Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 848276, 10 pages
http://dx.doi.org/10.1155/2013/848276

Research Article

2D Barcode Image Decoding

Jeng-An Lin and Chiou-Shann Fuh

Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

Correspondence should be addressed to Chiou-Shann Fuh; fuh@csie.ntu.edu.tw

Received 17 June 2013; Revised 25 October 2013; Accepted 16 November 2013

Academic Editor: Chung-Hao Chen

Barcode technology is one of the most important parts of Automatic Identification and Data Capture (AIDC). Quick Response code (QR code) is one of the most popular types of two-dimensional barcodes. How to decode various QR code images efficiently and accurately is a challenge. In this paper, we revise the traditional decoding procedure by proposing a serial of carefully designed preprocessing methods. The decoding procedure consists of image binarization, QR code extraction, perspective transformation and resampling, and error correction. By these steps, we can recognize different types of QR code images. The experiment results show that our method has better accuracy than Google open-source 1D/2D barcode image processing library Zxing-2.1. Moreover, we evaluate the execution time for different-size images. Our method can decode these images in real time.

1. Introduction

Barcode technology is one of the most important parts of Automatic Identification and Data Capture (AIDC); we can obtain decoded data through analysis of a barcode. According to the encoding type of barcodes, we divide barcode into two categories: one-dimensional barcode and two-dimensional barcode. One-dimensional barcode typically consists of varying the widths and spacings of parallel lines. Moreover, two-dimensional barcode is a graphical image that stores information both horizontally and vertically. Two-dimensional barcode compared with the one-dimensional barcode has the following advantages: (1) high data capacity; (2) no additional storages; (3) error correction ability.

Quick Response code (QR code) is one of the most popular types of two-dimensional barcodes developed in Japan by Denso Corporation in 1994. Although QR code was first designed for the automotive industry, QR codes are now used over much wider range of applications, including commercial tracking, transport ticketing, website Uniform Resource Locator (URL), and identity verification. QR code has several advantages. First, QR code has strong error correcting capability, which can restore 30% data for the maximum error correction level. Second, QR code can be scanned from any direction because the proportion of position detection patterns is not changed with the scanning direction. Third, QR code supports different encoding types and versions. We can choose an appropriate encoding type and version to reduce the size of QR code image.

Although there exist many commercial programs for QR code decoding, inventing more robust QR code decoding strategy still attracts many attentions from researchers. Many works are proposed recently. In [1, 2], Belussi and Hirata proposed a fast component-based two-stage approach for detecting QR codes in arbitrarily acquired images. Moreover Ohbuchi et al. [3] and Liu et al. [4] focused on enabling mobile phone to recognize QR code under poor conditions in real time. For decoding QR code image correctly and efficiently, we confront some challenges as follows. (1) When we capture QR code images using a phone camera in nonuniform background or uneven light, it might affect the decoded result. (2) The shooting angle might cause perspective distortion; to restore a distorted QR image is an important issue for decoding QR code images. (3) Decoding QR code images needs a sequence of sophisticated image processing. How to reduce the complexity of image processing and maintain the decoding accuracy rate is also a challenge.

In this paper, we focus on revising preprocessing methods in the traditional decoding procedure that is straightforward but still have much improving potential to overcome

the above challenges. The decoding steps consist of image binarization, QR code extraction, perspective transformation and resampling, and error correction. In image binarization, we use a modified local threshold method to resolve the nonuniform background and uneven light problems. Our QR code extraction algorithm uses the position detection patterns of QR code efficiently and accurately to locate the QR code position. After locating the QR code position, we restore a distorted image through perspective transformation and resample. Finally, we get the corrected data by Reed-Solomon error correction algorithm.

2. Related Work

QR code recognition technology has been studied in past years. We can simply divide the recognition into two steps, image preprocessing and QR code extraction.

In image preprocessing, some researchers focus on image denoising [5] or camera shaking [6]. In order to improve the performance of low resolution QR-code recognition, a previous work [7] uses the super-resolution technique that generates a high resolution image from multiple low-resolution images. Moreover, some researchers use different binarizations to improve the nonuniform background and uneven light problems [8, 9].

