IEEE Open Journal of the Computer Society

Received 19 February 2020; accepted 27 March 2020. Date of publication 2 April 2020;
date of current version 18 May 2020. The review of this article was arranged by Associate Editor Peng Li.

Digital Object Identifier 10.1109/OJCS.2020.2984473

Covert Communication via the QR Code Image by a Data Hiding Technique Based on Module Shape Adjustments

DA-CHUN WU AND YUAN-MING WU

Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 82445, Taiwan

CORRESPONDING AUTHOR: Da-Chun Wu (e-mail: dcwu@nkust.edu.tw)

This work was supported by the National Science Council, Taiwan under Grant 102-2221-E-327-023.

ABSTRACT A novel data hiding method for covert communication via the QR code image based on adjustments of the shapes of the QR code modules using image processing techniques is proposed. A module block consisting of two module pairs is taken as the unit for message-bit embedding by adjusting the vertical and horizontal internal boundaries in the module pairs. Eliminations of the resulting boundary line zigzaggedness and irregular overlapping and holing phenomena in the module block are also carried out. The resulting stego-image with secret bits embedded can be scanned by a barcode reader to obtain the facial data recorded in the QR code while a program implementing the message extraction process of the proposed method can process the stego-image to extract the hidden secret message. Good experimental results, tests of stego-image readability by QR code scanners and resistance to noise attacks, and a comparison with other methods from the viewpoint of data embedding rate show the feasibility and superiority of the proposed method for real covert communication applications.

INDEX TERMS Covert communication, data hiding, QR code, module shape adjustment, image processing.

I. INTRODUCTION

With the advance of digital and networking technologies, the issue of secure protection of secret messages like personal data, confidential text, etc. during communication has become more and more important. *Information hiding* [1], [2] via various types of multimedia and documents has been proposed as a solution to this issue., Often adopted for the application of *steganography*, information hiding is a technique for embedding secret message data into certain *cover media* by a data embedding process, resulting in so-called *stego-media* which have no difference from the cover media in appearance. Extraction of the embedded data is basically a reverse process of the data embedding process. The two processes are illustrated briefly in Fig. 1. Information hiding has many applications like covert communication of secret messages, copyright protection by watermarking, secret keep of important documents, etc. In this study, the issue of using the QR code image as the cover media for information hiding is investigated.

In the past decade, the use of the quick response (QR) code [3] is getting more and more popular for various applications like recording information about merchandises, managing the import and export of products, providing links to websites, etc. Almost every brand of smartphone now has its own application program to read QR codes. Therefore, it is appropriate to utilize QR codes as cover media for information hiding.

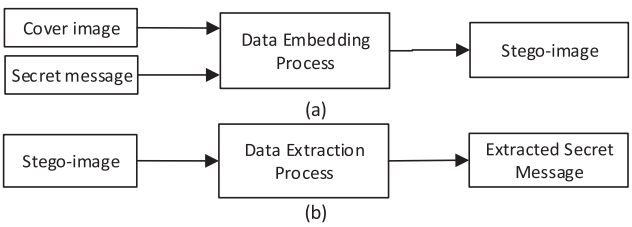

FIGURE 1. Basic concepts of information hiding (steganography). (a) The data embedding process. (b) The data extraction process.

The QR code is a type of *2D barcode*. Another major type of bar code is *linear (1D) barcode*. A typical linear barcode can only be used to record a little piece of information including 10 to 20 characters only with no error-correction function.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

FIGURE 2. Two types of barcode. (a) A linear barcode. (b) A QR code.

Fig. 2(a) shows an example of linear barcodes often seen on commercial products. On the contrary, a 2D barcode like the QR code can include more information than a linear barcode, and has an error-correction capability; i.e., the information in a damaged 2D barcode can often be recovered. An example of QR codes is shown in Fig. 2(b).

The existing studies of QR codes mainly focus on improving the effect of code reading or extending the applications of the code to various applications [4]. It is desired in this study to design a method to embed secret information into a QR code image to yield a *stego-image* from which a common barcode reader can extract just the 'facial' data stored in the QR code, while a specially programmed reading software can extract the additionally embedded *secret message*. Such an information hiding method via QR code images with a steganographic effect is very useful for covert communication of secret messages.

More specifically, in the proposed method it is desired to embed a secret message into a QR code image without destructing the black and white *module* structure of the original QR code image, yielding a stego-image whose appearance has almost no difference from that of the original QR code image. In particular, by the proposed method message bits can be embedded into the QR code image by making slight changes of the boundary positions at the junctions of black and white modules without alternating the original data stored in the QR code. While the QR code content can be read from the stego-image by any person using a common barcode reader, only an authorized user can reveal the secret message hidden in the stego-image by a software program which carries out the data extraction process of the proposed method. Tests of the readabilities of the facial data recorded in the stego-image of the QR code using common barcode readers were conducted. Also carried out were tests of the resistance capability of the proposed method against noise attacks. Good experimental results yielded by the proposed method show the correctness and feasibility of the proposed method for covert communication. A comparison with other methods from the viewpoint of data embedding rate is also conducted, with the results showing the superiority of the proposed method.

In the remainder of this paper, in Section II, a survey of related information hiding studies via QR codes and a review of the QR code structure are introduced. In Section III, the proposed process of adjusting the module contents in the QR code image for data embedding is presented, and in Section IV, schemes proposed for eliminating irregular shapes of module pairs resulting from message-bit embedding are described. In Section V, the proposed data extraction process is described. Experimental results of data embedding and extraction, tests of stego-image readability by scanners and resistance to noise attacks, and a comparison with other methods about data embedding rates are presented in Section VI, with some concluding remarks given in Section VII finally.

II. RELATED STUDIES

A. A SURVEY OF RELATED WORKS

Existing studies of information hiding via QR codes fall roughly into four categories: 1) message-carrier hiding, 2) data or error-correction codeword modification, 3) watermark embedding, and 4) information sharing. In the methods of the first category, the QR code image itself is *not* used as a cover image for data hiding; instead, secret information is encoded into a QR code whose image is then embedded into another image by traditional information hiding techniques, such as halftoning used by Gaikwad and Singh [5] and Garateguy, *et al.* [6], the discrete wavelet transform (DWT) used by Zhang and Yoshino [7], least significant bit (LSB) replacement used by Dey *et al.* [8], etc. These methods may be regarded as extensions of conventional image hiding techniques.

In the methods of the second category, the data or the error-correction codewords of the QR code are modified for information hiding. For example, Chiang *et al.* [9] and Barmawi and Yulianto [10] altered the error correction scheme of the QR code without changing its original information recovery function to achieve the steganographic effect. Specifically, Chiang *et al.* [9] use the concept of the wet paper code [11] to convey the secret into the modules of the QR code, while Barmawi and Yulianto [10] flips the black and white colors of the modules to embed a randomized message-bit sequence. Lin and Chen [12] changed the data codewords according to a modified LSB matching scheme to conduct message-bit embedding. Kamijo, *et al.* [13] used optimally two different error-correction codes, the Reed-Solomon (RS) code as well as the Bose-Chaudhuri-Hocquenghem (BCH) code, for message embedding in invisible QR code images overprinted on the articles of analog media using invisible ink. Huang *et al.* [14] used an improved version of a method proposed by Zhang and Wang [15] which exploits the possible ways of modifications made to a group of image pixels to embed message bits.

The third category includes those methods, each of which is used to embed a watermark into the QR code image by various image transform techniques, like the discrete cosine transform (DCT), discrete wavelet transform (DWT), and discrete Fourier transform (DFT), used by Rungraungsilp, *et al.* [16], Sun, *et al.* [17], Li *et al.* [18], etc. Finally, the information sharing methods in the fourth category were used to embed secret messages into QR code images by various techniques such as visual cryptography used by Cheng *et al.* [19] and Chow *et al.* [20], 2-level QR code construction used by Tkachenko *et al.* [21], wet-paper coding used by Lin [22], etc.

In this study, a new method for data hiding via the QR code image is proposed, which does not belong to any of

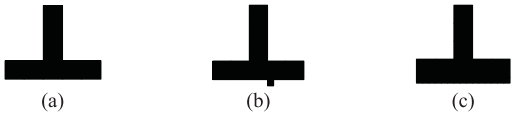

FIGURE 3. An illustration of salt-and-pepper noise reduction by Gou and Wu [24]. (a) An original image. (b) Flipping a white pixel into black under the boundary of black and white pixels. (c) All pixels on the line right under the boundary are flipped into black to form a black line.

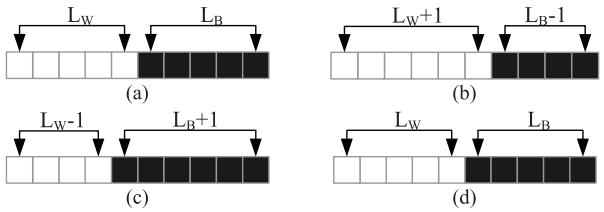

FIGURE 4. An illustration of a method of adjusting the lengths of run pairs for message-bit embedding [26]. (a) Original run pair. (b) and (c) Two ways for embedding a bit of '0' or '1'. (d) One way to embed a reverse bit of that embedded in (b) or (c) by conducting no adjustment of the run pair.

the aforementioned four categories of the existing methods. The proposed method, by regarding the QR code image as a *black and white* image, directly modifies the shapes of the modules in the QR code image *down to the pixel level* to embed message bits, without destroying the validity of the module shapes to cause incorrect reading of the original data stored in the QR code.

Black and white images, also called binary images, have only two levels of gray scales; therefore, information hiding in binary images is more difficult than in color images because the contrast in a binary image is extremely high and so it is not difficult to find any modification in the image. About information hiding in binary images, Wu, *et al.* [23] proposed a data hiding method in which pixels located at the boundary with high contrasts, called flippable pixels, are found and the number of black flippable pixels are changed to embed message data. Gou and Wu [24] proposed later a method which improves the one by Wu, *et al.* [23] by applying the so-called super-pixels to transform the boundary of black and white pixels in such a way that if there exists a black boundary pixel which has been flipped, then those pixels appearing on the same line as the flipped pixel are all flipped as well. In this way, the number of salt-and-pepper noise pixels can be reduced. An illustration of the method is illustrated in Fig. 3.

Run length (RL) encoding is a technique for binary image compression used by fax machines [25], which scan an image horizontally to find continuous black or white pixel runs and represent each run with two attributes: color (black or white) and length (the number of pixels in the run). Hsu and Wu [26] proposed a data hiding method based on RL encoding by which neighboring *run pairs of* different colors are found, and the lengths of pixels in the two runs are adjusted to embed message bits in the way as illustrated in Fig. 4 where the parameters L_W denotes the length of a white run and L_B denotes

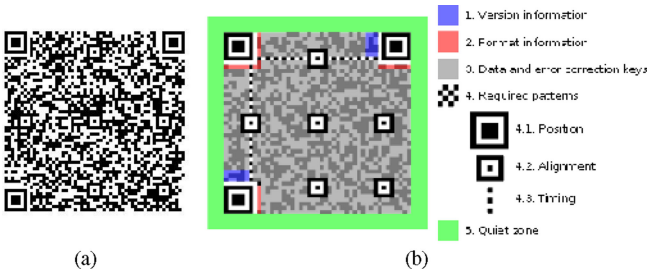

FIGURE 5. QR code structure. (a) A version-10 QR code image. (b) An illustration of the structure, highlighting the functional elements [28].

that of a black run. The adjustment of the pixel colors, namely, from black to white or from white to black, is conducted at the boundary of the run pairs and the resulting total length of the two neighboring runs is not changed; therefore, the modification is visually difficult to find out. Xuan *et al.* [27] used a RL histogram modification technique to embed bit data into a binary image. A sequence of RL couples, each consisting of a black RL with an immediately next white RL, is formed. An RL couple that is sufficient long is chosen for embedding a bit of '1' by increasing the length of the black RL and decreasing that of the white RL; or for embedding a bit of '0' by doing nothing. The method is reversible but not blind, because the changes made to the RL couples must be kept for use in message-bit extraction.

B. A REVIEW OF QR CODE STRUCTURE

The QR code was used initially in 1994 by the Denso-Wave Company in Japan for tracking automobile components. The code was used later extensively on many other applications for various object identification purposes. A QR code, like that shown in Fig. 5(a), is a matrix of *modules* which are black or white *squares* usually of the size of 4×4 pixels. There are 40 *versions* of QR codes with version 1 consisting of 21×21 modules, version 2 of 25×25 modules, version 3 of 29×29, and so on. Each side of every version is four modules larger than that of its former version, so that version 40 is of the size of $(21 + 4 \times 39) \times (21 + 4 \times 39)$ or equivalently, 177×177 modules.

As shown by the example in Fig. 5(a), the appearance of a QR code has a mosaic effect with each of its three corners being of the shape of two concentric squares like the symbol ▣ of the size of 7×7 modules, called *position pattern* (or *finder pattern*), which is used for positioning the QR code image during QR code reading with a barcode scanner. Furthermore, in the image of a QR code of a version larger than 1, there exist smaller squares of the same shape as that of the position pattern but with a smaller size of 3×3 modules, called *alignment patterns*, which are used for precise module alignment during QR code reading. It is noted that the larger the version, the larger the number of alignment patterns. More details of the QR code structure can be found in Fig. 5(b).

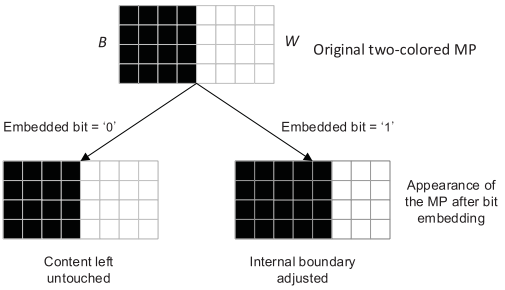

FIGURE 6. An example of horizontal message-bit embedding using a two-colored MP, where the content of the MP is left untouched when the embedded bit is '0' and the internal boundary of the MP is adjusted when the embedded bit is '1'.

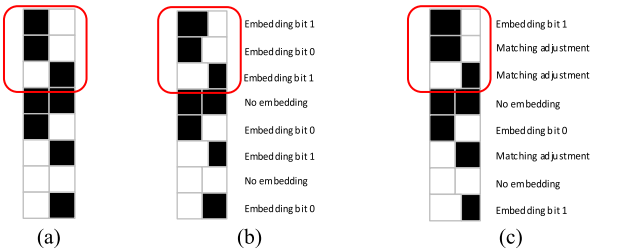

FIGURE 7. An example of applying the zigzaggedness remedy scheme. (a) Original image of a series of MPs. (b) The MPs after embedding message bits '101010', showing a zigzag internal boundary line. (c) The MPs after zigzaggedness remedy is applied.

