Published in final edited form as:
Proc IEEE Workshop Appl Comput Vis. 2009 December 7; 2009: 1–8. doi:10.1109/WACV.2009.5403098.

An Algorithm Enabling Blind Users to Find and Read Barcodes

Ender Tekin and **James M. Coughlan**
The Smith-Kettlewell Eye Research Insitute

Ender Tekin: ender@ski.org; James M. Coughlan: coughlan@ski.org

Abstract

Most camera-based systems for finding and reading barcodes are designed to be used by sighted users (e.g. the Red Laser iPhone app), and assume the user carefully centers the barcode in the image before the barcode is read. Blind individuals could benefit greatly from such systems to identify packaged goods (such as canned goods in a supermarket), but unfortunately in their current form these systems are completely inaccessible because of their reliance on visual feedback from the user.

To remedy this problem, we propose a computer vision algorithm that processes several frames of video per second to detect barcodes from a distance of several inches; the algorithm issues directional information with audio feedback (e.g. “left,” “right”) and thereby guides a blind user holding a webcam or other portable camera to locate and home in on a barcode. Once the barcode is detected at sufficiently close range, a barcode reading algorithm previously developed by the authors scans and reads aloud the barcode and the corresponding product information. We demonstrate encouraging experimental results of our proposed system implemented on a desktop computer with a webcam held by a blindfolded user; ultimately the system will be ported to a camera phone for use by visually impaired users.

1. Introduction

The 1D barcode symbology has become a ubiquitous system for labeling packaged goods with codes that uniquely identify product information. This type of symbology (which includes the UPC, widely used in North America) was designed to be read by a laser scanner, which is a standard tool in supermarkets and shops, but there is increasing interest in being able to read barcodes using a more portable device that nearly everyone has at their disposal: the camera phone. The cameras available on popular mobile phones now have enough resolution (and the ability to focus close up) to resolve barcode patterns at close range, and the CPUs available on such phones are powerful enough to execute a computer vision algorithm to decode the barcode images.

Products such as Red Laser (for the iPhone), Nokia's Point and Find and Realeyes3D have recently been released to provide this functionality, but is important to note that all such products force the user to locate and home in on the barcode using visual feedback from the camera viewfinder. Without such feedback, it is nearly impossible to guide the camera close enough to the barcode to view it with sufficient resolution. As a result, these products are only accessible to people with sufficiently good vision.

Blind and visually impaired persons could greatly benefit from the ability to read barcodes, which would be very useful for identifying products in a supermarket or in a pantry at home: while it is easy for a blind person to distinguish between a bag of potato chips and a box of crackers by feeling the packages, he or she may have no way of distinguishing between two cans with the same shape and weight (e.g. tomato sauce or beans). If the product barcodes could be located and read by a camera phone application, the relevant product information could immediately be looked up and read aloud to the user.

A computer vision algorithm enabling a blind person to locate and read barcodes could be implemented on a camera phone, perhaps in addition to other functionality providing access to printed information (such as the KNFB reader, which is a commercial OCR system that runs on a Nokia camera phone). Such a system would be much more convenient than conventional laser scanner technology that requires the user to carry a dedicated device, such as the i.d. mate Talking Bar Code Reader (from Envision America).

While a variety of computer vision algorithms exist for reading 1D barcodes from images, and some algorithms have been developed for automatically segmenting barcode patterns from images (which we review in the next section), *we are not aware of any algorithms specifically designed to detect 1D barcodes at a distance*. This ability is necessary to expedite the search process – otherwise a blind user will be forced to hold the camera close to the package and laboriously scan every square inch of its surface.

Indeed, although the 1D barcode is easily recognizable in close up, clean images by the presence of multiple parallel lines arrayed in a roughly rectangular shape, the problem of detecting it becomes very challenging in noisy (e.g. blurred), cluttered images captured with low-to-medium quality cameras (such as those in camera phones or webcams) at low resolution. The barcode may be present at any orientation in the image, and the scale is variable, depending on how large the barcode is and the distance at which it is viewed; patterns such as lines of text, which are commonly printed near barcodes on packages, may easily be confused with barcodes at low resolution. The challenge is even greater given the need for a fast implementation that can process several frames per second, in order to provide real-time feedback to guide a blind user holding the camera towards the barcode. It is of practical interest to detect a barcode accurately from a distance and direct a blind user towards the barcode until it is able to be resolved with sufficient accuracy to be decoded.

