Robust Recognition of 1-D Barcodes Using Camera Phones

Steffen Wachenfeld, Sebastian Terlunen, Xiaoyi Jiang
Computer Vision and Pattern Recognition Group,
Department of Computer Science, University of Münster, Germany
{*wachi, terlunen, xjiang*}*@uni-muenster.de*

Abstract

In this paper we present an algorithm for the recognition of 1D barcodes using camera phones, which is highly robust regarding the the typical image distortions. We have created a database of barcode images, which covers typical distortions, such as inhomogeneous illumination, reflections, or blurriness due to camera movement. We present results from experiments with over 1,000 images from this database using a Matlab implementation of our algorithm, as well as experiments on the go, where a Symbian C++ implementation running on a camera phone is used to recognize barcodes in daily life situations. The proposed algorithm shows a close to 100% accuracy in real life situations and yields a very good resolution dependent performance on our database, ranging from 90.5% (640×480) *up to 99.2%* (2592×1944)*. The database is freely available for other researchers.*

1. Introduction

Barcodes are ubiquitously used to identify products, goods or deliveries. Devices to read barcodes are allaround, in the form of pen type readers, laser scanners, or LED scanners. Camera-based readers, as a new kind of barcode reader, have recently gained much attention. The interest in camera-based barcode recognition is build on the fact, that numerous mobile devices are already in use, which provide the capability to take images of a fair quality. In combination with Bluetooth or WLAN connectivity, many applications become possible, e.g. an instant barcode-based identification of products and the online retrieval of product information. Such applications allow for the display of warnings for people with allergies, results of product tests or price comparisons in shopping situations.

Efforts concerning the recognition of 1D barcodes using camera phones have already been made. Adelmann et al. [1] have presented two prototypical applications: the display of literature information about scanned books, and the display of ingredient information about scanned food for allergic persons. They did not report recognition performances, but showed proof of concept for new applications. Wang et al. from Nokia [7, 8] have presented an algorithm which seems very fast but very simple as well. They report a recognition rate of 85,6% on an unpublished image database. From its description we consider the algorithm to be much less robust than the algorithm proposed in this paper. Further, Ohbuchi et al. [4] have presented a real-time recognizer for mobile phones, which is assembler-based and kept very simple. Chai and Hock [2] have also presented an algorithm, but without specifying recognition results. Early barcode recognition algorithms (e.g. Muniz et al. [5], and Jospeh and Pavlidis [3]) and parts of the mentioned approaches achieve their goal by applying techniques like Hough transformation, wavelet-based barcode localization, or morphological operations. Using such techniques leads to computationally expensive implementations, which may be not well suited for the use with mobile devices.

In this paper, we present an algorithm to recognize 1D barcodes, which works for the widely used standards UPC-A, EAN-13 and ISBN-13. Our algorithm uses image analysis and pattern recognition methods which rely on knowledge about structure and appearance of 1D barcodes. Given the computational power and the image quality of today's camera phones, our contribution is an algorithm which is both fast and robust.

The details of our algorithm are explained in Section 2. In Section 3 we present results of experiments using our database of over 1,000 images and of experiments on the go, where a Symbian C++ implementation running on a camera phone is used to recognize barcodes in daily life situations. Finally, Section 4 gives a conclusion and outlines future work.

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

2. Recognition algorithm

The goal of our algorithm is to be both fast and robust. As input we expect an image containing a 1D barcode which covers the image center. The barcode does not need to be centered, may be upside down, or may have the usual perspective distortions and rotations (approx. ± 15 degrees) that occur when images are taken by camera phones.

2.1. Preprocessing and binarization

Other approaches start with global smoothing, wavelet-based barcode area location [7] or even morphological operations [2]. We consider this as being too time-consuming and thus use a scanline-based approach.

We assume that a horizontal scanline in the middle of the image will cover the barcode. If this is not the case, or parts of the barcode which lay on the scanline are dirty, occluded or affected by strong reflections, we will detect this in a very early stage and will repeat our algorithm for alternate scanlines above and below.