To avoid interfering the functions of the position and alignment patterns, they are not used for message-bit embedding by the proposed method; i.e., the modules in the image part of these two types of patterns are excluded from being used as inputs to the proposed data embedding and extraction processes.

It is noted also that each QR code includes 8-bit codewords used by the Reed–Solomon error correction algorithm with four *error correction levels*, L (low), M (medium), Q (quantile), and H (high), representing respectively 7%, 15%, 25%, and 30% of the codewords that can be restored when errors occur during the of QR code reading process [3].

III. DATA HIDING BY MODULE SHAPE ADJUSTMENT

A. BASIC IDEAS

A basic idea of the proposed method for hiding a message into a QR code image consisting of modules is as follows.

(B1) Adjust the content of every two neighboring modules of different colors to embed a message bit of '1'; or leave it untouched if the bit to be embedded is '0' or if the two modules are of an identical color.

An example of message-bit embedding according to (B1) is shown in Fig. 6. Two neighboring modules is said in this study to consist of a *module pair (MP)* which may be *single-colored* and *two-colored*, depending on whether the two modules are of an identical color or not. Message-bit embedding *in the horizontal direction* will also be mentioned simply as *horizontal* message-bit embedding subsequently, in which the MP to be processed consists of a *left* module and a *right* one, as shown in the upper part of Fig. 6.

When a message bit of '1' is embedded into a two-colored MP in the horizontal direction, the proposed method enlarges the width of the left module in the MP for a certain number of pixels toward the right so that the *internal boundary* between the two modules in the MP is shifted to the right. This process is called *internal boundary adjustment* subsequently. Also, when the embedded bit is '0', the content of the two modules are left untouched. The number of pixels shifted in the internal boundary adjustment is called the *shift parameter* and is denoted by k in the sequel.

For the reverse case of Fig. 6 where the positions of the black module B and the white module W are swapped, the embedding operation is similar with the width of the white module W now at the left-hand side being enlarged. It is noted that a single-colored MP is *not* used for message-bit embedding by the proposed method.

When a series of two or more *consecutive* two-colored MPs are used to embed message pixels, the internal boundaries of all the consecutive MPs, when viewed integrally, will become a *zigzag line* which is visually easy to be noticed by a hacker. Therefore, to enhance the security of the embedded data, a scheme based on the following idea is proposed in this study as part of the proposed method to remedy this weakness.

(B2) If the currently-processed MP M_{curr} is two-colored, and if its precedently-processed MP M_{prec} is also two-colored, then do not use M_{curr} for message-bit embedding but for *matching adjustment*: *adjusting* the internal boundary of M_{curr} to *match* that of M_{prec}.

By this way of matching adjustment, the phenomenon of yielding *a zigzag internal boundary line* in M_{prec} and M_{curr} resulting from consecutive bit embedding can be removed. This process will be called the *zigzaggedness remedy scheme.*

For example, Fig. 7(a) shows the image of a partial QR code with eight MPs stacked from top to bottom. After a horizontal message-bit embedding process based on the aforementioned idea of (B1) is applied to Fig. 7(a) as illustrated by Fig. 6, the result will become the image shown in Fig. 7(b) in which, e.g., the original *straight line* formed by the three consecutive *internal boundaries of* the topmost three MPs (which are used to embed the three bits '101') becomes *a zigzag line*, consisting of three segments at different positions! Now, if the zigzaggedness remedy scheme is applied, then the image of Fig. 7(c) is yielded, in which the originally zigzag three-segment line in the middle of the topmost three MPs becomes *a single straight line*!

The price paid for zigzaggedness remedy is a possible decrease of the number of embedded bits because some of the two-colored MPs are used for matching adjustment instead of for message-bit embedding. For example, after zigzaggedness remedy only a bit of '1' instead of the original three bits of '101' is embedded in the topmost three MPs; and only three

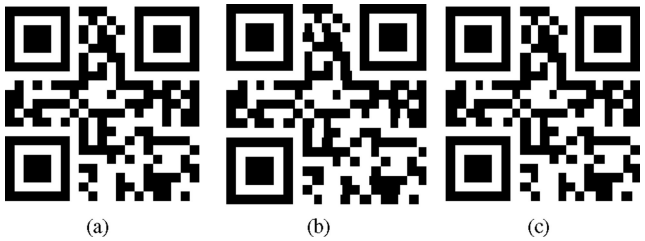

FIGURE 8. An example showing the effect of zigzaggedness remedy. (a) An original image of a QR code of version 1 with a size of 63 × 63 pixels, a module size of 3 × 3 pixels, the error correction level of Q, and the content of 'Data Hiding'. (b) Result of applying horizontal message-bit embedding to (a). (c) Result of applying horizontal message-bit embedding with zigzaggedness remedy to (a) where the zigzaggedness has been removed. Note: the three position patterns are excluded from being used for message-bit embedding.

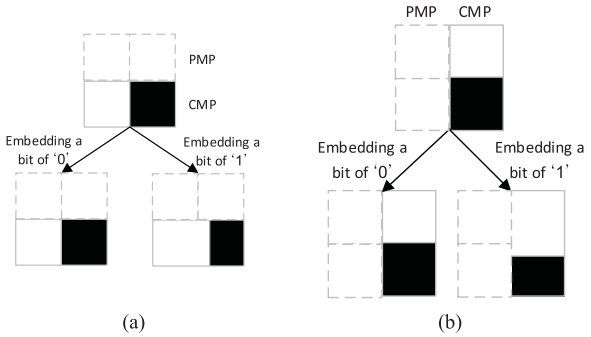

FIGURE 9. Illustrations of message bit embedding into a CMP in an MB. (a) Embedding in the horizontal direction. (b) Embedding in the vertical direction.

bits of '101' instead of the original six bits of '101010' are embedded into the eight MPs, as can be seen from Fig. 7(c).

Fig. 8 shows a more complicated example of the effect of the zigzaggedness remedy scheme after the scheme is applied to a complete QR code, where Fig. 8(a) shows the image of the original QR code which may be regarded to correspond to Fig. 7(a) but with more modules; Fig. 8(b) shows the resulting QR code image after message-bit embedding is applied to Fig. 7(a), which appears to have obvious zigzag boundaries between a few neighboring black and white modules; and Fig. 8(c) shows the result of applying the zigzaggedness remedy scheme to Fig. 8(a), in which the zigzag boundaries can be seen to have been eliminated and the modules look almost identical to the original ones.

B. ADJUSTMENT OF MP CONTENTS FOR DATA EMBEDDING TO ELIMINATE ZIGZAGGEDNESS

In the previous discussions, no detailed explanations were given about how the proposed method adjusts the contents of the MPs for message-bit embedding and internal boundary zigzaggedness elimination *for all possible cases*, which are presented in this section. In the proposed method, a block of 2 × 2 modules, called a *module block (MB)*, is taken as the unit for content adjustment, which includes two MPs, one being precedently processed, called the *preceding module pair (PMP)* followed by a currently processed one, called the *current module pair (CMP)*. During message-bit embedding in the *horizontal* direction, the CMP is the MP located *below* the PMP in the MB as shown in Fig. 9(a); and during message-bit embedding in the *vertical* direction, the CMP is located *to the right of* the PMP as shown in Fig. 9(b).

The processing of the MBs in a QR code image for message-bit embedding in the two directions is explained in detail here. Suppose that the modules of the QR code in the part of the top left corner are labeled as shown in Fig. 10(a). The embedding of message bits in the *horizontal* direction proceeds in a horizontal raster-scan order, starting from the leftmost module of the first row in an MP-by-MP manner *with module overlapping*. Each MP in the row is regarded as a CMP and assigned additionally a *nonexistent* PMP to form an MB as shown by the first line in Fig. 10(b). Then, the first and second rows of modules are combined to form a second row of MBs as shown by the second line in Fig. 10(b), and so on. Similarly, as shown in Fig. 10(c), the embedding of message bits in the *vertical* direction proceeds to do all the

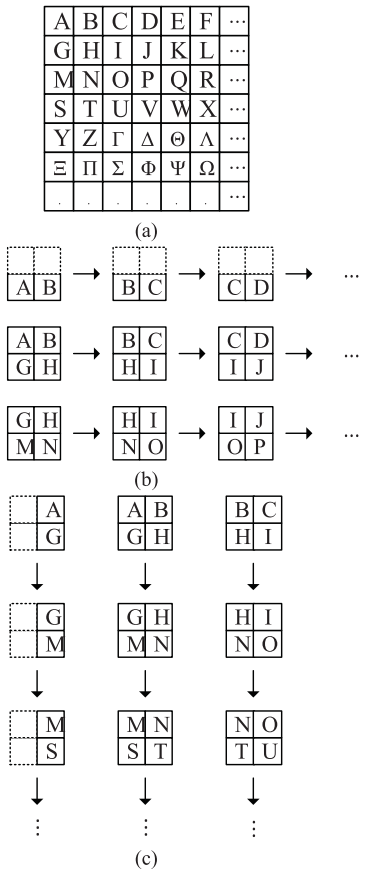

FIGURE 10. An illustration of MB processing for bit embedding in the two directions. (a) A 6 × 6 module block at the top-left corner of a QR code with each character representing a module. (b) and (c) MB processing orders in the horizontal and vertical directions, respectively.

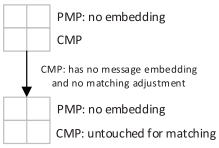

FIGURE 11. MP content adjustment for case h1.1.

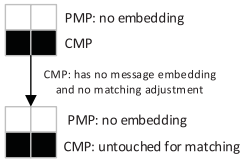

FIGURE 12. MP content adjustment for case h1.2.

same tasks as those done in the horizontal direction except that the processing is now carried out in the vertical direction.

When the first row of modules in a QR code is processed, each MP in the row has a nonexistent PMP (see the first line in Fig. 10(b)) as mentioned previously. Therefore, based on the idea of B1, each MP is 'free' for use: if the MP is single-colored, leave it untouched; else, use it to embed a bit by applying internal boundary adjustment to it if the message bit is '1' or leave it untouched if the bit is '0', as shown by Fig. 6. That is, we have the following first rule for message-bit embedding in the first row of modules where each MP in the row is regarded as a CMP with a nonexistent PMP.

(H1) When there exists *no* PMP: if the CMP is single-colored, it is left untouched; else, it is used for bit embedding.

Furthermore, to process the subsequent rows of MBs formed by MPs for message-bit embedding, at first we classify all the cases of the colors in an MB consisting of a CMP and a PMP into the following four categories:

(h1) both the CMP and the PMP are single-colored;
(h2) the CMP is single-colored and the PMP is two-colored;
(h3) the CMP is two-colored and the PMP is single-colored;
(h4) both the CMP and the PMP are two-colored.

Based on the ideas of B1 and B2, examples of possible cases of combinations of PMPs and CMPs of each of the above four categories for the *horizontal* direction are listed in the following, with the ways to process them for eliminating the resulting internal boundary zigzaggedness being described in the meantime.

(h1) Both the CMP and the PMP are single-colored

(h1.1) Both the CMP and PMP are single-colored and of the same color

When all of the four modules are of the same color like the example shown in Fig. 11, the CMP is used neither for message-bit embedding nor for matching adjustment because it is single-colored.

(h1.2) Both the CMP and PMP are single-colored but of different colors

In this case, the CMP is also used neither for message-bit embedding nor for matching adjustment because it is single-colored, as shown in Fig. 12.

(h2) The CMP is single-colored and the PMP is two-colored

(h2.1) The top left module is of a different color from the other three

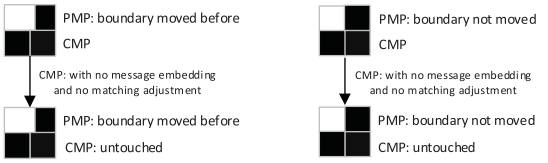

FIGURE 13. MP content adjustments for case h2.1.

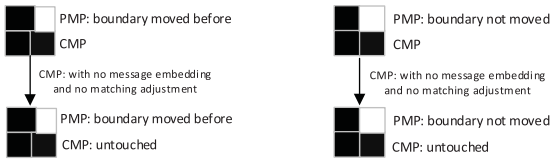

FIGURE 14. MP content adjustments for case h2.2.

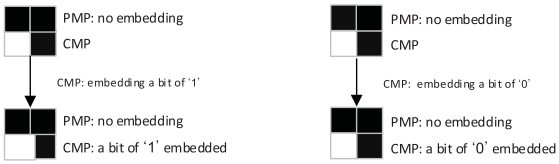

FIGURE 15. MP content adjustments for case h3.1.

In this case, the PMP has two possible usages with its internal boundary being moved before or not, but the CMP is used neither for message embedding nor matching adjustment because it is single-colored, as shown in Fig. 13.

(h2.2) The top right module is of a different color from the other three

This case is symmetric to case h2.1 with the PMP and CMP being processed in a similar way, as shown in Fig. 14.

(h3) The CMP is two-colored and the PMP is single-colored

(h3.1) The bottom left module is of a different color from the other three

In this case, the PMP is not used for message embedding, so the CMP can be used 'freely' for embedding a bit of '1' or '0': to embed '1', the width of the left module of the CMP is enlarged; and to embed '0', no width adjustment is necessary, as shown in Fig. 15.

(h3.2) The bottom right module is of a different color from the other three

This case is symmetric to case h3.1 with the PMP and CMP being processed in a similar way, as shown in Fig. 16.

(h4) Both the CMP and PMP are two-colored

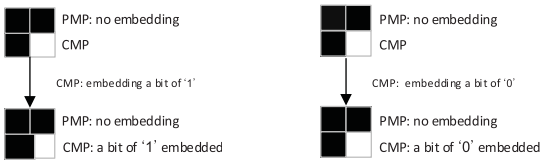

FIGURE 16. MP content adjustments for case h3.2.

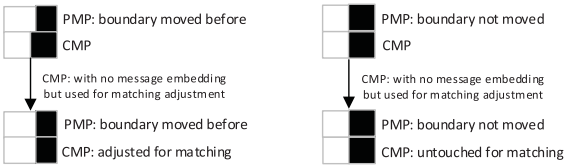

FIGURE 17. MP content adjustments for case h4.1.