Our main contributions are two-fold. First, we have developed a fast, novel algorithm for detecting barcodes at a distance based on a cascade of filters that progressively rule out more and more regions of the image; only regions of the image that resemble barcodes are processed intensively, which is the key to the algorithm's speed. Second, we have coupled our algorithm with an algorithm that we developed previously [9] to read barcodes, and have implemented a real-time version of our system with some improvements to the model using a PC and an inexpensive webcam. We added speech feedback to guide a blind user to a detected barcode, and we tested the system while wearing blindfolds, demonstrating the feasibility of the system for totally blind users. Ultimately the system will be ported to a camera phone and tested with blind and low vision users, but our current experimental results show that the system is eminently usable and reliable. We note that the webcam we used has the same resolution as the Nokia N95 phone running in video mode, which is our initial target platform for the mobile version of our system.

2. Related Work

In the past, most research on reading barcodes focused on the problem of decoding barcode signals from a laser scanner, which are 1D waveforms (time series) representing barcode slices. One work [4] focuses on an optimal procedure for deblurring the waveforms to restore the locations of closely spaced edges in order to improve the decoding process. Another paper [6] uses an HMM model to decode a barcode signal in the presence of spurious edges.

More recent work has addressed the problem of reading barcodes from images acquired by a camera. Most of this work has simplified the problem of locating and segmenting the barcode in the image by assuming certain constraints hold. In [5], the barcode is assumed to be horizontal in the image and viewed close enough for the long width of the barcode to subtend about two-thirds of the image width; although the authors point out that the method can be easily extended to the case of unknown orientation, it is unclear how well the algorithm would work with barcodes viewed from farther away. Other work [2] assumes that the barcode can be detected from the morphological structure of binarized regions of the image, but the binarization procedure may fail on images that are noisier than the crisp example shown in the paper; similarly, [7] assumes that a barcode can be detected by searching for a black bar using a spiral searching method, which scans in a spiral outward from the center of the image, but it is unclear that an individual black bar can be sufficiently well resolved at a distance.

In addition to the commercial products mentioned in Sec. 1 (Red Laser, Nokia Point and Find, and Realeyes3D), which use proprietary algorithms, there have been some publications describing research on reading 1D barcodes using camera phones. In [10], a barcode reading algorithm assumes that “a horizontal scanline in the middle of the image will cover the barcode,” and exploits the opportunity to capture multiple frames each second until this constraint is satisfied. (Our proposed system is similar in that it captures multiple frames until one of them can be read, but we detect the barcode from a distance and guide the user towards it *before* attempting to read it.) Finally, [12] specifically addresses the problem of detecting and localizing 1D barcodes at unknown orientation in cluttered, noisy images, but there is no mention of how far away the barcodes can be detected (in the examples shown in the paper, the barcodes appear close enough to be at moderately high resolution).

We build on our recent work [9,8] focusing on reading a barcode (assuming it has already been segmented) using a Bayesian deformable template algorithm that combines a prior model of the geometric shape of the barcode pattern with a likelihood model that evaluates evidence in the image for edges. Such an approach is a principled technique of decoding noisy barcode images that contain spurious (or missing) edges. A related approach [3], also based on deformable templates, can successfully decode barcodes from extremely blurry and noisy pictures.

Finally, we note that there is growing interest [11,1] in the use of 2D barcodes, which are better suited to acquisition by camera than 1D barcodes, and encode information more densely. While 2D barcodes will one day supplant their 1D predecessors, at present almost all packaged goods are still labeled with 1D barcodes, which is why this paper focuses on them.

3. Overview

Before we go into the details of our algorithms, we give a brief overview of the major steps, shown schematically in Fig. 1. The system can be broken down into four main sub-systems: a detection part that looks for evidence of a barcode in the image, a direction system that guides the user to a barcode if one is found, a decoding step that decodes the actual UPC-A code from the barcode once all the edges are seen, and the final stage which matches the UPC-A code to a product descriptions and outputs this information. Below is a summary of these steps:

1. **Detection:**
    - **a.** Lines in 4 different orientations swept to determine collection of edge points with alternating polarities.
    - **b.** Line scores tallied in direction perpendicular to sweep direction to get 2D representation of possible barcode areas.
    - **c.** Orientation entropy used to eliminate false positives (e.g. dense text).
2. **Direction:**
    - **a.** A maximal bounding box to enclose the detected barcode is calculated.

