G_l} and T_{G_r} are utilized to control the maximum length of *GET*, herein, T_{G_l} and T_{G_r} are used to control the count of group in the negative inner region and the positive inner region, and meanwhile, T_{G_l} and T_{G_r} are even number and odd number, respectively. To the end, the inner region is further divided into two parts as shown in Fig. 1: group region $[-G_{min},G_{max}]$ and group outer region $[-T_l,-1-G_{min}]\cup[1+G_{max},T_r-1]$, where $G_{min}=\min(T_{G_l},T_l)$ and $G_{max}=\min(T_{G_r},T_r-1)$. In the following part, we will discuss the embedding and extracting schemes of all different regions.

4.1.1. Inner region

According to the constraints of (11) and (14), we know that the pixels in the inner/embedding region should satisfy the following constraint (28)

$$\begin{cases} x+d\le 254 & \text{if } 0\le d\le T_r-1 \\ x+d\ge 0 & \text{if } T_l\le d<0 \end{cases} \tag{28}$$

which are located in the group region and the group outer region.

Group outer region: For the pixels located in group outer region, we embed one bit data according to (29), get it back and restore the original pixel according to (30) and

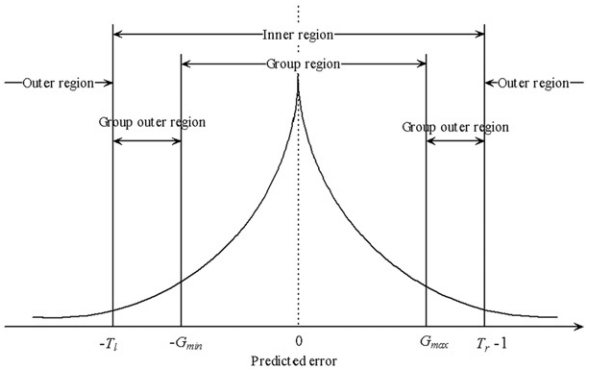

Fig. 1. Predicted error histogram division for GX.

(31), respectively:

$$\tilde{x}=\begin{cases} x+d+b & \text{if } 1+G_{max}\le d\le T_r-1 \\ x+d+b & \text{if } -T_l\le d\le -G_{min}-1 \end{cases} \tag{29}$$

$$b=\begin{cases} LSB(d') & \text{if } 2+2G_{max}\le d'\le 2T_r-1 \\ LSB(|d'|) & \text{if } -2T_l\le d'\le -2G_{min}-1 \end{cases} \tag{30}$$

$$x=\begin{cases} \hat{x}+\left\lfloor\frac{d'}{2}\right\rfloor & \text{if } 2+2G_{max}\le d'\le 2T_r-1 \\ \hat{x}+\left\lfloor\frac{d'}{2}\right\rfloor & \text{if } -2T_l\le d'\le -2G_{min}-1 \end{cases} \tag{31}$$

Group region: For the pixels located in the group region $[G_{min},G_{max}]$, FGX or BGX will be adopted depending on $\alpha(d)$. Therefore, $\alpha(d)$ should be calculated firstly through collecting the statistical information of every group, which will be discussed in Section 5. According to Lemma 1, for any pixel $x\in G_d$, $d\in[G_{min},G_{max}]$ and $\alpha(d)\le 1$, the embedding and extracting are implemented by FGX and FGX^{-1} according to (11)–(13). For $x\in G_d$, $d\in[G_{min},G_{max}]$ and $\alpha(d)>1$, the embedding and extracting are by BGX and BGX^{-1} according to (14)–(16).

4.1.2. Outer region

For the pixels in the outer shiftable region, which are determined by

$$\begin{cases} x+T_r\le 255 & \text{if } d\ge T_r \\ x-T_l\ge 0 & \text{if } d\le -T_l-1 \end{cases} \tag{32}$$

they are shifted right T_r units or left T_l units to avoid overlapping the embedded pixels in inner regions according to (33). In the extracting process, we shift them left T_r units or right T_l units to restore the original pixel by (34):

$$\tilde{x}=\begin{cases} x+T_r & \text{if } d\ge T_r \\ x-T_l & \text{if } d\le -T_l-1 \end{cases} \tag{33}$$

$$x=\begin{cases} \tilde{x}-T_r & \text{if } d'\ge 2T_r \\ \tilde{x}+T_l & \text{if } d'\le -2T_l-1 \end{cases} \tag{34}$$

4.1.3. Overflow and underflow region

The pixels, satisfying the condition (35) or (36), are regarded as overflow/underflow pixels, so a binary overflow location map matrix M is introduced to record their states as proposed in [6] where '1' indicates that the corresponding pixel remained unaltered while '0' indicates that the corresponding pixel has been shifted or embedded. However, in our paper, the bit '1' in M only indicates the corresponding pixel is specific and one bit data is still going to be embedded into the pixel through two pairs of delicately designed reversible integer transforms. During the embedding process, the M will be compressed into Mc embedded into the host image as a part of auxiliary data.

Inner overflow and underflow region: The pixels located in the inner overflow and underflow are determined by

$$\begin{cases} x+d\ge 255 & \text{if } 0\le d\le T_r-1 \\ x+d\le -1 & \text{if } -T_l\le d<0 \end{cases} \tag{35}$$

Obviously, those pixels should locate in $[256-T_r,255]$ and $[0,T_l-1]$, respectively.

