Journal of Network Intelligence ©2016 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **1**, Number **1**, February **2016**

Invariability of Remainder Based Reversible Watermarking

Shao-Wei Weng

School of Information Engineering
Guangdong University of Technology
wswweiwei@126.com

Jeng-Shyang Pan

Innovative Information Industrial Research Center
Shenzhen Graduate School
Harbin Institute of Technology
jengshyangpan@gmail.com

Jie-Hang Deng

School of Computers
Guangdong University of Technology

Received February 2015; revised December 2015

Abstract. *A novel and simple integer transform is proposed in this letter. This transform can also be considered a prediction process, in which the mean value of a n-sized block is used to predict each pixel in this block so that two bits are embedded into each of $(n-1)$ pixels. By employing the invariability of the reminder produced by the mean value divided by a given embedding parameter, we can achieve reversibility. The embedding distortion can be greatly controlled by embedding $2(n-1)$ bits into blocks with strong intra-block correlation while keeping the others unaltered. Experimental results reveal the proposed algorithm is effective.*

Keywords: Invariability of remainder, Reversible watermarking.

1. Introduction.

Reversible watermarking based on difference expansion was proposed by Tian [1]. The differences between two pixels were expanded to carry watermark information if neither overflow nor underflow occurred. Alattar [2] generalized the DE technique by taking a set containing multiple pixels rather than a pair. In Thodi's work [3], histogram shifting was incorporated into Tian's method to produce a new algorithm called Alg. D2 with a overflow map. Weng *et al.* proposed an integer transform based on invariability of the sum of pixel pairs [4]. In Wang *et al.*'s method [5], an generalized integer transform and a payload-dependent location map were constructed to extend the DE technique to the pixel blocks of arbitrary length.

When the embedding rule of Alattar's method is reformulated as an integer transform, this transform can be deemed to contain an additional term and a prediction process which uses the mean value of a block to predict each pixel in this block. This additional term is necessarily a guarantee of invariability of the mean value before and after embedding. However, its existence will result in great decrease of PSNR (peak signal to noise ratio) value in Alattar's method. This is also the reason that the performance of Alattar's method is not capable of exceeding that of Wang *et al.*'s.

To solve this problem above, a novel and simple integer transform that can remove the redundant term is proposed in this letter. Therefore, this integer transform can be considered a pure prediction process, in which the mean value of a n-sized block is used to predict each pixel in this block so that two bits are embedded into each of $(n-1)$ pixels. By employing the invariability of the reminder of the mean value divided by a given embedding parameter, we can achieve reversibility. Since the proposed integer transform can embed $2(n-1)$ (rather than (n-1)) bits into a n-sized block which introduces less distortion, we can obtain a better performance than Wang *et al.* at lager embedding rates.

2. **The related methods.** The integer transform defined in Eq. (3) of the paper [5] (proposed by Wang *et al.*) is listed in Eq. (1).

$$\begin{aligned} y_1 &= 2x_1 - a(\mathbf{x}) \\ y_2 &= 2x_2 - 2f(a(\mathbf{x})) + w_1 \\ &= 2x_2 - (a(\mathbf{x}) + LSB(a(\mathbf{x}))) + w_1 \\ \cdots & \\ y_n &= 2x_n - 2f(a(\mathbf{x})) + w_{n-1} \\ &= 2x_n - (a(\mathbf{x}) + LSB(a(\mathbf{x}))) + w_{n-1} \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{Z}^n$ and $\mathbf{y} = (y_1, \cdots, y_n) \in \mathbb{Z}^n$ respectively represent a n-sized pixel array and its corresponding watermarked one, $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $a(\mathbf{x}) = \begin{cases} \lfloor \bar{\mathbf{x}} \rfloor & \text{if } \bar{\mathbf{x}} - \lfloor \bar{\mathbf{x}} \rfloor < 0.5 \\ \lceil \bar{\mathbf{x}} \rceil & \text{otherwise} \end{cases}$, $f(\mathbf{x}) = \lceil \frac{\mathbf{x}}{2} \rceil$, and w_i $(i \in \{0, 1, \cdots, n-1\})$ denotes 1-bit watermark and $w_i \in \{0, 1\}$, and $LSB(\cdot)$ represents the least significant bit (LSB). $a(\mathbf{x})$ is actually the rounded value of $\bar{\mathbf{x}}$.