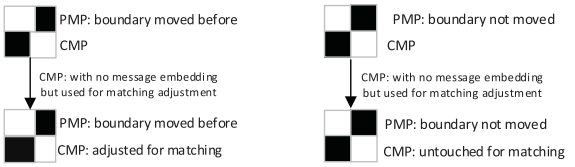

FIGURE 18. MP content adjustments for case h4.2.

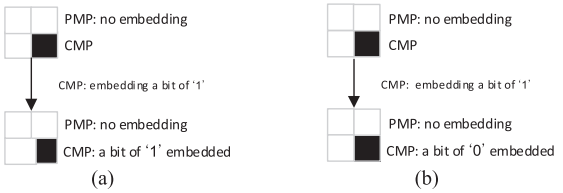

FIGURE 19. MP content adjustments for case h3.2s: the bottom right module is of a different color from the other three.

(h4.1) Both the CMP and the PMP are two-colored and of the same black-and-white pattern

Here, the PMP has two possible usages with its internal boundary being moved before or not, and the CMP is used for matching adjustment only, not for message-bit embedding, as shown in Fig. 17.

(h4.2) Both the CMP and PMP are two-colored and of different black-and-white patterns

Here, the PMP also has two possible usages with its internal boundary being moved before or not, and the CMP is used for matching adjustment only, not for message-bit embedding, as shown in Fig. 18.

The above discussions about possible MP combinations for bit embedding in the horizontal direction actually are *incomplete*; there still exists another set of cases totally *symmetric in black-and-white color* to those above. For example, symmetric to case h3.2 as illustrated in Fig. 16 is the symmetric case h3.2s shown in Fig. 19. But the uses of these symmetric cases for message-bit embedding in the horizontal direction are *totally identical* to their corresponding ones discussed above and so are omitted, except that an extra letter 's' is used to name these cases, resulting in the cases named h1.1s∼h1.2s, h2.1s∼h2.2s, h3.1s∼h3.2s, and h4.1s∼h4.2s.

TABLE 1. Rules H1 Through H3 for Horizontal Message-Bit Embedding

Rule	H1	H2	H3
PMP / CMP	Nonexistent	Single-colored	Two-colored
Single-colored	Bit not embedded (left untouched)		
Two-colored	Bit 0 or 1 embedded (internal boundary adjusted rightward if bit = '1'; or left untouched if bit = '0')		Matching adjustment (internal boundary adjusted to match that of PMP)

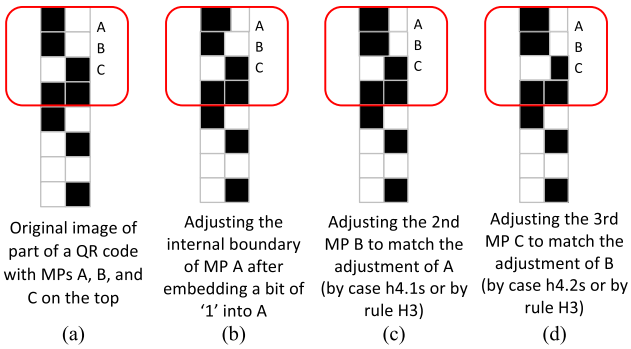

FIGURE 20. An example of remedying a zigzag internal boundary line. (a) The original series of MPs. (b) Embedding a bit of '1' into the top MP. (c) Using the second MP for matching adjustment according to case h4.1s (black-and-white symmetric to the left part of Fig. 17), or equivalently, according to rule H3. (d) Using the third MP for matching adjustment according to case h4.2s (black-and-white symmetric to the left part of Fig. 18), or equivalently, according to rule H3.

From the above discussions of various cases of MP combinations for message-bit embedding in the horizontal direction, the following two rules can be induced from the viewpoint of a PMP, which basically are based on the ideas of B1 and B2.

(H2) When there exists a single-colored PMP: if the CMP is single-colored, leave it untouched; else, use it for *bit embedding*.

(H3) When there exists a two-colored PMP: if the CMP is single-colored, leave it untouched; else, use it for *matching adjustment* instead of for bit embedding.

Rules H1 through H3 mentioned above will be said to form a *rule set H*, which can be summarized up by Table 1.

Based on the cases or rules discussed above, we can now illustrate, by Fig. 20, in more detail how the series of the topmost three MPs A, B, and C in Fig. 7(a) are adjusted by the proposed method to become those in Fig. 7(c).

In addition, the previous discussions about internal boundary adjustment for the various cases of combinations of the PMP and the CMP are conducted for message-bit embedding in the horizontal direction as illustrated by Fig. 9(a); especially, Figs. 11 through 18 are applicable only to the message-bit embedding *in the horizontal direction*. For message-bit embedding *in the vertical direction* as illustrated by Fig. 9(b), the discussions are similar except that the internal boundary

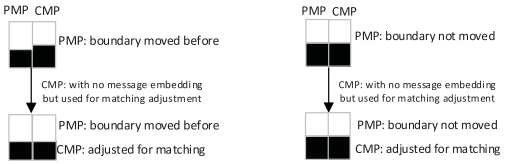

FIGURE 21. An illustration of MP content adjustments for case v4.1: both the PMP and the CMP are two-colored and of the same black-and-white pattern, which corresponds to case h4.1.

TABLE 2. Rules (V1∼V3) for Vertical Message-Bit Embedding

Rule	V1	V2	V3
PMP / CMP	Nonexistent	Single-colored	Two-colored
Single-colored	Bit not embedded (left untouched)		
Two-colored	Bit 0 or 1 embedded (internal boundary adjusted downward if bit = '1'; or left untouched if bit = '0')	Matching adjustment (internal boundary adjusted to match that of PMP)	

adjustments are performed *vertically*, and so the details like those shown in Figs. 11 through 18 are omitted. The corresponding cases are named v1.1∼v1.2, v2.1∼v2.2, v3.1∼v3.2, v4.1∼v4.2; and v1.1s∼v1.2s, v2.1s∼v2.2s, v3.1s∼v3.2s, v4.1s∼v4.2s, all with the prefixed letter 'v' meaning 'vertical'. For example, case v4.1 in the vertical direction, which corresponds to case h4.1 illustrated by Fig. 17, is illustrated by Fig. 21.

Furthermore, there also exist a rule set V which includes rules named V1 through V3 all symmetric to H1 through H3 in the rule set H, respectively, as shown in Table 2. These rules are used in message-bit embedding in the vertical direction.

C. AN ALGORITHM FOR MESSAGE EMBEDDING IN THE TWO DIRECTIONS RESPECTIVELY WITH ZIGZAGGEDNESS REMEDY

The above discussions about message-bit embedding in the two directions and the remedy of the zigzaggedness of the internal boundaries in the MPs may be described formally as an algorithm using pseudo-codes subsequently, where the processing of the modules in a given QR code image I, for example, is performed in a raster-scan manner *row by row in the horizontal direction*, with the MB as the processing unit which consists of a PMP A and a CMP B constructed in the following way.

a) If the row R of modules being scanned currently is the first row (i.e., the topmost row) in I, then regard A to be *nonexistent*, and B to be an MP formed by two neighboring modules in the first row of modules as illustrated in the first line of Fig. 10(b).

b) If R is *not* the first row in I, then take B to be an MP in the currently-scanned row R of modules in I and A to be the MP 'on top of B' in the row right above R as illustrated in the second or third line of Fig. 10(b).

This way of MB construction will call *horizontal MB generation* in the sequel. The term *vertical MB generation* is defined similarly as illustrated in Fig. 10(c).

Algorithm 1: Two-Directional Message Embedding with Zigzaggedness Remedy.

Input: a QR code image I consisting of pre-constructed modules; a nonempty bit string S of a secret message; and a width parameter k for matching adjustment during message-bit embedding.

Output: a stego-image I' of I.

Steps.

while the horizontal MBs in S is not processed exhaustively **do:**

while the input message bit string S is not empty **do:**

1.1 take out the leading bit b in S;

1.2 perform horizontal MB generation to construct sequentially an unprocessed MB M from I with PMP A and CMP B;

1.3 **if** CMP B is in the first module row in I, **then** *//Embedding bits into the 1st row by rule H1*

(a) **if** B is single-colored, **then** leave B untouched; *//Embedding no bit*

(b) **if** B is two-colored, **then** *//Embedding bit b into B*

if b = '0', **then** leave B untouched to embed b = '0';

else adjust the internal boundary in B rightward to embed b = '1';

1.4 **if** B is *not* in the first row of modules in I, **then** *//Embedding bits into rows after the first by rules H2 and H3*

(a) **if** PMP A is single-colored, **then**

(i) **if** B is single-colored, **then** leave B untouched; *//Embedding no bit*

(ii) **if** B is two-colored, **then** *//Embedding bit b into B*

if b = '0', **then** leave B untouched to embed b = '0';

else adjust the internal boundary in B rightward to embed b = '1';

(b) **if** PMP A is two-colored, **then**

(i) **if** B is single-colored, **then** leave B untouched; *//Embedding no bit*

(ii) **if** B is two-colored, **then** *//Using B for matching adjustment*

adjust the internal boundary of B to match that of A;

end while; *//End of checking emptiness of message string S*

end while. *//End of Step 1*

Step 2: *//Embedding bits vertically*

while the vertical MBs in S is not processed exhaustively **do:**

while the input message bit string S is not empty **do:**

2.1 take out the leading bit b in S;

2.2 perform vertical MB generation to construct sequentially an unprocessed MB M from I with PMP A and CMP B;

2.3 **if** CMP B is in the first module column of I, **then**
//Embedding bits into the 1st column by rule V1
(a) **if** B is single-colored, **then** leave B untouched;
//Embedding no bit
(b) **if** B is two-colored, **then**
//Embedding b into B
if b = '0', **then** leave B untouched to embed b = '0';
else adjust the internal boundary in B downward to embed b = '1';
2.4 **if** B is *not* in the first module column in I, **then**
//Embedding bits into columns after the first by rules V2 and V3
(a) **if** PMP A is single-colored, **then**
(i) **if** B is single-colored, **then** leave B untouched; *//Embedding no bit*
(ii) **if** B is two-colored, **then**
//Embedding b into B
if b = '0', **then** leave B untouched to embed b = '0';
else adjust the internal boundary in B downward to embed b = '1';
(b) **if** PMP A is two-colored, **then**
(i) **if** B is single-colored, **then** leave B untouched; *//Embedding no bit*
(ii) **if** B is two-colored, **then** adjust the internal boundary of B to match that of A;
//Using B for matching adjustment
end while; *//End of checking emptiness of message string S*
end while. *//End of Step 2*
Step 3: *//Ending of the algorithm*
exit with the resulting image I' as the desired output. ■

In the above algorithm, while adjusting the internal boundary of the CMP B, the required work includes updating the coordinates of the two corners at the two ends of the adjusted boundary in B to change the color of the resulting expanded area as illustrated in Fig. 6.

Additionally, some simpler versions of Algorithm 1 have also been designed in this study to conduct message-bit embedding works in just the horizontal or just the vertical direction with or without zigzaggedness remedy. These simpler algorithms were used in the experiments conducted in this study for the purpose of comparisons of the yielded images in different conditions as described next.

D. AN EXAMPLE OF EXPERIMENTAL RESULTS

An example of experimental results yielded by Algorithm 1 and its simpler versions aforementioned is shown in Table 3 where a QR code image of the size 243 × 243 pixels is used as the input (in Row 1 in the table), the module size is 9 × 9 pixels, and the shift parameter, k, (i.e., the number of pixels shifted for internal boundary adjustment in an MP) is 3. As can be seen from Rows 2 and 3 in the table, after horizontal and vertical bit embedding were carried out, respectively, with no zigzaggedness remedy using simpler versions of Algorithm 1, 191 and 214 message bits can be embedded, respectively. And when both the horizontal and vertical embedding works were carried out sequentially, the number of embedded message bits is 315 as described in Row 4, which is just the sum of those of the respective works, namely, 191 plus 214. However, inspecting the resulting QR code images, serious zigzaggedness phenomena can be found, especially in the image yielded by the combined horizontal and vertical embedding processes, i.e., in the image appearing in Row 4. Note that the three position patterns and the one alignment pattern were not used for message-bit embedding here.

Subsequently, message-bit embedding works *with* zigzaggedness remedy were implemented and the results are shown in Rows 5 through 7 in Table 3. As can be seen, the resulting images shown in the three rows all have *almost no* zigzaggedness, though the message-bit embedding

TABLE 3. An Example of Experimental Results of Combining Horizontal and Vertical Message-Bit Embedding [a]

No.	QR code image	message-bit embedding direction(s)	zigzag. adjustment	#embedded bits	#overlaps	#holes
1		no	no	0	0	0
2		horizontal	no	191	0	0
3		vertical	no	214	0	0
4		horizontal + vertical	no	315	45	35
5		horizontal	yes	121	0	0
6		vertical	yes	121	0	0
7		horizontal + vertical	yes	242	25	34

[a]Notes: (1) processed QR code properties: version = 2; error correction level = M; image size = 243 × 243 pixels; 27 × 27 modules; module size = 9 × 9 pixels; (2) shift parameter k = 3 pixels; (3) abbreviations: zigzag. = zigzaggedness).

FIGURE 22. The result of combined horizontal and vertical embedding with zigzaggedness remedy. (a) Enlarged version of the image in Row 7 of Table 3 which has many overlaps and holes. (b) Another version of (a) with the overlaps and holes filled with different colors (orange for holes and aqua for overlaps).

capabilities become weaker, with 121, 121, and 242 bits being embedded, respectively, which may be contrasted with the originally embedded 191, 214, and 315 bits, respectively.

However, by a careful inspection of the image shown in Row 7 (yielded by the horizontal message embedding process followed by the vertical message embedding process performed by Algorithm 1) and an enlarged version of the same image shown in Fig. 22(a), it can be seen that a number of *overlaps* of black and white modules and *holes* surrounded by modules have appeared in the image (marked as colored squares in Fig. 22(b)). The numbers of such overlapping and holing phenomena have been shown in the last two columns in Table 3, which are 25 and 34, respectively. Actually, such overlapping and holing phenomena also appear in the resulting image shown in Row 4 which is yielded by a simpler version of Algorithm 1 with no zigzaggedness adjustment. It is desired to eliminate such irregular shapes to improve the zigzaggedness of the images resulting from message-bit embedding. Removal of such overlapping and holing phenomena is the topic of the next section.

