This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

Two-Layer QR Codes

Tailing Yuan, Yili Wang, Kun Xu, *Member, IEEE,* Ralph R. Martin, Shi-Min Hu, *Senior Member, IEEE*

***Abstract*—A quick-response code (QR code) is a two-dimensional code akin to a barcode which encodes a message of limited length. In this paper, we present a variant of QR code, a *two-layer QR code*. Its two-layer structure can display two alternative messages when scanned from two different directions. We propose a method to generate such two-layer QR codes encoding two given messages in a few seconds. We also demonstrate the robustness of our method on both synthetic and fabricated examples. All source code will be made publicly available.** [1]

***Index Terms*—QR code, Two-layer QR code, barcode**

I. Introduction

A quick response code (QR code) is a two-dimensional barcode, consisting of black and white squares, called modules. It encodes information such as a URL or a text message. By scanning a QR Code using a mobile phone, a user can get immediate access to its content. QR codes have been widely used in numerous applications such as advertising and digital payments, as they are easy to create and scan.

A QR code can encode only a single message. However, encoding two or more messages is useful in many scenarios. For example, a store may provide alternative payment methods, an app may need different download links for different operating systems, and an exhibition item may have descriptions in multiple languages. While previous work [1], [2] has considered combining two messages in one QR code, they target particular applications. The former targets separate public and private messages, and requires a specialized reader for the latter. The second uses device characteristics (such as its operating system) to determine which alternative message to retrieve; the user is not involved in selecting which alternative message is desired. Such specific usages do not encompass all applications of QR codes needing multiple messages.

Various kinds of art exhibit different effects when viewed or illuminated from different angles or distances, such as *lenticular images* [3] which can change the displayed image depending on the viewing angle, *hybrid images* [4] which have different interpretations depending on the viewing distance, and *shadow art sculptures* [5] which cast different meaningful shadows under different lighting directions.

Inspired by such work [3]–[5], we present a variant of QR code called *two-layer QR code*, in which two messages are stored and can be retrieved separately from two different viewing directions. It is a specially designed two-layer structure, with a top layer and a bottom layer. Each layer contains a matrix of modules. The bottom layer modules are black or white, while the top layer modules may be transparent, allowing the code to appear differently from different viewing directions. The two-layer QR code can thus display two valid QR codes, encoding two different messages, by changing the view.

Our method is thus more generally applicable than [1], [2] and could be used in almost any real-world scenarios where two messages are needed. These might include providing alternative payment methods, giving different app download links for different operating systems, providing explanatory text in different languages, or giving multiple stories for an exhibit in a museum. Another possible use is in gaming, where two or more people sit at a table opposite each other. A two-layer QR code on the table would naturally be scanned from different directions, e.g., to provide separate messages to a card dealer and other card players. For puzzle games, scanning from the left could show puzzle questions, and scanning from the right could give the answers. Using a single code ensures that the questions will always be correctly associated with the answers, which may not always occur if two separate QR codes are used (one may get lost, become defaced, etc.).

As well as explaining the structure of our two-layer QR code, we also give a fully automatic algorithm to create a two-layer QR code encoding two given messages. Specifically, we optimize the module values of both layers, with the aim of maximizing its error correction capacity while also encoding the two messages. For speed, we have developed a *two-step optimization scheme* and *merge and reduce operators* to reduce the solution space. The optimization problem is then solved by simulated annealing in just a few seconds. We also show how to physically fabricate a two-layer QR code.

Our two-layer QR codes can also be combined with existing QR code beautification approaches, to display a visual logo within the QR code display area. We demonstrate the robustness of our method on a variety of examples, including both synthetic and fabricated ones.

II. Related Work

A. QR Code Beautification

Although a QR code is machine readable, it appears as a random pattern of black and white modules and is not visually meaningful. To address this, various works have considered QR code beautification, allowing e.g. the embedding of visual information such as a logo within a QR code. QR code beautification approaches are generally based on one of 3 techniques: direct superimposition, module modification and padding editing.

Tailing Yuan, Kun Xu, Shi-Min Hu are with the Department of Computer Science and Technology, Tsinghua University, Beijing, 100084 CN. Kun Xu is the corresponding author, email: xukun@tsinghua.edu.cn.

Yili Wang is with the Department of Foreign Languages and Literatures, Tsinghua University, Beijing, 100084 CN.

Ralph R. Martin was with the School of Computer Science and Informatics, Cardiff University, UK.

[1] https://github.com/yuantailing/two-layer-qrcode

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

1) Direct superimposition: One way to achieve QR code beautification is to directly superimpose a logo or a pattern onto a QR code [6]–[9]. Superimposing a logo will damage some codewords but as long as this is within the error correction capacity, the QR code can still be correctly decoded. Different schemes have been proposed to find appropriate positions, orientations, and scales of superimposed logos without damaging the code. Samretwit and Wakahara [10] measured the reliability of a superimposed QR code at different positions and scales. As an alternative to superimposition, Baharav and Kakarala [11] achieved QR code beautification by image blending.

2) Module modification: A QR code remains decodable as long as the values in the central pixels of each module are unchanged. Several beautification approaches [12]–[19] have been proposed, in which carefully crafted modules, usually of a smaller size and only covering the central pixels, replaces the original modules. For example, Chu et al. [12] generated *halftone QR codes* by subdividing modules into 3×3 submodules. Lin et al. [13] embellished QR codes by superimposition as well as stylizing module shape based on a specified exemplar. Gao et al. [14] showed how to embed QR codes into color images, making QR code perceptually invisible to humans but still machine-readable.

3) Padding editing: If the length of the encoded message is less than the code's capacity, padding bits follow in each data codeword. The values of these padding bits can be freely changed without altering the message. By taking advantage of this property, Fujita et al. [20] use non-systematic encoding of the Reed-Solomon code to allow free editing of a selected area without reducing the code's error correction capability. Other methods [21]–[27] have been proposed for generating colored QR codes by combining padding editing and module modification.

B. QR Code Alternatives

Various alternative 2D barcode formats have been proposed for different purposes. Picture-embedding 2D barcodes like *2-D image code* [28], *PiCode* [29], and *RA (Robust and Aesthetic) code* [30] have been proposed to embed images in 2D barcodes. Colored QR codes [31]–[33] have been proposed to increase data capacity by using more color channels. Yang et al. [34] proposed *ARTcode*, which look like an image, retaining aesthetic content and decoding feasibility in one visual pattern. However, a common drawback of all such codes is that additional software is needed when decoding.

C. QR Codes combining two messages

Other previous works [1], [2] have considered how to combine two messages into one QR code. The two-level QR code (2LQR) [1] stores a public message and an additional private message. The messages are of different kinds. The public message can be read by any QR code reader, but the private message can only be read by a specialized reader. This method is designed specifically for private message sharing and document authentication. However, it is not suited to general scenarios; users are required to install a specialized decoding App for the private message.

In [2], two App download links can be combined into one QR code, including e.g. one link for iOS and another for Android. This work does not modify QR code design or structure: the QR code simply stores only a single link to an external server. The linked webpage is redirected to the desired download link as determined by the operating system of the device scanning the QR code. While working well for this target usage, it has limited use in general scenarios. An Internet connection is required for message selection, while standard QR code decoding can be done offline. Users are not involved in choosing the message decoded, which is determined solely by the device characteristics (unless additional information is input). For example, this approach cannot readily use a single QR code for the puzzle game scenario (provide either the puzzle questions or answers).

While the above methods [1], [2] are specialized for particular applications, our method is more general and flexible. We allow two arbitrary messages, and the user can freely choose which message to read simply by viewing our code from different viewing directions, using any standard QR code reader. Changing viewing angles to obtain different messages provides a natural, simple interface which is easy to learn and to perform, and follows standard QR code practice. We do not require an Internet connection, external server, or specialized readers.

A disadvantage of our two-layer QR codes is that they require fabrication rather than simple printing. However, this only causes more work for the content provider (who creates the code), while users (who scan the code) can use existing QR code readers. The only difference to them is that the code is scanned from the left or right directions instead of from the front. We believe this is a reasonable trade-off for the potential benefits obtained.

The remainder of the paper is organized as follows. We briefly review standard QR codes in Section III, then explain the basis of our two-layer QR codes in Section IV. We present an algorithm for their generation in Section V. We provide experimental results in Section VI and an evaluation in Section VII. Finally, we conclude the paper in Section VIII.

III. QR Code Basics

To aid understanding, and introduce terminology, we briefly remind the reader of the structure and operation of an original QR code.

1) Structure: A QR code is a 2D image consisting of black and white squares: see Figure 1. Its parameters and components include:

- A **module** is the smallest element (black or white square) of the QR code. Each module represents 1 bit of data, i.e., a black module represents 1 while a white module represents 0.
- A **codeword** consists of 8 modules and represents 8 bits of data.
- An **error correction block**, or **block** for short, consists of a sequence of **codewords**. A block represents a message

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

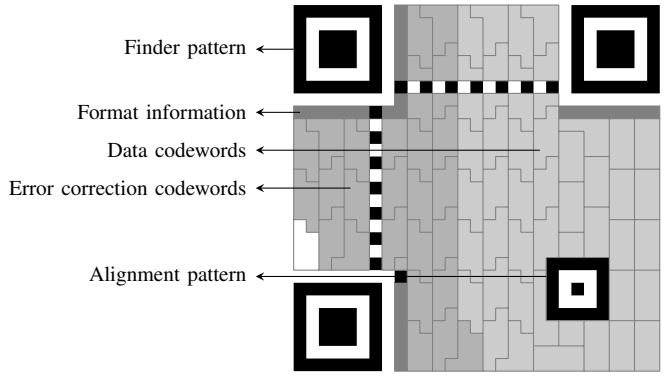


Fig. 1. QR code structure. Finder patterns and alignment patterns are located at fixed positions. The format information records the error correction level and data masking pattern. A codeword consists of 8 modules and represents 8 bits of data.

