PAPER Special Section on Enriched Multimedia —New Technology Trends in Creation, Utilization and Protection of Multimedia Information—

Image Watermarking Method Satisfying IHC by Using PEG LDPC Code*

Nobuhiro HIRATA[†], ***Student Member***, **Takayuki NOZAKI**[†], ***Member***, ***and*** **Masaki KAWAMURA**[†a)], ***Senior Member***

SUMMARY We propose a digital image watermarking method satisfying information hiding criteria (IHC) for robustness against JPEG compression, cropping, scaling, and rotation. When a stego-image is cropped, the marking positions of watermarks are unclear. To detect the position in a cropped stego-image, a marker or synchronization code is embedded with the watermarks in a lattice pattern. Attacks by JPEG compression, scaling, and rotation cause errors in extracted watermarks. Against such errors, the same watermarks are repeatedly embedded in several areas. The number of errors in the extracted watermarks can be reduced by using a weighted majority voting (WMV) algorithm. To correct residual errors in output of the WMV algorithm, we use a high-performance error-correcting code: a low-density parity-check (LDPC) code constructed by progressive edge-growth (PEG). In computer simulations using the IHC ver. 4 the proposed method could a bit error rate of 0, the average PSNR was 41.136 dB, and the computational time for synchronization recovery was less than 10 seconds. The proposed method can thus provide high image quality and fast synchronization recovery.

key words: *digital watermarking, LDPC code, progressive edge-growth, information hiding criteria*

1. Introduction

Digital watermarking is an information hiding technique that invisibly embeds a message (e.g., a copyright notice or ID) in digital content (e.g., image, video, or audio data) to prevent the content from being copied illegally. In this paper we focus on methods for watermarking still images.

An image in which a watermark is embedded is called a stego-image, and the watermark extracted from a stego-image that has been subjected to image processing (e.g., lossy compression, cropping, scaling, or rotation) may differ from the one that was embedded. Image processing is therefore regarded as an *attack* on the stego-image. The purpose of the research presented here was to develop a watermarking method that is robust against such attacks and keeps image quality high.

StirMark [1]–[3] is a tool for attacking stego-images, and some researchers have used it to evaluate their watermarking methods [4]–[7]. The methods are hard to compare, however, because the evaluation conditions (e.g., image size, message length, and type of attack) differed between the research groups. An IEICE therefore proposed evaluation standards, called information hiding criteria (IHC) [8], for information hiding and watermarking methods. In the IHC ver. 4 the stego-images are attacked by JPEG compression, cropping, scaling, and rotation, and then peak signal-to-noise ratio (PSNR) and mean structural similarity (MSSIM) are used to measure their quality [9]. The IHC for still images promotes developing technologies of robust watermarking with large payload (capacity). It requires that a 200-bit message (i.e., 25 alphabets) can be decoded from a 1920 × 1080-pixel region cropped from a stego-image. Therefore, the methods satisfying IHC ver. 4 can be applied to copyright protection and content management. Since the payload is almost equivalent to one of QR code version 2 with error correction level M, the IHC defines enough payload for several applications.

If we want robust watermarks in stego-images, we need to embed them in an appropriate domain. Lossy compression methods strongly compress the high-frequency domain of the two-dimensional discrete cosine transform (2D DCT) or two-dimensional discrete wavelet transform (2D DWT), so to ensure that a watermark that will withstand lossy compression attack we ordinarily embed it in the low- or middle-frequency domain of the Y component of the original image [4]–[7], [10]–[18].

Even if the watermark is embedded in the appropriate domain, the extracted watermark may differ from the embedded watermark because of attacks on the stego-image. In other words, the attacks may cause errors in the watermark. Digital watermarking methods avoid these errors by using spread-spectrum techniques or correct them by using error- correcting codes [4]–[6], [10]–[13]. A watermarking method that uses an error-correcting code embeds as a watermark the codeword generated from a message.

The marking position becomes unclear when a stego-image is cropped, so a marker or synchronization code used to detect the marking position is also embedded in the stego-image [12]–[14]. Since there is no information about the marking position available, the position can only be searched for by brute force. Moreover, some watermarks are repeatedly embedded throughout the image so that at least one can be extracted from anywhere in it.

Currently there are two methods satisfying IHC ver. 4, one proposed by Ogawa *et al.* [14] and the other proposed by Hirata and Kawamura [13]. The former encodes the mes-

Manuscript received March 26, 2016.
Manuscript revised August 2, 2016.
Manuscript publicized October 7, 2016.

[†]The authors are with the Graduate School of Science and Engineering, Yamaguchi University, Yamaguchi-shi, 753–8512 Japan.

*The materials of this paper were partly presented in IWDW2015 [13].

a) E-mail: m.kawamura@m.ieice.org

DOI: 10.1587/transinf.2016MUP0003

sages by using a convolutional code and embeds the watermarks in 256 × 256-pixel blocks in the 2D DCT domain. Synchronization code used to detect the marking position is also embedded with the watermarks. With this method, the average PSNR of the stego-images is over 40 dB, i.e., the image quality is high. However, no particular technique for robustness against scaling and rotation is introduced, since inverse transform of scaling and rotation can be applied. Moreover, because the block size for the 2D DCT is large, the computational time needed for the synchronization recovery is more than 20 minutes.

The method proposed by Hirata and Kawamura [13] encodes the messages by using a concatenated code [19], [20] combining BCH and low-density parity-check (LDPC) codes, and it embeds the watermarks in 8 × 8 blocks (i.e., 8 pixels by 8 pixels) in the 2D DCT domain. The synchronization code used for synchronization recovery is also embedded in the image. The watermark is embedded in the minified original image for robustness against scaling and rotation. The computational time for synchronization recovery is only about 10 seconds because the block size for the 2D DCT is small, but the average PSNR of the stego-images is under 40 dB, i.e., the image quality is lower than that obtained using the method proposed by Ogawa *et al.* [14].

In this paper, we propose a fast-extraction watermarking method that yields high-quality stego-image satisfying IHC ver. 4. More precisely, the proposed method is able to extract a watermark at the same computational cost as that of Hirata and Kawamura's method [13] and yields stego-images of higher quality than those obtained using the method proposed by Ogawa *et al.* [14] under the condition of IHC ver. 4. Hirata and Kawamura's method reduces the errors in the messages by using two techniques; (1) embedding a watermark in a minified image for robustness against scaling and rotation attacks, and (2) error correction by both a weighted majority voting (WMV) algorithm and a concatenated code. Embedding a watermark in a minified image, however, degrades the quality of the stego-images. We therefore should embed a watermark in an original (not minified) image.