   - **b.** The user is directed to the barcode by voice commands until enough edges are seen.

3. **Decoding:**

   - **a.** Slices with maximum number of edges are found and edges localized with sub-pixel accuracy.
   - **b.** Maximum likelihood (ML) estimation of the fundamental width and fixed edges.
   - **c.** ML estimation of the barcode digits using the check bit.
   - **d.** Detection attempted both right side up and upside down.

4. **Output:**

   - **a.** Product information retrieved from database and read out.

In the next few sections, the details of these stages are provided.

4. Algorithm for Finding Barcodes

1D barcode patterns are characterized by a rectangular array of parallel lines. The particular symbology we focus on in this paper is UPC-A (Fig. 2), which is widespread in North America and encodes a total of 12 decimal digits (one of which is a checksum digit that is a function of the preceding eleven digits). The UPC-A pattern contains a sequence of 29 white and 30 black bars, for a total of 60 edges of alternating polarity.

Any algorithm for finding a 1D barcode will conduct some sort of search for linear edge features in an image. While simple pre-processing steps such as intensity binarization and line extraction may be useful for identifying these features when they are clearly resolved, these steps may fail when the barcode is viewed from a distance. Instead, we decided to begin our detection algorithm by drawing on a simple, local image cue: the direction of the image gradient. The important signature of a barcode region is that, among pixels where the image gradient is significantly above zero, nearby pixels in the region have gradient directions that are either roughly aligned (corresponding to edges of the same polarity) or anti-aligned (corresponding to edges of opposite polarity).

Thus, in the first stage of our detection algorithm, we calculate the image gradient everywhere in the image, and at all locations where the gradient magnitude is above a threshold (which we refer to as *edge pixels*) we calculate the gradient direction as an angle from 0 to 2π. Next we scan the image in four different orientations: horizontal, vertical, and both diagonals (±45°). Let us consider the horizontal orientation first. The scan is conducted in raster order (top row to bottom row, and left to right within each row), and we search for edge pixels whose orientation is consistent with vertical bars. For each such edge pixel, we search for a nearby "mate" pixel with the opposite polarity. Once a sufficient number of these pixels are found close by on a line segment, this segment is saved for the next step which sweeps the lines in a direction perpendicular to the first sweep direction to see if there are any approximately consecutive segments that have similar beginnings and ends. If a number of candidate line segments with similar beginnings and ends are found in this manner, this area is saved as a possible barcode candidate and passed on to the next stage which eliminates false positives that may arise, such as dense text when seen from a distance. These algorithms are summarized in Figures 3 and 4.

The gradient angles which were quantized into 16 bins are histogrammed into 8 bins by combining pixels whose directions are 180 degrees apart. We then calculate the entropy of the resulting distribution, and compare it to a maximum threshold. Since a barcode is expected to only have lines of a single orientation, we expect a low entropy value. This stage eliminates false positives from the previous stage such as text, which has more orientations. As we direct the user to the barcode by giving directional feedback, the localization accuracy also increases.

5. Algorithm for Reading Barcodes

This part is based on a previous publication by the authors, [9], that models a barcode as a deformable template. We start with an initial estimate of the fundamental width, *X*, of the barcode (i.e. the width of the narrowest black or white bar) using the end points of the barcode bounding box from the previous stage. We first model the “fixed edges” of a UPC-A barcode, which are shown in Figure 2 as the guard-band edges and the digit boundaries shown in red. We model these fixed edges and digits conditioned on the barcode slice as obeying a Gaussian distribution centered around their expected geometric locations (which consists of their expected absolute distance from the left barcode edge and their relative distance from the previous fixed edge), and an exponential distribution in terms of their gradient strengths as given below:

$$P(E,D|S)\propto e^{-L(E,S)-G(E,D)} \quad (1)$$

where $L(E, S)$ is the (log) likelihood term that rewards edge locations lying on high-gradient parts of the scanline, and $G(E, D)$ is the geometric term that enforces the spatial relationships among different edges given the digit sequence.

