Reading LCD/LED Displays with a Camera Cell Phone

Huiying Shen and James Coughlan
Smith-Kettlewell Eye Research Institute
San Francisco, CA 94115
{hshen, coughlan}@ski.org

Abstract

Being able to read LCD/LED displays would be a very important step towards greater independence for persons who are blind or have low vision. A fast graphical model based algorithm is proposed for reading 7-segment digits in LCD/LED displays. The algorithm is implemented for Symbian camera cell phones in Symbian C++. The software reads one display in about 2 seconds by a push of a button on the cell phone (Nokia 6681, 220 MHz ARM CPU).

1. Introduction

Electronic appliances with LCD/LED displays have become ubiquitous in our daily lives. While they offer many conveniences to sighted people, blind people and those with low vision have difficulty using them. One way to alleviate this problem is to develop a device that will read the display aloud for the targeted users. One candidate for implementing such a device is the camera cell phone, or "smart phone".

So-called smart phones are actually small computers with formidable computational power. For example, the Nokia 6681 has a 220 MHz ARM CPU and 22 MB RAM. It also has a 1.3M pixel camera. Compared to the processing power afforded by typical desktop computers used in computer vision, however, the smart phone has substantially less processing power. In our experience, integer-based calculations are over an order of magnitude slower on a cell phone than on a typical desktop computer. Moreover, cell phones do not have a floating point processing unit (FPU) but use a software-simulated FPU instead to do floating point calculations, which are slower still. Thus, a computer vision algorithm implemented on a cell phone must work within significant computational constraints in order to be practical.

We address these constraints by choosing an application that is less computationally demanding than typical state-of-the-art computer vision applications designed to run on non-embedded systems: our domain is restricted to close-up images of LCD/ LED numeric displays, with only modest amounts of clutter that is typically confined to areas in the image a small distance away from the LED/ LCD characters.

We are developing a software application for Symbian cell phones, e.g. Nokia 7610, Nokia 6681/6682, to read seven-segment LCD displays. The user will push the OK button to take a picture, and the application will read out the digits on the display in digitized or synthetic speech.

2. Choice of Platform

Using the cell phone as a platform for this application offers many important advantages. The first is that it is inexpensive and most people already have one – no additional hardware needs to be purchased. This is particularly important since many visually impaired people have limited financial resources (unemployment among the blind is estimated at 70% [8]). The camera cell phone is also portable and becoming nearly ubiquitous; it is multi-purpose and doesn't burden the user with the need to carry an additional device. Another advantage of the cell phone is that it is a mainstream consumer product which raises none of the cosmetic concerns that might arise with other assistive technology requiring custom hardware [9].

Our past experience with blind people shows that they can hold a cell phone camera roughly horizontal and still enough to avoid motion blur, so that satisfactory images can be taken without the need for a tripod or other mounting.

We have chosen to use cell phones using the Symbian operating system for several reasons. First, Symbian cell phones (most produced by Nokia) have the biggest market share. Second, the Symbian operating system and C++ compiler are open and well documented, so that anyone can develop software for Symbian OS. In the future we plan to allow open access to our source code, which will allow other researchers and developers to modify or improve our software. Finally, the camera API is an integrated part of the OS, which allows straightforward control of the image acquisition process.

We note that the cell phone platform allows us to bypass the need for manufacturing and distributing a physical product altogether (which is necessary even for custom hardware assembled using off-the-shelf components). Our final product will ultimately be an executable file that can be downloaded for free from our website and installed on any Symbian camera phone.

3. Related Work

We are aware of no published work specifically tackling the problem of reading images of LCD/LED displays, although this function has been proposed for a visual-to-auditory sensory substitution device called The vOICe [10], and a commercial product to perform this task is under development at Blindsight [1]. A large body of work addresses the more general problem of detecting and reading printed text, but so far this problem is considered solved only in the domain of OCR (optical character recognition). This domain is limited to the analysis of high-resolution, high-contrast images of printed text with little background clutter. Recently we have developed a camera cell phone-based system to help blind/low vision users navigate indoor environments [4], but this system requires the use of special machine-readable barcodes.

The broader challenge of detecting and reading text in highly cluttered scenes, such as indoor or outdoor scenes with informational signs, is much more difficult and is a topic of ongoing research. We draw on a common algorithmic framework used in this field of research, in which bottom-up processes are used to group text features into candidate text regions using features such as edges, color or texture [5,6,7,14], in some cases using a filter cascade learned from a manually segmented image database [2].