Since scaling and cropping may cause many errors to occur, the error-correcting capability should be enhanced by embedding the same watermarks in as many positions as possible and using them for WMV. This, though, may degrade image quality. The image quality also gets worse if we use an error-correcting code with long codelength. We should therefore use a high-performance error-correcting code in the watermarking method.

A low-density parity-check (LDPC) code [21] is an error-correcting code defined by a sparse parity check matrix or a sparse bipartite graph called a Tanner graph. The LDPC codes with the sum-product decoding algorithm have good decoding performance [22]. Short cycles in the Tanner graph degrade the decoding performance of an LDPC code [23], so Hu *et al.* [24] developed an algorithm, the progressive edge-growth (PEG) algorithm, constructing Tanner graphs that do not contain short cycles. We are therefore able to obtain a high-performance LDPC code by using the PEG algorithm. Although several improved PEG algorithms have been proposed [25], [26], in the work presented here we used an LDPC code constructed by the PEG algorithm as a first step.

In summary, the proposed watermarking method can quickly synchronize the marking position where the watermark is embedded and can produce high-quality stego-images. The method also satisfies IHC ver. 4. The main ideas of the proposed method are as follows:

(1) A marker is embedded in small blocks for extraction as fast as that in the method proposed by Hirata and Kawamura [13].
(2) Tolerance against scaling and rotation is obtained by increasing of number of the same watermarks.
(3) The codeword length is reduced by using the LDPC code generated by the PEG.

In computer simulations the proposed method achieved a PSNR of about 41.1 dB and the computational time for synchronization recovery was less than 10 seconds. We thus confirmed that our method performs better than that of Ogawa *et al.* [14] in terms of image quality and performs as well as that of Hirata and Kawamura [13] in terms of the computational time.

The rest of this paper is organized as follows. Section 2 describes the preliminary knowledge of the paper. Section 3 provides an error-correcting code suitable for our watermarking method. Section 4 presents our digital watermarking method satisfying IHC ver. 4 with high-quality stego-image. Computer simulations in Sect. 5 show that the our method outperforms the method proposed by Ogawa *et al.* in terms of image quality and synchronization time. Section 6 concludes the paper.

2. Preliminaries

This section describes the watermarking method based on quantization index modulation (QIM) [27], describes IHC ver. 4 [8], and describes error-correcting codes.

2.1 Watermarking Method Based on QIM Using DCT Domain

For robustness against JPEG compression, watermarks are embedded in the DCT domain of the Y component of the original image. The original image is divided into 8 × 8-pixel blocks, which are compatible with DCT blocks. Each block is transformed by the 2D DCT. Let us assume that the location (i, j) in the DCT domain is selected as the location where each bit of watermarks is embedded.

The watermarks are embedded by using QIM [27], and then they are extracted without the original image. Both the location (i, j) and the quantization step size Δ are shared by the encoder and decoder. When one bit of the watermark, $w \in \{0, 1\}$, is embedded, the value of the DCT coefficient at the location (i, j), C_{ij}, will be changed to C'_{ij} as follows:

Fig. 1 IHC standard images

$$C'_{ij} = 2\Delta\left(\left\lfloor \frac{C_{ij}}{2\Delta} - \frac{w}{2} + 0.5 \right\rfloor + \frac{w}{2}\right), \tag{1}$$

where $\lfloor x \rfloor$ is the floor function defined by

$$\lfloor x \rfloor = \max\{y \in \mathbb{Z} \mid y \le x\} \tag{2}$$

and $\mathbb{Z}$ is the set of integers.

In the extracting phase, we also use the coefficient $\widehat{C}_{ij}$ at the same location (i, j). The extracted watermark $\hat{w}$ is given by

$$\hat{w} = \left\lfloor \frac{|\widehat{C}_{ij}|}{\Delta} + 0.5 \right\rfloor \mod 2. \tag{3}$$

Note that the extracted watermark $\hat{w}$ is equal to the original watermark w when $C'_{ij} - \frac{\Delta}{2} \le |\widehat{C}_{ij}| < C'_{ij} + \frac{\Delta}{2}$ holds. We select the location $(1, 1)$ in the DCT domain.

2.2 Information Hiding Criteria

In this section we briefly introduce IHC ver. 4 [8]. The details are in [8]. The IHC committee [8] defines evaluation criteria for image watermarking. Figure 1 shows the six original IHC standard images. The original images are provided in YUV422 format with a 4608×3456-pixel size. Ten messages are generated by using M-sequence. The message length is 200 bits. A stego-image is constructed by embedding a watermark, which may be an encoded message. The stego-image will be attacked by JPEG compression, cropping, scaling, and rotation. Figure 2 shows the attack model.

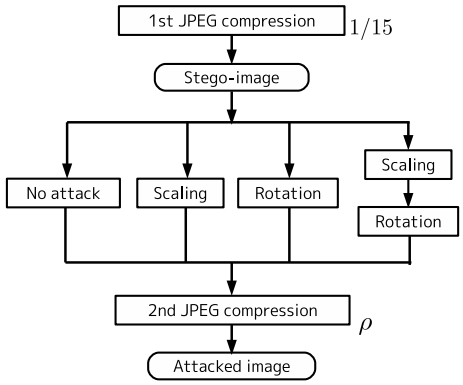

Fig. 2 Attack model

The details of the attack conditions are given as follows:

- After the first JPEG compression, the file size should be less than $1/15$ the original size. After the second compression, the file size should be less than ρ the original size, where ρ is second compression ratio.
- Scaling ratios are s (%) are $\{70, 90, 110, 130\}$.
- Degrees of angular rotation θ (°) are $\{3, 6, 9, 12\}$.
- Their combinations are $(s, \theta) = \{(90, 3), (90, 9), (110, 3), (110, 9)\}$.

