Camera Barcode Reader with Automatic Localization, Detection of Orientation and Type Classification

ADRIAN BURIAN†, MARKKU VEHVILÄINEN†, JARI KANGAS†
†Devices Research and Development
‡Nokia Research Center
Nokia
Visiokatu 1, 33720 Tampere
FINLAND

Abstract: - Being an area with major industrial interests, many techniques related to barcode information processing exists. There exist a large variety of barcode types and shapes, but the most used ones are the 1D barcodes that have successions of black and white bars of different thickness, and 2D barcodes of square shapes which contain successions of black and white dots (small squares, rectangles, hexagons, etc). It is generally accepted that the conventional way of a Barcode Reader to work is to firstly binarize its input, and then execute the rest of its operations in the binary domain. Not all existing Barcode Readers use the conventional solution, since the binarization can practically be implemented at any level of the applied procedures. In this paper we present our solution for a barcode reader using camera phone devices.

Key-Words: - Barcode, Binarization, Detection, Mesh, Localization, Classification.

1 Introduction
The first patent for a bar code symbology was issued in 1952, namely the US Patent No. 2,612,994. The bar code technology was first used commercially in 1966 [1]. These early symbologies were printed by generating a multiplicity of single width elements of lower reflectance, or "bars", which were separated by elements of higher reflectance, or "spaces." An element is a bar or a space. These symbologies, and many bar code symbologies used today, can be referred to as 1D or linear symbologies because data in a given symbol is decoded along one axis or dimension. When more information is needed to be stored, the elements become dots, or hexagons, or any other form, and they expand into two dimensions, so 2D barcodes. One major advantage of using 2D barcodes over 1D ones is their error correction capability. These symbologies are encoding “data patterns”. Data patterns include not only human readable characters, but also any special function characters that can provide certain functional data.

Bar codes have become quite common tools in every day’s activities. They can be found everywhere: from books and CDs, to advertisements in newspapers. In recent years, the management of goods or the like is generally performed using bar codes. For example, in stores and shops, information such as the type and price of an item of goods is coded into the format of a bar code, and the bar code is printed on the products. Thereafter, checkout is performed by reading the bar code using a bar code reader and this process proceeds in real time. Most of the existing systems were developed for images acquired by a laser scanner. However, as various kinds of camera are getting popular, and as the imaging phones made such a huge impact, the usage of the camera image is getting more important.

Data expressed in the bar-code is read out by a device called a bar-code reader. While in the past, the bar-code reader was a device which irradiated light onto the bar-code and received the light reflected from the stripe patterns composed of the black and white lines, currently other devices become capable of reading barcodes (cameras, fingerprint scanners, RFID devices ...). Basically, we can consider any device capable of reading barcodes to be a Barcode Reader. Then, depending on the characteristics of its input device, the Barcode Reader applies a package of algorithms to correctly retrieve the information coded into the barcode. The camera in mobile terminals can be used to collect information from the environment in many ways. One obvious method is to take images of barcodes, which can contain various types of information about products or services.
2 Problem Formulation

Camera image is more difficult to recognize than the scanner image, because with a camera it is difficult to control the imaging environment as much as we can with the scanner. Even if the user is assumed to take the image carefully, the obtained camera image can have many problems. Firstly, the brightness is not uniform because of uneven lighting or aberration of the camera lens. Secondly, the colour level surface of the camera image is smoother than that of the scanner image. In other words, the edges are not as clear as when using the scanner image, and therefore, the difference in intensity of the captured image content is obscure in many camera images. Thirdly, the captured image is distorted by: the sensor noise added to the captured image, optical blur and vignetting that are due to the optical system of the camera (lenses). These problems are especially significant for the images that are not well focused.

Algorithms and implementations of image reorganization for EAN/QR barcodes in mobile phones were presented in [2]. The used mobile phone system consisted 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 was based on the code area found by four corners detection for 2D barcode and spiral scanning for 1D barcode using the embedded DSP.

The used approach uses closely the EAN/QR barcode characteristics, and therefore lacks generality. The corner detection technique is using some of the graphical characteristics of the QR barcode and it will fail for other 2D barcode types.

The patent from [3] relates to a 2D code reader for reading 2D code which first detects a two dimensional code area, and only then binarizes the input image. This barcode reader is also specifically designed to read only 2D DataMatrix barcodes. We are using also, in one of the proposed variants, as a first step, a region of interest detection, and only after this the binarization follows. But, this patent uses differences between maximum and minimum values of local windows, while we propose to use sample differences on two orthogonal directions, and histograms of these differences. We note that our method has the speed advantage, since the processing of this step can be done on a smaller resolution. Also, the rest of the used procedures are completely different.

