A Contrast-Sensitive Reversible Visible Image Watermarking Technique

Ying Yang, Xingming Sun, *Senior Member, IEEE*, Hengfu Yang, Chang-Tsun Li, and Rong Xiao

***Abstract*— A reversible (also called lossless, distortion-free, or invertible) visible watermarking scheme is proposed to satisfy the applications, in which the visible watermark is expected to combat copyright piracy but can be removed to losslessly recover the original image. We transparently reveal the watermark image by overlapping it on a user-specified region of the host image through adaptively adjusting the pixel values beneath the watermark, depending on the human visual system-based scaling factors. In order to achieve reversibility, a reconstruction/recovery packet, which is utilized to restore the watermarked area, is reversibly inserted into non-visibly-watermarked region. The packet is established according to the difference image between the original image and its approximate version instead of its visibly watermarked version so as to alleviate its overhead. For the generation of the approximation, we develop a simple prediction technique that makes use of the unaltered neighboring pixels as auxiliary information. The recovery packet is uniquely encoded before hiding so that the original watermark pattern can be reconstructed based on the encoded packet. In this way, the image recovery process is carried out without needing the availability of the watermark. In addition, our method adopts data compression for further reduction in the recovery packet size and improvement in embedding capacity. The experimental results demonstrate the superiority of the proposed scheme compared to the existing methods.**

***Index Terms*— Data compression, information hiding, lossless recovery, reversible watermarking, visible watermarking.**

I. Introduction

VISIBLE WATERMARKING is the study of techniques that insert copyright information perceptibly into the contents of cover digital multimedia so as to identify the ownership in a displayable manner and to prevent the viewers from making unauthorized use. In most conventional visible watermarking schemes [1]–[5], a visible watermark is usually designed to be irremovable in order to effectively resist unintended editing and malicious attacks [6]–[8]. However, in some potential applications, a visible watermark is required to be removable [9]–[13].

Manuscript received January 15, 2008; revised August 11, 2008. First version published March 16, 2009; current version published June 10, 2009. This paper is supported in part by the National Basic Research Program 973 (No. 2006CB303000) and the National Natural Science Foundation of China (Nos. 60573045, 60736016, and 60873198). This paper was recommended by Associate Editor M. Barni.

Y. Yang, X. Sun, H. Yang, and R. Xiao are with the School of Computer and Communication, Hunan University, Hunan 410082, China (e-mail: yingyuang@gmail.com; sunnudt@163.com; hengfuyang@163.com; ruong.x@gmail.com).

C.-T. Li is with the Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: ctli@dcs.warwick.ac.uk).

Digital Object Identifier 10.1109/TCSVT.2009.2017401

Depending upon whether the original signal is perfectly recovered or not after watermark removal, removable visible watermarking can be further classified into the following two categories: *irreversible* and *reversible*. This paper focuses on the latter. In the past, various reversible schemes have been developed using the techniques of, for example, modulo arithmetic [14], the circular interpretation of the bijective transform [15], lowest levels replacement [16], or difference expansion [17]. Nevertheless, these methods are applicable only to invisible watermarking. Compared to an invisible watermark, the embedding distortion inflicted by a visible watermark is often far greater. Achieving lossless recovery of the original host signal from a visibly watermarked signal is still an acute challenge.

The necessity for invertible visible watermarking is apparent. But unfortunately, this type of watermarking techniques has not been sufficiently investigated up to now. In the literature, to the best of our knowledge, there are only three works concentrating on distortion-free visible watermarking [11]–[13]. Hu *et al.* [11] first proposed a reversible visible watermarking scheme by modifying one significant bit plane of the pixels of the host image. They achieved reversibility via losslessly hiding the compressed version of the altered bit plane into the non-watermarked image region. However, the embedded visible watermark with this method appears to be somewhat blurred, and the visual quality of the original image is significantly distorted. Yip *et al.* [12] presented two lossless visible watermarking methods based on pixel value matching and pixel position shift, respectively. Tsai *et al.* [13] mapped the pixel values of the host image underlying the watermark into a small range for showing the watermark and then reversibly inserted a reconstruction packet into the watermarked image for perfect restoration. Despite the merit of [12] and [13], they need the original watermark for original image recovery, making them unsuitable for most applications in which the original watermark is unavailable at the recovery stage. Moreover, all the existing three methods do not consider human visual system (HVS) characteristics in the visible watermark embedding process. As a result, they are less visually satisfactory and more intrusive.

Aiming at addressing the issues of the aforementioned methods and maintaining applicability, we propose a lossless visible watermarking scheme that adaptively varies the watermark strength to be embedded in different areas of the host image, depending on the underlying image content and HVS characteristics. For reversibility, a recovery packet is embedded into the image itself. We develop a simple pixel

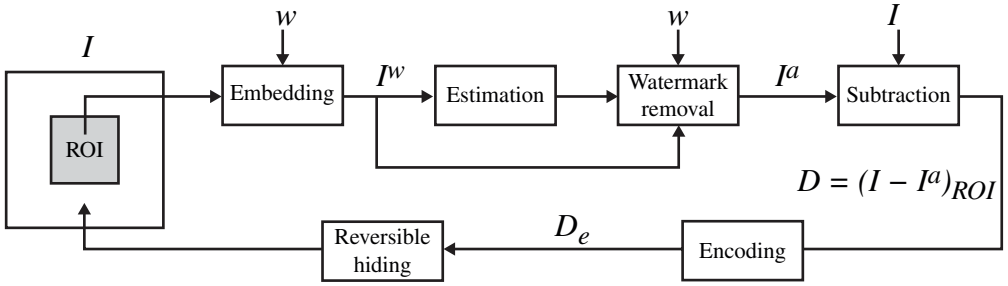

Fig. 1. Framework of our proposed embedding scheme.

prediction technique, and also exploit data compression, in order to alleviate the packet overhead and to improve embedding capacity. In addition, the proposed method adopts a unique encoding scheme for the recovery packet. This ensures that the original watermark pattern is not necessarily required when recovering the original host image.

The rest of the paper is organized as follows. In Section II, the embedding process of the proposed reversible visible watermarking algorithm is presented in detail. Section III briefly introduces the watermark removal and lossless image recovery. Section IV provides the experimental results for evaluating the performance of the algorithm. We draw the conclusion in Section V.

II. Embedding Process

Fig. 1 illustrates the framework of our proposed watermarking technique. The embedding process of the scheme mainly consists of two procedures: *visible watermark embedding* and *reversible data hiding*. The former procedure is to transparently overlap the binary watermark pattern W on the region of interest (ROI) in the host image I for the generation of the watermarked image I^w. And the latter is to reversibly embed a reconstruction data packet D into another area in I for lossless image recovery. The ROI is specified by image providers, and has the same size as W. To facilitate our description, we assume that the ROI is comprised of a multiple of some adjacent 8×8 blocks of the host image I.

A. Visible Watermark Embedding

With visible watermarking, a secondary image (the watermark in different regions) is inserted perceptibly into a primary (host) image so that the watermark is visible to the human eye. Generally, a visible watermark should be visible, yet must not significantly obscure the image details beneath it [3], [4]. Actually, the two requirements conflict with each other. If watermark energy is increased to improve visibility, the degradation in image quality becomes more significant, and vice versa. This motivates us to consider the HVS perception as well as the image content to accommodate a tradeoff between these conflicting requirements.

