Reading 1-D Barcodes with Mobile Phones Using Deformable Templates

Orazio Gallo, *Student Member, IEEE*, and Roberto Manduchi.

***Abstract*— Camera cellphones have become ubiquitous, thus opening a plethora of opportunities for mobile vision applications. For instance, they can enable users to access reviews or price comparisons for a product from a picture of its barcode while still in the store. Barcode reading needs to be robust to challenging conditions such as blur, noise, low resolution, or low quality camera lenses, all of which are extremely common. Surprisingly, even state-of-the-art barcode reading algorithms fail when some of these factors come into play. One reason resides in the early-commitment strategy that virtually all existing algorithms adopt: the image is first binarized and then only the binary data is processed. We propose a new approach to barcode decoding that bypasses binarization. Our technique relies on deformable templates and exploits all the gray level information of each pixel. Due to our parametrization of these templates, we can efficiently perform maximum likelihood estimation independently on each digit and enforce spatial coherence in a subsequent step. We show by way of experiments on challenging UPC-A barcode images from five different databases that our approach outperforms competing algorithms. Implemented on a Nokia N95 phone, our algorithm can localize and decode a barcode on a VGA image (640×480, JPEG compressed) in an average time of 400-500 ms.**

***Index Terms*— Barcodes, UPC-A, Mobile Devices, Deformable Templates.**

I. Introduction

TODAY, virtually every item on the market is labeled with at least one form of barcode, generally a flavor of either the EAN or the UPC standards. The success of barcode technology for identification, tracking, and inventory derives from its ability to encode information in a compact fashion with low associated cost.

Barcode reading via dedicated scanners is a mature technology. Commercial laser-based, hand-held barcode scanners achieve robust reading with a reasonable price tag. Recently, there has been growing interest in accessing barcodes also with regular cellphones, without the need for a dedicated device. Indeed, a number of cellphone apps have appeared that provide access via barcode reading to the full characteristics of and user reviews for a product found at a store.

Unfortunately, images taken by cellphone cameras are often of low quality. Many cellphone cameras on the market are equipped with low-grade lenses, generally lacking focusing capability, which often produce blurred images. Few cellphones have a flash and, therefore, motion blur and noise can be expected with low ambient light. All of these factors, possibly combined with low image resolution, make barcode reading difficult in certain situations. Indeed, all existing image-based barcode readers have limited performance when it comes to images taken in difficult light conditions, or when the camera is not close enough to the barcode. In order to improve accuracy, barcode reading apps usually prompt the user to precisely position the camera to ensure that the barcode covers as much of the frame as possible. This operation can be somewhat bothersome, as it requires a certain amount of interaction with the user, who needs to frame the barcode correctly using the viewfinder.

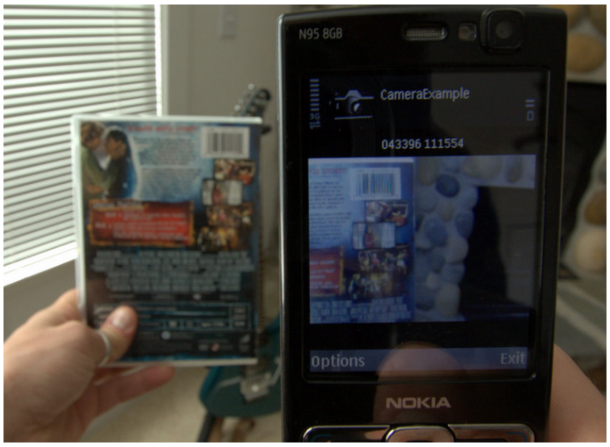


Fig. 1. Our algorithm is able to decode a barcode without requiring the user to precisely frame it within the viewfinder. The result is shown on the screen of the phone above the captured picture, in this case 043396 111554.

This paper presents a new algorithm for 1-D barcode reading that produces excellent results even for images that are blurred, noisy, and with low resolution. A quantitative comparison based on existing and new barcode image data sets shows that our technique outperforms other state-of-the-art software and other reported results. The execution time on a Nokia N95 running Symbian OS is 400-500 ms on VGA (640 × 480), JPEG compressed, still images (this is the wall-clock time from the moment the analysis of the frame begins).

The main novelty of our approach is that it never binarizes the image. Virtually any existing algorithm for barcode reading performs some sort of binarization of the input brightness data. We argue that this early-commitment operation translates into unrecoverable information loss, which makes the reader susceptible to noise, blur, and low resolution. This is especially the case for low-resolution images, where binarization errors may have catastrophic effects. For example, Fig. 2 shows a challenging barcode which would be hardly read by a binarization-based approach: any binarization strategy, even one based on an adaptive threshold, would not allow for the extraction of some of the bars.

In contrast to previous approaches, our algorithm uses the full gray-level information throughout its most critical stages. We employ a particular form of deformable template matching, that produces robust results even in difficult situations, such as the one

Orazio Gallo is with the University of California, Santa Cruz.
Roberto Manduchi is with the University of California, Santa Cruz.

shown in Fig. 2. In such cases, our approach implicitly enables 'soft' assignment of each pixel to either black or white via the pixel likelihood function in Eq. (9)-(10). Pixels with very high or very low brightness values contribute more than pixels with intermediate values to the digit likelihood function in Eq. (7). If, due to noise or blur, a few pixels have intermediate gray values as in Fig. 2, the ambiguity is generally resolved by the barcode-specific structural priors $\mathcal{M}$. We shift and scale the archetypal models of each individual barcode digit to match the measured brightness profile in a maximum likelihood framework. Although deformable template matching usually requires costly optimization techniques, we prove that in the case of these patterns, matching can be obtained *exactly* with a simple discrete search. In addition, we propose an optimization procedure to enforce spatial coherence of the individual digits found by deformable template matching.

Our decoding algorithm requires that the barcode has been localized with fairly good precision (within twice the width of the *base width*, that is, the smallest possible bar in the barcode). This operation is facilitated by the fact that a barcode is bordered to the side by a *quiet* (white) area whose size is prescribed by the standard. We propose a simple and fast algorithm for localization that assumes that the bars are approximately vertical. It should be understood that the main contribution of this work, and the more intellectually original one, is the barcode decoding algorithm. The localization algorithm has no pretense of optimality but works reasonably well in our tests, better so than other published algorithms that we also experimented with.

This paper is organized as follows. Sec II describes the previous work on image-based barcode localization and decoding. Sec. III describes in detail the proposed algorithm. Sec. IV-A reports a thorough performance analysis of the algorithm, implemented in Matlab and tested on different data sets. Sec. IV-B describes the Symbian implementation of the algorithm.

II. Previous work

Barcode reading has been studied and optimized for decades and it now represents a well established industrial standard. (Pavlidis *et al.* provide an interesting analysis of this technology from an information theory standpoint [9].) Until recently, however, barcode reading was performed almost exclusively with dedicated hardware. Despite the rapid growth of interest in camera-based readers, most of the challenges posed by this new approach are yet to be solved.

Commercial scanners, such as those used in supermarkets, shine a stripe of pulsed light on the code and measure the intensity of its reflection; the use of active illumination makes them virtually insensitive to changes of ambient illumination. Additionally, their design often requires the codes to be fairly close to the scanner. In general, these dedicated devices produce a high quality signal that allows for robust barcode reading. On the other hand, reading barcodes with cameras presents new challenges. In particular, the image may be of poor quality, due to noise and possibly low contrast.

The first step for a barcode reader is the localization of the barcode in the image. Most existing applications require the user to move the camera towards the barcode so as to maximize its apparent width in the image. Although beneficial to resolution, this procedure can be bothersome for the user. Our system lets the user take a snapshot of the barcode once it is at a reasonable

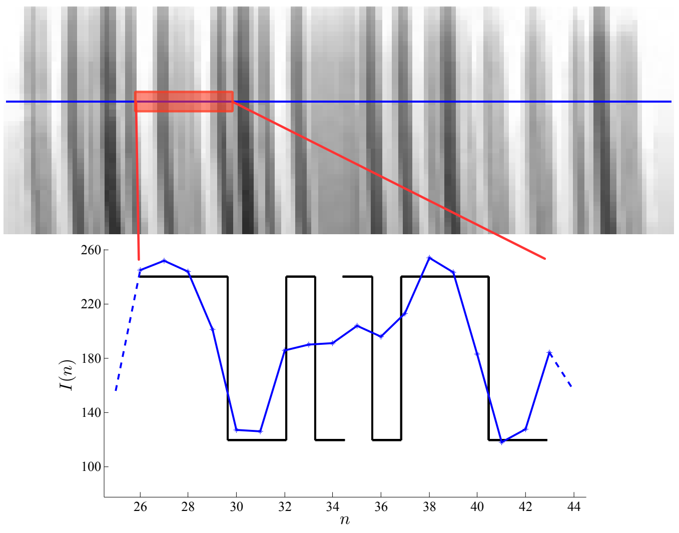


Fig. 2. A challenging barcode image that is correctly decoded by our algorithm. The intensity profile from the segment highlighted in red on the blue scanline is shown in the plot, where the black lines represent the symbols as output by our algorithm. Note how blur and low resolution affect the intensity profile. A system that binarizes the intensity would be hard-pressed to detect the correct pattern. For example, consider the segment between pixels 32 and 38 representing a sequence of two black and two white bars. The observed brightness has an almost flat profile which is nearly impossible to binarize. The barcode shown was automatically extracted from a 1152×864, JPEG image and its total width was 114 pixels.

distance from the camera; it then proceeds to identify the location of the endpoints of the barcode in the image. Once the barcode has been localized, decoding takes place. Keeping localization and decoding distinct allows for a higher computational efficiency. Existing approaches for barcode localization apply to the binarized image methods based on Hough transforms [6], edge orientation histograms [14], morphological operators [4], or wavelet transforms [13]. Other approaches assume that the center of the image falls within the barcode area [7], [12], thus greatly simplifying the problem: Ohbuchi *et al.* reduce the problem to one of finding the angular direction of the bars [7], while Wachenfeld *et al.* extract a scanline thought the center of the image and binarize it with an adaptive threshold; the endpoints are localized by counting the number of bars detected [12].

Decoding can be performed by simply finding the sequence of digits that best explains one or more binarized scanlines. Chai and Hock use a single scanline binarized using the average gray level as a threshold [4] whereas Wachenfeld *et al.* use a simple adaptive thresholding [12]. Adelmann *et al.* use multiple scanlines with different thresholds and a voting scheme [3]. These approaches report high accuracy, but unfortunately do not present comparative tests nor make their data sets public. Krešić-Jurić *et al.* find potential edges by computing brightness derivatives of the scanline and then use hidden Markov models to remove false positives [5]. Their method compares favorably with previous approaches although it was only implemented on laser-based barcode scanners. Tekin and Coughlan propose an elegant Bayesian framework for barcode decoding [10]. Their approach aims to find the edges of the bars in a fashion similar to Krešić-Jurić *et al.* [5] while allowing for undetected peaks.

Early-commitment approaches, whether based on binarization or edge detection, are efficient from a computational standpoint, but rely heavily on the good quality of the input image. Unfortunately, binarization is very sensitive to noise and blur, in particular when the features to be detected, such as the narrow bars of a barcode, are smaller in size than a couple of pixels. The same can be said for edge extraction. In contrast, *our decoding algorithm never performs a binarization or an edge extraction operation*. As a result, our approach is capable of decoding 1-D barcodes from noisy pictures even in the presence of motion blur or lack focus. Additionally, our method also proves effective on highly compressed pictures. In order to allow other researchers to compare their results against ours, we provide a publicly accessible barcode image data set [2].

III. The barcode reader

Given an image containing a barcode, two distinct operations are needed for accessing the information contained in the barcode: *localization* and *decoding*. Localization typically relies on the strong textural content of the barcode, without the need to exactly measure and interpret the width distribution of the bars. Decoding can be performed on one or more scanlines. Throughout this paper we make use of the following definitions (see the Appendix for a more complete description of the terminology). A UPC-A barcode encodes 12 *symbols* (each representing a number 0 – 9). Each of the consecutive, non-overlapping segments which encode the symbols are called *digits segments* or simply *digits*. For instance, in the barcode shown in Fig. 1, the symbol encoded in the second digit is '4'. Note that each symbol is encoded by two white and two black bars.

A. Barcode localization

The localization algorithm provides the decoding algorithm with a scanline, where the barcode's endpoints have been localized as accurately as possible. In principle, any reliable algorithm for localization could be used, including the techniques mentioned in Sec. II. However we have found out that the simple and fast algorithm presented in this section provides excellent results even in challenging situations. Note that, contrary to approaches that assume that the barcode is in the center of the captured frame (*e.g.* [1], [12]), our method only requires that the barcode be completely visible. We implemented other algorithms from the literature [11], [14]; however, these methods produced results comparable or inferior to our simple method, at a substantially higher computational cost.

Our localization algorithm assumes that the image of the barcode is captured with the camera oriented so that its vertical axis is approximately parallel to the bars. Thus, in correspondence of a barcode, one should expect an extended region characterized by strong horizontal gradients *and* weak vertical gradients. Accordingly, we first compute the horizontal and vertical derivatives, $I_x(n)$ and $I_y(n)$, at each pixel n. We then combine them together in a non-linear fashion as by

$$I_e(n) = |I_x(n)| - |I_y(n)|. \tag{1}$$

It is reasonable to assume that many points within a barcode should have a large value of $I_e(n)$. We run a block filter of size 31×31 over $I_e(n)$, obtaining the smoothed map $I_s(n)$. The size of the filter was chosen based on the range of the size of the input images and the minimum size of the barcode readable by our method. Note that block filtering can be implemented efficiently so that only few operations per pixel are required. Finally, we binarize $I_s(n)$ with a single threshold, selected using the method proposed by Otsu [8]. Note that this binarization is only used to localize the barcode while the decoding is performed on the graylevel image. As a consequence of thresholding, the map $I_s(n)$ may contain more than one blob. Rather than computing the connected components of the thresholded map, we simply select the pixel n_0 that maximizes $I_s(n)$, under the assumption that the correct blob, *i.e.* the one corresponding to the barcode, contains such pixel. In our experiments, this assumption was always found to be correct. Then, we expand a vertical and an horizontal line from n_0, and form a rectangle with sides parallel to the axes of the image and containing the intersections of these lines with the edge of the blob. The horizontal line $l(n)$ that passes through the center of this rectangle is chosen as the scanline for the analysis. Note that the leftmost and rightmost bars of a barcode are bordered by a *quiet zone*, a white region around the barcode which facilitates localization (see Appendix). The quiet zone, along with the large size of the block filter, ensures that the vertical sides of this rectangle fall outside the area of the barcode by at least a few pixels. Therefore, in order to localize the endpoints o_L and o_R of the barcode, we first determine the intersections i_L and i_R of the scanline $l(n)$ with the rectangle and then, the rectangle being larger than the actual barcode, we proceed inwards from each end (see Fig. 3-(d)). We stop when we find a value that is less than 85% of the average luminance from the intersection points to the current pixels:

$$o_L : l(o_L) < 0.85 \cdot \frac{\sum_{n=i_L}^{o_L-1} l(n)}{o_L - i_L - 1} \tag{2}$$

and

$$o_R : l(o_R) < 0.85 \cdot \frac{\sum_{i_R}^{n=o_R+1} l(n)}{i_R - o_R + 1}. \tag{3}$$

Although this algorithm relies on the assumption that the bars of the barcode are approximately vertical in the image, our studies show that the map $I_e(n)$ can be segmented even when this assumption is not satisfied. Indeed, as long as the barcode's bars are slanted by an angle smaller than 45°, the segmentation algorithm usually succeeds. However, it must be noted that our algorithm only uses horizontal scanlines for decoding and requires that all bars of a barcode be intersected by one such scanline. Due to the aspect ratio of typical barcodes, this adds the requirement that the bars form an angle no larger than 30° from the vertical. We present an assessment of the performance and speed of this algorithm when implemented in Matlab and Symbian in Sec IV-A and Sec IV-B.

B. Barcode decoding

Our decoding algorithm analyzes a single scanline extracted from the detected barcode area as described in Sec. III-A. The only requirement is that the beginning and the end of the barcode pattern in the scanline are detected with a certain accuracy. In our implementation we assume a localization tolerance in either end point equal to twice the width of the narrowest bar.

Algorithm outline. First, based on the previously detected endpoints of the scanline, we compute the spatial location of each digit segment in the barcode. As described in the Appendix, these digits are encoded independently of each other and occupy contiguous, non-overlapping intervals on the scanline. For each of the 12 digits in the barcode, we compare the intensity profile of

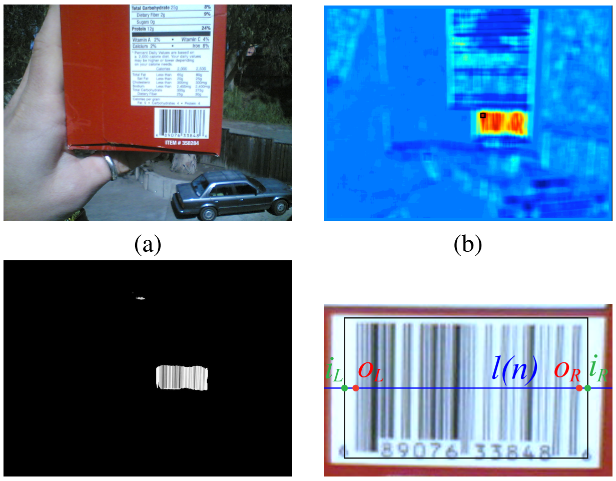


Fig. 3. An example of barcode localization with our algorithm. (a): Original image (1152×864, JPEG compressed). (b): The smoothed map $I_s(n)$ with its maximum marked by a black square. (c): Binarization by thresholding of $I_s(n)$. (d): The resulting rectangular segment (black square), along with the selected scanline $l(n)$, the intersection points i_L and i_R, and the endpoints o_L and o_R.

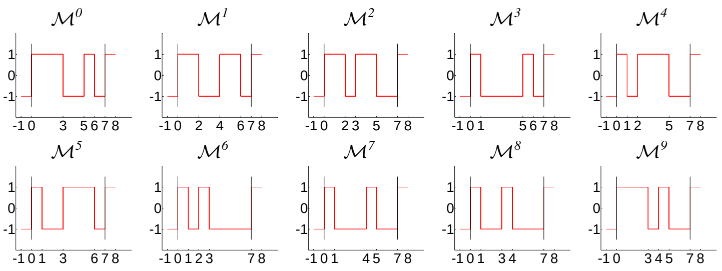


Fig. 4. Each digit in a UPC-A code is encoded with a sequence of two bars and two spaces, represented in these graphs by values of 1 and -1.

the corresponding segment of the scanline with binary templates, each representing a symbol as shown in Fig. 4. In order to account for inaccuracy in the localization of the spatial extent of each digit, we allow these templates to shift and scale in the horizontal direction. We then define a likelihood function to measure how well a deformed (shifted and scaled) template explains the observed intensity. A possible strategy could be to search for the deformation parameters that maximize the likelihood, *i.e.* the shifted and scaled template that best explains the data, hoping to avoid the many existing local maxima. Rather than focusing on a single deformation, we propose to integrate the likelihood over the space of deformations, having defined a prior distribution of the deformation parameters. One important contribution of this work is to derive an algorithm to compute this integral *exactly* and in affordable computing time.

Independent likelihood maximization over each digit segment produces a sequence of symbols. However, the result of this operation may be incorrect due to noise, blur or other causes. The risk of such errors can be reduced by exploiting global constraints on the overall sequence of symbols. The idea is that the "optimal" sequence of deformed templates should not present overlaps or gaps. We define a global cost function that, for each possible sequence of symbols, penalizes overlaps or gaps in the sequence of deformed templates, with the deformation parameters obtained by least squares regression. The minimum cost sequence can then be found via dynamic programming. We now describe in detail each of the steps of this procedure.

Deformable models. We define a *model* (or *template*) $\mathcal{M}^k$ for a given symbol k as a continuous piecewise constant function that alternates between -1 and 1, where a value of -1 (1) represents a black (white) bar (see Fig. 4). A model $\mathcal{M}^k$ for a symbol in the left half of a UPC-A barcode begins with a '-1' segment and ends with a '1' segment, where both such segments have length of 1. The lengths of the i-th constant segment between these two end segments is equal to the module width r_i^k (as defined in the Appendix). A model is therefore an archetypal representation of one symbol of a standardized scanline, plus one bar from each one of the nearby symbols. These two additional bars have base width and known polarity; adding such bars to the template increases the robustness of the matching process.

A parameterized model is a shifted and scaled (*deformed*) version of the original model:

$$\mathcal{M}_{o,w}^k(x) = \mathcal{M}^k((x-o)/w), \tag{4}$$

where o represents the starting point of the pattern and w represents the base width. (Note that models are functions of the continuous line, while the observation $I(n)$ is defined over the discrete space of pixels.) An example of deformed model is shown in Fig. 5.

Digit segment – conditional likelihood. Once the barcode has been localized in the image, and the endpoints (o_L, o_R) of the selected scanline have been estimated, the approximate position of each digit segment of the barcode is computed. More precisely, the j-th digit segment in the left side of the barcode is assumed to start at

$$o = o_L + 3w + 7w\ (j-1), \tag{5}$$

where:

$$w = \frac{o_R - o_L}{95} \tag{6}$$

is the estimated base width. These expressions derive from the fact that the overall length of the barcode is (ideally) equal to 95 times the base width, that each digit occupies a segment equal to 7 times the base width, and that the first 3 bars are guard bars.

We should stress the fact that, for a generic digit being considered, the value of o as computed in Eq. (5) is, in general, an incorrect estimate of the actual left edge of the digit segment, as a consequence of errors in the estimation of the endpoints (o_L, o_R) together with image distortion as due, for example, to perspective. However, suppose for a moment that the estimated location o and minimum bar width w are indeed correct. Then, in order to read the value of the digit, we could simply compare the intensity $I(n)$ within the segment with the models $\mathcal{M}_{o,w}^k$ for $0 \le k \le 9$, and pick the model that best fits the data. More precisely, we define the likelihood of the intensity within a generic digit segment for symbol k (conditioned on o and w) as

$$p_k(I|o,w) \propto e^{-D(I,\mathcal{M}_{o,w}^k)}, \tag{7}$$

where $I(n)$ represents the intensity profile of the considered scanline. The log-likelihood term D can be expressed as

$$D(I,\mathcal{M}_{o,w}^k) = \sum_{n=\lceil o-w \rceil}^{\lfloor o+8w \rfloor} D\left(I(n), \mathcal{M}_{o,w}^k(n)\right), \tag{8}$$

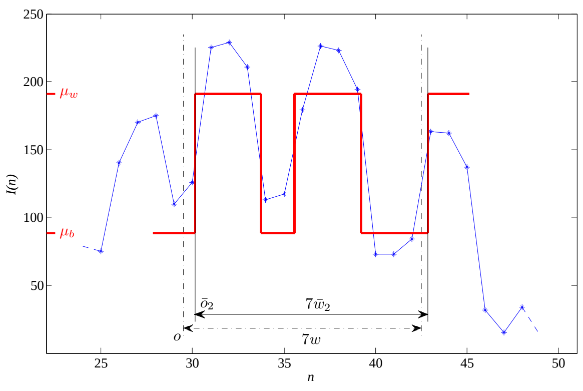


Fig. 5. A sample of intensity profile in a scanline (blue line). The segment $[o, o+7w]$ represents the location of the initial digit segment obtained from Eq. (5)-(6), whereas the segment $[\bar{o}_2, \bar{o}_2 + 7\bar{w}_2]$ is the estimated support segment as by Eq. (15) for $k = 2$. The red line represents the deformed model $\mathcal{M}^2_{\bar{o}_2,\bar{w}_2}$. For the sake of graphical clarity, the model was scaled in amplitude so that it alternates between μ_b and μ_w (as defined in Sec. III-B).

where the variable n takes on only integer values (see Fig. 5). Note that this sum is computed over all pixels that fall within the segment $[o-w, o+8w]$, which is the support of $\mathcal{M}^k_{o,w}(x)$.

A variety of functions can be considered for the log-likelihood D modeling the discrepancy between model and observation. We use the following robust formulation, which gave good results in our experiments. First, the quantities μ_w and μ_b representing the mean of the largest 50% and smallest 50% values of $I(n)$ are computed, along with their variance σ^2. Then,

$$D(I(n), -1) = \frac{[\max(I(n) - \mu_b, 0)]^2}{2\sigma^2} \tag{9}$$

and

$$D(I(n), 1) = \frac{[\min(I(n) - \mu_w, 0)]^2}{2\sigma^2}. \tag{10}$$

This function penalizes values of $I(n)$ that are small when $\mathcal{M}^k_{o,w}(n) = 1$ or large when $\mathcal{M}^k_{o,w}(n) = -1$. Note that this is *not* equivalent to binarizing the data. Indeed, the original value of $I(n)$ can be recovered from $D(I(n), -1)$ and $D(I(n), 1)$.

Digit segment – total likelihood. In order to compute the likelihood of an observed scanline for given symbol, it is necessary to take the uncertainty about o and w into consideration. This uncertainty derives from the finite tolerance on the estimation of o_L and o_R. Assume for example that both o_L and o_R are computed with a tolerance of $\pm\Delta o$. Then, barring deformations or perspective effects, o has a tolerance of $\pm\Delta o$ as well, whereas w has a tolerance of $\pm 2\Delta o/95$.

We approach this problem by first defining a probability density function $p(o,w)$ over the space of deformations. We then compute the total likelihood $p_k(I)$ by averaging $p_k(I|o,w)$ over such density:

$$p_k(I) = \int\int p_k(I|o,w)p(o,w)\, do\, dw. \tag{11}$$

Computing this integral may seem like a daunting task, especially if it needs to be performed on an embedded system such as a cellphone. On the contrary, we show that due to the particular nature of the model $\mathcal{M}^k$, and assuming a simple form for the prior $p(o,w)$, the integral in Eq. (11) can be computed *exactly* via numerical means with reasonably small complexity.

Our derivation exploits the fact that $D(I, \mathcal{M}^k_{o,w})$ is piecewise constant in the (o,w) space. This, in turn, is due to the very nature of the scanline which is itself piecewise constant: if the change in o and w is small enough, none of the boundaries d_i will "jump" to a different pixel. If we break up the sum in Eq. (8) into six pieces, corresponding to the segments in which $\mathcal{M}^k_{o,w}(x)$ takes on constant values of 1 or -1, we notice that, within segment $[d_i, d_{i+1}]$, where $d_i = o + w\sum_{l=0}^{i} r^k_l$ for $0 \le i \le 5$ and having set $r^k_0 = -1$ and $r^k_5 = 1$, the function $\mathcal{M}^k_{o,w}(x)$ is identically equal to $(-1)^i$.

$$D(I, \mathcal{M}^k_{o,w}) = \sum_{i=1}^{5} A_i, \tag{12}$$

with

$$A_i = \sum_{n=\lceil d_{i-1}\rceil}^{\lfloor d_i \rfloor} D(I(n), (-1)^i). \tag{13}$$

Hence, a variation of o or w determines a change of A_i (and therefore of $p_k(I|o,w)$) only when it causes d_{i-1} or d_i to cross over an integer value. Consequently, $p_k(I|o,w)$ is piecewise constant, and the integral in Eq. (11) can be computed exactly as a sum of a few dozen terms. Next, we show how to compute the terms in this sum.

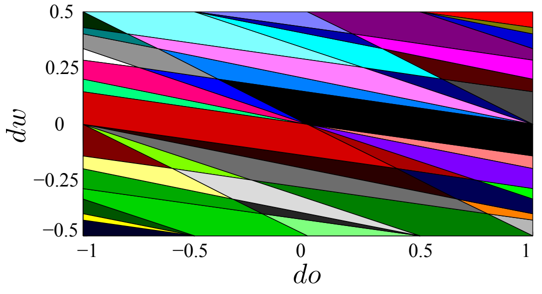
Let $\{\mathcal{V}^t_k\}$ be the minimum partition of the (o,w) plane such that $p_k(I|o,w)$ is constant within each cell $\mathcal{V}^t_k$ (with t representing the index of cells in the partition). Then

$$p_k(I) \propto \sum_t e^{-D_t} \int\int_{\mathcal{V}^t_k} p(o,w)\, do\, dw, \tag{14}$$

where $D_t = D(I, \mathcal{M}^k_{o,w})$ for any (o,w) in $\mathcal{V}^t_k$. Note that the cells $\mathcal{V}^t_k$ are polygonal, as they are defined by the lines of equation $o + w(\sum_{l=1}^{i} r^k_l - 1) = q$, where q is any integer, and i is any integer between 1 and 4 (see Fig. 6). The list of cells $\{\mathcal{V}^t_k\}$, as well as the integral of $p(o,w)$ within each cell, can be computed offline and stored for online use. In fact, one easily sees that the cells form a periodic pattern (with period equal to 1 both in o and w), hence only the cells within such a period need to be stored.

Regarding the implementation of this procedure, the following observations are in order:

1) The computation of the likelihood in Eq. (14) can be sped up by precomputing the sequences $D(I(n), 1)$ and $D(I(n), -1)$. Then, for each cell, one only needs to add together selected samples from the two sequences. Suppose that Eq. (11) requires summing over N_j cells. For each cell $\mathcal{V}^t_{j,k}$, the negative log-likelihood D_t needs to be computed, which requires two additions and two multiplications per sample. Overall, $2N_j$ additions and $2N_j$ multiplications per sample. However, it is easily seen that by precomputing $D(I(n), 1)$ and $D(I(n), -1)$, each computation of D_t only requires one addition per sample. This reduces the computational weight to $2 + N_j$ additions and 4 multiplications per sample.
2) At run time, a specific set of cells is chosen from the list based on the tolerance Δo and Δw on the estimated values of o and w, which are easily derived from the tolerance of the estimated endpoints o_L and o_R. More precisely, we compute the sum in Eq. (14) over the cells that intersect the rectangle with sides $[o-\Delta o, o+\Delta o]$ and $[w-\Delta w, w+\Delta w]$, where o and w are estimated as by Eq. (5).
3) The integration of $p(o,w)$ within each cell results particularly simple if $p(o,w)$ is assumed to be uniform within the



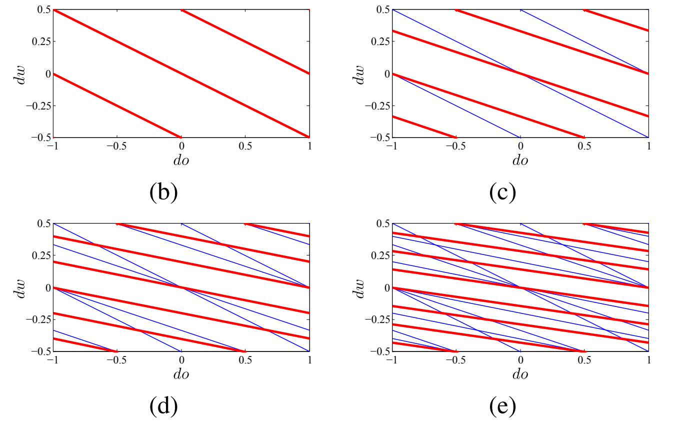


Fig. 6. The space (do, dw) can be broken into polygons in which the conditional likelihoods $p_k(I|o,w)$ are constant. Plot (a) shows this partition for symbol '2', for which $\{r_i^2\}$ is $\{2,1,2,2\}$. Each one of the four bars in a symbol defines a set of parallel lines in the space (do, dw); for example, when the values of (do, dw) in plot (b) cross one of the red lines, the right boundary of the first bar crosses a pixel and, therefore, the likelihood changes. Plots (c)-(d) show in red the sets of parallel lines corresponding to bars 2-4. The equations for these lines are easily computed; for the third bar (plot (d)), for instance, we can write: $dw = -\frac{1}{2+1+2}do + q$. The final partition is shown in plot (a). Intuitively, if (o_1, w_1) and (o_2, w_2) fall within the same cell of plot (a), the conditional likelihoods $p_2(I|o_1,w_1)$ and $p_2(I|o_2,w_2)$ are identical. In other words, instead of computing the integral in Eq. (11) over every point of the space, the problem can be made tractable by only considering one point per cell without introducing any approximation (see Sec. III-B).

considered rectangle in the (o, w) space. In this case, the integral is proportional to the area of the polygonal cell, which can be easily computed and stored offline. In our implementation we made use of this simple, yet effective model.

As will be shown shortly, it is also useful to estimate, for each possible symbol k, the deformation parameters (o, w) given the intensity profile $I(n)$ within a digit segment. We choose the least squares estimator $(\bar{o}_k, \bar{w}_k)$ of these quantities (under the density $p(o,w)$), which is given by the conditional expectation. Using Bayes rule, this is equivalent to

$$(\bar{o}_k, \bar{w}_k) = \int\int (o,w)\, \frac{p_k(I|o,w)p(o,w)}{p_k(I)}\, do\, dw \qquad (15)$$

$$\propto \frac{1}{p_k(I)} \sum_t e^{-D_t} \int\int_{\mathcal{V}_k^t} o\ w\ p(o,w)\, do\, dw.$$

The integrals in the above equation can be precomputed and stored for online use. If the assumption of uniformly distributed (o, w) is made (as in point 3. above), then the terms in the sum are the centroids of the cells $\{\mathcal{V}_k^t\}$.

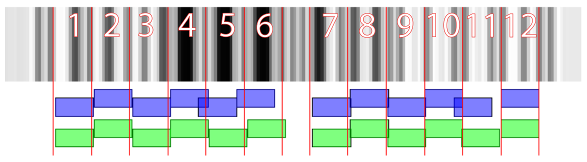


Fig. 7. The support segments $[\bar{o}_{j,k(j)}, \bar{o}_{j,k(j)} + 7\bar{w}_{j,k(j)}]$, where $k(j)$ are the maximizers of the total likelihood $p_{j,k(j)}(I)$ for each digit index j, are shown in blue against the support segments corresponding to the sequence of values $\{k\}$ minimizing the global cost C in Eq. (17), shown in green. Digits 5, 6, and 11 are not correctly decoded and their position is therefore miscalculated (blue). The algorithm described in Sec. III-B successfully enforces global consistency and, thus, correct decoding (green). The red lines represent the original digit segments, obtained from Eq. (5)-(6). In order to provide a visual intuition of the intensity profile of the scanline under consideration, the latter was repeated vertically to form a graylevel image.

Imposing spatial coherence. Our model makes the simplifying initial assumption that the digit segments are equally spaced (see Eq. (5)-(6)). This also implies that the base width w is constant across the barcode. In practice, we should expect that the digit segment length may vary from segment to segment, generally within the confidence intervals Δo and Δw. Ideally, however, the segment representing a given digit in the scanline (as computed from the estimates $\bar{o}_k$ and $\bar{w}_k$) should be adjacent to (but non overlapping with) the neighboring segments. The choice of an incorrect value of k due to single-digit analysis is likely to result in a supported segment that does not fit well together with the other segments (see Fig. 7). This observation can be exploited by imposing a global constraint as follows.

Suppose that the j-th digit takes value $k(j)$. (Note that we need to make the dependency on j explicit in our notation from now on.) The estimated deformation parameters $(\bar{o}_{j,k(j)}, \bar{w}_{j,k(j)})$ define the supported segment $[\bar{o}_{j,k(j)}, \bar{o}_{j,k(j)} + 7\bar{w}_{j,k(j)}]$. We define the overlap/gap extent between the j-th and $(j+1)$-th estimated digit segments as

$$O_{j,k(j),k_{j+1}} = |\bar{o}_{j,k(j)} + 7\bar{w}_{j,k(j)} - \bar{o}_{j+1,k(j+1)}|. \qquad (16)$$

Now define a global cost function as follows:

$$C(\{k\}) = \sum_j \alpha O^2_{j,k(j),k(j+1)} - \log p_{j,k(j)}, \qquad (17)$$

where α is a balancing parameter, and the sum extends to all digit segments in the left and right half of the barcode. (α was set to be equal to 0.1 in our experiments). The cost function in Eq. (17) penalizes sequences of symbols that create large overlaps or gaps between two consecutive digit segments or that produce low values of likelihood. Dynamic programming can be used to minimize the cost function C over the space of sequences $\{k\}$ of symbols. Fig. 7 shows the outcome of the application of this technique.

IV. Implementation and Tests

We implemented and tested our algorithm both in Matlab and in Symbian OS, a platform used by several cellphone brands. In the following subsections we provide results in terms of accuracy and computational load for both implementations.

A. Matlab implementation

Localization. The algorithm described in Sec. III-A proved rather robust in our experiments. Although the algorithm relies on

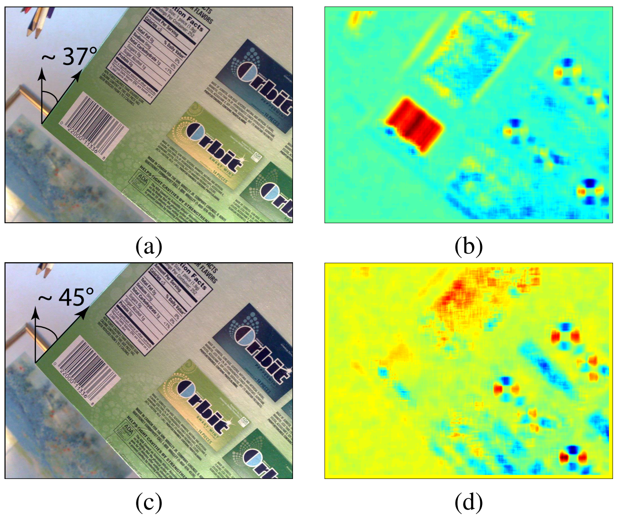


Fig. 8. The localization algorithm assumes that the cellphone is held in an approximately vertical fashion. However, it proves robust to a large deviation from the ideal setup. (a) an image of a barcode at 37°, (b) the corresponding energy map computed as described in Sec. III-A, (c) a barcode at 45°, and (d) the corresponding energy map. Note that, as long as the barcode's angle is below 45°, the map is still easy to segment, whereas the energy in the area corresponding to the barcode is dramatically low for angles from 45° and above.

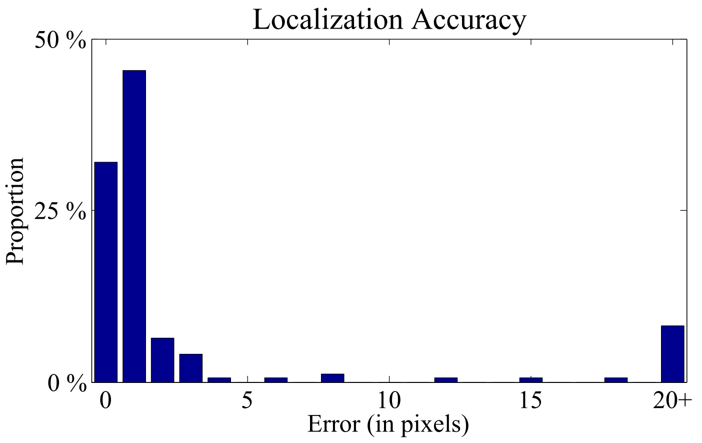


Fig. 9. Accuracy of the localization algorithm in pixels. The endpoints o_L and o_R were manually selected for each image in our data sets and compared with the output of the localization algorithm described in Sec. III-A. The bin labeled '20+' contains all the pixels that were at least 20 pixels away from the ground truth.

the assumption that the user is holding the cellphone so that the bars are vertical in the image, it produces good results even when the phone is rotated at an angle. The maximum rotation angle for correct localization is bound by simple geometric considerations. Recall that localization comprises two steps: segmentation of the barcode image, followed by determination of the horizontal scanline segment used for decoding. The first step fails if the barcode is rotated by more than 45°, since in that case the horizontal image gradient has smaller magnitude than the vertical gradient (see Fig. 8(d)). Our experiments show that the segmentation generally fails for angles larger than 40° - 45°, depending on the overall quality of the image.
The second step requires that the scanline intersect all the bars, from the left to the right guard bars; since we extract an horizontal line, this becomes harder as the angle increases too much. In our tests, the scanline was correctly extracted for angles up to 30°.

We also characterize the performance of the localization algorithm in terms of its accuracy. For each image in the data sets, we manually selected the endpoints o_L and o_R and compared them with the output of our localization algorithm (see Fig. 9). Note that approximately 75% of the points are detected within one pixel of the correct location. Approximately 10% of the points have a localization error of more than 10 pixels. In the case of these "gross errors", the decoding algorithm typically breaks down. These situations are responsible for most of the failures in the experiments described below.

The execution time is only a function of the size of the image. Our Matlab implementation takes from approximately 0.065 seconds for a 640×480 image to approximately 0.21 seconds for a 1152×864 image.

Decoding. The localization algorithm is used to provide a single scanline segment together with its endpoints as the input to our reader. The maximum tolerance Δo at the endpoints o_L and o_R was set to $\pm 2w$, where w, the initial estimate of the base width, was defined in Eq. (6). In practice, this means that we expect the localization algorithm to possibly miss up to one bar in both the start and end patterns. The minimum scanline width $o_R - o_L$ that our algorithm was able to decode was of 100 pixels. Note that this corresponds to a base width of only 1.05 pixels.

The computational speed of the algorithm depends heavily on the scanline width. This is mostly due to the fact that the number of cells V_k^t depends on the tolerances Δo and Δw, which are proportional to the scanline width. For example, when the scanline width is equal to 100 pixels, then 25 cells are generated. However, in the case of a high resolution image of a barcode at a short distance, producing a scanline width of 1128 pixels, 2185 cells are generated. In order to reduce the number of cells to be processed, we implemented a simple variation of the algorithm, by fixing the width of the first and last bars in the model to the minimum width considered. Remember that these are "overlap" bars with nearby digits, and that they always have unitary width in the model $\mathcal{M}^k$. With this modification, the number of cells is reduced to 17 in the 100 pixel scanline width case, and to 641 in the 1128 pixel case. The computational speed of the decoder (excluding localization) ranges between 0.076 seconds to 0.65 seconds.

In order to assess the performance of the system, we tested it on a variety of images. Unfortunately, we could find only one barcode image database for comparative assessment[1]. This database, which was created by Tekin and Coughlan, is accessible at `www.ski.org/Rehab/Coughlan_lab/Barcode`. The images are all of high resolution, and each image was manually cropped around the barcode. The authors divided it into "Hard" and "Clean" subsets, and showed results of their algorithm on both sets [10].

In order to assess our system in other realistic situations, we gathered a number of images taken from two different cellphones, and created three new data sets. Data set 1 contains images at high resolution (1024×768) from a Nokia N95 cell phone. This device has autofocus capability, although not all images collected were properly in-focus (whether because the focus was on the wrong object or because the focusing algorithm failed), and some were affected by motion blur. Data set 2 contains images taken

[1]Wachenfeld *et al.* [12] claim to have assembled a barcode image database for public use, but we have not been granted access to it.

by the same cell phone, but at 640×480 resolution, and highly compressed in JPEG (with apparent coding artifacts). Data set 3 contains images at 1152×864 resolution taken with an older Nokia 7610 phone, with fixed focus. All three data sets have multiple pictures of several barcodes, taken from different distances and in different conditions.

Our algorithm was tested against the algorithm of Tekin and Coughlan [10], for the "Hard" and "Clean" data set, as well as against two barcode reading softwares that are available online. The first one, from DataSymbol[2], was also considered by Tekin and Coughlan [10]. The second one, from DTK[3], was shown to produce impressive results. A third online software, from QualitySoft, was considered by Tekin and Coughlan [10], but we neglected comparison with it since it gives very poor results.

Numerical results from our tests are presented in Fig. 10. Note that the checksum digit can be used to determine whether a barcode is correctly decoded. In all the data sets, our algorithm outperforms the other techniques, in particular for the most challenging sets (Data set "Hard" and Data set 3). A sample of images correctly decoded by our algorithm is shown in Fig. 11, while examples of failures are shown in Fig. 12. Note that in many cases, failure was due to incorrect initial localization. Our algorithm correctly decodes barcodes even when the image quality is extremely poor, as can be appreciated by zooming in on the images.

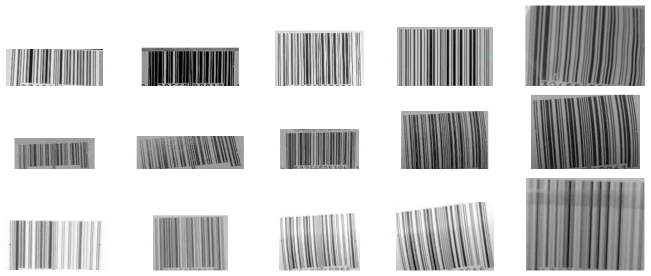


Fig. 11. Example of barcodes correctly decoded by our algorithm from the three data sets described in Sec. IV-A: the crop-outs in the first row are from 1024×768 images (data set 1), those in the second are from 640×480 images (data set 2), and those in the third from 1152×864 images (data set 3). These crop-outs are the output of the localization algorithm described in Sec. III-A. The green stars indicate (o_L, o_R), the original estimates for the endpoints of the scanline. Please note the extremely poor quality of the images by zooming in.

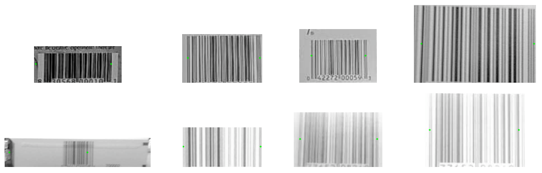


Fig. 12. Example of images in which our algorithm fails from data sets 1 (1024×768) and 3 (1152×864). These crop-outs are the output of the localization algorithm described in Sec. III-A. The green stars indicate (o_L, o_R), the original estimates for the endpoints of the scanline.

B. Symbian implementation

We implemented our algorithm in Symbian and tested it on a Nokia N95 8GB device using VGA (640×480) still images. The localization algorithm is implemented exactly as in the Matlab version. As for the decoding algorithm, we implemented a minor modification concerning the storage of the polygons in Sec. III-B. Indeed, storing the partition of the (do, dw) plane and intersecting it at runtime with a rectangle whose sides are given by the tolerances described in Sec. IV-A, would be computationally intensive. Therefore, we only store area and centroid location of each polygon. Then, at runtime, we compute Eq. (11) using all the polygons whose centroid falls within mentioned rectangle. While dramatically decreasing the search time, based on our experiments, this change does not seem to affect the accuracy of the algorithm.

From the moment that the frame is received from the camera and the analysis begins, our algorithm takes an average of 400-500 ms for both localization and decoding. (Localization is only a function of the image size and it takes approximately 250 ms for a VGA image.) This shows that our algorithm is suitable for mobile applications and also proves that, in the event of failure at reading the barcode, the process can be quickly repeated.

It is important to note that our algorithm allows the user to take a shot of the scene without strong constraints on the location or distance of the barcode, as long as the resolution is sufficient for decoding. This is in contrast to other mainstream applications, such as RedLaser for iPhone [1], which require the user to carefully maneuver the cellphone in close proximity of the barcode to center it and to maximize its apparent size in the viewfinder frame. Moreover, the user needs to wait, slightly adjusting the position of the phone, until a frame is grabbed that can be successfully decoded by the algorithm.

The barcodes printed on commercial products have sizes typically ranging from 2.5 to 4 cm, which directly affects the maximum distance at which it can be correctly read. Given the field of view of the Nokia N95, our implementation can successfully localize and read barcodes that are 2.5 cm wide when they are imaged from up to 30 cm away. Barcodes that are 4 cm wide can be decoded from up to 45 cm. This provides an advantage over other methods in cases where it might not be possible to place the phone in close proximity to the barcode, a strict requirements of RedLaser and similar applications.