The parameters s and θ are known to the decoder. The attacked images can be re-transformed to the original size and direction by using the parameters s and θ. Ten rectangular regions are cropped from each normalized image. The size of each region is 1920×1080-pixels, and the coordinates of the regions are given by IHC. After cropping, a watermark is extracted from each region. The watermarking method is evaluated with regard to both the quality of the stego-image and the accuracy of the message. The image quality is measured by peak signal-to-noise ratio (PSNR) and mean structural similarity (MSSIM) [9]. The message accuracy is measured by the bit error rate (BER), which is defined for a given message $\boldsymbol{\xi} = (\xi_1, \xi_2, \cdots, \xi_K)^\top \in \{0, 1\}^K$ and an estimated message $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_K)^\top \in \{0, 1\}^K$ by the equation

$$\mathrm{BER} = \frac{1}{K}\sum_{i=1}^{K} \xi_i \oplus \sigma_i, \tag{4}$$

where $\oplus$ stands for exclusive OR.

There are two categories of evaluation criteria for comparing methods: "Highest Image Quality" and "Highest Tolerance." The conditions of the former are given as follows:

- The worst BER should be less than or equal to 2% and the average BER should be lower than 1%.
- After the first compression, the file size of a stego-image should be less than $1/15$ the size of a YUV original image. After the second compression, the file size should be less than $\rho = 1/25$.

The method giving the higher PSNR under these conditions is superior in terms of image quality. The conditions of the

latter are given as follows:

- The BERs for all estimated messages should be equal to 0.
- The PSNR of a stego-image should be over than 30 dB.

The method giving the smaller compression ratio ρ in the second JPEG compression is superior in terms of tolerance.

2.3 Error-Correcting Codes

Even if a watermark is embedded in the appropriate domain, attacks on the stego-image may cause the extracted watermark to differ from the embedded watermark. In other words, attacks may cause errors in the watermark. To correct them, we use an error-correcting code.

In a coding system a message of length K is encoded to a codeword of length N, where $N > K$. The codeword is transmitted through a communication channel that causes some errors in the codeword. The receiver estimates the transmitted codeword from the received word by using a decoder. In a watermarking system the codewords correspond to the watermarks and the communication channel corresponds to the attacks on the stego-image. We refer to the fraction of errors in the watermark as the *channel error rate*. Note that the channel error rate depends on types of images and attack models.

If the length of watermark becomes short, the quality of the stego-image may be improved. Hence, to propose a watermarking method with high image-quality, we need to construct an error-correcting code with short codelength for a fixed message length and fixed channel error rate.

2.3.1 Linear Code and Tanner Graph

A binary (N, K) linear code is defined by a binary $(N-K)\times N$ parity check matrix $\mathbf{H}$. The encoder of the linear code maps a given message $\boldsymbol{\xi} \in \{0,1\}^K$ to the codeword $\boldsymbol{c} = (\boldsymbol{\xi} \mid \boldsymbol{p})^\top \in \{0,1\}^N$ as $\mathbf{H}\boldsymbol{c} = \mathbf{0}$. We denote the i-th component of $\boldsymbol{c}$ by c_i.

Each parity check matrix is described by a bipartite graph called a *Tanner graph*. The corresponding Tanner graph for a given $(N-K)\times N$ parity check matrix is described as follows. There are N symbol nodes and $(N-K)$ check nodes in the Tanner graph. The i-th check node c_i and j-th symbol node s_j are connected by an edge if the (i, j)-th entry of the parity check matrix H_{ij} is equal to 1. Note that, conversely, a parity check matrix is constructed from a Tanner graph. Figure 3 shows an example of a Tanner graph. The circles and squares in Fig. 3 represent the symbol nodes and check nodes, respectively.

2.3.2 Low-Density Parity-Check Code

An LDPC code is a linear code defined by a *sparse* parity check matrix or a sparse Tanner graph. It is well known that the LDPC codes are efficiently decoded by the sum-product algorithm [22]. An LDPC code is called (γ, δ)-regular if all the symbol nodes are of degree γ and all the check nodes

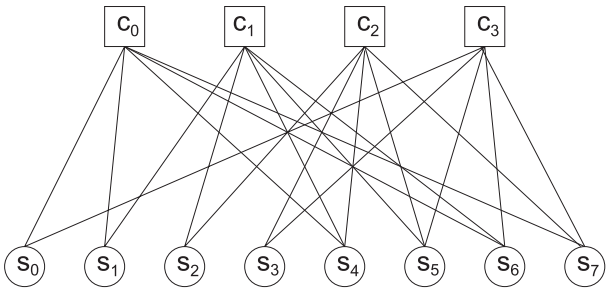

Fig. 3 Example of a Tanner graph

are of degree δ. The codelength N for a (γ, δ)-regular LDPC code with message length K is $\delta K/(\delta-\gamma)$.

The decoding performance of an LDPC code depends on the degree of the nodes and the connection of the edges in the Tanner graph. Roughly speaking, the decoder is able to estimate the correct codeword with high probability if the channel error rate is less than a certain parameter ϵ^*. It is known that ϵ^* mainly depends on the degrees of the nodes in the Tanner graph. On the other hand, even if the channel error rate is less than ϵ^*, decoding errors occur with low probability. The decoding errors occurring when the channel error rate is small are mainly caused by the short cycles in the Tanner graph [23]. Thus, optimizing the degree of nodes and avoiding short cycles in the Tanner graph are important when one is constructing a high-performance LDPC code.

The length of the shortest cycle in the Tanner graph is called the graph's *girth*. The progressive edge-growth (PEG) algorithm [24] constructs a Tanner graph with a large girth. In other words, the PEG is an algorithm constructing a high-performance LDPC code. The details of the PEG algorithm are given in [24, Sect.III] . To simplify the notation in this paper, we refer to an LDPC code constructed by the PEG algorithm as a PEG LDPC code.

3. Error-Correcting Code for Watermarking

In this section we describe a PEG LDPC code for our watermarking method satisfying IHC ver. 4 and we compare its decoding performance with that of the concatenated code used in Hirata and Kawamura's watermarking method [13].

3.1 Construction of PEG LDPC Code

In IHC ver. 4 the message length K is given as 200 bits. The message should be decoded with a very low error rate under the attacks specified in IHC ver. 4, so we use a $(3, 4)$-regular LDPC code. Since $K = 200, \gamma = 3$ and $\delta = 4$, the codelength N is 800. To avoid short cycles in the Tanner graph, we use the PEG algorithm. Therefore the watermarking method proposed in this paper uses a $(800, 200)$ PEG LDPC code as the error-correcting code.