In QR code extraction, researchers propose several different methods to locate and extract the QR code of images. Some researchers use the feature of finder pattern to find rough QR code position [10–12]. After estimating the QR code's four corners using the rough QR code position, they effectively extract the QR code of images. In research [6, 13], they use edge detection to find possible rough barcode area. Then, morphological dilation and closing are used to generate more compact regions. Finally, the position of QR code of images can be detected.

Although these researches have their contributions, there are some shortcomings we can improve. In [5], researchers only focus on image denosing, but locating QR code position is an important part of QR decoding. They did not propose their method to process this problem. Previous works [6, 7, 10] need enormous calculation. Their methods are difficult to decode QR code image in real time. In research [9], their binarizaiton method needs to know the version of QR code in advance. In this paper, we propose a decoding method to improve the deficiencies in previous work.

3. Background

3.1. Perspective Transformation. QR code images may be distorted due to camera perspective projection. Therefore, we use perspective transformation to restore the distorted QR code images. The general representation of a perspective transformation [14] is

$$\begin{bmatrix} x' & y' & w' \end{bmatrix} = \begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad (1)$$

where $x = x'/w'$ and $y = y'/w'$. In (1), each original coordinate (u, v) can be transformed to a new coordinate (x, y) by using the 3 by 3 matrix. Based on (1), x and y can be written as

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}},$$
$$y = \frac{y'}{w'} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}}. \quad (2)$$

Without loss of generality, the 3 by 3 matrix can be normalized, so that $a_{33} = 1$. Equation (2) can be rewritten as

$$x = a_{11}u + a_{21}v + a_{31} - a_{13}ux - a_{23}vx,$$
$$y = a_{12}u + a_{22}v + a_{32} - a_{13}uy - a_{23}vy. \quad (3)$$

Applying four pairs of correspondence points to (3), we can get a linear system as follows:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix} \times \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{13} \\ a_{23} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}. \quad (4)$$

The value of unknown coefficients will be determined by solving the linear system in (4).

3.2. Reed-Solomon Error Correction. Reed-Solomon codes are nonbinary cyclic error-correcting codes [15]. In QR code decoding procedures, we might obtain some error information because of distorted images or inappropriate binarization. Reed-Solomon codes help us to correct the error information. The general decoding algorithm [16] of RS codes consists of computing the syndromes, determining an error locator polynomial, and solving the error values.

For computing the syndromes, the received message is viewed as the coefficients of a polynomial $S(x)$:

$$S(x) = \sum_{i=0}^{n-1} r_i x^i, \quad (5)$$

where r_i is the received message at position i. Therefore, the syndromes $(S_0, S_1, \ldots, S_{2t-1})$ are defined as

$$S_0 = S\left(\alpha^0\right), \qquad S_1 = S\left(\alpha^1\right), \ldots, S_{2t-1} = S\left(\alpha^{2t-1}\right), \quad (6)$$

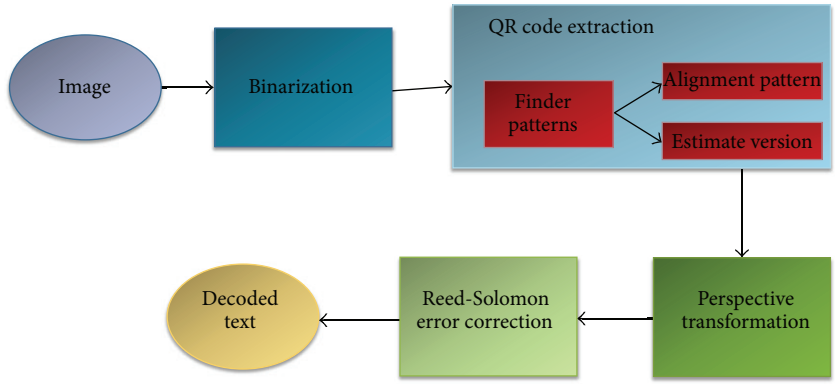


Figure 1: QR code decoding procedure.

where $2t$ is equal to the number of codewords available for error correction and α is the primitive element on the Galois field. Syndrome polynomial is defined as

$$S(x) = \sum_{i=0}^{2t-1} S_i x^i. \tag{7}$$

After computing syndrome polynomial, we use Euclidean algorithm to construct the error locator polynomial and error evaluator polynomial. Error locator polynomial can be used to find the error positions. Error evaluator polynomial is used to find the error values. The Euclidean algorithm is based on the key equation in

$$\Omega(x) = S(x)\Lambda(x) \left(\mathrm{mod} x^{2t}\right). \tag{8}$$

Given only $S(x)$ and t, we want to obtain the error locator polynomial $\Lambda(x)$ and the error evaluator polynomial $\Omega(x)$. Equation (8) means that

$$\theta(x)x^{2t} + S(x)\Lambda(x) = \Omega(x) \tag{9}$$

for some polynomial $\theta(x)$. Finally, we can use Euclidean algorithm to generate the error locator polynomial and error evaluator polynomial.