Our approach combines a bottom-up search for likely digit features, based on grouping sets of simple, rapidly detected features, with a graphical model framework that allows us to group the candidate features into figure (i.e. target digits) and ground (clutter). This framework is based on graphical models that are *data-driven* in that their structure and connectivity is determined by the set of candidate text features detected in each image. Such a model provides a way of pruning out false candidates using the context of nearby candidates. Besides providing a natural framework for modeling the role of context in segmentation, another benefit of the graphical model framework is the ability to learn the model parameters automatically from labeled data (though we have not done this in our preliminary experiments).

Recent work related to ours also uses a graphical model framework for text segmentation in documents [18] and in natural scenes [17]. Unlike our approach, these works require either images with little clutter or colored text to initiate the segmentation. By contrast, we have designed our algorithm to process cluttered grayscale images without relying on color cues, since digits come in a variety of colors (black for LCDs and green, blue or red for LEDs).

4. Algorithm

An example of a picture of an LCD display is shown in Fig 1. The display has low contrast, and the LCD digits are surrounded by clutter such as the display case and controls. Our goal is to construct an algorithm to find and read the group of 7-segment digits in the image.

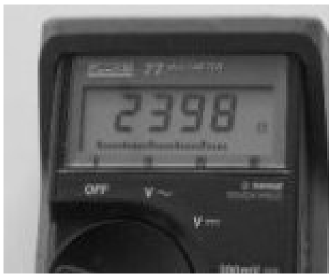

Figure 1: An electronic current/voltage meter.

It can be seen from Fig. 1 that 1) all the digits are of similar height *(h)* and width *(w),* 2) digits are horizontally next to each other and 3) neighboring digits are approximately at the same level. One can also see that for each digit, the ratio w/h is a number around 0.5. Our algorithm will exploit these observations.

4.1. Feature Extraction and Building

Compared to today's powerful desktop computers, a cell phone has very limited computational resources. Complex feature extraction algorithms and those using extensive floating point computations must be avoided. Therefore, we will only extract simple features, and build up needed features hierarchically.

The basic features we are extracting from the image are horizontal and vertical edge pixels. Each has two polarities: from light to dark, and from dark to light. Fig. 2 shows horizontal edge pixels of two polarities: green pixels are edge transitions from light to dark (traversing the image downwards), and the blue are ones from dark to light. The edge pixels are determined by finding local maxima and minima in the horizontal and vertical derivatives of the image intensity.

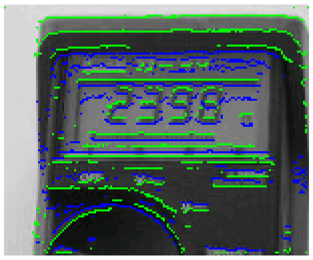

Figure 2: Horizontal edge pixels of two polarities: green for edges from light to dark, going downwards, and blue for ones from dark to light.

When two edge pixels of opposite polarities are next to each other, we construct an edge pair pixel. In Fig. 2, when there is a green pixel right above blue pixel, one can find a horizontal edge pair pixel, shown in yellow, in Fig. 3.

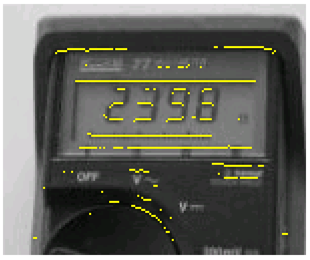

Figure 3: Horizontal edge pair pixels: when two edge pixels of opposite polarities are next to each other, an edge pair pixel is constructed, located between them.

We can group horizontal edge pair pixels into horizontal strokes. Similarly, we can find vertical strokes. Fig. 4 shows both horizontal strokes (yellow) and vertical ones (red). Note that long strokes are not shown in Fig. 4, as they are too large for the scale of digits we are looking for and are eliminated from further consideration.

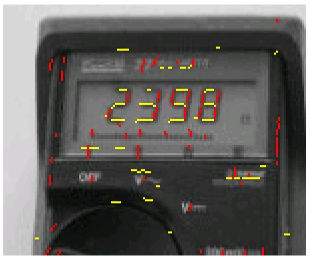

Figure 4: Horizontal (yellow) and vertical (red) strokes.

When vertical and horizontal strokes are sufficiently close, we can construct stroke clusters, as shown in Fig. 5. These stroke clusters serve as candidates for 7-segment digits.

