Barcode Readers using the Camera Device in Mobile Phones

Eisaku Ohbuchi*
eisaku.ohbuchi@necel.com
NEC Electronics

Hiroshi Hanaizumi†
hana@k.hosei.ac.jp
Hosei University

Lim Ah Hock‡
limahhock@es.nec.com.sg
NEC Electronics Singapore

Abstract

This paper shows new algorithms and the implementations of image reorganization for EAN/QR barcodes in mobile phones. The mobile phone system used here consists of a camera, mobile application processor, digital signal processor (DSP), and display device, and the source image is captured by the embedded camera device.

The introduced algorithm is based on the code area found by four corners detection for 2D barcode and spiral scanning for 1D barcode using the embedded DSP. This algorithm is robust for practical situations and the DSP has god enough perfor4mance for the real-time recognition of the codes.

The performance of our image processing is 66.7 frames / sec for EAN code and 14.1 frames / sec for QR code image processing, and this is sufficient performance for practical use. The released mobile phone had performance of 5-10 frames / sec including OS and subsystem overheads.

Keywords: image processing, barcode, mobile phone, DSP (Digital Signal Processor)

1 Introduction

Current mobile phones implement various new kinds of applications such as taking photos, and movie shooting by using embedded camera devices, and this progress also depends on the camera device technology. For example, 30% of mobile phones in Japan have embedded camera devices at 2003, and some of those have more than over mega-pixel range image sensor cameras. Also, the embedded camera devices can be used as new input interfaces such as for symbol recognition. In Japan, most mobile phones, with camera devices support symbol recognition, such as of EAN barcode [1] and QR-code (2D barcode) [2], and these code symbols are used for easily accessing phone provided network services by reading URLs or such address characters.

Previous work [4] shows the application using a special tag, and this tag is captured from the embedded camera device of a PC and recognized by the PC software. In this application the special 2D barcode symbol and the standard PC are used. A study of recognition of 1D barcodes, [5] describes the 1-D barcode reader using a Hough transform, and this system is implemented in a PC.

On the other hand, the system architecture of mobile phones has been changed by the semiconductor vendors. The current or very recent generation of phone devices has only one main processing unit which is called a baseband processor, but the computation power is not enough for the newer applications which combine audio and visual data. The application revolution has added an extra processor called an application processor into the newer mobile phones. From this background, the current high-end mobile phones contain two processors a baseband processor, which works for voice digital signal processing, protocol control, power management, and fundamental functions, and an application processor, which works for extra application such as movie codec, still photo codec and resizing, and other extra applications. The application processor, especially, defines the characteristics of mobile phones at present. Major application processors are Texas Instruments's OMAP[1] and Renesas's SH-Mobile[2].

We introduces a new image reorganization algorithm for EAN barcode and QR-code (2D barcode), and this is implemented in our application platform, which was introduced in [3]. This algorithm was developed for the current mobile phone architecture which implements an embedded camera and an application processor including DSP. This algorithm used the spiral scanning method to detect a key black bar for 1D barcodes, and code area finding by four corner detection for 2D barcodes. In this study, the image processing part is implemented in DSP and the decoding and user software part are implemented in the host CPU of the application processor.

The hardware system and processing flow are described in section 2. In section 3, the algorithm for image recognition for 1D/2D barcode symbols is discussed, and its implementation and performance evaluation are explained in section 4, and in section 5 shows experimental results are given.

2 Systems

This section describes the hardware system architecture for implementing the barcode reading system in mobile phones, and its processing flow.

The camera device and application processor are necessary hardware components for this system. The application processor is needed to implement the camera interface, LCD controller, DSP for image processing, and application host CPU for real-time computations (Figure 1). The application processor works for displaying of the menu and preview in the display and computing of code recognition and decoding in real-time. With these systems, the user can control the position of the camera and decide the capture timing. The processing flow is as follows.

1. **Execute the barcode reader application** The application processor is changed into barcode reader mode by user menu selection.