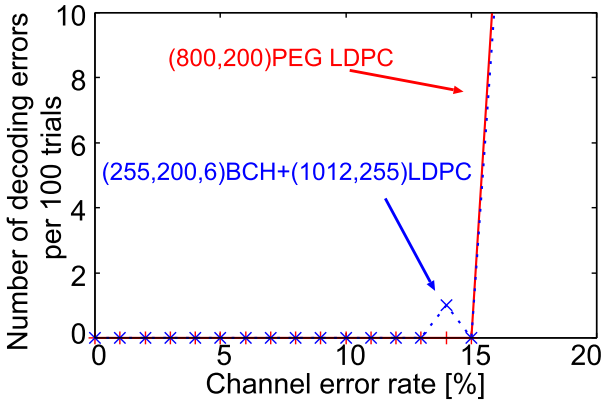

Fig. 4 Performance of concatenated code and PEG LDPC code

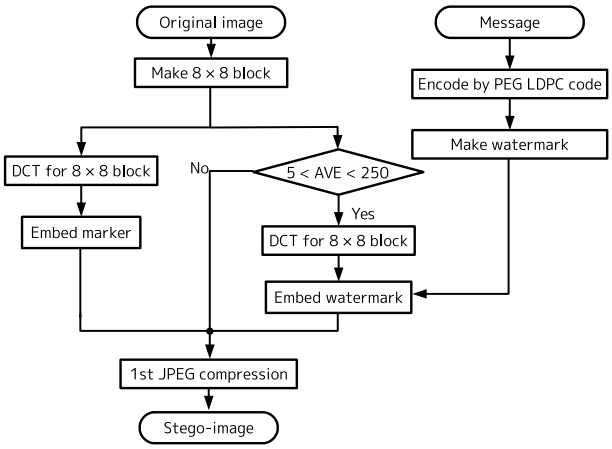

Fig. 5 Encoding and embedding processes in the proposed method. A message is encoded by PEG LDPC code. Markers and watermarks are embedded in the DCT domain. AVE: average of pixel values.

3.2 Performance Comparison

Figure 4 compares the decoding performance of the $(800, 200)$ PEG LDPC code described in Sect. 3.1 with that of the concatenated code used in [13]. The horizontal axis represents the channel error rate, and the vertical axis represents the frequency of decoding errors (i.e, the number of times in 100 trials that the estimated codeword was not the same as the transmitted codeword). The red solid line gives the decoding performance of the $(800, 200)$ PEG LDPC code. As shown in Fig. 4, there are no decoding errors if the channel error rate is less than or equal to 15%. The blue broken line shows the decoding performance of the concatenated code used in [13]. That concatenated code combined a $(255, 200, 6)$ BCH code as the outer code and a $(1012, 255)$ LDPC code as the inner code. Hirata and Kawamura constructed the LDPC code in [13] by connecting the edges in the Tanner graph randomly. As a result, the Tanner graph contains many short cycles and the decoding performance is degraded. Hence, as shown in Fig. 4, a decoding error occurred when $\epsilon = 14\%$.

As discussed in Sect. 2.3, the quality of the stego-image can be improved by reducing the codelength. From the above, the $(800, 200)$ PEG LDPC code has shorter codelength than the concatenated code used in [13] and outperforms this concatenated code. Hence, we conclude that the PEG LDPC code is suitable for a digital watermarking system.

4. Proposed Watermarking Method

In this section we describe a watermarking method using the PEG LDPC code. In the proposed method, watermarks are embedded into the DCT domain by QIM. As will be shown in Sect. 5 our method can satisfy IHC ver. 4. We describe the encoding and embedding processes in Sect. 4.1 and describe the extraction and decoding processes in Sect. 4.2.

4.1 Encoding-and-Embedding Algorithm

Figure 5 shows the encoding and embedding processes. Here we explain how to make a watermark. The watermark consists of an encoded message and check bits. The check bits are used for measuring the frequency of errors, which we reduce by using weighted majority voting (WMV). The layout of watermarked and marker areas in a segment is described in Sect. 4.1.2. The assignment of the watermarked area is important to robustness against cropping. A synchronization code or marker used to detect the marking position is also embedded with a watermark. Since there are some areas inappropriate areas for embedding, an exception process is described in Sect. 4.1.3.

4.1.1 Encoding Process

The PEG LDPC code is used to encode a K-bit message $\boldsymbol{\xi}$ to an N-bit codeword

$$\boldsymbol{c} = (c_1, c_2, \cdots, c_N)^\top \in \{0, 1\}^N. \tag{5}$$

The codeword should be able to be extracted from a stego-image that has been cropped, so the same encoded messages (i.e., codewords) are repeatedly embedded throughout an image. We can obtain a lot of extracted codewords, but there are errors in them. To measure the channel error rate, we introduce check bits. A watermark consists of the codeword $\boldsymbol{c}$ and check bits $\boldsymbol{s}$.

Because scaling and rotation attacks can cause many errors, we use WMV to reduce the errors in the watermarked areas. The check bits $\boldsymbol{s}$ are used for measuring the frequency of errors in each watermarked area. The check bits are given by

$$\boldsymbol{s} = (0, 1, 0, 1, \cdots, 0, 1)^\top. \tag{6}$$

The length of the check bits is B bit. Any bit sequence can be used. In this paper we a sequence consisting of 0s and 1s alternately. The watermark $\boldsymbol{w}$ consists of the codeword $\boldsymbol{c}$ and the check bits $\boldsymbol{s}$, and is given by

$$\boldsymbol{w} = \left(s_1, c_1, \cdots, c_{\frac{N}{B}}, s_2, c_{\frac{N}{B}+1}, \cdots, c_{(B-1)\frac{N}{B}}, s_B, \cdots, c_N\right)^\top. \tag{7}$$

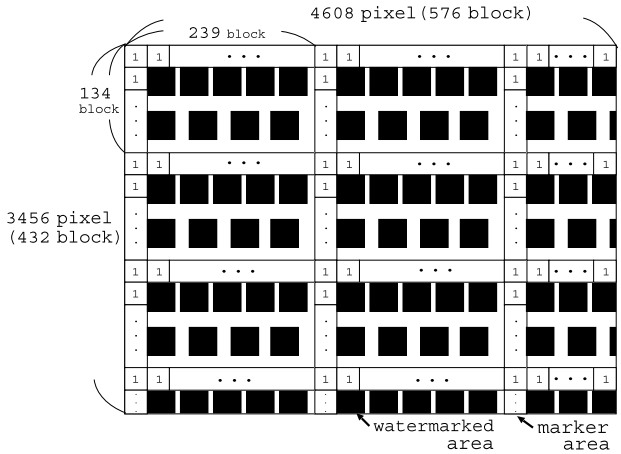

Fig. 6 Layout of watermarked and marker areas. Each segment consists of 239 × 134 blocks. There are nine watermarked areas (black) in each segment. Top and right-side blocks in a segment are marker areas. The same watermark is embedded in all watermarked areas.