Outer overflow and underflow region: The pixels in the outer overflow and underflow should satisfy the following constraint:

$$\begin{cases} x+T_r\ge 256 & \text{if } d\ge T_r \\ x-T_l\le -1 & \text{if } d\le -T_l-1 \end{cases} \tag{36}$$

Then it is easy to obtain that those pixels are also located in $[256-T_r,255]$ and $[0,T_l-1]$, respectively.

For those pixels in inner or outer overflow regions, the embedding and extracting can be carried out according to

$$\tilde{x}=2x+b-256 \tag{37}$$

$$b=\tilde{x}-2\left\lfloor\frac{\tilde{x}}{2}\right\rfloor \tag{38}$$

$$x=\left\lfloor\frac{\tilde{x}+256}{2}\right\rfloor \tag{39}$$

We firstly discuss the designing process of the reversible integer transform for the overflow pixels. For the pixel $[256-T_r,255]$ located in the inner overflow region or in the outer overflow region, the difference value d ($d=x-256$) is always smaller than or equals -1. If one bit data b is embedded in the difference value d using DE, the expanded difference value d' ($d'=2d+b$) is still smaller than or equals -1. Thus, when 256 is added to d', we can obtain the embedded pixel $\tilde{x}\in[256-2T_r,255]$. At the same time, $|\tilde{x}-x|\le T_r+1$, where $|\tilde{x}-x|=T_r+1$ only when $x=256-T_r$. Considering the length, T_r, of the overflow region, if any DE method is used to embed one bit data, then the length of the corresponding overflow region will be expanded to $2T_r+1$. In order to lessen the embedding distortion, we adopt integer transform (37)–(39) to arrange the expanded range $(2T_r+1)$ to overlap the original range completely, but still have the residual length T_r+1, that is to say $|\tilde{x}-x|\le T_r+1$. If any other integer transform is adopted for the pixels in the overflow region, which means the expanded range does not overlap the original range completely, then the residual length will exceed T_r+1, in other words, $|\tilde{x}-x|>T_r+1$. Therefore we conclude that (37)–(39) are the optimal integer transform utilized not only to embed one bit data in a pixel but also to keep the minimum embedding distortion.

For the pixels in the inner or outer underflow region, the embedding and extracting can be optimally carried out according to (40)–(42):

$$\tilde{x} = 2x + b \tag{40}$$

$$b = \tilde{x} - 2\left\lfloor \frac{\tilde{x}}{2} \right\rfloor \tag{41}$$

$$x = \left\lfloor \frac{\tilde{x}}{2} \right\rfloor \tag{42}$$

For the pixels in the underflow region, the grayscale value of the pixel located in $[0, T_l - 1]$ is regarded as the difference value between x and 0. So x is used to embed a bit data b by using DE. The expanded difference value is regarded as the embedded pixel $\tilde{x}$ corresponding to the original pixel x. The distortion deviation $|\tilde{x} - x| \le T_l$ can be obtained easily. For the similar reason as discussed for the overflow region, (40)–(42) constitute a pair of optimal reversible transform for those pixels in the inner or outer underflow region.

So far, we have obtained the embedding/extracting (or shifting/restoring) methods for all the pixels of a host image, which naturally brings about the GX algorithm corresponding to the realization of the hierarchical embedding in the first level.

4.2. CGX algorithm for the second level

In this algorithm, the only difference from GX is that the group region is further divided into two parts: composite group region $[-CG_{min}, CG_{max}]$ and composite group outer region $[-G_{min}, -CG_{min} - 1] \cup [1 + CG_{max}, G_{max}]$ as shown in Fig. 2, where, $CG_{min} = G_{min} - LSB(G_{min})$ and $CG_{max} = G_{max} - 1 + LSB(G_{max})$. Moreover, in the composite group region, two adjacent groups construct a composite group in the second level[1] such that Θ (or Ξ) can be utilized to further eliminate the embedding unbalance.

In the embedding and extracting process, the pixels in the outer region and group outer region are coped in the same way with those corresponding to pixels in GX algorithm. Regarding the pixels located in the composite group outer region, their embedding method is the same as that of pixels located in the group region of GX algorithm. Therefore we only emphasize the embedding method for the pixels located in the composite group region. According to the overflow and underflow constraints of (11), (14), (22), (25), the expandable pixels in the composite group region should satisfy the condition

$$\begin{cases} x + d - LSB(d) \le 253 & \text{if } 0 \le d \le T_r - 1 \\ x + d - LSB(|d|) \ge 0 & \text{if } -T_l \le d < 0 \end{cases} \tag{43}$$

For these pixels, we further divide the composite group region $[-CG_{min}, CG_{max}]$ into $(CG_{min} + CG_{max} + 1)/2$ composite groups. In the embedding process, four pairs of reversible transforms will be utilized to embed and extract data according to Theorem 3. Hence, we need to know O_0, O_1 and O_2 for each composite group C_d, that is to say, $\alpha(2d)$, $\alpha(2d+1)$ and $\beta(2d)$ should be firstly calculated according to (17) and (20). Secondly, we adopt different embedding methods depending on O_2O_1 and O_2O_0, respectively, as shown in Table 2. In order to exactly restore the original image in the receiving end, it is necessary to embed $O_2O_1O_0$ into the host image as a part of auxiliary data. As discussed in GX algorithm, all O_1O_0 bits construct the *GET*. In order to keep compatibility with GX algorithm, the O_2 bit is extracted from $O_2O_1O_0$ of each composite group to construct a binary composite group exchange table (*CGET*) which is embedded into the host image as a part of auxiliary data.

