Machine Vision and Applications (1998) 11: 74–82

Machine Vision and Applications

A system for a new two-dimensional code: Secure 2D code

Chung-Tsai Yeh, Ling-Hwei Chen

Institute of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 30050, Republic of China

Received: 9 September 1997 / Accepted: 2 March 1998

Abstract. A new 2D code called Secure 2D code is designed in this paper, both encoder and decoder are also proposed. Secure 2D code can store any kind of data and provides high security. With regard to security, the input data is divided into two parts: general and secret. The general data is transformed into a 2D code pattern, then secret data is hidden in the 2D code pattern. To raise the reading speed and allow various reading environments, some features are added around the 2D code pattern boundary. As to the reliability, RS code is adopted to treat damaged patterns.

Key words: 2D code – RS code – Data hiding – Security – Error correction

1 Introduction

The conventional bar code [1] used in supermarkets only consists of an identifying number, we need to access a database if detailed information is required. To remedy this disadvantage, 2D codes [2] with much higher density have recently been introduced; they can store the desired information. 2D codes also bring higher reliability and security, because their greater capacity allows adding extra data for error correction and helps to achieve data security aim. Besides, using 2D codes to store data is a new technique for automatic data collection (ADC).

2D codes [2–11] can be grouped into two major categories: stacked [2, 3] and matrix [4–11]. Stacked 2D codes are stacked by several rows of 1D bar codes; matrix 2D codes have their own methods to encode data.

For stacked 2D codes, Allais designed Code 49 [2] (see Fig. 1a), which is based on a (16, 4) code [1]. Williams [2] proposed Code 16K (see Fig. 1b), which is based on two kinds of 1D bar codes: Universal Product Code (UPC) [1] and Code 128 [1]. Codablock MLC-2D [2] (see Fig. 1c) is basically a stack of Code 39 [1]. PDF417 (Portable Data File) [3] shown in Fig. 1d is based on a (17, 4, 6) code [1].

For matrix 2D codes, Vericode [4] shown in Fig. 2a is a checkerboard code and highly encrypted. Datacode [5, 6]

Correspondence to: L.-H. Chen

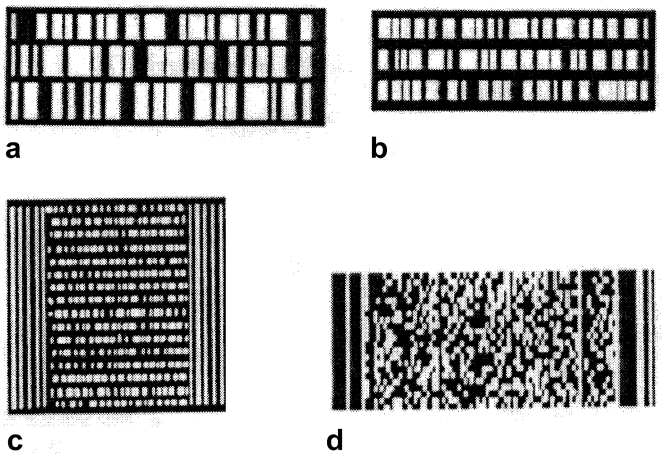


Fig. 1a–d. Some stacked 2D codes. **a** Code 49. **b** Code 16K. **c** Codablock MLC-2D. **d** PDF417

shown in Fig. 2b contains a special frame enclosing the code. The frame can be used to determine the pattern orientation and the module size. Phillips Dot Code [8] provides four corner dots to determine the orientation of a pattern (see Fig. 2c), its major applications are silicon wafer labeling and drug dosage labeling. Maxicode [9] is used in package sending, it provides a number of features. A central finder mark in the center of each pattern as shown in Fig. 2d is given such that the pattern in a package can be located easily; the nested hexagonal code words permit reading on warped or angled surfaces. Maxicode employs Reed-Solomon (RS) code [12] for error correction. Minicode [10] shown in Fig. 2e combines an easy-to-read "license plate" with a high-density "data file" on a small pattern, it uses BCH code [12] for error detection and correction. Triangle Code [11] is encoded using triangle elements as shown in Fig. 2f, it provides higher density of information than codes where each cell is a square, such as Vericode or Datacode.

All of the above-mentioned 2D codes do not emphasize security; here, a new 2D code is proposed, it emphasizes security and error correction, thus it is named Secure 2D code. It can be read quickly and store any kind of data, and hence can be applied in any environment. As regards security, data will be divided into general data and secret

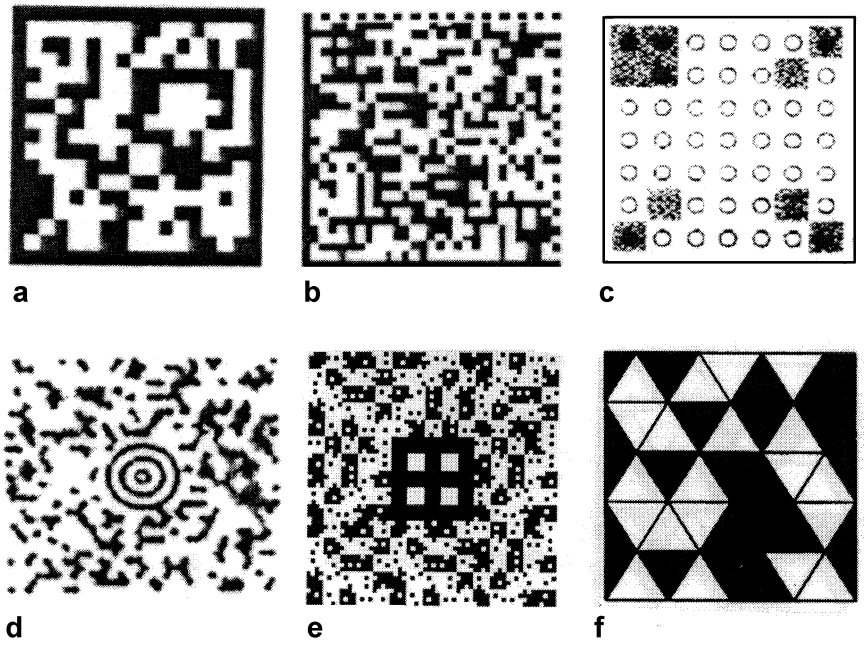


Fig. 2a–f. Six matrix 2D codes. **a** Vericode. **b** Datacode. **c** Phillips Dot Code. **d** Maxicode. **e** Minicode. **f** Triangle code

data, and a data-hiding method will be proposed to embed secret data into Secure 2D code. In the part concerning error correction, the RS code will be used. Furthermore, for some problems which often occur when 2D codes are read, a solution method will be provided.