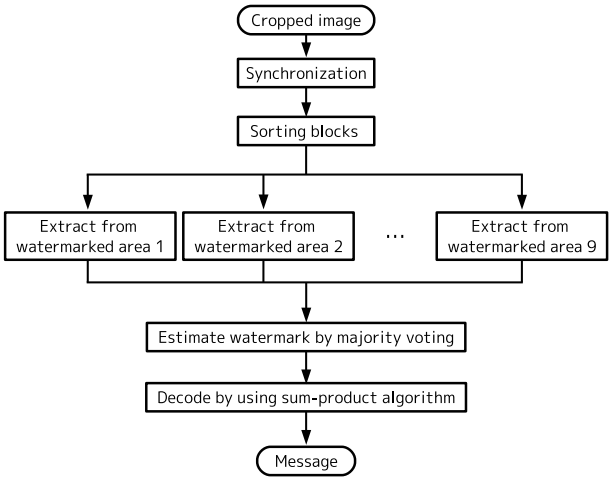

Fig. 7 Extraction and decoding processes in the proposed method. The marking position can be found by extracting marker candidates. After synchronization, watermarks can be extracted from nine watermarked areas. An estimated watermark is obtained from nine watermarks by WMV, and then an estimated message is decoded from the watermark.

The check bits are inserted among codewords $\boldsymbol{c}$ at regular intervals. The decoding process using WMV is explained in Sect. 4.2.2.

4.1.2 Embedding Process

As explained in Sect. 2.2, the watermarks should be able to be extracted from a 1920 × 1080 region. We assume that cropping is performed by a pixel unit. Since the marking position is unknown in the decoder, a synchronization code or marker is embedded so the position can be detected.

The Y component of an original image is divided into 239 × 134-block segments. Each segment is divided into 8 × 8-pixel blocks (DCT blocks). Each block is transformed by using the 2D DCT. Figure 6 shows the layout of watermarked and marker areas in an image. In the marker area, the marker is embedded for each $1920/8 - 1 = 239$ (resp. $1080/8 - 1 = 134$) blocks in column (resp. row) in a lattice pattern. The same watermarks are embedded in the watermarked areas shown as black squares in the segment. Recall that there are B-bit check bits and the codelength is N bits. Each watermarked area is a square ℓ blocks on a side, where

$$\ell = \left\lceil \sqrt{B+N} \right\rceil, \tag{8}$$

where $\lceil x \rceil$ stands for the ceiling function, which returns the smallest integer greater than x. We focus on 238 × 66 block areas which are half the size of 238 × 133 block areas (except for the marker area) in the segment. The number of the watermarked areas is determined by two requirements. One is that errors should be small. That is, more watermarked areas are better. The other is that image quality should be good. That is, fewer areas are better.

When we applied our method to the IHC, we found that nine watermarked areas was best under those conditions. Therefore, in the upper part in a watermarked area, five same watermarks are embedded at intervals of $(238-5\ell)/5$ blocks. In the lower part, four same watermarks are embedded at intervals of $(238 - 4\ell)/4$ blocks.

Each of the watermark and marker bits is embedded at a fixed location $(1, 1)$ in a DCT block in the watermarked and marker areas. Since JPEG compression mainly compresses high-frequency components, the watermarks and marker are embedded into low-frequency components by using QIM as explained in Sect. 2. Note that all marker values are 1.

4.1.3 Exception Process at Embedding

When the pixel value is near 255 (resp. 0), the value after embedding watermarks might be over 255 (resp. under 0). In this case, not only the watermark be incorrectly extracted but also image quality is degraded due to overflow. Therefore, we introduce the exception process for the following condition. The exception process is applied in the embedding of watermarks, not markers. Let a pixel value at (x, y) in an $L \times L$-pixel block be P_{xy} for $x, y \in \{1, 2, \cdots, L\}$. The average of pixel values is given by

$$\mathrm{AVE} = \frac{1}{L^2} \sum_{x=1}^{L} \sum_{y=1}^{L} P_{xy}. \tag{9}$$

Watermarks are embedded in an area only If $5 < \mathrm{AVE} < 250$. The length L might be set in multiples of 8, since it would be compatible with a DCT block. In this paper we set $L = 40$.

4.2 Extraction and Decoding Algorithm

Figure 7 shows the extraction and decoding processes. An estimated watermark is extracted from the cropped image. In Sect. 4.2.1, we explain synchronization recovery. Firstly, marking position is searched by extracting marker candidates. In Sect. 4.2.2, we explain the WMV and error-

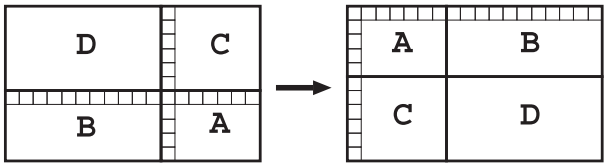

Fig. 8 Sorting blocks. Blocks in a segment are swapped in a manner such that marker row and column are arranged top and left, respectively.

correcting process after the synchronization recovery.

4.2.1 Synchronization Recovery

We want to extract watermarks from a 1920 × 1080-pixel region cropped from a stego-image. The marking position in the region is unclear. Therefore, all the possible blocks are transformed by using the 2D DCT, and then marker candidates are extracted by using QIM. Since the values of the marker are 1, the position which gives the largest summation of the marker candidates in rows and columns can be estimated as the marking position.