In [4], a dataform reader for locating dataform cell edges and methods to determine whether a transition segment represents a valid dataform cell, was introduced. In the patent, “dataforms” include 1D and 2D barcodes, words, numbers, or symbols. The patent deals with means to detect and correct edges. Binarization is not an issue, and it comes more close to [3], than to our solution.

A system and method to binarize the barcode image in the presence of noise is disclosed in [5]. This reference also suggests the usage of the conventional methods for barcode readers. We are not using this method for the binarization step, but a particular case of the method we proposed in [7]. We mention here that the binarization of text characters has different characteristics when compared to barcodes. The text detail restoration is much more important when compared to barcodes.

In [6], a quite general image processing approach suitable to locate, segment and decode the most common true 2D symbols was used. The authors focused their attention on codes like Maxicode and DataMatrix. They start by detecting the region of interest, but using a 3x3 Sobel mask, while we are using a modified 4x4 Prewitt gradient edge detector. From here on, we binarize and the rest of the processing is done in binary domain, while the authors of this paper continue the processing in the grey-level domain, and binarization is done just at the last moment.

We proposed in our earlier paper [7] a new binarization method that applies a local statistical parameter. We called this parameter the min-max parameter, given by

$$T_i = \frac{M_i - \text{Max}_i - \text{Min}_i + 2 \text{p}_{i+1}}{4 \text{Max} \{ \text{Max}_i, \text{Min}_i \}}$$

where Max_i and Min_i are the average of the maximum, respectively minimum values, of the current moving average and the one above it. We note that the correction factor of the average estimate is no longer constant, but it depends both on local details and global ones. The global trend is reflected by the global maximum and minimum values of the moving averages. The local one is given by the usage of next values that are to be used in the next step. We intended to insert with this parameter a primitive edge detector, since we noticed that this could be quite beneficial for the overall binarization process.

Because of the problems with the camera image, a Barcode Reader will face more challenges when its input comes from a camera, rather than a scanner. Our solution presents a new methodology to overcome these problems, and to be able to scan
correctly the barcode even in difficult environmental situations. Moreover, the proposed solution can be implemented for Bayer matrix data or for interpolated images. As a consequence, the proposed Barcode Reader can be incorporated into the image processing chain or it can be implemented as a separate software package that is installed on demand. The proposed algorithms are adaptive such that the user does not have to set the value of any parameter. Different needed parameters and functions are automatically detected and corrected from the input image; therefore, there is no necessity for calibration of the system. Also, the user does not need to know the nature or type of the barcode he is scanning, our Barcode Reader is capable of announcing him if the scanned barcode is outside the range of its capabilities, and it is able to automatically detect the type of the scanned barcode and adapt its computing procedures accordingly. Furthermore, our Barcode Reader can handle automatically both direct and inverse printed barcodes. In fact, everything is done automatically, and all the user has to do is to simply scan any barcode and read the result.

3 Used Solution
This paper relates to devices having an optical data reader. Many forms of handheld devices having an integrated optical data reader are presently available. These devices include optical barcode scanners, some handheld computers, or mobile phones. The object of the present paper is to provide a bar-code reader which is capable of inputting data using a camera device. It relates to a bar code reader which reads out data expressed on a bar code and has functions to correctly reading out the data pattern.

To accomplish this task, the image processing methodology is composed of five main steps, illustrated graphically in figures 1 and 2. We will note by $M1$ the methodology presented in Figure 1, and by $M2$ the methodology presented in Figure 2. $M2$ is using the conventional way to detect and decode a barcode. They can be summarized as follows:

1. Initialization
   1.1. Retrieving the input image.
2. $M1$. Detection of the Region of Interest (ROI)
   2.1. Binarization of the input image
3. $M1$. 1D/2D Classification
   3.1. Detection of Binary ROI
   4.1. Binary 1D/2D Classification

Fig.1. Processing methodology $M1$. 

Fig.2. Processing methodology $M2$. 