2 The proposed Secure 2D code system

The block diagram of the proposed Secure 2D code system is shown in Fig. 3. The system consists of two parts: the encoder and the decoder; the objective of the encoder is to convert a bit-stream into a Secure 2D code pattern. First, the general data is considered as a bit-stream, then some extra error correction codes are added behind the bit-stream. And the secret data is embedded in the bit-stream. Finally, the pattern of Secure 2D code is generated. On the other hand, the objective of the decoder is to convert the pattern of Secure 2D code into a bit-stream. First, the Secure 2D code is read as a bit-stream. Then, the secret data is extracted from the bit-stream. Finally, error correction is conducted and the extra error correction codes are removed, the remaining bit-streams are output.

The proposed system consists of three major topics: error correction, security, and pattern generation/reading. A brief description of these topics will be given in the following subsections.

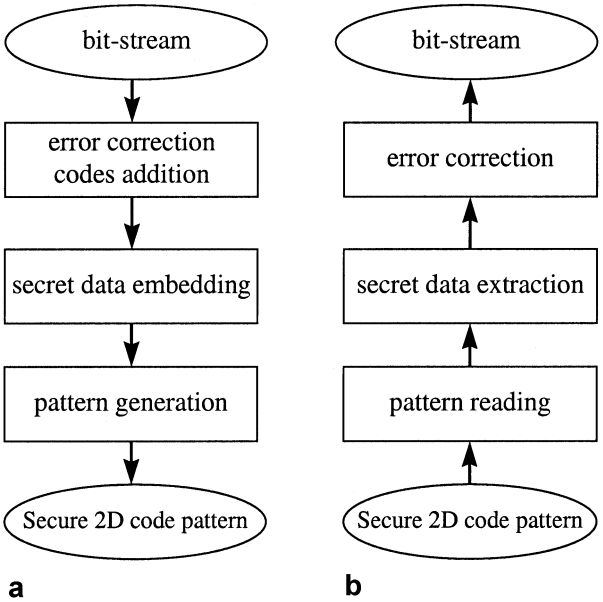


Fig. 3a,b. The block diagram of the proposed Secure 2D code system. **a** Secure 2D code encoder. **b** Secure 2D code decoder

2.1 Error correction

With 2D codes, many kinds of errors may occur in the patterns, e.g., spots, scratch, etc. It is quite obvious that 2D codes need a powerful error correction ability to ensure reading reliability. RS [12, 13] code plays an important role and is used in our system. The integral name of RS code is t-error-correcting (n, k) RS code, where k is the length of source codes, n is the length of total codes including source codes and redundancy codes for error correction, and t is the allowed maximum number of errors. RS codes are constructed using a Galois field (GF) [12]. An RS code has code words from GF(q) with $n = q - 1$, and each code word has length m with $2m = q$.

Note that there are two kinds of errors: located (erasure) errors and unlocated errors which may appear in the received messages. Let s be the number of located errors and t be the number of unlocated errors. For an (n, k) RS code, s and t will satisfy the following equation [13]: $s + 2t = n - k$. In our 2D code system, we do not consider the type of erasure errors.

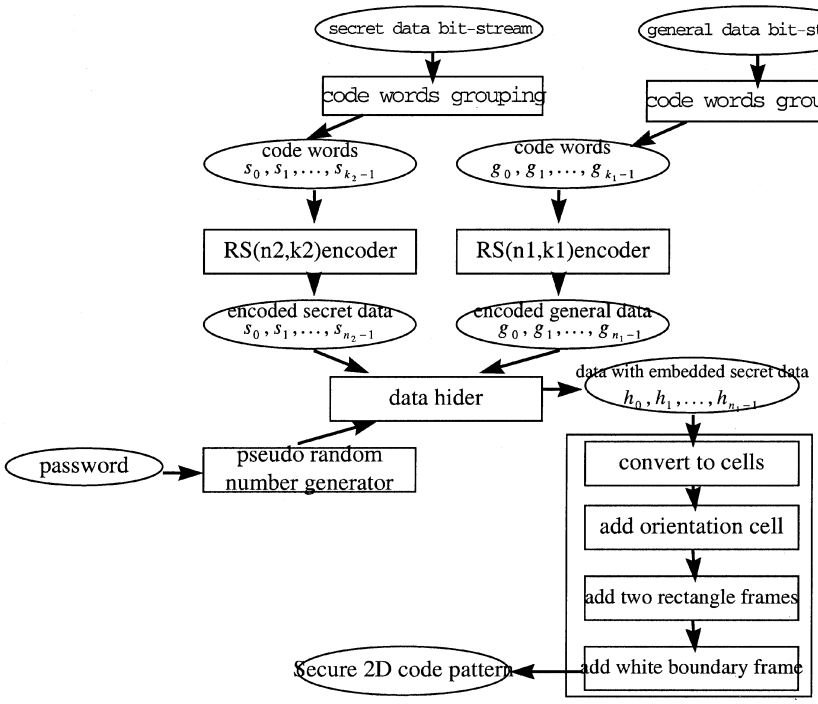


Fig. 4. The encoder of the proposed Secure 2D code system

The Secure 2D code is designed to code any kind of data, not only English characters, Chinese characters, numeric and special characters, but also images, voices, and graphics can easily be stored in Secure 2D codes. To achieve this aim, each kind of data is regarded as a bit-stream, and every consecutive m bits are grouped as a code word. Therefore, the system is application-independent, it will not interpret the bit-stream meaning, which will be done by another application program. Thus, data can be preprocessed (e.g., compressed, encrypted, etc.) before it is encoded.

