Detecting and Locating Crosswalks using a Camera Phone

Volodymyr Ivanchenko, James Coughlan and Huiying Shen
The Smith-Kettlewell Eye Research Institute
San Francisco, CA 94115
vivanchenko@ski.org

Abstract

Urban intersections are the most dangerous parts of a blind or visually impaired person's travel. To address this problem, this paper describes the novel "Crosswatch" system, which uses computer vision to provide information about the location and orientation of crosswalks to a blind or visually impaired pedestrian holding a camera cell phone. A prototype of the system runs on an off-the-shelf Nokia N95 camera phone in real time, which automatically takes a few images per second, analyzes each image in a fraction of a second and sounds an audio tone when it detects a crosswalk. Real-time performance on the cell phone, whose computational resources are limited compared to the type of desktop platform usually used in computer vision, is made possible by coding in Symbian C++. Tests with blind subjects demonstrate the feasibility of the system.

1. Introduction

Urban intersections are among the most dangerous parts of a blind person's travel. While most blind pedestrians have little difficulty walking to intersections using standard orientation and mobility skills, it is very difficult for them to *align* themselves precisely with the crosswalk. Proper alignment is necessary for them to enter the crosswalk in the right direction and avoid the danger of straying outside the crosswalk.

We have improved and simplified the algorithm we described in [2] for detecting and localizing zebra (striped) crosswalks in images so that it now runs in real-time on the cell phone platform. The new algorithm uses factor graphs (see Sec. 3.2 below) with higher-order interactions than the pairwise Markov random fields used in the previous algorithm, with interactions that have been learned from training data (rather than being chosen by trial and error), and it has been optimized for real-time performance on the Symbian cell phone platform running Symbian C++.

The limited memory and CPU power of an embedded system such as the cell phone platform limits the amount of processing that can be performed by a real-time computer vision algorithm, and has constrained the design and implementation of our algorithm. We have implemented a prototype of the algorithm in Symbian C++ on an off-the-shelf Nokia N95 cell phone that functions within these constraints. The algorithm processes approximately three frames per second, and an audio tone is sounded in any frame in which a crosswalk is detected. By panning the cell phone left and right, the user can determine if any crosswalk is visible and if so, align him/herself to it (see Fig. 1). Tests demonstrate that blind subjects are able to hold the cell phone steady enough to take usable pictures [3] and that the system is effective in helping them to determine whether or not a crosswalk is visible, and if so to orient themselves to it.

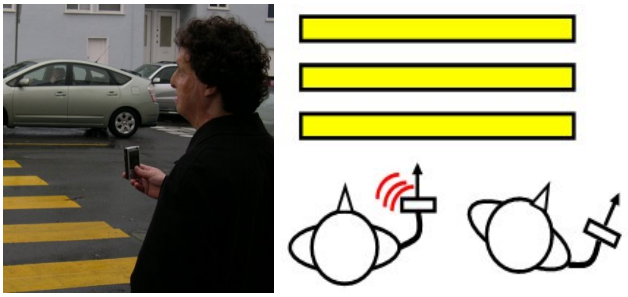

Figure 1. Left, camera cell phone held by blind user. Right, schematic diagram shows aerial view of zebra crosswalk and two users holding cell phone system: cell phone on left is aligned with crosswalk and makes an audio tone; cell phone on right is not aligned and makes no sound.

This is the first portable system we are aware of that provides real-time orientation information at traffic intersections based on visual cues – i.e. the existing crosswalk patterns already used by sighted pedestrians. The system has several advantages that make it a promising assistive technology tool. First, no added infrastructure is needed, as with systems such as Audible Pedestrian Signals (audio signals, located at some crosswalks, that sound when it is safe to cross) or other guidance systems requiring on-site installation (such as Bluetooth-based beaconing [1]). Second, the

978-1-4244-2340-8/08/$25.00 ©2008 IEEE

system runs on standard off-the-shelf cell phone technology – which is inexpensive, portable, multipurpose, and becoming nearly ubiquitous. Finally, the cell phone platform is a mainstream consumer product which raises none of the cosmetic concerns that might arise with other assistive technology requiring custom hardware.