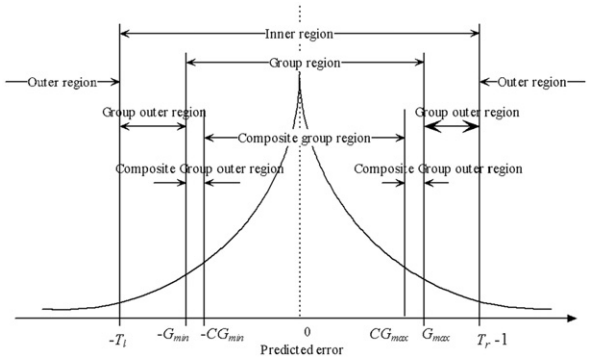

Fig. 2. Predicted error histogram division for CGX.

For the pixels which satisfy (35), (36) and (44) are regarded as overflow/underflow pixels, we assign '1' in the corresponding position of the location map matrix, and adopt the same embedding and extracting schemes as those in GX algorithm for these pixels:

$$\begin{cases} x + d - LSB(d) \ge 254 & \text{if } 0 \le d \le T_r - 1 \\ x + d - LSB(|d|) \le -1 & \text{if } -T_l \le d < 0 \end{cases} \tag{44}$$

Obviously, all those aforementioned embedding/extracting (or shifting/restoring) constructs, CGX, the realization of the hierarchical embedding in the second level.

4.3. Structure of auxiliary data

The hidden data (denoted as W) is composed as follows:

$$W = P + A$$

where P represents payload data, and A represents the auxiliary data which consists of a header file H and the compressed location map Mc. The structure of the header file H is composed as follows:

$$\begin{aligned} H = {} & T_r(8 \text{ bits}) + T_l(8 \text{ bits}) + ep(32 \text{ bits}) \\ & + GET(G_{min} + G_{max} \text{ bits}) \\ & + CGET\left(\frac{CG_{min} + CG_{max} + 1}{2} \text{ bits}\right) \end{aligned}$$

where ep is the 32 bits data which is formed by 16 bits vertical coordinate and 16 bits horizontal coordinate of the last pixel which is necessary to be modified.

Then we have $A = H + Mc$ and embed $P + A$ into the host image. In the embedding process, if we use the same way to

[1] This kind of composition just reveals the hierarchical characteristics of CG and CGX, which is an excellent characteristics of wavelet since difference expansion is the special wavelet transform in essence.

embed the $P+A$, we will encounter a problem for blind data extraction. If not provided with the embedding information, the decoder cannot extract data from the embedded image. So we must transmit A to the decoder through the available channel. One way of transmitting A is to save the bits of A in a place of the embedded image that is easy to locate. We preserve the bits of A into the LSBs of the first $Len(A)$ image pixels by LSB replacement, no matter whether these image pixels have been embedded or not, herein, $Len(A)$ represents the length of A. The starting pixel is still the upper left corner of the image. While performing LSB replacement operation, the replaced LSBs are collected in a temporary data package. Such a data package is then saved in the area originally allotted for the storage of A. As a result, we exchange the storage place of these two data packages. In the practical implementation, this storage place exchange operation is carried out during the process of data embedding [6].

In GX and CGX algorithms, we always assume that $T_{G_l}=10$, and $T_{G_r}=9$. Therefore, the length of GET, $G_{min}+G_{max}+1$, does not exceed 20 bits, and that of $CGET$, $(CG_{min}+CG_{max}+1)/2$, does not exceed 10 bits as well.

5. Collecting statistical information, embedding and extracting

In this section, we will give the procedure of collecting the statistical information of the two levels as well as the embedding/extracting procedures.

5.1. Collecting statistical information

- *Step*1: Assign $EC=0$, $T_r=1$, $T_l=0$, $T_{G_l}=10$, $T_{G_r}=9$, $N_0[i]=0$, and $N_1[i]=0$ $(T_{G_l}\le i\le T_{G_r})$.
- *Step*2: Initialize the binary location map M with zero and scan the image in raster manner from the left upper corner. For each x in the original image, calculate the difference value d according to (4). If x locates in the overflow or underflow region, we assign '1' to the corresponding bit in M. If x does not locate in the outer region, then increase the embeddable count EC by one. Note the last row and column are not used in the scanning process for convenience. In the rest of the paper, the scanning process complies with this constraint.
- *Step*3: Get the binary location map matrix M and compress it to Mc and record its length $Len(Mc)$.
- *Step*4: If EC is smaller than the length of W, we increase T_r or T_l by one in an interleaving manner until $T_r=64$ or $T_l=63$. Then go to step 2.
- *Step*5 (Get T_r and T_l): If EC is larger than or equals to the length of W, then we record T_r and T_l, calculate the length of GET, initialize the GET to '0', and construct the temporary W.
- *Step*6 (Get ep): Assign $EC=0$, and rescan the image in raster manner from the left upper corner. For each x in the original image, calculate the difference value d. If x does not locate in the outer region, then increase the embeddable count EC by one. If EC equals the length of W, then we get ep by combining the horizontal and vertical coordinates of x and go to step 7.
- *Step*7 (Collecting Statistical Information): Assign $EC=0$, and rescan the image in raster manner from the left upper corner to the last point indicated by ep. For each pixel x, calculate the difference value d. If x locates in the group outer region or overflow or underflow region, we increase the embeddable count EC by one. If x locates in the group region and $W[EC]=0$, increase $N_0[d]$ and EC by one. If x locates in the group region and $W[EC]=1$, increase $N_1[d]$ and EC by one. If x locates in other regions, we neglect it.
- *Step*8 (Get $\alpha(d)$): Get information of group: $N_0[j]$ and $N_1[j]$ $(G_{min}\le j\le G_{max})$, then $\alpha(d)=(N_1[G_{min}]/N_0[G_{min}])$ $(-G_{min}\le d\le G_{max})$.
- *Step*9: Construct the GET. For $-G_{min}\le d\le G_{max}$, if $\alpha(d)\ge 1$, then $GET[G_{min}+d]=1$; otherwise $GET[G_{min}+d]=0$.
- *Step*10 (Get $\beta(d)$): Get statistical information of each composite group,