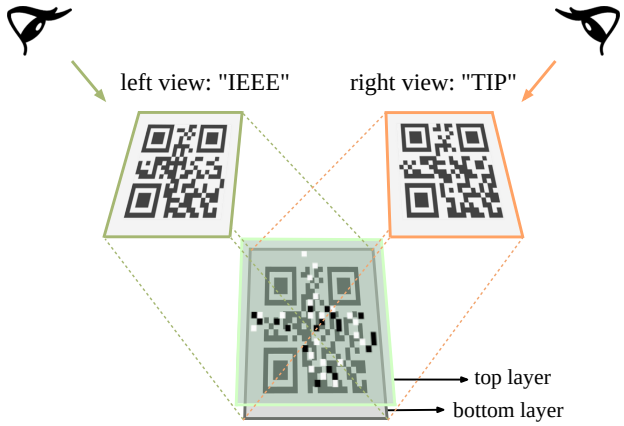


Fig. 2. A two-layer QR code consists of a top layer and a bottom layer with a small gap between them. The bottom layer is composed of black and white modules, while the modules of the top layer may also be transparent. When scanned from the left view, this QR code appears as a standard QR code encoding the string "IEEE"; when scanned from the right, it appears as a QR code encoding a different string, "TIP". In all figures, transparent modules are shown in translucent green to make them visible; in the actual QR code they are transparent.

with a specific length. The codewords within it comprise *data codewords* and *error correction codewords*. The former represent the underlying message, while the latter are Reed-Solomon [35] checksums computed from the data codewords, helping to make the QR code robust to noise and damage. A block with p codewords in total, out of which k are data codewords, can be correctly recovered if it has no more than $\lfloor (p-k)/2 \rfloor$ incorrect codewords.

- **Finder patterns** and **alignment patterns** are located at fixed positions and are used to locate and orient the QR code when scanning.
- **Format information** stores parameters describing the QR code, including its *error correction level (EC level)* and *data masking pattern*. The EC level determines the level of error tolerance, via the numbers of data codewords and error correction codewords in each block. Four EC levels (L, M, Q, and H) exist, with increasing reliability but decreasing data capacity. One of 8 predefined data masking patterns must be selected. A further parameter, *version*, defines the size of the QR code: a QR code having version V has $(4V+17) \times (4V+17)$ modules. The version and EC level together determine the data capacity of a QR code, i.e., the length of the message it can represent.

Further details can be found in the specification ISO/IEC 18004:2015 [36].

2) Message encoding: To encode a given message as a QR code, it is first converted to a sequence of bits using one of four encoding modes: numeric, alphanumeric, byte, and Kanji. Different modes use different character sets, with different numbers of bits for each character. Suitable values of version and EC level are selected, based on message length and desired reliability. Next, the sequence is put into data codewords in each block in a particular order, and corresponding error correction codewords are computed. Finally, the modules are bit-wise XORed with one of the eight predefined data masking patterns, using the one which best avoids adjacent modules appearing in the same color.

3) Message decoding: To decode a message in a scanned QR code, the code is first located by use of the finder and alignment patterns. The value of each module is obtained by binarization. Next, the EC level and data masking pattern are obtained from the format information. The bit sequence of the message may then be recovered block by block, the error correction codewords helping to repair noise and damage.

IV. Two-Layer QR Code

In this section, we introduce our *two-layer QR codes*, which aim to display two valid QR codes when scanned from two different views. It has a specially designed structure, which consists of a top layer and a bottom layer. The bottom layer is composed of black and white modules, which is similar to a standard QR code. The top layer has a similar number of modules (see later), but some of them may be transparent. The space between the two layers allows our proposed structure to have different appearances when scanned from different directions. Figure 2 shows the structure of a two-layer QR code.

The bottom and top layers are designed as follows. The bottom layer has $N \times N$ modules. Each module on the bottom layer is colored white (for 0) or black (for 1). The top layer has $(N+1) \times N$ modules, with $N+1$ and N in x and y directions respectively. Each of its modules may be white (0), black (1), or transparent (t). The top and bottom layers are aligned along the y-axis but symmetrically offset by 0.5 units along the x-axis (where the width of a module is 1 unit). A typical case is a QR code with version 5, which has $N = 37$.

When viewed from a carefully chosen direction from the left side, the projected offset between top and bottom layers will change exactly to 0. Similarly, when viewed from a specific direction from the right side, the projected offset changes exactly to 1. See Figure 3. Let $L_t(i,j)$ be the value of module (i,j) in the top layer ($1 \le i \le N+1, 1 \le j \le N$), and $L_b(i,j)$ be the value of module (i,j) in the bottom layer

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

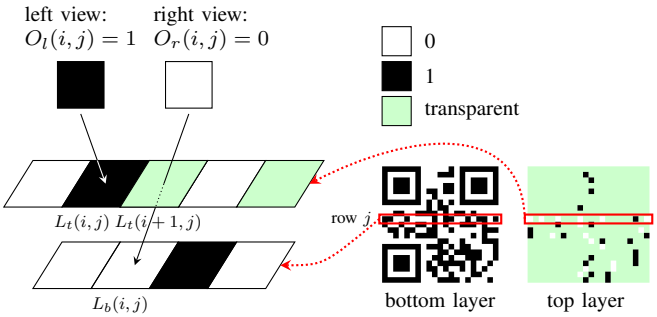


Fig. 3. Each module on the bottom or top layer affects two different modules in the viewing patterns. The color of module $L_b(i,j)$ affects the colors of $O_l(i,j)$ and $O_r(i,j)$.

Fig. 4. A generated two-layer QR code. Left to right: bottom layer, top layer, left view, and right view. These two views show QR codes representing the strings "Two-Layer QR Code" and "Computer Science" respectively.

$(1 \le i,j \le N)$. The left and right viewing patterns, O_l and O_r respectively, can be computed to be:

$$O_l(i,j) = \begin{cases} L_b(i,j) & \text{if } L_t(i,j) = t \text{ (transparent)} \\ L_t(i,j) & \text{otherwise} \end{cases},$$
$$O_r(i,j) = \begin{cases} L_b(i,j) & \text{if } L_t(i+1,j) = t \text{ (transparent)} \\ L_t(i+1,j) & \text{otherwise} \end{cases}. \quad (1)$$

Without further constraints, the left and right viewing patterns are not necessarily valid QR codes. In Section V, we describe a generation algorithm for two-layer QR codes which ensures both left and right viewing patterns are valid QR codes. An example of a generated two-layer QR code is shown in Figure 4.

V. Generation Algorithm

A. Problem Definition

Given two target messages, our goal is to generate top and bottom layer patterns (L_t and L_b) such that the left and right viewing patterns (O_l and O_r) encode the two desired messages. We first use conventional QR code generation algorithms [36] to convert each message into a standard QR code, which we refer to as left and right *target patterns* (T_l and T_r), respectively. They must have the same version (i.e. size) but the EC levels and data masking patterns may differ. For now, we restrict their EC levels and data masking patterns to be the same. We will discuss how to handle cases when they are different in Section VI-A2.

A strict requirement would demand that the viewing patterns should exactly match the target patterns (i.e., $O_l = T_l$, $O_r = T_r$), but this is in practice impossible to achieve. Instead, we require that the same messages can still be decoded from the viewing patterns as from the target patterns. At the same time, we should maximize the reliability of the viewing patterns, i.e. their ability to still be correctly decodable when damaged. To measure the reliability of a two-layer QR code, we define its *effective recovery ratio* E to be:

$$E = \min_{i,s} D(O_s^i, T_s^i), \quad (2)$$

where O_s^i denotes the i-th block in one pattern O_s ($O_s \in \{O_l, O_r\}$, either the left viewing pattern O_l or the right viewing pattern O_r), T_s^i denotes the i-th block in the corresponding target pattern (which covers the same area as O_s^i), and $D(\cdot)$ is the *remaining recovery ratio* for each block. The effective recovery ratio E is thus computed as the minimum of the remaining recovery ratios over all blocks of the two viewing patterns. The remaining recovery ratio D for a block O_s^i is defined as:

$$D(O_s^i, T_s^i) = \left(\lfloor (p-k)/2 \rfloor - \delta(O_s^i, T_s^i)\right)/p, \quad (3)$$

where p and k respectively denote the number of codewords in total, and of data codewords in this block. Here, $\lfloor (p-k)/2 \rfloor$ is the maximum allowed number of mismatched codewords that still allows correct decoding from the QR code structure, while $\delta(O_s^i, T_s^i)$ denotes the number of codewords in block O_s^i which mismatch those in block T_s^i (a codeword consists of 8 modules: if any of its modules mismatches, then the codeword mismatches). The remaining recovery ratio D represents the remaining error capacity of a block.

Formally, we may now consider generation of a two-layer QR code as an optimization problem. Given two target patterns T_l and T_r, the goal is to find top and bottom layer patterns L_t and L_b by minimizing an objective function:

$$\min_{L_t, L_b} -E, \quad (4)$$

where E is the effective recovery ratio defined in Equation 2. It is computed by comparing the module values in target patterns (T_l and T_r) with those in the viewing patterns (O_l and O_r), which are computed from the top and bottom layer patterns (L_t and L_b) according to Equation 1.

Equation 4 represents a combinatorial optimization problem. Since the top layer L_t and the bottom layer L_b contain $N(N+1)$ and N^2 modules, with 3 and 2 possible module values, respectively, the size of the solution space is $2^{N^2}3^{N(N+1)}$, which is far too large to explore exhaustively. We need a more efficient approach.

Recall the definitions of the effective recovery ratio E and remaining recovery ratio D in Equations 2 and 3. They are computed from the set of all mismatched codewords in the viewing patterns, which we call the *full mismatched codeword set*. We observe that E can be computed in two steps, firstly obtaining the full mismatched codeword set by comparing module values of viewing and target patterns, and secondly, computing E from the full mismatched codeword set. We will show how to make use of this separation property in the following subsections.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

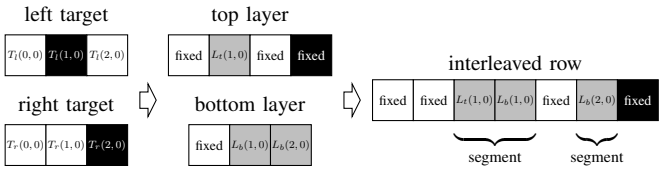


Fig. 5. Fixed modules and segments.