Eq. (1) is rearranged as follows

$$\begin{aligned} y_1 &= a(\mathbf{x}) + 2(x_1 - a(\mathbf{x})) \\ y_2 &= a(\mathbf{x}) + 2(x_2 - a(\mathbf{x})) + w_1 - LSB(a(\mathbf{x})) \\ \cdots & \\ y_n &= a(\mathbf{x}) + 2(x_n - a(\mathbf{x})) + w_{n-1} - LSB(a(\mathbf{x})) \end{aligned} \tag{2}$$

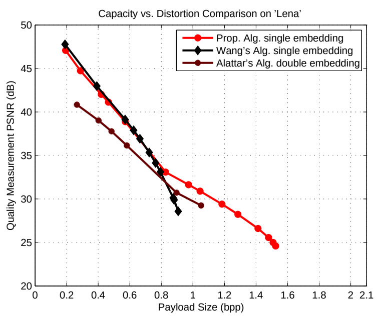
The integer transform defined in the paper [2] (proposed by Alattar) can be summarized in Eq. (3).

$$\begin{aligned} y_1 &= \lfloor \bar{\mathbf{x}} \rfloor - \left\lfloor \frac{2(x_2 - x_1) + w_1 + \cdots + 2(x_n - x_1) + w_{n-1}}{n} \right\rfloor \\ &= \lfloor \bar{\mathbf{x}} \rfloor - \left\lfloor \frac{2\sum\limits_{i=1}^{n} x_i + \sum\limits_{i=1}^{n-1} w_i - 2nx_1}{n} \right\rfloor \\ y_2 &= y_1 + 2\left(x_2 - x_1\right) + w_1 \\ \cdots & \\ y_n &= y_1 + 2\left(x_n - x_1\right) + w_{n-1} \end{aligned} \tag{3}$$

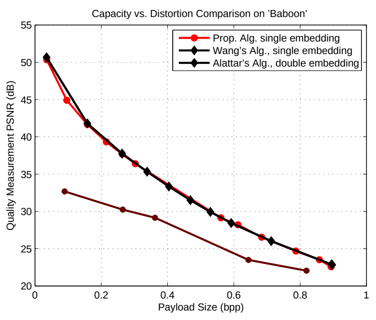
Suppose $k_2 = \sum_{i=1}^{n} x_i - n\lfloor \bar{\mathbf{x}} \rfloor$, and then $k_2 \in \{0, \cdots, n-1\}$. Substitute $\sum_{i=1}^{n} x_i$ into Eq. (3), we have

$$\begin{aligned} y_1 &= \lfloor \bar{\mathbf{x}} \rfloor + 2(x_1 - \lfloor \bar{\mathbf{x}} \rfloor) - \lfloor \tfrac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor \\ y_2 &= \lfloor \bar{\mathbf{x}} \rfloor + 2(x_2 - \lfloor \bar{\mathbf{x}} \rfloor) + w_1 - \lfloor \tfrac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor \\ \cdots & \\ y_n &= \lfloor \bar{\mathbf{x}} \rfloor + 2(x_n - \lfloor \bar{\mathbf{x}} \rfloor) + w_{n-1} - \lfloor \tfrac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor \end{aligned} \tag{4}$$

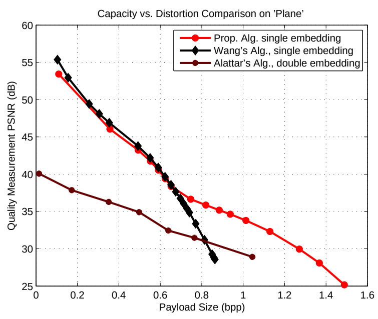
Comparing Eq. (2) with Eq. (4), we see that the integer transform proposed by Alattar has its own disadvantage: it introduces the higher distortion than the transform proposed



(a) Lena



(b) Baboon



(c) Plane

FIGURE 1. Performance comparison of our method with the other two methods.