The synchronization recovery is performed as follows. Now we assume that the region is divided into 8 × 8-pixel blocks from a position (x, y), the number of blocks in a column, b_c, is 240 and the number of blocks in a row, b_r, is 135. From the (i, j)-th block we extract marker candidate $V_{xy}(i, j)$ by using QIM. Summation at the r-th row and c-th column, $S_{xy}(r, c)$, is given by

$$S_{xy}(r,c) = \sum_{i=1}^{b_c} V_{xy}(r,i) + \sum_{j=1}^{b_r} V_{xy}(j,c). \tag{10}$$

The row and column (r, c) that give the largest summation $S_{xy}(r, c)$ are estimated as a position of the marker candidate. The candidate position is given by

$$(\hat{r},\hat{c}) = \underset{1\le r\le b_r,\ 1\le c\le b_c}{\arg\max} S_{xy}(r,c). \tag{11}$$

Since there are $8 \times 8 = 64$ candidate marking positions, the most likely position $(\hat{x}, \hat{y})$ is given by

$$(\hat{x},\hat{y}) = \underset{1\le x,y\le 8}{\arg\max} S_{xy}(\hat{r},\hat{c}). \tag{12}$$

After we could find the marking position, the blocks in the segment are swapped, as shown in Fig. 8, in a manner such that the marker row and column are arranged top and left, respectively. When the cropped image is larger than the 1920 × 1080-pixel region defined in IHC, more than one marking positions might be found. In this case, one can be selected and the others ignored. After the synchronization recovery, watermarks are extracted from the watermarked areas by QIM.

4.2.2 Decoding Using Weighted Majority Voting

There are nine watermarked areas in a segment. The extracted watermarks $\hat{\boldsymbol{w}}^\mu, \mu = 1, 2, \cdots, 9$ consist of extracted check bits $\hat{\boldsymbol{s}}^\mu$ and received words $\boldsymbol{y}^\mu$. Errors in the received words can be reduced by using WMV. Here, in order to apply the WMV, all the binary values $\{0, 1\}$ of check bits and received words are converted to $\{+1, -1\}$. By abuse of notation, we also denote the converted check bits and received words for $\{+1, -1\}$ as $\hat{\boldsymbol{s}}^\mu$ and $\boldsymbol{y}^\mu$, respectively.

The reliability of the received words may differ from area to area, so we measure the reliability by using the check bits. The μ-th reliability α_μ for the extracted check bits $\hat{\boldsymbol{s}}^\mu \in \{+1, -1\}^B$ is given by

$$\alpha_\mu = \frac{1}{B}\sum_{j=1}^{B} s_j \hat{s}_j^\mu, \quad \mu = 1, 2, \cdots, 9, \tag{13}$$

where $s_j \in \{+1, -1\}$ is the check bit in (6). If the reliability is very small, we assume that there are many errors in the received words. Therefore when $\alpha_\mu \le 0.4$, we reset it to $\alpha_\mu = 0$. Moreover, when there are many non-embedded blocks, explained in Sect. 4.1.3, the reliability for an area might be small. That area would therefore be ignored automatically.

Using the reliability α_μ, the estimated word $\hat{\boldsymbol{y}}$ is obtained from the nine received words $\boldsymbol{y}^\mu$ by WMV as follows:

$$\hat{y}_k = \mathrm{sgn}\left(\sum_{\mu=1}^{9} \alpha_\mu y_k^\mu\right), \tag{14}$$

where the function $\mathrm{sgn}(x)$ is defined by

$$\mathrm{sgn}(x) = \begin{cases} +1, & x \ge 0, \\ -1, & x < 0. \end{cases} \tag{15}$$

The estimated message $\boldsymbol{\sigma}$ is decoded from the estimated word $\hat{\boldsymbol{y}}$ by using the sum-product algorithm [22].

5. Computer Simulations

We evaluated our proposed method with computer simulations in accordance with IHC [8].

5.1 Simulation Conditions

The simulation parameters are as follows;

- 200-bit messages are encoded to 800-bit codewords using the PEG LDPC code.
- The number B of check bits used in the WMV is 25, so the watermark length is $B + N = 825$ bits.
- Nine same watermarks are embedded in a segment.
- The step size Δ used in the QIM is 50.

5.2 Results of the Highest Image Quality

Table 1 shows the average compression ratio, PSNR, and MSSIM for IHC "highest image-quality." Since ten different messages were embedded, the values in Table 1 are average values for ten trials. The compression ratio could be under $1/15 = 6.67\%$ for the first coding (compression) and $1/25 =$

Table 1 Average compression ratio, PSNR, and MSSIM for the Highest Image Quality by the proposed method

	Compression ratio [%]		PSNR of Y channel [dB]		MSSIM of Y channel	
	1st coding	2nd coding	1st coding	2nd coding	1st coding	2nd coding
Image 1	6.626	3.958	41.910	40.805	0.981	0.976
Image 2	6.432	3.929	41.713	41.714	0.971	0.971
Image 3	6.531	3.941	41.400	41.151	0.967	0.965
Image 4	6.510	3.966	41.368	41.476	0.962	0.963
Image 5	6.615	3.974	41.274	41.004	0.963	0.961
Image 6	6.582	3.976	41.683	40.666	0.975	0.970
Average	6.550	3.957	41.558	41.136	0.970	0.968

Table 2 Average compression ratio, PSNR, and MSSIM for the Highest Image Quality by the method proposed by Ogawa *et al.* [14]

	Compression ratio [%]		PSNR of Y channel [dB]		MSSIM of Y channel	
	1st coding	2nd coding	1st coding	2nd coding	1st coding	2nd coding
Image 1	6.467	3.455	40.796	39.371	0.997	0.994
Image 2	6.716	3.647	42.055	40.916	0.993	0.995
Image 3	6.608	3.456	42.732	41.164	0.996	0.993
Image 4	6.419	3.174	43.055	41.666	0.994	0.989
Image 5	6.546	3.327	42.649	41.203	0.997	0.995
Image 6	6.513	3.569	41.527	39.568	0.997	0.996
Average	6.545	3.438	42.195	40.648	0.996	0.994

Table 3 Average compression ratio, PSNR, and MSSIM for the Highest Tolerance by the proposed method

	Compression ratio [%]		PSNR of Y channel [dB]		MSSIM of Y channel	
	1st coding	2nd coding	1st coding	2nd coding	1st coding	2nd coding
Image 1	6.626	3.252	41.910	41.141	0.981	0.978
Image 2	6.432	3.289	41.713	41.429	0.971	0.969
Image 3	6.531	3.324	41.400	40.691	0.967	0.962
Image 4	6.510	2.443	41.368	41.672	0.962	0.964
Image 5	6.615	2.474	41.274	39.741	0.963	0.951
Image 6	6.582	3.976	41.683	40.666	0.975	0.970
Average	6.550	3.126	41.558	40.890	0.970	0.966

4.0% for the second coding. For image quality, the average PSNR was 41.136 dB at the second coding. All PSNRs were over 30 dB at first coding. Table 2 shows the corresponding results for the method proposed by Ogawa *et al.* [14]. The average PSNR was 40.648 dB. Therefore, the average PSNR of the proposed method is about 0.5 dB higher than that of the method proposed by Ogawa *et al.* [14].