IV. MESSAGE-BIT EMBEDDING WITHOUT IRREGULAR SHAPE GENERATION

A. IRREGULAR SHAPE PHENOMENA YIELDED IN TWO-DIRECTIONAL MESSAGE-BIT EMBEDDING

To raise the bit embedding rate, it is desired to conduct the embedding process both horizontally and vertically. However, if the embedding is conducted firstly from the horizontal direction according to cases h1.1 through h4.2 and h1.1s through h4.2s (or equivalently, according to rules H1 through H3) and then from the vertical direction according to cases v1.1 through v4.2 and v1.1s through v4.2s (or equivalently, according to rules V1 through V3), then the aforementioned overlapping and holing phenomena might arise. The reasons are explained here. An example is shown in Fig. 23 where an MB, which may be regarded both as consisting of a PMP stacked on top of a CMP (denoted as PMP_h and CMP_h, respectively, with the subscript h meaning *horizontal*) or as consisting of a PMP located to the left of a CMP (denoted as PMP_v and CMP_v with the subscript v meaning *vertical*). The processing of the MB includes bit embedding in the *horizontal* direction using the PMP_h and the CMP_h, followed later by

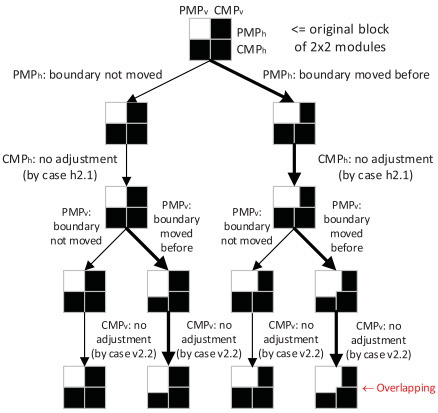

FIGURE 23. Overlapping phenomenon appearing in one of the four results (the bottom right one) yielded by horizontal and vertical message-bit embedding according to cases h2.1 and v2.2, respectively.

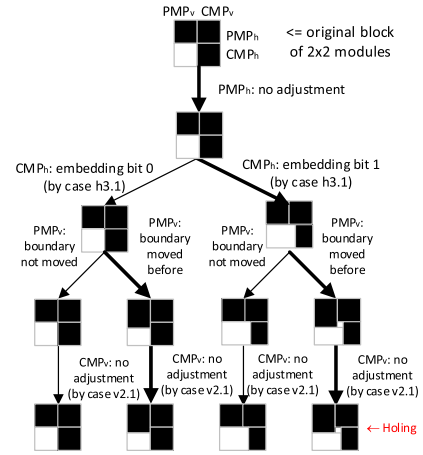

FIGURE 24. Holing phenomenon appearing in one of the four results (the bottom right one) yielded by horizontal and vertical message-bit embedding according to cases h3.1 and v2.1, respectively.

bit embedding using the PMP_v and the CMP_v in the *vertical* direction.

As can be seen in the four possible resulting MBs (drawn at the bottom of the figure) yielded by horizontal bit embedding (using case h2.1) followed by vertical bit embedding (using case v2.2), the rightmost MB appears to have a black module at the bottom right corner *intruding* into the top left white module to form a black convex corner, creating an overlapping phenomenon. Another example of conducting bit embedding processes in the two directions sequentially yields a result as shown in Fig. 24 where the rightmost MB at the bottom appears to have a concavity in the middle of the four modules, creating a *holing* phenomenon. These two types of phenomenon, overlapping and holing, will be called *irregular shape phenomena*.

Actually, according to an exhaustive investigation conducted in this study about all possible color combinations in an MB as shown in Table 4, there are totally eight cases that

TABLE 4. Exhaustive Listing of all Possible Color Combinations of An MB[a]

No.	Structure	Frame	Color combination	Validity
1	OOOO			valid
2	OOOX		nonexistent	nonexistent
3	OOXO		nonexistent	nonexistent
4	OOXX			valid
5	OXOO		nonexistent	nonexistent
6	**OXOX**			**overlapping a1, a2**
7	**OXXO**			**holing b3, b4**
8	OXXX			valid
9	XOOO		nonexistent	nonexistent
10	**XOOX**			**holing b1, b2**
11	**XOXO**			**overlapping a3, a4**
12	XOXX			valid
13	XXOO			valid
14	XXOX			valid
15	XXXO			valid
16	XXXX			valid

[a]*Note:* 'O' in the structure of the second column means: adjusting the internal boundary rightward or downward for embedding a bit of '1' or for matching adjustment and 'X' means no such adjustment.

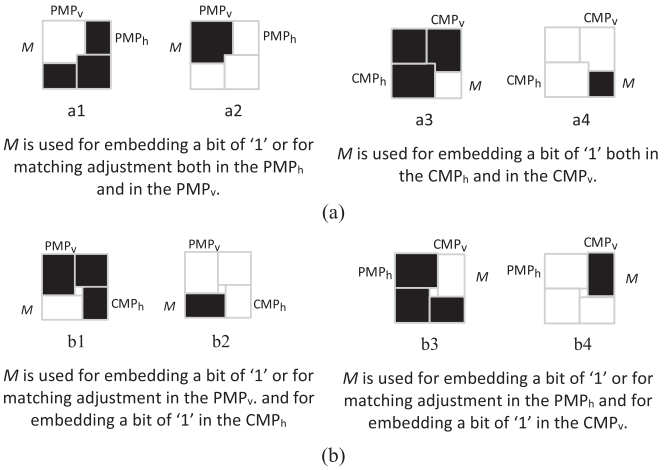

FIGURE 25. Exhaustive cases of overlapping and holing. (a) Overlapping cases a1 through a4. (b) Holing cases b1 through b4.

will yield the phenomena of overlapping or holing in the result of message-bit embedding in the horizontal direction followed by message-bit embedding in the vertical direction, as shown in Fig. 25. And a careful inspection on these cases leads to the following summary about how the two types of irregular shape phenomena are formed.

1) *A common feature of the two types*: an overlapping or holing phenomenon will occur in an MB only when the MB includes three modules of an identical color and one module of the other color.
2) *When and where an overlapping phenomenon will occur:* an overlapping phenomenon will occur only when the processing of a module *M* satisfies either condition below (see cases 6 and 11 in Table 4).
   (2.1) *M* appears in an MB *B*, satisfying the structure condition represented by the symbol sequence 'OXOX' in case 6 of Table 4, where
      a) symbol 'O' in the first symbol pair of 'OX' means that *in the horizontal direction* the PMP_h in *B* is used for *embedding* a bit of '1' or for matching adjustment in the horizontal direction (so that the vertical internal boundary is moved to the right);
      b) symbol 'X' in the first pair of 'OX' means that the CMP_h in *B* is *not used in the same way*; and
      c) the second symbol pair of 'OX' means similarly but about the uses of the PMP_v and CMP_v *in the vertical direction*,

   so that *M* becomes the intersection of PMP_h and PMP_v represented by the two symbols of 'O' and so appears as the *top left module* in *B* as in the case of a1 or a2 shown in Fig. 25(a).
   (2.2) *M* appears in an MB *B*, satisfying the structure condition of 'XOXO' in case 11 of Table 4, with the two XOs interpreted similarly to case (2.1) above but with the uses of the PMPh and CMPh or the uses of the PMPv and CMPv interchanged, so the *M* appears as the bottom right module in *B* as in the case of a3 or a4 shown in Fig. 25(a).
3) *When and where a holing will occur:* a holing will occur only when the processing of a module *M* satisfies either condition below (see cases 7 and 10 in Table 4).

(3.1) M appears in an MB B, satisfying the structure condition represented by the symbol sequence 'OXXO' in case 7 of Table 4, where
   a) the first symbol pair 'OX' means the same as described in (a) and (b) of (2.1) above; and
   b) the second two symbols 'XO' meaning *the reverse* about the uses of the PMP_v and CMP_v *in the vertical direction*,

   so that M becomes the intersection of PMP_h and CMP_v represented by the two symbols of 'O' and so appears as the *top right module* in B as in the case of b3 or b4 shown in Fig. 25(b).

(3.2) M appears in an MB B, satisfying the structure condition of XOOX in case 10 of Table 4, with the two symbol pairs 'XO' and 'OX' interpreted similarly to case (3.1) above but with the uses of the PMP_h and CMP_h or the uses of the PMP_v and CMP_v *interchanged*, so the M appears as the *bottom left module* in B as in the case of b1 or b2 shown in Fig. 25(b).

Obviously, the aforementioned two types of shape irregularity, overlapping and holing, should be eliminated because they are unnatural in common shapes of QR codes and might arouse a hacker's notice. By the way, it is assumed in this study that the 'background color' of the MB is white so that a hole appears to be white as well.

B. ELIMINATION OF OVERLAPPING AND HOLING PHENOMENA

From the above discussions, it can be figured out that an overlapping or holing case occurs at the $k \times k$ square area to the right and below the central point of the MB, as can be seen from Fig. 25. To use this geometric information for eliminating overlapping and holing, whether a case of such phenomena has occurred must be decided in advance. For this, a decision scheme is proposed in this study, which is based on the idea of detecting the locations of the horizontal and vertical internal boundaries in an MB, as described subsequently.

Shown in Fig. 26 is an illustration of an MB with four modules in which the location of each module M in the MB can be fully specified by the coordinates of its top left and bottom right corners. For example, the location of the top left module $M_{i,j}$ can be specified by the coordinates $(Hs_{i,j}, Vs_{i,j})$ of its top left corner $S_{i,j}$ (with the symbols 'S' and 's' in the names of the coordinates and the corner here both mean '*start*') and the coordinates $(He_{i,j}, Ve_{i,j})$ of its bottom right corner $E_{i,j}$ (with the symbols 'E' and 'e' here both mean '*end*'). The other three modules can be specified similarly. Here it is assumed that the origin (0, 0) of the image coordinate system is located at the top left corner of the image with the positive H and V coordinates being on its right and lower sides, respectively.

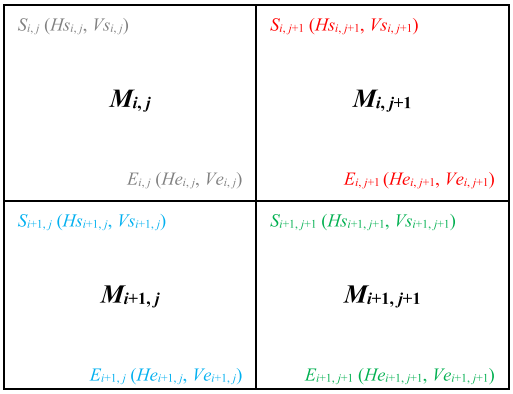

FIGURE 26. An illustration of the original locations of the four modules in an MB specified by their respective top left and bottom right corners.

When the MB is not used for data embedding yet, the following equalities can be seen to be true:

$$Hs_{i,j+1} = He_{i,j} = Hs_{i+1,j+1} = He_{i+1,j}.$$
$$Vs_{i+1,j} = Ve_{i,j} = Vs_{i+1,j+1} = Ve_{i,j+1}$$

But after the horizontal and vertical message-bit embedding processes are started, if the *vertical* and *horizontal* internal boundaries in the MB are moved *rightward* and *downward*, respectively, then each of the above two *long* equalities are cut into two *halves* as follows:

$$Hs_{i,j+1} = He_{i,j}; \qquad Hs_{i+1,j+1} = He_{i+1,j};$$
$$Vs_{i+1,j} = Ve_{i,j}; \qquad Vs_{i+1,j+1} = Ve_{i,j+1},$$

where each "half" equation corresponds to a boundary side of the four modules in the MB. In addition, the color of each module are known in advance. Therefore, each module $M_{i,j}$, like the one shown in the top left corner of the MB shown in Fig. 26, may be specified by a *5-tuple feature vector* $f_{i,j}$ as follows:

$$f_{i,j} = (Hs_{i,j}, Vs_{i,j}, He_{i,j}, Ve_{i,j}, c_{i,j})$$

where $(Hs_{i,j}, Vs_{i,j})$ and $(He_{i,j}, Ve_{i,j})$ are the coordinates of the top left and bottom right corners $S_{i,j}$ and $E_{i,j}$ of $M_{i,j}$, respectively, and $c_{i,j}$ is the color of $M_{i,j}$.

After the occurrence of an overlapping or holing case is detected, it can be eliminated according to the knowledge that the case must occur at the $k \times k$ square area located to the right and just below the central point of the MB, as said before. Note that such a central point always exists because each MB processed in this study consists of 2×2 modules.

The proposed elimination scheme is based on the observations of the specific shapes of the overlapping cases a1 through a4 and the holing cases b1 through b4 shown in Fig. 25, as well as on the geometry of the positions of the corners in each case. For example, for case a1 which appears in the left part of Fig. 25(a), the following condition can be seen to be true:

> the bottom right corner $E_{i,j}$ of module $M_{i,j}$ at $(He_{i,j}, Ve_{i,j})$ is located *to the right and below* the top left corner $S_{i+1,j+1}$ of

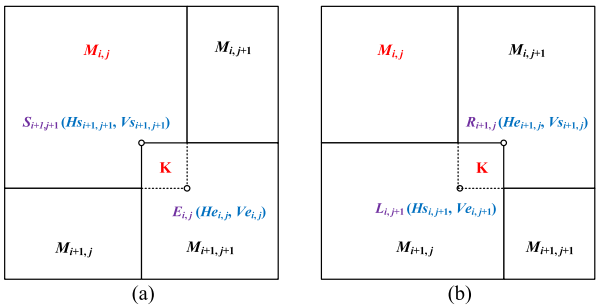

FIGURE 27. Illustrations of MB structures of the four cases of overlapping. (a) Structure of a1 and a2. (b) Structure of a3 and a4.