by Wang *et al.*. Alattar's method can be deemed to contain an additional term and a process which uses the mean value of a block to predict each pixel in the block. Specially, for each y_i $(i \in \{0, 1, \cdots, n\})$ in Eq. (4), it has an term, i.e., $\lfloor \frac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor$. Since $k_2 \in \{0, 1, \cdots, n-1\}$ and $w_i \in \{0, 1\}$, $0 \leq \lfloor \frac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor \leq \lfloor \frac{3(n-1)}{n} \rfloor = \lfloor 3 - \frac{1}{n} \rfloor = 2$, i.e., $\lfloor \frac{2k_2 + \sum_{i=1}^{n-1} w_i}{n} \rfloor \in \{0, 1, 2\}$. For instance, when k_2 reaches its maximum value (i.e., $n-1$), if all to-be-embedded bits are set to 1, then this term will reach its maximum value (namely 2). Similarly, Wang *et al.*'s method contains a term, i.e., $LSB(a(\mathbf{x})) \in \{0, 1\}$ except the process which uses $a(\mathbf{x})$ to predict each pixel in the block.

Therefore, relative to this term in Wang *et al.*'s method, the existence of the one in Alattar's method will result in greater reduction of PSNR value. Abundant experiments also demonstrate that Wang *et al.*'s method has superior performance to Alattar's.

For the purpose of increasing algorithm performance, we propose a novel and simple integer transform that can remove the redundant term. Therefore, this integer transform can be considered a pure prediction process, in which the mean value of a n-sized block is used to predict each pixel in this block so that two bits are embedded into each of $(n-1)$ pixels. By employing the invariability of the reminder of the mean value divided by a given embedding parameter, we can achieve reversibility. Since the proposed integer transform can embed $2(n-1)$ (rather than (n-1)) bits into a n-sized block which introduces less distortion, we can achieve a better performance than Wang *et al.* at lager embedding rates.

3. The proposed method.

3.1. An integer transform. In the proposed method, a grayscale image is partitioned into non-overlapping $m \times m$-sized sub-blocks, where $n = m \times m$. An integer transform exploiting invariability of the remainder produced by dividing the mean value by a given embedding parameter is proposed below

$$\begin{aligned} y_1 &= \lfloor \bar{\mathbf{x}} \rfloor + 4(x_1 - \lfloor \bar{\mathbf{x}} \rfloor) + e_1 \\ &= 4x_1 - 3\lfloor \bar{\mathbf{x}} \rfloor + e_1 \\ &\cdots \\ y_{n-1} &= \lfloor \bar{\mathbf{x}} \rfloor + 4(x_{n-1} - \lfloor \bar{\mathbf{x}} \rfloor) + e_{n-1} \\ &= 4x_{n-1} - 3\lfloor \bar{\mathbf{x}} \rfloor + e_{n-1} \\ y_n &= \lfloor \bar{\mathbf{x}} \rfloor + 4(x_n - \lfloor \bar{\mathbf{x}} \rfloor) \\ &= 4x_n - 3\lfloor \bar{\mathbf{x}} \rfloor \end{aligned} \tag{5}$$

where e_i represents 2-bit watermark for each $i \in \{1, \cdots, n-1\}$, namely $e_i \in \{0, 1, 2, 3\}$, the number 4 indicates the given embedding parameter. This integer transform can also be considered as a prediction process, in which the mean value is used to predict each pixel in a n-sized block so that two bits are embedded into each of $(n-1)$ pixels (see Eq. (5)). In this transform, $\text{mod}(\lfloor \bar{\mathbf{x}} \rfloor, 4)$ remains unaltered before and after embedding, where the modulo function $\text{mod}(a, b)$ returns the remainder of a divided by b, which is expressed by the following formula: $\text{mod}(a, b) = a - b\lfloor \frac{a}{b} \rfloor$. This invariability is the single most important element to retrieve the original pixel values.