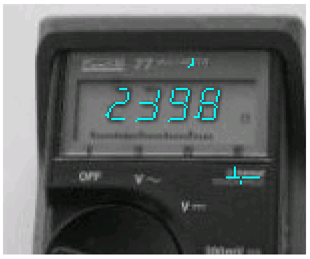

Figure 5: Stroke clusters: when vertical and horizontal strokes are close to each other, stroke clusters are constructed.

4.2. Figure-Ground Segmentation

While simple clustering gives good segmentation results in many cases, there are still false positives that need to be eliminated (as well as some false negatives to be "filled in"). We use a figure-ground segmentation algorithm to eliminate the false positives from the clustering results, building on our previous work on detecting pedestrian crosswalks [3]. This approach was inspired by work on clustering with graphical models [11], normalized cut-based segmentation [12] and object-specific figure-ground segmentation [16]. In this study, a data-driven graphic model is constructed for each image, and belief propagation is used for figure-ground segmentation of stroke clusters. This technique may be overly complex for the images shown in this paper, but we anticipate that it will be useful for noisier images taken by blind users, and it will be straightforward to extend to alphanumeric characters in the future.

Each stroke cluster, represented by its bounding rectangle (x_{min}, y_{min}, x_{max}, y_{max}), defines a node in the data-driven graph. Two nodes interact with each other when they are close enough. The goal of the figure-ground process is to assign "figure" labels to the nodes that belong to the target (LED/LCD digits) and "ground" labels to the other nodes.

4.3. Belief Propagation for Fixed Point Computation

Most embedded systems, including handheld computers and smart cell phones, do not have a floating point processing unit (FPU). Symbian cell phones are no exception. Symbian OS does have a software simulated FPU, but this is one to two orders of magnitude slower than integer computation.

Traditional belief propagation (BP) algorithms are computationally intensive, and typically require floating point computation. In this study, we perform max-product BP [15] in the log domain so that all message updates can be performed with addition and subtraction. Further, the messages can be approximated as integers by a suitable rescaling factor, so that *only integer arithmetic is required*.

The max-product message update equation is expressed as follows:

$$m_{ij}(x_j) = c_{ij} \max_{x_i} \{\psi_{ij}(x_i, x_j)\psi_i(x_i) \prod_{k \in N(j)\backslash i} m_{ki}(x_i)\}$$

where $m_{ij}(x_j)$ is the message from node *i* to node state x_j of node *j*. $\psi_{ij}(x_i, x_j)$ is the compatibility function between state x_i of node *i* and x_j of node *j*. $\psi_i(x_i)$ is unitary potential of node *i* for state x_i. *N(j)* is the set of nodes neighboring (i.e. directly connected to) node *j,* and *N(j)\i* denotes the set of nodes neighboring *j* except for *i.* c_{ij} is an arbitrary normalization constant.

Taking the log of both sides of the equation, we have:

$$L_{ij}(x_j) = \max_{x_i}\{E_{ij}(x_i, x_j) + E_i(x_i) + \sum_{k\in N(j)\setminus i} L_{ki}(x_i)\} + z_{ij}$$

where $L_{ij}(x_j)$ = log($m_{ij}(x_j)$), $E_{ij}(x_i, x_j)$ = log($\psi_{ij}(x_i, x_j)$), $E_i(x_i)$ = log($\psi_i(x_i)$), and z_{ij} = log(c_{ij}). z_{ij} is chosen such that $L_{ij}(x_j)$ will not over-or underflow.

In our figure-ground segmentation application, each node *i* has only two possible states: x_i=0 for the ground state and x_i=1 for the figure state.

One can see from the equation above that only addition/subtraction is needed for message updating. For C++, which we choose to use on the Symbian cell phone, we can perform the addition/subtraction using only integer calculations and no floating point. This allows the algorithm to run fast enough to be practical on the cell phone.

4.4. Unitary Energy

The unitary energy $E_i(x_i)$ represents how likely node *i* is to be at state x_i. Without losing generality, we set $E_i(x_i = 0) = 0$ for all nodes in the graph, since only the difference between $E_i(x_i = 0)$ and $E_i(x_i = 1)$, matters. As stated previously, each stroke cluster is represented by a rectangle (x_{min}, y_{min}, x_{max}, y_{max}), and its width and width are $w = x_{max}$,- x_{min}, and $h = y_{max}$ - y_{min} , respectively.

For the figure state, $E_i(x_i = 1)$ represents how likely a stroke cluster is a 7-segment digit by looking at the cluster itself. We use the width/height ratio *(R_{wh})* to determine this value: $E_i(x_i = 1)$=0 when R_{wh}>0.3 and R_{wh}<0.6, $E_i(x_i = 1)$=0.5 when R_{wh}>0.6 and R_{wh}<1.0, and $E_i(x_i = 1)$=2.0 otherwise.