2.2 Security

In the security part, a new data-hiding [16] technique, called data disguising, will be introduced. It will hide secret data in RS codes. The technique is different from "cryptology", but has the same security aim, it hides the secret data in meaningful data to form disguised data. The hiding process must make sure that the disguised data is still meaningful; thus, if an illegal user grabs the disguised data, he/she will consider it as true data. This will achieve the security aim.

Traditional cryptology methods regard the whole data as secret data. In contrast to the traditional method, the new idea that data is divided into general data and secret data is presented. Based on the data-disguising method mentioned above, the presented encoder translates the general data into a 2D code, then it embeds the secret data into the 2D code to form a Secure 2D code. The objective of this idea is that a general decoder can read the general data correctly without finding the embedded secret data. Furthermore, by using the proposed decoder, not only the general data can be obtained, the secret data can also be extracted. Thus, the data in the Secure 2D code is protected by means of different levels, and each user gets the data he/she is allowed to receive. For example, if the idea is applied in a name-card system, the home telephone and address can be set to be secret data, and it can only be extracted with permission.

2.3 Pattern generation/reading

A video camera is the input device of our system. Since the Secure 2D codes are in an open environment, many reading problems, such as rotation, scaling, and projected distortion, must be considered before designing the Secure 2D code pattern. In order to solve these problems, in the phase of pattern design, some features are added in a Secure 2D code to handle these problems and to speed up the process of pattern reading.

In the following two sections, we will describe the proposed encoder and decoder in detail.

3 Encoder

A detailed block diagram of the encoder is shown in Fig. 4. First, each set of consecutive m bits from the general and secret input bit-streams is grouped as a code word. Let $\{g_0, g_1, \ldots, g_{k_1-1}\}$ represent the grouped general data code words and $\{s_0, s_1, \ldots, s_{k_2-1}\}$ the secret data code words. Then, the RS(n_1, k_1) encoder is applied to $\{g_0, g_1, \ldots, g_{k_1-1}\}$ to obtain the redundancy code words $\{g_{k_1}, g_{k_1+1}, \ldots, g_{n_1-1}\}$ for error correction purposes. Here, RS(n_1, k_1) is a t_1-error correction code with $t_1 = (n_1 - k_1)/2$, and we assume that at most t_p errors will occur in a practical environment. In the part concerning security, the error correction codes $\{s_{k_2}, s_{k_2+1}, \ldots, s_{n_2-1}\}$ are added behind $\{s_0, s_1, \ldots, s_{k_2-1}\}$ by the RS(n_2, k_2) encoder, where $n_2 \leq t_1 - t_p$ and $2t_p \leq n_2 - k_2$. Here, RS(n_2, k_2) is a t_2-error correction code with $t_2 = (n_2 - k_2)/2$. After the error correction scheme is applied, the data disguising is conducted. The main idea is to

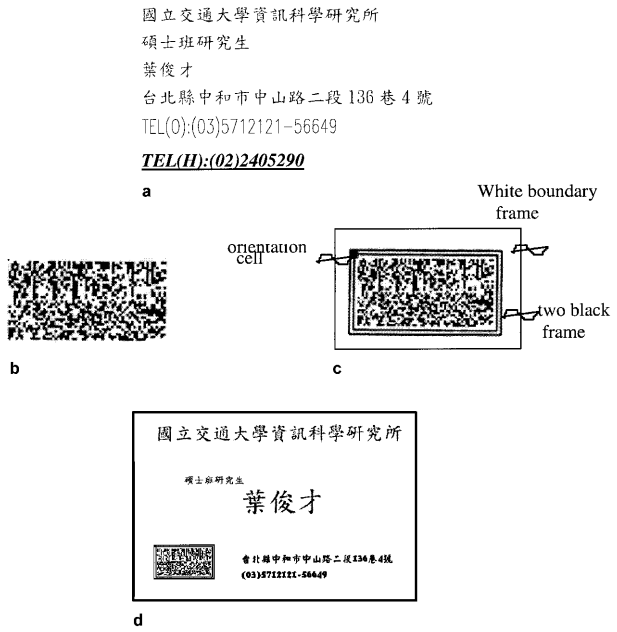


Fig. 5a–d. An example of a Secure 2D code pattern generated. **a** The information about the person "葉俊才". **b** The coded result of **a** through the RS(255, 127) code for general data and RS(48, 16) code for secret data. **c** The completed Secure 2D code pattern with three features added. **d** The personal name card of **a** with size 9×5cm on which the corresponding 2D code pattern shown in **c** is imposed

hide each secret data code word in the general data code word. That is, for each s_i, find a g_j and replace g_j by s_i. Obviously, the data g_j will be lost. Fortunately, the RS error correction decoder can recover g_j. To implement this idea, a password is used as the seed of a pseudo-random-number generator, and n_2 integer numbers between 0 to $n_1 - 1$ are generated without repetition. These n_2 numbers are used as the replaced addresses, the data with the embedded secret data, represented by $\{h_0, h_1, \ldots, h_{n_1-1}\}$, are obtained. $\{h_0, h_1, \ldots, h_{n_1-1}\}$ is regarded as a bit-stream, every bit is converted to a black cell (the bit value is 1) or a white cell (the bit value is 0), the cell placement order is from left to right and from top to bottom, and a rectangular pattern of 2D code is generated (see Fig. 5b). After a 2D code is generated, three important features are added around the 2D code boundary; these features assist the pattern reader to detect pattern orientation (see Fig. 5c), to reduce the effect of complex background, to solve the project distortion, to treat the scaling problem and to aid the pattern to be read as soon as possible. First, a 1×1 black orientation cell at the top-left corner of the 2D code is added to determine the pattern orientation (in fact, in the proposed system, the last m cells in the lowest row of the 2D code pattern are not used and can be used as orientation cells, although this is not done in the proposed system; but modifying the system to achieve the aim is an easy job). Secondly, two black rectangular frames with one cell width and one cell distance between them are added. Finally, a white boundary frame with more than five cells widths surrounding the boundary of the two rectangular frames is added. Thus, the Secure 2D code is generated and can be created by a laser printer.