2. Related work

Only a small amount of work has been done on algorithms for detecting zebra crosswalks for the blind and visually impaired [10, 6, 7, 9]. Past algorithms have been tested on fairly simple images in which the Hough transform is sufficient for extracting the borders of the crosswalk stripes. However, in real-world conditions in which an image of crosswalk stripes contains clutter, shadows, saturation effects, slightly curved stripes and occlusions, the Hough transform often fails to extract the desired lines. Instead of relying on a tool for grouping structures *globally* such as the Hough transform, we use a more flexible, local grouping process based on figure-ground segmentation using graphical models, drawing on past work [2] on crosswalk detection.

Our algorithm extracts straight-line segments from the image and applies simple grouping criteria such as parallelism, edge polarity and color consistency to construct candidate groupings of segments into the figure. The problem of assigning each segment to figure or ground is formulated using a *factor graph*, which is a graphical model (Markov random field) that expresses interactions among sets of two or more variables. In this case the factor graph consists of variable nodes, one for each straight-line segment extracted from the image, and interactions among neighboring sets of two to four nodes. Each node has one of two possible unknown states, figure or ground; the most likely combination of states for each node in the graph is inferred using factor BP (belief propagation), which is a generalization of standard BP (typically used on graphs with pairwise interactions).

Figure-ground segmentation has been successfully applied [12, 5] to the detection and segmentation of *specific* objects or structures of interest from the background. Standard techniques such as deformable templates [13] are poorly suited to finding some targets, such as printed text, stripe patterns, vegetation or buildings, particularly when the targets are regular or texture-like structures with widely varying extent, shape and scale. The cost of making the deformable template flexible enough to handle such variations in structure and size is the need to estimate many template parameters, which imposes a heavy computational burden. In these cases it seems more appropriate to group target features into a common foreground class, rather than seek a detailed correspondence between a prototype and the target in the image, as is typically done with deformable template and shape matching techniques.

We formulate our approach in a general figure-ground segmentation framework and apply it to the problem of finding zebra crosswalks in urban scenes. Future research will focus on extending the approach to find two-stripe crosswalks, which are another common type of crosswalk. (These crosswalk patterns consist of two narrow white stripes bordering the crosswalk, perpendicular to the stripes in a zebra crosswalk; the small number of features makes the two-stripe crosswalk much more challenging to detect.)

3. Algorithm

Our algorithm consists of two main stages: feature extraction, in which straight-line edge segments are extracted from the image, and figure-ground segmentation, which assigns a figure or ground label to each feature. Figure-ground segmentation is performed as follows. First, a factor graph is constructed, in which one variable with unknown state (figure or ground) corresponds to each segment. Second, factor graph belief propagation (BP) is performed to infer the most likely unknown states in the factor graph. Features that BP classifies as figure are the ones the algorithm believes to lie on a crosswalk, and all other features are assigned to the background. If enough features of sufficient length are classified as figure, then the algorithm decides that a crosswalk is present in the image, and sounds a tone to communicate its presence to the user.

3.1. Feature Extraction

The first stage of our algorithm is the extraction of straight-line segment features from the image. Straight-line edge segments were chosen as useful features to extract because they appear many times in a typical crosswalk and form a compact representation of the borders of the painted zebra crosswalk pattern.

Segment extraction is done in several stages that include blurring the image, edge extraction and edge pixel grouping. The original color image (photographed by the Nokia N95 cell phone) has resolution 320 x 240 pixels. A grayscale version of the original color image is created and then slightly blurred with a [1, 2, 1] kernel in horizontal and then vertical directions. Horizontal and vertical edge maps are then computed by applying horizontal and vertical derivative filters with kernels [-1, 0, 1] and then taking the absolute value of the derivatives. Non-maximal suppression for each of the two edge maps is performed by setting all values of the edge map to 0 except at local maxima of the horizontal/vertical derivatives; additionally, any values from the resulting map less than a certain threshold are also set to 0. A single edge map is computed as the pixel-wise maximum of the horizontal and vertical edge maps. The resulting map is grayscale and sparse (i.e. many pixel values

are 0).

In order to extract line segments we perform a simple edge pixel grouping procedure that tracks non-zero edge points in a roughly horizontal direction (within $\pm 45^\circ$ of horizontal) from left to right, and groups connected sets of non-zero edge pixels that form a sufficiently straight line in this direction. (Note that our edge grouping procedure is not designed to detect line segments that are within $\pm 45^\circ$ of vertical; restricting our search to near-horizontal segments suffices for typical viewing conditions, in which the camera is held roughly horizontal, and reduces the amount of computation.) Next we attempt to close gaps between neighboring sets of grouped pixels when the gaps are sufficiently small (up to 5 pixels) and the pixel groups are roughly collinear.