Let N be the number of 8×8 blocks of the host image I and $\mathcal{S} \subset \{1, 2, \ldots, N\}$ be the set of block ID numbers corresponding to the 8×8 blocks in ROI. In the proposed algorithm, the binary watermark pattern W is adaptively embedded into the host image I using

$$I_n^w(i,j) = \begin{cases} \lfloor \alpha_n \times I_n(i,j) \rfloor, & \text{if } W_{\tilde{n}}(i,j) = 1 \\ I_n(i,j), & \text{if } W_{\tilde{n}}(i,j) = 0 \end{cases} \quad 1 \le i, j \le 8 \text{ and } n \in \mathcal{S} \tag{1}$$

and

$$I_n^w(i,j) = I_n(i,j), 1 \le i, j \le 8 \text{ and } n \in \{1, 2, \ldots, N\} - \mathcal{S} \tag{2}$$

where the symbol $\lfloor \bullet \rfloor$ represents the mathematical floor function, $I_n^w(i,j)$ and $I_n(i,j)$ denote the (i,j)th spatial pixel values in the nth 8×8 blocks I_n^w and I_n of the watermarked image I^w and the host image I, respectively, $W_{\tilde{n}}(i,j)$ stands for the (i,j)th spatial pixel value in the $\tilde{n}$th 8×8 block $W_{\tilde{n}}$ of the watermark pattern W, and α_n is the adaptive scaling factor for the nth block of I. Note that we use the subscript $\tilde{n}$ instead of $n$for Win (1) because the nth block I_n in I may not correspond to the nth block W_n in W. The relationship between n and $\tilde{n}$ can be easily established once I, W, and ROI are known.

Now, we describe the determination of the scaling factor α_n, which is to determine the weights of the host image. Also, it determines the visibility of the watermark pattern and robustness in the marked image. In order to better conform to HVS characteristics, the texture features of the host image are taken into account in our proposed algorithm. There are two aspects of the HVS to consider when formulating the scaling factor.

1) First, the HVS is more sensitive to changes in mid-luminance areas [18]. That is, to maintain the quality of the visibly marked image, assigning greater value of the scaling factor for the mid-luminance areas is desirable. However, we also want the watermark pattern to be visible enough but not too intrusive. Therefore the optimal choice would be assigning greater scaling factor in the mid-luminance areas and attenuating its value at darker and brighter components. Intuitively, the histogram of the scaling factor is roughly parabola-shaped.
2) Second, because the HVS is less sensitive to changes made in highly textured regions [18], it is helpful to use a lower value for the scaling factor in textured regions.

The steps for determining the scaling factor are as follows. Note that, in what follows, we will use the same symbol,

e.g., I, to represent the same image in both the spatial and transform domains for simplifying our description.

Step 1: Transform all the non-overlapping 8 × 8-pixel blocks of the host image into the DCT domain.

Step 2: Compute the scaling factor according to the dc coefficients of the host image. This is due to the fact that most energy is concentrated in low-frequency components, especially the dc coefficients. Reininger *et al.* [19] demonstrated that, for many images, the dc coefficients are best approximated by a normal distribution. Here, the distribution model of the dc coefficients of the host image is expressed as

$$I_n(1,1) \sim \mathcal{N}(\mu, \sigma^2), \quad 1 \le n \le N \tag{3}$$

where $I_n(1,1)$ is the dc coefficient of the nth 8 × 8 block I_n of the host image I, and μ and σ^2 are the mean and variance of the dc coefficients of I, respectively. In order to create a parabola-shaped scaling factor α_n, we can formulate it as

$$\alpha_n = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{\{-[I_n(1,1)-\mu]^2/2\sigma^2\}}, \quad 1 \le n \le N \tag{4}$$

where the mean value μ and variance value σ^2 are, respectively, defined as

$$\mu = \frac{1}{N}\sum_{n=1}^{N} I_n(1,1) \tag{5}$$

and

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}[I_n(1,1) - \mu]^2. \tag{6}$$

This step reflects the first aspect of the HVS we mentioned earlier because we only take the dc components, which convey the luminance of the corresponding blocks, of the host image into account.

Step 3: To take the second aspect of the HVS into account in order to improve the performance, the scaling factor is corrected by involving the ac coefficients, which mainly reflect the texture features of the image. Less energy of the host image should be transferred to the visibly marked regions which are strongly textured because HVS is less sensitive to such regions. It has been observed that in strongly textured blocks, energy tends to be more evenly distributed among the ac coefficients, and therefore the variance of the ac coefficients tends to be smaller [3]. So for simplicity we assume that the scaling factor α_n is in direct proportion to the variance υ_n, which is the variance of the ac coefficients of the nth host image block I_n

$$\upsilon_n = \frac{1}{63} \times \sum_{(i,j)\neq(1,1)} [I_n(i,j) - \eta_n]^2, \quad 1 \le i,\ j \le 8 \text{ and } 1 \le n \le N \tag{7}$$

where $I_n(i,j)$ is the (i,j)th DCT coefficient in the nth host image block I_n, and η_n denotes the mean value of the ac coefficients of the nth host image block I_n; that is

$$\eta_n = \frac{1}{63} \times \sum_{(i,j)\neq(1,1)} I_n(i,j), \ 1 \le i,j \le 8 \text{ and } 1 \le n \le N. \tag{8}$$

Based on the aforementioned assumption, (4) can be revised as

$$\alpha_n = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{\{-[I_n(1,1)-\mu]^2/2\sigma^2\}} + \hat{\upsilon}_n, \quad 1 \le n \le N. \tag{9}$$

Here, $\hat{\upsilon}_n$ is the normalized logarithm of υ_n, calculated using

$$\hat{\upsilon}_n = \frac{\bar{\upsilon}_n - \min_n(\bar{\upsilon}_n)}{\max_n(\bar{\upsilon}_n) - \min_n(\bar{\upsilon}_n)}, \quad 1 \le n \le N. \tag{10}$$

In (10), $\bar{\upsilon}_n$ is the nature logarithm of υ_n, that is

$$\bar{\upsilon}_n = \ln(\upsilon_n), \quad 1 \le n \le N. \tag{11}$$

The parameter $\hat{\upsilon}_n$ is so defined in (10) to make the scaling factor α_n controlled in a narrow range so that the visual quality of the watermarked image can be kept well.

Step 4: Scale α_n to the range $[p', p'']$ so as to avoid obtrusive embedding. The two parameters p' and p'' are predetermined empirical constants.

Given the availability of the host image I, the watermark pattern W, and the scaling factors α_n, we can obtain the visibly watermarked image I^w after applying (1) and (2) on each 8×8 block in I.

B. Approximate Image Generation

From the proposed embedding method, we know that all the pixels not belonging to the ROI in the host image I are kept unchanged in its watermarked version I^w. This means that preserving the distortion/change in the ROI is sufficient for lossless recovery of the original host image. Clearly, one way to achieve this purpose is utilizing the difference image between the original ROI and the watermarked ROI as the reconstruction packet, which is denoted as D; that is

$$D = (I - I^w)_{\text{ROI}} \tag{12}$$

where the subscript "ROI" is used to denote the spatial ROI in the image. However, through experiments we observed that this approach suffers from high embedding overhead since the dynamic range and the power of the component $D(i,j)$ in D are fairly large. This motivates us to devise an alternative method that establishes the recovery packet D using the difference image between the original ROI and its approximation, that is

$$D = (I - I^a)_{\text{ROI}} \tag{13}$$

where I^a is an approximate version of the host image I. There are many methods that can be used to calculate the approximate image I^a. Utilizing a different method will result in different I^a, and hence, different D. Despite this, we only investigate one potential scheme to calculate I^a since seeking the optimal scheme is not the focus of this paper.