V. Conclusions

We have presented a new algorithm for barcode decoding (localization and reading) that can deal with images that are blurred, noisy, and with low resolution.

Our localization technique is fast and accurate even for cluttered images; in more than 75% of the images in our challenging data sets, the detected endpoints of the barcodes are within one pixel of their actual location, a higher accuracy than that required by our decoding algorithm.

Unlike previous approaches, our algorithm does not binarize the image, thus sidestepping a critical and often error-prone, early-commitment procedure. We use deformable templates to represent each digit of the barcode, averaging over the set of all expected deformations. A final procedure for global spatial coherence helps reducing the risk of errors at the individual digit level. We extensively tested our algorithm showing improved performance with respect to other state-of-the-art software and algorithms, especially for the most challenging images. We also made the data sets public to allow other researchers to test and compare their methods.

[2]http://www.datasymbol.com
[3]http://www.dtksoft.com

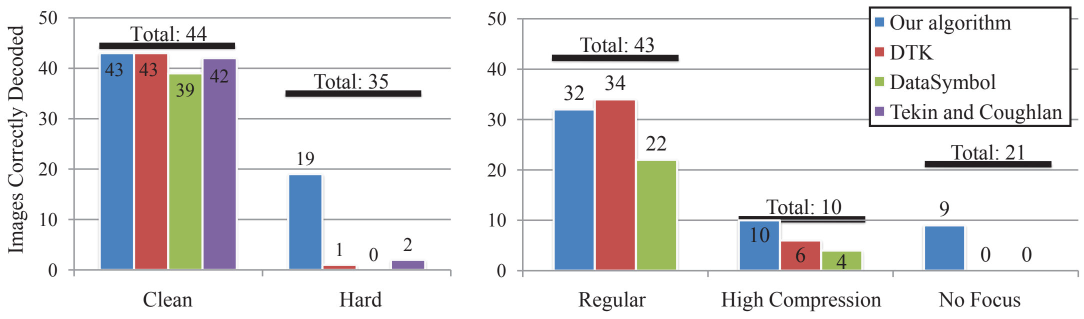


Fig. 10. Comparative results showing the number of barcodes correctly detected in the different data sets considered (on the left the data sets from Tekin and Coughlan [10], on the right our data sets). For each data set, the black, horizontal line indicates the total number of images.

Implemented on a Nokia N95, the complete reading algorithm is executed in less than 0.5 seconds and is therefore suitable for mobile vision applications.

References

[1] Redlaser. `http://redlaser.com/`, accessed on February15th, 2010.

[2] UCSC UPC Dataset. `http://soe.ucsc.edu/~orazio/Data/UCSC_UPC_Dataset.zip`, accessed on March13th, 2010.

[3] R. Adelmann, M. Langheinrich, and C. Flörkemeier. A toolkit for bar-code recognition and resolving on camera phones—jump starting the internet of things. In *Workshop Mobile and Embedded Interactive Systems (MEIS06) at Informatik*, 2006.

[4] D. Chai and F. Hock. Locating and decoding EAN-13 barcodes from images captured by digital cameras. pages 1595–9, 2005.

[5] S. Krešić-Jurić, D. Madej, and F. Santosa. Applications of hidden Markov models in bar code decoding. *Pattern Recognition Letters*, 27(14):1665–1672, 2006.

[6] R. Muniz, L. Junco, and A. Otero. A robust software barcode reader using the Hough transform. *Information Intelligence and Systems, 1999. Proceedings. 1999 International Conference on*, pages 313–319, 1999.

[7] E. Ohbuchi, H. Hanaizumi, and L. Hock. Barcode readers using the camera device in mobile phones. In *Proceedings of the Third International Conference on Cyberworlds (CW'04)*, volume 00, pages 260–265, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[8] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–6, 1979.

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

[10] E. Tekin and J. Coughlan. A Bayesian algorithm for reading 1D barcodes. In *Sixth Canadian Conference on Computer and Robot Vision*, 2009.

[11] A. Tropf and D. Chai. Locating 1-D bar codes in DCT-domain. In *2006 IEEE International Conference on Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings*, volume 2, 2006.

[12] S. Wachenfeld, S. Terlunen, and X. Jiang. Robust recognition of 1-D barcodes using camera phones. In *International Conference of Pattern Recognition*, pages 1–4, 2008.

[13] K. Wang, Y. Zou, and H. Wang. 1D bar code reading on camera phones. *International Journal of Image and Graphics*, 7(3):529–550, July 2007.

[14] C. Zhang, J. Wang, S. Han, M. Yi, and Z. Zhang. Automatic real-time barcode localization in complex scenes. In *International Conference of Image Processing*, pages 497–500, 2006.

Appendix: UPC-A Barcodes - Syntax

UPC-A is a technology to encode numbers with 12 decimal digits (*symbols*) as an alternating sequence of black bars and white bars (*spaces*) with different widths. (The last digit is an error correcting check digit which can be used to detect decoding errors.) Each bar may have width equal to $r \times w$, where r (the *module width*) is an integer between 1 and 4, and w, the *base width* (sometime called *X-dimension*), is the width of the narrowest bar. The code is divided into two halves separated by a sequence of three spaces and two bars (central *guard bars*), all of unitary module width. At the two ends of the barcode there is a sequence of two bars separated by a space, all of unitary module width (lateral guard bars). The lateral guard bars are sided by a space of width equal to at least 9 times the base width (*quiet zone*), although this requirement is sometimes violated in real-world instances. Between the lateral and the central guard bars, the code is divided into 6 equally spaced *digit segments*, or simply *digits*, each of which with length equal to 7 times the base width. Thus, the overall length of the barcode is equal to 95 base widths. Each digit represents one symbol as a sequence of two spaces and two bars. The value k of a symbol is encoded by the sequence of module widths $(r_1^k, r_2^k, r_3^k, r_4^k)$ of the bars and spaces in the digit segment. The standardized UPC-A sequences for the left half of the code are shown in Fig. 4. In the right half of the code, the same sequence of widths is used to encode a symbol, however the role of spaces and bars is inverted.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant No. IIS - 0835645 and in part by the the National Institute of Health under Grant 1 R21 EY017003-01A1.

Orazio Gallo received his Laurea degree (MS) in Biomedical Engineering from the "Politecnico di Milano", Italy, in 2004. He then worked on a novel bioimaging technique at the Smith-Kettlewell Eye Research Institute from 2004 to 2006. Currently he is a Ph.D. candidate at the department of Computer Engineering at the University of California, Santa Cruz. His main research interests are computer vision and computational photography.

Roberto Manduchi obtained his Ph.D. in Electrical Engineering from the University of Padova, Italy, in 1993. He is currently an Associate Professor of Computer Engineering at the University of California, Santa Cruz. Prior to joining UCSC in 2001, he worked at Apple Computer, Inc., and at the NASA Jet Propulsion Laboratory.