$$\beta(d)=\begin{cases}\dfrac{\max\{N_0[2d+1],N_1[2d+1]\}}{\min\{N_0[2d],N_1[2d]\}} & \text{if } 0\le d\le\dfrac{CG_{max}-1}{2}\\ \dfrac{\min\{N_0[2d],N_1[2d]\}}{\max\{N_0[2d+1],N_1[2d+1]\}} & \text{if } -\dfrac{CG_{min}}{2}\le d<0\end{cases}$$

- *Step*11: Construct the $CGET$. For $-CG_{min}/2\le d\le (CG_{max}-1)/2$, if $\beta(d)$ satisfies (21), then $CGET[CG_{min}/2+d]=1$; otherwise $CGET[CG_{min}/2+d]=0$.
- *Step*12: Construct the complete W.

5.2. Embedding

The embedding procedure is as follows:

Algorithm 1. Embedding.

1: Initialize $T_{G_l}=10$, and $T_{G_r}=9$;
2: From the pixel in the left upper corner of the original image to the last pixel indicated by ep, we calculate d by formulas (4). If x belongs to:
   - overflow region, (37) is used to embed one bit data;
   - underflow region, (40) is used to embed one bit data;
   - outer region, shift x according to (33);
   - group outer region, (29) is used to embed one bit data;
   - composite group outer region, (11) or (14) is used to embed one bit data depending on $\alpha(d)$;
   - composite group region, (22) or (25) is used to embed one bit data depending on $\alpha(d)$ and $\beta(\lfloor d\rfloor)$ (only in CGX);

   Each time a pixel is scanned, we exchange the LSB of the pixel with one bit of A in sequence until the end of A;
3: The embedded image is obtained.

5.3. Extracting

The extracting procedure is as follows:

Algorithm 2. Extracting.

1: Get back the auxiliary information A from the LSBs of the first $Len(A)$ image pixels beginning from the left upper corner of the embedded image;
2: Extract T_r, T_l, ep, GET, $CGET$, and Mc, and decompress Mc to M;
3: Scan the embedded image from the pixel indicated by ep to the left upper corner in an inverse raster manner. If the corresponding bit in the location map equals '1',
   - $\tilde{x}\ge 128$, extracting and restoring according to (38) and (39);
   - $\tilde{x}<128$, extracting and restoring according to (41) and (42).

   If the corresponding bit in the location map equals '0', calculate $d'=\tilde{x}-\hat{x}$. If $\tilde{x}$ locates in

- outer region, shift $\tilde{x}$ according to (34);
- group outer region, (30) and (31) are used;
- composite group outer region, (12), (13) or (15), (16)are used depending on $GET[G_{min}+\lfloor d'/2 \rfloor]$;
- composite group region, (23), (24) or (26), (27), are used depending on $GET[G_{min}+\left\lfloor \frac{d'}{2} \right\rfloor]$ and $CGET[\frac{CG_{min}}{2}+\left\lfloor \frac{d'}{4} \right\rfloor]$ (only in CGX);

4: The hidden data and the original image are obtained.

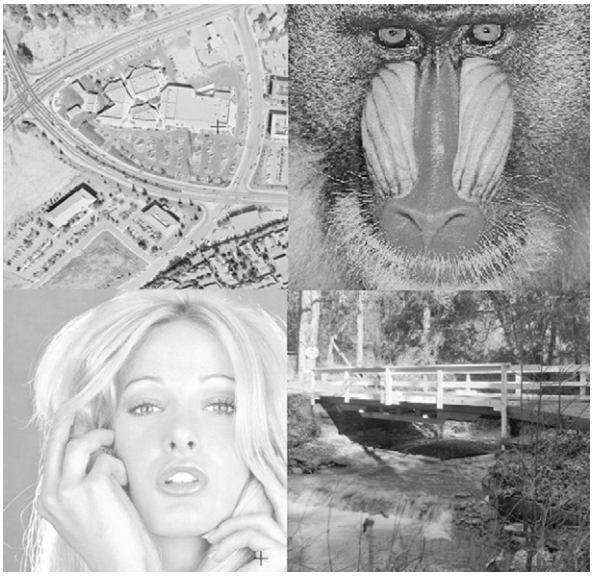

Fig. 3. From left top to right bottom: Aerial, Baboon, Girl, and stream and bridge.