TABLE I
MODULE VALUES WITH DIFFERENT COMBINATIONS OF VALUES FOR L_t AND L_b WHEN $T_l(i,j) = T_r(i,j) = 1$.

			$L_b(i,j)=0$	$L_b(i,j)=1$
$L_t(i,j)=0$	$L_t(i+1,j)=0$	$O_l(i,j)$	0	0
		$O_r(i,j)$	0	0
	$L_t(i+1,j)=1$	$O_l(i,j)$	0	0
		$O_r(i,j)$	**1**	**1**
	$L_t(i+1,j)=t$	$O_l(i,j)$	0	0
		$O_r(i,j)$	0	**1**
$L_t(i,j)=1$	$L_t(i+1,j)=0$	$O_l(i,j)$	**1**	**1**
		$O_r(i,j)$	0	0
	$L_t(i+1,j)=1$	$O_l(i,j)$	**1**	**1**
		$O_r(i,j)$	**1**	**1**
	$L_t(i+1,j)=t$	$O_l(i,j)$	**1**	**1**
		$O_r(i,j)$	0	**1**
$L_t(i,j)=t$	$L_t(i+1,j)=0$	$O_l(i,j)$	0	**1**
		$O_r(i,j)$	0	0
	$L_t(i+1,j)=1$	$O_l(i,j)$	0	**1**
		$O_r(i,j)$	**1**	**1**
	$L_t(i+1,j)=t$	$O_l(i,j)$	0	**1**
		$O_r(i,j)$	0	**1**

B. Fixed Modules and Segments

We next introduce concepts of *fixed modules*, *interleaved rows*, *segments*, and *partial mismatched codeword sets*. Fixed modules are modules whose value can be directly set without optimization. An interleaved row is formed by interleaving modules in a row of the top layer with those in the same row of the bottom layer. A segment is an array of contiguous modules in an interleaved row terminated by fixed modules. See Figure 5. Each segment may contain mismatched modules, resulting in mismatched codewords, which are grouped into the partial mismatched codeword set for the segment. The full mismatched codeword set is the *union* of the partial mismatched codeword sets of all segments.

We now explain how to compute all of the above.

1) Fixed modules: We observe that there are two cases when we can directly find the values of some modules, the fixed modules, in L_t and L_b without the need for optimization. These two cases are:

Case 1: if $T_l(i,j) = T_r(i,j) = c$ (where $c = 0$ or 1), $L_b(i,j)$ can be fixed likewise: $L_b(i,j) = c$.

Case 2: if $T_l(i,j) = T_r(i-1,j) = c$, $L_t(i,j)$ can be fixed likewise: $L_t(i,j) = c$.

Proof: Consider **Case 1** (the proof for **Case 2** is similar). Without loss of generality, suppose $T_l(i,j) = T_r(i,j) = 1$. According to Equation 1, the values of $O_l(i,j)$ and $O_r(i,j)$ (and these two modules only) depend on the value of $L_b(i,j)$. In Table I, we provide the values of $O_l(i,j)$, $O_r(i,j)$ for different combinations of values for L_t and L_b. The viewing patterns should have the same values as the target patterns wherever possible, i.e. we desire $O_l(i,j) = T_l(i,j)$ and $O_r(i,j) = T_r(i,j)$. It is clear that in all cases, setting $L_b(i,j) = 1$ is always better than $L_b(i,j) = 0$. Hence, $L_b(i,j)$ may be fixed as $L_b(i,j) = 1$, without further optimization.

We determine all fixed modules and fix their values accordingly.

2) Segments: Consider the j-th rows of L_t and L_b. We may interleave their modules to form an *interleaved row*:

$$\{L_t(0,j), L_b(0,j), L_t(1,j), \ldots, L_b(N-1,j), L_t(N,j)\},$$

in which top layer modules and bottom layer modules alternate. The value of a viewing pattern module is determined by two neighboring modules in an interleaved row (i.e., one top layer module and one bottom layer module). Therefore, after dividing the interleaved row into segments, we could determine each module value in the viewing pattern from just one corresponding segment instead of the whole row. This allows us to analyze the reliability of each segment separately.

3) Partial mismatched codeword set: Each segment may contain mismatched modules, resulting in mismatched codewords. These form its partial mismatched codeword set. We obtain the full mismatched codeword set by computing the union of the partial mismatched codeword sets of all segments.

C. Two-Step Optimization

By making use of segments, the optimization problem in Equation 4 can be solved using a two-step scheme.

1) Candidate family: A *candidate family* is the set containing all possible partial mismatched codeword sets for a segment (a set of sets). In the first step, we obtain a candidate family for each segment, by iterating through all combinations of a segment's module values. Note that different combinations of module values may result in duplicate partial mismatched codeword sets, which lead to the same full mismatched codeword set, and thus the same effective recovery ratio E. Hence, in a candidate family, we only store unique partial mismatched codeword sets, and remove any duplicates.

Below we show an example of computing a candidate family. Consider a very simple segment A with two modules, one from the top layer and one from the bottom layer. There are 6 possible combinations of module values: $\{c_1, c_2\}$ ($c_1 \in [0,1,t], c_2 \in [0,1]$). Suppose the partial mismatched codeword set for each value combination is as below:

$$P(\{0,0\})\text{: } \{C_l^1\}, \quad P(\{1,0\})\text{: } \{C_r^2\}, \quad P(\{t,0\})\text{: } \{C_l^1\},$$
$$P(\{0,1\})\text{: } \{C_l^1\}, \quad P(\{1,1\})\text{: } \{C_r^2\}, \quad P(\{t,1\})\text{: } \emptyset,$$

where $P(\cdot)$ is the partial mismatched codeword set for a specific combination of module values, C_l^i is the i-th codeword of the left target, and C_r^j is the j-th codeword of the right target; $P(\{0,0\}) = \{C_l^1\}$ means that setting the module values of the segment to $\{0,0\}$ causes one codeword C_l^1 to be mismatched. The candidate family S for segment A is, after removing duplicates:

$$S = \{\emptyset, \{C_l^1\}, \{C_r^2\}\}.$$

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

This candidate family is a set of 3 different partial mismatched codeword sets.

We compute the candidate families for all segments accordingly.

2) Optimization on candidate families: In the second step, we solve the optimization problem in Equation 4. Instead of simultaneously determining values for all modules in L_t and L_b, a problem of size $2^{N^2}3^{N(N+1)}$, we operate on segments, and aim to determine a combination of partial mismatched codeword sets, i.e. to select one partial mismatched codeword set for each segment from its candidate family. These give the full mismatched codeword set and hence the effective recovery ratio E.

Let u be the number of segments, A_i be the i-th segment ($1 \leq i \leq u$), S_i be its candidate family as computed in the first step, and $\|S_i\|$ be the size of S_i. The optimization problem in Equation 4 is reduced to:

$$\min_{\mathbf{s}} -E, \tag{5}$$

where $\mathbf{s} = \{s_1, \ldots, s_u\}$ is an unknown index vector to be found, and s_i gives the index of the selected partial mismatched codeword set from the i-th candidate family S_i ($1 \leq s_i \leq \|S_i\|$).

For each candidate family S_i, there are $\|S_i\|$ choices for the partial mismatched codeword set. Hence, the complexity of the solution space of the problem in Equation 5 is:

$$\prod_{1 \leq i \leq u} \|S_i\|. \tag{6}$$

This two-step scheme greatly reduces the size of the solution space.

D. Merge and Reduce

To further reduce the solution space and accelerate optimization, we introduce two operators on candidate families: the *reduce operator* and the *merge operator*.

1) Reduce operator: Given a candidate family S, let P_1 and P_2 be two different partial mismatched codeword sets in S. If $P_1 \subsetneqq P_2$, P_2 is a *redundant* partial mismatched codeword set in S, since choosing P_1 would always give fewer mismatches than choosing P_2. Hence P_2 can safely be ignored. Reducing a candidate family means removing all such redundant mismatched codeword sets from it. An example of reducing a candidate family is given below:

$$S = \{\{C_l^1\}, \{C_l^1, C_r^2\}\} \xrightarrow{\text{reduce}} S = \{\{C_l^1\}\}.$$

Reducing a candidate family decreases its size and hence the size of the solution space.

2) Merge operator: The merge operator takes two candidate families as input and combines them into one candidate family. Given two candidate families S_1 and S_2, the merged candidate family S is given by:

$$S = \{P_1 \cup P_2 | P_1 \in S_1, P_2 \in S_2\}.$$

At a first glance, the merge operator does not seem to decrease the size of the solution space, since it converts two candidate families (S_1 with size m_1 and S_2 with m_2) into a candidate family with size $m_1 m_2$. However, the merged candidate family may possibly be further reduced. For example, consider two candidates S_1 and S_2 both with size 2:

$$S_1 = \{\{C_l^1\}, \{C_r^2\}\}, \qquad S_2 = \{\{C_l^1\}, \{C_r^3\}\}. \tag{7}$$

The merged candidate family S is:

$$S = \{\{C_l^1\}, \{C_l^1, C_r^2\}\}, \{C_l^1, C_r^3\}, \{C_r^2, C_r^3\}\}.$$

Its size is 4 but it can be decreased to 2 by applying the reduce operator:

$$S \xrightarrow{\text{reduce}} S = \{\{C_l^1\}, \{C_r^2, C_r^3\}\}.$$

For convenience, we define two candidate families to be *mergeable* if at least one identical codeword is contained in both. For example, S_1 and S_2 in Equation 7 both contain the same codeword C_l^1, so S_1 and S_2 are mergeable. If two candidate families are not *mergeable*, there is no benefit in merging them.