Next, we will introduce in detail how to correctly extract watermark bits. Firstly, subtracting y_n from y_i for each $i \in \{1, \cdots, n-1\}$ is to obtain the difference values as follows

$$y_i - y_n = 4(x_i - x_n) + e_i \tag{6}$$

Since e_i is a positive integer smaller than or equal to 3, i.e., $e_i \in \{0, 1, 2, 3\}$, and meanwhile, $4(x_i - x_n)$ is a multiple of 4, the remainder will remain invariant when e_i and $4(x_i - x_n) + e_i$ are divided by 4, respectively. That is to say, $\text{mod}(y_i - y_n, 4) = \text{mod}(e_i, 4) = e_i$. Consequently, watermark bits e_i can be correctly extracted. We subtract the correctly extracted bits e_i from $y_i^{'}$ for each $i \in \{1, \cdots, n-1\}$ so as to be capable of correctly retrieving the original block $\mathbf{x}$. The difference value $y_i^{'}$ between y_i and e_i is calculated via the following equation Eq.(7)

$$y_i^{'} = 4x_i - 3\lfloor \bar{\mathbf{x}} \rfloor \tag{7}$$

For the convenience of description, we use $\mathbf{y}^{'} = (y_1^{'}, \cdots, y_{n-1}^{'}, y_n)$ represents the pixel-value-array via Eq. (7). Suppose $\lfloor \bar{\mathbf{x}} \rfloor = 4k_1 + k_2$, where $k_1 \in \mathbb{R}$ and $k_2 \in \{0, 1, 2, 3\}$. Notice that, $\text{mod}(\lfloor \bar{\mathbf{x}} \rfloor, 4) = k_2$. Substitute $\lfloor \bar{\mathbf{x}} \rfloor$ into Eq. (7) and y_n, we have

$$\begin{aligned} y_i^{'} &= 4(x_i - 3k_1) - 3k_2 = 4x_i^{'} - 3k_2 \\ &\cdots \\ y_n &= 4(x_n - 3k_1) - 3k_2 = 4x_n^{'} - 3k_2 \end{aligned} \tag{8}$$

We use $x_i^{'}$ to replace $4(x_i - 3k_1)$ so as to further simplify Eq. (8). We will prove that $\text{mod}(4x_i^{'} - 3k_2, 4)$ is identical to k_2 when $x_i^{'}$ is set as some integer using the following equation

$$\begin{aligned} &\text{mod}(4x_i^{'} - 3k_2, 4) = \text{mod}(-3k_2, 4) \\ &= \text{mod}(4k_2 - 3k_2, 4) = \text{mod}(k_2, 4) = k_2 \end{aligned} \tag{9}$$

In a word, $\text{mod}(4x_i^{'} - 3k_2) = \text{mod}(\lfloor \bar{\mathbf{x}} \rfloor, 4)$. On the decoding side, after we can correctly get the remainder of $\lfloor \bar{\mathbf{x}} \rfloor$ divided by 4, i.e., k_2, each pixel value of $\mathbf{y}^{'}$ is subtracted by k_2 to get the difference value $y_i^{''}$ according to the following equation

$$y_i^{''} = y_i^{'} - k_2 \;\; = \;\; 4x_i^{'} - 4k_2 = 4x_i - 3\lfloor \bar{\mathbf{x}} \rfloor - k_2 \tag{10}$$

After both sides of Eq.(10) is divided by 4, $\frac{y_i^{''}}{4} = x_i - \frac{3\lfloor \bar{\mathbf{x}} \rfloor + k_2}{4}$ for each $i \in \{1, \cdots, n\}$. We have

$$\sum_{i=1}^{n} \frac{y_i^{''}}{4} = \sum_{i=1}^{n} x_i - n\frac{3\lfloor \bar{\mathbf{x}} \rfloor + k_2}{4} \tag{11}$$

It yields that $\lfloor \frac{1}{n} \sum_{i=1}^{n} \frac{y_i^{''}}{4} \rfloor = \lfloor \frac{1}{n} \sum_{i=1}^{n} x_i \rfloor - \frac{3\lfloor \bar{\mathbf{x}} \rfloor + k_2}{4} = \frac{\lfloor \bar{\mathbf{x}} \rfloor - k_2}{4} = k_1$. So

$$\lfloor \bar{\mathbf{x}} \rfloor = 4\lfloor \frac{1}{n} \sum_{i=1}^{n} \frac{y_i^{''}}{4} \rfloor + k_2 \tag{12}$$

Substitute $\lfloor \bar{\mathbf{x}} \rfloor$ via Eq. (12) into Eq. (5), we can correctly retrieve x_n. And then, $\lfloor \bar{\mathbf{x}} \rfloor$ is substituted into Eq. (7) so as to retrieve x_i for each $i \in \{0, 1, \cdots, n-1\}$.

