applied sciences

MDPI

Article

Information Hiding in QR Code Images via Module Content Modification Based on Corner-Pixel Grayscale Adjustment

Da-Chun Wu * and Yuan-Ming Wu

Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 824005, Taiwan; 9551843@nkust.edu.tw

* Correspondence: dcwu@nkust.edu.tw; Tel.: +886-7-6011000 (ext. 32019)

Abstract

QR codes are widely used in digital authentication, mobile payment, access control, and information exchange, making QR code-based covert communication and hidden message delivery relevant to cybersecurity applications. This study proposes a QR code-based information-hiding method via module content modification based on corner-pixel grayscale adjustment. Secret bits are embedded into a QR code image through subtle grayscale adjustments of module corner pixels. To reduce visually noticeable embedding artifacts, embedding in adjacent module corners that share identical original grayscale values is prohibited, thereby preventing noticeable local contrast that could reveal hidden content. Gradient-based smoothing is further applied to suppress contrast artifacts within module interiors after embedding. A theoretical analysis of bilinear interpolation and quarter subregion averages is presented to justify the proposed extraction thresholds. Experiments across multiple QR code versions, error-correction levels, and embedding parameters show that the proposed method achieves an average embedding rate of approximately 1.9 bits per module while generally maintaining QR code readability across the tested commercial scanning devices. Exploratory random-noise experiments further show that hidden-message recovery remains possible up to the observed highest tolerable noise rates in the tested realizations. A comparison with four representative methods indicates that the proposed method achieves substantially higher embedding rates under the adopted comparison framework, demonstrating its potential for high-capacity QR code information-hiding while enabling successful hidden-message recovery under the evaluated conditions.

Keywords: cybersecurity; information hiding; QR code image; module content modification; corner-pixel grayscale adjustment; gradient operation

Check for updates

Academic Editor: Karim Elish

Received: 7 July 2026
Revised: 14 August 2026
Accepted: 25 August 2026
Published: 26 August 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Information hiding through various types of multimedia has been proposed for the protection of secret messages in covert communication applications [1–4]. Secret messages can be embedded into different types of cover media using information-hiding techniques, resulting in stego-media visually similar to the original cover media. In this study, information hiding is investigated using quick response (QR) code images [5] as cover media. The primary reason for the use of QR code images is their widespread adoption across various applications.

While existing studies on QR codes have primarily focused on enhancing code readability and expanding their applications [6,7], research on information hiding via QR codes

can be categorized into at least five groups: *message-carrier hiding*, *data or error correction codeword modification*, *watermark embedding*, *information sharing*, *and use of encoding regions or coding rules*.

In the first category, QR code images are not used as cover images; instead, secret messages are encoded into QR codes, which are then embedded into other images using traditional information-hiding techniques [8–10]. In the second category, data or error-correction codewords within QR codes are modified in various ways for information hiding [11–15]. The third category consists of methods in which watermarks are embedded into QR code images through various image transformation techniques [16–18]. The fourth category includes approaches where different information-sharing techniques are employed to embed messages into QR code images [19–21]. Finally, in the fifth category, the encoding regions or coding rules of QR codes are utilized in different ways to embed message bits [22–24].

Another line of research hides information by modifying the internal content or fine-scale structure of QR code modules while preserving their module-level black/white interpretation. Teraura and Sakurai [25] subdivided each module into finer subcells and embedded additional information through modifications of the subcell pattern. Erlangga and Barmawi [26] extended this subcell-based approach by using four selected subcells in a 3×3 subcell structure, thereby providing four times the embedding capacity of the earlier subcell-based method. Wu and Wu [27] later proposed a boundary-based method for module content modification in which message bits are embedded by adjusting the internal boundaries formed by pairs of adjacent modules. For discussion purposes, these approaches are collectively referred to in this paper as methods based on module content modification. Ref. [27] serves as a direct methodological precursor to the present study; however, the proposed method differs substantially from it in both the embedding carrier and the basic processing structure. In Wu and Wu's method [27], the embedding carrier is the internal boundary formed by a pair of adjacent modules, whereas in the proposed method, it is an individual corner pixel within a module. Accordingly, the basic processing structure shifts from a module pair and its shared boundary to a module corner and its associated quarter subregion.

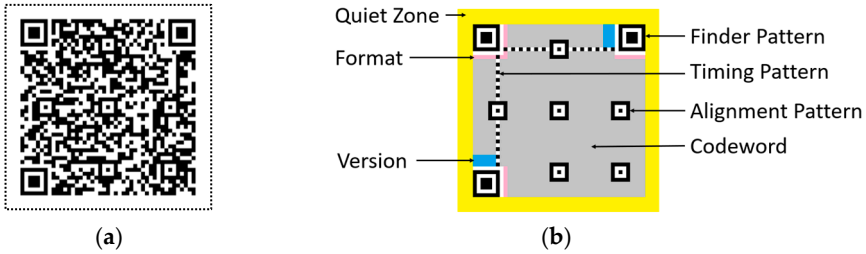
The main scientific contributions of the present study are summarized as follows:

1. A new QR code information-hiding method based on corner-pixel grayscale adjustment is proposed to achieve high embedding capacity while generally maintaining QR code readability under the evaluated conditions;
2. A gradient-smoothing operation is introduced to reduce the visual contrast caused by the modification of module contents;
3. A theoretical model of bilinear interpolation and quarter-subregion averaging is developed to provide a formal basis for the message-extraction decision thresholds under the assumed bilinear-interpolation model;
4. Experiments are conducted to evaluate the embedding performance of the proposed method across different QR code conditions and to compare its embedding capacity with selected existing methods.

The remainder of this paper is organized as follows. In Section 2, an overview of the QR code structure is provided. In Section 3, the proposed data-hiding method via module content modification is described. The experimental results are reported in Section 4, including results yielded by the proposed data embedding and extraction algorithms, evaluations of stego-image readability with scanners, exploratory tests of hidden-message extraction under random noise, and a comparison of embedding efficiency with related existing approaches. Finally, Section 5 concludes the paper with closing remarks.

2. An Overview of QR Code Architecture

As illustrated in Figure 1, a QR code consists of a matrix of square black and white modules, consistent with the QR Code symbology specification [5]. In the image representation considered in this study, each module consists of $w \times w$ pixels, where w is a positive integer. There are 40 versions of QR codes, with version 1 containing 21×21 modules, version 2 containing 25×25 modules, and so on, up to version 40, which consists of $(21 + 4 \times 39) \times (21 + 4 \times 39)$ modules, or equivalently, 177×177 modules.



(**a**) (**b**)

Figure 1. Illustration of QR code composition. (**a**) Version-7 QR code instance. (**b**) Structural diagram showing the main functional components.

In Figure 1, three corners of the QR code contain a 7×7-module pattern known as a finder pattern, which assists in positioning the QR code during scanning by a QR code reader. Additionally, each QR code includes 8-bit codewords used in the Reed–Solomon error-correction algorithm, which supports four error-correction levels: L (low), M (medium), Q (quartile), and H (high). These levels correspond to the restoration of approximately 7%, 15%, 25%, and 30% of the codewords, respectively, in cases where errors occur during QR code reading [5].

3. Data Hiding by Module Content Modification

3.1. Design Principles of the Proposed Method

The proposed method for embedding a message into the modules of a QR code image is based on the following ideas.

(B1) *Message-bit embedding*—Embed a message bit b into one of the four corner pixels of each module M, whose grayscale value v is either black (B) or white (W), where B = 0 and W = 255, according to the following rule, where k_b and k_w are two preselected fixed values between 1 and 127, referred to as the *embedding change parameters* of black or white corner pixels

if $b = 1$, **then**
{set $v = v + k_b$ **if** v = B; or set $v = v - k_w$ **if** v = W};
else leave v unchanged.

The applicability of Idea (B1) arises from the error-tolerance capability of QR codes, which allows moderate grayscale modifications of module corner pixels to be tolerated under suitable parameter settings. However, implementing Idea (B1) results in modules with contrasting corner pixels, which differ in color from the rest of the module, as illustrated in Figure 2a,b. This visual inconsistency may raise suspicion when inspected. Therefore, the following additional idea, Idea (B2), is adopted in this study to address this issue.

(B2) *Gradient creation*—Create a grayscale gradient within each module to reduce the visual contrast caused by modifying the corner pixels for message-bit embedding.

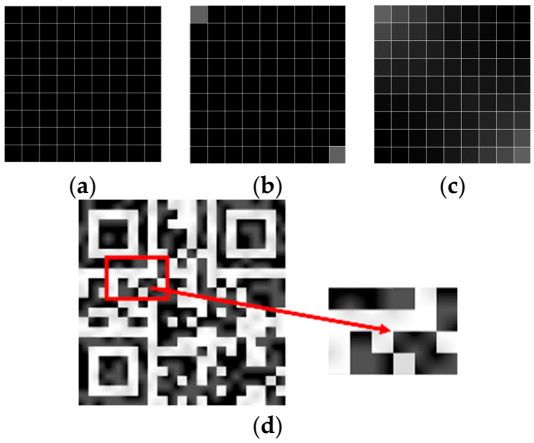


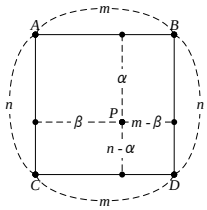
Figure 2. Illustration of bit embedding and gradient smoothing in a black module. (**a**) A black module. (**b**) A module with two corner pixels' values modified to embed bits. (**c**) The module with a gradient created to reduce the contrast effect of the two corner pixels. (**d**) A QR code image with message bits embedded and gradient operations applied, and an enlarged part of the resulting image showing the gradient effect.

An example of the gradient creation applied to a module is shown in Figure 2c, where the strong contrast between the two modified corner pixels and the inner pixels in Figure 2b is visibly reduced. An example of a QR code image with embedded message bits and gradient operations applied using large embedding change parameters ($k_b = 50$, $k_w = 50$) is shown in Figure 2d, where a portion of the resulting stego-image is enlarged to highlight the gradient effect. The QR code image, as a whole, retains its normal shape, though slight grayscale degradation and irregularities remain, which can still be tolerated by the QR code reader. The resulting QR code image retains its overall visual appearance when k_b and k_w are not excessively large, although some grayscale degradation and local irregularities remain, as further evaluated experimentally in Section 4.

The gradient operation used in this study is based on a bilinear interpolation scheme, as illustrated in Figure 3. Figure 3a depicts a general $n \times m$ rectangular block with four corner values A, B, C, and D, and an interior point P whose value is computed by the bilinear interpolation formula shown in Figure 3b. In the proposed method, each QR code module corresponds to a square block with side length w. The corresponding bilinear interpolation model for a $w \times w$ module and its associated quarter subregions, which provide the theoretical foundation for the proposed extraction procedure, will be introduced in Section 3.2. Moreover, the gradient operation is applicable only when the module side length satisfies $w \geq 4$, because each quarter subregion then contains at least 2×2 pixels, allowing local intensity variations to be effectively smoothed.

Assume for the moment that the modules in a QR code image are processed in a raster scan order, starting from the top-leftmost module. Additionally, assume that the four corner pixels in each module are processed in the following sequence: top-leftmost, top-rightmost, bottom-leftmost, and finally bottom-rightmost, or simply in a raster scan order.

In this way, the previously introduced Ideas (B1) and (B2) together are found to be *insufficient* to prevent noticeable contrast between neighboring module corner pixels. An example of such an undesirable case is shown in Figure 4, where the currently processed module M_{cur} is the bottom-right one, which is black, and its upper-left neighboring module M_{ul} is also black (i.e., of the same color as M_{cur}). The currently processed corner pixel P_{cur} of M_{cur} is the top-leftmost one, where message-bit embedding is being considered. However, the bottom-rightmost corner pixel P_{psd} of M_{ul}, which is adjacent to P_{cur}, has already been processed, resulting in a modified grayscale value, denoted as c_{psd}, which may differ from the black color of M_{cur}.



(**a**)

$$p = \frac{(n-\alpha)(m-\beta)A + (n-\alpha)\beta B + \alpha(m-\beta)C + \alpha\beta D}{m \times n}$$

(**b**)

Figure 3. General bilinear interpolation on an $n \times m$ rectangular block. (**a**) A rectangular block with corner values A, B, C, and D, and an interior point P. (**b**) Bilinear interpolation for computing the value of P.

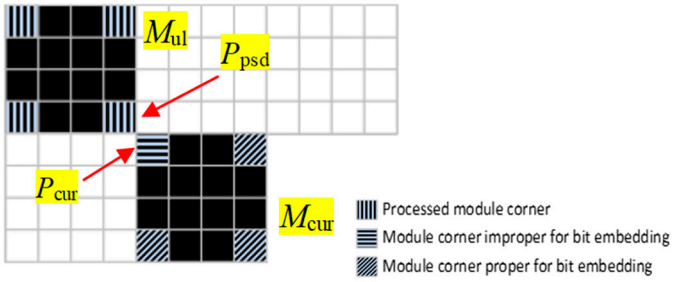


Figure 4. Illustration of a module corner P_{cur} improper for message-bit embedding.

To avoid creating an undesirable contrast between P_{cur} and P_{psd}, P_{cur} is deemed *improper* for message-bit embedding. Instead, its grayscale value should be modified to match that of P_{psd}, namely c_{psd}. This requirement leads to the following third principle for message embedding in QR code images.

(B3) *Avoiding the use of improper module corner pixels for message embedding*—avoid embedding a message bit into a corner pixel P_{cur} of the currently processed module M_{cur}, if any of its neighboring module corner pixels, denoted as P_{psd}, satisfy the following conditions:

(1) the module M_{psd} that contains P_{psd} is of the same color as that of M_{cur};

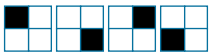
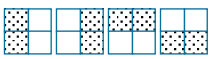
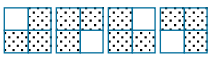
(2) P_{psd} has already been used for embedding a message bit.

In such cases, instead of embedding a bit at P_{cur}, simply adjust the grayscale value of P_{cur} to match that of P_{psd}.

Under the design of Idea (B3), the proposed embedding method ensures that, for a set of N mutually adjacent corner pixels originating from N distinct modules in a QR code, at most one white corner pixel and at most one black corner pixel are used to embed a message bit, while all other corner pixels of the same color in the set remain unused for embedding. Here, $N = 4$ indicates that the four corner pixels form a mutually adjacent 2×2 group; $N = 2$ indicates that the two corner pixels are adjacent along a horizontal or vertical direction, appearing at the outer boundary (edges) of the QR code; $N = 1$ indicates that the corner pixel belongs to a single module and has no adjacent corner pixel from any other module, appearing at one of the four extreme corners of the entire QR code boundary.

All possible combinations of mutually adjacent corner pixels located at the central junctions of distinct 2×2 modules within a QR code, together with their embedding capacities, are shown in Table 1, where the dotted box denotes the area outside the QR code boundary for $N = 2$ and $N = 1$.