3) Merge and reduce procedure: We now show how to systematically make use of the merge and reduce operators, starting from the initial set of all candidate families computed as in Section V-C, i.e., $\Omega = \{S_1, \ldots, S_u\}$.

We repeatedly apply reduce and merge operators to the candidate families in Ω, as follows:

- Reduce all candidate families in Ω.
- *Repeat*
  - Shuffle the ordering of all candidate families in Ω.
  - *For* each mergeable pair S_1, S_2 of candidate families
    - *If* $\|S_1\| \times \|S_2\| <$ threshold m *then*
      - $S \leftarrow$ Merge(S_1, S_2)
      - Remove S_1 and S_2 from Ω
      - $S \leftarrow$ Reduce(S)
      - Add S to Ω
      - *Goto Repeat*
    - *End If*
  - *End For*
- *Until* there are no more pairs in Ω that can be merged.

A naive implementation of enumerating over all mergeable pairs of candidate families takes $u(u-1)/2$ pairwise tests, where u is the number of candidate families. We improve the algorithm performance by building an index from codewords to candidate families, which quickly allows us to find all mergeable pairs. Furthermore, the merge size threshold m is used to avoid very large candidate families: applying reduce and merge operators to a very large candidate family may even reduce overall performance. In our implementation, the merge size threshold is set to $m = 300$. We will evaluate different choices of m in Section VII-A1.

E. Simulated Annealing

The solution space of the problem in Equation 5 is still too large to be fully explored even after the merge and reduce procedure. To quickly obtain an approximate solution, we use simulated annealing [37].

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

1) Simulated annealing: We aim to simultaneously maximize the minimum of the remaining recovery ratio D for all blocks. Optimization techniques like simulated annealing which implicitly rely on the use of derivatives E are unlikely to work well in this case, since E does not necessarily change if only small changes are made to the index vector $\mathbf{s}$. Hence, we slightly modified the optimization target in Equation 5 to:

$$\min_{\mathbf{s}} -E', \tag{8}$$

where E' is defined as:

$$E' = \min_{i,s} D(O_s^i, T_s^i) + \lambda \sum_{i,s} D(O_s^i, T_s^i). \tag{9}$$

E' is composed of two terms. The first term is just E: the minimum of the remaining recovery ratio D for all blocks. The second term is the sum of the remaining recovery ratio D for all blocks.

We incorporate the second term to improve the derivative properties of the overall optimization target, making it more suitable for simulated annealing. Recall that our target is to maximize E (minimize $-E$) as given in Equation 5. Maximizing the second term (by definition, the sum of all D) shares a similar goal to maximizing E (by definition, the minimum of all D): a small sum cannot, in general, give a large minimum. In practice, we set the weight parameter λ to a small value ($\lambda = 0.1$) to ensure that the second term makes only a small contribution, and E' is still dominated by E.

We start with a random choice for $\mathbf{s}$. On each iteration, a neighborhood is computed by reassigning the indices of the selected mismatched codeword set for several candidate families (i.e. we change the values of s_i for some i, $1 \leq i \leq u$). The temperature is initialized to 1.0 and decays by 0.999 every 20 iterations. The acceptance probability function is set to:

$$\min\left(1, \exp\left(\frac{E'_{\text{new}} - E'_{\text{old}}}{T}\right)\right),$$

where E'_{old} and E'_{new} are the energies before and after transition, respectively; T is the temperature. Iteration stops when there has been no further energy decrease for the last 140,000 iterations (i.e. the temperature has decayed by about 10^{-3}). Note that E' is only used in the acceptance probability function. We record the solutions in all iterations and adopt the solution with the highest actual energy E.

2) Module determination: Having obtained the partial mismatched codeword set for each candidate family, the next step is to determine corresponding module values. For each segment, there may be multiple module value combinations which result in the same partial mismatched codeword set. We prefer the combination with the highest number of transparent modules on the top layer, because more transparent modules will lead to fewer occlusions, producing fewer shadows and and potentially fewer scanning errors.

After determining all module values for the segments, we further change the values of fixed modules on the top layer to transparent if doing so does not introduce additional mismatched codewords.

3) Data masking patterns: The above process is carried out 8 times for the 8 different data masking patterns, and the solution with the highest recovery ratio from all 8 data masking patterns is selected; it is done in parallel using 8 threads.

F. Beautification

Two-layer QR codes also support beautification, e.g. to add a given logo. Beautification is done during preprocessing. Given a reference image, we first beautify the input left and right target patterns using an existing QR code beautification method [20]. We then optimize the top and bottom layers as usual.

However, since we use the same reference image to beautify the left and right target patterns, many modules in the left and right target patterns are identical, resulting in a large number of fixed modules. As a result, beautification reduces the search space and also potentially increases the recovery ratio.

VI. Results and Fabrication

A. Results

We tested our method on a PC with a 3.5 GHz Intel Core i7-5930K CPU and 32 GB RAM. We implemented our algorithm in C++. All two-layer QR codes shown in the paper were generated within a few seconds; the time needed to generate and beautify the input target patterns was less than 5 ms and can be ignored.

1) Diverse examples: To show the robustness of our algorithm in handling diverse examples, we generated 12 examples of two-layer QR codes: see Figures 6 and 7. Their versions and EC levels range from $(2, M)$ to $(17, H)$; 6 of the 12 examples are beautified.

We show 4 examples in Figure 6. Messages are encoded in alphanumeric mode. The top two rows of Figure 6 show two examples encoding the same messages, with different versions and different EC levels. As EC level increases, the results become more reliable with larger effective recovery ratio, but a larger version value is required to encode the message. However, a larger version value requires smaller modules (for the same overall physical size of QR code), which makes the codes harder to scan reliably. There is thus a trade-off between the EC level and version.

The bottom two rows of Figure 6 provide two more beautified examples encoding the same messages, with the same versions but different EC levels. It shows that better reliability can also be achieved for beautified examples by increasing EC level. However, a higher EC level requires more error correction codewords, resulting in fewer modules being available to meet beautification design requirements (if we keep the version value unchanged). Therefore, an appropriate EC level should be chosen to balance aesthetics and reliability.

Further 8 examples are shown in Figure 7. Here, the examples are encoded in byte mode. Our method can robustly handle these diverse examples.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

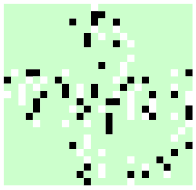
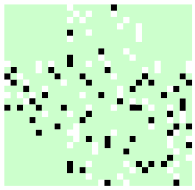
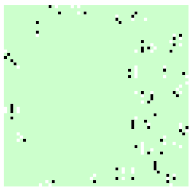
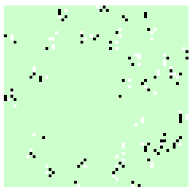
version	EC level	message length	bottom layer	top layer	left view	right view	effective recovery ratio
2	Q	27, 23					$\frac{3}{44}$
3	H	27, 23					$\frac{5}{35}$
10	L	27, 37					$\frac{4}{87}$
10	M	27, 37					$\frac{9}{69}$

Fig. 6. Four examples of two-layer QR codes with different versions and different EC levels. Left to right: version, EC level, string lengths of the two encoded messages, generated bottom and top layers, left and right views, and the effective recovery ratio. Messages are encoded in alphanumeric mode. Messages encoded in the top 2 rows: left view: "A B C D E F G H I J K L M N", right view: "O P Q R S T U V W X Y Z". Messages encoded in the bottom 2 rows: left view: "HTTP://WWW.TSINGHUA.EDU.CN/", right view: "TSINGHUA UNIVERSITY. BEIJING . CHINA".

2) Different EC levels and different data masking patterns: The EC level and data masking pattern (short as '*mask*') of a QR code are stored in the format information. As shown in Figure 1, format information is located next to the finder patterns. It contains 15 bits of data and can tolerate up to 3 bit errors. Format information is identified by a scanner before message decoding, and its correctness is important.

As explained in Section V-A, our generation algorithm restricts the left and right views to have the same EC levels and masks. However, Our algorithm could be slightly modified to support different EC levels, and different masks, by taking format information into account. To do so, care must be taken in dealing with format information, since differences in EC levels and masks can introduce errors in format information.

To achieve this, we slightly modify our algorithm in two ways. Firstly, we add a precomputation stage. For all combinations of different EC levels and different masks ($4^2 \times 8^2 = 1024$ combinations in total), we precompute the optimal values for those top and bottom layer modules which affect format information, using exhaustive search. For each combination, the associated module values which give the fewest error bits are considered as optimal and kept. These precomputed module values are considered as fixed modules and their values are not changed during later optimization. Secondly, during the run-time stage, after EC levels are provided by users, we need to determine which combination to use. We iterate over combinations of the selected EC levels and all masks. For each combination, we run the optimization and obtain an effective recovery ratio. The combination with the highest effective recovery ratio is selected. If multiple combinations result in the same effective recovery ratio, the one with the fewest error bits in format information is selected.

Figure 8 shows an example using different EC levels. In this example, the EC levels of the left and right views are user-specified to be H and M, respectively, to allow different error correction capacities. The resulting effective recovery ratio is $2/39$ (left $2/39$, right $3/49$).

To allow left and right views to have different masks, we have to iterate over 8×8 combinations of masks and select the combination with the highest effective recovery ratio. Using such a *different mask scheme* thus requires approximately $8\times$ as much computation as the original *same mask scheme* described in Section V-E3, where both views use the same mask.