The resulting set of segment candidates (an example is shown in Fig. 2) is pruned to discard any segment with length less than 20 pixels; usually we obtain up to 100 segments in a 320x240 image. Each segment is represented by the pixel coordinates of its two endpoints.

Figure 2. Left, picture of zebra crosswalk. Right, straight-line segment features extracted from image (superimposed in white).

3.2. Grouping of Features Using Factor Graphs

Once straight-line segments have been extracted from the image, we use a factor graph model to group them into figure and ground, representing crosswalk and background, respectively. In this framework, cues describing relationships among neighboring segments are used to provide evidence about whether the segments are likely to belong to figure or ground. Such evidence is formulated statistically based on the empirical differences between segments belonging to crosswalks and those belonging to the background. For instance, neighboring segments on the crosswalk tend to be nearly parallel in the image – more parallel, on average, than similar segment pairs drawn from the background. The differential properties of such cues are learned from training data in which all segments have been manually labeled as figure or ground. The evidence from all the cues is combined in a factor graph, which is a graphical model (i.e. Markov random field, or MRF) that expresses the joint distribution of the figure/ground labels of each segment in the image given the evidence from all the cues.

We now describe the factor graph mathematically. We are given n segments that we have extracted from the image. The unknown states we want to infer are $x_1, x_2, \ldots, x_n$, where $x_i = 0$ or $x_i = 1$ mean that segment i has state 0 (ground) or 1 (figure), respectively; the global state assignments of all segments are denoted by $X = (x_1, x_2, \ldots, x_n)$. A simple i.i.d. prior is defined on X: $P(X) = \prod_{i=1}^{n} P_i(x_i)$ where $P_i(x_i = 0) = p_0$ and $P_i(x_i = 1) = 1 - p_0$. From labeled training data we estimate that $p_0 = 0.65$ since most segments belong to ground.

Next we describe the likelihood function for the special case of a binary cue, which is a measurement that expresses a grouping relationship between two segments; it is straightforward to generalize to multiple cues with arbitrary arities (i.e. the number of segments grouped in a single cue measurement). A simple example of a binary cue used in our crosswalk algorithm is the absolute value of the orientation difference between two neighboring segments, which we denote C_{ij} for segments i and j. As mentioned above, segment pairs on the same crosswalk tend to nearly parallel, whereas segment pairs elsewhere in the image are less likely to be parallel. This tendency is expressed in terms of the conditional distribution $P(C_{ij}|x_i, x_j)$. We define $P_{on}(C_{ij}) = P(C_{ij}|x_i x_j = 1)$ and $P_{off}(C_{ij}) = P(C_{ij}|x_i x_j = 0)$; note that the product $x_i x_j = 1$ implies that both segments i and j belong to figure, and $x_i x_j = 0$ implies that one or both segments belong to ground. Both distributions have been learned from labeled training data, and are shown in Fig. 3. Differences between the distributions $P_{on}(.)$ and $P_{off}(.)$ provide evidence for a pair of segments belonging to figure or ground, which is reflected in $\log[P_{on}(C_{ij})/P_{off}(C_{ij})]$, also shown in Fig. 3.

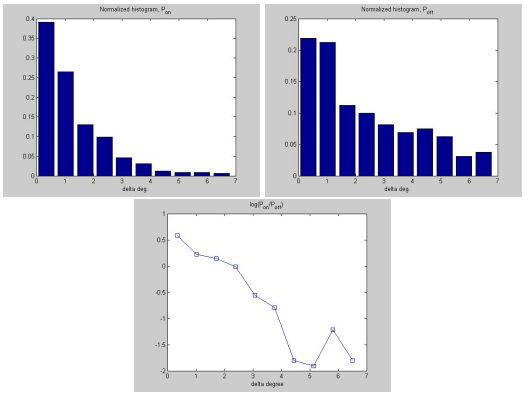

Figure 3. Top two panels: $P_{on}(.)$ and $P_{off}(.)$ distributions for the parallelism cue, which measures the absolute value of the orientation difference between two segments. Bottom, $\log P_{on}(.)/P_{off}(.)$. In all three panels, the horizontal axis is the orientation difference in degrees.