4.5. Binary Energy

Binary $E_{ij}(x_i, x_j)$ represents the compatibility of node *i* having state x_i and node *j* having state x_j. Since $E_{ij}(x_i = 0, x_j = 0)$, the ground-ground energy, and $E_{ij}(x_i = 0, x_j = 1)$, the ground-figure energy, are difficult to learn, we will set them to the same constant, E_b (say, 1.5) for all the nodes.

$E_{ij}(x_i = 1, x_j = 1)$ represents how likely it is that nodes *i* and *j* are both figure.

$$E_{ij}(x_i = 1, x_j = 1) = c_x\Delta x + c_y\Delta y + c_h\Delta h + c_w\Delta w$$

where

$$\Delta x = \min(|x^i_{\min} - x^j_{\max}|, |x^i_{\max} - x^j_{\min}|),$$

$$\Delta y = \min(|y^i_{\min} - y^j_{\min}|, |y^i_{\max} - y^j_{\max}),$$

$$\Delta h = \min(|h^i - h^j|, |h^i - 2h^j|, |2h^i - h^j|),$$

$$\Delta w = |h^i - h^j|.$$

The *c*'s are coefficients to be determined by experience and/or statistical learning. There is a cutoff value for $E_{ij}(x_i = 1, x_j = 1)$: when it is greater than E_b, it is set to E_b. In other words, when node *i* and *j* can't send positive messages to help each other be classified as "figure", they don't say anything negative either.

4.6. Read the Digits

After stroke clusters are identified as figure, they are mapped to the 7-segment template, see Fig. 6.

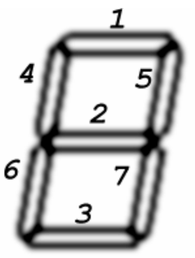

Figure 6: Seven-segment digit template. The numbers in the image indicate the ordering of the segments.

A mapping result is a series of seven 0's and 1's, with 1's indicating the stroke exists, and 0 indicating the stroke is missing. For example, a mapping result of '1110101' indicates that strokes 4 and 6 are missing, which consequently means the digit is '3'. '1111011' means the digit is a '6'.

To determine each digit, each string of 0's and 1's is matched to the digit with the most similar sequence. Sometimes a segment can be missing (i.e. false negative). In this case the cluster is then mapped to the closest digit. For example, the cluster on top of the digit 3 in Fig. 5 is missing segment 1, and the mapping result will be '0110101'. Still it is best mapped to digit '3'.

5. Results

The algorithm is implemented and installed on a Nokia 6681 cell phone. The executable .SIS file (compiled on a desktop computer) is only about 73 KB, which means that it leaves plenty of space on the cell phone's flash memory for other applications and data. After the application is launched, it is in video preview mode: the screen shows that the camera is capturing. (The display is used for debugging purposes but obviously may not be useful for a low vision or blind user.) When the user pushes the OK button, the software will take a picture, run the display reader algorithm, and read aloud the numbers on the screen. (This is currently done using pre-recorded .wav files for each digit, but a text-to-speech system suitable for the Symbian OS will be used in the future.) The whole process takes approximately 2 seconds.

We show several results in Fig. 7. Note that the displays are only roughly horizontal in the images. There are few false positives, and those that occur (as in the last image in Fig. 7) are rejected by the digit-reading algorithm.

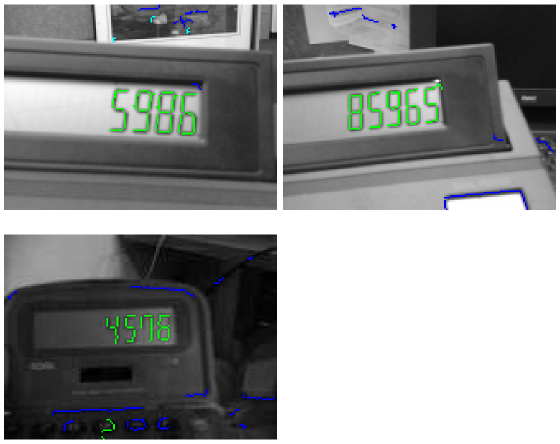

Figure 7: Experimental results for LCD displays. Stroke clusters assigned to "figure" are shown in green and "ground" in blue. False positive in bottom of last image is rejected by the algorithm for reading individual digits.