Note that the n_2 locations used to hide the secret data seems known in advance and can be considered erasure-error locations. However, for a user not allowed to read the secret data, these locations are still unknown, thus the proposed system will not consider them as erasure-error locations.

Figure 5 shows an example of a generated Secure 2D code pattern. Figure 5a shows the information of a person's name card, the first row with 13 Chinese characters means "institute of computer and information science, National Chiao-Tung University", the second with 6 Chinese characters means "graduated student", the third with 3 Chinese characters shows the student's Chinese name, the fourth describes the student's Chinese address. The meaningful text shown in Fig. 5a is divided into secret data and general data. Since the home telephone number is private, it is considered as secret data, and others are general data. For these data, except the phone number with each numerical character represented by 1-byte ASCII code, each character is represented by 2-byte BIG5 code. Figure 5b shows the result of Fig. 5a coded by means of the RS(255, 127) code for general data (127 source bytes are needed; here, we only have 92 bytes, the remaining 35 bytes are filled with 0) and the shortened RS(48, 16) code for secret data. Figure 5c shows a completed Secure 2D code pattern with three features added. A prototype system has been developed for a name card management. Figure 5d shows a name card generated by our 2D code card management system.

4 Decoder

Figure 6 shows the detailed block diagram of the decoder for Secure 2D code. First, the image of Secure 2D code is captured by a video camera and is processed by the pattern reader. Since the pattern reader consists of several steps, it will be described in Sect. 6. After the pattern is processed, the code words $\{h'_0, h'_1, \ldots, h'_{n_1-1}\}$, which are from $\{h_0, h_1, \ldots, h_{n_1-1}\}$ and may be corrupted, are obtained. Then, a general user without the password can retrieve the general data by using the RS(n_1, k_1) decoder to carry out error correction and to remove redundancy codes. On the other hand, a permitted user with a password on hand can obtain not only the general data via the RS(n_1, k_1) decoder, but also the secret data by the following strategies. The secret data is obtained by using the password as the seed of the pseudo-random-number generator, the addresses of secret data code words embedded in $\{h'_0, h'_1, \ldots, h'_{n_1-1}\}$ are generated, the code words $\{s'_0, s'_1, \ldots, s'_{n_2-1}\}$, which are from $\{s_0, s_1, \ldots, s_{n_2-1}\}$, and may be corrupted, are then extracted. Finally, the secret data is obtained by using RS(n_2, k_2) to correct errors and to remove redundancy. Note that, although the secret data is embedded in the general data, a general user can obtain the integral general data without using a password. This will reduce the doubt that the secret data is embedded and will provide the first defense for secret data. On the other hand, even if the general user notices that the secret data is embedded, he/she is unable to extract the secret data without a password. This provides the second defense for secret data. Furthermore, our method also puts emphasis on the stability of secret data, that is,

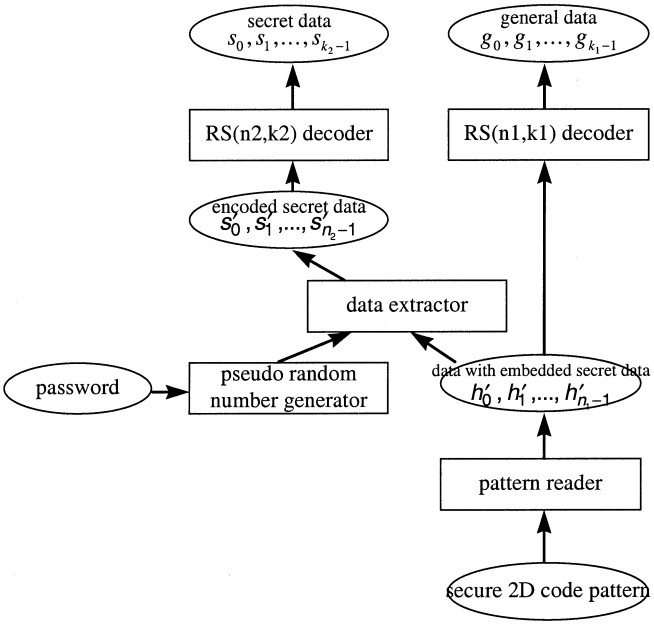


Fig. 6. The decoder of the proposed Secure 2D code system

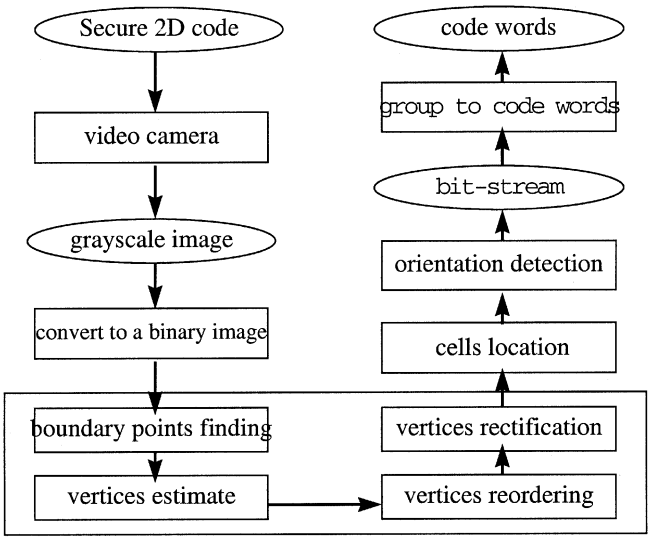


Fig. 7. The block diagram of pattern reader

even if the embedded secret data is polluted, it can still be recovered correctly.