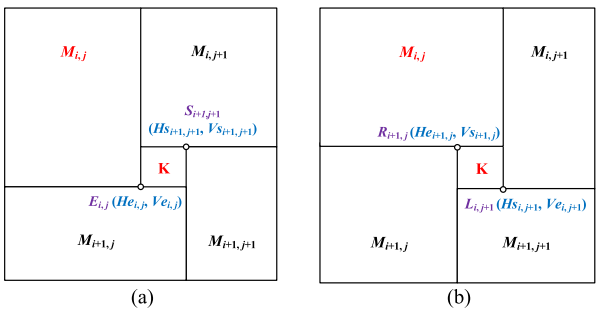

FIGURE 28. Illustrations of MB structures of the four cases of holing. (a) Structure of b1 and b2. (b) Structure of b3 and b4.

module $M_{i+1,j+1}$ at $(Hs_{i+1,j+1}, Vs_{i+1,j+1})$ as shown in Fig. 27(a) so that the following inequalities are true:

$$He_{i,j} > Hs_{i+1,j+1} \text{ and } Ve_{i,j} > Vs_{i+1,j+1}.$$

That is, case a1 can be decided by judging the relative positions of the corners $E_{i,j}$ and $S_{i+1,j+1}$ with respective to each other in the above way using the above two inequalities. In short, it can be said that case a1 occurs if $E_{i,j}$ is located *to the right and below* the top left corner $S_{i+1,j+1}$.

To eliminate case a1 which, as said before, occurs in the $k \times k$ square area K located to the right and just below the central point of the currently-processed MB, the top left and bottom right corners of K at coordinates $(Hs_{i+1,j+1}, Vs_{i+1,j+1})$ and $(He_{i,j}, Ve_{i,j})$, respectively, as shown in Fig. 27(a) can be used to construct K in the following way:

for each pixel P with coordinates (H, V) in image I, **if**

$$Hs_{i+1,j+1} \le H \le He_{i,j} \text{ and } Vs_{i+1,j+1} \le V \le Ve_{i,j},$$

then decide P to be in K.

In other words, we "use the two corners $S_{i+1,j+1}$ and $E_{i,j}$ to construct K" as can be seen from Fig. 27(a). Subsequently, the color of each pixel P in K can be replaced by that of the top left module $M_{i,j}$, namely, $c_{i,j}$, as can be seen from the left part of Fig. 25(a) again. For the other three cases a2, b1, and b3 shown in Figs. 25, the detection and elimination of each of them can also be conducted in a similar way but by the relative positions of a respective pair of corners, as can be seen from the illustrations of Figs. 27(b), 28(a) and 28(b), respectively, again.

It is noted by the way that elimination processes like that described above need not be conducted for cases a3, a4, b2, and b4 because either a3 or a4 creates *no convexity* in black-and-white appearance as shown in the right part of Fig. 25(a), and either b2 or b4 creates *no concavity* as shown in Fig. 25(b), under the previously-mentioned assumption that the background color of the MB is white. The square area K in which any of cases a1 through a4 occurs will be called an *overlapping square*, and that in which any of cases b1 through b4 occurs will be called a *holing square*. Also, as can be seen from Fig. 25(b), either of cases b1 and b3 can be eliminated by filling up the holing square with the color of module $M_{i,j}$, namely, color $c_{i,j}$.

Now, the proposed scheme for detection and elimination of the overlapping or holing phenomenon in an MB can be described integrally as an algorithm in the following.

Algorithm 2: Detection And Elimination of Irregular Shape Phenomena in an MB.
Input: the feature vectors of the four modules in an MB B in a QR code image I as shown in Fig. 26.
Output: a modified MB B' with no overlapping or holing phenomenon.
Steps. *//Check Figs. 25, 27 and 28 for notations in the algorithm*
Step 1: *//Detecting any overlapping case and making a decision D*
if the corner $E_{i,j}$ is located *to the right and below* the corner $S_{i+1,j+1}$ as shown in Fig. 27(a), **then**
if the color $c_{i,j}$ of $M_{i,j}$ is white, **then** set D = a1; **else**, set D = a2;
Step 2: *//Constructing the overlapping square* K
if D = a1 or a2, **then** use the two corners $S_{i+1,j+1}$ and $E_{i,j}$ to construct the overlapping square K.
Step 3: *//Detecting holing cases and making a decision D*
3.1 **if** the corner $E_{i,j}$ is located to the left and below the corner $S_{i+1,j+1}$ as shown in Fig. 28(a), **then**
if the color $c_{i+1,j}$ of $M_{i+1,j}$ is white, **then** set D = b1;
3.2 **if** the corner $L_{i,j+1}$ is located to the right and below corner $R_{i+1,j}$ as shown in Fig. 28(b), **then**
if the color $c_{i,j+1}$ of $M_{i,j+1}$ is white, **then** set D = b3.
Step 4: *//Constructing the holing square* K
4.1 **if** D = b1, **then** use the two corners $S_{i+1,j+1}$ and $E_{i,j}$ to construct the holing square K.
4.2 **if** D = b3, **then** use the two corners $R_{i+1,j}$ and $L_{i,j+1}$ to construct the holing square K.
Step 5: *//Eliminating overlapping or holing phenomena in* K
if D = a1, a2, b1, or b3, **then** replace the color of each pixel in K by the color of $M_{i,j}$, namely, $c_{i,j}$.
Step 6: *//Ending of the algorithm*
exit with the resulting MB B' as the desired output. ■

C. MESSAGE EMBEDDING WITH ZIGZAGGEDNESS REMEDY AND IRREGULAR SHAPE ELIMINATION

It is now ready to give a complete algorithm for embedding a message into a given QR code image both in the horizontal and vertical directions with both internal boundary zigzaggedness remedy and irregular shape elimination being conducted.

Algorithm 3: Two-Directional Message Embedding With Both Zigzaggedness Remedy And Irregular Shape Elimination Being Conducted.

Input: a QR code image I; a secret message S consisting of characters to be hidden in I; and the shift parameter k for matching adjustments during message-bit embedding.

Output: a stego-image I_s with no internal boundary zigzaggedness and no overlapping and holing phenomenon.

Steps.

Step 1: *//Decomposing QR code image into modules*
perform the following steps to locate the modules in the input image I:

1.1 raster-scan the pixels of I horizontally and vertically to find all the black and white runs in I, and compute the length of each run;

1.2 accumulate the different lengths of the runs in both the horizontal and the vertical directions to construct a run-length histogram G, and find the run length appearing the most frequently in G as the side length w of the (square) module in I;
//Each module is of the size w×w pixels

1.3 locate the modules in I by using the value of w to compute respectively the coordinates $(Hs_{i,j}, Vs_{i,j})$ and $(He_{i,j}, Ve_{i,j})$ of the top left and bottom right corners of each module $M_{i,j}$ in I;

1.4 construct a 5-tuple feature vector $(Hs_{i,j}, Vs_{i,j}, He_{i,j}, Ve_{i,j}, c_{i,j})$ for each $M_{i,j}$ in I where $c_{i,j}$ is the color of $M_{i,j}$, and regard I to consist of the located modules with their respective 5-tuple feature vectors;

1.5 count the number n of modules in I and compute the version number V of I according to the QR code specification [3] as

$$V = (\sqrt{n} - 17) / 4;$$

//Version N is 4 modules larger than version N − 1 at each side

1.6 use V to find the image parts of the position and alignment patterns according to the QR code specification [3] and remove them from I, resulting in a new image I';

1.7 transform the characters of the input message S into a bit string S'.

Step 2: *//Embedding bits into the QR code image*
perform Algorithm 1 with image I' consisting of located modules, the bit string S', and the shift parameter k as the inputs to embed S' into I', yielding a stego-image I_0, each of whose modules has an updated 5-tuple feature vector.
//The resulting mage I_0 might include overlaps and holes in its MBs

Step 3: *//Eliminating holes and overlaps in the stego-image I_0*
while the horizontal MBs in image I_0 is not processed exhaustively **do:**

3.1 perform horizontal MB generation to construct sequentially an unprocessed MB M from I_0 with PMP A and CMP B;

3.2 perform Algorithm 2 with M as the input to eliminate possible overlapping and holing phenomena in M, resulting in a new MB.

end while.

Step 4: *//Ending of the algorithm*
combine all the processed MBs with those of the position and alignment patterns to compose a new stego-image I_s as the desired output and exit. ■

D. AN EXAMPLE OF EXPERIMENTAL RESULTS OF ELIMINATING OVERLAPPING AND HOLING PHENOMENA

As a continuation of the example shown in Sec. III.D, Algorithm 2 was applied in this study to a result I_1 yielded by Algorithm 1 and shown in Row 7 in Table 3 which includes 25 overlaps and 34 holes. The stego-image I_2 yielded by Algorithm 2 is shown in Row 9 in Table 5. As can be seen from I_2, the holing and overlapping phenomena found in the last result I_1 have all been eliminated. For convenience of comparing I_2 with I_1, the result I_1 yielded by Algorithm 1 is also included in Rows 7 and 8 of Table 5 (with the latter in Row 8 being marked by colors). Furthermore, larger versions of the images in Rows 8 and 9 of Table 5 are shown in Fig. 29 for more detailed inspections.

TABLE 5. An Experimental Result of Algorithm 2 Which Eliminates the Holes and Overlaps Yielded by Algorithm 1 (the Result is the Image Shown in Row 9)

No.	QR code image	Message-bit embedding direction(s)	zigzag. adjustment	#embedded bits	#overlaps	#holes
7		horizontal + vertical	yes	242	25	34
8 [a]		horizontal + vertical	yes	242	25 (marked in color)	34 (marked in color)
9 [b]		horizontal + vertical	yes	242	0 (holes filled)	0 (overlaps removed)

[a] The overlapping and holing squares in the image of this row are marked in color, and a larger version of the image is shown in Fig. 29(a).
[b] A larger version of the image of this row is shown in Fig. 29(b).

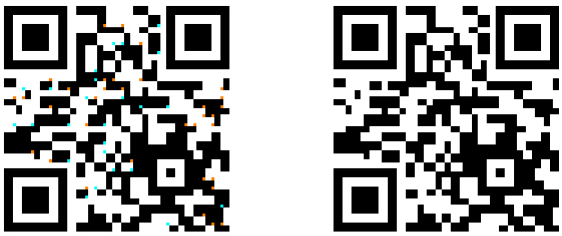

FIGURE 29. Result of applying Algorithm 2 to eliminate the overlapping and holing phenomena yielded by Algorithm 1. (a) A result of Algorithm 1 shown in Row 7 of Table 3 and repeated here in a larger version which has many overlaps and holes marked by different colors (becoming the image of Row 8 in Table 4 with orange colors for holes and aqua colors for overlaps). (b) The result of applying Algorithm 2 to (a) with all the overlaps and holes filled up with correct colors (i.e., all the irregular shape phenomena have been eliminated correctly) (becoming the image of Row 9 in Table 4).

V. MESSAGE EXTRACTION FROM STEGO-IMAGES

A. FACTS ABOUT THE MBS OF THE STEOG-IMAGE OF A QR CODE

To conduct message extraction from a stego-image, at first whether the vertical and horizontal internal boundaries in the two-colored MB have been adjusted according to rules H1 through H3 or according to V1 through V3 should be known

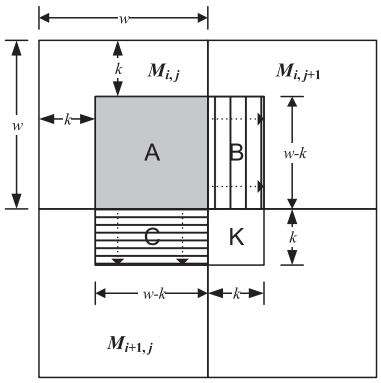

FIGURE 30. An illustration of the location of an MB in the process of message bit extraction after both horizontal and vertical message-bit embedding processes have been applied to it where *k* is the shift parameter used in matching adjustment.

in advance. For this purpose, the illustration in Fig. 30 which depicts a MB can be used, where possible changes of the area of the top left module, $M_{i,j}$, are shown in detail. Based on Fig. 30, the following observations can be made.

1) About the extraction of the color $c_{i,j}$ of module $M_{i,j}$ —

It is known that each possible vertical internal boundary adjustment is to move the boundary to the right for k pixels and each possible horizontal internal boundary adjustment is to move the boundary downward for k pixels as well, where k is the shift parameter. However, no matter how the boundaries are moved during message-bit embedding, the area labeled as A shown in Fig. 30 with size $(w - k) \times (w - k)$ pixels will *never be touched*, from which the color $c_{i,j}$ of the module $M_{i,j}$ can therefore be extracted correctly.

2) About the condition for checking rightward internal boundary movement, called the *move-right condition*—

If the color of area B is identical to that of area A, then it can be decided that a *rightward* movement of the vertical internal boundary must have been conducted during message-bit embedding, possibly in a case of embedding a bit of '1' or performing a matching adjustment according to the rules in rule set H.

3) About the condition for checking downward internal boundary movement, called the *move-down condition*—

If the color of area C is identical to that of area A, then it can be decided that a *downward* movement of the horizontal internal boundary must have been conducted during message-bit embedding, possibly in a case of embedding a bit of '1' or performing a matching adjustment according to the rules in rule set V.

4) About the acquisition of the color in area A, B, or C—

The color c of the area A, B, or C mentioned previously may be obtained by the following steps:

i) compute the range R of area A, B, or C in terms of the coordinates of the left top corner of module $M_{i,j}$ as well as the values of the two parameters, w and k, as can be easily figured out from the illustration shown in Fig. 30;

ii) find out the colors of all the pixels in R;

iii) conduct a voting using the colors to find the color with the maximum votes and take it to be the desired color c in the area.

The voting operation mentioned above is necessary because the input QR code images used in the experiments in this study are assumed to be subject to possible noise attacks, so that the pixels in area A, B, or C are *not uniformly white or black in color*. Therefore, whenever the color of $M_{i,j}$ is to be extracted or whenever either of the two conditions mentioned above is to be checked, it is assumed that a preliminary step is performed to conduct the necessary voting operation as described above to obtain the desired colors from the related areas.

B. EXTRACTION OF EMBEDDED MESSAGE BITS FROM MBS

After observing the above facts about the MB in the stego-image of a QR code, it is now ready to describe how to extract a message bit from an MB from the two directions as two algorithms, respectively, in the following.