Without searching for the barcode boundaries, we binarize all pixels on the scanline in a fast manner, starting from a seed point in the middle. For the binarization a dynamic threshold is required, to be robust against dirt, badly printed barcodes or illumination changes. First, we smooth the scanline pixels and compute the luminance value $Y(x) \in [0..1]$ for each position x on the scanline ($Y(x) = 0.299R(x) + 0.587G(x) + 0.114B(x)$). Then we search for local minima and maxima along the scanline, so that neighbored minima and maxima have a luminance difference $\Delta Y \geq 0.01$. The last step before the threshold computation is a pruning step, where we remove unusually dark maxima as well as unusually light minima.

For each position x from the middle of the scanline to the image border, the threshold $t(x)$ for binarization is computed by evaluating a function depending on the last inward seven minima/maxima. We do not consider the outer minima/maxima, as we do not want regions outside the barcode to impact thresholds within the barcode area. Figure 1 shows the profile, minima and maxima, pruned extrema and the resulting threshold. The evaluated function averages the luminance value of the second lowest maxima and of the second highest minima to achieve a good robustness against local errors like dirt or heavy noise.

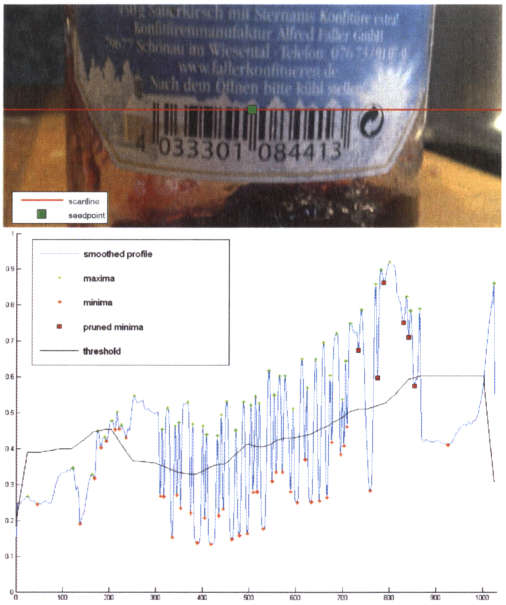

Figure 1. Dynamic threshold for binarization.

2.2. Barcode boundary detection

In the following, we use some knowledge about the UPC-A/EAN-13/ISBN-13 barcode. These barcodes consists of 13 digits. The last digit is a checksum which is computed from the first 12 digits.

The barcode starts by a left-hand guard bar A (black-white-black) and ends with a right-hand guard bar E (black-white-black). Between the guard bars, there are two blocks B and D of 6 encoded digits each, separated by a center bar C (white-black-white-black-white).

A module is the smallest unit. Bars and spaces can cover one to four modules of the same color. Each digit is encoded using seven modules (two bars and two spaces with a total width of 7 modules). The width of a complete EAN-13 barcode is 59 black and white areas ($3+6*4+5+6*4+3$) which consist of 95 modules ($3+6*7+5+6*7+3$).

Two alphabets can be used to encode a digit, the even alphabet or the odd alphabet. While the last 12 digits are directly coded using these two alphabets, the first of the 13 digits is determined by the alphabets that have been used to encode the first six digits. Thus, the first digit is called meta-number or induced digit.

The barcode boundary detection also starts at a seed point in the middle of the profile line. Our starting position is wanted to be a bar-pixel, so either the seed point is black ($Y(x) < t(x)$) or we start with the closest point left or right which is black. From this starting bar we

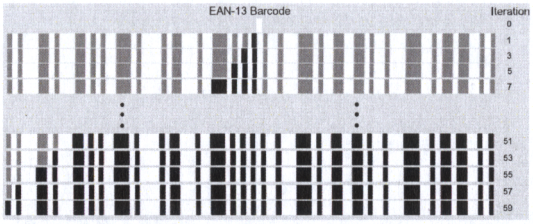

Figure 2. Barcode bounds detection.

successively add spaces and bar to the left and to the right. By adding smaller 'space-bar'-pairs first, we prevent to add non-barcode areas. By remembering the sizes of added bars, we can determine candidates for guard bars. We found the guard bars if the number of bars and spaces between is 59.