6. Performance evaluation

In this section, we evaluate the performance of the proposed solutions and compare them with existing relevant propositions in the literature. Specifically, we implement three algorithms proposed in [6,15,11], respectively, and term them as Hu-DE, P2 and C-DE respectively for the ease of presentation. We compare the performance of our proposed GX and CGX algorithms w.r.t. the above reference algorithms on four 512×512 images that are 'Aerial', 'Baboon', 'Girl', and 'Stream and bridge' as shown in Fig. 3. Let γ be the ratio of bit '1' to the size of payload. First, we consider the setting where the embedding rate increases from 0 bit per pixel (bpp) to 1 bpp while the ratio of the count of '0' to '1' in payload equals 1, that is to say, $\gamma=0.5$. Second, We will consider the cases $\gamma=0.6,0.8,0.95$. In this experiment, we give the embedding capacity (bpp) versus auxiliary size curve and the embedding capacity versus image quality (dB) curve.

Aerospace image 'Aerial' has large flat regions, some of which are bright. As shown in Fig. 4(a), an obvious result is that P2 compressed auxiliary information has a constant length in despite of embedding rate. Different from P2, the

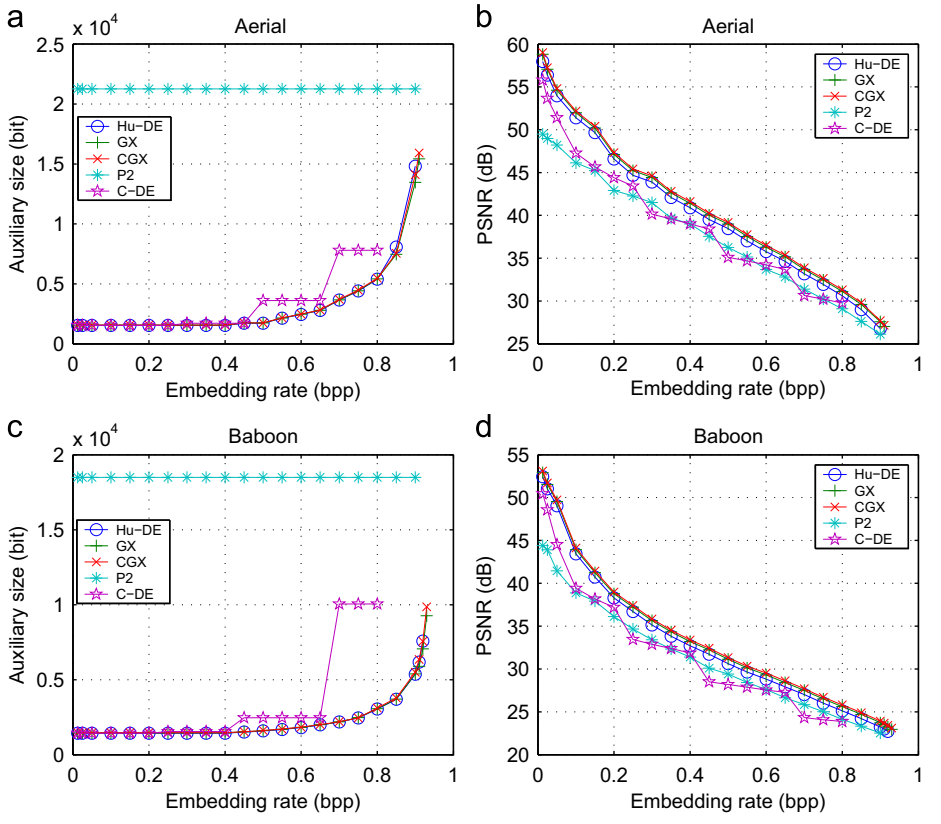

Fig. 4. Performance comparison on test images: Aerial and Baboon ($\gamma=0.5$). We give the embedding capacity versus auxiliary size curve in the left column and the embedding capacity versus image quality curve in the right column.

compressed overflow location maps of GX, CGX and Hu-DE have variable length for different payloads, especially dramatically increasing when the embedding rate is larger than 0.8 bpp. This result indicates that most pixels in the image will enter overflow region after 0.8 bpp. The two curves of our algorithms GX and CGX coincide with that of Hu-DE since at most 20 bits or 30 bits auxiliary data are consumed compared with Hu-DE (Section 4.3). The curve of C-DE is located between the curves of P2 and ours' because C-DE combines the DE embedding and histogram-based embedding. Specifically, from 0.4 bpp, the auxiliary size of C-DE is larger than that of our algorithms and Hu-DE. This is because C-DE only uses DE scheme basically when the embedding rate is less than 0.4 bpp. Fig. 4(b) shows that the performance of our algorithms GX and CGX is best among five curves. Compared with Hu-DE, GX and CGX approximately have 0.5 bpp improvement. One reason is that each underflow or overflow pixel is used to accommodate one bit data in GX and CGX while discarded as useless pixel in Hu-DE. Another reason is that GX and CGX adopt two kinds of symmetrical operations to eliminate the embedding unbalance as well as image distortion. The curves of P2 and C-DE are located below our curves because our payload-dependent overflow location map is highly compression-efficient.

For texture images like 'Baboon', the histogram is flat. Fig. 4(c) shows that the length of our compressed overflow location map increases apparently only when the embedding rate is greater than 0.9 bpp. This is because most of the expandable predicted errors are used for large payloads. Fig. 4(d) exhibits the advantage of our methods over Hu-DE method. From 0.0125 bpp to 0.1 bpp, the curves of GX and CGX almost coincide with that of Hu-DE, but after 0.1 bpp our algorithms GX and CGX have about 0.5 bpp improvement compared with Hu-DE. When the embedding rate is less than 0.1 bpp, our proposed mechanism—embedding in each overflow pixel and group exchange—does not work well because there are not enough overflow pixels and statistical information for each group.

