Robust 1D Barcode Recognition on Mobile Devices

Johann C. Rocholl, Sebastian Klenk, Gunther Heidemann
Intelligent Systems Department, Stuttgart University, Universitätsstrasse 38, 70569 Stuttgart
ais@informatik.uni-stuttgart.de

Abstract

In the following we will describe a novel method for decoding linear barcodes from blurry camera images. Our goal was to develop a algorithm that can be used on mobile devices to recognize product numbers from EAN or UPC barcodes.

1. Introduction

This paper describes a novel method for decoding linear barcodes from blurry camera images. The algorithm can be used on mobile devices to recognize EAN or UPC barcodes and retrieve information from the Internet without input from the user.

Conventional decoders for linear barcodes are based on detecting the edges between bars and spaces. However, the specific locations of these edges may change or become undetectable if the input is very blurry. This is a problem for mobile devices with fixed-focus lenses. Their cameras are not designed to focus correctly in the macro range, which is required for capturing barcodes.

The proposed algorithm locates the barcode in the camera image and extracts a scan line of brightness values. It simulates the blurry barcode according to a mathematical model and chooses digits for which the simulation best approximates the camera input.

A prototype was implemented to recognize UPC-A and EAN-13 barcodes on the Apple iPhone and on the MacBook, using their built-in cameras. The decoder was tested with several hundred images from different cameras. The proposed method correctly recognizes a high percentage of blurry barcode images. The performance of the prototype is also compared to four different existing decoders for linear barcodes, with good results.

Challenges Mobile devices have limited processing power and memory, because of size constraints and battery usage. The iPhone for example has an ARM processor running with a clock rate of 400 MHz and 128MB of system RAM [2]. This seems sufficient for simple computer vision tasks, but not for computational intensive algorithms.

Figure 1. Image of ISBN-13 barcode, captured with the iPhone

Many cameras used in mobile devices, including the iPhone, have a fixed focus lens. They are not designed for macro photos and produce blurry pictures if the distance between camera and object is less than 20cm. EAN-13 barcodes are usually printed less than 5cm wide. If the camera is distant enough to avoid macro blur, the barcode will appear too small and the resolution will be too low for decoding.

Figure 1 shows an example of an ISBN-13 barcode that was captured with the iPhone's built-in camera. The camera blur is severe: bars and spaces are blurred together, forming ranges of varying brightness. The edges between bars and spaces cannot be detected. The image also contains some perspective distortion because the camera was not exactly in front of the center of the barcode.

Related work Working with highly blurred images, Esedoglu et. al. and Wittman et. al. present approaches to barcode deblurring by blind deconvolution [1, 7]. These are very processing intense and therefore infeasibly expensive on mobile hardware. Wang et. al. present a method based on wavelet feature extraction and statistical recognition [6, 5]. We try to avoid the statistical approach and rather estimate the digits directly.

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.664

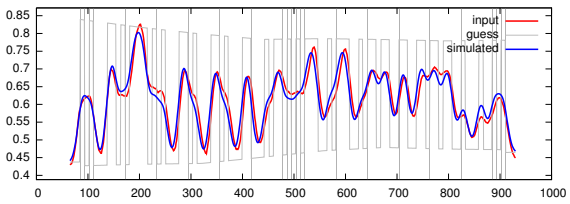

Figure 2. Input and output of the blurry barcode simulation

2 Locating the barcode

The first step for locating the barcode is to find a straight line that intersects all bars. It does not have to be perpendicular to the bars, but it must go across the entire barcode.

One possible solution is to display this line through the middle of a live camera preview and require the user to move the camera and/or target until the line intersects all barcode stripes. This approach has two advantages. First and foremost, the user experience benefits from the visual clue and live feedback because it is immediately obvious how the barcode should be captured.

3 Blurry barcode model

In order to guess barcode digits, the blurry barcode must be simulated according to a mathematical model. The results of the simulation can then be compared to the camera input to find matching digits. This section describes the model and the parameters of the blurry barcode simulation.

Continuous scan line After the barcode has been located, processing can be restricted to monochrome brightness values on a scan line across the barcode area. The region of interest ($X_{\mathrm{ROI}} = [l_{\mathrm{ROI}}, r_{\mathrm{ROI}}]$) is an interval from one quiet zone to the other. Its endpoints are denoted by l_{ROI} and r_{ROI} respectively. For example, $l_{\mathrm{ROI}} = 65$ and $r_{\mathrm{ROI}} = 930$ in figure 2.

The waveform of brightness values on the scan line can be expressed as a continuous function over the region of interest. Several similar functions are introduced throughout this chapter. They all have the same domain and codomain, and are denoted by f with a subscript. Here the subscript "cam" stands for the camera input. It is shown as the red curve in figure 2.

$$f_{\mathrm{cam}} : X_{\mathrm{ROI}} \to [0, 1]$$

Brightness is expressed in the range from zero to one, with 0.0 designating bright white (spaces) and 1.0 designating black (bars).

Perspective projection If the camera axis is not orthogonal to the barcode surface, one side of the barcode may be closer to the camera and thus appear larger. This results in horizontal offset for the positions of the bars and digits, because the bars appear wider on the near side and narrower on the far side.

We model the perspective projection in a Cartesian coordinate system with the origin at the middle of the barcode The projection plane is the x axis. For the purpose of this subsection, the barcode area is normalized so that $x = -1$ indicates the left side and $x = 1$ indicates the right side of the barcode.

Let d denote the distance between the camera and the center of the barcode, or more precisely the ratio between the camera distance and half the width of the barcode. The value of d is unchanged if both the camera distance and the size of the barcode are doubled. The camera is placed on the y axis at $y = -d$.

Next we define a parameter $u \in \mathbf{R}$ that is linear across the barcode and goes from $u = -1$ at the left side of the barcode to $u = 1$ at the right side. Using the angle α between the barcode and the projection plane, the coordinates of the interior points of the barcode can be expressed like this:

$$x_u = u \cos\alpha \qquad y_u = u \sin\alpha$$

Now we can formulate the camera ray as a straight line through the camera and a barcode point. The x-intercept of this line gives the projection point $p(u)$. It is the abscissa of the intersection of the camera ray and the projection plane.

$$y = \frac{d + u\sin\alpha}{u\cos\alpha}x - d \qquad p(u) = \frac{d \cdot u\cos\alpha}{d + u\sin\alpha}$$

To use this projection for estimating the locations of bits between the edges of the barcode l and r, we need a normalized projection function s that satisfies $s(0) = l$ and $s(95) = r$. The following definition achieves that by scaling the function argument to the interval $[-1, 1]$ and the result to $[l, r]$:

$$s(i) = l + (r - l)\frac{p(-1 + \frac{2}{95}i) - p(-1)}{p(1) - p(-1)}$$

Non-uniform illumination The background brightness and contrast may change across the region of interest. This can be modeled by two continuous functions for the white and black level across the scan line, called

f_{white} and f_{black} respectively. The barcode guess (before blurring) alternates between the values of these two functions, according to the bit pattern.

$$f_{\text{guess}}(x) = \begin{cases} f_{\text{black}}(x) & \text{for high bits} \\ f_{\text{white}}(x) & \text{for low bits} \end{cases}$$

A good initial estimate for the white level is a straight line that passes through the brightness values of the quiet zones on the left and right side of the barcode. For the black level, the current implementation uses a constant (horizontal) line at the 95th percentile of brightness values in the barcode area.

Gaussian blur Typical camera blur can be simulated by convolution with a Gaussian kernel.

The camera blur can be simulated by a convolution of the guessed bit pattern with a Gaussian kernel.

$$f_{\text{sim}} = G * f_{\text{guess}} \quad \text{with} \quad G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Residual For each digit, the difference is computed between the result of the simulation and the brightness values from the camera input, using the sum of squares:

$$e_i = \int_{l_i}^{r_i} (f_{\text{sim}}(x) - f_{\text{cam}}(x))^2 dx$$

The values l_i and r_i are the estimated locations of the edges between digits, based on the function s that models perspective projection.

4 Decoding

The traditional method for decoding linear barcodes is to find the edge locations between bars and spaces. This approach fails when the blurring is severe, i.e. when the standard deviation of the blurring kernel is greater than the module size (the width of the thinnest bars or spaces). In this case, the separate peaks and valleys in the gray value curve disappear. The input consists of varying shades of gray, rather than black bars and white spaces. Even if the inverse of the blurring kernel could be found, the presence of noise makes it impossible to derive the original barcode from the blurry picture. Because the edge locations cannot be determined reliably, a different approach is required for robust decoding.

This work proposes to estimate the boundaries of the 12 digits in the barcode, and to guess each digit separately, as there are only 10 possible choices for each digit in a UPC barcode. EAN-13 uses two possible codes for the third through seventh digit to specify the extra first digit, so there are 20 possible choices. But many invalid guesses can be eliminated because there are only 10 valid combinations, see [4].

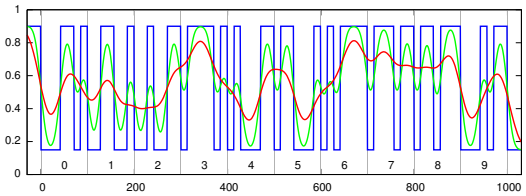

Figure 3. One possible code with scan line and blurry barcode model

4.1 Guessing digits

After the parameters of the barcode model have been estimated, we can attempt to guess each of the digits by setting the corresponding seven bits, blurring with a Gaussian kernel, and comparing the result with the input.

Figure 3 shows one possible code for the digits from 0 to 9. The blue graph is the bit pattern, showing the seven code bits for each digit. The module size (width of one bit) is roughly 14 samples in these graphs because each digit occupies 100 samples.

The green and red curves in figure 3 are the results of Gaussian blurring with a standard deviation parameter of $\sigma = 10$ samples and $\sigma = 20$ samples, respectively. This demonstrates the effects of blurring: if the standard deviation is smaller than the module size, the distinct peaks and valleys are preserved, but they are joined together for larger amounts of blur. The blurring requires additional input on the left and right of each digit, half the width of the Gaussian kernel. Some of these bits may be unknown because those neighboring digits have not been guessed yet. But the left, right and middle guard bars are known, as well as the first and last bit in each digit. The six digits on the left side of the middle guard all start with a space (white) and end with a bar (black), and conversely on the right side. The inner five bits can be set to gray, as the average between the black and white brightness levels. It is even better (but less obvious when plotted) to copy the input gray values as initial guesses for the inner five bits. For iterative adjustment, each additional pass can use the bits that were guessed in the previous pass.

Checksum The last digit of the EAN-13 (and thus also UPC-A) is a check digit that can be used to detect when a barcode was decoded incorrectly. The check-

sum is calculated as a weighted sum of all other digits modulo 10, with different weights for the digits at odd and even positions.

$$d_{13} \equiv 10 - \sum_{i=1}^{6}(3d_{2i-1} + d_{2i}) \mod 10$$

If only one digit is incorrect, there will be a checksum mismatch. If several digits are incorrect, it is possible that the checksum will report a false match because the separate errors may cancel each other out.

Codes A and B The limited number of choices for the first digit of the EAN-13 barcode can also be used to reject invalid combinations of A and B codes for the digits d_3 to d_7. For example, there is no case where d_3 to d_7 would all use the B code, so that result would have to be incorrect. Some of the digits would need to be changed to generate a valid result.

5 Test results

To evaluate the reliability of the proposed method, it was tested with barcode images from several different cameras. The prototype is also compared to four existing decoders with a smaller selection of test images.

Test suite We collected a test suite of 466 images. It includes barcodes of grocery and retail products, books, CDs, and DVDs. Table 1 shows test results for the prototype decoder on five sets of test images. The first two sets were captured with an iPhone using the built-in camera. For the second set, the sharpness of the images was improved by a Griffin Clarifi protective case with macro lens. The third set of images was taken with a Sony Ericsson T650i camera phone with auto-focus lens. The focus mode on the phone was set to "macro" for most of the images in this set, but the automatic focus was not always accurate, so there are also several blurry images.

Image source	Total	Correct	Success
iPhone	112	66	58.9%
iPhone, macro lns.	113	93	82.3%
Sony Ericsson T650i	121	96	79.3%
PDAbar correct	76	64	84.2%
PDAbar not found	44	21	47.7%
All test images	466	340	73.0%

Table 1. Test results for five sets of test images

The last two sets of test images were provided by the developers of the PDAbar library [3]. Their decoder correctly recognized the barcodes in the "correct" set but rejected the images in the other set because they are out of focus or contain reflections.

Comparison with other solutions A smaller set of 60 images was selected from the large test suite. It contains images of 20 UPC-A barcodes, 20 EAN-13 barcodes, and 20 ISBN-13 barcodes, some of which are accompanied by an EAN-5 barcode for the suggested retail price. Each of these 60 images was processed with the thesis prototype and four different existing decoders. The test images cover a wide variety of problematic test cases, including non-uniform illumination, perspective distortion and camera blur. The detection results between 80% and 95% indicate a clear advantage of our prototype as compared to PDAbar (up to 60%) and the Delicious Library[1] ($65\% - 80\%$).

6 Discussion

As the prototype was developed to cope with highly distorted images, the successful decoding rates in the comparison indicate that the proposed method can correctly recognize barcodes with higher levels of blur and perspective distortion than existing solutions. Despite these encouraging results there are still some tasks left. Especially the detection of clear but otherwise transformed barcodes (for example rotated images) is an important requirement for an everyday solution.

References

[1] S. Esedoglu. Blind deconvolution of bar code signals. *Inverse Problems*, 20:121–135, Feb. 2004.

[2] C. Hockenberry. What the iPhone specs don't tell you... furbo.org, Aug. 2007. Last checked Apr. 11, 2009.

[3] A. Kritzner. PDAbar-Bibliothek. Logic Way GmbH, Schwerin, 2008. Last checked Apr. 14, 2009.

[4] T. Pavlidis, J. Swartz, and Y. P. Wang. Fundamentals of bar code information theory. *Computer*, 23(4):74–86, 1990.

[5] K.-Q. Wang, Y.-M. Zou, and H. Wang. Bar code reading from images captured by camera phones. *IEE Conference Publications*, 2005(CP496):42, Nov. 2005.

[6] K.-Q. Wang, Y.-M. Zou, and H. Wang. 1D bar code reading on camera phones. *Int. J. Image Graphics*, 7(3):529–550, 2007.

[7] T. Wittman. Lost in the supermarket: Decoding blurry barcodes. *SIAM News*, 37(7), Sept. 2004.

[1] Delicious Library,Delicious Monster `http://www.delcicious-monster.com/`