Table 1. Exhaustive listing of all possible mutually adjacent corner-pixel combinations at the central junctions of distinct 2 × 2 modules.

Number (N) of Mutually Adjacent Corner Pixels	Number of Black Corner Pixels	Number of White Corner Pixels	Corner-Pixel Combination	Embedding Capacity (bits)
4	4	0		1
	3	1		2
	2	2		2
	1	3		2
	0	4		1
2	2	0		1
	1	1		2
	0	2		1
1	1	0		1
	0	1		1

In the proposed embedding method, the maximum embedding capacity is the same whether message bits are embedded into the module corner pixels of a QR code following a raster scan order or a pseudorandom permutation order. In the subsequent examples illustrating the embedding process, the aforementioned raster scan order will be used for convenience of explanation.

However, when the number of module corner pixels available for message embedding in a QR code exceeds the number of message bits, the raster scan order embedding strategy results in a QR code where the upper portion contains gradient modules, while the lower portion retains its original non-gradient appearance. This creates a noticeable visual contrast between the two sections, thereby making the embedding modifications more visually apparent. Therefore, the following approach is required.

(B4) *Visiting module corner pixels in pseudorandom permutation order for message-bit embedding*—visit each candidate module corner pixel exactly once according to a pre-generated pseudorandom permutation of the indices, to embed a message bit when applicable or otherwise adjust its grayscale value to match those of neighboring pixels in adjacent modules, in accordance with the principles underlying Ideas (B1) and (B3). Finally, apply gradient operations to all modules in the QR code based on their grayscale values.

It is noted that while Idea (B4) is being carried out, the message bits may be exhausted *before* the pseudorandom permutation sequence is fully traversed in the message-embedding process. This occurs when the number of message bits, N_{mes}, is smaller than

the number of module corner pixels, N_{pro}, that are proper for message-bit embedding in the QR code image. In this case, each subsequently visited module corner pixel, P_{cur}, is not used for message-bit embedding. However, if P_{cur} has a neighboring pixel, P_{psd}, that satisfies the two properties mentioned in Idea (B3), then the grayscale value of P_{cur} must still be adjusted to match that of P_{psd}.

The respective effects of corner-pixel grayscale adjustment, adjacent-corner consistency adjustment, and gradient creation are further illustrated using representative intermediate and enlarged results in Section 4.3.

3.2. Theoretical Analysis of Quarter Subregion Averaging

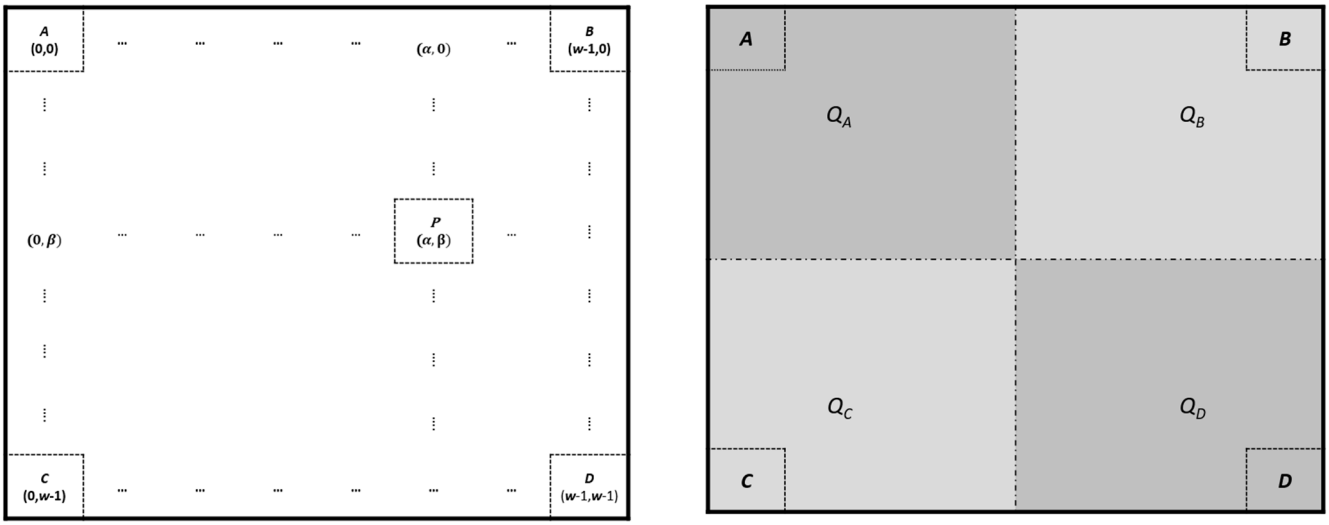
Although Figure 3 illustrates the general bilinear interpolation scheme, the proposed extraction method relies on a particular $w \times w$ square module structure. Therefore, this section further investigates the properties of bilinear interpolation and quarter subregion averages for a $w \times w$ QR code module.

3.2.1. Bilinear Interpolation Model for QR Code Modules

As illustrated in Figure 5a, a $w \times w$ module is represented in an image coordinate system whose origin is located at the upper-left corner of the module. The four corner pixels are denoted by A, B, C, and D, respectively, and the grayscale value of an interior pixel $P(\alpha, \beta)$ is generated according to the bilinear interpolation model

$$P(\alpha,\ \beta) = \frac{(w-1-\alpha)(w-1-\beta)A + \alpha(w-1-\beta)B + (w-1-\alpha)\beta C + \alpha\beta D}{(w-1)^2} \quad (1)$$

where $0 \le \alpha,\ \beta \le w-1$.



(**a**) (**b**)

Figure 5. Bilinear interpolation and quarter subregions of a $w \times w$ QR code module. (**a**) Coordinate system of a $w \times w$ module, showing the four corner pixels A, B, C, and D, and an interior pixel $P(\alpha, \beta)$. (**b**) Four quarter subregions Q_A, Q_B, Q_C, and Q_D, each containing $\lfloor w/2 \rfloor \times \lfloor w/2 \rfloor$ pixels and used for computing local average grayscale values during message extraction.

To facilitate message extraction, the module is further partitioned into four quarter subregions, denoted by Q_A, Q_B, Q_C, and Q_D, as shown in Figure 5b. Let

$$N = \left\lfloor \frac{w}{2} \right\rfloor. \quad (2)$$

Each quarter subregion therefore contains

$$N^2 = \left\lfloor \frac{w}{2} \right\rfloor^2 \tag{3}$$

pixels. For odd values of w, the central row and/or column of pixels are excluded so that all four quarter subregions contain the same number of pixels and preserve geometric symmetry.

The average grayscale values from the four quarter subregions constitute the theoretical foundation of the proposed extraction method and lead to the threshold properties established in Theorem 1.

3.2.2. Quarter Subregion Averages

Let Q_A, Q_B, Q_C, and Q_D denote the four quarter subregions shown in Figure 5b. The average intensity of the quarter subregion adjacent to an arbitrary corner pixel $X \in \{A, B, C, D\}$ can be expressed as

$$\overline{P}_x = w_0 X + w_1 Y_1 + w_2 Y_2 + w_3 Y_3, \tag{4}$$

where Y_1 and Y_2 are the two corner pixels adjacent to X, and Y_3 is the diagonally opposite corner pixel. The weighting coefficients are

$$w_0 = \left(1 - \frac{N-1}{2(w-1)}\right)^2, \tag{5}$$

$$w_1 = \frac{(N-1)(2w-N-1)}{4(w-1)^2}, \tag{6}$$

$$w_2 = \frac{(N-1)(2w-N-1)}{4(w-1)^2}, \text{ and} \tag{7}$$

$$w_3 = \left(\frac{N-1}{2(w-1)}\right)^2. \tag{8}$$

By symmetry,

$$w_1 = w_2, \tag{9}$$

and

$$w_0 + w_1 + w_2 + w_3 = 1. \tag{10}$$

Moreover,

$$w_0 > w_1 = w_2 > w_3. \tag{11}$$

As $w \to \infty$,

$$w_0 : w_1 : w_2 : w_3 \to 9 : 3 : 3 : 1. \tag{12}$$

The derivations of (5)–(8) and (10)–(11), and of (12), are given in Appendices A.1 and A.2, respectively. Equation (11) indicates that the quarter subregion average is primarily determined by its nearest corner pixel. Moreover, the asymptotic ratio in (12) shows that the nearest corner pixel contributes more than one-half (approximately 9/(9 + 3 + 3 + 1) = 56.25%) of the average intensity, whereas the remaining three corner pixels collectively contribute less than one-half.

3.2.3. Theoretical Basis of Quarter Subregion Averaging

The following theorem establishes the theoretical foundation for identifying the values of module corner pixels from the average grayscale values of their corresponding quarter subregions.

Theorem 1. *Consider a $w \times w$ QR code module with $w \geq 4$, whose interior pixels $P(\alpha, \beta)$, $0 \leq \alpha, \beta \leq w - 1$, are generated according to the bilinear interpolation model in (1). Let $N = \lfloor w/2 \rfloor$, and let Q_X, $X \in \{A, B, C, D\}$, denote the $N \times N$ quarter subregion adjacent to corner pixel X. Let*

$$\overline{P}_x = \frac{1}{N^2} \sum_{(\alpha,\beta) \in Q_X} P(\alpha, \beta) \tag{13}$$

denote the average grayscale value of all pixels in Q_X. Suppose that the four corner pixels of the module belong to the grayscale pair: $\{0, k_b\}$, $0 < k_b < 128$, for a black module, or $\{255 - k_w, 255\}$, $0 < k_w < 128$, for a white module. Then, for a black module,

$$\overline{P}_x < \frac{k_b}{2} \leftrightarrow X = 0, \text{ and} \tag{14}$$

$$\overline{P}_x > \frac{k_b}{2} \leftrightarrow X = k_b. \tag{15}$$

For a white module,

$$\overline{P}_x < 255 - \frac{k_w}{2} \leftrightarrow X = 255 - k_w, \text{ and} \tag{16}$$

$$\overline{P}_x > 255 - \frac{k_w}{2} \leftrightarrow X = 255. \tag{17}$$

Furthermore,

$$\frac{k_b}{2} \text{ and } 255 - \frac{k_w}{2} \tag{18}$$

are strict separating thresholds because equality cannot occur under the theoretical model.

Proof of Theorem 1. See Appendix A.3. □

According to Theorem 1, the values of the four corner pixels can be independently determined solely from their corresponding quarter subregion averages. Consequently, the thresholds

$$\frac{k_\mathrm{b}}{2} \text{ and } 255 - \frac{k_\mathrm{w}}{2} \tag{19}$$

are employed in the proposed bit-extraction algorithm described in Section 3.4 to extract an embedded message bit from a corner pixel by examining the corresponding candidate module corner.

The threshold-separation property established in Theorem 1 assumes that the relevant pixel values and module geometry of the generated stego-image are preserved. Accordingly, the theoretical thresholds are directly applicable to the generated stego-images and to lossless storage or transmission operations that leave the relevant pixel values and module geometry unchanged. Post-processing operations that alter pixel values or module geometry, such as rounding, quantization, lossy compression, resampling, blurring, and geometric distortion, are not covered by the theorem and may affect the reliability of message extraction.

3.3. Message-Embedding Algorithm

Based on the ideas presented in Section 3.1, the message-embedding algorithm is presented below.

In the present implementation, the pseudorandom mechanism P is implemented using the standard C rand() function in Dev-C++, with srand() initialized by the 32-bit unsigned seed K. Thus, 2^{32} possible seed values are available. It should be emphasized that rand() is not a cryptographically secure pseudorandom number generator; its use here is intended to generate a pseudorandom visiting order rather than to provide cryptographic security. Further security implications are discussed in Section 3.6.

The computational complexity of Algorithm 1 can be analyzed in terms of the number of QR code modules and the module side length. Let N_{m} denote the total number of modules in the QR code image and w denote the module side length in pixels. Since each module contains four corner pixels, the construction and pseudorandom visiting of the module corner pixels in Steps 1–4 require $O(N_{\mathrm{m}})$ time, because each corner pixel is processed at most once and only a constant number of neighboring corner pixels are examined for each candidate. In Step 5, the gradient operation based on bilinear interpolation is applied to every $w \times w$ module, requiring $O(w^2)$ operations per module. Therefore, the overall time complexity of Algorithm 1 is $O(N_{\mathrm{m}} \times w^2)$. When the module side length w is fixed, the complexity reduces to $O(N_{\mathrm{m}})$.

Algorithm 1: Message-bit embedding

Input: a QR code image I; a string S_m of message bits; the embedding change parameters k_{b} and k_{w} for adjusting the grayscale values of black and white module corner pixels, respectively; and seed K for a pseudorandom mechanism P.

Output: a stego-image I' of I.

Steps:

Step 1. Construct a sequence S_c of N_c module corner pixels by visiting the modules in a raster scan order and the corner pixels in each module also in a raster scan order.

Step 2. Generate a pseudorandom permutation sequence Q of the indices from 1 to N_c via the pseudorandom mechanism P with seed K for use in visiting each candidate module corner pixel exactly once.

Step 3.

while the message-bit string S_m is not empty **do:**

3.1.

if there are any unvisited numbers in Q

then

take a corner pixel P_{cur} with grayscale value v from S_c specified by the first unvisited number in Q; and denote the module including P_{cur} as M_{cur};

else

terminate the embedding process and report that the message length exceeds the available embedding capacity.

3.2. check the neighboring corner pixels of P_{cur}:

if one of them, denoted as P_{psd} and coming from module M_{psd}, satisfies the following conditions:

(1) M_{psd} is of the same color as that of M_{cur};

(2) P_{psd} has been processed to embed a message bit,

then change the grayscale value v of P_{cur} to match that of P_{psd};
else take out the leading bit b in S_m and conduct the following operations to embed b:
if $b = 1$
then set $v = v + k_b$ **when** $v = 0$ (black);
or set $v = v - k_w$ **when** $v = 255$ (white);
else leave v unchanged. // Dealing with $b = 0$
end while. // End of checking exhaustion of message bits
Step 4. // Dealing with unprocessed module corners
while not all the corner pixels in I are processed **do:**
4.1. take a corner pixel P_{cur} with grayscale value v from S_c specified by the first unvisited number in Q; and denote the module including P_{cur} as M_{cur};
4.2. check the neighboring corner pixels of P_{cur}:
// Same as the operations of Step 3.2, excluding the first "else" part of Step 3.2
if one of them, denoted as P_{psd} and coming from module M_{psd}, satisfies the following conditions:
(1) M_{psd} is of the same color as that of M_{cur};
(2) P_{psd} has been processed to embed a message bit,
then change the grayscale value v of P_{cur} to match that of P_{psd};
end while. // End of dealing with unprocessed corner pixels
Step 5. Apply gradient operations as described in Section 3.2.1 to each of the modules in I.
Step 6. Exit with the resulting I as the output stego-image I'

3.4. Message Extraction from Stego-Images

Before extracting the hidden secret message from a QR code stego-image, it is necessary to first determine the module side length and the embedding change parameters k_b and k_w, which were used to adjust the grayscale values of black and white module corner pixels, respectively, during message embedding. The module side length w is used to localize the module boundaries and corner pixels, whereas k_b and k_w are required to determine the decision thresholds used for message-bit extraction.