For high-tone images like 'Girl', the histogram is sharp and narrow. Fig. 5(a) shows that CGX has a larger compressed overflow location map compared to GX and Hu-DE. This is because the CGX' overflow constraint (44) is much stricter than that of GX, and then for the high-tone image, CGX leads to the larger size of overflow location map. As for the C-DE, the curve has a jumping point at 0.6 bpp because the histogram-based embedding dominates DE embedding in this high-tone image. Fig. 5(b) shows that our method is superior to others. The performance difference between our methods and P2 dwindles as the payload increases while the performance difference between GX (CGX) and Hu-DE increases as the payload. The latter is because the mechanism—embedding in each overflow pixel and group exchange—works well as the increase of embedding rate.

There are some images with partial textures and uneven brightness, for example, 'Stream and bridge'. The

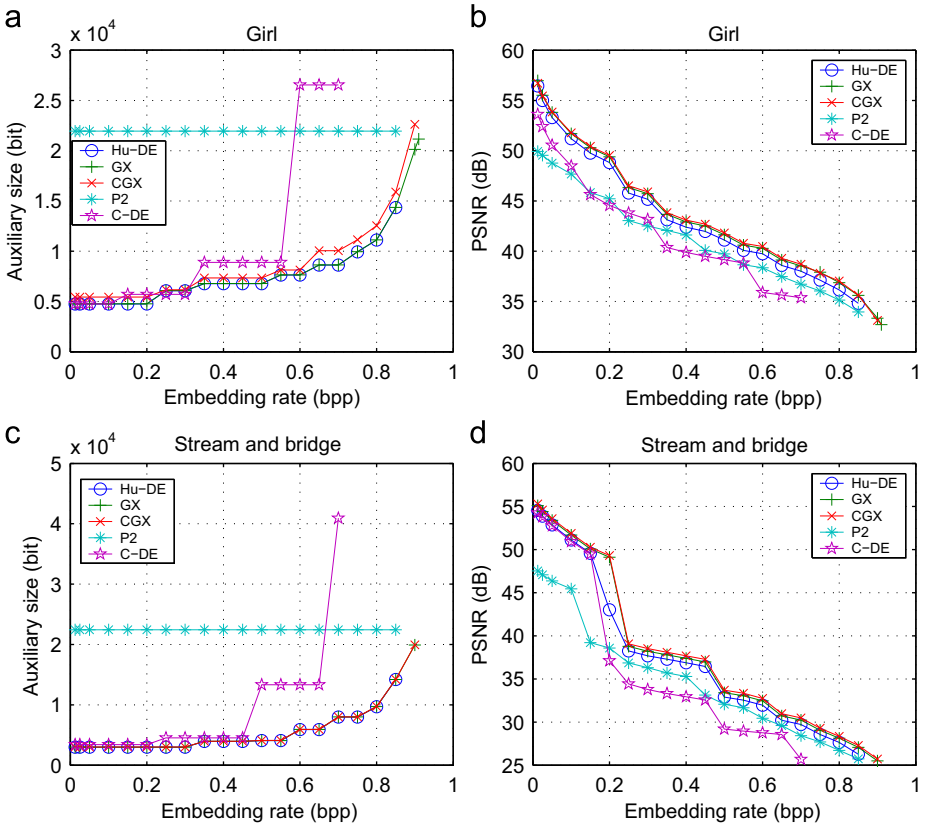

Fig. 5. Performance comparison on test images: Girl, and Stream and bridge ($\gamma = 0.5$). We give the embedding capacity versus auxiliary size curve in the left column and the embedding capacity versus image quality curve in the right column.