3.2. **Data Embedding.** To prevent the overflow/underflow, each watermarked pixel value should be contained in $[0, 255]$. We define

$$D = \left\{ \begin{array}{l} \mathbf{x} \in A : 0 \le 4x_i - 3\lfloor \bar{\mathbf{x}} \rfloor \le 252 (1 \le i \le n-1), \\ 0 \le 4x_n - 3\lfloor \bar{\mathbf{x}} \rfloor \le 255 \end{array} \right\}$$

where $A = \{\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{Z} : 0 \le x_i \le 255\}$.

For a pixel-value array $\mathbf{x} \in A$, it is classified into one of two sets: $E_t = \{\mathbf{x} \in D : v(\mathbf{x}) \le T_h\}$ and $O_t = \{\mathbf{x} \in A - E_t : v(\mathbf{x}) > T_h\}$, where T_h is a given threshold, $v(\mathbf{x})$ represents the variation of a block, i.e., $v(\mathbf{x}) = \sqrt{\frac{1}{n}(x_i - \bar{\mathbf{x}})^2}$, E_t and O_t are used to denote the sets of pixels which are altered so as to carry $2(n-1)$ bits or kept unaltered, respectively.

A location map is generated in which the locations of the pixel arrays belonging to E_t are marked by '1' while the others are marked by '0'. The location map is compressed losslessly by an arithmetic encoder and the resulting bitstream is denoted by $\mathcal{L}$. L_S is the bit length of $\mathcal{L}$. For each $\mathbf{x} \in E_t$, it can carry $2(n-1)$ watermark bits, so the maximum hiding capacity is $C_{ap} = 2(n-1)\|E_t\| - L_S$ bits, where $\| \cdot \|$ represents the cardinality of a set. Namely, the size of the payload equals C_{ap}.

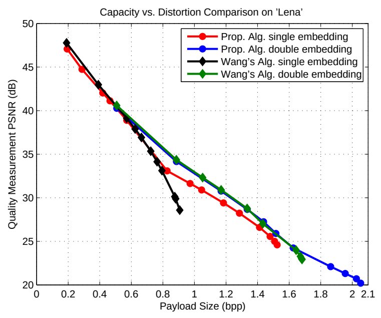
For each $\mathbf{x} \in O_t$, then it is kept unaltered, i.e., $\mathbf{y} = \mathbf{x}$. For each $\mathbf{x}$, if it belongs to E_t, then $2(n-1)$ bits are embedded into it according to Eq. (5).

After the first $n\lceil \frac{L_S}{n} \rceil$ pixel arrays have been processed (meaning either altered to carry $2(n-1)$ bits or kept unaltered), the LSBs of the first L_S pixels of their corresponding watermarked pixel arrays $\mathbf{y}$ are firstly appended to the payload $\mathcal{P}$, and then replaced by the compressed location map $\mathcal{L}$. After all the sub-blocks are processed, a new marked image I_w is obtained.

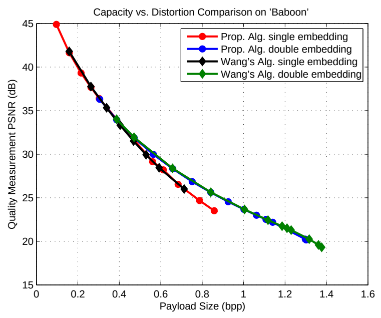
3.3. **Data Extraction and Image Restoration.** The LSBs of the pixels in I_w are collected into a bitstream $\mathcal{B}$ according to the same order as in embedding. $\mathcal{B}$ is decompressed by an arithmetic decoder to retrieve the location map.

For each watermarked pixel array $\mathbf{y} = (y_1, \cdots, y_n)$, if its location is associated with '0' in the location map, then it is ignored, i.e., $\mathbf{x} = \mathbf{y}$. Otherwise, the watermark can be extracted using the following formula: $e_i = \mathrm{mod}(y_i - y_n, 4)$ for each i $\in \{0, \cdots, n-1\}$. And meanwhile, the original mean value is retrieved by Eq. (12). Finally, the original image is recovered.