A 5-M QR code stego-image with the module side length $w = 4$ and the embedding change parameters $k_b = 50$ and $k_w = 25$ is shown in Figure 6a, in which the message data have been exhaustively embedded using Algorithm 1. The corresponding run-length histogram of dark and light pixels is presented in Figure 6b. Distinct peaks can be observed at run lengths of 4, 8, 12, 16, 20, ... pixels, all of which are integer multiples of $w = 4$, i.e., $n \times w$ for positive integers n. This behavior arises because the modules have a side length of 4 pixels, resulting in histogram peaks aligned with the module dimension.

The histogram of the grayscale values of the pixels in Figure 6a is shown in Figure 6c. A distinct peak within the range 1–127 occurs at a grayscale value of 50, while another peak within the range 128–254 is located at a grayscale value of 230. These values correspond to $0 + k_b = 50$ and $255 - k_w = 230$, respectively, indicating the use of change parameters $k_b = 50$ and $k_w = 25$. This behavior arises because the module corner pixels can be modified through either k_b or k_w.

Based on the module-structure analysis described above, Algorithm 2 for determining the module side length w of a QR code stego-image is presented below. The module side length is obtained from the run-length distribution of the image and is subsequently used to identify the module boundaries and corner-pixel locations required by the parameter-determination and message-extraction procedures.

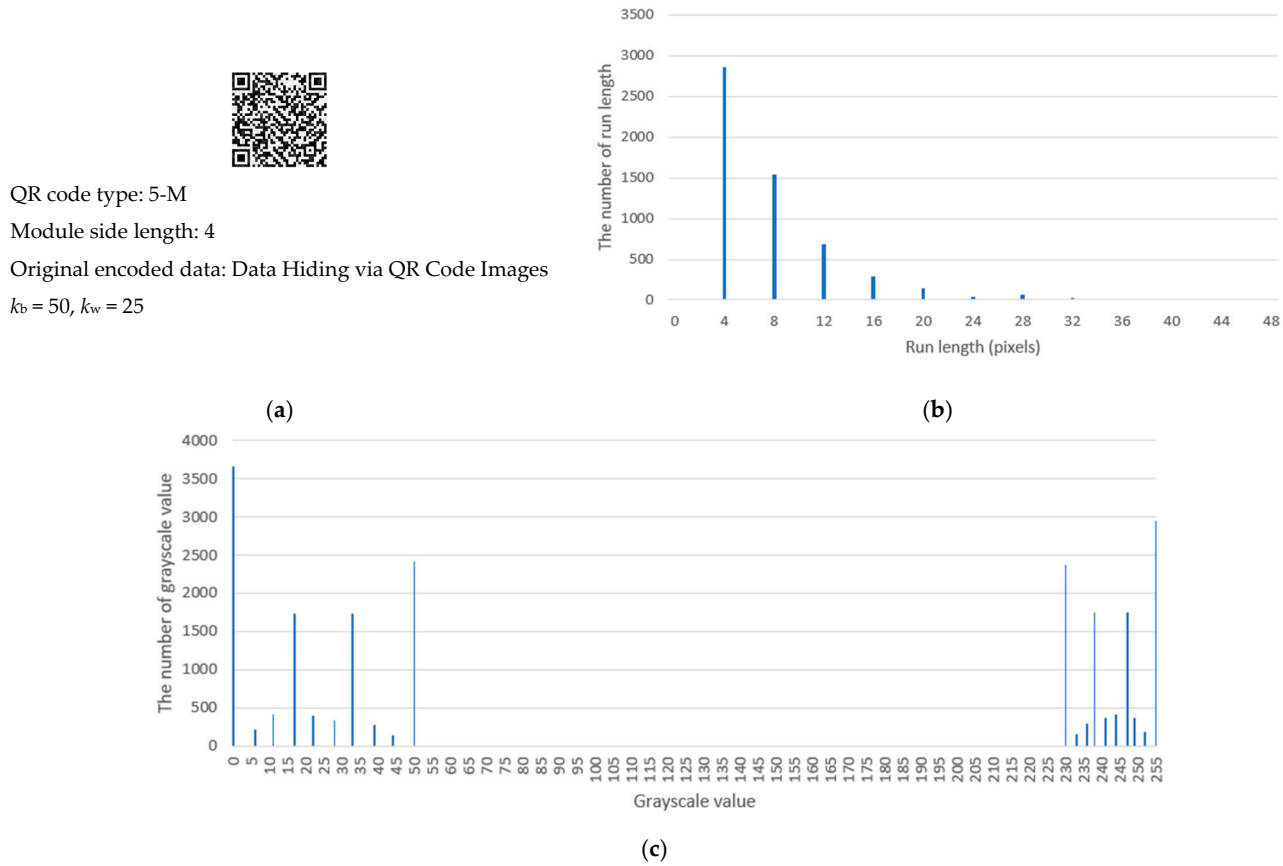


Figure 6. Run-length histogram and grayscale value histogram of a 5-M QR code stego-image. (**a**) 5-M QR code stego-image with the module side length $w = 4$, the embedding change parameters $k_b = 50$ and $k_w = 25$. (**b**) Run-length histogram obtained for a module side length of $w = 4$. (**c**) Grayscale value histogram obtained for change parameters $k_b = 50$, $k_w = 25$.

Algorithm 2: Determination of the module side length w in a QR code stego-image

Input: a stego-image I of a QR code with message bits embedded.
Output: the module side length w.
Steps:

Step 1. Raster-scan the pixels of I in both horizontal and vertical directions to identify all runs of dark pixels (grayscale values 0–127) and light pixels (grayscale values 128–255). The length of each run is then computed, and a run-length histogram R of I is constructed accordingly.

Step 2. Determine the module side length corresponding to the largest peak in R within the range of values ≥ 4, and take it as the desired side length w of the modules in I, where w can be expressed as

Step 3. Exit with the resulting w as the output.

An algorithm for automatically determining the embedding-change parameters k_b and k_w, which are used to adjust the grayscale values of black- and white-module corner pixels, respectively, is presented as Algorithm 3 below. The procedure assumes that the largest histogram peaks within the specified grayscale ranges correspond to k_b and $255 - k_w$. However, if the message is short or highly unbalanced, or if too few "1" bits are assigned to black- or white-module corners, one or both of the required peaks may be absent. If either required peak is absent, Algorithm 3 reports a parameter-determination failure.

Algorithm 3: Determination of the embedding change parameters k_b and k_w for adjusting the grayscale of black and white module corner pixels, respectively, in a QR code stego-image

Input: a stego-image I of a QR code with message bits embedded.
Output: the embedding change parameters k_b and k_w, corresponding to black and white corner pixels, respectively, or a parameter-determination failure.
Steps:
Step 1. Scan the pixels of I to collect all pixel values in I and construct the pixel-value histogram G;
Step 2. Find the embedding change parameters for adjusting the grayscale values of corner pixels in black and white modules, respectively, by locating the largest peak in G within 1–127 and 128–254, respectively.
Step 3. **if** the frequencies of the largest peak in G within 1–127 and 128–254 are both nonzero
then
Take these values as the desired embedding change parameters k_b and k_w for the modules in I, where k_b and k_w can be expressed as

$$k_b = \underset{1 \le i \le 127}{\operatorname{argmax}} G(i), \text{ and } k_w = 255 - \underset{128 \le i \le 254}{\operatorname{argmax}} G(i). \quad (20)$$

Exit with the resulting k_b and k_w as the output.
else
Exit and report parameter-determination failure.

It should also be noted that the grayscale-histogram characteristics used for automatic parameter determination may provide statistical evidence of the embedding process to an informed adversary. The corresponding security implications are discussed in Section 3.6.

After describing how to determine the module side length and the embedding change parameters in a stego-image of a QR code, Algorithm 4 is proposed for extracting a message bit from a module corner.

The data embedded in a QR code stego-image are assumed to consist of a 16-bit field representing the bit length of a character message S_m, followed sequentially by the binary representation of S_m. It is assumed that the message length can be represented by a 16-bit unsigned binary value. If its binary representation contains fewer than 16 bits, leading zeros are added to form the complete 16-bit length field before embedding.

In the proposed method, the embedding-change parameters k_b and k_w may be explicitly shared between the sender and the intended receiver to avoid parameter-determination failure during message extraction. Accordingly, k_b and k_w are treated as optional inputs to the message-extraction algorithm. If both parameters are explicitly provided, their values are used directly. Otherwise, Algorithm 3 is invoked to determine them automatically. If Algorithm 3 reports a parameter-determination failure, the extraction process is terminated. Therefore, when the embedded message is short or highly unbalanced, explicitly sharing and providing k_b and k_w during extraction is required to avoid a parameter-determination failure in Algorithm 3.

Based on this hybrid parameter-acquisition mechanism, the message-extraction algorithm is presented as Algorithm 5 below. In Algorithm 5, Algorithm 2 is used to determine the module side length w, Algorithm 3 is conditionally invoked to determine k_b and k_w, and Algorithm 4 is used to extract individual message bits from the QR code stego-image.

Algorithm 4: Extraction of a message bit from a module corner

Input: a corner pixel P_{cur} with grayscale value v in a QR code stego-image I; the module M_{cur} containing P_{cur}; the module side length w; and the embedding change parameters k_{b} and k_{w}.

Output: the extracted bit b, which may be empty, or a bit-extraction failure.

Steps:

Step 1. Set bit b to be empty initially.

Step 2. // check the neighboring corner pixels of P_{cur}

if one of the neighboring corner pixels, denoted as P_{psd} and coming from module M_{psd}, satisfies the following conditions:

(1) M_{psd} is of the same color as that of M_{cur};

(2) P_{psd} has been processed to extract a message bit,

then skip to Step 5; otherwise, proceed to the next step.

Step 3. // extract out the embedding bit b from P_{cur}

From the $\lfloor w/2 \rfloor$ by $\lfloor w/2 \rfloor$ sub-block of M_{cur} that contains P_{cur}, exclude all pixels whose grayscale values fall outside the valid range corresponding to c_M (black module: 0–127; white module: 128–255), where c_M denotes the color of M_{cur}.

if no pixel remains

then

Exit and report a bit-extraction failure,

else

Compute v_{ave} as the average grayscale value of all remaining pixels.

Step 4. // extract a message bit b

if $c_M = 0$ (black)

then

if $v_{\text{ave}} > (k_{\text{b}}/2)$ then set b to 1 else set b to 0;

else // $c_M = 255$ (white)

if $v_{\text{ave}} < 255 - (k_{\text{w}}/2)$ then set b to 1 else set b to 0;

Step 5. Exit with the extracted bit b as the desired output.

Algorithm 5: 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_m of characters followed by the binary version of S_m; and an identical seed K with the pseudorandom mechanism P employed in the embedding process; and, optionally, the explicitly provided embedding parameters k_{b} and k_{w}, corresponding to black and white corner pixels, respectively, for adjusting the grayscale values of corner pixels in black and white modules.

Output: the message S_m.

Steps:

Step 1. // *Finding the side length of the modules and the embedding change parameters*

Perform Algorithm 2 with I as the input to yield the side length w of the modules.

Step 2.

if the embedding parameters k_{b} and k_{w} are not explicitly provided

then

Perform Algorithm 3 with I as the input to yield the embedding change parameters k_{b} and k_{w}.

if Algorithm 3 fails to determine the parameters
then Terminate the message extraction process.

Step 3. *// Divide I into modules and determine the color of each module*
Divide I into modules according to w and determine the color c_M of each module M in I by computing the average grayscale value a of all pixels in M. If a is less than 128, assign 0 (black) as the desired color; otherwise, assign 255 (white).

Step 4. Construct a sequence S_c of N_c module corner pixels by visiting the modules in a raster scan order and the corner pixels in each module also in a raster scan order.

Step 5. Generate a pseudorandom permutation sequence Q of the indices 1 through N_c, which is identical to that used in the embedding process, determined by the pseudorandom mechanism P with seed K.

Step 6.

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

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

6.3. **while** $n \neq 16$ and an unvisited number in Q is available **do:**

(a) take a corner pixel P_{cur} with grayscale value v from S_c specified by the first unvisited number in Q; and denote the module including P_{cur} as M_{cur};

(b) perform Algorithm 4 with P_{cur}, M_{cur}, w, k_b, and k_w as the inputs to extract a bit b from P_{cur};

if Algorithm 4 reports a bit-extraction failure
then Terminate the message extraction process and report extraction failure.

// b might be empty

(c) **if** b is not empty, **then** append b to S_0 and set $n = n + 1$;

end while.

if $n \neq 16$
then Terminate the message extraction process and report extraction failure.

6.4. 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 7. *//Extracting the m embedded message bits*

7.1. reset the bit string S_0 to be empty;

7.2. **while** $m \neq 0$ and an unvisited number in Q is available **do:**

// where "m ≠ 0" means bits not all extracted yet

(a) take a corner pixel P_{cur} with grayscale value v from S_c specified by the first unvisited number in Q; and denote the module including P_{cur} as M_{cur};

(b) perform Algorithm 4 with P_{cur}, M_{cur}, w, k_b, and k_w as the inputs to extract a bit b from P_{cur};

if Algorithm 4 reports a bit-extraction failure
then Terminate the message extraction process and report extraction failure.

(c) **if** b is not empty, **then** append b to S_0 and set $m = m - 1$;

end while. *// End of checking exhaustion of message bits*

if $m \neq 0$
then Terminate the message extraction process and report extraction failure.

Step 8. Convert the bit string S_0 into characters to obtain the desired message S_m and exit with S_m as the output.

3.5. An Example of Message Extraction Results

Figure 7b shows the stego-image obtained by applying the proposed embedding algorithm (Algorithm 1) to the version-1-L QR code cover image shown in Figure 7a. The original content encoded in the QR code cover image is "www.nkust.edu.tw". A secret message, "Account:Da-Chun, Password:Hope&Faith", was embedded in the stego-image. The aforementioned raster scan order was used in the embedding process for convenience of explanation. Despite the presence of hidden data, the stego-image retained an overall appearance similar to that of the cover image. However, subtle pixel-level modifications introduced during the embedding process are made evident in the difference image shown in Figure 7c.



Figure 7. Demonstration of the proposed message embedding and extraction process. (**a**) Input cover image: a version-1 QR code. (**b**) Stego-image generated by embedding a 288-bit message into (**a**) using Algorithm 1. (**c**) Difference image highlighting pixel-level changes between (**a**) and (**b**). (**d**) Embedded bitstream consisting of a 16-bit message-length header followed by the 288-bit message. (**e**) Reconstructed verification matrix obtained using Algorithm 5, matching the binary data

shown in (**d**). (**f**) Stego-image in (**b**) with highlighted region shown by the red rectangular window. (**g**) Extracted bit values from the highlighted region in (**f**), illustrating correct module-level extraction by Algorithm 5.