By assuming conditional independence of a fixed edge from the previous fixed edges given the immediately prior edge, we can come up with a Markovian description of the fixed edges. This allows us to the find the maximum likelihood estimate of these locations efficiently using the Viterbi algorithm. We then iteratively refine this estimate and the fundamental width until we are satisfied with our estimate.

Once we find the fixed edge locations, we calculate the probabilities of the “in-digit” edges for each barcode digit, which gives us a distribution on the probabilities of each digit 0,…, 9 for this location. These are then used in conjunction with fixed edge estimates to get an overall estimate of the barcode. Since the digits are not conditionally independent due to the check bit, we use an auxiliary variable that is a running parity and preserves these probabilities as well as obeying the Markovian property. Hence, we can once more use the Viterbi algorithm to efficiently calculate the maximum likelihood estimate of the barcode.

We use a multi-candidate Viterbi algorithm to ensure that the probability of our estimate is sufficiently larger than the probability of the second best ML estimate. We also ensure that the estimate is at most 1 digit away from the individually most likely digit estimates, since the parity digit is only guaranteed to find single-digit errors. This algorithm is summarized in Figure 5.

6. System Implementation

After designing and testing the algorithms primarily in Matlab, the entire code base was ported to C++ for speed. The system was executed on a desktop computer with an inexpensive webcam, and the manual focus of the webcam was set to an intermediate focal distance: far enough for the webcam to resolve barcodes sufficiently well to be detected at a distance, but close enough for the webcam to resolve the barcode clearly enough to read properly at close range. We also experimented with autofocus webcams, but the time lag due to the autofocus feature proved to be impractical for a real-time system. Microsoft Speech API was utilized for the oral directional feedback.

We devised a simple acoustic user interface to guide the user to the barcode. For each image frame, if a candidate barcode is detected then the system issues directional feedback instructing the user to move the camera left, right, up or down to better center the barcode in the field of view. If the barcode is oriented diagonally then the user is instructed to rotate the barcode, to allow the barcode to be aligned either approximately horizontally or vertically with the pixel lattice; this was done because the square pixel lattice maximally resolves the 1D barcode pattern when the code axis is perfectly horizontal or vertical, whereas barcodes oriented diagonally are harder to resolve. (Note that it is unnecessary to tell the user which direction to rotate in, since the user need only align the barcode to the pixel lattice modulo 90°) If the barcode is close enough to detect but too far to read then the system tells the user to bring the camera closer, or if the barcode covers a very big portion of the webcam, the user is instructed to move farther to ensure the whole barcode is captured.

Once the barcode is sufficiently close and well centered, the system attempts to read the barcode repeatedly (sounding a beep each time to inform the user) until the barcode is decoded with sufficiently high confidence. The barcode digit string read by the algorithm is looked up in a UPC code database (freely available online at http://www.upcdatabase.com/); if the string exists in the database then the corresponding descriptive product information is read aloud (e.g. “Walgreens Fancy Cashew Halves with Pieces. Size/Weight: 8.5 oz. Manufacturer: WALGREEN CO.”). If the string is not present in the database then the system alerts the user to this fact and outputs the barcode string.

Even though the detection stage worked well at 320×240 resolution at around 15fps, for our experiments we used 640×480 resolution to be able to resolve more lines and read the barcode when it is not exactly aligned. In this mode, using a 2.4Ghz Intel Pentium processor with 2GB of RAM, our algorithm ran at up to 7fps (detection and decoding) without sound. However, due to the lag caused by the TTS (text-to-speech) system, in normal circumstances we are limited to only a few frames a second, which seemed to be sufficient for this experiment.

7. Experimental Results

We first experimented informally with our system to determine appropriate scanning strategies for locating barcodes as quickly as possible. Prior knowledge of the likely locations of barcodes on various types of packages proved to be helpful. For instance, on a cylindrical can, the barcode is always on the circumference of the cylinder near the flat top or bottom. Another rule we have inferred is that the barcode's orientation is always aligned to the dimensions of the box, and it is best to try to align the camera orientation to that of the barcode.