Figure 9 (bottom) shows an example of using different masks, while the same masks are used in Figure 9 (top); other settings and encoded messages are unchanged. In this example, while the result with different masks achieves a better effective

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

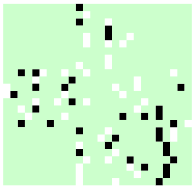
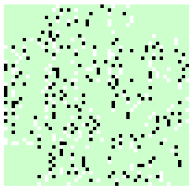
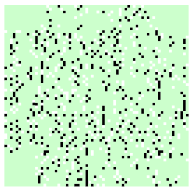
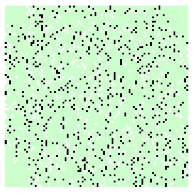
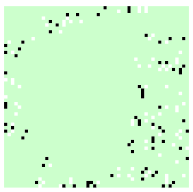
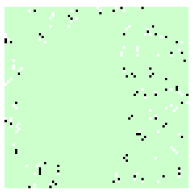
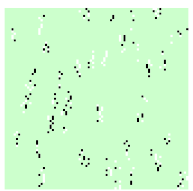
version	EC level	message length	bottom layer	top layer	left view	right view	effective recovery ratio
2	M	17, 21					$\frac{1}{44}$
8	Q	95, 83					$\frac{3}{40}$
12	H	147, 133					$\frac{5}{42}$
17	H	256, 256					$\frac{4}{43}$
9	L	44, 20					$\frac{2}{146}$
13	L	48, 44					$\frac{6}{133}$
13	M	6, 8					$\frac{9}{60}$
16	Q	35, 14					$\frac{9}{43}$

Fig. 7. Further examples of two-layer QR codes. Left to right: version, EC level, string lengths of the two encoded messages, generated bottom and top layers, left and right views, and the effective recovery ratio. Messages are encoded in byte mode.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

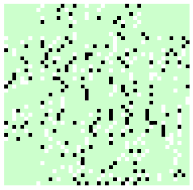
version	EC level	message length	bottom layer	top layer	left view	right view	effective recovery ratio
7	H, M	39, 122					$\frac{2}{39}, \frac{3}{49}$ overall: $\frac{2}{39}$

Fig. 8. Example whose views have different EC levels. EC level: H (left view), M (right view). Messages are encoded in byte mode. Left: "https://www.iso.org/standard/62021.html", right: "Information technology -- Automatic identification and data capture techniques -- QR Code bar code symbology specification". Time taken: 0.635 s.

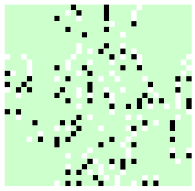
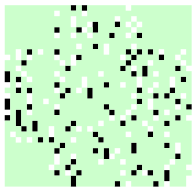
type	ver.	EC level	msg. len.	bottom layer	top layer	left view	right view	effective recovery ratio
same mask	4	H	50, 50					$\frac{2}{25}$
different mask	4	H	50, 50					$\frac{3}{25}$

Fig. 9. Examples whose views have (top row) the same data masks (pattern 010), and (bottom row) different masks (left: 100, right: 011). Both encode the same messages, randomly generated strings of length 50, in alphanumeric mode. Time taken: 0.412 s (same masks), 2.726 s (different masks).

recovery ratio (3/25) than when using the same mask (2/25), it additionally introduces 1 bit error in format information.

We tested the different mask scheme described above for all 14 examples in Figures 6–9. Perhaps surprisingly, except for one example in Figure 9, the different mask scheme performed no better than the same mask scheme, i.e. the optimal combination turned out to be a pair of identical masks.

To assess scanning robustness, we conducted a synthetic scanning comparison between the different mask scheme and the same mask scheme. See Figure 18. The success rates of these two schemes were nearly the same.

We believe the reasons are twofold. Firstly, the padding codewords are probably the same when we apply the same mask scheme, while different masks may cause potential conflicts in padding codewords. Secondly, different masks may make scanning less robust because the use of different masks may cause errors in the format information.

Overall, since the different mask scheme has limited benefits but greatly increases the computation required, we suggest using the same masks for both left and right QR codes. Unless explicitly stated otherwise, all results in the paper thus use the same mask scheme.

B. Fabrication

1) Manufacture: We now consider the fabrication of a real two-layer QR code. We print the top layer pattern on a transparent plastic sheet and the bottom layer pattern on paper. We then paste these two patterns onto the two sides of a transparent acrylic plate with a thickness of $h = 3\,\mathrm{mm}$. We set the width of the top layer modules to $w_t = 1.5\,\mathrm{mm}$.

An example of a fabricated two layer QR code is shown in Figure 10.

2) Module widths: Intuitively, it is clear that the bottom layer modules should be a little larger than the top layer modules to achieve the best alignment, due to the foreshortening effect of perspective projection. We next consider how to set the size of the bottom layer modules in terms of camera distance and position.

Figure 11 illustrates a typical use case and the camera position. Suppose the field of view angle of the camera is 30°, and it has an aspect ratio of 1:1. Let the distance from the camera to the center of the top layer pattern be d, the width of the top layer pattern be w, and the polar and azimuthal angles of the camera direction be θ and ϕ, respectively. We denote the refractive index of the acrylic plate by n, which has a typical value of $n = 1.5$. Since the QR code pattern usually covers the majority of the captured image when scanning, the camera distance d is likely to satisfy $2.4w \leq d \leq 3.6w$, and so we may assume $d = 3w$. (In computing module widths, without loss of generality, we also assume the camera is to the right of the QR plate, i.e. $\theta > 0$)

The best camera direction corresponds to a viewing direc-

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

Fig. 10. A fabricated two-layer QR code.

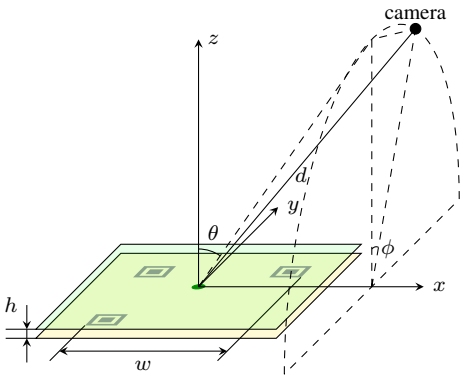


Fig. 11. Camera position.

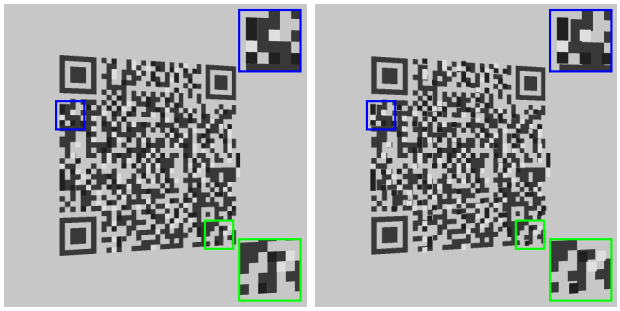


Fig. 12. Synthetic images of two-layer QR codes with corrected (left) and uncorrected bottom module widths (right). Both are taken from the optimal camera directions. Left: the bottom module width is computed using Equation 11. Right: the bottom module width is set equal to the top module width. The modules in the left image are better aligned.

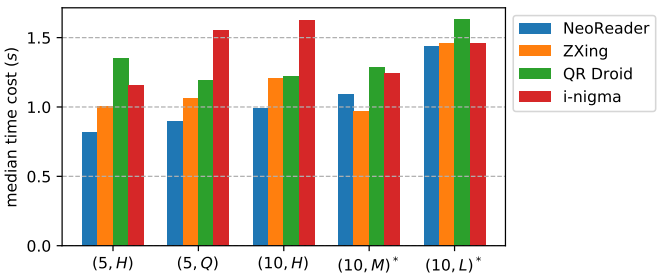


Fig. 13. Median scanning times using each reader app for 5 fabricated examples with varying parameters of version and EC level. '*' refers to a beautified two-layer QR code.

tion in which the central point of a top layer module just occludes the central point of the corresponding bottom layer module. Given the widths of the bottom layer modules along x- and y-axes, respectively, denoted w_b^x, w_b^y, the best camera direction is (see the supplementary document for detailed derivations of formulae in this section):

$$\theta = \arcsin\left(\frac{n(w_b^x/2)}{\sqrt{h^2 + (w_b^x/2)^2}}\right), \quad \phi = 0°, \tag{10}$$

where n and h are the refractive index and thickness of the transparent acrylic plate, respectively. Given the camera direction, for best alignment between the top and bottom layers, the widths of bottom layer modules (w_b^x and w_b^y) are given by:

$$\begin{aligned} w_b^x &= w_t\left(1 + \frac{\left(h^2 + (w_b^x/2)^2\right)^{3/2}\cos^2\theta}{nd\sqrt{h}}\right), \\ w_b^y &= w_t\left(1 + \frac{w_b^x}{2d\sin\theta}\right). \end{aligned} \tag{11}$$

While Equations 10 and 11 show how to compute the optimal θ, w_b^x and w_b^y, they are nonlinear. We initialize $w_b^x = w_t = 1.5\,\mathrm{mm}$ and compute these values iteratively. The resulting optimal values are: $\theta \approx \pm 21.57°$ for version 5 QR code, and $\theta \approx \pm 21.49°$ for version 10 QR code.

Figure 12 shows two synthetic images of two-layer QR codes. The left image takes perspective projection into account and sets the bottom module width according to Equation 11 (referred to as *corrected bottom module width*), while the right image ignores perspective projection and simply sets bottom module width equal to top module width (referred to as *uncorrected bottom module width*). The close-ups in Figure 12 help to verify that the modules in the left image are better aligned.

3) Scanning time: We printed and fabricated five different two-layer QR codes. We then invited 5 users to scan the fabricated two-layer QR codes with 4 widely used mobile phone QR code reader applications (NeoReader, ZXing, QR Droid, and i-nigma). The mobile phone used was a *MEIZU PRO 6s*, running *Android 7.1.1 (Flyme 6.3.0.2A)*. Each user was asked to scan each code twice: the user was asked to align the camera with the center of the two-layer QR code first, and then turn left (first scan) or right (second scan) to scan the two-layer QR code. The scanning process was recorded by a screen capture plugin, and the scanning interaction time for each view was also recorded. In the experiment, the two-layer QR codes were scanned successfully and decoded correctly in every case. Figure 13 gives, for each example, for each reader app, the median scanning times for the 5 users. The results demonstrate that our fabricated two-layer QR codes can easily be scanned in practice.

The supplementary video illustrates the fabricated examples and the scanning process.