Detection and Decoding
5. Detection and Decoding
5.1. If 1D barcode, detect its type, decode the barcode and return the result.
5.2. Correction of ROI area
5.3. Re-sampling and Geometrical Image Correction
5.4. Detection of 2D barcode type
5.5. Compute the scanning mesh
5.6. Decode and return the result to the user

3.1 Implementation
In this section we describe in detail all the procedures and algorithms that have been implemented.

1. Retrieve the image:
This is the first operation of our proposed method. The input image can come from hardware sources or software ones, so from the Bayer matrix image data obtained directly from the CCD sensor, from its EXIF output, from the colour image obtained at the output of the imaging chain or from the JPG saved image obtained after compression. For instance when the Bayer matrix data is used as input, the image obtained from the sensor contains four components: one Red component, two Green components, and one Blue component. If the input image is a color image, it is first transformed into a gray level image. E.g., if the RGB format of the color image is being used, then the gray level is the average value of all Red, Green, and Blue values. In one of our implementations, we have used only one of the Green components and the Red component for processing. The reasons of this are the following: first, the size of one component from the Bayer matrix is one quarter the size of the whole sensor image and this reduces the computational complexity and increases the processing speed. Secondly, the Blue component is the noisiest one, so its absence makes the whole process easier. If the color image is present at the input, it is transformed to a gray scale image by appropriate averaging of the three colour components Red, Green, and Blue. It is also possible to use just one component of the color image instead of the gray level image. We tried using only the Green component, but the results are better if all colours are used. For speed reasons, we have used the following formula to compute the grey level: \((R + 2G + B)/4\).

2. M1. Detection of ROI: For detecting the region of interest a modified 4x4 Prewitt gradient edge detector is being used:

\[
\nabla_x = \begin{bmatrix}
-1 & 0 & 0 & 1 \\
-1 & 0 & 0 & 1 \\
-1 & 0 & 0 & 1 \\
-1 & 0 & 0 & 1 \\
\end{bmatrix}, \quad \nabla_y = \begin{bmatrix}
-1 & -1 & -1 & -1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

We note that we changed the standard operator for reasons of speed increase. This is the only processing that analyzes the whole input image, so it's the slowest from the whole processing chain. The price we pay is that several corrections must follow, but we accepted this price, since these corrections are done just on the detected ROI, and not on the whole image. The increase in speed for mobile phone implementations is noticeable.

The absolute value of the output of the gradient edge detector directly fills two horizontal and vertical histogram buffers, so a dramatic save of memory requirements is obtained (the size of the buffers is given by the size of the input image). After this step, only the histogram information is processed so it has a very high speed. The region of interest is detected from these two histograms. We look for the barcode situated in the central area of the image. If there is more than one barcode in the input image, the one in the centre will be selected.
and decoded. The following procedure is repeated for both horizontal and vertical histograms.

2.1 The maximum histogram value in the central area and its index are detected. Let’s denote them by \( h_{\text{max}} \) and \( i_{\text{max}} \).

2.2 Compute the minimum value of the histogram to the left of the detected maximum, so for histogram indexes between 0 and \( i_{\text{max}} \). Let’s denote this value by \( h_{\text{min}} \).

2.2 Compute the left threshold:

\[
th_{l} = \frac{h_{\text{max}} + 7h_{\text{min}}}{8} + 1.
\]

2.3 Compute in a similar way the right threshold, for indexes bigger than \( i_{\text{max}} \):

\[
th_{r} = \frac{h_{\text{max}} + 7h_{\text{min}}}{8} + 1.
\]

2.4 Starting from \( i_{\text{max}} \) move on both left and right directions, until the histogram is smaller than the corresponding threshold for 3 consecutive values. The detected region is our region of interest.


For binarizing we have used two binarization methods. The first one is a global binarization. This method is fast, but it does not work well with shadows. If the barcode detection/correction fails using this method, the next image will use a particular case of the binarization we proposed in [7].

3. M1. 1D/2D Classification

If one dimension of the ROI is less than 16 times smaller than the other, the decision that this is a 1D barcode is taken.

3. M2. Detection of Binary ROI

At this stage in M2 we are already working in the binary domain. We are using a somehow similar idea like at step 2. M1 but computing the orientation of the edges, quantized on 11 levels for each direction. The orientation is considered to be zero, if there is no binary edge detected in a 4 pixels neighbourhood.

A 4 x 4 processing window is being used. To speed up the processing, if both dimensions of the input image are bigger than 300, a downsampling by 4 is used; this is done by jumping 4 pixels after each window processing in both directions. In each window the contained edge orientation, denoted by \( E_{0} \) is computed. We mention that this method of computation is also completely original, not published anywhere, and not taken from any source. Let us denote the window’s input content is as follows:

\[
\begin{bmatrix}
B_{0} & B_{1} & B_{2} & B_{3} \\
B_{4} & X & X & B_{5} \\
B_{6} & X & X & B_{7} \\
B_{8} & B_{9} & B_{10} & B_{11}
\end{bmatrix}
\]

where we note that we are using only 12 bits out of the 16 present in the window. So, we are interested only in the binary values of \( B_{0} - B_{11} \). The following steps are executed to compute \( E_{0} \):

3. M2.1 Compute \( L_{1} = (B_{4} \text{XNOR} B_{6}) \text{XOR} (B_{5} \text{XNOR} B_{7}) \), where XNOR and XOR are the negate exclusive OR, and exclusive OR logical operations.