5 Performance analysis for data hiding

"Hide data in RS code" is a new concept and one of the main contributions of the proposed method. It provides high security, but some error correction capability will be sacrificed. Here, we will give an analysis of the quantity of the embedded data and the error correction capability of RS code. Since RS(n_1, k_1) code is adopted for general data, the error correction capability t is $t = (n_1 - k_1)/2$. Suppose the maximum error t_p in the practical environment is known, then the maximum quantity for hiding data is $t - t_p$. Since the shortened RS(n_2, k_2) code is used to treat secret data, this implies $n_2 \leq t - t_p$. Assume t_p errors all occur in the embedded data, the secret data should be able to be recovered correctly. To achieve this aim, the maximum quantity of secret data k_2 is determined by the following formula:

$$2t_p \leq n_2 - k_2 \quad \text{and} \quad n_2 \leq t - t_p$$
$$\Rightarrow 2t_p \leq t - t_p - k_2 \Rightarrow k_2 \leq t - 3t_p\,.$$

When $k_2 = t - 3t_p$, the quantity of secret data is maximum, and the encoded secret data is still stable.

Example 1. Suppose RS$(255, 127)$ over GF(2^8) is used to code general data. In the practical environment, at most 16 errors will occur, i.e., $t_p = 16$. Thus, the real error correction capability t will be $t = (255-127)/2 = 64$ and the maximum quantity of the embedded data can be $t - t_p = 64 - 16 = 48$. Assume that a shortened RS$(48, k_2)$ code over GF(2^8) is used to code secret data. For the robustness of secret data, the secret data must be able to be recovered when a maximum of 16 errors all occur in the 48 embedded data. To achieve this, the quantity of secret data k_2 should be not greater than $16(64 - 3 \times 16)$. That is, a maximum of 16 bytes of secret data can be encoded by the shortened RS$(48, 16)$ code and the 48 bytes of encoded data can be embedded into 255 bytes of general data to form disguising data.

6 Pattern reader

Figure 7 shows the block diagram of the pattern reader. The objective of the pattern reader is to convert the image of Secure 2D code to code words. The gray-scale image of Secure 2D code is converted to a binary image by means of a local binary thresholding method [15].

After the binary image is obtained, we then proceed to locate the coordinates of four vertices of the Secure 2D code pattern. The job includes four parts: boundary points finding, vertices estimate, vertices reordering, and vertices rectification.

For boundary points finding, four scanning directions shown in Fig. 8a are used sequentially to find the boundary of the rectangle pattern. First, taking one scanning direction and tracing the first scanning line, while a black pixel is encountered, one run in front of this pixel and three runs starting from this pixel are checked to see if they are consistent. The consistent condition is that the ratio of these run lengths should be $m : n : n : n$ $(m \geq 5n)$. If consistent, the location of this pixel is recorded as a boundary point and the next scanning line in the same direction is processed. In order to save time, the distance between scanning lines can be chosen experimentally. Figure 8b shows the result of boundary points finding with one pixel scanning line distance.

After extracting boundary points, the four vertices of the outer rectangle frame can be estimated. First, the Hough transform [15] is applied to the obtained boundary points to obtain four boundary lines and then the four vertices can be easily estimated by finding the intersections of these four lines. A consistency check whether any intersection point is out of this image is carried out. If the consistency check fails, our 2D code system requests the user to capture the 2D code again.

After the four vertices are extracted, the only information we have is four coordinates of these four vertices. A

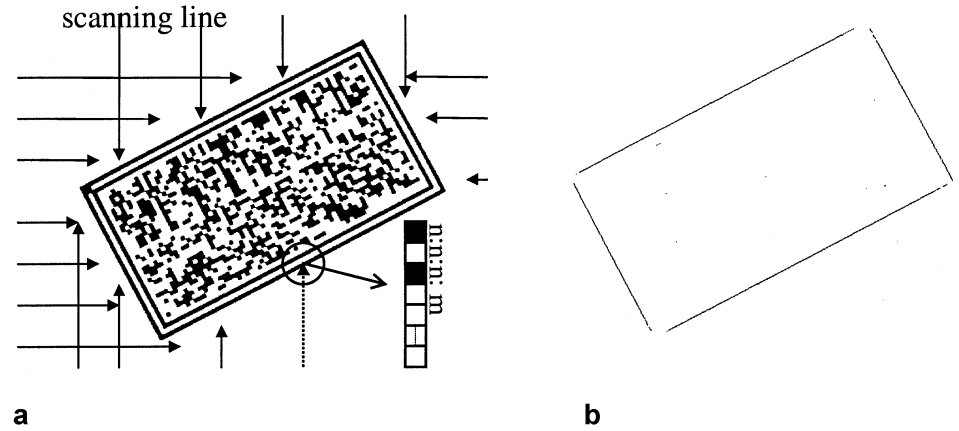


Fig. 8a,b. An example to illustrate boundary finding algorithm. **a** Four scanning directions: left to right, right to left, up to down and down to up shown by symbols "→", "←", "↓" and "↑". **b** Extracted boundary

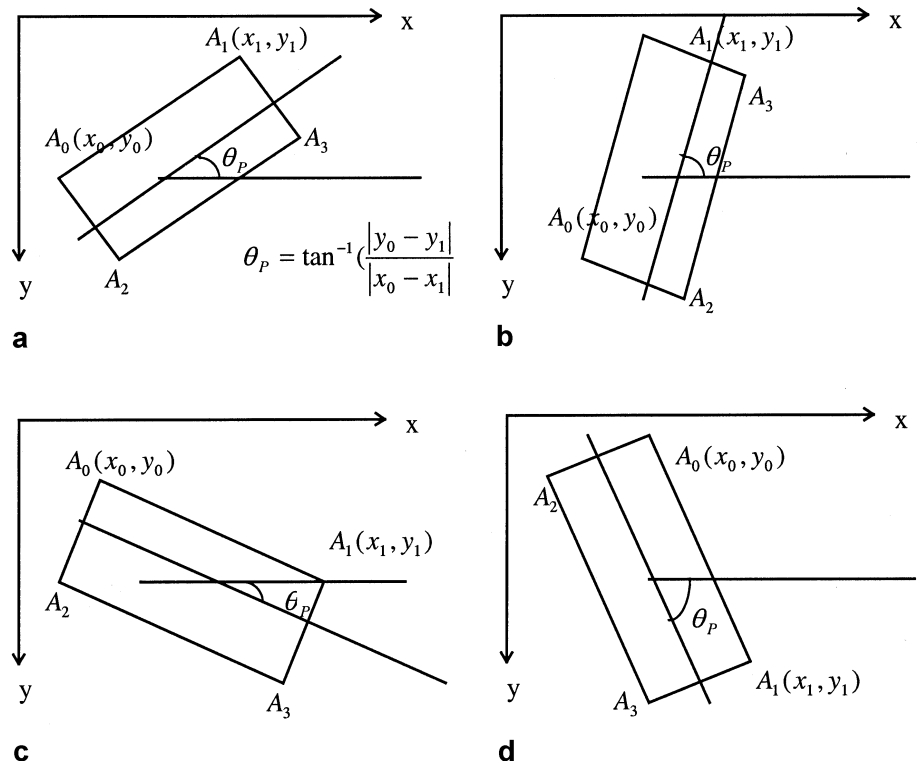


Fig. 9. Four different kinds of rotations are used to illustrate the "order" which is defined by the vertex reordering algorithm