It is expected that the overhead of the reconstruction packet D is as light as possible. This means that we should attempt to minimize the difference between the host image I and the approximate image I^a. To this end, the generation process of I^a is designed to contain two procedures: one is estimating the original image so as to obtain an estimated version $\hat{\alpha}_n$ of the original scaling factor α_n, and the other is removing the embedded visible watermark using the estimated

$\hat{\alpha}_n$ in order to yield the approximate image I^a, as depicted in Fig. 1. Equation (1) suggests that, given the watermarked image, I^a approximates I better if $\hat{\alpha}_n$ is closer to α_n, and vice versa. Hence, we will not calculate $\hat{\alpha}_n$ from the watermarked image. Instead, we derive $\hat{\alpha}_n$ from an estimated version of the original image to improve estimation accuracy. This reflects the reason why it is necessary to estimate the original host image. Also, it is worthwhile to mention here that, in the entire calculation process, we will involve the watermarked image and the watermark pattern only, but not the original image because it is unavailable at the recovery phase.

To estimate the original image, we develop a pixel prediction technique that utilizes the non-watermarked pixels as auxiliary information to predict the original pixel values of their neighboring watermarked pixels. The pixels within the ROI in the marked image can be classified into two categories, namely, *non-watermarked* and *watermarked* pixels. Usually, the difference between two adjacent non-watermarked/unaltered pixels in most natural images is small, but the difference is typically quite large for two neighboring non-watermarked and visibly watermarked pixels. Based on this fact, the prediction function for each marked pixel $P(i, j)$, located at the center of a window P of $R \times R$ pixels in Fig. 2, can thus be formulated as

$$P(i, j) = \frac{1}{\|\mathcal{C}\|} \times \sum_x \sum_y P(x, y) \tag{14}$$
$$(x,y) \in \mathcal{C}$$

where $\|\bullet\|$ represents the number of elements in a set, and $\mathcal{C}$ is the set of coordinate pairs of all the non-watermarked and recovered/estimated pixels inside window P. The marked and non-marked points can be identified using the original watermark pattern. In this paper, once a marked pixel is recovered, it will be also used for the estimation of its neighboring watermarked points, if necessary. An estimated image is yielded via repeatedly moving the window P in a raster scan until all the watermarked pixels $P(i, j)$ get recovered. Other moving patterns, e.g., zigzag scanning and field scanning, may be used to move the window as well. Note that we design the embedder in such a way that it starts the prediction process from the pixel at the upper-left corner of the visibly watermarked ROI. As a result, when the visible watermark is embedded into the upper-left region of the host image, it is possible that no recovered pixels exist in the window. When this case happens, the prediction process is repeated.

By replacing the original host image in (3)–(11) with the image generated above, we have the estimated the scaling factor $\hat{\alpha}_n$. Moreover, we obtain the approximate version I^a of the original image I by removing the embedded visible watermark W from the watermarked image I^w utilizing

$$I_n^a(i, j) = \begin{cases} \lfloor \dfrac{I_n^w(i, j)}{\hat{\alpha}_n} \rfloor, & \text{if } W_{\tilde{n}}(i, j) = 1 \\ I_n^w(i, j), & \text{if } W_{\tilde{n}}(i, j) = 0 \end{cases}$$
$$1 \le i,\ j \le 8 \text{ and } n \in \mathcal{S} \tag{15}$$

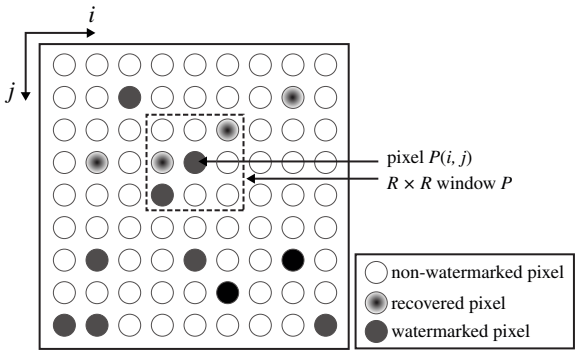

Fig. 2. Diagram showing the prediction process according to (14). An $R \times R$ pixel window P is highlighted. In this figure, $R = 3$.

and

$$I_n^a(i, j) = I_n^w(i, j), 1 \le i, j \le 8 \text{ and } n \in \{1, 2, \ldots, N\} - \mathcal{S} \tag{16}$$

where $I_n^a(i, j)$ denotes the (i, j)th spatial pixel of the nth 8×8 block I_n^a in the approximate image I^a. Plugging I and I^a into (13), we have the reconstruction data packet D.

C. Encoding, Decoding, and Watermark Reconstruction

For lossless recovery, the recovery data packet D must be reversibly inserted into the non-visibly-marked regions of the host image, as illustrated in Fig. 1. On the other hand, it is also expected that the image recovery process can be performed without requiring the original watermark pattern. Unfortunately, we observe that, if hiding D directly, the original watermark pattern may be needed at the recovery phase unless it is also embedded as auxiliary information. However, additional embedding of the watermark will increase the overhead of the hidden payload. This becomes especially worse when the watermark size is large enough. To cope with this problem, we propose an interesting scheme to encode the reconstruction packet D and embed the encoded version D_e of the original D instead. Using the encoding method, we can derive the original watermark from the encoded packet, D_e.

1) Reconstruction Packet Encoding: Because the watermark pattern W used in this paper is binary, we can classify its pixels into two categories: one corresponding to watermark bit $W(i, j) = 1$, and the other corresponding to $W(i, j) = 0$. Let Γ and $\hat{\Gamma}$ be, respectively, the sets of their corresponding spatial indices such that $\Gamma = \{(i, j) | W(i, j) = 1\}$ and $\hat{\Gamma} = \{(i, j) | W(i, j) = 0\}$. From our algorithm, neither the watermarking modification nor the pixel estimation is applied to those pixels of the host image I that correspond to $W(i, j) = 0$. This implies that these pixels remain unchanged in the approximate image I^a. Therefore, according to (13), we derive that

$$D(i, j) = \begin{cases} 0, & \forall (i, j) \in \hat{\Gamma} \\ \text{any value}, & \forall (i, j) \in \Gamma \end{cases} \tag{17}$$

where $D(i, j)$ is the (i, j)th component in the recovery packet D. For any $(i, j) \in \Gamma$, the value of $D(i, j)$ has the following three possible cases.

Case 1: $\min_{\forall(i,j)\in\Gamma} D(i,j) > 0$, meaning the elements in the set $\{D(i,j)|(i,j)\in\Gamma\}$ are all greater than zero.
Case 2: $\max_{\forall(i,j)\in\Gamma} D(i,j) < 0$, meaning the elements in the set $\{D(i,j)|(i,j)\in\Gamma\}$ are all smaller than zero.
Case 3: Other potential case except **Case 1** and **Case 2**, meaning the element in the set $\{D(i,j)|(i,j)\in\Gamma\}$ may be negative, or zero, or positive.

For different cases, we adopt different encoding schemes.

For **Case 1**, each component $D(i,j)$ is kept unchanged without encoding, i.e.

$$D_e(i,j) = D(i,j), \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{18}$$

where $D_e(i,j)$ denotes the decoding result corresponding to the original $D(i,j)$.

For **Case 2**, we encode each component $D(i,j)$ by employing

$$D_e(i,j) = \begin{cases} D(i,j) = 0, & \text{if } (i,j)\in\hat{\Gamma} \\ -D(i,j), & \text{if } (i,j)\in\Gamma. \end{cases} \tag{19}$$

Equation (19) suggests that every negative number $D(i,j)$ is converted into a positive number, and all the zero-valued components are kept unaltered. The reason why we make such a conversion is for compression, as explained later.