*Mobile Systems Division, NEC Electronics Corporation. 1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan. Voice: +81-44-435-1792 FAX: +81-44-435-1811

†Computer and Information Science, Hosei University

‡Engineering Division, NEC Electronics Singapore Pte Ltd. 238A Thomson Road #12-08 Novena Square Singapore 307684, Voice: +65-6253-8311 FAX: +65-6250-3583

[1] http://www.ti.com/

[2] http://www.renesas.com/

Proceedings of the 2004 International Conference on Cyberworlds (CW'04)
0-7695-2140-1/04 $ 20.00 IEEE

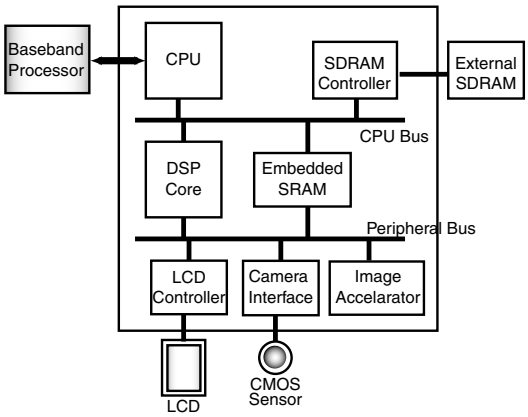

Figure 1: An example architecture of application processor for mobile phone.

2. **Capture from embedded camera device** The source images are captured by the embedded camera device via the camera interface, and these image are sent to two units, the DSP for image processing and the LCD controller for displaying the user preview.
3. **Process the image in DSP** The code is detected and processed in the DSP from the captured source image, and the processed image in this phase is output as the normalized size and binarized image of the code area.
4. **Decode the code** The processed code data in the previous phase is decoded in the host CPU, and the decoded code is derived to the application software.
5. **Display the results** The host application displays the decoded results.

In this paper, the algorithm of code recognition in the DSP is introduced mainly, and in the implementation and performance discussions, we estimated based on this system, because we believe the mobile phone system point of view is very important in practical terms.

3 Algorithms

Image processing for EAN code (1D barcode) and QR-code (2D barcode) are proposed in this section, and these two codes are adopted as the de facto standard codes in the Japanese mobile market. This study does not discuss the decoding process of the code, because the decoding process is in accordance with the ISO/IEC standard specification and is implemented in the host CPU in our system. The input and output for these algorithms are defined as follows:

- **Input data :** Captured image by the embedded camera in the mobile phone, which consists of YUV color components. This image may draw a lot of noise, blur, uneven lighting, and characters in the neighborhood
- **Output data:** 1D sample scanning (8-bit gray scale line segments) in barcode part for 1D barcode and 2D normalized bi-level (1-cell=1-bit) data for 2D barcode.

3.1 EAN code

EAN code has four width types for black and white bars, and these line directions are always located in parallel. In this algorithm, these parallel black bars are used to recognize the area of the barcode, and we assume the center of the picture includes part of the code area, because the computation cost of searching the barcode area is reduced by this assumption.

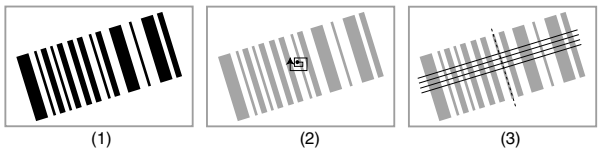

Figure 2: The processing of EAN code reorganization consists of three procedures, (1)defining of threshold as pre-processing, (2) detecting a black bar using spiral search method, and (3)finding the sampled scanning line which is perpendicular to the detected bar in phase (2).

This algorithm consists of the following phases (Figure 2):

- **Pre-processing** The input image from the embedded camera is pre-processed in this phase to define the threshold of the boundary between black and white intensity level, and the color space of the captured image consists of the YUV components, so we picked the Y component for this image processing.
- **A black bar detection** The pre-processed image is scanned using a spiral searching method, which scans in a spiral from the center of the image, to find a black bar.
- **Sampling** The line perpendicular to black bars is calculated using the detected black bar in the previous phase, and the bar patterns are sampled based on the calculated direction.

The result for the decoder program is not a bi-level image but 8-bit gray scale original data, because the black and white bar have four types of width and the decoding result is needed to decide the better threshold for each part in the line segment.

In addition, some characters are written at the bottom of the EAN code, but in our algorithm, these parts can be rejected by capturing the line length of black bars for some sampling lines using the spiral scanning, and these lengths determine the top and bottom boundaries of the code area. Also perepsctive and viewing distraction of ENA code can be ignored from our experimental results.

3.2 QR-code

In this section, the image reorganization and processing of QR-code are described. The QR-code, which was developed by DENSO[3], is known as a kind of 2D barcode. The features of this code symbol are large capacity, small printout size and high speed scanning.

Figure 3 illustrates the structure of QR-code, and this code is comprised of the following patterns: finder pattern, timing pattern, format information, alignment pattern, and data cell. This code is specified by the specification: [2].

In the specification, the example of reorganization process is introduced, which is based on the found pattern and alignment, and this alignment is used for adjustment of code deformation. However, to use this alignment mark for error correction of deformation, the code reading process is needed to find the alignment mark. In our system, the image reorganization and decoding part are separated for recognizing the code at the mobile phone platform in real-time and applying to other 2D code symbol. For the above reasons, we developed the new reorganization scheme.

[3] http://www.qrcode.com

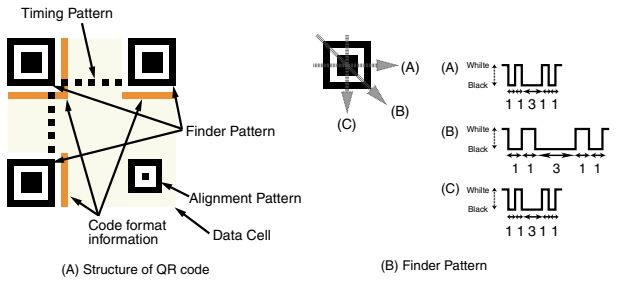

Figure 3: (A) Structure of QR code and (B) Finder pattern.

The overview of this algorithm is shown as follows (Figure 4):

- **Pre-processing** The gray level histogram calculation is adopted,
- **Corner marks detection** Three marked corners are detected using the finder pattern,
- **Fourth corner estimation** The fourth corner is detected using the special algorithm,
- **Inverse perspective transformation** Inverse transformation is adopted based on the obtained corner geometry positions to normalize the size of the code, and
- **Scanning of code** Sample the inside of code and output the normalized bi-level code data to host CPU.

The above scheme can be apply to other kinds of code, because this scheme is just detection of a square area using a key pattern like a finder pattern in QR-code. The details of the algorithm are described in the next sub-sections.

3.2.1 Pre-processing

The input image from the camera interface has the YUV color space components (the luminance and chrominance), and we use the Y component (the luminance), which has 8-bit (256 level) gray scale data, for this image processing. As the pre-processing, three steps of image processing are applied: histogram calculation to define the threshold of black-white boundaries, resizing of original image to reduce the calculation costs in the next phase: recognizing QR-code area, and the filtering for the area dilatation.

In the histogram calculation, nine parts of 60x60 square areas nearby the image center are selected as sampling points to define the threshold value between the black and white luminance levels to convert the bi-level image. The pixel density is sorted for each selected area, and the threshold in this area is calculated as a center value of sorted density (the median luminance). After that, a threshold is defined as the minimum values from the threshold of all selected areas according to experimental results.

After the definition of threshold, the resized image which is scaled down into half for x and y coordinate directions, is used for the original luminance component image to reduce the calculations.

Finally, the filtering, which works for filling the holes (area dilatation), is used to recognize the QR-code area. The minimum density among the original four pixels is assigned to one pixel, and in our implementation, this filtering is also combined with the resizing process.

3.2.2 Code area detection

The filtered and resized image are used for the rough code area findings, and these processes are shown in the following list.

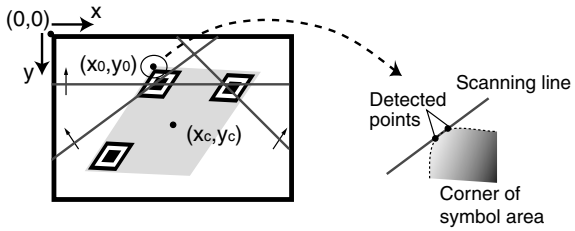

Figure 5: Three scanning lines are used for the corners detection.

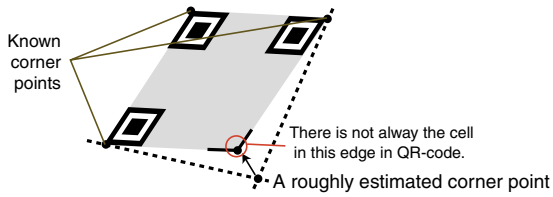

Figure 6: Forth corner detection method.

1. Calculate the position of the gravity center (x_c, y_c) in Figure 5 which is assumed as the center of captured image.
2. Scan a line from outer to inner in eight directions until the line touches the area.
3. If there are two or more pixels on a line (Figure 5 right), their edge points are obtained.
4. After eight direction scanning, we get 16 points at most.
5. The nearest point to $(0,0)$ is one of the targets.
6. Define a vector P as $(x_0, y_0) - (x_c, y_c)$.
7. Calculate the inner product for finding other corner points.

After the above process, the positions of corners are refined in the original size of image as follows.

1. Set a line from (x_0, y_0) to (x_c, y_c).
2. Cross-point is in the outer edge of the mark.
3. Start recursive area growing from the cross-point.
4. Get gravity center of outer mark and this is in the inner mark.
5. Restart recursive area growing from the gravity center.
6. Mark center is defined as the gravity center of inner mark.
7. Corner point is defined as the farthest point from the image center.

This process obtains the three corner points using finder patterns, but the fourth corner point has no found pattern, and also the case where there is no corner cell in the fourth corner exists. Because of these features of QR-code, we introduce the new corner detection algorithm for the fourth corner point reorganization.

1. Set line from known corner points to roughly obtained point (Figure 6).
2. Move the cross-point so that line segments are shown by touching the code area (line attachment method).

The calculated code size and recognized code feature by the above processes can be used for the verification of code specification about the code size (code size is always odd), equally both width and height, and position of alignment mark before the decoding (Figure 7).

3.2.3 Inverse perspective transformation

The input image has a deformed shape because of being captured from the embedded camera device; for example, is Figure 8, and we use the inverse perspective transformation to normalize the code shape. This equation is shown as follows,

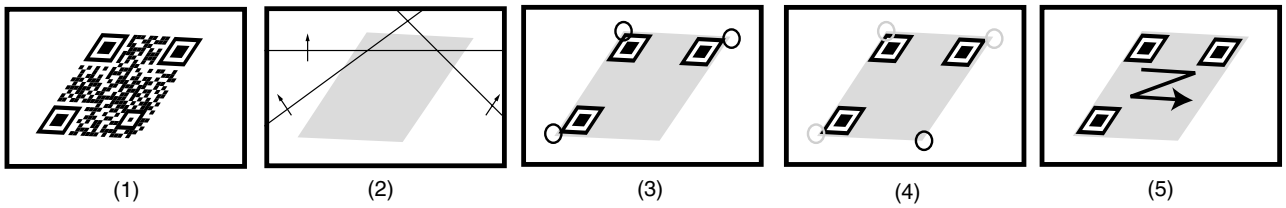

Figure 4: The processing of QR-code detection consists of five procedures, (1)Pre-processing, (2)detecting corners roughly, (3)defining the three marked corners (4) detecting of the fourth corner which has no mark, and (5)scanning of the code to create the size normalized and bi-level code image.

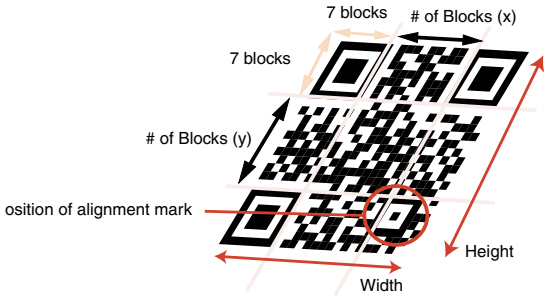

Figure 7: The version is defined from the size of QR-code based on the size of found pattern which has 7x7 cells.

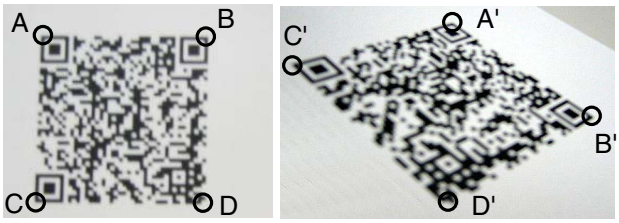

Figure 8: Examples of QR codes which are captured from two different positions by embedded camera in mobile phone.

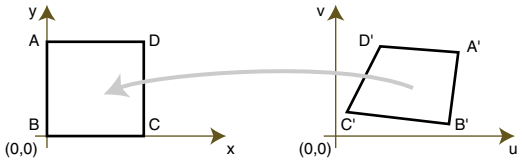

Figure 9: Inverse perspective transformation.

$$u = \frac{c_0x + c_1x + c_2}{c_6x + c_7y + 1} \qquad v = \frac{c_3x + c_4x + c_5}{c_6x + c_7y + 1}$$

where uv coordinate is original image coordinate which is deformed and xy coordinate is the normalized coordinate. In the above equations, coefficients $c_0 \sim c_7$ can be obtained from the following four point pairs (Figure 9),

$$A(x_0,y_0) \Leftrightarrow A'(u_0,v_0), \quad B(x_1,y_1) \Leftrightarrow B'(u_1,v_1),$$
$$C(x_2,y_2) \Leftrightarrow C'(u_2,v_2), \quad D(x_3,y_3) \Leftrightarrow D'(u_3,v_3)$$

3.2.4 Scanning of code

The general QR-code reorganization method (specification mentioned method) is scanning using the clock pattern and alignment pattern, but our algorithm is based on the four corners points, so each cell in the code symbol can be scanned directly by using the inverse perspective transformation. In our method, the calculation cost depends on the size of the code, not the size of image, and this is good for big image size input.

Finally, the scanned image is output as a normalized bi-level image to host application.

3.3 Limitations

The quality performance of our algorithm strongly depends on the accuracy of the symbol corner position, and the threshold value at the boundary between the black and the white part will incur a 1-3 pixel difference, which will be the cause of mis-recognition sometimes, in corner points, from the practical study. Of course, this weak point can be improved by some threshold value retrying. On the other hand, if the threshold value will be better even with a blurred image, our algorithm always scans the code effectively the direct scanning based on the transformation equation (In the cell scanning case, our algorithm always scans one cell as one pixel directory) .

In the practical implementation, the threshold is changed by changing the exposure value (EV) in the camera by the host application.

This algorithm assumes that input image is the skewed code symbol but not curved for all edges. However, this algorithm is applied to the curved code symbol using the recursive approximations shown as Figure 10. The control points are added by the searching based on the obtained edge points, and the position of cells can be calculated based on these edge points by the same transformation scheme.

4 Implementations

This application is implemented in our example hardware architecture which has the following specifications:

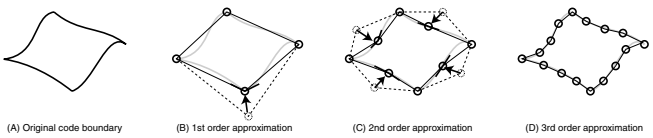

Figure 10: Approximation flow for the curved code symbol.

Table 1: The required memory size shown in the table above does not include the size of input and output image buffer

Memory	Type	EAN code	QR code
Instruction	-	1.5K	5.2K
X memory	Static RAM	0K	0.6K
	Scratch RAM	0.45K	8K
Y memory	Static RAM	0K	1K
	Scratch RAM	9.1K	12.5K

Size[Words]

- **Host CPU** V850 (@60MHz) which can access outer Synchronous DRAM (SDRAM) directly.
- **DSP** SPXK4 core (@100MHz) which has the embedded RAMs, 64KByte for instruction, and two sets of 32KByte for X and Y data memory.
- **Peripherals** Camera interface, LCD controller including window compositing hardware, and so on.

The followings are implementation of algorithm for each phase,

- **Input image** is resized by using hardware, and also filtered.
- **Area estimation and 4th corner estimation** are implemented in DSP.
- **Inverse perspective transformation**
  The original floating point library is developed for this phase, because the DSP has no floating arithmetic unit in this implementation. This original floating point format shows as follows: 8-bit for exponent and 16-bit for fraction in 2's complement to specify the DSP architecture.

The image processing is coded by the full assembler code to execute in real-time, and the DSP uses the local memory only for the calculations. The source data is prepared by Static RAM (SRAM) controller, and this data transforms words in parallel with processing at DSP.

Table 1 shows the memory and the instruction code size for the image processing part, and this indicates the tiny code size for this processing making it capable of DSP implementation in the mobile phone platform. The image sources are stored in the main memory, and the scratch memory is used as a temporary area for DSP calculations.

5 Experimental Results

In this section, the experimental results and performance are described based on the previous implementations.

Figure 11 shows the core performance for EAN/QE code image processing and shows the maximum instruction cycle counts for each condition. 66.7 frames / sec for EAN code and 14.1 frames / sec for QR code image processing performances are obtained by our algorithm and implementation in the following conditions.