4) Instructions for users: Given familiarity with standard QR codes, users might at first try to scan two-layer QR codes from the front. Therefore, we recommend placing clear guidance besides such new QR codes to explain how to use

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

them. For example, a two-layer QR code describing an exhibit in a museum could be accompanied by text labels and arrows saying 'English (scan this way)' pointing from the left, and 'Chinese (scan this way)' pointing from the right.

VII. Evaluation

A. Parameter Evaluation

In this section, we evaluate how features and parameters in our algorithm affect correction capability and run time. The features include the merge and reduce operators, and the parameters are the merge size threshold m and the weight λ in simulated annealing.

For each experiment, for each example in Figures 6 and 7, we ran our algorithm 100 times. The run time and effective recovery ratio were recorded. In each case, if the effective recovery ratio was not less than the *desired effective recovery ratio* (i.e., the median effective recovery ratios over all experiments for an example), we considered the output to be *acceptable*. Finally, we give the *acceptability rate* and average run time for each example for a given experimental setup. In some cases, no additional merging occured on increasing the threshold m, so the acceptability rate and run time remained unchanged. We mark these cases as 'same'.

Next, we explain each experiment in detail.

1) Merge and reduce operators: Table II (top 7 rows) show results concerning the merge and reduce process, including: whether or not merge operators and reduce operators were used, and the value of the merge size threshold m.

The results indicate that use of reduce and merge operators significantly increases the acceptability rate and decreases run time. Without reduce and merge operators, it is hard to generate acceptable results for complex examples (see Table II (rows 1 and 2)). For the merge size threshold, $m = 300$ is a reasonable compromise, since it achieves a nearly optimal acceptability rate, and larger m does little but increase run time. In fact, the results show that varying m has little effect on the algorithm. Varying m between 30 and 3000 always results in an acceptability rate over 99%, while run time changes by under 10%. There is little to be gained by adjusting m for different cases.

2) Simulated annealing process: Table II (bottom 4 rows) shows the effect of changing the simulated annealing weight parameter λ in the modified optimization target (Equation 9).

The results show that using the modified optimization target ($\lambda > 0$) leads to much larger acceptability rates than the original optimization target ($\lambda = 0$) for complex examples. However, for $\lambda > 0$, we observe that increasing λ improves computational efficiency but reduces the acceptability rate. We suggest setting $\lambda = 0.1$ provides a good trade-off.

B. Robustness for Arbitrary Input Messages

We conducted an experiment to evaluate the robustness of our algorithm for arbitrary input messages. We tested different settings of version (1, 5, 10, 15, and 20) and EC level (L, M, Q, and H).

For each parameter setting, we first generated 1,000 pairs of random messages in alphanumeric mode with maximal encodable message length for this setting. We then ran our algorithm. The generation was considered to be *successful* if the effective recovery ratio E was non-negative when simulated annealing terminated.

Table III shows the success rate of our algorithm and the corresponding message length for each parameter setting. It demonstrates that our algorithm is robust for arbitrary input messages if version and EC level are chosen appropriately. Our algorithm is always successful for EC levels Q and H. In contrast, the success rate is low for EC levels L and M, which provide smaller error correction capability. The table suggests that it is more robust to use EC levels Q or H.

We provide some sample two-layer QR code examples which encode randomly generated messages in the supplementary document.

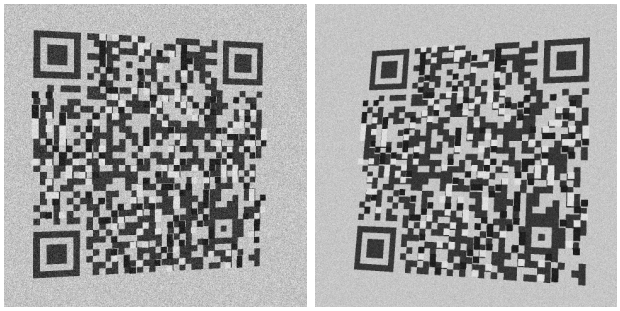


Fig. 14. Two synthetic images of the same two-layer QR code, rendered from different camera directions with different noise levels. Left: $\theta = -15°$, $\phi = 2°$, $\sigma = 32$, $d = 2.4w$, right: $\theta = 18°$, $\phi = -10°$, $\sigma = 16$, $d = 2.4w$.

C. Scanning Robustness

We implemented a simple ray tracer to render synthetic two-layer QR codes. Gaussian noise with standard deviation σ was added to each pixel on the top layer, bottom layer, and the transparent plate, respectively. We assume the plate has a transmittance of 0.75. Figure 14 shows two synthetic images of the same code rendered from different camera directions with different noise levels.

To test scanning robustness, we conducted experiments with two settings of version and EC level: (5,H) and (10,H). For each setting, we generated one thousand two-layer QR codes encoding randomly generated messages with appropriate length. We used the ray tracer to render synthetic images of these two-layer QR codes from different camera directions with different noise levels. The synthetic images were rendered with a resolution of 960×960 pixels, and were resized to 5 different resolutions (160, 240, 320, 400 and 480 pixels) using bilinear interpolation. Scanning was considered *successful* if one of the 5 resized images could be correctly decoded by the open-source QR code reader ZXing [2].

1) Camera direction: Figure 15 shows how the success rate changes with camera direction. Rows 1 and 3 show how success rate η varies with polar angle θ, while rows 2 and 4 show how success rate η varies with azimuthal angle ϕ. When computing success rate η for a polar angle θ, we

[2]ZXing: https://code.google.com/p/zxing

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

TABLE II
EVALUATION OF MERGE AND REDUCE OPERATORS (TOP 7 ROWS), AND EVALUATION OF MODIFIED OPTIMIZATION TARGET (BOTTOM 4 ROWS). WE GIVE ACCEPTABILITY RATE (ABOVE) AND AVERAGE RUN TIME (BELOW, IN SECONDS), FOR EACH EXAMPLE, FOR EACH SETTING. '*' REFERS TO A BEAUTIFIED TWO-LAYER QR CODE.

	Fig. 6 row 1 (2,Q)	Fig. 6 row 2 (3,H)	Fig. 6 row 3 (10,L)*	Fig. 6 row 4 (10,M)*	Fig. 7 row 1 (2,M)	Fig. 7 row 2 (8,Q)	Fig. 7 row 3 (12,H)	Fig. 7 row 4 (17,H)	Fig. 7 row 5 (9,L)*	Fig. 7 row 6 (13,L)*	Fig. 7 row 7 (13,M)*	Fig. 7 row 8 (16,Q)*
no reduce, no merge $\lambda = 0.1$	74% 0.770	81% 0.948	0% -	0% -	77% 0.638	0% -	0% -	0% -	29% 2.089	0% -	0% -	0% -
reduce, no merge $\lambda = 0.1$	100% 0.493	100% 0.399	22% 0.384	83% 0.381	100% 0.313	89% 0.843	100% 1.516	60% 2.768	100% 0.855	100% 0.398	100% 0.325	31% 0.487
reduce, merge $m = 30$, $\lambda = 0.1$	100% 0.463	100% 0.354	99% 0.340	100% 0.347	100% 0.265	99% 0.640	100% 1.091	100% 2.024	100% 0.791	100% 0.363	100% 0.293	100% 0.449
reduce, merge $m = 100$, $\lambda = 0.1$	100% 0.444	100% 0.338	99% 0.332	same	100% 0.241	100% 0.617	100% 1.023	100% 1.857	100% 0.793	100% 0.362	same	100% 0.436
reduce, merge **$m = 300$, $\lambda = 0.1$**	**100%** **0.422**	**100%** **0.332**	**same**	**same**	**100%** **0.243**	**99%** **0.619**	**100%** **1.018**	**99%** **1.827**	**100%** **0.782**	**same**	**same**	**same**
reduce, merge $m = 1000$, $\lambda = 0.1$	100% 0.431	100% 0.335	same	same	100% 0.234	99% 0.622	100% 1.028	100% 1.858	100% 0.790	same	same	same
reduce, merge $m = 3000$, $\lambda = 0.1$	100% 0.455	100% 0.343	same	same	same	100% 0.663	100% 1.099	100% 2.059	same	same	same	same
reduce, merge $m = 300$, $\lambda = 0$	100% 0.426	100% 0.329	99% 0.351	100% 0.358	100% 0.244	1% 0.599	0% -	0% -	100% 0.787	100% 0.368	100% 0.286	100% 0.437
reduce, merge $m = 300$, $\lambda = 0.03$	100% 0.430	100% 0.340	100% 0.343	100% 0.353	100% 0.242	100% 0.652	100% 1.103	100% 1.935	100% 0.792	100% 0.367	100% 0.283	100% 0.426
reduce, merge $m = 300$, $\lambda = 0.20$	100% 0.425	100% 0.325	100% 0.333	100% 0.342	100% 0.244	98% 0.604	100% 0.978	100% 1.792	100% 0.786	100% 0.365	100% 0.288	100% 0.447
reduce, merge $m = 300$, $\lambda = 0.50$	100% 0.418	100% 0.319	100% 0.336	100% 0.371	100% 0.234	96% 0.580	100% 0.937	100% 1.714	100% 0.782	100% 0.364	100% 0.292	100% 0.467

TABLE III
ROBUSTNESS FOR ARBITRARY INPUT MESSAGES: SUCCESS RATE OF OUR ALGORITHM, AND CORRESPONDING MAXIMAL ALPHANUMERIC MODE MESSAGE LENGTH FOR DIFFERENT SETTINGS OF VERSION AND EC LEVEL.

EC Level	H		Q		M		L	
Version	suc. rate	len	suc. rate	len	suc. rate	len	suc. rate	len
1	100%	10	100%	16	95.3%	20	4.6%	25
5	100%	64	100%	87	18.4%	122	0.0%	154
10	100%	174	100%	221	0.4%	311	0.0%	395
15	100%	321	100%	426	0.0%	600	0.0%	758
20	100%	557	100%	702	0.0%	970	0.0%	1249

randomly selected azimuthal angle $\phi \in [-2°, 2°]$ accordingly to a uniform distribution. Similarly, when computing η for an azimuthal angle ϕ, the polar angle θ was randomly sampled from $\pm[20°, 22°]$. In both cases, d was randomly sampled from $[2.4w, 3.6w]$. The results show that scanning from directions with polar angle $\theta \in \pm[10°, 30°]$ and azimuthal angle $[-15°, 15°]$ is relatively successful.