Algorithm 4: Extraction of a Message Bit From an MB From The Horizontal Direction.
Input: the feature vectors of an MB M in a stego-image I of a QR code as shown in Fig. 26 with a message bit b embedded.
Output: the bit b.
Steps.
Step 1: *//Initialization*
1.1 denote the two MPs of the MB M in the horizontal direction as PMP_h A and CMP_h B;
//Labeling the MPs in M
1.2 set bit b to be empty initially.
Step 2: if CMP_h B is in the first row of modules in I, **then**
//Extracting the bit from the 1st row according to rule H1
(i) **if** B is single-colored, **then** skip B;
//Extracting no bit
(ii) **if** B is two-colored, **then**
//Extracting the embedded bit
if the move-right condition is met in B, **then** set b = '1'; **else** set b = '0'.
Step 3: if CMP_h B is *not* in the first row of modules in I, **then**
//Extracting the bit from the other rows according to rules H2 and H3
(i) **if** B is single-colored, **then** skip B;
//Extraction no bit
(ii) **if** B is two-colored, **then**
(1) **if** PMP_h A is two-colored, **then** skip B;
// Extraction no bit
(2) **if** PMP_h A is single-colored, **then**
if the move-right condition is met in B, **then** set b = '1'; **else** set b = '0';
Step 4: *//Ending of the algorithm*
exit with the extracted bit b as the desired output. ■

The above algorithm is designed for message-bit extraction from the horizontal direction. Note that the result yielded by the algorithm might be empty if no message bit was embedded

in the input MB. For message-bit extraction from the vertical direction, an algorithm, called Algorithm 5, is proposed as follows, which is almost identical to Algorithm 4 except that the bit extraction operation is carried out from the vertical direction according to rules V1 through V3 mentioned in Section III, and the move-right condition in Algorithm 4 is replaced by the move-down condition.

Algorithm 5: Extraction of a Message Bit From an MB From The Vertical Direction.
Input: the feature vectors of an MB M in a stego-image I of a QR code as shown in Fig. 26 with a message bit b embedded.
Output: the bit b.
Steps.
Step 1: *//Initialization*
1.1 denote the two MPs of the MB M in the vertical direction as PMP_v A and CMP_v B; *//Labeling the MPs in M*
1.2 set bit b to be empty initially.
Step 2: if CMP_v B is in the first column of modules in I, **then** *//Extracting the bit from the 1st column according to rule V1*
(i) **if** B is single-colored, **then** skip B; *//Extracting no bit*
(ii) **if** B is two-colored, **then** *//Extracting the embedded bit*
if the move-down condition is met in B, **then** set b = '1'; **else** set b = '0'.
Step 3: if CMP_v B is *not* in the first column of modules in I, **then**
//Extracting the bit from the other rows according to rules V2 and V3
(i) **if** B is **single**-colored, **then** skip B; *//Extraction no bit*
(ii) **if** B is **two**-colored, **then**
(1) **if** PMP_v A is two-colored, **then** skip B; *// Extraction no bit*
(2) **if** A is single-colored, **then** *//Extracting the embedded bit*
if the move-down condition is met in B, **then** set b = '1'; **else** set b = '0';
Step 4: *//Ending of the algorithm*
exit with the extracted bit b as the desired output. ■

C. FINDING THE MODULES IN A QR CODE IMAGE

Before the data extraction algorithm proposed in this study can be presented completely, one more task which need be elaborated is how to find the modules in a given stego-image. For this aim, at first the parameters w and k which are the side length of the module and the shift parameter, respectively, should be found The basic idea adopted in this study for this is to utilize the appearing frequencies of the lengths of the abundant black and white runs in the QR code image. One reason is that even when the original QR code image has been processed to embed the message bits, most of the runs are still of the side length w of the modules.

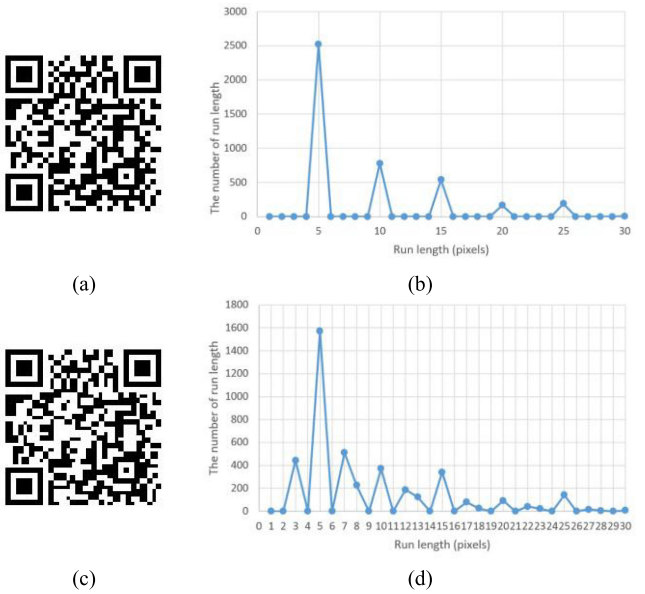

FIGURE 31. Illustrations of finding the side length w of the module and the shift parameter k for internal boundary adjustments. (a) A QR code image with side length $w = 5$. (b) Run length histogram of (a) with peaks occurring at run lengths of multiples of w, i.e., at $n \times w = 5, 10, 15, \ldots$, where n is a positive integer. (c) A stego-image with bits embedded into the image of (a) (c) Run length histogram of (c) with peaks occurring at $n \times w$ and $n \times w \pm k = 3, 5, 7, 8, 10, 12$, and so on.

In more detail, given a QR code image with the module size of $w \times w = 5 \times 5$ pixels, e. g., in which no message data are embedded like that shown in Fig. 31(a), the histogram of the run lengths found in the image is shown in Fig. 31(b), from which it can be seen that the peaks appear at the run lengths of 5, 10, 15, 20, and 25 pixels, which are all multiples of $w = 5$, i.e., are all of the lengths of $n \times w$ where n is a positive integer. This phenomenon can be figured out easily because the modules are just either black or white, and each of them has the side width of 5 pixels.

However, after message bits are embedded into the image by the proposed data embedding algorithm (Algorithm 3) with shift parameter $k = 2$ pixels (i.e., the internal boundary adjustments are rightward or downward shifts of $k = 2$ pixels) to yield a stego-image as shown in Fig. 31(c), the histogram of the run lengths of the resulting image becomes that shown in Fig. 31(d). As can be seen, the peaks appear not only at run lengths of multiples of $w = 5$, but also at run lengths with a distance of $k = 2$ from the multiples, i.e., at run lengths of the values of both $n \times w$ and $n \times w \pm k$ which are 3, 5, 7; 8, 10, 12; 13, 15, 17, and so on. Fortunately, the largest peak P_0 is still at $w = 5$, and the second largest peaks P_1 are in the range of $[w - \lfloor (w-1)/2 \rfloor, w + \lfloor (w-1)/2 \rfloor] = [3, 7]$ from which P_1 can be found out to be at 7, so that k can be computed to be $k = |P_0 - P_1| = |5 - 7| = 2$.

Based on the aforementioned observations, the following algorithm is proposed for finding the modules in a given stego-image I of a QR code with message bits embedded by

Algorithm 3 *without* the knowledge of the data of the module side length w and the shift parameter k.

Algorithm 6: Finding The Modules in The Stego-Image of a QR Code.

Input: the stego-image I of a QR code with a sequence of binary message bits embedded.

Output: the *original* modules in I with each represented by a feature vector including the coordinates of its top right and bottom left corners and its color.

//'Original' here means 'before message-bit embedding'

Steps.

Step 1: *//Constructing the run length histogram*

raster-scan the pixels of I horizontally and then vertically to find all the runs in I in the two directions, compute the length of each run, and construct accordingly a run length histogram G of I.

Step 2: *//Finding the side length w of the modules*

find the run length where the largest peak occurs in G and take it as the desired side length w of the modules in I.

Step 3:

//Finding the parameter k used in internal boundary adjustments

3.1 find in the range $[w - \lfloor (w-1)/2 \rfloor,\ w + \lfloor (w-1)/2 \rfloor]$ the run length ℓ where the second largest peak occurs;

3.2 compute $k = |w - \ell|$ and take k as the desired shift parameter.

Step 4: *//Finding the modules*

while the pixels in I is not exhausted **do:**

4.1 in a raster-scan manner, use every square of $w \times w$ pixels to form a module $M_{i,j}$ and take out the coordinates $(Hs_{i,j}, Vs_{i,j})$ and $(He_{i,j}, Ve_{i,j})$ of the top left and bottom right corners $S_{i,j}$ and $E_{i,j}$ of $M_{i,j}$, respectively;

4.2 extract the color $c_{i,j}$ of $M_{i,j}$ by voting using the pixels' colors in the area A shown in Fig. 30, which has its top left and bottom right corners being located at coordinates $(Hs_{i,j} + k, Vs_{i,j} + k)$ and $(He_{i,j}, Ve_{i,j})$, respectively;

//A is the middle area shown in Fig. 30 whose color will not be changed even after internal boundary adjustments

4.3 construct the 5-tuple $(Hs_{i,j}, Vs_{i,j}, He_{i,j}, Ve_{i,j}, c_{i,j})$ as the feature vector $f_{i,j}$ of $M_{i,j}$.

*//*Regarding *I to consist of the original modules described by such feature vectors*

end while.

Step 5: *//Ending of the algorithm*

exit with all modules $M_{i,j}$ with their respective feature vectors $f_{i,j}$ as desired output data. ■

D. DATA EXTRACTION ALGORITHM

The above two algorithms are used in the proposed message extraction algorithm as described next. It is assumed that the *maximum* length of the message, when represented as a binary value, can be specified completely by 16 bits. If the binary version of the message is shorter than 16 bits in length, then leading 0's are affixed to it to make it become so before the message is embedded. It is also assumed that the data embedded into the stego-image is a concatenation of the 16-bit message length and the binary message itself in order.

Algorithm 7: Message Extraction From a Stego-Image of a QR Code.

Input: the stego-image I of a QR code with a pre-embedded bit string which is a concatenation of 16 bits specifying the bit length of a message S of characters followed by the binary version of S.

Output: the message S.

Steps.

Step 1: *//Finding the modules in image I*

perform Algorithm 6 with I as the input to yield all the original modules $M_{i,j}$ in I with respective feature vectors $f_{i,j}$.

Step 2: *//Extracting the leading N message bits*

2.1 create a bit string S_0, set to be empty initially;

2.2 create a counter n to count the number of extracted bits so far, set to be 0 initially;

2.3 create two matrices E_h and E_v both of the same size of I but in the unit of MB and write the label P into those cells of E_h and E_v, which correspond to the locations of the position and alignment patterns;

//E_h and E_v are created for verifying correctness of extracted bits from the two directions.

2.3 **while** $n \neq 16$ **do:**

//Assuming the 16-bit message length were embedded horizontally

(a) perform horizontal MB generation to construct sequentially an unprocessed MB B from I;

(b) perform Algorithm 4 with the 5-tuple feature vectors of all the modules in B as the input to extract a bit b from B; *//b might be empty*

(c) **if** b is not empty, **then** append b to S_0, write b into the cell C in matrix E_h corresponding to B, and set $n = n + 1$;

else, write the label m into C **if** the CMP in B is used for matching adjustment; or write the label s into C **if** the CMP is single-colored and so untouched;

end while;

2.4 transform the 16 bits in S_0 into a decimal value m.

//m specifies the length of the binary string of the embedded message

Step 3: *//Extracting the m embedded message bits*

3.1 reset the bit string S_0 to be empty;

3.2 **while** $m \neq 0$ and the horizontal MBs in I are not processed exhaustively **do:**

//Extracting bits horizontally where 'm ≠ 0' means 'bits not all extracted yet'

(i) perform *horizontal* MB generation to construct sequentially an *unprocessed* MB B from I;

(ii) perform Algorithm 4 with the 5-tuple feature vectors of B as the input to extract a bit b from B; *//b might be empty*

(iii) **if** b is not empty, **then** append b to S_0, write b into the cell C of matrix E_h corresponding to MB B, and set $m = m - 1$;

The creation of the matrices E_h and E_v in the above algorithm, which are called the *verification matrices* subsequently,

else, write the label *m* into *C* **if** the CMP in *B* is used for matching adjustment; or write the label *s* into *C* **if** the CMP is single-colored and so untouched;

end while;

3.3 **while** $m \neq 0$ and the vertical MBs in *I* are not processed exhaustively **do:**

//Extracting bits vertically where 'm ≠ 0' means 'bits not all extracted yet'

(i) perform *vertical* MB generation to construct sequentially an *unprocessed* MB *B* from *I*;

(ii) perform Algorithm 5 with the 5-tuple feature vectors of *B* as the input to extract a bit *b* from *B*; *//b might be empty*

(iii) **if** *b* is not empty, **then** append *b* to S_0, write *b* into the cell *C* of matrix E_v corresponding to MB *B*, and set $m = m - 1$;

else, write the label *m* into *C* **if** the CMP in *B* is used for matching adjustment; or write the label *s* into *C* **if** the CMP is single-colored and so untouched;

end while;

end while.

Step 4: *//Ending of the algorithm*

transform the bit string S_0 into characters to become the desired message *S* and exit with *S*, E_h, and E_v as the output. ■

are aimed at recording the *module-processing information* of how every MP in the stego-image *I* is processed in the data extraction process. More specifically, with its size identical to *I* in the unit of MP, either of E_h and E_v is used to record module-processing information which includes the extracted bit '0' or '1' as well as three types of label, namely, '*P*', 'm', and 's', which means respectively position or alignment pattern, matching adjustment, and single-colored pattern, all having been explained before.

E. AN EXAMPLE OF MESSAGE EXTRACTION RESULTS

As an example of message extraction results, shown at first in Fig. 32(b) is a stego-message resulting from applying the message embedding algorithm, Algorithm 3, to the input cover image of a version-3 QR code shown in Fig. 32(a) to embed a 40-character secret message: 'Account: Da-Chun Wu Password: love&peace'. The facial data recorded in the cover image of the QR code is the domain name 'nkust.edu.tw'. As can be seen, the stego-image looks quite similar to the cover image, although there indeed exists difference between them as shown by Fig. 32(c) which is a difference image between the two. This difference image was then superimposed onto the cover-image to yield the overlapping image shown in Fig. 32(d) which shows more clearly how the cover image were processed for message embedding to yield the stego-image.