For solving the error values, Forney's algorithm is selected. The error values for a Reed-Solomon code are computed by

$$e_{i_k} = -\frac{\Omega\left(X_k^{-1}\right)}{\Lambda'\left(X_k^{-1}\right)}, \tag{10}$$

where e_{i_k} is the corrected value at position i_k; X_k^{-1} is the root of $\Lambda(x)$ at position k; and $\Lambda'(x)$ is the formal derivative of $\Lambda(x)$. By (10), we obtain the corrected data of QR code images.

4. The Proposed Method

We design a QR code recognition procedure for decoding QR code accurately and rapidly. First, we propose a new local binarization method to overcome images with uneven light and nonuniform background problems. And then we use three finder patterns and an alignment pattern of QR code efficiently to locate the QR code position. After locating QR code position, we resample QR code image by perspective transformation. Finally, we get the corrected data by Reed-Solomon error correction algorithm. Figure 1 illustrates the procedure.

4.1. Binarization. Binarization is an important process for accurately recognizing black-and-white module in QR code images. Therefore, we propose a new binarization method to improve the recognition of images with nonuniform background and uneven light. Our method is a local threshold algorithm based on Sauvola's method [8]. We decompose each image into a series of blocks and then calculate a threshold T_1 for each block using

$$T_1(x, y) = \mathrm{mean}(x, y) \times \left[1 + \left(\frac{\mathrm{standard\ deviation}(x, y)}{R}\right)\right]. \tag{11}$$

The default R is 1250. For each block, we convolve with the kernel in (12) to calculate a threshold T_2. By using the threshold T_2, we split each block into bright area and dark area. Figures 2 and 3 shows that our method can achieve the same performance with less computing cost, compared with modified Sauvola's method [8]:

$$C = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{12}$$

$$f(A, B, C) = (B \cdot x - A \cdot x)(C \cdot y - A \cdot y) - (B \cdot y - A \cdot y)(C \cdot x - A \cdot x). \tag{13}$$

4.2. Estimating Version. QR code extraction is responsible for searching QR code area in images. QR code extraction consists of searching finder patterns, estimating version, and

Figure 2: An uneven light image [17].

Figure 3: Binary image using our method.

searching alignment pattern. We use the proportion of finder patterns to locate the QR code position. After searching three finder patterns, we use the distance between these patterns to estimate version. Finally, we develop a connected-component method to search alignment pattern accurately.

4.2.1. Searching Finder Patterns. In searching finder patterns, we need to determine the upper left, lower left, and upper right finder patterns in the QR code. First, we scan the binary image horizontally to find all points matching the ratios 1 : 1 : 3 : 1 : 1. For each point matching the ratios, we check its vertical direction. Finally, we have collected all points in images matching the ratios 1 : 1 : 3 : 1 : 1 both horizontally and vertically.

After collecting points, we merge points in the same finder pattern. If the number of finder patterns is more than three, we filter out some inappropriate points. First, we calculate the angle between three points. If the maximum angle is greater than a threshold T_1 or the minimum angle is less than a threshold T_2, the points will be filtered. And then we identify the direction of finder patterns. The maximum angle corresponding to the finder pattern is the upper left finder pattern. Finally, we use cross product in (13) to determine the lower left and upper right finder patterns, where A is the upper left finder pattern; B and C are the other finder patterns; and $A \cdot x$ and $A \cdot y$ represent the x and y coordinate of A. If $f(A, B, C) < 0$, then B is upper right finder pattern, and C is lower left finder pattern.

Figure 4: Red line is the width of finder pattern, and blue line is the estimated width.