2) Camera distance: Figure 16 shows how the success rate changes with camera distance for different noise levels. When computing success rate η for a camera distance d, we randomly selected polar angle $\theta \in \pm[20°, 22°]$ and azimuthal angle $\phi \in [-2°, 2°]$. The results indicate that restricting camera distance $d \in [2.4w, 3.6w]$ is a good choice for scanning in the presence of noise (see Figure 16, right two columns). As d further increases, the success rate drops as the QR code becomes smaller and harder to recognize.

3) Bottom module width: In Figure 17, we plot success rate η against polar angle θ and azimuthal angle ϕ, using the corrected bottom module width (orange curve) and uncorrected bottom module width (green curve), respectively. The results show that using corrected bottom module width (Equation 11) increases success rate and improves scanning robustness.

In all plots of Figures 15–17, we also provide scanning success rates for standard QR codes (blue curves). Compared to standard QR codes, two-layer QR codes have a lower success rate due to damage introduced during the generation process. However, they still retain a high success rate if camera distance and direction are appropriately set.

4) Same versus different masks: While the different mask scheme described in Section VI-A2 can slightly increase the effective recovery ratio, it can also introduce errors in format information. To demonstrate how it affects scanning robustness overall, in Figure 18, we plot success rate η against polar angle θ using the different mask scheme (orange curve) and the same mask scheme (blue curve), respectively. Their scanning success rates are almost identical. Since the different mask scheme brings few benefits but requires 8 times the computation, we prefer the same mask scheme.

VIII. CONCLUSION

This paper has presented two-layer QR codes which can encode two independent messages. They can be decoded separately by changing scanning directions. We have given an automatic algorithm for generating such codes, and demonstrated the robustness of the proposed approach. Beautification can be used with these codes, as for standard QR codes. In future, we hope to extend this method to allow more than two messages to be encoded in such a specially designed two-layer structure, and to incorporate more types of code beautification techniques.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

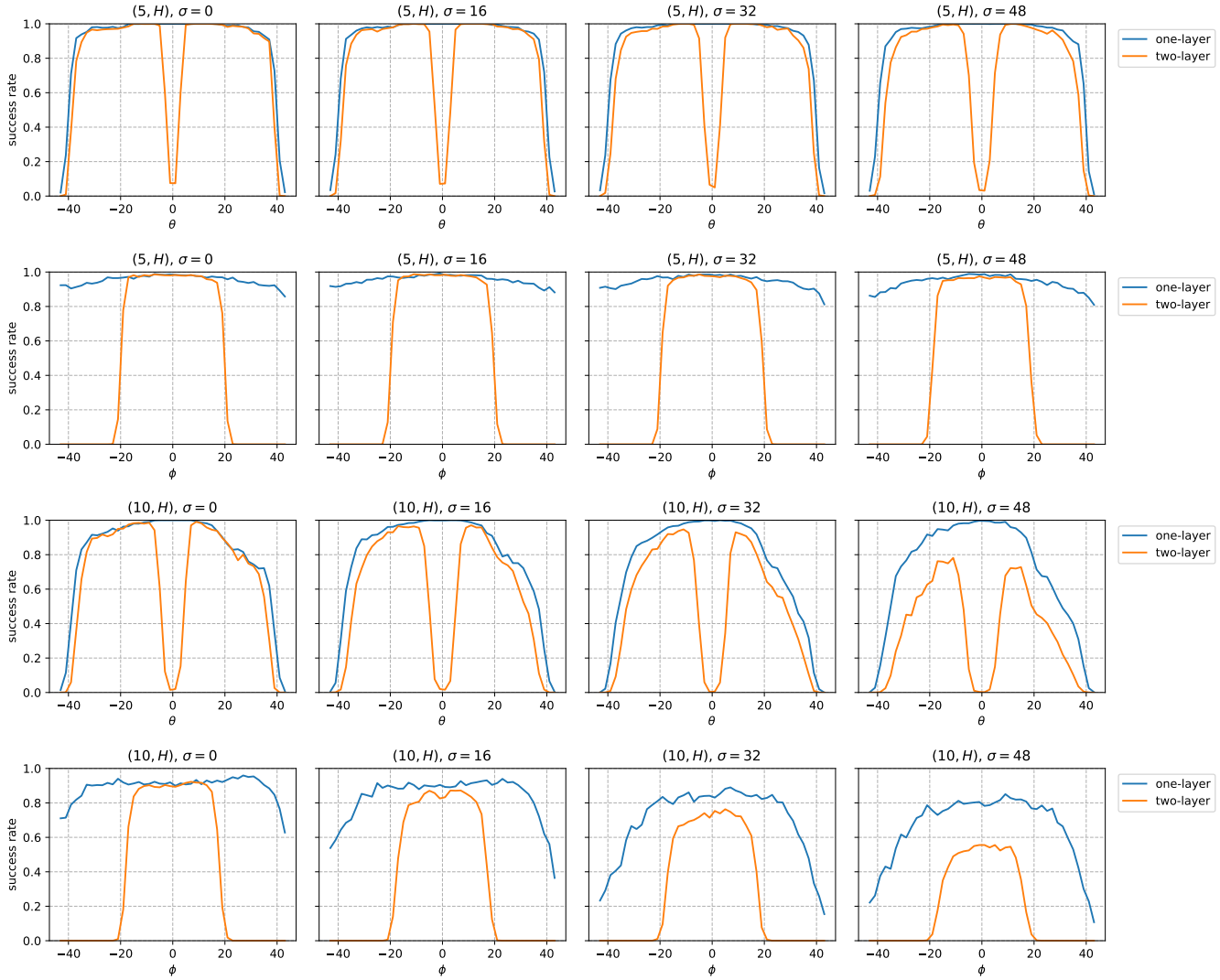


Fig. 15. Scanning robustness with respect to camera direction. Orange curves: success rate (η) versus camera direction for two-layer QR codes. Blue curves: success rate for standard QR codes for comparison.

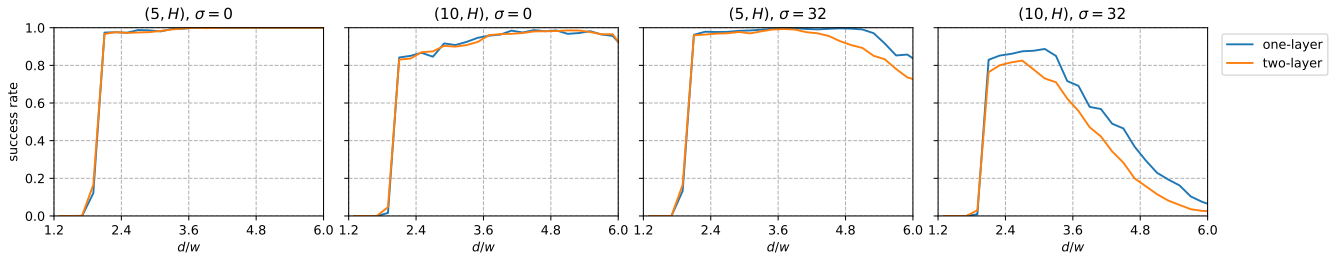


Fig. 16. Scanning robustness with respect to camera distance. Orange curves: success rate (η) versus camera distance for two-layer QR codes. Blue curves: success rate for standard QR codes for comparison.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Project Number 61822204, 61521002), a research grant from the Beijing Higher Institution Engineering Research Center, and the Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

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

[2] "hotapp." [Online]. Available: https://www.hotapp.cn/

[3] "Lenticular printing." [Online]. Available: https://en.wikipedia.org/wiki/Lenticular_printing

[4] A. Oliva, A. Torralba, and P. G. Schyns, "Hybrid images," *ACM*

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

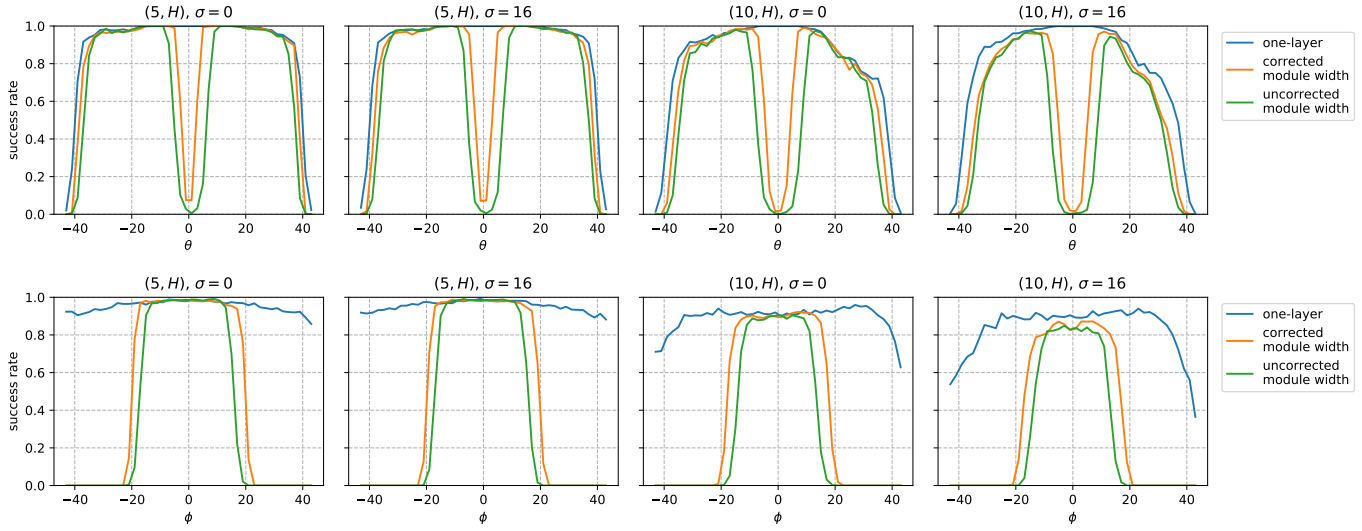


Fig. 17. Scanning robustness with respect to bottom module width. Orange curves: success rate for two-layer QR codes with corrected bottom module width computed by Equation 11. Green curves: success rate for two-layer QR codes with uncorrected bottom module width (equal to top module width). Blue curves: success rate for standard QR codes.