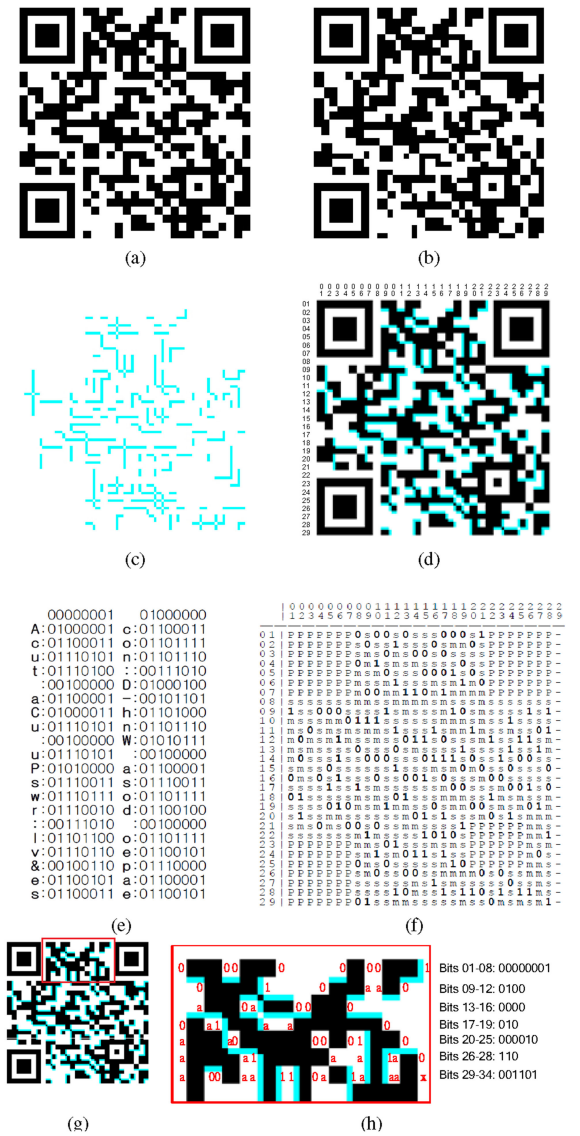

FIGURE 32. **Illustrations of message embedding and extraction results. (a) Input cover image of a version-3 QR code. (b) Result yielded by the proposed message embedding process (Algorithm 3) with (a) as the input to embed a 320-bit message with overlaps and holes eliminated. (c) Difference image of (a) and (b). (d) Overlapping image of (a) and (c). (e) Input bit string resulting from created by combining the 16-bit message length string and the 320-bit message string in order. (f) First half of the verification matrix E_h yielded by the proposed message extraction process (Algorithm 7), which matches correctly with the first half of (e). (g) Part of the overlapping image of (d) enclosed by the red rectangular window. (h) Detailed labels of the extracted bits and the matching adjustments showing the operations applied correctly by the proposed message extraction process to the MPs in the enclosed part in (g).**

The binary version of the aforementioned 40-character secret message embedded in the stego-image is a 320-bit string. The decimal value, 320, which is the bit length of the secret message, was transformed in this study into a 16-bit string and then prefixed to the 320-bit secret message string to form the binary message data embedded in the stego-image, which is shown in Fig. 32(e). The verification matrix E_h yielded by the message extraction algorithm during horizontal message-bit

TABLE 6. Bit Embedding Rates of Various QR Code Versions Yielded by Proposed Message-Bit Embedding Algorithm

Version	No. of modules (A)	No. of position patterns	No. of alignment patterns	No. of modules for embedding	Embedding capacity B (bits) (B = average #bits of L, M, Q, and H)				No. of bits per module (A/B)
					Error correction level				
					L (7%)	M (15%)	Q (25%)	H (30%)	
1	441	3	0	294	140	128	135	131	0.303
5	1,369	3	1	1,213	646	645	630	607	0.462
10	3,249	3	6	3,048	1,729	1,403	1,428	1,445	0.462
20	9,409	3	13	9,145	4,235	4,294	4,465	4,422	0.463
30	18,769	3	33	18,325	8,303	8,408	9,243	8,780	0.463
40	31,329	3	46	30,768	13,635	13,960	14,460	14,289	0.450

extraction (including the extracted message bits of '0' and '1' in bold in the figure) is shown in Fig. 32(f) which can be seen to be correct when compared with the first half of the message data shown in Fig. 32(e). As to the verification matrix E_v, it is omitted to save space in this paper. Furthermore, in Fig. 32(g) an enlarged part of the overlapping image of Fig. 32(d) is shown, and in Fig. 32(h) the details of the operations applied to the MPs in the enlarged part are shown. In Fig. 32(h), in addition to showing the extracted bits of '0' and '1', the operations of matching adjustments are labeled by 'a' (instead of 'm' in Fig. 32(f)). Finally, it is mentioned that no label is attached to those MPs which are single-colored and untouched.

VI. EXPERIMENTAL RESULTS

A. BIT EMBEDDING RATES YIELDED BY USE OF VARIOUS QR CODE VERSIONS

A series of experiments has been conducted in this study to test the bit embedding rate of the proposed message-bit embedding algorithm (Algorithm 3) for each of six versions (versions 1, 5, 10, 20, 30, and 40) of the QR code using the four error correction levels (L, M, Q, and H). The input to each case is a sufficiently long sequence of random bits, and data embedding was conducted horizontally and then vertically until no more bit can be embedded into the image. The results are shown in Table 6, from which the following facts can be observed.

1) The numbers of modules that appear in different versions of QR code images are different (from 441 to 31,329) so that the numbers of modules used in the data embedding process and the resulting bit embedding capacities are also different.
2) However, for each of the four error correction levels, L, M, Q, and H, the resulting data embedding capacity is *roughly* identical; i.e., the capacity *almost* has nothing to do with the error correction level. This can be figured out because in each case of the four levels, an identical number of modules (and so an identical number of MBs) are used for data embedding, though the modules are different in their contents.
3) The data embedding capability computed in terms of bits is roughly a half of the number of modules used for data embedding in the horizontal and vertical directions.
4) Finally, excluding the data of the version-1 QR code, the most noticeable is that the *bit embedding rate*, which is computed as the *average number of embedded bits per module* (shown in the last column in the table) in the following way, is *roughly identical* ($\cong 0.46$):

$$\begin{aligned} \textit{bit embedding rate} &= \textit{number of bits per module} \\ &= B/A \end{aligned}$$

where

A = *number of modules*
B = *average number of embedded bits of the four error correction levels*

5) An exception of the observation of (4) is the case of using a version-1 QR code image as the input to yield a bit embedding rate of about 0.30 that is lower than 0.46. The reason is that the version-1 QR code image is small in size in which the three position patterns with fixed and relatively large sizes are excluded from being used for data embedding, so that the remaining image part is proportionally very small for data embedding when compared with images of the other five versions.

It is mentioned by the way that given the image of a QR code with its version, error correction level, as well as recorded facial data being fixed, the number of modules in the image will also be fixed; therefore, no matter what message is embedded into the image, the amount of the embedded bits and so the number of embedded bits per module will roughly be the same.

B. TESTS OF STEGO-IMAGE READABILITY USING A BARCODE READER

Two major parameters used in this study are the side length w of a module in the QR code as well as the shift parameter k for internal boundary adjustment. When a bit of '1' is embedded into an MP, the internal boundary between the two modules in the MP is shifted rightward or downward for the amount of k pixels as described previously in Section III. Though the stego-image yielded by the proposed method can be read by a barcode reader when k is small with respective to w, yet there is a limit on how large the ratio $r = k/w$ can be, which need be found out, as done in this study.

Specifically, a series of experiments have been conducted by varying the values of w and k to produce stego-images of QR codes of versions 5 and 10 using the proposed message embedding process, Algorithm 3. Each of the stego-images is then scanned using three commercially available barcode reading tools: (1) Barcode Scanner GT10Q-SR manufactured by the Denso Wave Company; (2) Barcode Scanner App developed by the Zxing Team (run on a Samsung smartphone of model NOTE-9); and (3) QR Code Reader App developed

TABLE 7. Results of Tests of Stego-image Readability by Use of Barcode Readers of Different Brands

QR code type	w	Testing Equipment	k			
			1	2	3	4
5-L	9	Denso	○	○	○	×
		Samsung	○	○	○	×
		iPad	○	○	○	×
5-Q	9	Denso	○	○	○	**○**
		Samsung	○	○	○	×
		iPad	○	○	○	×
10-L	5	Denso	○	×	-	-
		Samsung	○	×	-	-
		iPad	○	**○**	-	-
10-Q	5	Denso	○	×	-	-
		Samsung	○	×	-	-
		iPad	○	×	-	-

by the Komorebi Inc. (run on an Apple iPad Pro of model A1584). The aim of the experiments is to see how large the ratio value $r = k/w$ can be as long as the stego-image is still readable by each of the three types of barcode reader. In the experiments, the minimum value of w is set to be 4, and k is selected to be smaller than $w/2$ (or equivalently, k/w is selected to be smaller than 1/2).

The results yielded by the experiments of this test of stego-image readability using the barcode readers are summarized in Table 7, where the QR code type is specified as v-L with v specifying the version of the QR code and L denoting one of the four error correction levels, with the symbols '○' meaning 'readable', ' × ' meaning 'failing to read', and '-' denoting 'not conducted (because $k/w \geq 1/2$)'.

The facial data recorded in the QR code of types 5-L and 5-Q is 'Da-Chun Wu and Yuan-Ming Wu are with NKUST' and the embedded message is 'A new data hiding method for use in covert communication via the QR code ' where the last character is a bank space. And the facial data recorded in the QR code of types 10-L and 10-Q is 'Da-Chun Wu and Yuan-Ming Wu are with the Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology' the embedded message is 'A new data hiding method for use in covert communication via the QR code image based on adjustments of the shapes of the QR code modules using image processing techniques is ' where the last character is again a bank space.

From the table, the following facts can be observed.

1) When $w = 9$, and $k = 1$ through 3, the facial information recorded in the stego-image of the QR code can be read successfully by the barcode readers developed by all the three companies.
2) When $w = 5$, only when $k = 1$ can the facial information recorded in the stego-image be read out successfully by the barcode readers developed by all the three companies.
3) When $w = 9$ and $k > 3$, and when $w = 5$ and $k > 1$, the QR code reading operations all failed no matter what barcode reader is used, with two exceptions marked by bold symbols of '○' in the table.
4) The reason for (3) above to occur is easy to induce: the ratio $r = k/w$ is *too large* so that the internal boundary adjustment in the MP during embedding of the bit of '1' causes too serious module shape distortions which are intolerable by the barcode readers.
5) The limit of the ratio $r = k/w$ mentioned above is roughly 1/3 or 33%, i.e., when $r > 33\%$, the barcode reader cannot read out the facial information recorded in the QR code. Fortunately, the proposed method, on the contrary, *can* extracted the embedded data no matter whether the facial information can be read out or not.

C. TESTS OF RESISTANCE OF THE PROPOSED MESSAGE EXTRACTION PROCESS AGAINST RANDOM NOISE

A series of experiments have been conducted to test the resistance ability of the proposed message extraction process (Algorithm 7) against random noise 'attacks'. Specifically, the black and white pixels in a stego-image of a QR code are flipped randomly to simulate random noise which 'attacks' the stego-image and so hinders successful extraction of the embedded data. The noise is added to the stego-image in a gradual increase manner, starting from the amount of 0.1% of the pixels in the stego-image, with an increment of 0.1%, until the proposed method cannot extract the message correctly.

An example of noisy stego-images so generated of a 5-L QR code with $w = 9$ and $k = 2$ is shown in Fig. 33, where Fig. 33(a) shows a stego-image with 8% noise added, from which the embedded message still can be extracted by the proposed message extraction algorithm (Algorithm 7), and Fig. 33(b) shows one with 8.1% noise added, from which message extraction fails. A difference image of the two noisy stego-images is shown in Fig. 33(c) which looks like a pepper-and-salt image; and an enlarged image part of the bottom left position pattern in Fig. 33(b) is shown in Fig. 33(d), in which the two red pixels are the additionally generated noise pixels when the noise is increased from 8% to 8.1%. The percentage of noise which is tolerable by the proposed message extraction process, like 8% of this example here, is called the *maximum tolerable noise rate* subsequently.

The overall experimental results are summarized a table, namely, Tables 8 in the following where the symbol '-' denotes 'not carried out' (because the ratio $= k/w$ is too large). From the data shown in the table, the following facts can be observed.

1) When the side length w of the modules is small, the maximum noise tolerable rate is small as well, as can be seen from the last two rows in the table.

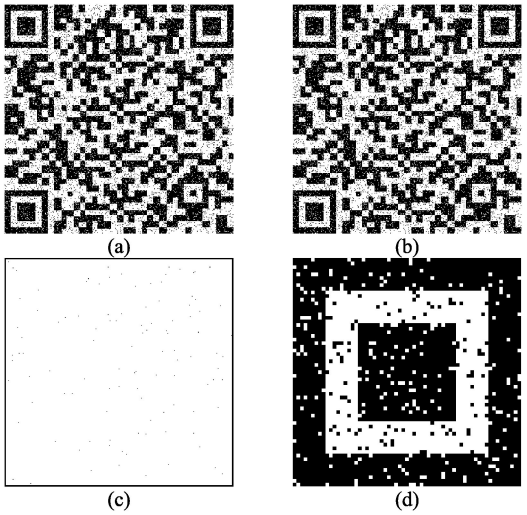

FIGURE 33. Noisy stego-images of a 5-L QR code used in the tests of the proposed method against noise attacks. (a) A stego-image with 8% noise added from which the embedded message still can be extracted. (b) A stego-image with 8.1% noise from which extraction of the message fails. (c) A difference image of (a) and (b) which looks like a pepper-and-salt image. (d) An enlarged image of the bottom left position pattern in (b) in which the two red pixels are the additionally added noise pixels when the noise is increased from 8% to 8.1%.

TABLE 8. Maximum Tolerable Noise Rates Obtained by Tests of Resistance of the Proposed Message Extraction Process to Noise Attacks

QR code type	w	k			
		1	2	3	4
5-L	9	7.1%	8.0%	8.1%	8.0%
5-Q	9	8.3%	8.1%	8.2%	8.2%
10-L	5	1.8%	3.7%	-	-
10-Q	5	2.0%	3.6%	-	-

2) When the side length of the modules is large like $w = 9$, then for all $k = 1$ through 4, the maximum noise tolerable rate is about the same (in the small range of 8.0 to 8.3 with only one exception) with no relation to the error correction level, as can be seen from the first two rows in the table.
3) Failure of message extraction occurs when the 'attacking' noise exceeds the maximum tolerable rate in all the cases shown by the table, which may be attributed to the erroneous voting results about the colors (black or white) of the module areas A, B, and C mentioned in Step 5.2 of Algorithm 6 and shown in Fig. 30, leading to incorrect module color recovery and so incorrect message bit extraction.