A few scanning strategies quickly became clear: (a) it is best to keep the package fixed and to move the camera relative to it; if a portion of the package's surface has been scanned without finding any barcodes, then the package should be rotated before continuing the scan; (b) the camera should be held several inches from the package surface and moved slowly to avoid motion blur; (c) the user must take care not to occlude any part of the package being scanned with his/her fingers; and (d) once the system reliably indicates the presence of a barcode, and the system has directed the user to home in on it, the user may have to wait a few to several seconds for the system to read the barcode.

To prove the feasibility of our combined detection and reading system, we conducted an experiment with a normally sighted subject who was wearing a blindfold the entire time. In a training session he successfully located and read all ten packages that he was given (consisting of cans, boxes and tubes); he was completely unfamiliar with four of the packages, but had previously seen the remaining six. Next, in the experiment session he was given ten packages, none of which he had seen before. He was able to locate all ten barcodes, and the system correctly read nine of the ten barcodes. (The incorrectly read barcode was correct except for two incorrect digits.) In our algorithm, we use the check digit in the barcode estimation, which creates a tradeoff between efficiency and accuracy. It is possible to tune the parameters so as to gain more speed by sacrificing some accuracy and vice versa. This may be left as a choice for the end user.

We also investigated the types of image regions that passed various stages of the detection process. Fig. 6 shows sample images captured by the webcam, with colored lines denoting the following: (a) red lines indicate regions that passed the line tally test but failed the entropy test; (b) yellow lines are for regions that passed the entropy test but had too few edges to be considered as a full barcode candidate; and (c) green lines indicate full barcode candidates that invoked the reading (decoding) algorithm.

Finally, we experimented with the webcam focus to see how far we could detect a barcode. By setting the focus near infinity, the algorithm was able to detect barcodes at distances of up to 12 inches. Unfortunately, this focus setting made close-up views of barcodes too blurry to be read. In the future we hope to use an autofocus lens to be able to focus well at all distance ranges, assuming it can be made to focus fast enough for real-time use.

8. Conclusion

We have described a novel algorithm for finding and reading 1D barcodes, intended for use by blind and visually impaired users. A key feature of the algorithm is the ability to detect barcodes at some distance, allowing the user to rapidly scan packages before homing in on a barcode. Experimental results with a blindfolded subject demonstrate the feasibility of the system.

In the future we plan to port our system to a camera phone, and to extend our system to symbologies other than UPC-A, such as the the EAN-13 (which is widespread in Europe).

Acknowledgments

The authors were supported by the The National Institutes of Health grant 1 R01 EY018890-01.

References

1. Adelmann, A.; Langheinrich, M.; Floerkemeier, G. A 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 2006; 2006.
2. Chai, D.; Hock, F. Locating and decoding EAN-13 barcodes from images captured by digital cameras. Information, Communications and Signal Processing, 2005 Fifth International Conference on; 2005. p. 1595-1599.
3. Gallo, O.; Manduchi, R. Reading challenging barcodes with cameras. IEEE Workshop on Applications of Computer Vision (WACV) 2009; Snowbird, Utah. December 2009;
4. Joseph E, Pavlidis T. Deblurring of bilevel waveforms. IEEE Transactions on Image Processing 1993;2
5. Kongqiao W. 1D barcode reading on camera phones. International Journal of Image and Graphics 2007;7(3):529–550.
6. Krešić-Jurić S, Madej D, Santosa F. Applications of hidden Markov models in bar code decoding. Pattern Recogn Lett 2006;27(14):1665–1672.
7. Ohbuchi, E.; Hanaizumi, H.; Hock, LA. Barcode readers using the camera device in mobile phones. CW '04: Proceedings of the 2004 International Conference on Cyber-worlds; Washington, DC, USA. 2004. p. 260-265.