vertex reordering algorithm is provided to obtain the pattern rotation angle θ_P and to give these four intersection points a defined order $\{A_0, A_1, A_2, A_3\}$. θ_P and $\{A_0, A_1, A_2, A_3\}$ will be used in the next step for vertex rectification. In this step, the orientation cell cannot be detected, because these points are still estimate vertices. Figure 9 shows the order which is defined by the vertex reordering algorithm. Four kinds of rotations are used to represent any angles of rotation. That is, for any four input coordinates, we will give them a defined order $\{A_0, A_1, A_2, A_3\}$ to stand for the top-left, top-right, bottom-left and bottom-right vertices of the rectangle frame.

6.1 Vertex reordering algorithm

input: four intersection vertices B_0, B_1, B_2, B_3 with arbitrary arrangement

output: A_0, A_1, A_2, A_3 which represent the top-left, top-right, bottom-left, bottom-right vertices of the rectangle frame.

Step 1. Find the leftmost point B_i, and assign it to C_0.

Step 2. Check

$$\overline{B_iB_k}\ \forall k \neq i \begin{cases} if\ (\overline{B_iB_k}\ .EQ.\ \max\{\overline{B_iB_j} \forall j \neq i\}), \\ \qquad setC_3 = B_k \\ if\ (\overline{B_iB_k}\ .EQ.\ \min\{\overline{B_iB_j} \forall j \neq i\}), \\ \qquad setC_2 = B_k \\ else\ C_1 = B_k \end{cases}$$

Step 3. Determine

$$\overline{C_0C_1}\ and\ \overline{C_0C_2} \begin{cases} if\ (From\ \overline{C_0C_1}\ To\ \overline{C_1C_2}\ isclockwise) \\ \qquad \Rightarrow A_i = C_i, 0 \leq i \leq 3 \\ else\ A_0 = C_2, A_1 = C_3, A_2 = C_0, A_3 = C_1 \end{cases}$$

Step 4. Let $(x_0, y_0), (x_1, y_1)$ be the coordinates of A_0 and A_1, then the orientation angle, θ_P, is as follows.

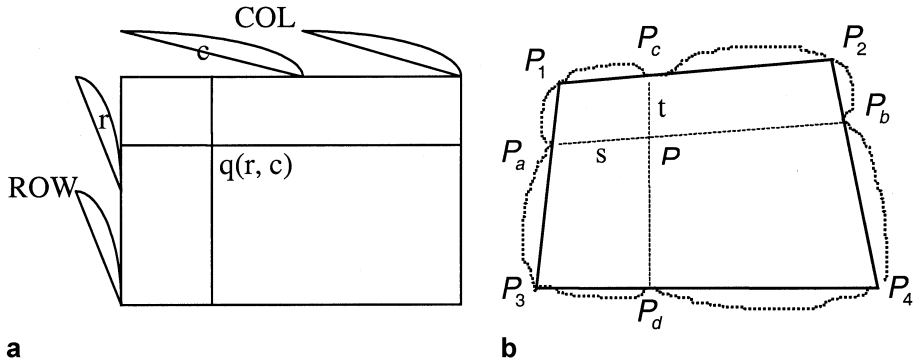


Fig. 10a,b. The relation between one rectangle pattern and its projection version. **a** The source pattern and q is one point in it. **b** The projected version of **a**, and P is the projected point of q

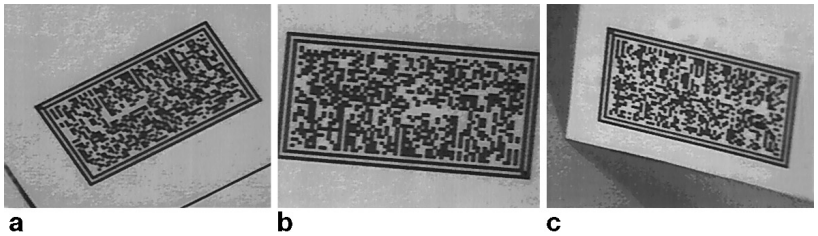


Fig. 11a–c. Patterns under different situations can still be read correctly. **a** Rotated. **b** Scaled. **c** Projected distortion

$$\theta_P = \tan^{-1}\left(\frac{|y_0 - y_1|}{|x_0 - x_1|}\right), \quad \text{where } x \neq x_1.$$

Due to the noise effect, the four reordered vertices A_0, A_1, A_2, A_3 and pattern angle θ_P are only rough positions and value. Since the preciseness of these four reordered vertices will affect the correctness of translation, a rectification method is proposed to obtain the precise positions of these four points.

To carry out vertex rectification, a "matching" algorithm is provided and applied in four areas around these four reordered vertices to obtain four more accurate vertices. First, four standard corner patterns C_0(◪), C_1(◧), C_2(◨), C_3(◩) with size 5×5 corresponding to A_0, A_1, A_2, A_3 are created. Secondly, rotate these four standard corner arrays by the θ_P angle to obtain the rotated corner patterns C'_0, C'_1, C'_2, C'_3.