3. M2.2. Compute \( L_{2} = (B_{1} \text{XNOR} B_{2}) \text{XOR} (B_{9} \text{XNOR} B_{10}) \).

3. M2.3 If \( L_{1} \) is TRUE and \( L_{2} \) is FALSE Compute \( n_{1} = B_{0} + B_{1} + B_{2} + B_{3} \) and \( n_{2} = B_{8} + B_{9} + B_{10} + B_{11} \). If both \( n_{1} \) and \( n_{2} \) are equal to 0 or 4, \( E_{0} = 0 \) otherwise \( E_{0} = (-1)^{B_{4}} (n_{1} - n_{2}) - 5 \).

3. M2.4. If \( L_{1} \) is FALSE and \( L_{2} \) is TRUE Compute \( n_{1} = B_{0} + B_{4} + B_{6} + B_{8} \) and \( n_{2} = B_{3} + B_{5} + B_{7} + B_{11} \). If both \( n_{1} \) and \( n_{2} \) are equal to 0 or 4, \( E_{0} = 0 \) otherwise \( E_{0} = (-1)^{B_{4}} (n_{1} - n_{2}) + 5 \).

3. M2.5. If \( L_{1} \) is TRUE and \( L_{2} \) is TRUE \( E_{0} = (-1)^{B_{4}} 10 \).

3. M2.6. Otherwise \( E_{0} = 0 \).

We then compute the histograms of the orientation on both dimensions of the image. The Binary ROI is selected as the area with more edges closer to the centre of the image. Then steps 2.1-2.4 are executed but with the computed histograms of the orientation at input.


4. M2. Binary 1D/2D Classification

From the histogram of the orientation computed at step 3. M2 we compute the maximum values on both directions. If they have close values we decide that binary ROI contains a 2D barcode. If one is at least 16 times greater than the other, we decide that the input contains a 1D barcode.

5.1. 1D Barcode Processing
For M1 the detected position of ROI (larger dimension) and for M2 the isolated orientation peak will give the scanning line positions. The scanning lines will be horizontal, if the larger dimension is on X. Eight scanning lines with inclination ±10, ±20, ±30, and ±40 degrees, as well as the horizontal line, will be tested for the presence of 1D barcodes. From the numbers of bars, and from the thickness of the endings, we decide the type of 1D barcode. The decoding result, successful or not, is presented to the user.

5.2. Correct ROI
From this step on, both M1 and M2 exist in the binary domain, so all the processing will be common. The first step is the application of a recursive binary median filter, such that impulse noises are removed from the image. Then the ROI position is corrected by correcting its corner positions inside already detected ROI. This is done by correcting the corners in their local neighbourhoods, such that ROI will contain the whole barcode, including the optically deformed areas. The need for this step comes from the fact that the corners of ROI are not yet the real corners of the barcode, so we have to move them to the correct position.

5.3. Re-sampling and Geometrical Image Correction
After all 4 corners have been corrected the geometrical corrections are done for each of the 4 borders of the new ROI. After the new border and corners are obtained, a new re-sampling is applied such that the obtained corrected ROI to have a rectangle shape.

5.4. Detect 2D Barcode Type
From the specifications of each 2D barcodes type we want to detect, we extract their geometrical characteristics. At this step we will look for specific component patterns, that are used to identify one 2D barcode (e.g., Aztec bar codes have two concentric rectangles in the middle area, DataMatrix has an L shape on borders, QRcode has concentric rectangles, with a filled rectangle the interior rectangle on three of its corners, etc).

5.5. Compute the scanning mesh
We start with the specifications of the detected 2D barcode. The true size of the scanned barcode is then detected by computing the sizes inside the current rectangle ROI, and knowing the possible sizes from specifications. The size from the specifications which is closer to computing size is then chosen to be used.

5.6. Decode and return the result to the user
Finally, the nodes of the mesh will give the resulting bit pattern of the barcode. This bit pattern is checked in order to find the encoded message, exploiting the error correcting mechanisms embedded into the detected bar code specifications. The result is presented to the user.

4 Conclusion
We present in this paper a generalized barcode reader implemented successfully on several Nokia camera phones. Our solution introduces several innovative approaches such as: generalization capabilities, used algorithms and the order of different image processing procedures applied to the image. We are using automatic methods for barcode type detection, orientation detection, and localization.

References: