TOWARD REAL TIME EYES-FREE BARCODE SCANNING ON SMARTPHONES
IN VIDEO MODE

Aliasgar Kutiyanawala¹ Vladimir Kulyukin¹ John Nicholson²
¹Department of Computer Science, Utah State University. ²Department of
Computer Science and Information Technology, Austin Peay State University

INTRODUCTION

In our previous research [1], we have shown that visually impaired (VI) individuals
Can shop independently by scanning MSI barcodes on shelves and UPC barcodes on
products. This system was called ShopTalk and used a hand-held barcode scanner, a shoulder-
mounted keypad and headphones connected to an ultra-portable OQO computer. ShopMobile-II
[2, 3] is the next version of this system that
reduces the system's hardware complexity by
allowing VI users to shop independently using
only a smartphone.

ShopMobile-II has three software modules –
an eyes-free barcode scanner, an OCR engine
and remote guidance. This paper describes our
barcode scanning algorithm that operates in
video mode on the Google Nexus One Android
2.2 smartphone. Two pilot experiments are
presented: the first one, conducted in a grocery
store evaluated the contributions of the
algorithm’s modules; the second one, performed in a laboratory with two VI users,
evaluated the effectiveness of the algorithm in
finding UPC barcodes on various grocery
products.

EYES-FREE BARCODE SCANNER

RedLaser [4] and ZXing [5] are two
applications for scanning barcodes on
smartphones. However, they require users to
carefully align the camera with the barcodes
prior to decoding and cannot decode MSI
barcodes. Our algorithm is designed to find
both UPC and MSI barcodes in real video mode
without prior camera alignment [6].

The eyes-free barcode scanning algorithm is
comprised of three modules – interactive
camera alignment loop, barcode localization,
and barcode decoding. The barcode scanning
algorithm operates in video mode. Images are
taken continuously and processed by the
barcode localization and decoding modules. If a
barcode is decoded, the user is notified through
text-to-speech.

Interactive Camera Alignment Loop

We have previously conducted a lab study
with one VI participant and four blindfolded
sighted participants, where participants had to
find, retrieve and verify products from a
simulated shopping aisle [2]. The participants
scanned MSI barcodes on shelves to find
product locations and UPC barcodes on
products to identify them. MSI barcodes are a
type of linear 1-D barcodes that are used
mostly for inventory control, and marking
storage containers and shelves in warehouse
environments [7]. To scan a barcode,
participants would first align the camera with
the product or shelf and then slowly move the
camera away. The VI participant would
frequently misalign the camera in the pitch and
yaw planes as he moved it away from
barcodes. This misalignment caused skew
distortions in barcode images, which resulted in
several decoding failures.

The interactive camera alignment module is
designed to minimize skew distortions by
keeping the camera aligned with the barcode in
the pitch and yaw frames. The shopper starts
the loop by pressing the touch screen. The
system captures the phone's orientation
sensor's readings of the pitch and yaw planes
for subsequent reference. The system takes
these readings as the shopper moves the
camera away from the barcode and compares
them with the reference readings. When the
readings deviate from the reference readings,
the phone begins to vibrate until the camera is
realigned.
Barcode Localization

The area of the barcode region in the image varies with the distance between the barcode and the camera. If the camera is held close to the barcode, the barcode region occupies a large area in the image and the barcode decoding stage can easily decode the barcode. However, if the camera is held at a distance to the barcode, the barcode region is small and there is a possibility of other components, such as text and graphics, being alongside the barcode, which may cause a detection failure. Localizing and segmenting possible barcode regions prior to decoding increases the probability of accurate decoding.

A barcode is a small homogeneous region consisting of alternate black and white lines. We define two properties – alternating frequency and vertical continuity that characterize a region as a potential barcode. If a line is drawn horizontally across the barcode, alternating frequency is the number of black to white and white to black transitions along that line. Vertical continuity is the continuity of black and white lines along vertical lines. Vertical continuity is estimated by the similarity between two parallel horizontal lines, separated vertically, placed over the region. The longest common subsequence is used as the similarity measure.

The first step in the barcode localization is to scale the image down to the 320 x 240 resolution for efficiency. This scaled image is passed through a line filter presented in [2], which allows vertical lines to pass through and filters out everything else. The filtered image is scanned in a rasterized pattern to look for areas with high alternating frequencies and vertical continuities. These are translated into the corresponding areas of the original image and segmented from it.

Barcode Decoding

Our barcode decoding algorithm uses scanlines and consists of three main procedures – strip generation, strip processing, and scanline decoding. The strip generation procedure selects horizontal and diagonal n-pixel wide strips in the image. The strips are converted to one-pixel wide scanlines. This conversion is achieved through three methods – luminance based method (LM), two-level binary method (TLB) and a two-level binary method with alternating frequency filter (TLB-AF).

LM populates the scanline with the central (n/2) row of the strip, and each pixel in the scanline p, ∈ [0, 255]. TLB binarizes the strip using a modified Niblack filter [8] before populating the scanline with the central row of the strip. Each pixel in the scanline pi ∈ {0, 255}. TLB-AF takes this process a step further by filtering out areas with low alternating frequencies in the binarized strip before populating the scanline.

The processed scanline is now converted to a line-widths representation W = {w1, w2, ..., wn}. To obtain W, strings of consecutive zeros and ones in the scanline are replaced with their run lengths. A separate data structure C = {c1, c2, ..., cn} records the color associated with each line-width wi. For example, if the scanline S = {255, 0, 0, 255, 255}, W = {1, 3, 2} and C = {1, 0, 1}.

Table 1: UPC barcode symbology in line-widths representation.

<table>
<thead>
<tr>
<th>Pattern / Digit</th>
<th>Template of line widths Tm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Pattern</td>
<td>111</td>
</tr>
<tr>
<td>Middle Pattern</td>
<td>11111</td>
</tr>
<tr>
<td>End Pattern</td>
<td>111</td>
</tr>
<tr>
<td>Digit 0</td>
<td>3211</td>
</tr>
<tr>
<td>Digit 1</td>
<td>2221</td>
</tr>
<tr>
<td>Digit 2</td>
<td>2122</td>
</tr>
<tr>
<td>Digit 3</td>
<td>1411</td>
</tr>
<tr>
<td>Digit 4</td>
<td>1132</td>
</tr>
<tr>
<td>Digit 5</td>
<td>1231</td>
</tr>
<tr>
<td>Digit 6</td>
<td>1114</td>
</tr>
<tr>
<td>Digit 7</td>
<td>1312</td>
</tr>
<tr>
<td>Digit 8</td>
<td>1213</td>
</tr>
<tr>
<td>Digit 9</td>
<td>3112</td>
</tr>
</tbody>
</table>

A UPC barcode consists of two sets of six digits D enclosed by start (S)/end (E) patterns and separated by a middle pattern (M) as SDddddDMDDDDDDE [9]. The start, middle and end patterns as well as the digits are encoded by a series of alternating black and white lines of varying widths. Table 1 shows the
UPC barcode symbology in the line-widths representation. To decode the barcode in the scanline $W$, we find the start pattern index ($s$) and the end pattern index ($e$) within $W$ using the following equations:

$$
\begin{align*}
    s &= \arg \max (w_i - \text{std}(w_i, w_{i+1}, w_{i+2})) \\
    e &= \arg \max (w_i - \text{std}(w_i, w_{i+1}, w_{i+2}))
\end{align*}
$$

where, $c_i = 0$.

Each digit $d$ is encoded by four lines and the index $i$ of the $j^{th}$ digit $d_j$ within $W$ can be found as:

$$
\begin{align*}
    i &= s + (j - 1) \times 4, \text{if } j \leq 6 \\
    i &= s + (j - 1) \times 4 + 5, \text{if } j > 6
\end{align*}
$$

If $T_m$ represents the template of the digit $m$ in Table 1., the value of the $j^{th}$ digit $d_j$ can be found as follows:

$$
d_j = \arg \min \left( \sum_{k=0}^{3} (w_{i+k} - T_k^m)^2 \right)
$$

where, $0 \leq j \leq 12$ and $0 \leq m \leq 9$.

The twelfth digit is a checksum digit to verify accurate decoding.

**EXPERIMENTS**

The objective of the first experiment was to find the contribution of barcode localization as well as the contributions of the three strip processing methods. Our claim is that barcode localization prior to decoding increases the decoding rate. To test this claim, a database of 68 product images, which included boxes, cans, and bottles, was obtained in a grocery store with the Google Nexus One smartphone with Android 2.2. We then logged the number of barcodes decoded in the scans with and without the localization stage. As Fig. 1 shows, localization prior to decoding resulted in an increase of 14 (20.59%) decoded barcodes.

As mentioned earlier, the three methods are used to convert n-pixel wide strip to one-pixel wide scanlines. We wanted to test the contributions of each method to decoding. As shown in Fig. 2, we found that the LM decoded the most barcodes (53), followed by TLB (26) and finally TLB-AF (15). Only one barcode out of the 15 decoded by TLB-AF is not decoded by the other methods. Hence, this method can be eliminated without a significant impact on the overall decoding rate.

![Figure 1. Contributions of the barcode localization stage.](image1)

![Figure 2. Contributions of the different strip processing methods.](image2)

To test the accuracy of barcode recognition in video mode, we conducted two single subject experiments at the Smith Kettlewell Eye Research Institute in San Francisco, CA. The software was implemented on the Google Nexus One smartphone with Android 2.2. Both subjects were completely blind staff electrical engineers. One had a cellphone; the other never owned or used one. The camera alignment module was not tested. The camera operated in video mode.
The experiment consisted of a tutorial and an actual test. In the tutorial, each subject was given two sample products (a plastic bottle and a juice box) and was shown how to move the camera along the surface of the product. The tutorial continued until each subject could detect the barcodes on both products without the experimenter’s help. The first subject’s tutorial took approximately ten minutes. The second subject’s tutorial took approximately eight minutes. During the test, each subject was given ten products: a cereal box, a small tea box, two small juice bottles, a small milk carton, a Pringles tube, a toothpaste box, a larger juice bottle, a small water bottle, and a yogurt cup. The detection time for each product was recorded on the phone. When the subject could not detect the barcode for over five minutes, the detection was considered a failure and the subject was given the next product. The order of the products was randomized for each subject.

In informal feedback collected from both subjects after the experiments, they indicated a preference for run time image alignment signals when the camera is misaligned with the object. They also said that the camera should work over a wider range of positions and the software must operate much faster.

**CONCLUSION**

Our experiments indicate that smartphones can be used for real time eyes-free barcode scanning in video mode. Barcode localization was observed to improve the decoding rates by as much as 20%. The TLB-AF can be eliminated without a significant reduction in barcode decoding rates. Two VI participants were able to successfully scan most UPC barcodes on various grocery products using our algorithm.

**REFERENCES**


**Table 2**: Results of barcode scanning experiments with VI participants.

<table>
<thead>
<tr>
<th>Product</th>
<th>Participant 1</th>
<th>Participant 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time Taken</td>
<td>False Positives</td>
</tr>
<tr>
<td>Cereal box</td>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>Tea box</td>
<td>140</td>
<td>0</td>
</tr>
<tr>
<td>Small juice bottle 1</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Small juice bottle 2</td>
<td>142</td>
<td>0</td>
</tr>
<tr>
<td>Milk carton</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>Pringles</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>Toothpaste</td>
<td>184</td>
<td>0</td>
</tr>
<tr>
<td>Large juice bottle</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td>Water bottle</td>
<td>87</td>
<td>0</td>
</tr>
<tr>
<td>Yogurt cup</td>
<td>17</td>
<td>0</td>
</tr>
</tbody>
</table>

The average barcode recognition times were 83.6 seconds and 93.4 seconds for subject 1 and 2, respectively. Subject 2 had two failures where the software failed to detect the barcode in five minutes. There were also two false positives when the software detected a barcode over printed text.