To implement the rotation work, for every point $(\hat{x}, \hat{y})$ in a rotated pattern C'_i, use the following equation

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta_P & \sin\theta_P \\ -\sin\theta_P & \cos\theta_P \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}$$

to get the corresponding point (x, y) in C_i. Set the value of $(\hat{x}, \hat{y})$ to be the value of (x, y). Finally, each rotated pattern C'_i is used to find the best match array with its origin around vertex A_i. To do this, for each array A in the pattern with the same size and orientation as C'_i and the origin inside the 7×7 window centered at A_i, evaluate the matching score according to the rule: if $A(i, j)$ equals $C'_i(i, j)$, then $score = score + 1$. Four modified vertices MA_0, MA_1, MA_2, MA_3, which are the origins of arrays with the highest scores, are obtained.

Let **R** be a rectangle pattern, **S** is its image through a camera system. If the image plane is not parallel to the plane containing **R**, **S** can be considered to be the projection version of **R** on the image plane. If the angle between the plane with **R** and the image is small, we can assume that the mapping between one point $q(r, c)$ in **R** and its projected point P in **S** (see Fig. 10) is expressed by

$$\begin{aligned} P = &\left(\frac{COL - c}{COL}\right)\left(\frac{ROW - r}{ROW}\right) P_1 \\ &+ \frac{c}{COL}\left(\frac{ROW - r}{ROW}\right) P_2 \\ &+ \frac{COL - c}{COL}\left(\frac{r}{ROW}\right) P_3 + \frac{c}{COL}\left(\frac{r}{ROW}\right) P_4 , \end{aligned}$$

where P_1, P_2, P_3, P_4 are four vertices of **S**, ROW and COL are the width and height of **R**.

The equation above is obtained under the assumption that $\overline{P_1P_a} : \overline{P_1P_3} = \overline{P_2P_b} : \overline{P_2P_4} = r : ROW$ and $\overline{P_1P_c} : \overline{P_1P_2} = \overline{P_3P_d} : \overline{P_3P_4} = c : COL$ (this means that the lengths of these line segments are proportional).

So far, MA_0, MA_1, MA_2, MA_3, are the obtained coordinates of the four vertices of Secure 2D code pattern. These four coordinates will be used to determine the coordinates of every cell and its color. By means of Eq. 1, for each point (i, j) in the original pattern, the projected point (x, y) in the image pattern can be determined by the following equation.

$$\begin{aligned} (x, y) = &\left(\frac{COL - j}{COL}\right)\left(\frac{ROW - i}{ROW}\right) MA_0 \\ &+ \frac{j}{COL}\left(\frac{ROW - i}{ROW}\right) MA_1 \\ &+ \frac{COL - j}{COL}\left(\frac{i}{ROW}\right) MA_2 \\ &+ \frac{j}{COL}\left(\frac{i}{ROW}\right) MA_3 \end{aligned} \tag{1}$$

where MA_0, MA_1, MA_2, MA_3, are the four vertices of the Secure 2D code pattern, and ROW and COL are the width and height of Secure 2D code pattern.

As soon as each cell is located, whether the color of the cell is black or white can be determined. Then, the bit-stream can be obtained by converting each black cell into a bit with value 1 and each white cell into a bit with value 0. Finally, the orientation cell is used to determine the orientation of the pattern. If the orientation cell is located around MA_0,

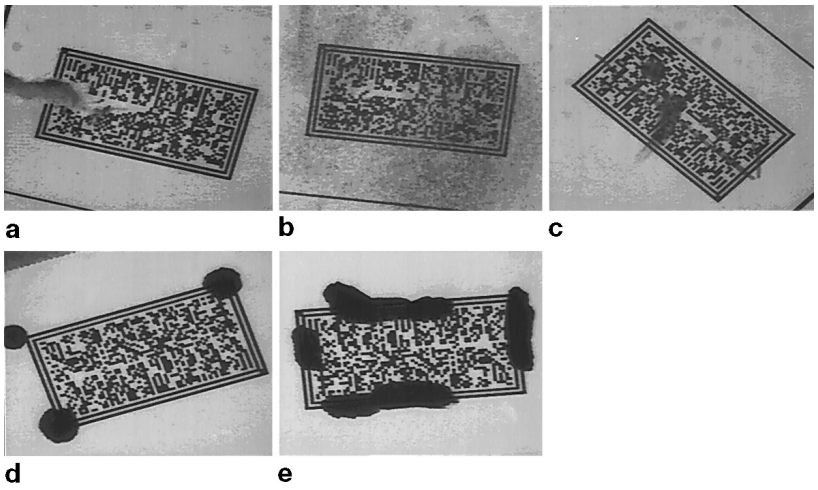


Fig. 12a–e. Patterns with different errors can still be read correctly. **a** Broken. **b** Dirty. **c** Pen drawing. **d** Error in vertices. **e** Error in boundary

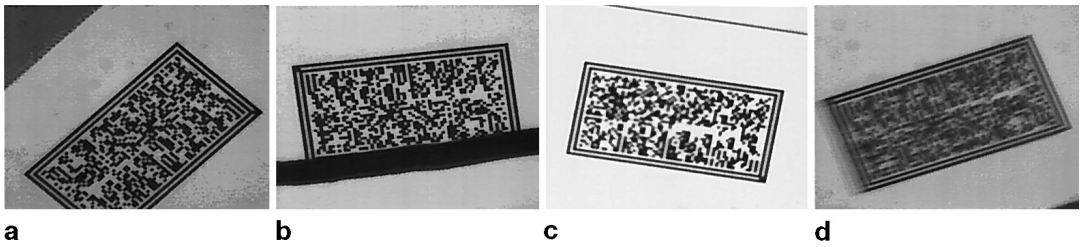


Fig. 13a–d. Four examples of requesting the user to capture again. **a** Incomplete pattern. **b** Pattern with corrupted edge. **c** Blurry pattern. **d** Pattern with quick-motion

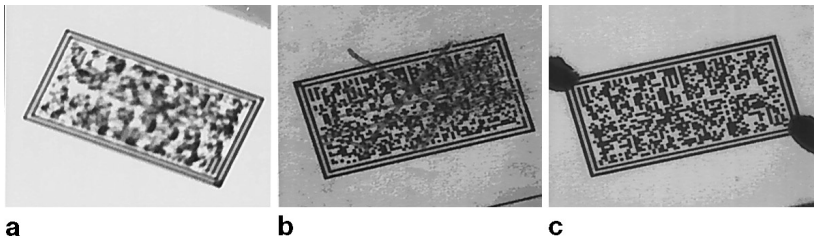


Fig. 14a–c. Three fail examples. **a** Blurry pattern. **b** Pattern with too much noise. **c** Pattern with the orientation cell corrupted

the reading direction is from MA_0 to MA_1 and the reading sequence is row by row from MA_0 to MA_3. If the orientation cell is located around MA_3, the reading direction is from MA_3 to MA_2 and the reading sequence is row by row from MA_3 to MA_1. Finally, code words are obtained from the bit-stream.