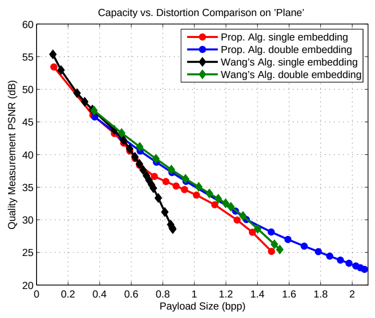
4. **Experimental results.** The capacity vs. distortion comparisons among the proposed method, Wang *et al.*'s, Alattar's are shown in Figs. 1 and 2. Three standard 512×512-sized grayscale images are used in our experiments: 'Lena', 'Baboon' and 'Plane'. Suppose the thresholds corresponding to single embedding and double embedding are T_1 and T_2,



(a) Lena



(b) Baboon



(c) Plane

FIGURE 2. Performance comparison of our method with Wang *et al.*s method, for single and double embedding.

respectively. Suppose also that T_1 has an initial value which is set to a value capable of ensuring the embedding rate larger than 0. Then, T_1 is gradually increased from this initial value until the given embedding rate is achieved. Once T_1 is determined, its corresponding PSNR value under the given embedding rate is obtained experimentally. These obtained numerical results on three test images are plotted in Fig. 1, respectively. For double embedding, T_2 is set to half of T_1. When T_1 is gradually increased from the initial value, T_2 is also accordingly raised. Similarly, once T_1 and T_2 is determined, each corresponding data pair containing a given embedding rate and its PSNR value is plotted in Fig. 2 on three test images, respectively.

From Fig. 1, it can be seen that the two curves corresponding to the proposed method and Wang *et al.*'s are very close when the embedding rate is low. When the embedding rate is increased, the proposed method outperforms Wang *et al.*'s for 'Lena' and 'Plane'. Moreover, the embedding capacity in Wang *et al.*'s method is upper bounded by $(1-\frac{1}{n})$ bpp for a single embedding process. However, we can get a bpp close to $2(1-\frac{1}{n})$ without multiple embedding. As illustrated in Fig. 1, we get a significant performance increase relative to Alattar's.

Since the correlations between the neighboring pixels in 'Baboon' not as high as in the others so that little improvement was made by the proposed method. So, the proposed method has almost the same performance as Wang *et al.*'s, while provide a better performance than Alatter's (see Figs 1 and 2).

Double embedding is the process of embedding data into an always embedded image. We also perform double embedding for three test images. As shown in 2, we can achieve the same performance as Wang *et al.*'s when the embedding rate is smaller than 1.6 bpp for 'Lena' or 1.4 bpp for 'Plane'. Our superiority becomes more obvious when the embedding rates gradually approach to 2.0 bpp, especially for 'Lena' and 'Plane'. When the threshold value is set to a large integer, and correspondingly, the number of pixel arrays used for embedding is greatly increases. Therefore, we can obtain higher embedding rates than Wang *et al.*'s. Since double embedding that embeds more data into blocks with high correlation will lead to lower distortion, double embedding may be better than single embedding.

5. **Conclusions.** A novel integer transform is proposed in this letter. This transform can also be considered a prediction process, in which the mean value of a n-sized block is used to predict each pixel in this block so that two bits are embedded into each of the $(n-1)$ pixels. Experimental results reveal the proposed algorithm is effective.

Acknowledgment. This work was supported in part by National NSF of China (No. 61201393, No. 61202267), New Star of Pearl River on Science and Technology of Guangzhou (No. 2014J2200085).

REFERENCES

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

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

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

[4] S. W. Weng, Y. Zhao, J. S. Pan, and R. R. Ni, "Reversible watermarking based on invariability and adjustment on pixel pairs," *IEEE Signal Process. Lett.*, vol. 45, no. 20, pp. 1022–1023, 2008.

[5] X. Wang, X. L. Li, B. Yang, and Z. M. Guo, "Efficient generalized integer transform for reversible watermarking," *IEEE Signal Process. Lett.*, vol. 17, no. 6, pp. 567–570, 2010.