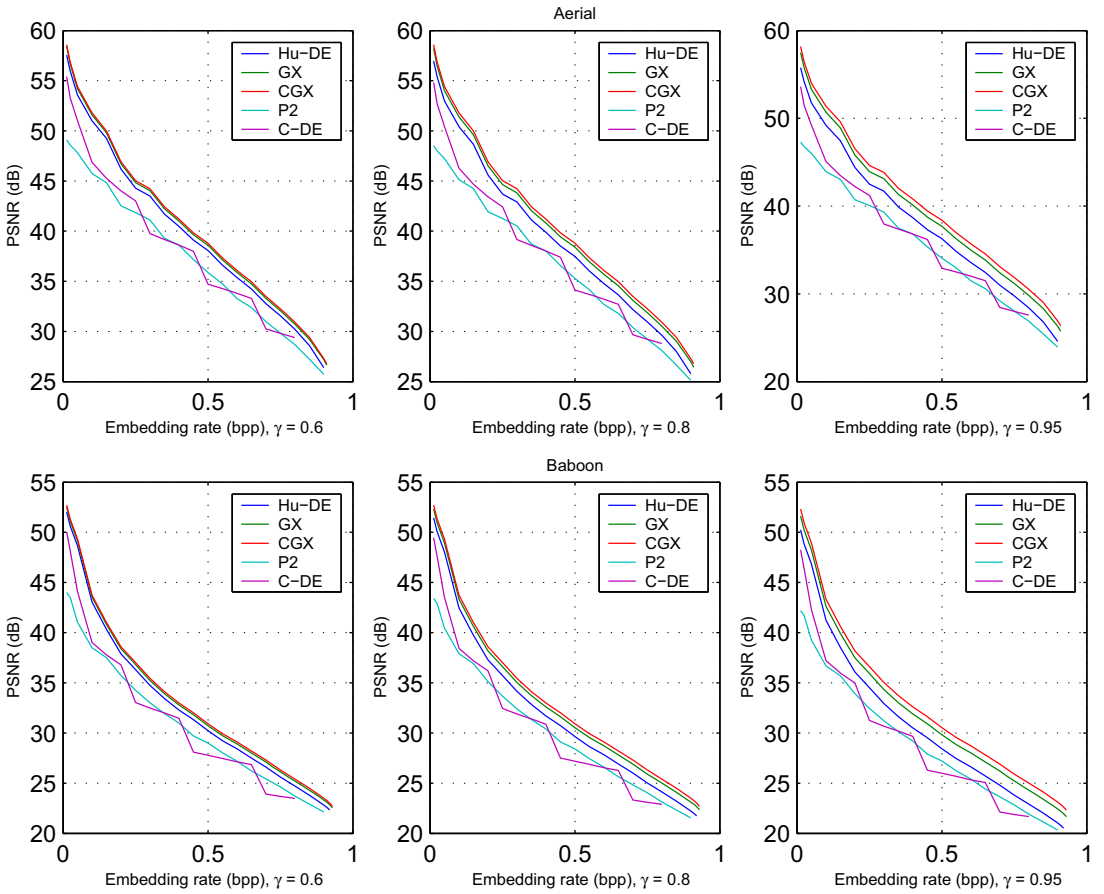

Fig. 6. Performance comparison on test images: Aerial and Baboon. We give the embedding capacity versus image quality curve for $\gamma = 0.6,0.8,0.95$.

image is very bright in some regions but very dark in others. The histogram is not sharp but very ragged. Fig. 5(c) shows that the length of our compressed overflow location map increases slowly. Fig. 5(d) demonstrates that our method outperforms the other three, and the performance difference is largest when the embedding rate approximates 0.2 bpp. Moreover, our algorithms, GX and CGX, can still embed data but Hu-DE cannot when the embedding rate is larger than 0.85 bpp.

For $\gamma = 0.6,0.8,0.95$, we only plot the embedding rate versus image quality curve in Figs. 6 and 7 because the embedding rate versus auxiliary size curve is the same with Figs. 4(a), (c) and 5(a), (c). As shown in Figs. 6 and 7, the performance difference between CGX (GX) and others increases with γ. This result shows that CGX (GX) outperforms Hu-DE, P2 and C-DE as the increase of γ. This is because the group exchange and composite group exchange perform better as the increase of embedding unbalance.

From the results obtained for the four test images in the case of $\gamma = 0.5,0.6,0.8,0.95$, we conclude that our proposed hierarchical methods perform better than Hu-DE, P2 and C-DE, in reducing the embedding distortion no matter what ratio of the count of '1' to that of '0' in payload. This is because adaptive reversible transforms are utilized to embed data depending on the statistical information and meanwhile, those pixels in the overflow and underflow regions are also utilized to embedded data. On the other hand, compared with GX, the better performance of CGX reveals the fact that the embedding distortion from the embedding unbalance can be removed as the increase of level.

7. Conclusion

In this paper, we firstly analyze the embedding unbalance, and then point out the embedding unbalance will lead to poor performance when the ratio of '1' to '0' in payload does not equal '1'. Next, we introduce the hierarchical embedding operation based on the prediction error and design FGX (FGX^{-1}) and BGX (BGX^{-1}) to remove embedding unbalance in each group of the first level. In order to further eliminate the embedding unbalance between the adjacent groups, we combine two adjacent groups to form a composite group in the second level, and then design two operations Ξ and Θ to remove the embedding unbalance in each composite group of the second level. On the other hand, we propose two pairs of integer transforms for those overflow and underflow pixels such that 1 bit data can be embedded in each overflow or underflow pixel. Finally, based on FGX, BGX, Ξ and Θ, two hierarchical algorithms are proposed to remove the embedding unbalance and to increase the embedding capacity.