The secret message, consisting of 36 characters, was first transformed into a 288-bit binary sequence. To indicate its length, a 16-bit header encoding the value 288 was prefixed, forming a complete 304-bit bitstream for embedding purposes. This full binary stream is depicted in Figure 7d. During the extraction process, the embedded data were extracted using Algorithm 5, which successfully reconstructed the bitstream and generated the verification matrix displayed in Figure 7e, precisely matching the binary content shown in Figure 7d. To provide additional clarity, Figure 7f presents a highlighted region of the stego-image, while Figure 7g displays the corresponding extracted bits from that region, confirming the accurate functionality of the extraction process at the QR module level.

3.6. Security Assumptions and Limitations

The security assumptions of the proposed method are clarified as follows. The embedding and extraction algorithms are assumed to be publicly known to an adversary. The embedding-change parameters k_b and k_w are not treated as cryptographic secrets because they may either be explicitly shared with the intended receiver or automatically estimated from the grayscale distribution of the stego-image, as described in Algorithm 3. Accordingly, in the present implementation, the seed K used to determine the pseudorandom visiting order is the only secret element associated with the pseudorandom visiting-order mechanism.

An informed adversary may be able to decode the visible QR code content and, when sufficient QR code configuration information is available, construct a corresponding standard binary QR code for comparison with the stego-image. Such a comparison may reveal the grayscale modifications introduced by the embedding process. Moreover, the grayscale modifications introduced during embedding may produce characteristic histogram distributions or peaks. Although these histogram characteristics are exploited by Algorithm 3 to facilitate automatic estimation of k_b and k_w, they may also provide statistical evidence of the embedding process and may enable an informed adversary to estimate the embedding-change parameters. Therefore, the histogram-based parameter-determination procedure should be regarded as an extraction mechanism rather than as a security mechanism. The proposed method is not intended to guarantee steganographic undetectability against an adversary possessing an appropriate reference QR code or employing dedicated steganalytic techniques.

In the current implementation developed using C in Dev-C++, the pseudorandom visiting order is generated using the standard C rand() function, with srand() used to initialize the pseudorandom-number generator. The seed is represented by a 32-bit unsigned integer and therefore has 2^{32} possible values. It should be noted that the C standard does not guarantee that rand() generates an identical sequence across different C library implementations, even when the same seed is used. Therefore, exact cross-platform reproducibility would require both the embedding and extraction sides to use the same explicitly specified pseudorandom-number generator and permutation procedure, together with the same seed K. However, rand() is not a cryptographically secure pseudorandom number generator (CSPRNG). Consequently, the seed-based pseudorandom visiting order is employed primarily to avoid a fixed deterministic embedding sequence rather than to provide cryptographic security. For applications requiring stronger security, a CSPRNG or a cryptographically secure pseudorandom permutation with an adequately sized secret key should be employed.

Finally, steganographic embedding should be distinguished from cryptographic protection of the hidden payload. The proposed method itself does not provide encryption, message authentication, or integrity protection. When confidentiality, integrity, or authenticity of the hidden message is required, appropriate cryptographic encryption and authentication mechanisms should be applied to the payload before it is embedded into the QR code image. Therefore, the present evaluation is limited to the information-hiding characteristics examined experimentally and should not be interpreted as establishing cryptographic security or resistance to dedicated steganalysis.

4. Experimental Results

4.1. Evaluation of Bit Embedding Rate Across QR Code Versions

A number of experiments were conducted in this study to evaluate the bit embedding rate of the proposed message embedding algorithm (Algorithm 1) for six versions of the QR code (versions 1, 5, 10, 20, 30, and 40) using the four error-correction levels: L, M, Q, and H. In total, $6 \times 4 = 24$ different QR code images were tested. The test QR code images, random seeds, experimental configurations, numerical results, raw runtime measurements, and pseudorandom-procedure code excerpts associated with the quantitative experiments in Sections 4.1, 4.2 and 4.4–4.7 are provided in the Supplementary Materials. The input secret message in each case was a sufficiently long sequence of random bits used to exhaust the entire available embedding capacity of the QR code image generated from a specific item of encoded data.

Both raster scan order and pseudorandom permutation order were applied to visit each module corner pixel exactly once during message embedding, and different values of the embedding change parameters k_b *and* k_w were tested. Two results, each including an original QR code image and three stego-images produced by Algorithm 1, are shown in Figure 8. It can be observed that as the values of k_b *and* k_w increase, the gradient effect in the appearance of the stego-image becomes more noticeable. However, the original mixed black-and-white grid structure created from the QR code encoded data remains intact, ensuring that the data can still be decoded from the stego-image. Furthermore, all embedded secret messages were correctly extracted.

Original image 1 of QR code		Property: Version: 1 Error-correction level: M Original encoded data: *Hiding*	
Image with message embedded and gradient applied			
Parameters k_b and k_w	10	20	40
Order	Raster scan	Raster scan	Raster scan
Capacity	877 bits	877 bits	877 bits

(**a**)

Original image 2 of QR code		Property: Version: 5 Error-correction level: H Original encoded data: *Data Hiding via QR Code Images*	
Image with message embedded and gradient applied			
Parameters k_b and k_w	10	20	40
Order	Pseudorandom permutation	Pseudorandom permutation	Pseudorandom permutation
Capacity	2648 bits	2648 bits	2648 bits

(**b**)

Figure 8. Two experimental results of message embedding in QR codes. (**a**) version 1 with error-correction level M. (**b**) version 5 with error-correction level H.

A statistical analysis of the experimental results for the case of visiting the module corners in pseudorandom permutation order across 24 different QR code images is shown in Table 2. The 24 QR code images comprise six QR code versions (1, 5, 10, 20, 30, and 40), with four error-correction levels (L, M, Q, and H) evaluated for each version. From Table 2, the following observations can be made: (1) for each version of the QR code, the images corresponding to the four error-correction levels yield nearly equal embedding capacities in bits and, consequently, nearly equal embedding rates in bits per module (bpm); (2) the values of the embedding capacities increase as the QR code version number increases; (3) for each version, the embedding capacity for each error-correction level is roughly double the number of modules in the QR code image; and (4) for all versions, the embedding rates, or equivalently, the bpm values, are all close to the value of 1.9. The mean ± sample standard deviation of the embedding rate for versions 1, 5, 10, 20, 30, and 40 is 1.967 ± 0.018, 1.954 ± 0.017, 1.920 ± 0.021, 1.893 ± 0.007, 1.889 ± 0.010, and 1.881 ± 0.005 bpm, respectively. The overall mean ± sample standard deviation calculated from all 24 QR code images is 1.917 ± 0.036 bpm. A broader evaluation involving additional payloads and QR code patterns may be considered in future work to further examine the generality of these results.

Table 2. Bit embedding rates of 24 QR code images across six versions and four error-correction levels.

Version	Error Correction Level	No. of Modules in QR Code			Embedding Capacity (bits)			Embedding Rate (bpm)			Version Mean ± Sample SD
		Black Modules	White Modules	Total Modules (A)	Black Modules (B)	White Modules (C)	Total Modules (D)	Black Modules (B/A)	White Modules (C/A)	Total Modules (D/A)	
1	L	226	215	441	458	412	870	1.04	0.93	1.97	1.967 ± 0.018
	M	228	213	441	458	419	877	1.04	0.95	1.99	
	Q	244	197	441	463	402	865	1.05	0.91	1.96	
	H	240	201	441	457	401	858	1.04	0.91	1.95	
5	L	687	682	1369	1365	1339	2704	1.00	0.98	1.98	1.954 ± 0.017
	M	707	662	1369	1368	1312	2680	1.00	0.96	1.96	
	Q	723	646	1369	1368	1302	2670	1.00	0.95	1.95	
	H	671	698	1369	1340	1308	2648	0.98	0.96	1.93	
10	L	1620	1629	3249	3186	3144	6330	0.98	0.97	1.95	1.920 ± 0.021
	M	1652	1597	3249	3129	3093	6222	0.96	0.95	1.92	
	Q	1616	1633	3249	3129	3114	6243	0.96	0.96	1.92	
	H	1708	1541	3249	3136	3026	6162	0.97	0.93	1.90	
20	L	4617	4792	9409	8808	8952	17,760	0.94	0.95	1.89	1.893 ± 0.007
	M	4709	4700	9409	8874	8888	17,762	0.94	0.94	1.89	
	Q	4823	4586	9409	9007	8857	17,864	0.96	0.94	1.90	
	H	4823	4586	9409	8963	8911	17,874	0.95	0.95	1.90	
30	L	9233	9536	18,769	17,510	17,760	35,270	0.93	0.95	1.88	1.889 ± 0.010
	M	9319	9450	18,769	17,621	17,801	35,422	0.94	0.95	1.89	
	Q	9395	9374	18,769	17,891	17,820	35,711	0.95	0.95	1.90	
	H	9389	9380	18,769	17,713	17,687	35,400	0.94	0.94	1.89	
40	L	15,430	15,899	31,329	29,220	29,708	58,928	0.93	0.95	1.88	1.881 ± 0.005
	M	15,392	15,937	31,329	29,198	29,659	58,857	0.93	0.95	1.88	
	Q	15,634	15,695	31,329	29,575	29,556	59,131	0.94	0.94	1.89	
	H	15,738	15,591	31,329	29,345	29,429	58,774	0.94	0.94	1.88	
Overall											1.917 ± 0.036

4.2. Evaluation of Computational Runtime for Different QR Code Versions

The computational runtime of the proposed message-embedding algorithm (Algorithm 1) was evaluated on a computer equipped with an Intel i7-8700 CPU @ 3.20 GHz and 16.0 GB of RAM, using Dev-C++ as the development environment. The same 24 QR

code images listed in Table 2 and considered in Section 4.1 were used, comprising six QR code versions (1, 5, 10, 20, 30, and 40) and four error-correction levels (L, M, Q, and H). Each of the 24 QR code images was evaluated with two module side lengths, $w = 4$ and $w = 8$, yielding a total of 48 experimental cases. Because the execution time for processing a single QR code image was very short, each experimental case was consecutively executed 100 times, and the total elapsed time was divided by 100 to estimate the runtime per execution. This measurement procedure was independently repeated 20 times for each case, and the mean runtime per execution was then calculated.

Figure 9 presents the resulting mean runtimes as a function of the number of QR code modules for the 48 experimental cases. For each value of w, the four QR codes belonging to the same version have the same number of modules and exhibit only small differences in their mean runtimes. Consequently, their corresponding data points are closely grouped together, forming six clusters for $w = 4$ and another six clusters for $w = 8$. For both module side lengths, the runtime increases approximately linearly with the number N_m of QR code modules. For each fixed value of w, this trend is consistent with the $O(N_m)$ time complexity derived in Section 3.3. In addition, the runtimes obtained with $w = 8$ are consistently higher than those obtained with $w = 4$, which is also consistent with the overall time complexity of $O(N_m \times w^2)$.

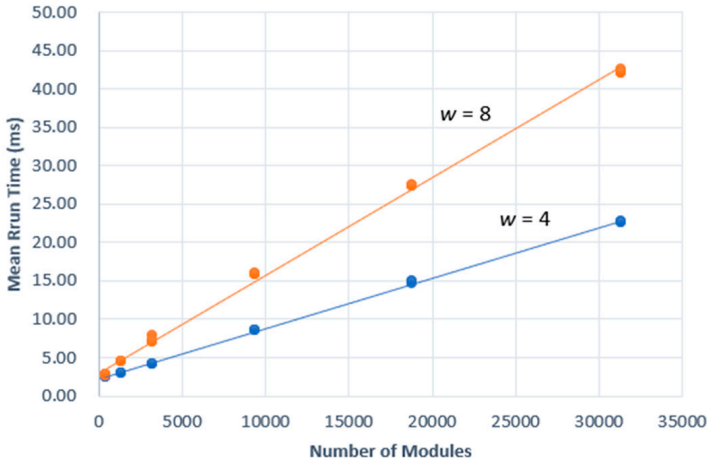


Figure 9. Mean runtime per execution versus the number of QR code modules for 24 QR code images evaluated with $w = 4$ and $w = 8$, yielding 48 data points. For each value of w, six clusters correspond to the six QR code versions, and the solid lines represent the fitted linear trends.

4.3. Component-Wise Analysis of the Proposed Method

To clarify the respective roles of the principal processing components, representative QR code images at selected processing states and their enlarged upper-central regions are presented in Figure 10 using a version 1 QR code with error-correction level Q, module side length $w = 9$, and $k_b = k_w = 50$. Figure 10b,c were specially generated for component-wise illustration and do not represent literal stage-by-stage execution states of Algorithm 1, whereas Figure 10d shows the final stego-image after completion of the gradient operation. In the actual embedding process, Ideas (B1) and (B3) are conditionally selected on a corner-by-corner basis according to the processing status of the current and neighboring module corner pixels, whereas Idea (B2) is applied after the corner-pixel processing has been completed.

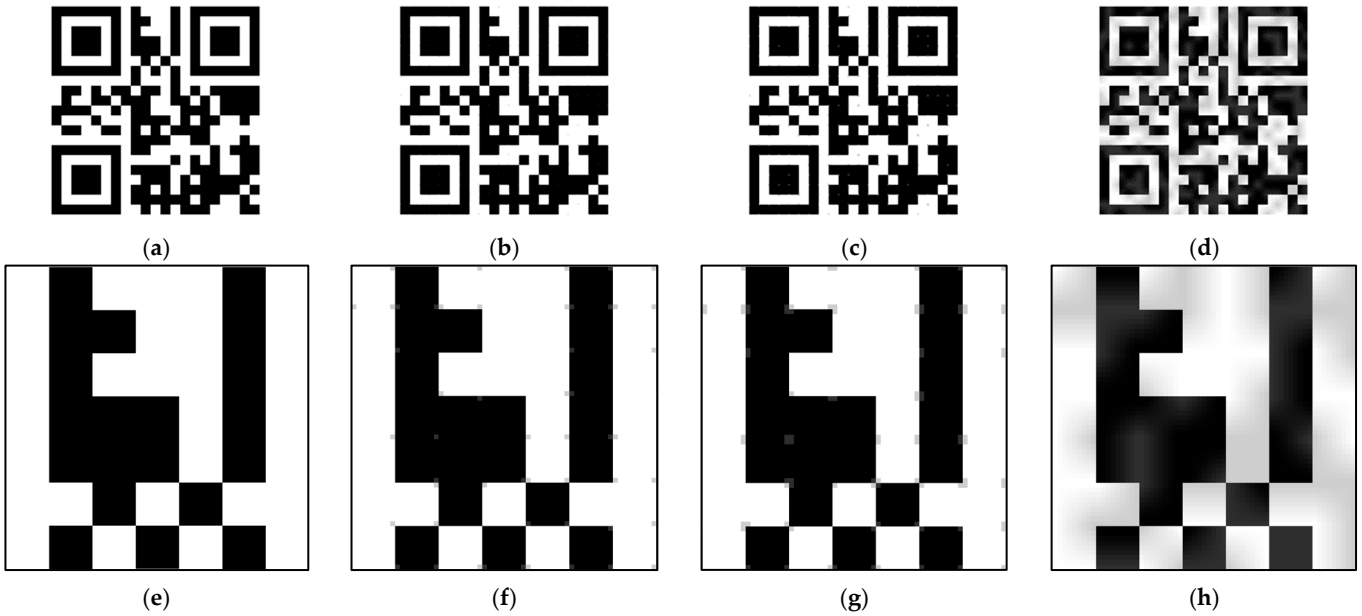


Figure 10. Component-wise illustration of the principal processing steps of the proposed method for a version 1 QR code with error-correction level Q and $w = 9$, using $k_b = k_w = 50$. (**a**) Original QR code image. (**b**) Result after corner-pixel message embedding according to Idea (B1). (**c**) Result illustrating the adjacent-corner consistency adjustment according to Idea (B3). (**d**) Final stego-image after gradient creation according to Idea (B2). (**e**–**h**) Enlarged views of the corresponding upper-central regions in (**a**–**d**), respectively.