We also show a result for an LED display in Fig. 8. In order to read this display, the image contrast was manually inverted so that the digits became dark on a light background, the same as for LCD digits. In the future we will search for digits with both image polarities so that both types of display are accommodated.

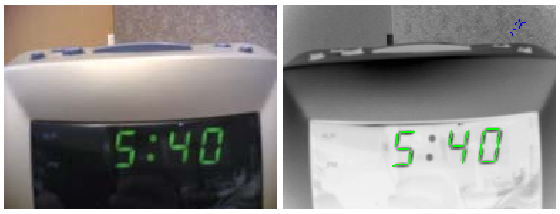

Figure 8: Experimental result for LED display. Left: original image. Right: results (same convention as in previous figure).

6. Summary and Discussion

Being able to read LCD/LED displays would be a very important step to help blind/low vision persons gain more independence. This paper presents an algorithm to perform this task, implemented on a cell phone. It reads 7-segment LCD/LED digits in about 2 seconds by the push of a button on the phone.

The algorithm extracts only very simple features (edges in four orientations) from the image, and builds up complex features hierarchically: edge pairs, vertical and horizontal strokes, and stroke clusters. A data-driven graph is constructed and a belief propagation (BP) algorithm is used to classify stroke clusters as figure or ground. The stroke clusters labeled as "figure" are read by matching them to digit templates (0 through 9).

Future work will include thorough testing of the algorithm by blind and visually impaired users, who will furnish a dataset of display images that will be useful for improving and tuning the algorithm. We also are in the process of extending the figure-ground framework to handle alphanumeric displays, as well as to detect text signs in natural scenes, such as street names and addresses.

Acknowledgments

We would like to thank John Brabyn for many helpful discussions. The authors were supported by the National Institute on Disability and Rehabilitation Research (grant no. H133G030080), the National Science Foundation (grant no. IIS0415310) and the National Eye Institute (grant no. EY015187-01A2).

References

[1] http://www.blindsight.com
[2] X. Chen and A. L. Yuille. ``Detecting and Reading Text in Natural Scenes." CVPR 2004.
[3] J. Coughlan and H. Shen. "A Fast Algorithm for Finding Crosswalks using Figure-Ground Segmentation." 2nd Workshop on Applications of Computer Vision, in conjunction with ECCV 2006. Graz, Austria. May 2006.
[4] J. Coughlan, R. Manduchi and H. Shen. "Cell Phone-based Wayfinding for the Visually Impaired." 1st International Workshop on Mobile Vision, in conjunction with ECCV 2006. Graz, Austria. May 2006.
[5] J. Gao and J. Yang. ``An Adaptive Algorithm for Text Detection from Natural Scenes." CVPR 2001.
[6] A.K. Jain and B. Tu. ``Automatic Text Localization in Images and Video Frames." Pattern Recognition. 31(12), pp 2055-2076. 1998.

[7] H. Li, D. Doermann and O. Kia. Automatic text detection and tracking in digital videos. IEEE Transactions on Image Processing, 9(1):147-156, January 2000.
[8] The National Federation for the Blind. “What is the National Federation of the Blind?” http://www.nfb.org/whatis.htm
[9] M. J. Scherer. ``Living in the State of Stuck: How Assistive Technology Impacts the Lives of People With Disabilities." Brookline Books. 4th edition. 2005.
[10] http://www.seeingwithsound.com/ocr.htm
[11] N. Shental, A. Zomet, T. Hertz and Y. Weiss. ``Pairwise Clustering and Graphical Models." NIPS 2003.
[12] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905, August 2000.
[13] http://www.seeingwithsound.com/voice.htm
[14] V. Wu, R. Manmatha, and E. M. Riseman. Finding Text In Images. Proc. of the 2nd intl. conf. on Digital Libraries. Philadaphia, PA, pages 1-10, July 1997.
[15] J.S. Yedidia, W.T. Freeman, Y. Weiss. ``Bethe Free Energies, Kikuchi Approximations, and Belief Propagation Algorithms". 2001. MERL Cambridge Research Technical Report TR 2001-16.
[16] S. X. Yu and J. Shi. ``Object-Specific Figure-Ground Segregation." CVPR 2003.
[17] D.Q. Zhang and S.F. Chang, ``Learning to Detect Scene Text Using a Higher-Order MRF with Belief Propagation." CVPR 04.
[18] Y. Zheng, H. Li and D. Doermann, ``Text Identification in Noisy Document Images Using Markov Random Field." Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003).