The proposed method could achieve a bit error rate (BER) of 0 for all attacks in six IHC standard images. Because there were ten different messages, ten different cropped regions, and thirteen kinds of attacks, BER was averaged over 1300 trials. The WMV and the PEG LDPC code worked well and could correct every error.

5.3 Results of the Highest Tolerance

The definition of the criteria for "highest tolerance" was described in Sect. 2.2. Under those criteria, the method that can achieve highest compression ratio at second coding is superior in terms of tolerance. Table 3 shows the average compression ratio and, image quality by PSNR and MSSIM for highest tolerance. Since ten different messages were embedded, the values in Table 3 are the average values over ten trials. The compression ratios for first coding could be under $1/15 = 6.67\%$. Those for second coding are important for highest compression ratio. The worst compression ratio was 3.976% (1/25) for image 6 and the best one was 2.443% (1/40) for image 4. Our method could achieve an average compression ratio of about 3.126% (1/32).

Table 4 displays the results for the method proposed by Ogawa *et al.* [14]. In each image the compression ratios for the method proposed here were worse than those for the method proposed by Ogawa *et al.* However, for image quality, PSNRs achieved by the method proposed here were almost 9.3 dB higher than those achieved by the method proposed by Ogawa *et al.*

5.4 Computational Costs

An evaluation criterion for computational cost has not been defined in IHC yet, but a low computational cost is important in some applications. In general, synchronization requires $O(n^2)$ searches, where n is the block size. The amount of DCT calculation for each search is $O(n \log n)$. The most

Table 4 Average compression ratio, PSNR, and MSSIM for the Highest Tolerance by the method proposed by Ogawa *et al.* [14]

	Compression ratio [%]		PSNR of Y channel [dB]		MSSIM of Y channel	
	1st coding	2nd coding	1st coding	2nd coding	1st coding	2nd coding
Image 1	6.467	0.917	41.144	30.516	0.967	0.928
Image 2	6.657	0.894	42.043	30.701	0.960	0.930
Image 3	6.611	0.956	42.709	32.111	0.953	0.920
Image 4	6.420	0.857	43.089	33.873	0.902	0.871
Image 5	6.548	0.891	42.644	30.590	0.959	0.940
Image 6	6.515	0.939	41.537	31.582	0.972	0.943
Average	6.536	0.909	42.195	31.562	0.952	0.922

Table 5 Average computational costs of synchronization recovery

	Computational cost [sec]	
	Proposed method	Ogawa *et al.* [14]
Image 1	6.97	1212.51
Image 2	6.96	1189.25
Image 3	7.12	1251.23
Image 4	7.08	1192.32
Image 5	6.98	1165.97
Image 6	7.03	1472.90
Average	7.02	1247.33

Table 6 Machine specifications

OS	Vine Linux 6.3
CPU	Intel Core i7 3930K 3.2GHz 6 core 12 thread
Memory	32 GB
OpenCV	OpenCV-2.4.10
Compiler	Intel C/C++ Compiler Version 15.0.0.090

time-consuming process is the synchronization recovery. In the proposed method, there are only 8×8 candidates ($n = 8$) for the marking position. In this paper we evaluated the synchronization time needed to find the marking position. Table 5 shows the computational cost for the synchronization recovery for each image. Machine specifications are listed in Table 6.

The average computational time for our proposed method was less than 10 seconds. That for the method proposed by Ogawa *et al.* [14] was about 1200 seconds, since the block size of Ogawa *et al.* is 256×256 pixels ($n = 256$), i.e., 65536 candidates. One sees from Table 5 that the method proposed here was 170 times faster than the method proposed by Ogawa *et al.*

Moreover, when we would focus on the computational cost for not only cropping but also scaling and rotation, the computational time difference between the proposed method here and the method proposed by Ogawa *et al.* [14] would be more significant.

6. Conclusion

The IHC specify a detailed scheme of evaluation for watermarking methods. They include cropping, JPEG compression, scaling, and rotation as attacks. These attacks damage watermarks embedded in the stego-images, so the watermarks may have a lot of errors. In watermarking methods, many techniques are used to reduce errors and correct messages without degrading image quality. The image quality and message accuracy are evaluated by PSNR and BER, respectively.

We proposed the method satisfying the IHC. The proposed method could generate high-quality stego-images. To encode messages robust against attacks, we used the PEG LDPC code, which is one of the effective error-correcting codes. This resulted in codewords shorter than those obtained when using Hirata and Kawamura's watermarking method [13], so higher image quality could be achieved.

We introduced many techniques for watermark robustness. Against JPEG compression, a watermark bit and a marker bit were embedded in a lower-frequency component in the DCT domain. Against cropping, same watermarks were repeatedly embedded throughout an image. In the extracting phase, the marking position was unclear. To detect the position, the synchronization code or marker was embedded with the codeword in a lattice pattern. Since a lot of watermarks could be extracted from an attacked image, WMV could be applied by using the check bits. The number of errors could be reduced a lot by using WMV, and residual errors could be corrected by the PEG LDPC code. Note that the attack parameters of scaling ratio and rotation angles are known in the IHC ver. 4. The synchronization code could also be used to detect the watermarked areas in images attacked with scaling and rotation procedures. If no attack parameters are known, we can perform a brute-force search. Although it spends much time searching for the areas, watermarks can be estimated by using our method. Our method depends on success in the synchronization recovery. We must further develop watermarking methods robust to harder attacks under more severe conditions.

Although there is no criterion for the computational cost, it is important to be able to detect and decode a message faster. The proposed method could detect the watermarked areas much faster than the other method [14]. Since the proposed method used 8×8-pixel blocks for embedding, the time needed to detect the marking position was very short.