8. Tekin, E.; Coughlan, J. Barcode project. www.ski.org/Rehab/Coughlan_lab/Barcode
9. Tekin, E.; Coughlan, JM. A bayesian algorithm for reading 1-D barcodes. Sixth Canadian Conference on Computer and Robot Vision (CRV 2009); Kelowna, BC, CA. May 2009;
10. Wachenfeld, S.; Terlunen, S.; Jiang, X. Robust recognition of 1-D barcodes using camera phones. Proceedings of the International Conference on Patern Recognition; 2008. p. 1-4.
11. Wang, H.; Zou, Y. Camera readable 2D bar codes design and decoding for mobile phones. Proceedings of the International Conference on Image Processing; 2006. p. 469-472.
12. Zhang, C.; Wang, J.; Han, S.; Yi, M.; Zhang, Z. Automatic real-time barcode localization in complex scenes. Proceedings of the International Conference on Image Processing; 2006. p. 497-500.

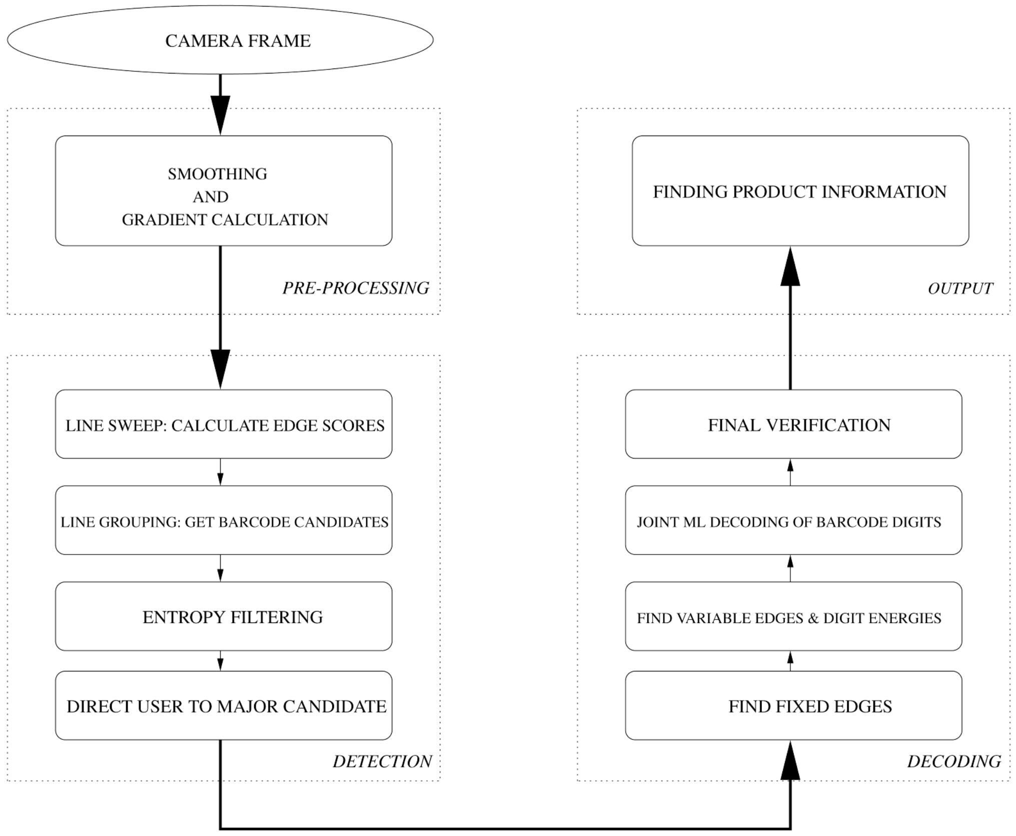
CAMERA FRAME
SMOOTHING AND GRADIENT CALCULATION
PRE-PROCESSING
LINE SWEEP: CALCULATE EDGE SCORES
LINE GROUPING: GET BARCODE CANDIDATES
ENTROPY FILTERING
DIRECT USER TO MAJOR CANDIDATE
DETECTION
FIND FIXED EDGES
FIND VARIABLE EDGES & DIGIT ENERGIES
JOINT ML DECODING OF BARCODE DIGITS
FINAL VERIFICATION
DECODING
FINDING PRODUCT INFORMATION
OUTPUT

Figure 1.
Barcode system.

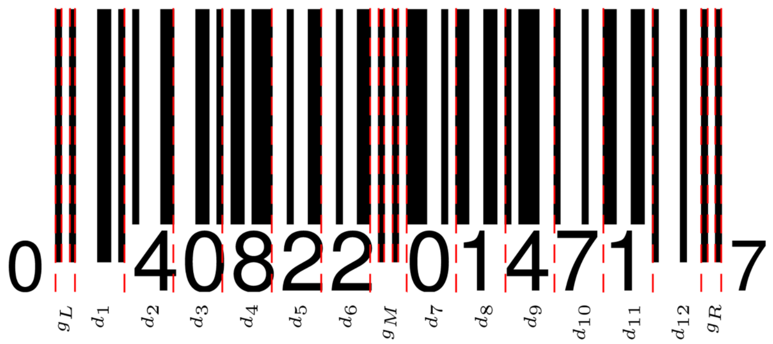

Figure 2.
UPC-A barcode, encoding 12 digits. The code axis runs left to right in this image and the bar axis runs vertically upwards. Note that the the bar patterns representing any specific digit have opposite polarity on the left and right sides of the barcode.

```
**INITIALIZATION:**
$\tau_G$ = minimum gradient threshold
$n_E$ = minimum # of edges required
$d_E$ = maximum distance between consecutive edges
**SWEEP:**
**for** orientation $t = 0, 45, 90, 135$ **do**
    **for** line $l = 1, \ldots, lastLineInThisOrientation$ **do**
        $count \leftarrow 0$
        **for** pixel $i = 1, \ldots, lastPixelOnThisLine$ **do**
            Let $j$ be the last pixel on this line that was counted.
            **if** $|\nabla I_i| > \tau_G$ **then**
                  //Gradient above threshold and angle approx. perpendicular to sweep line
                **if** $\angle \nabla I_i \approx \perp$ orientation **then**
                    **if** $|\nabla I_i| \geq \max\{|\nabla I_{i-1}|, |\nabla I_{i+1}|\}$ **then**    //non-maximum suppression
                        **if** $\angle \nabla I_i$ is $\approx 180$ degrees out of phase with $\angle \nabla I_j$, and $d_{ij} < d_E$ **then**
                            $count \leftarrow count + 1$    //count this pixel
                **else**
                    $count \leftarrow count - 1$    //pixel with strong gradient at wrong orientation
            **else if** $d_{ij} > d_E$ **then**    //no edge pixel seen in a while
                $count \leftarrow count - 1$;
                **if** $d_{ij} > 2 * d_E$ **then**    //no edges in a long while
                    $count \leftarrow 0$    //end of candidate segment
            **if** $count = 0$ **then**    //see if end of segment has been reached
                $score \leftarrow \max_{i \in lastSegment} count(i)$    //score is the max count for this segment
                **if** $score > n_E$ **then**    //if the minimum # edges has been seen
                    Record this segment as a barcode candidate segment for this line
                **else**
                    Discard this segment
```

Figure 3.
Line Scan Algoritm

```
INITIALIZATION:
$n_L =$ minimum # of lines required
$d_L =$ maximum distance between matching lines
$\tau_S =$ minimum score required to declare barcode candidate
$\mathcal{B} = \{\}$    //list of candidate areas
SWEEP:
for orientation $t = 0, 45, 90, 135$ do
  for line $l = 0, \ldots, lastLineInThisOrientation$ do
    for barcode segment candidate $c = 1, \ldots, lastCandidateOnThisLine$ do
      if $\exists b \in \mathcal{B}$: $begin_b \approx begin_c, end_b \approx end_c$ and $d_{lb} < d_L$ then
        //There was a previous barcode area candidate with similar beginning and end a little earlier
        $count_b \leftarrow count_b + 1$
      else
        $\mathcal{B} \leftarrow \mathcal{B} + c$    //Add this segment as the beginning of a new candidate area
        $count_c \leftarrow 1$
      for $b \in \mathcal{B}$ do    //check that the candidates are still valid
        if $d_{lb} > d_L$ then    //have not seen a match in a while
          $count_b \leftarrow count_b - 1$
          if $d_{lb} > 2 * d_L$ then    //have not seen a match in a long while
            $count_b \leftarrow 0$
        if $count_b = 0$ then    //end of candidate
          $score_b \leftarrow max_{l \in b} count(l)$    //score is the max count over all lines in this area
          if $score_b > \tau_S$ then    //if the minimum # lines has been seen
            Record $b$ as a barcode area segment
          else
            $\mathcal{B} \leftarrow \mathcal{B} - b$    //Discard this candidate
```