2.3. Digit classification

The bars and spaces found in the previous step, have to be classified as digits. As already mentioned, there are two alphabets of 10 digits each, which results in 20 classes c_i. Each digit s is encoded using two black and white areas having a total width of 7 modules.

As preparation for the classification, we generate an image specific prototype for each class. To do this, we investigate the two guard bars on the left and right as well as the center bar. Since we know that their black and white areas have the width of one module each, we can compute the average width of single black and white modules. From the width of single modules we can compute the width of the double, triple and quadruple modules. Please note that in contrast to what could be expected, the binarized area of a double module is not twice as large as the area of a single module. Further, the width of a single black module is not the barcode's total width divided by 95. Due to brighter or darker illumination the widths of single white w_{w1} and black w_{b1} modules may differ by $\Delta_{wb1} = w_{w1} - w_{b1}$. This has to be taken into account for the computation of the widths of double, triple and quadruple modules and requires to separately determine the widths of black and white single modules.

Based on the widths of bars and spaces in the guard and center bars, we compute a reference pattern r_k for each class c_k. The digits s from the barcode are then presented to a distance based classifier which assigns normalized similarity values $p(c_k, s)$ for all classes c_k.

Since each digit is encoded by four black and white areas, the pattern r can be represented as 4-tupels $r \in \mathbb{R}^4$. The similarity $p(c_k, s)$ is based on the squared distance $d(r_k, r_s)$ between the corresponding pattern r_k

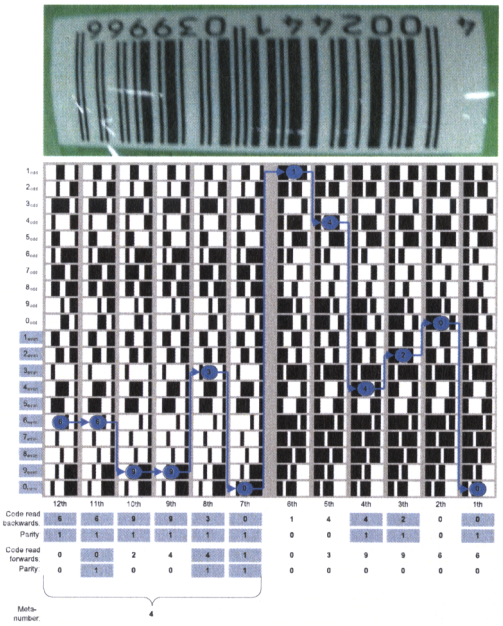

Figure 3. Code hypothesis as path.

and r_s:

$$p'(c_k, s) = 1 - \frac{d(r_k, r_s)}{\max_i(d(r_i, r_s))} \quad (1)$$

$$p(c_k, s) = \frac{p'(c_k, s)}{\sum_{i=1}^{20} p'(c_i, s)} \quad (2)$$

2.4. Search for the most similar code

We now have a similarity for each digit s to each class c_k. By combining the results of the twelve encoded digits $s^{(1)}, s^{(2)}, \ldots, s^{(12)}$ we can successively generate code hypotheses $(m, c^{(1)}, c^{(2)}, \ldots, c^{(12)})$. m is the induced meta number and encoded by the choice of alphabets used to encode digits $s_7, \ldots, s_{13}$ (see Figure 3).

If we consider the similarity values of the digits to be independent of each other and to be probability-like, we can consider the probability of a hypothesis to be:

$$p(c^{(1)}, \ldots, c^{(12)} | s^{(1)}, \ldots, s^{(12)}) = \prod_{i=1}^{12} p(c^{(i)} | s^{(i)})^{1/12}$$

Starting with the code hypothesis that consists of the most similar classes for each digit, we successively investigate hypotheses with decreasing probability. For images of good quality the first hypothesis is the correct one. In case of strong distortions the first hypotheses

may be wrong. Digit s_{13} is a checksum, which allows to detect and to reject wrong hypotheses.