Figure 10a shows the original QR code image. Figure 10b illustrates the effect of Idea (B1), in which message bits are embedded by adjusting the grayscale values of selected module corner pixels. This step provides the basic embedding mechanism but may introduce noticeable grayscale discontinuities between adjacent corner pixels belonging to modules of the same original color.

Figure 10c illustrates the effect of Idea (B3). When a neighboring corner pixel belonging to a module of the same original color has already been used for message-bit embedding, the current corner pixel is not used to embed another bit; instead, its grayscale value is adjusted to match that of the processed neighboring corner pixel. Consequently, abrupt grayscale transitions between adjacent modules are reduced.

Finally, Figure 10d illustrates the effect of Idea (B2). Bilinear-interpolation-based gradient creation propagates the corner-pixel grayscale modifications into the module interiors, thereby reducing abrupt intra-module intensity transitions and producing smoother local grayscale variations. Figure 10e–h provide enlarged views of the corresponding upper-central regions in Figure 10a–d, respectively, allowing these local effects to be observed more clearly.

For visualization purposes, relatively large embedding-change parameters ($k_b = k_w = 50$) are intentionally used in Figure 10 so that the effects of the individual processing components can be clearly distinguished. Such large parameter values are not required in practical applications and may result in readily visible grayscale changes. In practice, smaller values may be selected when higher visual fidelity is desired. It should also be emphasized that Figure 10b,c are specially prepared intermediate results for illustrating the effects of Ideas (B1) and (B3), respectively, and should not be interpreted as exact sequential execution stages of Algorithm 1. Figure 10d, in contrast, represents the final stego-image after Idea (B2) has been applied.

4.4. Evaluation of QR Code Images Under Different Change Parameter Settings

In this study, embedding change parameters k_b and k_w are used to adjust the grayscale values of corner pixels during data embedding. The following presents QR code stego-images generated with different k_b and k_w values, together with visual and objective image-quality evaluations under different embedding-change parameter settings.

The original QR code cover images used in this experiment include version-5 and version-10 QR codes. The encoded content in the version-5 QR code image is "Data Hiding via QR Code Images", whereas that in the version-10 QR code images is "Data Hiding via QR Code Images by Module Content Adjustments by Da-Chun Wu and Yuan-Ming Wu". For each version, error correction levels M and H are selected, and module side lengths w of 4 and 8 are employed. The actual sizes of these eight QR code cover images are illustrated in Figure 11, where the QR code type is specified as v–E, with v indicating the QR code version and E denoting one of the four error correction levels.

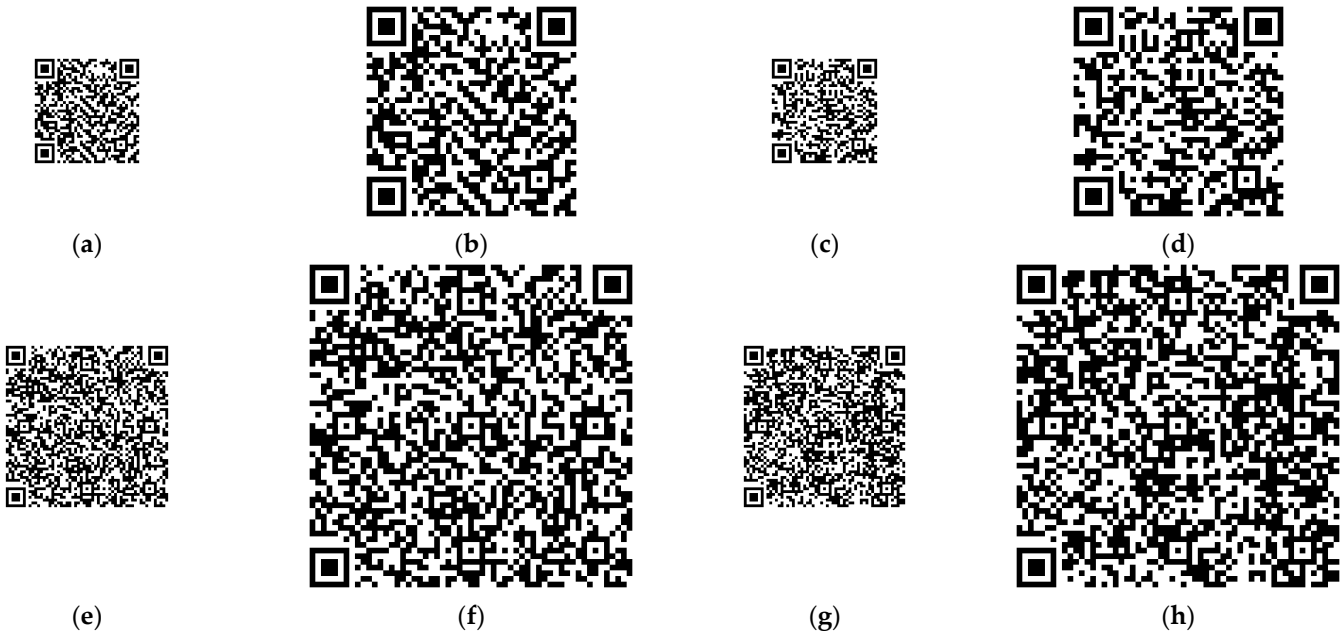


Figure 11. Eight representative QR code cover images shown at their actual sizes, with the corresponding types and module side lengths w. (**a**) 5-M with w = 4. (**b**) 5-M with w = 8. (**c**) 5-H with w = 4. (**d**) 5-H with w = 8. (**e**) 10-M with w = 4; (**f**) 10-M with w = 8. (**g**) 10-H with w = 4. (**h**) 10-H with w = 8.

The secret message "Password: Love & Peace." was repeatedly embedded into the QR code images shown in Figure 11 using a raster scan order. In this evaluation, the embedding change parameters k_b and k_w were selected from the set {25, 50, 75, 100, 125}. In particular, k_b and k_w could be assigned the same value or different values. The resulting stego-images generated from the cover QR code images in Figure 11a,b with $k_b = k_w = 25$ (or 75), along with Figure 11g,h with $k_b = k_w = 25$ (or 75), are illustrated in Figure 12.

From Figure 12, it can be observed that: (1) Larger values of k_b and k_w make differences in the stego-images more apparent. (2) QR codes with larger module side lengths show more noticeable changes in the stego-images. It is noted that, for the same QR code cover image, the choice of k_b and k_w values or the module side length does not affect the total embedding capacity.

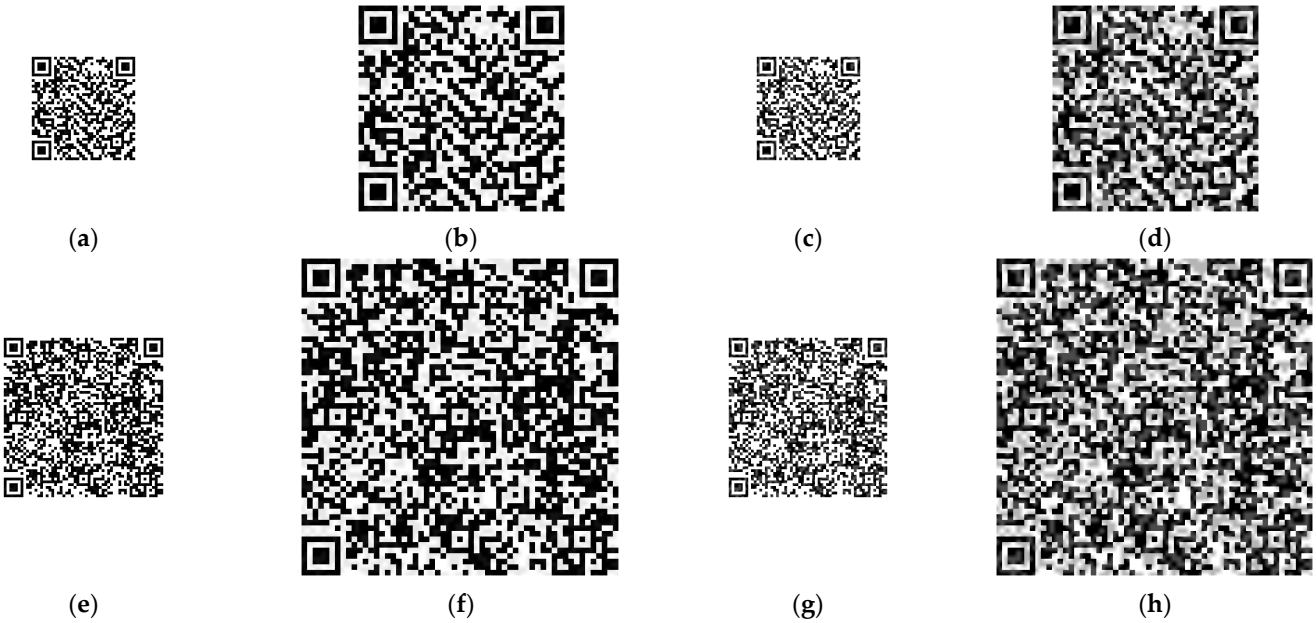


Figure 12. Representative stego-images generated from Figure 11a,b,g,h under different embedding change parameter settings. (**a**) Figure 11a with $k_b = k_w = 25$. (**b**) Figure 11b with $k_b = k_w = 25$. (**c**) Figure 11a with $k_b = k_w = 75$. (**d**) Figure 11b with $k_b = k_w = 75$. (**e**) Figure 11g with $k_b = k_w = 25$. (**f**) Figure 11h with $k_b = k_w = 25$. (**g**) Figure 11g with $k_b = k_w = 75$. (**h**) Figure 11h with $k_b = k_w = 75$.

Figure 13 presents an objective image-quality analysis of the 40 stego-images generated from the eight QR code cover images shown in Figure 11 using five embedding change parameter settings, where $k_b = k_w$ and their values were selected from {25, 50, 75, 100, 125}. Image quality was evaluated using MATLAB R2026a (MathWorks, Natick, MA, USA). Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) were computed using the built-in psnr() and ssim() functions from the Image Processing Toolbox, respectively. Visual Information Fidelity in the pixel domain (VIFp) was computed using the reference MATLAB implementation (vifp_mscale.m) provided by the LIVE Image Quality Assessment (IQA) project [28]. The VIFp metric is a pixel-domain implementation of the Visual Information Fidelity (VIF) framework proposed by Sheikh and Bovik [29]. Throughout this study, for the computation of PSNR, SSIM, and VIFp, the quiet zones surrounding both the cover and stego QR code images were excluded, and only the QR code symbol areas were used for image-quality evaluation.

As shown in Figure 13, all three image-quality metrics consistently decrease as k_b and k_w increase. For the eight QR code images, increasing $k_b = k_w$ from 25 to 125 decreases the PSNR values from approximately 24.62–25.03 dB to 10.62–11.03 dB, the SSIM values from approximately 0.819–0.956 to 0.496–0.645, and the VIFp values from approximately 0.592–0.650 to 0.143–0.186. This consistent tendency indicates that increasing the magnitude of the embedding changes introduces greater distortion into the QR code images. Furthermore, for a fixed module side length and embedding change parameter setting, only relatively small variations are observed among the QR codes with different versions and error correction levels, particularly in terms of PSNR and VIFp. Thus, the embedding change parameters have a substantially greater influence on the measured image quality than the QR code version or error correction level under the evaluated conditions.

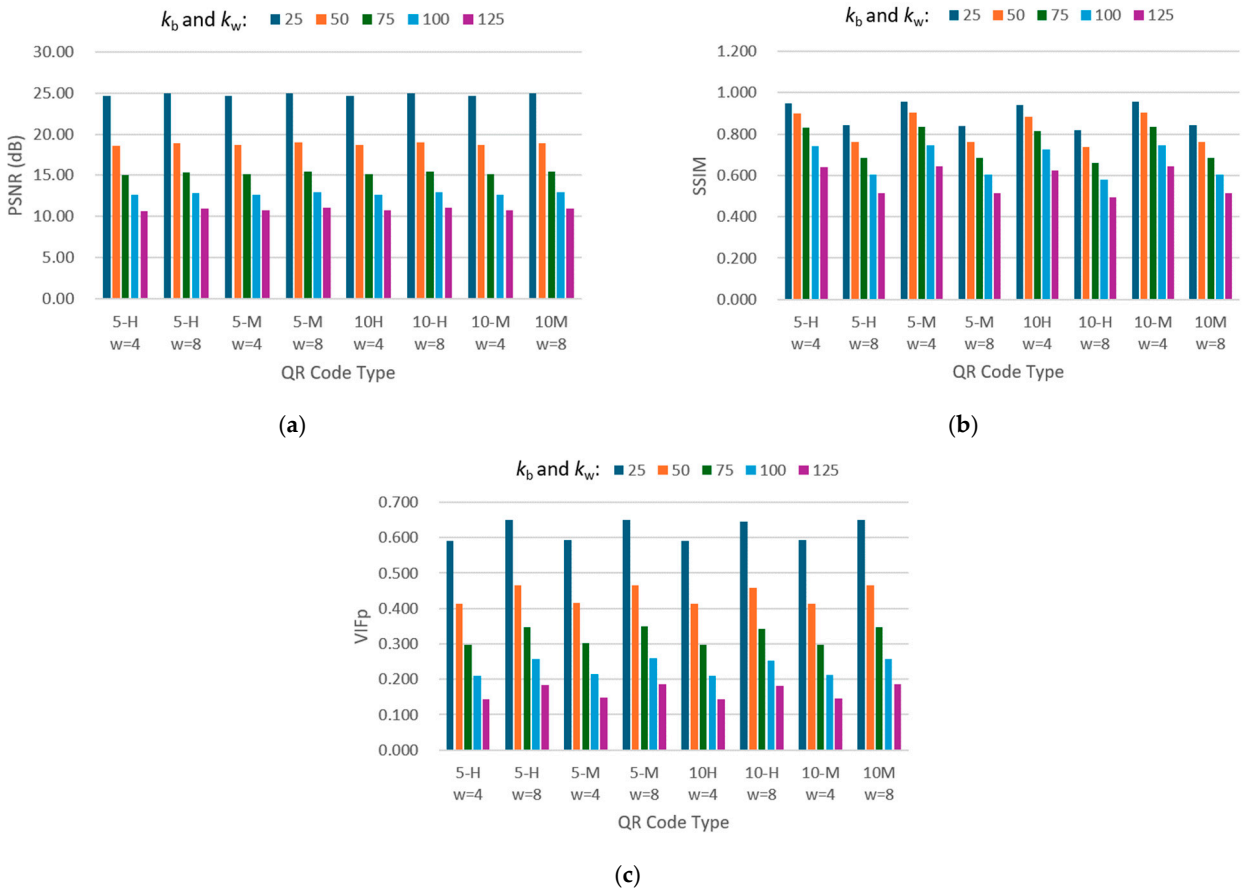


Figure 13. Objective image-quality evaluation of stego-images generated using different embedding change parameters: (**a**) PSNR; (**b**) SSIM; and (**c**) VIFp.

The effect of the module side length w, however, differs among the three quality metrics. For the same QR code version, error correction level, and embedding change parameter setting, the stego-images with $w = 8$ generally exhibit slightly higher PSNR and VIFp values than those with $w = 4$, whereas the SSIM values are consistently higher for $w = 4$. In particular, the lower SSIM values obtained with $w = 8$ are consistent with the more noticeable structural changes visually observed in Figure 12, while the PSNR and VIFp results show a slightly different tendency. This difference indicates that PSNR, SSIM, and VIFp respond differently to the modifications introduced by different module side lengths and therefore provide complementary assessments of the visual fidelity of the resulting stego-images. Overall, the results quantitatively characterize the trade-off between the magnitude of the embedding changes and the visual fidelity of the generated stego-images.

Examples of the resulting stego-images generated from the cover QR code image in Figure 11g, with a module side length of 4, are illustrated in Figure 14. The stego-images were produced under three different embedding conditions: (1) The secret message was embedded exclusively into the black modules, with k_b chosen from the set {25, 50, 75, 100, 125}; the white modules remained unchanged. (2) Conversely, embedding was carried out solely in the white modules, where k_w took values from {25, 50, 75, 100, 125}, leaving the black modules intact. (3) Embedding the secret message in both black and white modules with k_b and k_w set to an identical value selected from the set {25, 50, 75, 100, 125}. The black-module-only and white-module-only cases were included solely for visual and objective image-quality comparison and were not used for hidden-message recovery evaluation. From Figure 14, the following observations can be made: (1) Larger values of k_b or k_w result in more noticeable differences in the stego-images. (2) Under identical embedding change

parameter settings, stego-images generated by embedding only in black modules are less visually distinguishable than those generated by embedding only in white modules or in both black and white modules. For example, stego-images generated by embedding only in the black modules with k_b set to 50 show relatively low visual noticeability of the embedding modifications, whereas stego-images generated by embedding only in the white modules with $k_w = 50$ and those generated by embedding in both black and white modules with $k_b = k_w = 50$ show noticeable distinctions.

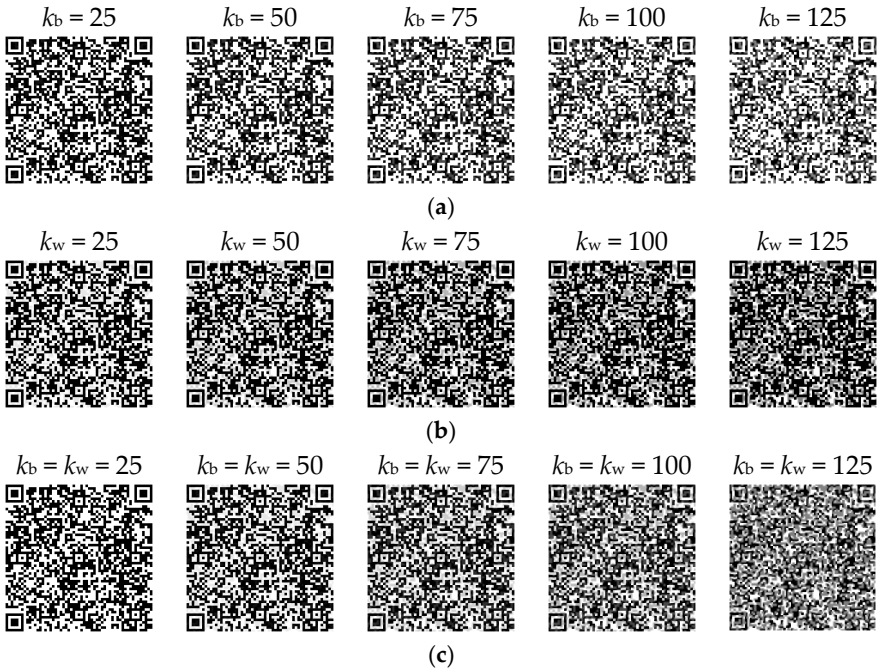


Figure 14. Stego-images derived from Figure 11g, generated under different embedding change parameters. (**a**) k_b only, (**b**) k_w only, and (**c**) both k_b and k_w.

Figure 15 presents an objective image-quality analysis of the 15 stego-images shown in Figure 14 using PSNR, SSIM, and VIFp. As shown in Figure 15, all three metrics decrease consistently as the embedding change parameter increases from 25 to 125 under all three embedding conditions. This trend confirms that a larger embedding change introduces greater distortion into the QR code image, regardless of whether the secret message is embedded in the black modules, the white modules, or both. Among the three embedding conditions, embedding in both black and white modules consistently yields the lowest PSNR, SSIM, and VIFp values at each parameter setting, indicating that modifying both types of modules produces the greatest overall change relative to the original QR code image.

An additional observation is that embedding only in the white modules consistently yields slightly higher PSNR, SSIM, and VIFp values than embedding only in the black modules. For example, when the embedding change parameter is 50, the PSNR values are approximately 21.96 dB and 21.45 dB for white-module-only and black-module-only embedding, respectively, whereas embedding in both types of modules results in a substantially lower value of approximately 18.69 dB. Similar relationships can be observed for SSIM and VIFp throughout the five parameter settings. This ordering differs somewhat from the visual observation in Figure 14, where changes introduced only into the black modules appear less visually noticeable than those introduced only into the white modules. The difference suggests that the objective image-quality metrics and direct visual observation respond differently to modifications occurring in black and white module regions.

Therefore, the results of Figures 14 and 15 should be regarded as complementary: Figure 14 illustrates the visually observable effects of the embedding operation, whereas Figure 15 quantitatively characterizes the corresponding image distortion.

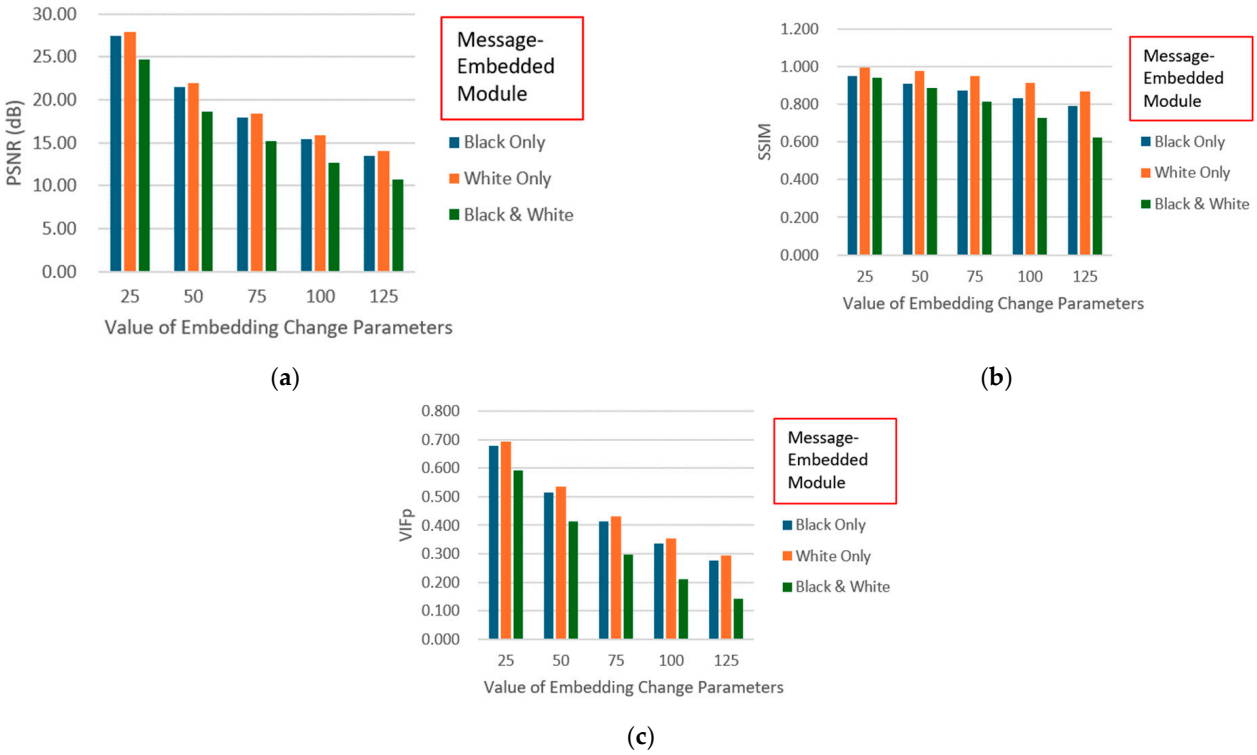


Figure 15. Objective image-quality evaluation of the stego-images shown in Figure 14 under different message-embedding conditions: (**a**) PSNR; (**b**) SSIM; and (**c**) VIFp.

4.5. Evaluation of Stego-Image Readability with Barcode Readers

In the proposed method, the embedding change parameters k_b and k_w play a key role in modifying the grayscale values of module corner pixels. Evaluating how different values of these parameters affect QR code readability is therefore an important part of this study.

Specifically, a number of experiments were conducted in which parameters k_b and k_w were varied to generate stego-images of QR codes of versions 5 and 10 using the proposed embedding algorithm (Algorithm 1). The stego-images, generated at 300 dpi, were printed using an HP LaserJet Pro 4003dn printer and subsequently scanned by three barcode-reading tools: (1) the AT-30Q-SM scanner produced by Denso Wave, (2) the QR and barcode scanner App by Gamma Play running on a Samsung Galaxy Note 9, and (3) the QR Code Reader App by Komorebi Inc. operating on an Apple iPad Air 3. The experiments examined which tested values of k_b and k_w preserved stego-image readability. In this evaluation, w was set to 4 and 8, while k_b and k_w were selected from 25, 50, 75, 100, and 125.

The outcomes of the readability tests with barcode readers are summarized in Table 3. The notation used is as follows: (1) '○' indicates *readable*, meaning the original encoded data stored in the QR code is decoded successfully in a single scan; (2) '△' denotes *readable after multiple scans*, where decoding succeeds only after repeated attempts; and (3) '×' represents *unreadable*, where decoding fails despite multiple scans.

Table 3. QR code readability under different embedding parameters using various barcode readers.

QR Code Type	w	Testing Equipment	Embedding Change Parameter					
			Black Pixels Only		White Pixels Only		Both Black and White Pixels	
			25, 50, 75, and 100	125	25, 50, 75, and 100	125	25, 50, 75, and 100	125
5-M	4	Denso	○	○	○	○	○	×
		Samsung	○	○	○	○	○	×
		iPad	○	○	○	○	○	△
	8	Denso	○	○	○	○	○	△
		Samsung	○	○	○	○	○	△
		iPad	○	○	○	○	○	△
5-H	4	Denso	○	○	○	○	○	△
		Samsung	○	○	○	○	○	△
		iPad	○	○	○	○	○	△
	8	Denso	○	○	○	○	○	△
		Samsung	○	○	○	○	○	△
		iPad	○	○	○	○	○	△
10-M	4	Denso	○	○	○	○	○	×
		Samsung	○	○	○	○	○	×
		iPad	○	○	○	○	○	×
	8	Denso	○	○	○	○	○	×
		Samsung	○	○	○	○	○	×
		iPad	○	○	○	○	○	×
10-H	4	Denso	○	○	○	○	○	×
		Samsung	○	○	○	○	○	△
		iPad	○	○	○	○	○	△
	8	Denso	○	○	○	○	○	△
		Samsung	○	○	○	○	○	×
		iPad	○	○	○	○	○	△

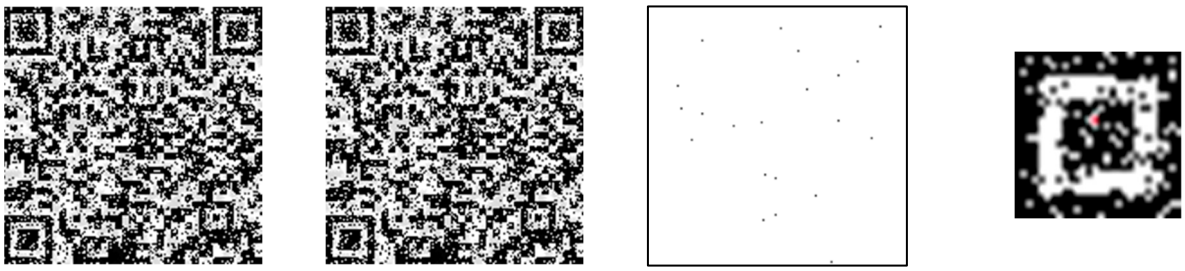
From Table 3, the following facts can be observed. (1) When the QR code stego-image—where the secret message was inserted either exclusively into the black modules with $k_b \in \{25, 50, 75, 100, 125\}$ or exclusively into the white modules with $k_w \in \{25, 50, 75, 100, 125\}$—is scanned by the aforementioned barcode readers, the original data encoded in the QR code are successfully decoded regardless of whether w is 4 or 8. (2) For the QR code stego-image in which the secret message was embedded into both black and white modules with $k_b = k_w \in \{25, 50, 75, 100\}$, the barcode readers described above were able to successfully decode the original encoded data with w set to either 4 or 8. However, when $k_b = k_w = 125$, the original encoded data could not be successfully decoded in a single scan. (3) From (1) and (2), the readability of QR code stego-images with secret messages embedded in both black and white modules is slightly lower than that of stego-images with messages embedded in either module alone.

Notably, for the corresponding digital stego-images generated by embedding the secret message into both black and white modules, the embedded secret messages were correctly extracted using Algorithm 5 before printing, irrespective of the QR code readability outcomes obtained in the subsequent barcode-reader tests. In this barcode-reader experiment, the black-module-only and white-module-only cases were used solely for QR code readability evaluation and were not included in the hidden-message recovery evaluation.

4.6. Assessment of the Proposed Extraction Method Under Random Noise

The robustness of the proposed message extraction algorithm (Algorithm 5) against random noise was examined through small-scale exploratory experiments. Random flipping of black and white pixels in QR code stego-images was employed to simulate noise interference. The tested nominal pixel-flipping rate started at 0.1% and was increased in increments of 0.1% until message extraction first failed. For each target noise level, a noisy image was generated directly from the stego-image in a single scan. The same pseudorandom seed was used at all tested noise levels, so that the same pseudorandom sequence was reproduced. Consequently, the sets of flipped pixels at successive noise levels were nested, thereby forming a cumulative noise realization.

Enlarged views of noisy stego-images of a 5-H QR code (w = 4, k_b = 25 and k_w = 25) are illustrated in Figure 16. With 13.6% noise, shown in Figure 16a, the embedded message remains recoverable using Algorithm 5, but extraction fails when the noise rises to 13.7% as in Figure 16b. The difference between these two images is presented in Figure 16c, exhibiting salt-and-pepper-like noise. A zoomed-in top-right finder pattern from Figure 16b is given in Figure 16d, where the red pixel marks one of the 21 additional noise instances introduced by the increase from 13.6% to 13.7%. The largest noise level at which successful extraction was observed in a tested random-noise realization (13.6% in this case) is referred to herein as the *observed highest tolerable noise rate*. This value represents the highest tolerable noise rate observed for the particular random-noise realization examined and should not be interpreted as a statistically established or universally applicable noise-tolerance threshold. Accordingly, the present exploratory experiments are not intended to estimate the overall extraction-success probability or the statistical distribution of noise-tolerance levels.



(**a**) (**b**) (**c**) (**d**)

Figure 16. Enlarged views of noisy stego-images of a 5-H QR code under random noise. (**a**) 13.6% noise: embedded message remains recoverable. (**b**) 13.7% noise: message extraction fails. (**c**) Difference image between (**a**) and (**b**) with 21 altered pixels, resembling salt-and-pepper noise. (**d**) Zoomed-in top-right pattern of (**b**), with a red pixel indicating one of the 21 additional noise instances introduced when the level increases from 13.6% to 13.7%.

The overall results are summarized in Table 4. From the data, two observations can be made: (1) under the evaluated conditions, the observed highest tolerable noise rates for each QR code showed relatively little variation across the tested change parameters (k_b and k_w); and (2) in the tested random-noise realizations, message extraction failed when the noise level was increased beyond the corresponding observed highest tolerable noise

rate, mainly because of incorrect estimation of the module side length, which subsequently caused errors in module reconstruction and bit recovery.

Table 4. Highest tolerable noise rates observed in the tested random-noise realizations.

QR Code Type	w	Embedding Change Parameters in Both Black and White Pixels				
		25	50	75	100	125
5-M	4	10.3%	10.3%	10.3%	10.3%	10.3%
	8	11.8%	11.8%	11.8%	11.8%	11.8%
5-H	4	13.6%	13.6%	13.6%	13.6%	13.6%
	8	11.7%	11.7%	11.7%	11.7%	11.7%
10-M	4	12.2%	12.2%	12.2%	12.2%	12.2%
	8	11.8%	11.8%	11.8%	11.8%	11.8%
10-H	4	11.6%	11.6%	11.6%	11.6%	11.6%
	8	11.6%	11.6%	11.6%	11.6%	11.6%

4.7. Evaluation of Stego-Image Readability at the Observed Highest Tolerable Noise Rates

As a further exploratory evaluation, additional experiments were carried out to evaluate the readability of stego-images at the observed highest tolerable noise rates identified in Section 4.6 for the tested random-noise realizations. The corresponding results are summarized in Table 5, from which the following observations can be made: (1) some stego-images that were readable before the addition of random pixel-flipping noise became unreadable after the noise was introduced; (2) all cases with the change parameters set to 25 remain readable; (3) all cases with the change parameters set to 50 are also readable, except for the 10-M QR code with a module side length of 4 when tested using Denso; and (4) the iPad generally showed better readability than the other two testing devices under the evaluated conditions.

Table 5. Results of stego-image readability tests using different barcode-reading tools at the observed highest tolerable noise rates.

QR Code Type	w	Testing Equipment	Embedding Change Parameters in Both Black and White Pixels				
			25	50	75	100	125
5-M	4	Denso	○	○	×	×	×
		Samsung	○	○	×	×	×
		iPad	○	○	○	×	×
	8	Denso	○	○	○	○	×
		Samsung	○	○	○	○	×
		iPad	○	○	○	○	×
5-H	4	Denso	○	○	△	×	×
		Samsung	○	○	×	×	×
		iPad	○	○	△	×	×
	8	Denso	○	○	○	○	×
		Samsung	○	○	○	○	×
		iPad	○	○	○	○	×

Table 5. *Cont.*

QR Code Type	w	Testing Equipment	Embedding Change Parameters in Both Black and White Pixels				
			25	50	75	100	125
10-M	4	Denso	○	×	×	×	×
		Samsung	○	○	×	×	×
		iPad	○	○	○	×	×
	8	Denso	○	○	○	×	×
		Samsung	○	○	○	×	×
		iPad	○	○	○	○	×
10-H	4	Denso	○	○	×	×	×
		Samsung	○	○	○	×	×
		iPad	○	○	○	×	×
	8	Denso	○	○	○	○	×
		Samsung	○	○	○	○	×
		iPad	○	○	○	○	×

4.8. Comparison of Embedding Capacity and Rate with Other Techniques

In addition, a comparison of the message embedding capacities and embedding rates (in terms of bpm) of the proposed method with those of four related methods—Lin and Chen [12], Lin et al. [15], Huang et al. [24], and Wu and Wu [27]—was conducted. The capacity values of these related methods were collected from the literature for five QR code versions and the four error-correction levels, where available, and the corresponding embedding rates were determined using a common module-based normalization. The results are presented in Table 6.

The methods included in Table 6 were selected as representative approaches from different categories of QR code information hiding for which sufficiently detailed capacity information is available to support a literature-based quantitative comparison. Specifically, Lin and Chen [12] and Lin et al. [15] represent methods based on data- or error-correction-codeword modification; Huang et al. [24] represents a method that exploits QR code encoding regions or coding rules, specifically by embedding secret data in the padding region; and Wu and Wu [27] represents a method based on module content modification.

The subcell-based methods of Teraura and Sakurai [25] and Erlangga and Barmawi [26] also belong to the category of module content modification but are not included in Table 6 because the available reported results do not provide a sufficiently consistent basis for quantitative normalization.

As shown in Table 6, the proposed method achieves substantially higher embedding rates than the selected comparison methods under the adopted comparison framework. However, because the reported results were obtained under different operating assumptions and experimental conditions, the comparison should be interpreted as indicative rather than as a strictly equivalent head-to-head comparison. These differences include QR code contents, mask-selection procedures, module sizes and spatial-resolution requirements, embedding mechanisms, error-correction usage, distortion or perceptual-quality constraints, capacity-determining conditions, and readability-testing settings.

Table 6. Comparison of maximum embedding capacities and embedding rates.

Version	No. of Modules in QR Code	Error Correction Level	Lin & Chen [12]		Lin et al. [15]		Huang et al. [24]		Wu & Wu [27]		Proposed Method	
			Capacity (bits)	Rate (bpm)	Capacity (bits)	Rate (bpm)	* Capacity (bits)	Rate (bpm)	Capacity (bits)	Rate (bpm)	Capacity (bits)	Rate (bpm)
1	441	L	50	0.11	48	0.11	57	0.13	140	0.32	870	1.97
		M	83	0.19	80	0.18	48	0.11	128	0.29	877	1.99
		Q	97	0.22	96	0.22	39	0.09	135	0.31	865	1.96
		H	72	0.16	128	0.29	27	0.06	131	0.30	858	1.95
10	3249	L	613	0.19	576	0.18	822	0.25	1729	0.53	6330	1.95
		M	1086	0.33	1040	0.32	648	0.20	1403	0.43	6222	1.92
		Q	1232	0.38	1536	0.47	462	0.14	1428	0.44	6243	1.92
		H	976	0.30	1792	0.55	366	0.11	1445	0.44	6162	1.90
20	9409	L	1853	0.20	1792	0.19	2583	0.27	4235	0.45	17,760	1.89
		M	3474	0.37	3328	0.35	2007	0.21	4294	0.46	17,762	1.89
		Q	3880	0.41	4800	0.51	1455	0.15	4465	0.47	17,864	1.90
		H	3080	0.33	5600	0.60	1155	0.12	4422	0.47	17,874	1.90
30	18,769	L	3735	0.20	3600	0.19	5205	0.28	8303	0.44	35,270	1.88
		M	6752	0.36	6496	0.35	4119	0.22	8408	0.45	35,422	1.89
		Q	7880	0.42	9600	0.51	2955	0.16	9243	0.49	35,711	1.90
		H	5960	0.32	11,520	0.61	2235	0.12	8780	0.47	35,400	1.89
40	31,329	L	6249	0.20	6000	0.19	8868	0.28	13,635	0.44	58,928	1.88
		M	11,445	0.37	10,976	0.35	7002	0.22	13,960	0.45	58,857	1.88
		Q	13,328	0.43	16,320	0.52	4998	0.16	14,460	0.46	59,131	1.89
		H	10,208	0.33	19,440	0.62	3828	0.12	14,289	0.46	58,774	1.88
Average embedding rate (bpm)			0.29		0.37		0.17		0.43		1.91	

* Reported maximum capacity with the encoded data length set to zero (maximum padding region).

5. Conclusions

In this paper, a QR code information-hiding method based on module content modification through corner-pixel grayscale adjustment is proposed. Secret bits are embedded by modifying selected module corner pixels, while embedding at adjacent corner pixels with identical original grayscale values is restricted to reduce undesirable local contrast. Bilinear-interpolation-based gradient operations are further applied within modules to improve visual fidelity. A theoretical analysis of bilinear interpolation and quarter-subregion averaging is developed, leading to Theorem 1, which establishes strict decision thresholds for message extraction under the assumed bilinear-interpolation model when the relevant pixel values and module geometry are preserved.

Experiments involving 24 QR code images across six versions and four error-correction levels yielded an overall embedding rate of 1.917 ± 0.036 bpm. Objective PSNR, SSIM, and VIFp analyses quantitatively characterized the trade-off between the magnitude of the embedding changes and the visual fidelity of the resulting stego-images. The barcode-reader experiments showed that the original QR code content generally remained readable across the tested commercial devices and parameter settings, while the embedded messages were correctly extracted from the corresponding digital stego-images generated by embedding the secret message into both black and white modules before printing. In addition, the small-scale exploratory random-noise experiments showed that hidden-message recovery remained possible up to the observed highest tolerable noise rates in the tested realizations. A literature-based comparison with four representative methods indicated substantially higher embedding rates under the adopted comparison framework. These findings should be interpreted according to their respective evaluation criteria: QR code readability concerns decoding of the original QR code content, hidden-message recoverability concerns

extraction of the embedded message, and robustness in the present study refers specifically to the evaluated random-noise conditions.

Several practical limitations should nevertheless be noted. End-to-end hidden-message recovery after physical acquisition or post-processing, such as print–scan or camera capture, lossy compression, resampling, blurring, or geometric distortion, has not been systematically investigated. Such operations may alter the pixel values or module geometry assumed by Theorem 1; therefore, the theoretical extraction thresholds should not be interpreted as providing a general robustness guarantee under these transformations. Moreover, the reported random-noise results were obtained from small-scale exploratory tests in which noise was cumulatively increased within each tested realization. The observed highest tolerable noise rates are therefore realization-specific rather than statistically established thresholds. Future work should investigate end-to-end recovery under practical acquisition and post-processing conditions, possible adaptation of the extraction procedure or thresholds, and independent repeated random-noise trials to estimate extraction-success probabilities and associated statistical variability. Broader readability evaluations involving additional QR code images, embedding conditions, random-noise realizations, and testing devices may also be considered.

From a security perspective, the proposed method does not guarantee steganographic undetectability against an informed adversary or dedicated steganalysis. The grayscale modifications and histogram characteristics used for automatic parameter estimation may provide statistical evidence of hidden-data embedding, and comparison with a reconstructed standard binary QR code may reveal modified regions. In addition, the current implementation employs the standard C rand() function to generate the pseudorandom visiting order and therefore does not provide cryptographically secure randomization. Future work may investigate dedicated steganalysis, the use of a CSPRNG or a cryptographically secure pseudorandom permutation with an adequately sized secret key, and human-subject perceptual evaluation, such as a visual Turing test with expert participants. When confidentiality, integrity, or authenticity of the hidden payload is required, appropriate encryption and authentication mechanisms should also be applied before embedding.

Finally, the comparison with existing methods is based primarily on results reported in the literature rather than on a unified reimplementation of all compared methods. Although a common module-based normalization was adopted where applicable, differences in QR code contents, embedding mechanisms, error-correction usage, spatial-resolution requirements, capacity-determining conditions, and readability-testing settings prevent a strictly controlled head-to-head comparison. The comparative results should therefore be regarded as indicative. Future work may reimplement selected representative methods under a unified experimental framework using common QR code datasets, payloads, image dimensions, perceptual-quality criteria, and readability tests. Within these stated limitations, the results support the feasibility of the proposed framework for high-capacity QR code information hiding under the evaluated conditions.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app16178498/s1, Data S1: Supplementary experimental data, including the image and seed manifests, experimental configurations, numerical experimental results, and raw runtime measurements; Text S1: Supplementary randomization and visiting-order procedures, including C code excerpts for pseudorandom message-bit generation, module-corner visiting orders, and random pixel-flipping noise generation; Data S2: Test QR code images comprising 48 BMP files representing 24 distinct QR code cases rendered with module side lengths of $w = 4$ and $w = 8$; File S1: Supplementary README describing the package contents, image naming convention, quiet-zone handling, random seeds, and experimental conventions.

Author Contributions: Conceptualization, D.-C.W. and Y.-M.W.; methodology, D.-C.W. and Y.-M.W.; software, D.-C.W. and Y.-M.W.; validation, D.-C.W. and Y.-M.W.; formal analysis, Y.-M.W.; investigation, Y.-M.W.; resources, D.-C.W.; data curation, Y.-M.W.; writing—original draft preparation, Y.-M.W.; writing—review and editing, D.-C.W.; visualization, D.-C.W.; supervision, D.-C.W.; project administration, D.-C.W.; funding acquisition, D.-C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by the National Science and Technology Council, Taiwan under Project No. NSC 102-2221-E-327-023.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The test images and experimental configurations associated with the reported experiments are provided in the Supplementary Materials. The experimental configurations include the random seeds used in the experiments. Further implementation information may be requested from the corresponding author.

Acknowledgments: The authors wish to express their sincere gratitude to the anonymous reviewers for their careful evaluation and valuable comments. The authors are particularly grateful to one of the reviewers for the prompt, constructive, and helpful responses to their requests for clarification during the revision process. Special thanks are also extended to the Assistant Editor, for her kind assistance and thoughtful coordination throughout the review and revision process, particularly in facilitating communication with the reviewer. Their comments, guidance, and support have greatly contributed to improving the quality and clarity of this manuscript. The authors are also grateful to the Department of Computer and Communication Engineering, National Kaohsiung University of Science and Technology, Taiwan, for providing the facilities and resources necessary to conduct this research.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Derivations of (5)–(8) and (10)–(11)

The derivations of (5)–(8) and (10)–(11) are given as follows. Since the average intensity of the quarter subregion adjacent to corner A is defined as

$$\overline{P}_A = \frac{1}{N^2} \sum_{\alpha=0}^{N-1} \sum_{\beta=0}^{N-1} P(\alpha,\ \beta), \tag{A1}$$

where $N = \lfloor w/2 \rfloor$. Substituting (1) into (A1) gives

$$\overline{P}_A = \frac{\sum_{\alpha=0}^{N-1} \sum_{\beta=0}^{N-1} [(w-1-\alpha)(w-1-\beta)A + \alpha(w-1-\beta)B + (w-1-\alpha)\beta C + \alpha\beta D]}{N^2(w-1)^2}. \tag{A2}$$

Since the double summation in (A2) is separable,

$$\overline{P}_A = w_0 A + w_1 B + w_2 C + w_3 D, \tag{A3}$$

where

$$w_0 = \frac{\left(\sum_{i=0}^{N-1}(w-1-i)\right)^2}{N^2(w-1)^2}, \tag{A4}$$

$$w_1 = \frac{\left(\sum_{i=0}^{N-1} i\right)\left(\sum_{i=0}^{N-1}(w-1-i)\right)}{N^2(w-1)^2}, \tag{A5}$$

$$w_2 = \frac{\left(\sum_{i=0}^{N-1}(w-1-i)\right)\left(\sum_{i=0}^{N-1} i\right)}{N^2(w-1)^2}, \text{ and} \tag{A6}$$

$$w_3 = \frac{\left(\sum_{i=0}^{N-1} i\right)^2}{N^2(w-1)^2}. \tag{A7}$$

Using

$$\sum_{i=0}^{N-1} i = \frac{N(N-1)}{2} \text{ and } \sum_{i=0}^{N-1}(w-1-i) = N(w-1) - \frac{N(N-1)}{2}, \tag{A8}$$

we obtain

$$\sum_{i=0}^{N-1}(w-1-i) = N\left((w-1) - \frac{N-1}{2}\right). \tag{A9}$$

Substituting (A9) into (A4) gives

$$w_0 = \frac{N^2\left((w-1) - \frac{N-1}{2}\right)^2}{N^2(w-1)^2} = \left(1 - \frac{N-1}{2(w-1)}\right)^2. \tag{A10}$$

Similarly,

$$w_1 = \frac{N\left((w-1) - \frac{N-1}{2}\right)\cdot\frac{N(N-1)}{2}}{N^2(w-1)^2} = \frac{(N-1)(2w-N-1)}{4(w-1)^2}. \tag{A11}$$

By symmetry,

$$w_2 = w_1 = \frac{(N-1)(2w-N-1)}{4(w-1)^2}. \tag{A12}$$

Finally,

$$w_3 = \frac{\left(\frac{N(N-1)}{2}\right)^2}{N^2(w-1)^2} = \left(\frac{N-1}{2(w-1)}\right)^2. \tag{A13}$$

Let

$$r = \frac{N-1}{2(w-1)}, \tag{A14}$$

Then,

$$w_0 = (1-r)^2,\ w_1 = w_2 = r(1-r),\ w_3 = r^2. \tag{A15}$$

Therefore,

$$w_0 + w_1 + w_2 + w_3 = 1, \tag{A16}$$

and

$$w_0 > w_1 = w_2 > w_3. \tag{A17}$$

Appendix A.2. Derivation of the Asymptotic Ratio in (12)

The asymptotic ratio in (12) is derived as follows. Since $N = \lfloor w/2 \rfloor$, we have

$$\frac{N-1}{2(w-1)} \to \frac{1}{4},\ \mathrm{w} \to \infty. \tag{A18}$$

Therefore,

$$w_0 = \left(1 - \frac{N-1}{2(w-1)}\right)^2 \to \left(1 - \frac{1}{4}\right)^2 = \frac{9}{16}, \tag{A19}$$

$$w_1 = \frac{(N-1)(2w-N-1)}{4(w-1)^2} \to \frac{3}{16}, \tag{A20}$$

$$w_2 \to \frac{3}{16}, \text{ and} \tag{A21}$$

$$w_3 = \left(\frac{N-1}{2(w-1)}\right)^2 \to \left(\frac{1}{4}\right)^2 = \frac{1}{16}. \tag{A22}$$

Consequently,

$$w_0 : w_1 : w_2 : w_3 \to 9 : 3 : 3 : 1. \tag{A23}$$

Appendix A.3. Proof of Theorem 1

Since

$$N = \left\lfloor \frac{w}{2} \right\rfloor \leq \frac{w}{2}, \tag{A24}$$

we have

$$N - 1 < \frac{w-1}{2}, \tag{A25}$$

which implies

$$\frac{N-1}{2(w-1)} < \frac{1}{4}. \tag{A26}$$

From (A10),

$$w_0 = \left(1 - \frac{N-1}{2(w-1)}\right)^2 > \left(\frac{3}{4}\right)^2 = \frac{9}{16} > \frac{1}{2}. \tag{A27}$$

Since

$$w_0 + w_1 + w_2 + w_3 = 1, \tag{A28}$$

we further obtain

$$w_1 + w_2 + w_3 = 1 - w_0 < \frac{1}{2}. \tag{A29}$$

Hence,

$$(w_1 + w_2 + w_3)k_\mathrm{b} < \frac{k_\mathrm{b}}{2} < w_0 k_\mathrm{b}, \text{ and} \tag{A30}$$

$$255 - w_0 k_\mathrm{w} < 255 - \frac{k_\mathrm{w}}{2} < 255 - (w_1 + w_2 + w_3)k_\mathrm{w}. \tag{A31}$$

A. Black modules

If $X = 0$, then

$$0 \leq \overline{P}_x \leq (w_1 + w_2 + w_3)k_\mathrm{b} < \frac{k_\mathrm{b}}{2}. \tag{A32}$$

If $X = k_\mathrm{b}$, then

$$\frac{k_\mathrm{b}}{2} < w_0 k_\mathrm{b} \leq \overline{P}_x \leq k_\mathrm{b}. \tag{A33}$$

Therefore, the intervals $[0, (w_1 + w_2 + w_3)k_\mathrm{b}]$ and $[w_0 k_\mathrm{b}, k_\mathrm{b}]$ are disjoint by (A30). Consequently,

$$\overline{P}_x = \frac{k_b}{2} \tag{A34}$$

is impossible. Therefore,

$$\overline{P}_x < \frac{k_b}{2} \leftrightarrow X = 0, \text{ and} \tag{A35}$$

$$\overline{P}_x > \frac{k_b}{2} \leftrightarrow X = k_b. \tag{A36}$$

B. White modules

If $X = 255 - k_w$, then

$$255 - k_w \leq \overline{P}_x \leq 255 - w_0 k_w < 255 - \frac{k_w}{2}. \tag{A37}$$

If $X = 255$, then

$$255 - \frac{k_w}{2} < 255 - (w_1 + w_2 + w_3)k_w \leq \overline{P}_x \leq 255. \tag{A38}$$

Therefore, the intervals

$$[255 - k_w,\ 255 - w_0 k_w] \text{ and } [255 - (w_1 + w_2 + w_3)k_w,\ 255] \tag{A39}$$

are disjoint by (A31). Consequently,

$$\overline{P}_x = 255 - \frac{k_w}{2}, \tag{A40}$$

is impossible. Therefore,

$$\overline{P}_x < 255 - \frac{k_w}{2} \leftrightarrow X = 255 - k_w, \tag{A41}$$

and

$$\overline{P}_x > 255 - \frac{k_w}{2} \leftrightarrow X = 255. \tag{A42}$$

Therefore,

$$\frac{k_b}{2} \text{ and } 255 - \frac{k_w}{2}. \tag{A43}$$

are strict separating thresholds because the corresponding intervals are disjoint and equality can never occur under the theoretical model. This completes the proof of Theorem 1.

References

1. Cox, I.J.; Miller, M.L.; Bloom, J.A.; Fridrich, J.; Kalker, T. *Digital Watermarking and Steganography*, 2nd ed.; Morgan Kaufmann: Burlington, MA, USA, 2008. [CrossRef]
2. Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362. [CrossRef]
3. Wu, D.C.; Hsu, Y.T. Authentication of LINE chat history files by information hiding. *ACM Trans. Multimed. Comput. Commun. Appl.* **2022**, *18*, 22. [CrossRef]
4. Alanzy, M.; Alomrani, R.; Alqarni, B.; Almutairi, S. Image steganography using LSB and hybrid encryption algorithms. *Appl. Sci.* **2023**, *13*, 11771. [CrossRef]
5. *ISO/IEC 18004:2024*; Information Technology—Automatic Identification and Data Capture Techniques—QR Code Bar Code Symbology Specification. ISO: Geneva, Switzerland, 2024.
6. Pandya, K.H.; Galiyawala, H.J. A survey on QR codes in context of research and application. *Int. J. Emerg. Technol. Adv. Eng.* **2014**, *4*, 258–262.
7. Chou, K.-C.; Wang, R.-Z. Dual-Message QR Codes. *Sensors* **2024**, *24*, 3055. [CrossRef] [PubMed]

8. Dey, S.; Mondal, K.; Nath, J.; Nath, A. Advanced steganography algorithm using randomized intermediate QR host embedded with any encrypted secret message: ASA_QR algorithm. *Int. J. Mod. Educ. Comput. Sci.* **2012**, *4*, 59–67. [CrossRef]
9. Gaikwad, A.M.; Singh, K.R. Information hiding using image embedding in QR codes for color images: A review. *Int. J. Comput. Sci. Inf. Technol.* **2015**, *6*, 278–283.
10. Bhardwaj, J.; Panwar, R. Safeguarding information in QR codes through steganographic technique. In Proceedings of the International Conference on Communication, Control and Intelligent Systems (CCIS), Mathura, India, 6–7 December 2024. [CrossRef]
11. Chiang, Y.J.; Lin, P.Y.; Wang, R.Z.; Chen, Y.H. Blind QR code steganographic approach based upon error correction capability. *KSII Trans. Internet Inf. Syst.* **2013**, *7*, 2527–2543. [CrossRef]
12. Lin, P.Y.; Chen, Y.H. High payload secret hiding technology for QR codes. *EURASIP J. Image Video Process.* **2017**, *2017*, 14. [CrossRef]
13. Huang, P.C.; Lin, Y.H.; Chang, C.C.; Liu, Y. Efficient scheme for secret hiding in QR code by improving exploiting modification direction. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 2348–2365. [CrossRef]
14. Barmawi, A.M.; Yulianto, F.A. Watermarking QR code. In Proceedings of the 2nd International Conference on Information Science and Security (ICISS), Seoul, Republic of Korea, 14–16 December 2015. [CrossRef]
15. Lin, P.Y.; Lan, W.S.; Chen, Y.H.; Wu, W.C. A confidential QR code approach with higher information privacy. *Entropy* **2022**, *24*, 284. [CrossRef] [PubMed]
16. Li, L.; Wang, R.L.; Chang, C.C. A digital watermark algorithm for QR code. *Int. J. Intell. Inf. Process.* **2011**, *2*, 29–36. [CrossRef]
17. Sun, M.; Si, J.; Zhang, S. Research on embedding and extracting methods for digital watermarks applied to QR code images. *N. Z. J. Agric. Res.* **2010**, *50*, 861–867. [CrossRef]
18. Tiwari, S.K.; Sharma, D.; Swami, S. QR code image steganography using 2-level DWT and DES. In Proceedings of the 4th OPJU International Technology Conference (OTCON 2025), Raigarh, India, 9–11 April 2025. [CrossRef]
19. Lin, P.Y. Distributed secret sharing approach with cheater prevention based on QR code. *IEEE Trans. Ind. Inform.* **2016**, *12*, 384–392. [CrossRef]
20. Pan, J.S.; Liu, T.; Yan, B.; Yang, H.M.; Chu, S.C. Using color QR codes for QR code secret sharing. *Multimed. Tools Appl.* **2022**, *81*, 15545–15563. [CrossRef]
21. Wang, R.Z.; Wang, L.Y.; Chen, K. Extended dual-message QR codes with visual secret sharing. *IEEE Sens. Lett.* **2025**, *9*, 6005504. [CrossRef]
22. Zhou, Y.; Luo, W. A QR data hiding method based on redundant region and BCH. *J. Big Data* **2021**, *3*, 127–133. [CrossRef]
23. Koptyra, K.; Ogiela, M.R. Steganography in QR codes—Information hiding with suboptimal segmentation. *Electronics* **2024**, *13*, 2658. [CrossRef]
24. Huang, P.-C.; Chang, C.-C.; Li, Y.-H.; Liu, Y. High-payload secret hiding mechanism for QR codes. *Multimed. Tools Appl.* **2019**, *78*, 22331–22350. [CrossRef]
25. Teraura, N.; Sakurai, K. Information hiding in subcells of a two-dimensional code. In Proceedings of the 1st IEEE Global Conference on Consumer Electronics (GCCE 2012), Tokyo, Japan, 2–5 October 2012; pp. 652–656. [CrossRef]
26. Erlangga, W.; Barmawi, A.M. Increasing secret data hiding capacity in QR code using 3 × 3 subcells. In *Digital Forensics and Watermarking, Proceedings of the 15th International Workshop on Digital Forensics and Watermarking (IWDW 2016), Beijing, China, 17–19 September 2016*; Shi, Y.Q., Kim, H.J., Perez-Gonzalez, F., Liu, F., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10082, pp. 327–342. [CrossRef]
27. Wu, D.C.; Wu, Y.M. Covert communication via the QR code image by a data hiding technique based on module shape adjustments. *IEEE Open J. Comput. Soc.* **2020**, *1*, 12–34. [CrossRef]
28. Laboratory for Image and Video Engineering (LIVE). Image & Video Quality Assessment Algorithms; University of Texas at Austin: Austin, TX, USA, n.d. Available online: https://live.ece.utexas.edu/research/Quality/index_algorithms.htm (accessed on 1 August 2026).
29. Sheikh, H.R.; Bovik, A.C. Image information and visual quality. *IEEE Trans. Image Process.* **2006**, *15*, 430–444. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.