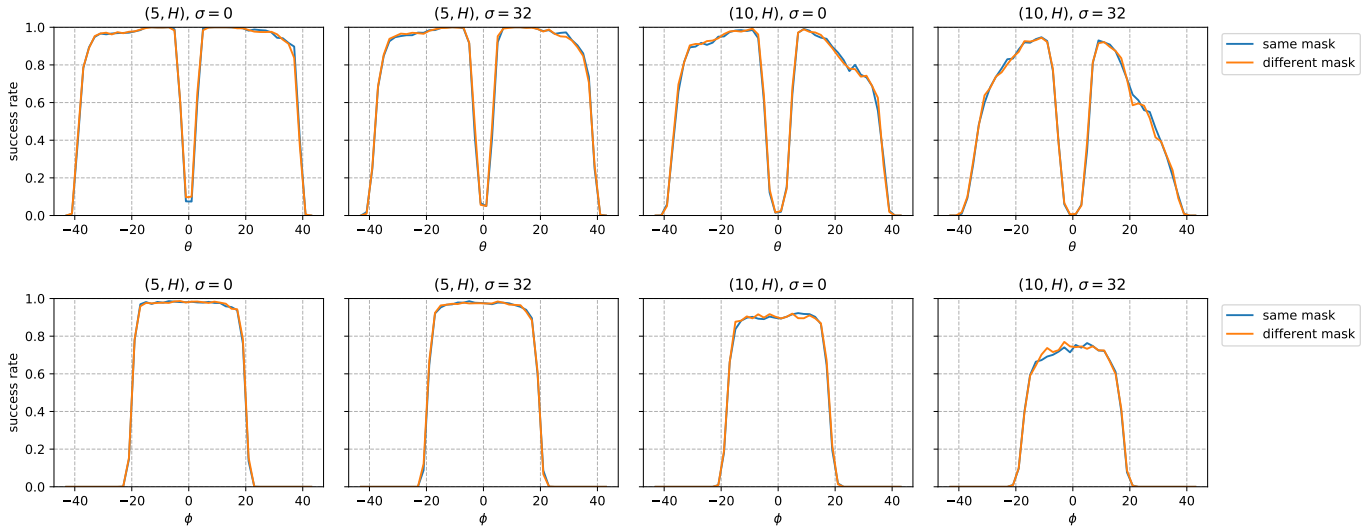


Fig. 18. Scanning robustness with respect to mask schemes. Blue curves: success rate (η) for two-layer QR codes generated using the same mask scheme. Orange curves: success rate (η) for two-layer QR codes generated using the different mask scheme. Their scanning success rates are almost identical.

Transactions on Graphics, vol. 25, no. 3, pp. 527–532, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1141911.1141919

[5] N. J. Mitra and M. Pauly, "Shadow art," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 156:1–156:7, Dec. 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618502

[6] S. Ono, K. Morinaga, and S. Nakayama, "Two-dimensional barcode decoration based on real-coded genetic algorithm," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1068–1073.

[7] ——, "Animated two-dimensional barcode generation using optimization algorithms–redesign of formulation, operator, and quality evaluation," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 13, no. 3, pp. 245–254, 2009.

[8] S. Ono and S. Nakayama, "Fusion of interactive and non-interactive evolutionary computation for two-dimensional barcode decoration," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[9] M. Kamizono, K. Shimomura, M. Tajiri, and S. Ono, "Two-dimensional barcode decoration using module-wise non-systematic coding and cooperative evolution by user and system," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 2016, pp. 245–252.

[10] D. Samretwit and T. Wakahara, "Measurement of reading characteristics of multiplexed image in QR code," in *2011 Third International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 2011, pp. 552–557.

[11] Z. Baharav and R. Kakarala, "Visually significant QR codes: Image blending and statistical analysis," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.

[12] H.-K. Chu, C.-S. Chang, R.-R. Lee, and N. J. Mitra, "Halftone QR codes," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 217, 2013.

[13] Y.-S. Lin, S.-J. Luo, and B.-Y. Chen, "Artistic QR code embellishment," in *Computer Graphics Forum*, vol. 32, no. 7. Wiley Online Library, 2013, pp. 137–146.

[14] Z. Gao, G. Zhai, and C. Hu, "The invisible QR code," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1047–1050.

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

[16] C. Fang, C. Zhang, and E.-C. Chang, "An optimization model for aesthetic two-dimensional barcodes," in *International Conference on Multimedia Modeling*. Springer, 2014, pp. 278–290.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TIP.2019.2908490

[17] S. Wang, T. Yang, J. Li, B. Yao, and Y. Zhang, "Does a QR code must be black and white?" in *2015 International Conference on Orange Technologies (ICOT)*. IEEE, 2015, pp. 161–164.
[18] A. Mittal, "Generating visually appealing QR codes using colour image embedding," *The Imaging Science Journal*, vol. 65, no. 1, pp. 1–13, 2017.
[19] L. Lin, S. Wu, S. Liu, and B. Jiang, "Interactive QR code beautification with full background image embedding," in *Second International Workshop on Pattern Recognition*, vol. 10443. International Society for Optics and Photonics, 2017, p. 1044317.
[20] K. Fujita, "Expansion of image displayable area in design QR code and its applications," in *Forum on Information Technology 2011 (FIT2011)*, 2011.
[21] S.-S. Lin, M.-C. Hu, C.-H. Lee, and T.-Y. Lee, "Efficient QR code beautification with high quality visual content," *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1515–1524, 2015.
[22] Y. Zhang, S. Deng, Z. Liu, and Y. Wang, "Aesthetic QR codes based on two-stage image blending," in *International Conference on Multimedia Modeling*. Springer, 2015, pp. 183–194.
[23] M. Kuribayashi and M. Morii, "Enrichment of visual appearance of aesthetic QR code," in *International Workshop on Digital Watermarking*. Springer, 2015, pp. 220–231.
[24] S. Qiao, X. Fang, B. Sheng, W. Wu, and E. Wu, "Structure-aware QR code abstraction," *The Visual Computer*, vol. 31, no. 6-8, pp. 1123–1133, 2015.
[25] L. Li, J. Qiu, J. Lu, and C.-C. Chang, "An aesthetic QR code solution based on error correction mechanism," *Journal of Systems and Software*, vol. 116, pp. 85–94, 2016.
[26] M. Kuribayashi, E.-C. Chang, and N. Funabiki, "Watermarking with fixed decoder for aesthetic 2D barcode," in *International Workshop on Digital Watermarking*. Springer, 2016, pp. 379–392.
[27] M. Kuribayashi and M. Morii, "Aesthetic QR code based on modified systematic encoding function," *IEICE Transactions on Information and Systems*, vol. 100, no. 1, pp. 42–51, 2017.
[28] J.-C. Liu and H.-A. Shieh, "Toward a two-dimensional barcode with visual information using perceptual shaping watermarking in mobile applications," *Optical Engineering*, vol. 50, no. 1, p. 017002, 2011.
[29] C. Chen, W. Huang, B. Zhou, C. Liu, and W. H. Mow, "PiCode: A new picture-embedding 2D barcode," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3444–3458, 2016.
[30] C. Chen, B. Zhou, and W. H. Mow, "RA code: a robust and aesthetic code for resolution-constrained applications," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.
[31] M. E. V. Melgar, A. Zaghetto, B. Macchiavello, and A. C. Nascimento, "CQR codes: Colored quick-response codes," in *2012 IEEE Second International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2012, pp. 321–325.
[32] H. Blasinski, O. Bulan, and G. Sharma, "Per-colorant-channel color barcodes for mobile applications: An interference cancellation framework," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1498–1511, 2013.
[33] H. J. Galiyawala and K. H. Pandya, "To increase data capacity of QR code using multiplexing with color coding: An example of embedding speech signal in QR code," in *India Conference (INDICON), 2014 Annual IEEE*. IEEE, 2014, pp. 1–6.
[34] Z. Yang, Y. Bao, C. Luo, X. Zhao, S. Zhu, C. Peng, Y. Liu, and X. Wang, "ARTcode: preserve art and code in any image," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 904–915.
[35] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
[36] "Information technology – automatic identification and data capture techniques – QR code bar code symbology specification," https://www.iso.org/standard/62021.html, pp. 1–117, 2015.
[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

Tailing Yuan is a PhD student in the Department of Computer Science and Technology, Tsinghua University. He received his bachelor's degree from the same university in 2016. His research interests include computer graphics and computer vision.

Yili Wang is a senior undergraduate student in the Department of Foreign Languages and Literatures, Tsinghua University. She plans to pursue a PhD degree in computer science in future. Her research interests include computer graphics and image editing.

Kun Xu is an associate professor in the Department of Computer Science and Technology, Tsinghua University. Before that, he received his bachelor and doctor degrees from the same university in 2005 and 2009, respectively. His research interests include realistic rendering and image/video editing.

Ralph R. Martin recently retired after a long career in visual computing, and is now an emeritus professor of Cardiff University. He is currently enjoying life in the Welsh countryside.

Shi-Min Hu received the PhD degree from Zhejiang University, in 1996. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conference. He is editor-in-chief of the *Computational Visual Media*, and on editorial board of several journals, including the *IEEE Transactions on Visualization and Computer Graphics* and the *Computer Aided Design and Computer & Graphics*.

Copyright (c) 2019 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.