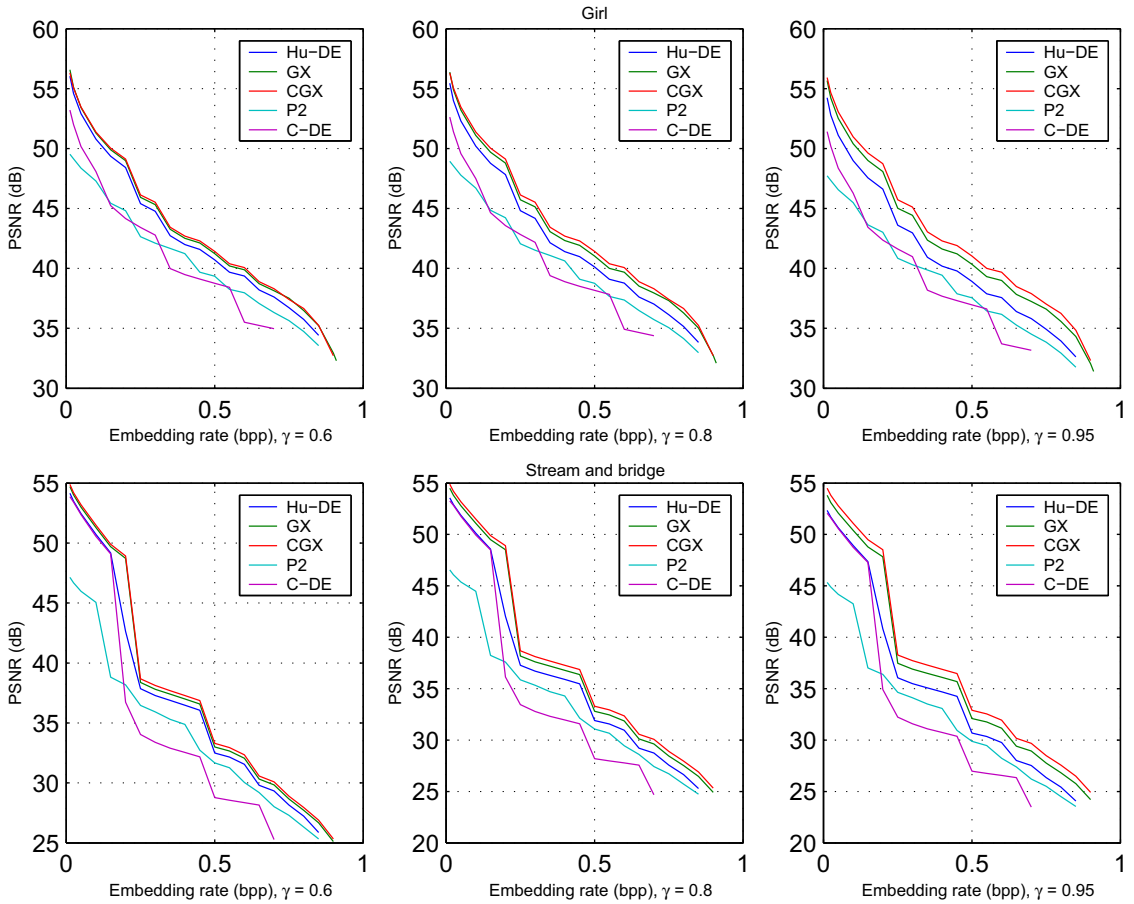

Fig. 7. Performance comparison on test images: Girl, and Stream and bridge. We give the embedding capacity versus image quality curve for $\gamma = 0.6, 0.8, 0.95$.

Acknowledgements

The authors would like to thank the editor and the anonymous referee for their valuable comments and suggestions that improved the clarity and quality of this manuscript. This work was supported by the National Natural Science Foundation of China (no. 51175389).

References

[1] J. Tian, Reversible data embedding using a difference expansion, IEEE Transactions on Circuits and Systems for Video Technology 13 (8) (2003) 890–896.
[2] E. Chrysochos, E.E. Varsaki, V. Fotopoulos, A.N. Skodras, High capacity reversible data hiding using overlapping difference expansion, in: Proceedings of the 10th Image Analysis for Multimedia Interactive Services, 2009, pp. 121–124.
[3] J.-Y. Hsiao, K.-F. Chan, J.M. Chang, Block-based reversible data embedding, Signal Processing 89 (4) (2009) 556–569.
[4] H.J. Kim, V. Sachnev, Y.Q. Shi, N. Jeho, H.-G. Choo, A novel difference expansion transform for reversible data embedding, IEEE Transactions on Information Forensics and Security 3 (3) (2008) 456–465.
[5] Y. Hu, H.-K. Lee, K. Chen, J. Li, Difference expansion based reversible data hiding using two embedding directions, IEEE Transactions on Multimedia 10 (8) (2008) 1500–1512.
[6] Y. Hu, H.-K. Lee, J. Li, De-based reversible data hiding with improved overflow location map, IEEE Transactions on Circuits and Systems for Video Technology 19 (2) (2009) 250–260.
[7] L. XinLuo, Z. Chen, M. Chen, X. Zeng, Z. Xiong, Reversible image watermarking using interpolation technique, IEEE Transactions on Information Forensics and Security 5 (1) (2010) 187–193.
[8] C.-F. Leea, H.-L. Chenb, H.-K. Tso, Embedding capacity raising in reversible data hiding based on prediction of difference expansion, Journal of Systems and Software 83 (10) (2010) 1864–1872.
[9] O.M. Al-Qershi, B.E. Khoo, High capacity data hiding schemes for medical images based on difference expansion, Journal of Systems and Software 84 (1) (2011) 105–112.
[10] Y.-Y. Tsai, C.-S. Chan, A novel cluster-based difference expansion transform for lossless data hiding, in: 2011 Fifth International Conference on Genetic and Evolutionary Computing (ICGEC), 2011, pp. 172–175.
[11] H.-C. Huang, Y.-H. Chen, Y.-Y. Lu, Histogram-based difference expansion for reversible data hiding with content statistics, in: 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2011, pp. 37–40.
[12] M. Khodaei, K. Faez, Reversible data hiding by using modified difference expansion, in: 2010 Second International Conference on Signal Processing Systems (ICSPS), 2010, pp. 31–34.
[13] O.M. Al-Qershi, B.E. Khoo, Reversible watermarking scheme based on two-dimensional difference expansion, in: 2010 Second International Conference on Computer Research and Development, 2010, pp. 228–232.
[14] M. Azzoni, G. Boato, M. Carli, K. Egiazarian, Reversible watermarking using prediction and difference expansion, in: 2010 Second European Workshop Visual Information Processing (EUVIP), 2010, pp. 71–76.
[15] D.M. Thodi, J.J. Rodriguez, Expansion embedding techniques for reversible watermarking, IEEE Transactions on Image Processing 16 (3) (2006) 721–730.