Figure 4.
Line Tally Algorithm

```
INITIALIZATION:
$X_{initial} = \frac{lastEdge - firstEdge}{95}$
FIND EDGES AND DIGIT PROBABILITIES:
Find the $N_{slices}$ lines in the barcode with the highest edge count
for Slice $i = 1, \ldots, N_{slices}$ do
    Estimate $N_{fixedEdgeEstimates}$ fixed edges
    for Fixed Edge Estimate $j = 1, \ldots, N_{fixedEdgeEstimates}$ do
        for Barcode digit $d = 1, \ldots, 12$ do
            Get digit probabilities for each numeric digit $0, \ldots, 9$
    Marginalize digit probabilities over all fixed edge estimates
Marginalize digit probabilities over all slices
BARCODE ESTIMATION:
for Barcode digit $d = 1, \ldots, 12$ do
    Calculate auxiliary running parity check digit probabilities.
ML Estimation of the 2 most likely sequence of auxiliary random variables
Convert auxiliary random variables back to barcode digits
BARCODE VERIFICATION:
if Probability of most likely sequence $> K \times$ Probability of the second most likely sequence then
    if Most likely sequence differs from individually most likely digits by at most 1 digit then
        OUTPUT BARCODE
Get new frame
```

Figure 5.
Barcode Decoding Algorithm

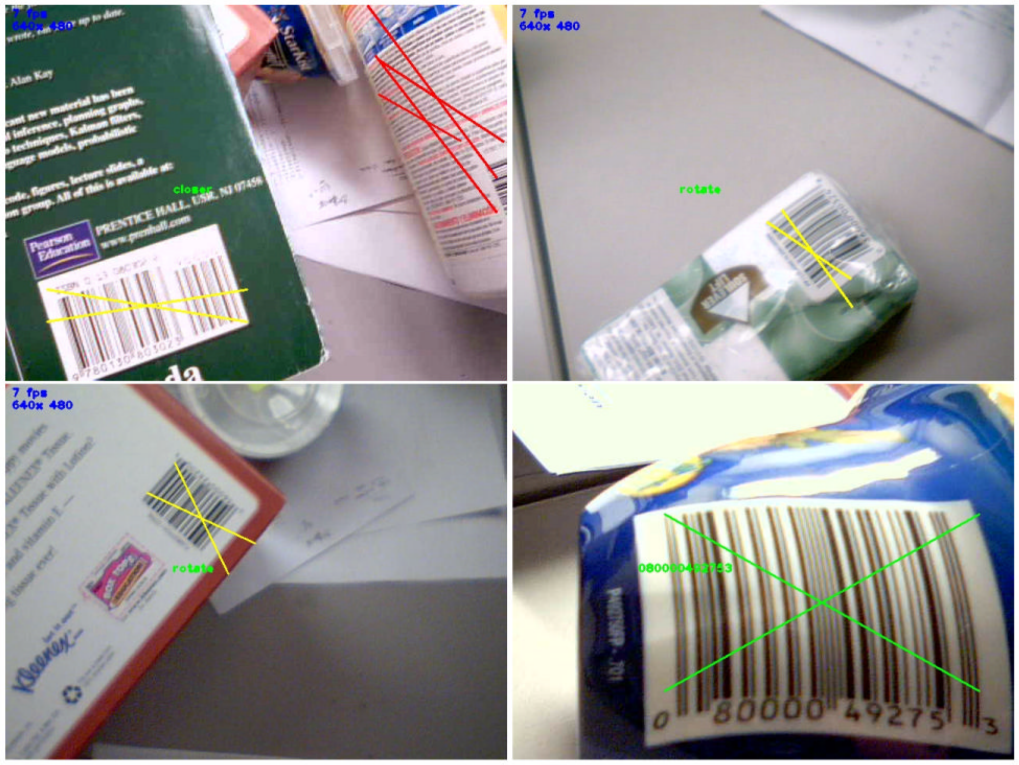

Figure 6.
Sample intermediate detection stages. Red, yellow and green lines denote regions that pass successively more stringent detection tests (see text for details). Audio instructions issued to user (such as “closer”) are printed in green type.