4.2.2. Estimating Version. After searching three finder patterns, we can estimate version using module size and the distance between two finder patterns. Module size can be calculated by

$$\text{Module\_size} = \frac{\text{Width of finder pattern}}{7}. \tag{14}$$

We calculate version by

$$\text{Version} = \frac{\left(\left(\text{Euclidean\_distance}\left(f_1, f_2\right)/\text{Module\_size}\right) - 10\right)}{4}, \tag{15}$$

where f_1 is upper left finder pattern and f_2 is upper right finder pattern. We can get rough version by (15), but (15) misjudges the version when QR code image is rotated. We overestimate the width of finder pattern in rotated image in Figure 4.

As a result, we underestimate the version of QR code image. Therefore, we revise (15) to be

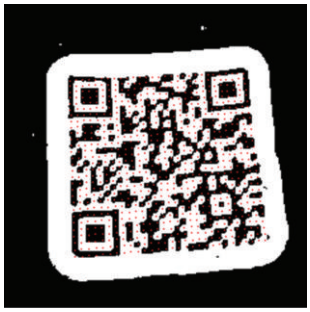
$$\text{Version} = \frac{\left(\text{Euclidean\_distance}\left(f_1, f_2\right)/\text{Module\_size} * \cos\theta - 10\right)}{4}, \tag{16}$$

where θ is the angle between blue line and red line in Figure 4. We estimate version more accurately by (16) and thus before perspective transformation.

Table 1: Eight coordinates used for perspective transformation ($D = 17 + 4^*$ version).

Function pattern of QR code	Original coordinate	Normalized coordinate
Upper left finder pattern	(x_0, y_0)	(3.5, 3.5)
Upper right finder pattern	(x_1, y_1)	(D-3.5, 3.5)
Lower left finder pattern	(x_2, y_2)	(3.5, D-3.5)
Lower right alignment pattern	(x_3, y_3)	(D-6.5, D-6.5)

Table 2: Result of resampling.

Original image	Normalized image
	
	
	

4.3. Searching Alignment Pattern. Low resolution image or inappropriate binarization method will result in the deformation of alignment pattern. Therefore, we propose a new searching alignment method to overcome the problem. First, we roughly locate the alignment pattern using three finder patterns. Moreover, we extract subimage with a specified radius in Figure 5. We calculate the connected component in the subimage, both black pixels and white pixels. After calculating the connected component, we get two images in Figures 6(b) and 6(c).

For each connected component of black pixel C_i in Figure 6(b), we check its border pixels. If all border pixels are adjacent to the same connected component of white pixel C'_j in Figure 6(c), then we check all border pixels in C'_j. Once all border pixels in C'_j are adjacent to the same black pixel connected component C_k in Figure 6(b), these components possibly contain alignment pattern. Finally, we calculate the centroid of C_i as the centroid of alignment pattern.

Table 3: Test images.

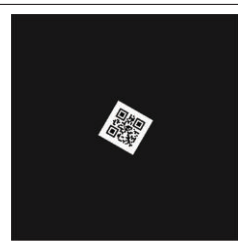
Set 1		
Set 2		
Set 3		
Set 4		
Set 5		
Set 6		

Table 4: Comparison between our method and Zxing-2.1.

	Our method	Zxing-2.1
Set 1	20/20	17/20
Set 2	32/34	30/34
Set 3	38/42	38/42
Set 4	36/48	36/48
Set 5	19/19	19/19
Set 6	15/15	15/15
Total	160/178	155/178

Table 5: Images we successfully recognize and Zxing-2.1 fails.

4.4. Perspective Transformation. After estimating the version and locating the centroid of three finder patterns and alignment pattern, we resample QR code image through perspective transformation. The eight coordinates can be presented as shown in Table 1.

To avoid the appearance of cracks and holes, we use inverse warping in perspective transformation. First, we calculate a transformation T from normalized coordinate to original coordinate. For each pixel in normalize image, we calculate its corresponding location in original image using the transformation T. Finally, we copy the pixel value of original image to the normalized image. Table 2 shows our resampling results.

Figure 5: We extract blue rectangle to find the alignment pattern (radius = 30 pixels).

5. Experiment Result

Experimental environment:

(1) CPU: Intel core i5-3210M CPU 2.50 GHz,

(2) memory: 4 GB,

(3) OS: Windows 7,

(4) programming language: Java.