As a result, with the proposed method the average PSNR was 41.136 dB and the average bit error rate of messages 0. Moreover, the average computational time for the proposed method was less than 10 seconds. Therefore, the proposed method could achieve the fastest synchronization time and the highest image quality.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 25330028 and JP16K00156. The computer simulations were carried out on PC clusters at Yamaguchi University.

References

[1] StirMark benchmark (Ver.4.0), http://www.petitcolas.net/fabien/watermarking/stirmark/, accessed March 25, 2016.
[2] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn, "Attacks on copyright marking systems," Information Hiding, Lecture Notes in Computer Science, vol.1525, pp.218–238, Springer-Verlag, 1998.
[3] F.A.P. Petitcolas, "Watermarking schemes evaluation," IEEE Signal Process. Mag., vol.17, no.5, pp.58–64, 2000.
[4] T. Yamamoto and M. Kawamura, "Method of spread spectrum watermarking using quantization index modulation for cropped images," IEICE Trans. Information and Systems, vol.E98-D, no.7, pp.1306–1315, 2015.
[5] K. Solanki, N. Jacobsen, U. Madhow, B.S. Manjunath, and S. Chandrasekaran, "Robust image-adaptive data hiding using erasure and error correction," IEEE Trans. Image Process., vol.13, no.12, pp.1627–1639, 2004.
[6] X. Kang, J. Huang, Y.Q. Shi, and Y. Lin, "A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.8, pp.776–786, 2003.
[7] M. Cancellaro, F. Battisti, M. Carli, G. Boato, F.G.B. De Natale, and A. Neri, "A commutative digital image watermarking and encryption method in the tree structured Haar transform domain," Signal Processing: Image Communication, vol.26, no.1, pp.1–12, 2011.
[8] Information hiding and its criteria for evaluation, IEICE, http://www.ieice.org/iss/emm/ihc/en/, accessed March 25, 2016.
[9] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Trans. Image Processing, vol.13, no.4, pp.600–612, 2004.
[10] I.J. Cox, J. Kilian, F.T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia,' IEEE Trans. Image Process., vol.6, no.12, pp.1673–1687, 1997.
[11] I. Usman and A. Khan, "BCH coding and intelligent watermark embedding: Employing both frequency and strength selection," Applied Soft Computing Journal, vol.10, no.1, pp.332–343, 2010.
[12] N. Hirata and M. Kawamura, "Digital watermarking method using LDPC code for clipped image," Proc. 1st Inter. Workshop on Information Hiding and its Criteria for Evaluation (IWIHC2014), Kyoto, pp.25–30, 2014.
[13] N. Hirata and M. Kawamura, "Watermarking method using concatenated code for scaling and rotation attacks," Proc. 14th Inter. Workshop on Digital-forensics and Watermarking (IWDW2015), LNCS, vol.9569, pp.259–270, Springer-Verlag, 2016.
[14] H. Ogawa, M. Kuribayashi, M. Iwata, and K. Kise, "DCT-OFDM based watermarking scheme robust against clipping, rotation, and scaling attacks," Proc. 14th Inter. Workshop on Digital-forensics and Watermarking (IWDW2015), LNCS, vol.9569, pp.271–284, Springer-Verlag, 2016.
[15] I. Nasir, F. Khelifi, J. Jiang, and S. Ipson, "Robust image watermarking via geometrically invariant feature points and image normalisation," IET Image Processing, vol.6, no.4, pp.354–363, 2012.
[16] Y. Jiang, Y. Zhang, W. Pei, and K. Wang, "Adaptive spread transform QIM watermarking algorithm based on improved perceptual models," AEU - International Journal of Electronics and Communications, vol.67, no.8, pp.690–696, 2013.
[17] L.P. Feng, L.B. Zheng, and P. Cao, "A DWT-DCT based blind watermarking algorithm for copyright protection," 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol.7, pp.455–458, 2010.
[18] N. Kashyap and G.R. Sinha, "Image watermarking using 3-level discrete wavelet transform (DWT)," International Journal of Modern Education and Computer Science, vol.4, no.3, pp.50–56, 2012.
[19] G.D. Forney, Jr., Concatenated Codes, MIT Press, Cambridge, MA, 1966.
[20] ETSI EN 302 755, V1.3.1, Digital Video Broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system, 2012.
[21] R.G. Gallager, "Low-density parity-check codes," IRE Trans. Information Theory, vol.8, no.1, pp.21–28, 1962.
[22] T. Wadayama, "A coded modulation scheme based on low density parity check codes," IEICE Trans. Fundamentals, vol.E84-A, no.10, pp.2523–2527, Oct. 2001.
[23] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inform. Theory, vol.45, no.2, pp.399–431, 1999.
[24] X.-Y. Hu, E. Eleftheriou, and D.M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," IEEE Trans. Inform. Theory, vol.51, no.1, pp.386–398, 2005.
[25] T. Tian, C.R. Jones, J.D. Villasenor, and R.D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," IEEE Trans. Commun., vol.52, no.8, pp.1242–1247, 2004.
[26] X. He, L. Zhou, J. Du, and Z. Shi, "The multi-step PEG and ACE constrained PEG algorithms can design the LDPC codes with better cycle-connectivity," IEEE International Symposium on Inform. Theory, pp.46–50, 2015.
[27] B. Chen and G.W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," IEEE Trans. Inform. Theory, vol.47, no.4, pp.1423–1443, 2001.

Nobuhiro Hirata received a B.S. degree from Yamaguchi University in 2014. Currently he is a master's degree student at Yamaguchi University's Graduate School of Science and Engineering. His research interests include digital watermarking. He is a student member of IEICE.

Takayuki Nozaki received B.E., M.E. and D.E. degrees from Tokyo Institute of Technology in 2008, 2010, and 2012. From 2010 to 2013 he was a Research Fellow of the Japan Society for the Promotion of Science. From 2013 to 2015 he was a Research Associate at Kanagawa University. Since 2015 he has been an Assistant Professor at Yamaguchi University. His research interests are codes on graph and iterative decoding algorithms. He is a member of IEEE, ACM, and IPSJ.

Masaki Kawamura received B.E., M.E., and Ph.D. degrees from the University of Tsukuba in 1994, 1996, and 1999, respectively. He joined Yamaguchi University as a research associate in 1999. Currently he is an associate professor there. His research interests include associative memory models and information hiding. He is a senior member of IEICE and a member of JNNS, JPS, and IEEE.