D. TESTS OF STEGO-IMAGE READABILITY USING BARCODE READERS UNDER MAXIMUM TOLERABLE NOISE RATES

Further experiments have been conducted to test the readability of the stego-images yielded by the proposed message extraction process (Algorithm 7) under the maximum tolerable noise rate. The results are shown in Table 9 from which we can see that the resulting readabilities of all the cases shown in the table are the same as those of Table 7 except two cases: QR code type 5-Q with $w = 9$, $k = 4$ using the Densor barcode reader and QR code type 10-L with $w = 5$, $k = 2$ using the iPad. In each of these two exceptional cases, the stego-image, being readable originally, becomes unreadable due to the addition of the flipping noise.

TABLE 9. Results of Tests of Stego-Image Readability by Use of Barcode Readers of Different Brands Under the Maximum Tolerable Noise Rates

QR code type	w	Testing Equipment	k			
			1	2	3	4
5-L	9	Denso	○	○	○	×
		Samsung	○	○	○	×
		iPad	○	○	○	×
5-Q	9	Denso	○	○	○	×
		Samsung	○	○	○	×
		iPad	○	○	○	×
10-L	5	Denso	○	×	-	-
		Samsung	○	×	-	-
		iPad	○	×	-	-
10-Q	5	Denso	○	×	-	-
		Samsung	○	×	-	-
		iPad	○	×	-	-

E. COMPARISON WITH OTHER METHODS ABOUT THE MESSAGE-BIT EMBEDDING CAPACITY AND RATE

Also conducted in this study is a comparison of the message-bit embedding capacity and the number of bits per module (bpm) of the proposed method with those of four other related methods, namely, Chiang *et al.* [9], Lin [22], Lin and Chen [12], and Huang *et al.* [15], which have been surveyed in Section II. In detail, the message-bit embedding capacities yielded by the four other methods taking as inputs the images of five versions of the QR code with all of the four error correction levels were collected from the literature [9], [12], [15], [22], and the corresponding numbers of the bpm were computed and compared with those yielded by the proposed method. The results are summarized in Table 10. Here the number of bpm is computed as the ratio of the message-bit embedding capacity to the number of modules in the QR code.

From the table, it can be seen that for each possible variation of the QR code type, the proposed method outperforms the other four methods to yield both the largest message-bit embedding capacity and the largest number of bpm. Excluding the data of the version-1 QR code, the number of bpm yielded by the proposed method, on the average, is about 0.46.

The superiority of the proposed method to the other four methods in the aspect of comparing the numbers of bpm mainly comes from the *direct use of nearly all* the black and

TABLE 10. Comparison of Message-Bit Embedding Capacities and Rates of Five Methods

Version	No. of modules in QR code	Error correction level	Chiang et al. [9] & Lin [22]		Lin & Chen [12]		Huang et al. [15]		Proposed method	
			capacity (bits)	# of bpm	capacity (bits)	# of bpm	capacity (bits)	# of bpm	capacity (bits)	# of bpm
1	441	L	24	0.05	50	0.11	6	0.01	140	0.32
		M	40	0.09	83	0.19	15	0.03	128	0.29
		Q	48	0.11	97	0.22	24	0.05	135	0.31
		H	64	0.15	72	0.16	30	0.07	131	0.30
10	3,249	L	288	0.09	613	0.19	144	0.04	1,729	0.53
		M	520	0.16	1,086	0.33	258	0.08	1,403	0.43
		Q	768	0.24	1,232	0.38	384	0.12	1,428	0.44
		H	896	0.28	976	0.30	447	0.14	1,445	0.44
20	9,409	L	896	0.10	1,853	0.20	447	0.05	4,235	0.45
		M	1,664	0.18	3,474	0.37	831	0.09	4,294	0.46
		Q	2,400	0.26	3,880	0.41	1,200	0.13	4,465	0.47
		H	2,800	0.30	3,080	0.33	1,398	0.15	4,422	0.47
30	18,796	L	1,800	0.10	3,735	0.20	900	0.05	8,303	0.44
		M	3,248	0.17	6,752	0.36	1,623	0.09	8,408	0.45
		Q	4,800	0.26	7,880	0.42	2,400	0.13	9,243	0.49
		H	5,760	0.31	5,960	0.32	2,880	0.15	8,780	0.47
40	31,329	L	3,000	0.10	6,249	0.20	1,500	0.05	13,635	0.44
		M	5,488	0.18	11,445	0.37	2,742	0.09	13,960	0.45
		Q	8,160	0.26	13,328	0.43	4,080	0.13	14,460	0.46
		H	9,720	0.31	10,208	0.33	4,860	0.16	14,289	0.46

Note: 'bpm' means 'bits per module'.

white modules (with the exceptions being those of the position and alignment patterns) in the QR code image for message-bit embedding. It can be figured out that when the input message bits are random enough, the number of modules used for message-bit embedding, seen from an overview, is about a half of all the modules in the QR code, leading to the yield of the number of bpm of about 0.50. But due to the large-sized position patterns and the multiple alignment patterns whose modules are not used for bit embedding, the number of bpm reduces to a rough value of 0.46, as can be seen from the table. On the other hand, the other four compared methods are all based on modifying or augmenting the error correction scheme inherent of the QR code to embed secret messages. The percentage of the modules in the QR code assigned for error correction lies in the range of 7% to 30% of the codewords representing the facial data in the QR code, which are just *part of* the modules in the QR code. Therefore, the maximum numbers of bpm yielded by these methods in general are smaller than those yielded by the proposed method.

VII. CONCLUSION

In this paper, a novel method for secret message hiding via QR code images for covert communication applications has been proposed, which includes processes for message-bit embedding and embedded-bit extraction. Message hiding is achieved by a new technique of adjusting the internal boundary in the module pairs in the QR code image for message-bit embedding according to the bit value to be embedded. Such a bit embedding process may be conducted in both the horizontal and the vertical directions to double the bit embedding capacity. The technique makes a nearly full use of the modules in the QR code image, in contrast with many other data hiding methods based on manipulating the error correction scheme of the QR code, in which at most 30% of the modules of the codewords representing the QR data are utilized. As a consequence, the proposed method has a better performance than these methods in the yield of the bit embedding rate (measured in terms of the number of embedded bits per module), as shown by a comparison conducted in this study about the yielded numbers of bpm.

Specifically, except for the version-1 QR code image, the proposed method yields a bit embedding rate of about 0.46 per module for all types of QR code images, which is not achieved by other methods. This merit provides quite a large amount of space for secret message hiding, making the proposed method advantageous for covert communication applications. Also, the method has a property showing that the larger the version number of the QR code, the larger the resulting message hiding space. For example, up to about 14,000 bits may be provided by the proposed method if a version-40 QR code image is adopted as the cover image. In addition to being considered as a scheme for secret message hiding, the proposed method may be regarded as well as an extension of the QR data recording scheme from the viewpoint of increasing the data capacity in the QR code.

Though the internal boundaries in the module pairs of a QR code image are adjusted by message-bit embedding process in the proposed method, the resulting stego-image can be scanned by a common barcode scanner to read out the facial data recorded in the QR code as long as the ratio k/w of the shift parameter k to the side width w of the QR code is kept no larger than 1/3. Furthermore, the yielded stego-image is resistant to attacks of up to 8% pepper-and-salt noise.

On the other hand, the embedded message is invisible to common people and can only be extracted by a program implementing the proposed message extraction process. In addition, if higher data security is required, the binary version of the message may be randomized by a random number generator controlled by a secret key before the message is embedded into the QR code image. It is also mentioned that the proposed method may as well be applied to other 2D barcodes with *rectangular* module shapes.

Good experimental results and the aforementioned advantages and superiority of the proposed method show its feasibility for real applications of covert communication.

REFERENCES

[1] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—A survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, Jul. 1999.

[2] I. Cox, M. Miller, J. Bloom, J. Fridrich and T. Kalker, *Digital Watermarking and Steganography*. Burlington, MA, USA: Morgan Kaufmann, 2007.

[3] *Information Technology—Automatic Identification and Data Capture Techniques—QR Code 2005 Bar Code Symbology Specification*, ISO/IEC 18004:2006, 2006.

[4] K. H. Pandya1 and H. J. Galiyawala, "A survey on QR codes in context of research and application," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 3, pp. 258–262, Mar. 2014.

[5] A. Gaikwad and K. R. Singh, "Information hiding using image embedding in QR codes for color images: A review," *Int. J. Comput. Sci. Inf. Technologies*, vol. 6, no. 1, pp. 278–283, 2015.

[6] G. J. Garateguy, G. R. Arce, D. L. Lau, and O. P. Villarreal, "QR images: Optimized image embedding in QR codes," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 2842–2853, Jul. 2014.

[7] S. Zhang and K. Yoshino, "DWT-based watermarking using QR code," *Sci. J. Kanagawa University*, vol. 19, pp. 3–6, 2008.

[8] S. Dey, K. Mondal, J. Nath, and A. Nath, "Advanced steganography algorithm using randomized intermediate QR host embedded with any encrypted secret message: ASA_QR Algorithm," *Int. J Modern Educ. Comput. Sci.*, vol. 6, pp. 59–67, 2012.

[9] Y. J. Chiang, P. Y. Lin, R. Z. Wang, and Y. H. Chen, "Blind QR code steganographic approach based upon error correction capability," *KSII Trans. Internet Inf. Syst.*, vol. 7, no. 10, pp. 2527–2543, Oct. 2013.

[10] A. M. Barmawi and F. A. Yulianto, "Watermarking QR code," in *Proc. 2nd Int. Conf. Inf. Sci. Secur. (ICISS '15)*, Seoul, South Korea, Dec. 14–16, 2015.

[11] J. Fridrich, M. Goljan, and D. Soukal, "Wet paper codes with improved embedding efficiency," *IEEE Trans. Inf. Forensics Secur.*, vol. 1, no. 1, pp. 102–110, Mar. 2006.

[12] P. Y. Lin and Y. H. Chen, "High payload secret hiding technology for QR codes," *EURASIP J. Image Video Process.*, vol. 2017, p. 14, 2017.

[13] K. Kamijo, N. Kamijo, and Z. Gang, "Invisible barcode with optimized error correction," in *Proc. 15th IEEE Int. Conf. Image Process.*, San Diego, CA, USA, pp. 2036–2039, Oct. 12–15, 2008.

[14] P. C. Huang, Y. H. Lin, C. C. Chang, and Y. Liu, "Efficient scheme for secret hiding in QR code by improving exploiting modification direction," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 5, pp. 2348–2365, 2018.

[15] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Commun. Lett.*, vol. 10, no. 11, pp. 781–783, Nov. 2006.

[16] S. Rungraungsilp, M. Ketcham, V. Kosolvijak, and S. Vongpradhip, "Data hiding method for QR code based on watermark by compare DCT with DFT domain," in *Proc. Int. Conf. Comput. Commun. Technologies*, Visakhapatnam, India, 2012, pp. 144–148.

[17] M. Sun, J. Si and S. Zhang, "Research on embedding and extracting methods for digital watermarks applied to QR code images," *New Zealand J. Agricultural Res.*, vol. 50, pp. 861–867, 2007.

[18] L. Li, R. L. Wang, and C. C. Chang, "A digital watermark algorithm for QR code," *Int. J. Intell. Inf. Process.*, vol. 2, no. 2, pp. 29–36, 2011.

[19] Y. Cheng, Z. Fu, and B. Yu, "Improved visual secret sharing scheme for QR code applications," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 9, pp. 2393–2403, Sep. 2018.

[20] Y. Chow, W. Susilo, G. Yang, J. G. Phillips, I. Pranata, and A. Barmawi, "Exploiting the error correction mechanism in QR codes for secret sharing," in *Proc. 21st Australasian Conf. Inf. Secur. Privacy*, Melbourne, Australia, 2016, vol. 9722, pp. 409–425.

[21] I. Tkachenko, W. Puech, C. Destruel, O. Strauss, J.-M. Gaudin, and C. Guichard, "Two-level QR code for private message sharing and document authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 3, pp. 571–583, Mar. 2016.

[22] P. Y. Lin, "Distributed secret sharing approach with cheater prevention based on QR code," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 384–392, Feb. 2016.

[23] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary image," in *Proc. IEEE Int. Conf. Multimedia Expo.*, 2000.

[24] H. Gou and M. Wu, "Improving embedding payload in binary images with super-pixels," *Proc. Int. Conf. Image Process.*, vol. 3, no. 1, pp. 277–280, Sept. 16–Oct. 19, 2007.

[25] K. Sayood, *Introduction to Data Compression*. Morgan Kaufmann Publishers Book Company, 2000.

[26] D. C. Wu, M. K. Hsu, and J. H. Jheng, "Data hiding and authentication techniques for 2-color digital documents based on adjusting lengths of runs," in *Proc. 2003 Conf. Comput. Vision, Grap. Image Process.*, Kinmen, Taiwan, Republic of China, 2003, pp. 818–822.

[27] G. Xuan, Y. Q. Shi, P. Chai, X. Tong, J. Teng, and J. Li, "Reversible binary image data hiding by run-length histogram modification," in *Proc. 19th Int. Conf. Pattern Recognit.*, Tampa, FL, USA, Dec. 8–11, 2008.

[28] Wikipedia, "QR code," [Online]. Available: https://en.wikipedia.org/wiki/QR_code Accessed: Oct. 15, 2019.

DA-CHUN WU was born in Taiwan, in 1959. He received the B.S. degree in computer science and the M.S. degree in information engineering from Tamkang University, Taipei, Taiwan, in 1983 and 1985, respectively, and the Ph.D. degree in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, in 1999. He joined the faculty of the Department of Information Management with Ming Chuan University, Taipei, Taiwan, in 1987. From 2002 to 2018, he was with the National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan. From 2010 to 2014, he was the Director of Library and Information Center of the university, and from 2015 to 2018, he was the Head of the Department of Computer and Communication Engineering. He is currently an Associate Professor with the Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan. His recent interests include multimedia security, image processing, machine learning, and artificial intelligence.

YUAN-MING WU received the B.S. degree from the Department of Computer Science and Information Engineering Ming Chuan University, Taoyuan, Taiwan, in 2005 and the M.S. degree from the Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 2009. He is currently an IT Engineer with the Wholegene Co., Ltd. in Taiwan. His current research interests include information hiding and web technology.