In this section, we conduct several experiments to evaluate the performance. First, we compare the accuracy between our QR code decoding algorithm and 2D barcode image processing library (Zxing) [17]. The experiment result shows that our decoding algorithm has better accuracy than Zxing. And then we use different sizes of images to evaluate the decoding time. The experiment results show our method decoding images in real time. Moreover, we implement a QR code scanner on Android phone. We test the decoding ability of some defaced images. The experiment result shows that our QR code scanner successfully decodes these images.

5.1. Image Source. Our test images in Table 3 are from Google open-source 1D/2D barcode image processing library Zxing-2.1 [17]. The images are classified into six data sets, a total of 178 images. The first data set contains low resolution QR code images. The second data set contains trademark QR code image. The skew QR code images are included in the third data set. The fourth data set contains distorted QR code images. Uneven light QR code images are included in the fifth data sets. The sixth data set contains rotated QR code images.

5.2. Accuracy Analysis. We compare the accuracy between our decoding algorithm and Google open-source 1D/2D barcode image processing library Zxing-2.1 [17]. Table 4 shows the results.

In 178 test images, there are 8 images in Table 5 that we successfully recognize and Zxing-2.1 fails. Moreover, there are 3 images in Table 6 that Zxing-2.1 successfully recognizes and our method fails.

The accuracy of our method is about 90% and that of Zxing-2.1 is only 87%. Our method has better recognition rate

Table 6: Images Zxing-2.1 successfully recognizes and our method fails.

(a) (b) (c)

Figure 6: (a) Extracted image in Figure 5; (b) found connected components of black pixels; (c) found connected components of white pixels.

Table 7: The execution time between our method and Zxing for decoding different-size images.

	Our method	Zxing
320 ∗ 240	214 ms	139 ms
640 ∗ 480	312 ms	190 ms
800 ∗ 600	375 ms	237 ms
1024 ∗ 768	448 ms	264 ms

in Sets 1 and 2. Zxing-2.1 cannot accurately locate the alignment pattern because of the low resolution or inappropriate binarization. Our method improves the problem, so that we have better result.

5.3. Efficiency Analysis. In this section, we show the execution time between our method and Zxing for decoding different sizes of QR code images. We use four different sizes of QR code images to calculate the execution time. Each size contains five images. Table 7 shows the results.

Our method needs more decoding time than Zxing, but our method still can decode QR code images in real time. We only need 448 milliseconds for decoding resolution 1024 ∗ 768 pixel images.

5.4. Comparison of Different Barcode Applications. We implement a QR code scanner on Android phone. Moreover, we collect four popular barcode scanners from Google Play and iOS market. We compare the decoding ability of some defaced images. Table 8 shows the result. Our application can successfully decode all test images. Our application has the best result of these applications in these images. The reason for our better results is that two steps of the binarization stage in our method will get a more appropriate threshold to split each block into bright area and dark area. Moreover compared with current applications, our new searching alignment method could handle a large deformation of alignment pattern as the result of inappropriate binarization and low resolution which can be observed in sample images.

6. Conclusion

In this paper, our carefully designed preprocessing methods improve the accuracy of the traditional decoding procedure. Although refined steps improve accuracy at the expense of increased decoding time [21], the QR code still can meet the real-time requirement. The decoding steps consist of image binarization, QR code extraction, perspective transformation, and error correction. We efficiently resolve the non-uniform background and uneven light problems by using a modified local threshold algorithm. In QR code extraction, we use the position detection patterns of QR code efficiently and accurately to locate the QR code position. After locating the QR code position, we restore a distorted image through perspective transformation and resampling. Finally, we get the corrected data by Reed-Solomon error correction algorithm.

In the experiment, we compare the accuracy between our method and Zxing. The experiment result shows our

Table 8: Comparison of different barcode scanner applications.

	Our	Zxing [17]	Quickmark [18]	Bakodo [19]	BarCodeScan [20]
	O	O	X	X	X
	O	X	X	X	X
	O	X	O	O	O
	O	X	O	O	O
	O	O	O	O	X
	O	O	O	O	X

Table 8: Continued.