7 Experimental result

Currently, our system runs on a Pentium 133-MHz PC with an HP laser printer and a video camera. The laser printer with 600 dpi is used to print the 2D code pattern, and the video camera is used to capture the pattern of Secure 2D code at a resolution of 640 $\times$ 480 pixels under different lighting conditions. Our system includes two main parts: the Secure 2D code encoder and the Secure 2D code decoder.

From experience, if each cell has at least 5 pixels width, then rotated, scaled and projected distortion patterns can be read correctly. Figure 11 shows some examples. However, if θ_P of the rotated pattern is 90°, the user is requested to capture again.

Since a Secure 2D code pattern is allowed to be exposed under an open environment, many kinds of errors may occur. We created many kinds of errors in 2D patterns to simulate the real situations shown in Fig. 12. Although some errors are hyperbolic or in extreme positions, we can still recover them. The errors we create consist of dirty, breaking, projection distortion, pen drawing, noise in boundary and in vertices. The system can deal with various rotations, scales and unbalanced lighting conditions. Since a consistent check is carried out, we will reject some patterns with bad quality and request the user to capture again, as shown in Fig. 13. An incomplete pattern with only three corners is shown in Fig. 13a, although the reader is able to handle a small corner missing, our system is designed to request the user to capture again; this is due to that we do not know how many data have been lost. Some fail decoding examples are shown in Fig. 14. From these figures, we can see that the proposed system is very reliable.

8 Conclusions

In this paper, a 2D code system called Secure 2D code system applied to three major topics (error correction, security, pattern generator/reader) is presented. In the part of error correction, RS code is used to ensure the reliability. In the part of security, a data-disguising method based on the concept of data hiding is proposed to achieve the security aim. In the part of pattern generator/reader, any kind of data can be encoded by the pattern generator. The pattern reader can treat noisy, distorted and broken patterns, and can also handle the different situations of rotation, scaling and lighting conditions. From experimental results, we can see that the pattern reader is robust and reliable.

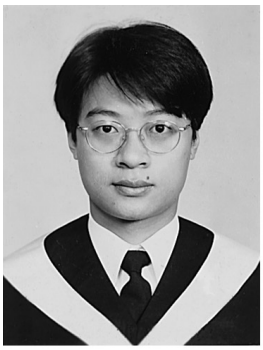
We have applied the Secure 2D codes to store name card information. In fact, some applications such as stock certificate, book management, passport, ID card, case history, and so on, can also adopt Secure 2D codes to store data.

Acknowledgements. This research was supported in part by the National Science Council of R.O.C. under contract NSC 87-2213-E-009-060

References

1. Pavlidis T, Swartz J, Wang YP (1990) Fundamentals of bar code in formation theory. Computer 23(4): 74–86
2. Pavlidis T, Swartz J, Wang, YP (1992) Information encoding with two-dimensional bar codes, Computer, 25(6):18–28
3. Wang YP (1989) Spatial information and coding theory, Ph.D. thesis, State University of New York at Stony Brook, N.Y.
4. Veritec Inc. Vericode Identification System, the Electronic Fingerprint (brochure). Calabasas Park, Calif.
5. IDMatrix Int. Inc. (1989) Datacode (brochure), Safety Harbor, Fla.
6. Pridy DG, Cymbalski RS (1989) Machine Readable Binary-Code. UK Patent Application GB 2 218 240 A, Nov. 8, Application No. 8910214.9
7. Little W, Kirlin L. Array-code identification tags. pp 1510–1514
8. Gils WJ van (1987) Two-dimensional dot codes for product identification. IEEE Trans Inf Theor IT-33(5):620–631
9. `http://www.cspi.com/vsystems/maxi_bd.htm`
10. `http://www.omniplanar.com/minicode.html`
11. Hattori M, Saitoh Y (1994) New Codes for Bar Code Type Recording Systems, Singapore ICCS'94 Conference Proceeding, pp 1054–1058
12. Lin S, Costello DJ (1995) Error Control Coding, Fundamentals and Applications, Prentice Hall, Englewood Cliffs, N.J.
13. Wicker SB, Stephen B (1995) Error Control Systems for Digital Communication and Storage
14. Wicker SB, Bhargave, VK (1994) Reed-Solomon Codes and Their Applications
15. Gonzalez RC, Woods, RE (1993) Digital Image Processing
16. Bender W, Gruhl D, Morimoto N, Lu A (1996) Techniques for data hiding, IBM Syst J 35(3&4)

Ling-Hwei Chen was born in Changhua, Taiwan, Republic of China, on February 18, 1954. She received the B.S. degree in Mathematics and the M.S. degree in Applied Mathematics from National Tsing Hua University, Hsinchu, Taiwan in 1975 and 1977, respectively, and the Ph.D. degree in Computer Engineering from National Chiao Tung University, Hsinchu, Taiwan in 1987. She is now a Professor of the Department of Computer and Information Science at the National Chiao Tung University. Her current research interests include image processing, pattern recognition, document processing, image compression, image cryptography and distributed database system.



Chung-Tsai Yeh was born in Taipei, Taiwan, Republic of China, on May 22, 1972. He received the B.S. degree in Applied Mathematics from Fu Jen University, Taipei, Taiwan in 1995, and the M.S. degree in Computer and Information Science from National Chiao Tung University, Hsinchu, Taiwan in 1997. His current research interests include imige processing, pattern recognition, and image cryptography.