3. Experimental results

Here, we distinguish between two kinds of experiments. First, experiments based on a database, where a Matlab based algorithm is applied to a fixed number of images, taken by a camera phone. And second, experiments on the go, where a Symbian C++ implementation running on a camera phone is used to recognize barcodes in daily life situations.

3.1. Experiments on our barcode database

We took over 1,000 images of barcodes using a Nokia N95 camera phone and stored them in a database which is freely available to other researchers [6]. Using this fixed set of images, which includes heavily distorted, shaky and out-of-focus pictures, we developed our algorithm using Matlab. To investigate the impact of resolution on the recognition performance, the images were taken in the highes supported resolution (2592×1944) and scaled down to typical resolutions of camera phones (1024×768 and 640×480). The recognition performance of our Matlab implementation is 90.5% at 640×480, 93.7% at 1024×768, and 99.2% at 2592×1944 pixels.

The execution time changes with resolution, image quality, and with the varying time for the generation of hypotheses. For the recognition of one barcode, our Matlab implementation needs between 25ms (640×480) and 50ms (2592×1944) on a 1,7 GHz single-core notebook.

3.2. Experiments on the go

To perform experiments on the go, we implemented our algorithm in Symbian C++. The execution time of our current implementation for Symbian OS is between 50ms (640×480) and 70ms (1024×768) on a Nokia N95 camera phone. Our code is not yet optimized, but the recognition speed is already sufficient for real-time recognition on a video stream with about 20fps.

We used the camera phone to recognize barcodes in daily life situations. At a resolution of 1024×768 pixels, we took pictures of 150 barcodes and all barcodes were recognized correctly without using the macro mode. One barcode was recognized at the second try, as the first image was totally out of focus. All images from the experiments on the go are collected, stored, and can be found on our website [6].

4. Conclusion and future work

In this paper we presented an algorithm for the recognition of 1-D barcodes using camera phones. We have performed experiments on a large database using a Matlab implementation and experiments on the go using a Symbian C++ implementation on a Nokia camera phone. Our results show, that our algorithm is very robust and moreover very fast.

Our next step is to use the network connectivity of current camera phones or PDAs to allow for useful applications. Further, we will extend our database of barcodes taken with camera phones and PDAs. Based on our database, we want to compare approaches of other researchers. Perhaps a barcode reading competition can be realized in the short future.

References

[1] R. Adelmann, M. Langheinrich, C. Flörkemeier: Toolkit for Bar Code Recognition and Resolving on Camera Phones – Jump Starting the Internet of Things. Workshop on Mobile and Embedded Interactive Systems (MEIS'06) at Informatik, GI LNI, 2006.

[2] D. Chai, F. Hock: Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. 5th Int. Conf. on Information, Communications and Signal Processing, 1595–1599, 2005.

[3] E. Joseph, T. Pavlidis: Bar Code Waveform Recognition Using Peak Locations. IEEE Trans. on PAMI, 16(6):630–640, 1998.

[4] E. Ohbuchi, H. Hanaizumi, L. A. Hock: Barcode Readers Using the Camera Device in Mobile Phones. In Proc. of the Int. Conf. on Cyberworlds, 260–265, 2004.

[5] R. Muniz, L. Junco, A. Otero: A Robust Software Barcode Reader Using the Hough Transform. In Proc. of the Int. Conf. on Information Intelligence and Systems, 313–319, 1999.

[6] S. Wachenfeld, S. Terlunen, X. Jiang: http://cvpr.uni-muenster.de/research/barcode

[7] K. Wang, Y. Zou, H. Wang: Bar Code Reading from Images Captured by Camera Phones. 2nd Int. Conf. on Mobile Technology, Applications and Systems, 6–12, 2005.

[8] K. Wang, Y. Zou, H. Wang: 1D Bar Code Reading on Camera Phones. Int. Journal of Image and Graphics, 7(3):529–550, 2007.