	Our	Zxing [17]	Quickmark [18]	Bakodo [19]	BarCodeScan [20]
	O	X	O	O	O
	O	X	O	O	O
	O	X	X	X	X
	O	X	X	X	X
Total	10/10	3/10	6/10	6/10	4/10

method to have better accuracy than Zxing. Moreover, we use different-size images to calculate the execution time. The experiment shows that our method can efficiently decode images. All images can be decoded in less than one second.

References

[1] L. F. F. Belussi and N. S. T. Hirata, "Fast component-based QR code detection in arbitrarily acquired images," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 3, pp. 277–292, 2013.

[2] L. F. F. Belussi and N. S. T. Hirata, "Fast QR code detection in arbitrarily acquired images," in *Proceedings of the Conference on Graphics, Patterns and Images (SIBGRAPI '11)*, pp. 281–288, Maceió, Brazil, August 2011.

[3] E. Ohbuchi, H. Hanaizumi, and L. A. Hock, "Barcode readers using the camera device in mobile phones," in *Proceedings of the International Conference on Cyberworlds (CW '04)*, pp. 260–265, Tokyo, Japan, November 2004.

[4] Y. Liu, J. Yang, and M. Liu, "Recognition of QR Code with mobile phones," in *Proceedings of the Chinese Control and Decision Conference (CCDC '08)*, pp. 203–206, Yantai, China, July 2008.

[5] C. C. Lin and M. S. Chen, *A general scheme for QR-code image denoising on the camera phone [M.S. thesis]*, Department of Electrical Engineering, National Taiwan University, 2009.

[6] C.-H. Chu, D.-N. Yang, and M.-S. Chen, "Image stablization for 2D barcode in handheld devices," in *Proceedings of the 15th ACM International Conference on Multimedia (MM '07)*, pp. 697–706, September 2007.

[7] Y. Kato, D. Deguchi, T. Takahashi, I. Ide, and H. Murase, "Low resolution QR-code recognition by applying super-resolution using the property of QR-codes," in *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR '11)*, pp. 992–996, September 2011.

[8] J. J. Zhou, Y. F. Liu, and P. Li, "Research on binarization of QR code image," in *Proceedings of the International Conference on Multimedia Technology (ICMT '10)*, pp. 1–4, Ningbo, China, October 2010.

[9] Z. Xiong, H. Cuiqun, L. Guodong, and L. Zhijun, "A binarization method of quick response code image," in *Proceedings of the 2nd International Conference on Signal Processing Systems (ICSPS '10)*, vol. 3, pp. 317–320, Dalian, China, July 2010.

[10] Y. H. Chang, C. H. Chu, and M. S. Chen, *A general scheme for extracting QR code from a non-uniform background in camera*

phones and applications [M.S. thesis], Department of Electrical Engineering, National Taiwan University, 2007.

[11] C. Y. Lai and M. S. Chen, *Extracting QR Code from a non-uniform background image in embedded mobile phones [M.S. thesis]*, Department of Electrical Engineering, National Taiwan University, 2007.

[12] Y. L. Pan and M. S. Chen, *2D barcodes from images with spatial distortion [M.S. thesis]*, Department of Electrical Engineering, National Taiwan University, 2008.

[13] D. Munoz-Mejias, I. Gonzalez-Diaz, and F. Diaz-De-Maria, "A low-complexity pre-processing system for restoring low-quality QR code images," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1320–1328, 2011.

[14] P. S. Heckbert, *Fundamentals of texture mapping and image warping [M.S. thesis]*, Department of Electrical Engineering, University of California, Berkeley, Calif, USA, 1989.

[15] Wikipedia, "Reed-Solomon Error Correction," 2013, http://en.wikipedia.org/wiki/Reed-Solomon_error_correction.

[16] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley, New York, NY, USA, 2005.

[17] S. Owen, "Zxing ("Zebra Crossing")," 2013, https://code.google.com/p/zxing/.

[18] SimpleAct, "QuickMark," 2013, http://www.quickmark.com.tw/.

[19] A. Dougla, "Bakodo," 2013, http://bako.do/.

[20] P. Giudicelli, "Barcode-Scanner," 2013, https://itunes.apple.com/tw/app/barcode-scanner/id478602171?mt=8.

[21] T. Falas and H. Kashani, "Two-dimensional bar-code decoding with camera-equipped mobile phones," in *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops '07)*, pp. 597–600, New York, NY, USA, March 2007.