As in **Case 1** and **Case 2**, where only one type of values exists in $\{D(i,j)|(i,j)\in\Gamma\}$, we do not apply special encoding scheme to them. That is because the decoder can correctly retrieve each watermark bit $W(i,j)$ by checking whether its corresponding $D(i,j)$ is equal to zero or not. However, the encoding scheme used for **Case 1** or **Case 2** is not valid for **Case 3** because $\{D(i,j)|(i,j)\in\Gamma\}$ in **Case 3** may take any value. To deal with **Case 3**, we therefore have to solve two crucial problems: when recovering, how to judge the sign (positive or negative) of the original $D(i,j)$, and where $(i,j)\in\Gamma$, and how to reconstruct the watermark pattern W based on the recovery packet. Note that when reconstructing the watermark pattern W, the watermark bit $W(i,j)=1$ may correspond to $D(i,j)=0$, so we cannot reconstruct $W(i,j)$ simply by checking whether $D(i,j)$ is zero or not. The two problems are related to the reconstruction of the recovery packet and the watermark pattern W. Fortunately, we found that the aforementioned problems can be efficiently handled by introducing a redundant bit (also referred to as a mark bit). With the assistance of this bit, the decoder is able to not only correctly determine the sign, but also to exactly retrieve the original watermark pattern.

For **Case 3**, the details of the encoding processes are described below. Each $D(i,j)$ is first converted into an L-bit temporary binary string $\hat{D}(i,j)$ by

$$\hat{D}(i,j) = \mathbf{Dec2Bin}_L(|D(i,j)|), \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{20}$$

where the function $\mathbf{Dec2Bin}_L(\cdot)$ returns the L-bit binary value of a decimal number, the symbol $|\cdot|$ denotes the absolute value of a number, and

$$L = \left\lceil \log_2\left(\max_{(i,j)\in\Gamma\cup\hat{\Gamma}} |D(i,j)| + 1\right)\right\rceil + 1 \tag{21}$$

where $\lceil\cdot\rceil$ denotes the ceiling function. Note that, during conversion, if $|D(i,j)|$ needs only l bits for representation, where $l < L$, $L-l$ zeros should be padded at the beginning of the l bits to ensure that each $\hat{D}(i,j)$ is composed of L bits. An additional bit is assigned to each $\hat{D}(i,j)$ [see (21)], and hence it is inferred that the leftmost bit of $\hat{D}(i,j)$ must be 0. Then, we replace the leftmost bit of each $\hat{D}(i,j)$ with the mark bit 1 for $D(i,j)\le 0$, where $(i,j)\in\Gamma$, and keep this bit unchanged for $D(i,j) > 0$, where $(i,j)\in\Gamma$, and $D(i,j)=0$, where $(i,j)\in\hat{\Gamma}$. Finally, the modified $\hat{D}(i,j)$ is converted according to (22) to yield the final encoding result D_e

$$D_e(i,j) = \mathbf{Bin2Dec}(\hat{D}(i,j)), \quad (i,j)\in\Gamma\cup\hat{\Gamma}. \tag{22}$$

Here, the function $\mathbf{Bin2Dec}(\cdot)$ converts a binary string to its corresponding decimal number. In practice, the above encoding process is equivalent to

$$D_e(i,j) = \begin{cases} 2^{L-1} - D(i,j), & \text{if } D(i,j)\le 0 \text{ and } (i,j)\in\Gamma, \\ D(i,j), & \text{otherwise} \end{cases} \quad (i,j)\in\Gamma\cup\hat{\Gamma}. \tag{23}$$

The encoding scheme for **Case 3** appears to be somewhat complex, so we give an example in the following to reinforce its encoding process. Here, the mark bit is indicated using an underline "_". If $L=4$, $D(i,j)=2$ and $D(i,j)=-2$ are, respectively, converted into $\hat{D}(i,j) = \underline{0}010$ and $\hat{D}(i,j) = \underline{1}010$. Thus, the final encoding results for them are $D_e(i,j) = 2$ and $D_e(i,j) = 10$. Besides, when $L=4$, $D(i,j) = 0$, where $(i,j)\in\Gamma$, and $D(i,j)=0$, where $(i,j)\in\hat{\Gamma}$, are converted into $\hat{D}(i,j) = \underline{1}000$ and $\hat{D}(i,j) = \underline{0}000$, respectively. Therefore, the final encoding results for them are $D_e(i,j)=8$ and $D_e(i,j)=0$.

2) Reconstruction Packet Decoding and Watermark Reconstruction: Decoding the given D_e, we obtain not only the original reconstruction packet D, but also the watermark pattern W. For different cases mentioned above, different decoding scheme will be adopted.

For **Case 1** and **Case 2:** The recovery processes for $D$and W are fairly simple, and can be, respectively, given by

$$D(i,j) = \begin{cases} D_e(i,j), & \text{for } \mathbf{Case\ 1}, \\ -D_e(i,j), & \text{for } \mathbf{Case\ 2}, \end{cases} \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{24}$$

and

$$W(i,j) = \begin{cases} 0, & \text{if } D(i,j)=0. \\ 1, & \text{if } D(i,j)\neq 0. \end{cases} \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{25}$$

For **Case 3**: Each decimal $D_e(i,j)$ in D_e is first converted into an L-bit binary string $\hat{D}(i,j)$ via

$$\begin{aligned} \hat{D}(i,j) &= \mathbf{Dec2Bin}_L(D_e(i,j)), \\ &= b_1 b_2 \cdots b_L \quad (i,j)\in\Gamma\cup\hat{\Gamma} \end{aligned} \tag{26}$$

where $b_n\in\{0,1\}$ and $1\le n\le L$. Now, we can retrieve the original D by binary-to-decimal conversion, that is

$$D(i,j) = \begin{cases} \mathbf{Bin2Dec}(b_1 b_2 \cdots b_L), & \text{if } b_1 = 0, \\ -\mathbf{Bin2Dec}(b_1 b_2 \cdots b_L), & \text{if } b_1 = 1, \end{cases} \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{27}$$

and furthermore have the original watermark W by

$$W(i,j)=\begin{cases}0, & \text{if } b_1=0 \text{ and } D(i,j)=0,\\ 1, & \text{otherwise}\end{cases} \quad (i,j)\in\Gamma\cup\hat{\Gamma} \tag{28}$$

the highest bit, i.e., b_1, in each $\hat{D}(i,j)$ is a mark bit, hence it is excluded from the binary-to-decimal conversion process (27). From the encoding method, it is easy to see the watermark bit $W(i,j)=0$ if and only if the mark bit $b_1=0$ and the original $D(i,j)=0$. We thereby reconstruct the original watermark image W according to (28).

D. Reversible Data Hiding

Unlike most reversible watermarking approaches that incorporate lossless data compression [16], [17], [22], [23], the RCM (reversible contrast mapping) based algorithm achieves high-capacity data embedding without any additional data compression stage [24]. Let $[0, G]$ be the image gray-level range ($G=255$ for an 8-bit grayscale image), and (δ_1,δ_2) be a pair of pixels. The forward and inverse transforms are, respectively, defined as

$$\begin{cases}\delta_1'=2\delta_1-\delta_2\\ \delta_2'=2\delta_2-\delta_1\end{cases} \tag{29}$$

and

$$\begin{cases}\delta_1=\left\lceil\frac{2}{3}\delta_1'+\frac{1}{3}\delta_2'\right\rceil\\ \delta_2=\left\lceil\frac{1}{3}\delta_1'+\frac{2}{3}\delta_2'\right\rceil\end{cases} \tag{30}$$

where $\lceil\bullet\rceil$ represents the ceiling function. To prevent overflow and underflow, the transformed pixel pair (δ_1',δ_2') should be limited to $[0, G]\times[0, G]$, i.e.

$$\begin{cases}0\le 2\delta_1-\delta_2\le G\\ 0\le 2\delta_2-\delta_1\le G.\end{cases} \tag{31}$$

It is clear that, even if the LSB of either δ_1' or δ_2' is lost, (30) can still exactly recover δ_1 and δ_2 with the help of the ceiling function. But when both LSBs of δ_1' and δ_2' are lost, (30) fails to exactly recover original pair (δ_1,δ_2). In the light of this fact, the data bits are embedded into the space occupied by these LSBs of the transformed pixel pairs. The reader is referred to [24] for more details about the processes of data embedding and lossless recovery. In consideration of large embedding capacity and low mathematical complexity, we employ the reversible data hiding technique [24] in this paper.

To further reduce the overhead of the embedded payload, we remove the redundancy existing in the encoded reconstruction packet D_e using lossless compression schemes. To this end, we use a JBIG2 codec, which is an international standard for lossless compression [20]. More specifically, we chose the open C code of JBIG-KIT [21] to compress D_e.

In addition, for security reasons, a $\{0, 1\}$ sequence that follows uniform distribution is first generated using a pseudo random number generator seeded with a secret key. Next, the compressed encoded recovery packet D_e is converted into its corresponding binary sequence. And then, we perform bitwise Exclusive-OR operation on the key-dependent binary sequence and the binary sequence to be embedded. Finally, we losslessly hide the resulting encrypted binary sequence, exploiting the method of [24]. The reason why we utilize the simple Exclusive-OR encryption algorithm is that it will not change the size of the original plain data after encrypting. When the application requires higher security level, one can use other complex encryption algorithms, e.g., AES, but they may probably increase the original plain data size. The security discussion is beyond the scope of this paper, and thereforewe will not discuss it in detail here.

III. Watermark Removal and Original Image Recovery

The original host image can be losslessly retrieved after removing the embedded visible watermark pattern. The procedure of recovering the host image is carried out by reversing the operations of the embedding process as illustrated in Fig. 1. We simply describe the recovery process below. Note that, during recovery, we do not need the original watermark pattern, but some side information to help the decoder. The side information includes the secret key, the spatial position of the ROI in the host image, watermark pattern size, the case (**Case 1**, or **Case 2**, or **Case 3**) to which the recovery packet D belong to, encoding length L assigned to each component $D(i,j)$ in D, JBIG-compressed data size, and the length of the automatically produced bits in the reversible embedding phase (see [24]).

First, we extract the embedded binary sequence from the area outside the ROI of the image to obtain the watermarked image I^w. Second, using the secret key, we produce the same $\{0, 1\}$ sequence in the embedding process and perform bitwise Exclusive-OR operation on the key-controlled binary sequence and the extracted binary sequence. Third, the encoded payload D_e is attained after applying decompression to the decrypted data, and furthermore, by decoding D_e we reconstruct the original reconstruction packet$D$and the original watermark pattern W (see Section II-C). Fourth, we apply the pixel prediction technique as utilized during embedding to the watermarked image I^w so as to generate a roughly estimated version of the original host image I. Note that in the estimation process, the watermark W constructed earlier is required to indicate which pixels in the marked image have been watermarked. This useful information helps the decoder know which pixels need to be estimated. Fifth, according to the estimated image, we calculate the estimated scaling factor $\hat{\alpha}_n$, and furthermore, obtain the approximate version I^a of the original image I after plugging $\hat{\alpha}_n$ into (15) and (16) to remove W from I^w. Based on (13), the original pixels within ROI of the host image I can be recovered via pixel-to-pixel addition of the reconstruction packet$D$and the ROI of the approximate image I^a: that is, $I_{\mathrm{ROI}}=I^a_{\mathrm{ROI}}+D$. Finally, we losslessly retrieve the host image I by replacing the pixels in ROI of the watermarked image I^w with corresponding values in I_{ROI}.

(a) *Lena* (b) *F-16* (c) *Baboon* (d) *Barbara* (e) *Watermark*

Fig. 3. Images for test. (a)–(d) are 512 × 512 gray-scale host images and (e) is a 128 × 128 binary watermark image.

IV. Experimental Results

The proposed watermarking algorithm has been implemented and tested on a number of grayscale images (obtained in part from [25]) and different watermark patterns for evaluating its performance. These images are of various texture characteristics. Some images used in the evaluation are shown in Fig. 3. In the experiments, we set the embedding parameters $p' = 0.84$ and $p'' = 0.89$ (see **Step 4** of Section II-A). They are empirical values determined by conducting several experiments to strike a balance between the visual quality of the host image and the visibility of the visible watermark pattern. Considering the estimation accuracy, the size of the sliding window ($R \times R$) is set to 3×3 (see Fig. 2).

A. Embedding With Different ROIs

Fig. 4 shows the visibly watermarked images that are yielded by embedding the watermark pattern Fig. 3(e) into different areas of the host images. From these images, we find that their apparent difference is in the watermark visibility. The watermark is more visible in smooth image areas.

To subjectively assess the visual quality of the visibly watermarked image regions, we utilize the perceptually inspired metrics PSNR, WPSNR [26], and Structural SIMilarity (SSIM) [27] in this paper, and the relevant experimental results are list in Table I. From the data in the "PSNR-2" column in the table, we can see that the image *F*-16 with large smooth regions has the lowest PSNR values (around 23 dB) among all images. This suggests that larger embedding distortion resulting from the visible watermark has been introduced into *F*-16. Also, this conclusion can be further demonstrated by comparing the WPSNR and SSIM values of these test images. The average SSIM for *F*-16 is as low as 0.52, while the average values of SSIM for *Lena*, *Baboon*, and *Barbara* are, respectively, 0.74, 0.74, and 0.86. Nevertheless, the watermark is more visible in the watermarked *F*-16.

On the other hand, Table I also shows that the size of the hidden payload (see the "Payload size" column), which is reversibly embedded into the area left for the recovery of ROI. As shown in this table, the image *F*-16 almost has the smallest recovery packet size than the other host images. This phenomenon is especially apparent when the watermark image is inserted at the bottom-left area of the host image. Under such circumstance, a shorter encoding length $L = 2$ is utilized during encoding for *F*-16, while $L = 3$ or $L = 4$ is needed for other test images. This results in a significantly smaller embedded payload size for *F*-16 and, hence, higher PSNR values (see "PSNR-1" column in Table I) as compared to other images. These experimental results agree quite well with theoretical analysis. The prediction result is much more accurate in smooth regions, and hence the approximate image considerably approximates the original host image. Therefore, the component $D(i, j)$ of the recovery packet D has a small amplitude value and, furthermore, a smaller L.

Fig. 4. Watermarked images generated by embedding the watermark pattern Fig. 3 (e) into different areas of the host images.

B. Embedding With Various Watermark Sizes

Clearly, the larger the size of the visible watermark pattern, the smaller the size of the area left for the embedding of the reconstruction data packet, thereby affecting embedding performance. To investigate its impact on the embedding performance, we conduct the experiments by varying the watermark size. In the experiments, each watermark pattern is embedded into the bottom-left region of the host images.

TABLE I

PERFORMANCE EVALUATION. THE WATERMARKED IMAGES ARE OBTAINED AFTER EMBEDDING FIG. 3(E) INTO VARIOUS REGIONS OF THE HOST IMAGES. HERE, (x, y) DENOTES THE SPATIAL LOCATION OF THE MOST BOTTOM-LEFT WATERMARK POINT IN THE HOST IMAGE. "PAYLOAD SIZE" (BYTE) IS THE SIZE OF THE PURE EMBEDDED DATA FOR RECOVERING THE WATERMARKED ROI. PSNR-1 (DB) IS CALCULATED WITHOUT ROI, AND PSNR-2, WPSN, AND SSIM ARE COMPUTED ACCORDING TO ROI, WHERE BOTH PSNR-1 AND PSNR-2 INDICATE PSNR

Images	Watermark size	(x, y)	Payload size	L	PSNR-1	PSNR-2	WPSNR	SSIM
Lena	128 × 128	(1, 1)	2726	3	37.37	28.27	35.78	0.83
		(257, 257)	2817	4	37.16	24.95	32.56	0.66
		(321, 321)	2849	4	37.10	25.50	32.76	0.72
F-16	128 × 128	(1, 1)	2567	3	36.71	23.26	30.69	0.46
		(257, 257)	2871	4	37.69	24.30	31.72	0.64
		(321, 321)	1550	2	42.03	23.34	30.77	0.47
Baboon	128 × 128	(1, 1)	2550	3	30.96	25.01	32.48	0.76
		(257, 257)	2723	3	30.98	23.48	30.90	0.59
		(321, 321)	2704	4	31.02	25.80	33.22	0.88
Barbara	128 × 128	(1, 1)	2715	3	30.34	31.53	38.88	0.85
		(257, 257)	2497	3	33.02	25.98	33.40	0.86
		(321, 321)	2664	3	32.56	28.47	35.95	0.87

TABLE II

PERFORMANCE EVALUATION. THE VISIBLE WATERMARK PATTERN IS EMBEDDED INTO THE BOTTOM-LEFT AREA OF THE HOST IMAGE. "PAYLOAD SIZE" (BYTE) IS THE SIZE OF THE PURE EMBEDDED DATA FOR RECOVERING THE WATERMARKED ROI. PSNR-1 (DB) IS CALCULATED WITHOUT ROI, AND PSNR-2, WPSN, AND SSIM ARE COMPUTED ACCORDING TO ROI, WHERE BOTH PSNR-1 AND PSNR-2 INDICATE PSNR

Images	Watermark size	Payload size	L	PSNR-1	PSNR-2	WPSNR	SSIM
Lena	32 × 32	265	3	47.15	27.52	35.39	0.80
	64 × 64	668	3	43.49	25.62	31.99	0.72
	128 × 128	1607	3	40.41	27.86	33.88	0.89
	256 × 256	2994	4	39.05	31.10	37.14	0.97
F-16	32 × 32	165	2	54.57	24.60	32.50	0.45
	64 × 64	671	3	47.58	22.69	29.03	0.43
	128 × 128	1479	3	39.95	22.91	28.91	0.65
	256 × 256	3203	4	33.80	27.08	33.08	0.93
Baboon	32 × 32	264	3	44.61	26.69	34.58	0.74
	64 × 64	668	3	39.23	24.10	30.44	0.67
	128 × 128	1625	3	32.72	24.43	30.45	0.86
	256 × 256	4007	5	30.38	28.08	34.15	0.96
Barbara	32 × 32	161	2	52.27	30.08	38.01	0.71
	64 × 64	664	3	42.80	31.56	37.92	0.86
	128 × 128	1580	3	33.95	30.73	36.74	0.90
	256 × 256	3024	4	27.66	32.49	38.53	0.97

The experimental results are tabulated in Table II. As shown in this table, for each of the test images, the size of the embedded payload becomes larger as the watermark size increases. As a result, the PSNR value (see "PSNR-1" column in Table II) of the area utilized for the payload decreases gradually. Besides, the encoding length L shows a rising trend with increasing watermark size.

In addition, Table II shows the PSNR, WPSNR, and SSIM values for the ROI of each of the test images. As shown in this table, they, on the whole, increase gradually as the watermark size becomes larger. Also, Table II discloses that the SSIM values are large for most test images, and it is especially close to one when the watermark is of 256 × 256 size. This implies that our proposed watermarking technique well preserves the visual quality of the host image. Achieving such superior performance is attributed to the exploitation of the HVS characteristics, as well as the image content, in our method. Again, the relatively low PSNR, WPSNR, and SSIM values for *F*-16 demonstrate that the embedding of the visible watermark may easily distort the smooth images.

In summary, the size of watermark image largely affects three factors, i.e., the size of the reconstruction packet, the size of the image area used for the payload, and the image quality.

C. Performance Comparison

In this section, we compare the performance of the presented method against that of the previous reversible visible watermarking algorithms [11]–[13]. For comparison, we consider the following aspects.

1) *Image quality*: Often, the embedded visible watermark pattern is expected to be visible enough, but not too unobtrusive. We quantify the visual quality by employing both WPSNR and SSIM, where a larger WPSNR or SSIM value indicates a higher quality watermarked image.
2) *Watermark visibility*: Unfortunately, to the best of our knowledge, there is no commonly accepted indicator or model so far to measure the visibility of the visible

Fig. 5. Performance comparison in terms of watermark visibility and image degradation between our proposed algorithm and Hu *et al.*'s [11]. The visibly watermarked images in the first and third column are obtained by utilizing the second MSB and the MSB planes, respectively, of the host images as the bit planes altered by the watermark with the method of [11]. The images in the second and fourth column are the magnified versions of the watermarked areas of the watermarked images in first and third column, respectively. The images in the fifth column are the magnified versions of the watermarked areas generated by embedding the watermark into the same image area as columns 1 and 3 with our scheme.

watermark. We therefore compare the visibility through close inspection of the watermarked image.

3) *Availability of the watermark pattern at the recovery stage*: In some applications, the original watermark pattern is unavailable at the recovery stage, meaning that only the schemes that do not require the availability of the original watermark are acceptable for these circumstances.

Our implementation of the schemes reported in [11], [12], and [13] are described as follows.

1) *Implementation of the Method by Hu et al.*: Our implementation of the method of Hu *et al.* [11] allows the MSB (most significant bit) plane and the second MSB plane of the original image as the planes to be altered by the binary watermark pattern. The insertion of the watermark is based on the key-controlled {0, 1} sequence with uniform distribution.
2) *Implementation of the Method by Yip et al.:* Our implementation of the method of Yip *et al.* [12] uses the lossless embedding function formulated as follows:

$$Q(x,y) = \begin{cases} (P(x,y)+c+n) \bmod 256, & \text{if } W(x,y)=0 \\ P(x,y), & \text{if } W(x,y)=1 \end{cases} \tag{32}$$

where $P(x,y)$, $Q(x,y)$, and $W(x,y)$, are the original and the watermarked pixels, and the watermark bit at the spatial location (x,y), respectively. c is a user -defined constant and n is a variable integer number generated by a secret key. Note that, for consistency reason, we watermark the pixels corresponding to $W(x,y)=1$ and keep the pixels corresponding to $W(x,y)=0$ intact when implementing the method. Like Yip *et al.*, we set c to be 30 in the comparison experiment.

3) *Implementation of the Method by Tsai et al.:* Our implementation of the method of Tsai *et al.* [13] generates the discrete random variable with uniform distribution in the interval $[-12, 12]$.

Fig. 5 shows the performance comparison in terms of the visibility of the visible watermark and the degradation of the host image of our proposed method against that of Hu *et al.*'s method. The visibly watermarked images in the first and third column are obtained using the MSB plane and the second MSB plane as the bit planes replaced by the watermark with Hu *et al.*'s method. As shown in the figure, it is easy to see that the visible watermark is light in the first case but heavy in the second case. That is because the modification in the second case is performed in the MSB plane so that a larger embedding distortion has incurred. From the magnified images illustrated in the second and fourth columns of Fig. 5,

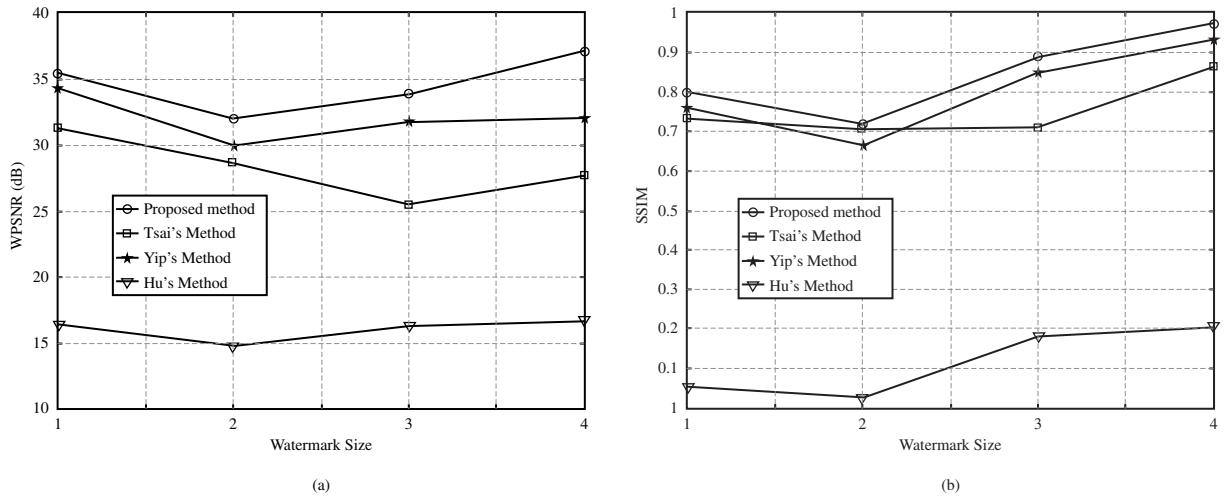

Fig. 6. Performance comparison of embedding distortion in terms of WPSNR and SSIM with various watermark sizes. The number in the x-axis represents the watermark size. 1: 32 × 32; 2: 64 × 64; 3: 128 × 128; and 4: 256 × 256. The test image is *Lena*.

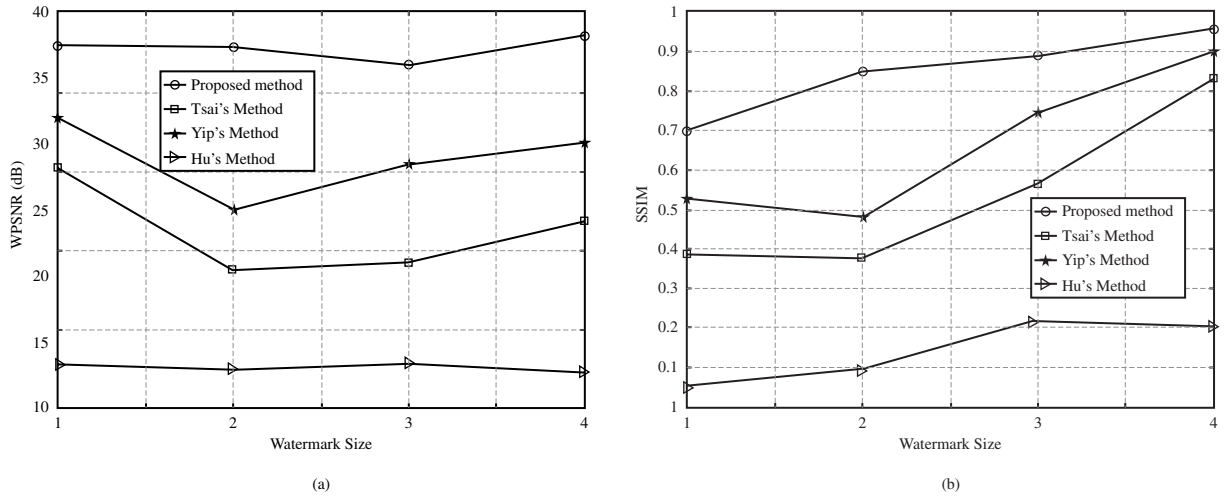

Fig. 7. Performance comparison of embedding distortion in terms of WPSNR and SSIM with various watermark sizes. The number in x-axis represents the watermark size. 1: 32 × 32; 2: 64 × 64; 3: 128 × 128; and 4: 256 × 256. The test image is *Barbara*.

we can easily see that the watermark pattern in Hu *et al.*'s approach is somewhat illegible in the watermarked images, and, furthermore, the visual qualities of the original images beneath the watermark are greatly degraded. It is almost unable to know the original image details after watermarking. Clearly, from the images shown in the fifth column of Fig. 5, our proposed watermarking scheme not only maintains a better visibility of the watermark, but also inflicts less distortion on the original host image than in Hu *et al.*'s method.

Since Yip *et al.*'s, Tsai *et al.*'s, and our proposed methods achieve almost the same level of watermark visibility, we do not show the images with watermarks embedded here. We therefore compare the performance only from the perspective of the distortion degree of the host images. Figs. 6 and 7, respectively, show the comparison of the embedding distortion in terms of both WPSNR and SSIM with various watermark sizes inflicted by our algorithm and [11]–[13] for images *Lena* and *Barbara*. In the experiments, the WPSNR and SSIM values are calculated according to the watermark-covered regions before and after embedding. The MSB plane is used as the plane altered by the watermark image with Hu *et al.*'s method. Apparently, from Figs. 6 and 7, our proposed approach consistently achieves higher WPSNR and SSIM values than the other three state-of-the-art approaches, while Hu *et al.*'s scheme has the lowest the WPSNR and SSIM for each of the test images. These comparison results suggest that the fidelity of the images watermarked by our method is better than those by the other three methods. This may be attributed to the fact that the HVS characteristics and the image content are taken into consideration in our method, but not considered in the other three methods.

Like our method, Yip *et al.*'s and Tsai *et al.*'s schemes reveal the watermark pattern by modifying those pixels of the host image according to the corresponding watermark bits. During the recovery process, the watermark image is required to help the decoder identify the watermarked pixels. However, theses two methods require the watermark as the input to the decoder since they cannot obtain it from the watermarked image. In contrast, our proposed method allows the decoder to reconstruct the watermark pattern so that it does not need

to be available. This is attributed to the encoding scheme that we apply to the recovery data packet. Clearly, the proposed watermarking technique outperforms the two schemes because it, on the one hand, requires cheaper storage space, and, on the other hand, it can be used in more applications without requiring the availability of the original watermark pattern.

D. Security Consideration

In practical applications, it is expected that unauthorized users cannot losslessly restore the original host image. For this reason, we make the proposed watermarking algorithm dependent on a secret key. During embedding, the reconstruction packet is encrypted by the secret key prior to hiding. At the recovery stage, the same secret key is required to decrypt the encrypted data. Thus, only authorized users with the correct key can perfectly recover the original image.

V. Conclusion

We have presented a reversible visible watermarking technique in this paper, which allows lossless recovery of the original host image. Contrary to the previous reversible visible methods [11]–[13], our proposed method considers the HVS characteristics, as well as the image content, to achieve the desired features of visible watermarking. To alleviate the overhead of the data packet used for exact recovery, a pixel prediction technique has been developed to construct an approximate version of the host image. The proposed algorithm is also based on data compression, thereby greatly enhancing the embedding capacity. As demonstrated by the experimental results, our method not only reveals the visible watermark pattern in a more visible manner, but also better preserves the visual quality of the host image than the existing methods [11]–[13]. In addition, our proposed scheme is superior to [12] and [13] because its recovery process does not require the availability of the original watermark pattern. As a key-dependent method, our proposed watermarking scheme only allows authorized users with the correct secret key to recover the original image.

We are currently investigating other techniques for obtaining reconstruction data packet of reduced size and the possibility of devising a general metric for evaluating the visibility of visible watermark.

References

[1] M. S. Kankanhalli, Rajmohan, and K. R. Ramakrishnan, "Adaptive visible watermarking of images," in *Proc. IEEE Int. Conf. Multimedia Comput. Syst.*, vol. 1. Florence, SC, Jul. 1999, pp. 568–573.

[2] J. Meng and S. F. Chang, "Embedding visible video watermarks in the compressed domain," in *Proc. IEEE Int. Conf. Image Process.*, vol. 1. Chicago, IL, Oct. 1998, pp. 474–477.

[3] S. P. Mohanty, K. R. Ramakrishnan, and M. S. Kankanhalli, "A DCT domain visible watermarking technique for images," in *Proc. IEEE Int. Conf. Multimedia Expo.*, vol. 2. New York, NY, 2000, pp. 1029–1032.

[4] B. B. Huang and S. X. Tang, "A contrast-sensitive visible watermarking scheme," *IEEE Multimedia*, vol. 13, no. 2, pp. 60–67, Apr.-Jun. 2006.

[5] R. Lukac and K. N. Plataniotis, "Secure single-sensor digital camera," *Electron. Lett.*, vol. 42, no. 11, pp. 627–629, May 2006.

[6] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. 27th Ann. Conf. Comput. Graph. Interactive Technol.* New Orleans, LA, 2000, pp. 417–424.

[7] C. H. Huang and J. L. Wu, "Attacking visible watermarking schemes," *IEEE Trans. Multimedia*, vol. 6, no. 1, pp. 16–30, Feb. 2004.

[8] S. C. Pei and Y. C. Zeng, "A novel image recovery algorithm for visible watermarked images," *IEEE Trans. Inf. Forens. Security*, vol. 1, no. 4, pp. 543–550, Dec. 2006.

[9] Y. Yang, X. Sun, H. Yang, and C.-T. Li, "Removable visible image watermarking algorithm in the discrete cosine transform domain," *J. Electron. Imaging*, vol. 17, no. 3, pp. 033008-1–033008-11 Jul.–Sep. 2008.

[10] Y. J. Hu, S. Kwong, and J. Huang, "An algorithm for removable visible watermarking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 129–133, Jan. 2006.

[11] Y. J. Hu and B. Jeon, "Reversible visible watermarking and lossless recovery of original images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 11, pp. 1423–1429, Nov. 2006.

[12] S. K. Yip, O. C. Au, C. W. Ho, and H. M. Wong, "Lossless visible watermarking," in *Proc. IEEE Int. Conf. Multimedia Expo*, Toronto, ON, Canada, 2006, pp. 853–856.

[13] H. M. Tsai and L. W. Chang, "A high secure reversible visible watermarking scheme," in *Proc. IEEE Int. Conf. Multimedia Expo*, Beijing, China, 2007, pp. 2106–2109.

[14] C. W. Honsinger, P. W. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," U.S. Patent #6278 791, Aug. 2001.

[15] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Trans. Multimedia*, vol. 5, no. 1, pp. 97–105, Mar. 2003.

[16] 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.

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

[18] C. H. Chou and Y. C. Li, "A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 6, pp. 467–476, Dec. 1995.

[19] R. C. Reininger and J. D. Gibson, "Distributions of the two-dimensional DCT coefficients for images," *IEEE Trans. Commun.*, vol. COM-31, no. 6, pp. 835–839, Jun. 1983.

[20] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 838–848, Nov. 1998.

[21] M. Kuhn. (1995, Jun. 8). *JBIG-KIT* [Online]. Available: http://www.cl.cam.ac.uk/~mgk25/jbigkit/

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

[23] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding——new paradigm in digital watermarking," *EURASIP J. Appl. Sig. Process.*, vol. 2002, no. 2, pp. 185–196, Feb. 2002.

[24] D. Coltuc and J. M. Chassery, "Very fast watermarking by reversible contrast mapping," *IEEE Signal Process. Lett.*, vol. 14, no. 4, pp. 255–258, Apr. 2007.

[25] A. Weber, Image Database. Signal & Image Process. Inst., Univ. Southern California, LA. [Online]. Available: http://sipi.usc.edu/database/

[26] M. Miyahara, K. Kotani, V. R. Algazi, "Ojective picutre quality scale (PQS) for image coding," *IEEE Trans. Commun.*, vol. 46, no. 9, pp. 1215–1226, Sep. 1998.

[27] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error measurement to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

Ying Yang received the B.E. degree in information security with the School of Computer and Communication Hunan University, Hunan, China, in 2006, where he is currently pursuing a graduate degree.

His research interests include digital signal processing, data hiding, watermarking, steganography, and sensor network security.

Xingming Sun (SM'07) received the B.S. degree in mathematics from Hunan Normal University, Hunan, China, in 1984, the M.S. degree in computing science from Dalian University of Science and Technology, Dalian, Liaoning, China, in 1988, and the Ph.D. degree in computing science from Fudan University, Shanghai, China, in 2001.

He is currently a Professor in the School of Computer and Communication, Hunan University, Hunan, China. His research interests include network and information security, digital watermarking, database security, and natural language processing.

Hengfu Yang received the B.S. degree in management information system from Xiangtan University, Xiangtan, Hunan, China, in 1996, and the M.S. degree in computer application from Guizhou University, Guiyang, Guizhou, China, in 2003. He is currently pursuing the Ph.D. degree in computer application at the School of Computer and Communication, Hunan University, Hunan, China.

His research interests include information hiding and image processing.

Chang-Tsun Li received the B.S. degree in electrical engineering from Chung-Cheng Institute of Technology (CCIT), National Defense University, Taiwan, in 1987, the M.S. degree in computer science from the U.S. Naval Postgraduate School, Monterey, CA, in 1992, and the Ph.D. degree in computer science, University of Warwick, Coventry, U.K., in 1998.

He was an Associate Professor from 1999 to 2002 in the Department of Electrical Engineering at CCIT and a visiting professor in the Department of Computer Science at the U.S. Naval Postgraduate School in the second half of 2001. He is currently an Associate Professor in the Department of Computer Science at the University of Warwick, Coventry, U.K. His research interests include image processing, pattern recognition, computer vision, multimedia security, and content based image retrieval.

Rong Xiao is currently pursuing the M.E. degree in computer science and technology in the School of Computer and Communication, Hunan University, Hunan, China.

Her main research interests include watermarking, steganography, and security issues of wireless ad hoc networks.