- DSP: μPD77210 (Clock Frequency: 100MHz)
- Input image: QVGA (320x240).
- Output data format (MAX):
  Gray scaled 320 pixels image for decoding of 1-D Version 13 EAN code

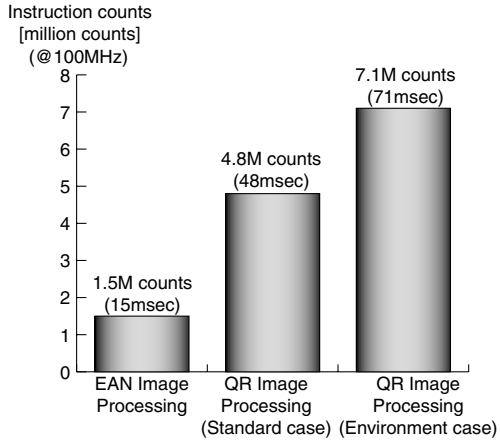

Figure 11: Instruction counts for EAN/QR code image processing core

Binaries 57x57 pixels image for decoding of QR code (version 10)

Figure 12 shows the some examples of experimental results using our algorithm, and shows various situations such as CRT displayed image and label attached into a can. For these general scenes, this algorithm is suitable without a performance reduction; even through they were implemented in the mobile phone platform.

The practical results of calculation time in the different environments are shown in Figure 13. These points plot the error rate after the image processing and processing performance using two kinds of situations indoors, which are in different places and lighting conditions, and this indicates the reorganization rate strongly depends on the environment, especially lighting conditions, because of the recognition accuracy of the boundary between the black and white parts. Also this result shows there is no relationship between the error rate and its image processing performance. This characteristic is important for general usage because the performance drop may be due to the captured image being blurred by hand shaking.

Using this core, the code recognition functions in the marked released mobile phone, can realize 5-10 frames / sec including system overhead such as OS and device subsystems.

6 Conclusion

The new algorithms of EAN code (1D barcode) and QR-code (2D barcode) reorganization are proposed for robustness in practical use and tiny code size, and this is implemented in our mobile application processor architecture which has embedded host CPU, DSP, camera interface, and LCD controller.

In our algorithm, the introduced algorithms are based on the code area finding by four corners detection for 2D barcode and spiral scanning method for 1D barcode using the embedded DSP. The calculation cost is suitable for the current mobile application, and this method is robust for practical situations.

As a result, field testing indicates that this algorithm can be adopted, and the performance, 66.7 frames / sec for EAN code and 14.1 frames / sec for QR code image processing in the environment use, is enough for practical use.

This algorithm can be applied to other 2D barcode recognition tasks, and these are the development of the combined algorithm of

Figure 12: Some experimental results of image processing using our algorithm.

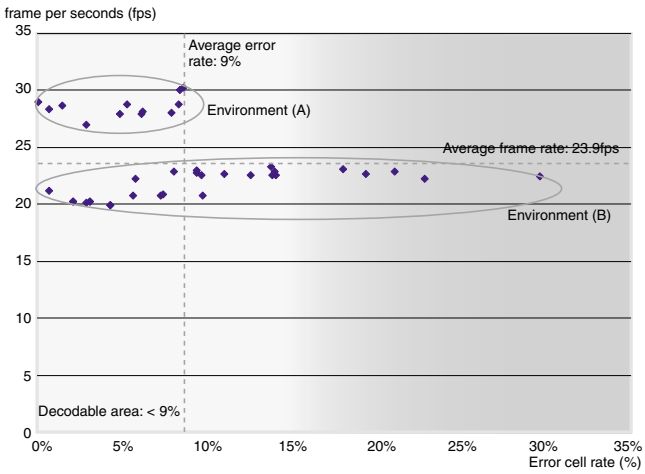

Figure 13: Experimental results of performance using two kinds of situations. Both environments are indoors and the same size of code reorganization and the difference is experiment place only.

any 2D barcode as future work. Also, as the major future work, the threshold deciding method will be developed based on the modeling of the camera device model.

Acknowledgement

We would like to thank all the members of our laboratory and our development team for various advice and support.

References

[1] ISO/IEC 15420:2000. *Information technology – Automatic identification and data capture techniques – Bar code symbology specification – EAN/UPC*, 2000.

[2] ISO/IEC 18004:2000. *Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code*, 2000.

[3] H. Kurosaka, K. Miyamoto, Y. Nagase, H. Ikegami, K. Sato, and M. Otsubo. Mobile Phone Application LSI Design Using C based Design Methodology. Technical Report 4, NEC Corporation, 4 2003.

[4] J. Matas, L. Soh, and J. Kittler. Object recognition using a tag. In *1997 International Conference on Image Processing (ICIP '97) 3-Volume Set-Volume 1*, pages 877–880. IEEE, IEEE Computer Society Press, 10 1997.

[5] R. Muniz, L. Junco, and A. Otero. A Robust Software Barcode Reader Using the Hough Transform. In *Proc. of 1999 International Conference on Information Intelligence and Systems*, pages 313–319. IEEE, IEEE Computer Society Press, 1999.