If we assume that the cues are conditionally independent across the image (and also that multiple cues are condition-

ally independent in the general case), then we can express the posterior distribution $P(X|C)$, where C denotes the set of cue values C_{ij} for all neighboring segments i and j, as follows:

$$P(X|C) \propto P(X) \prod_{(ij)} P(C_{ij}|x_i, x_j) \quad (1)$$

where the product over (ij) denotes all pairs of neighboring segments, and the constant of proportionality depends only on C. It is straightforward to show that $\prod_{(ij)} P(C_{ij}|x_i, x_j)$ can be re-expressed as $[\prod_{(ij)} P_{off}(C_{ij})][\prod_{i,j:x_i x_j=1} P_{on}(C_{ij})/P_{off}(C_{ij})]$, where the restriction $x_i x_j = 1$ in the product ensures that only those segment pairs whose nodes both belong to figure are included. Since the term $\prod_{i,j} P_{off}(C_{ij})$ is independent of X, the MAP (maximum a posterior) can be determined by maximizing the following expression:

$$R(X|C) = P(X) \prod_{(ij):x_i x_j=1} P_{on}(C_{ij})/P_{off}(C_{ij}) \quad (2)$$

Taking logarithms, an equivalent way of estimating the MAP is to maximize the following function:

$$\log R(X|C) = \sum_i \log P_i(x_i) + \sum_{(ij)} x_i x_j \log \frac{P_{on}(C_{ij})}{P_{off}(C_{ij})} \quad (3)$$

where the product $x_i x_j$ ensures that the sum is taken only over those segment pairs whose nodes both belong to figure. We will use factor graph BP to estimate the MAP of this function, as described in Sec. 3.4.

3.3. Multiple Cues in Factor Graph

The previous section described the factor graph framework for just one cue, parallelism, which is an arity-2 (i.e. binary) cue. In this section we describe seven other cues that are used, as well as an outline of how the factors are constructed from the extracted segments. All eight cues were explored and refined by evaluating a set of candidate cues: for each candidate, $P_{on}(.)$ and $P_{off}(.)$ distributions were learned from training data and an ROC curve was calculated to determine the cue's power to discriminate between on- and off-crosswalk distributions.

Note that for a cue C of any arity m, $P_{on}(C)$ is the distribution of the cue conditioned on *all* m segments contained within it belonging to figure, and $P_{off}(C)$ is the distribution conditioned on *one or more segments* belonging to ground. Also note that Eq. 3 should be augmented by one term for each additional cue, for instance a ternary cue C_{ijk} requires an additional term $\sum_{(ijk)} x_i x_j x_k \log \frac{P_{on}(C_{ijk})}{P_{off}(C_{ijk})}$.

In our resulting algorithm, two arity-1 (unitary) cues are used, length and edge gradient magnitude. The length cue exploits the fact that segments that lie on the crosswalk stripe edges tend to be longer than those from the background. The edge gradient magnitude measures the average magnitude of the intensity difference on both sides of the segment; segments on crosswalks tend to have stronger edge gradients than those off the crosswalk.

Three other arity-2 (binary) cues are used in addition to parallelism: horizontal overlap and two color consistency cues. In all cases the binary cues are applied only to segments of *opposite* edge polarity, i.e. segments for which the intensity gradients point in opposite directions, which is appropriate for segments bordering both sides of a painted stripe (or a gap between stripes). Horizontal overlap refers to the number of pixel columns shared by a pair of segments; under typical viewing conditions, segment pairs inside a crosswalk have a high degree of horizontal overlap.

The color consistency cues are designed to capture the consistency of color among pixels inside and outside stripes. Unlike *absolute* RGB color cues, color consistency is robust to variable lighting conditions and to the unknown intrinsic color of the crosswalk stripes (either yellow or white). Both cues are computed using RGB samples taken just above and just below the segment.

Half of the RGB samples are "paint" pixels (corresponding to white or yellow crosswalk paint) and the other half are "pavement" pixels (corresponding to the dark pavement area between stripes) – the polarity of the segment determines which is which, since paint is always brighter than pavement. The first color consistency cue measures the RGB consistency of pixels within each region; the second cue measures the consistency of RGB gradient values (i.e. RGB differences across both sides of a segment) at points along both segments.

One arity-3 (ternary) cue is used, which we call "stripe width monotonicity." This cue exploits the fact that under typical viewing conditions, the width of stripes (and gaps between stripes) decreases towards the top of the image, where the crosswalk is farther from the camera. The corresponding cue is calculated as the difference between the upper and lower vertical widths among the three segments in the factor.

Following the work of [9], one arity-4 cue is used, the cross ratio, which is a geometric quantity invariant to viewing perspective. This cue exploits our knowledge that stripes (and gaps between stripes) have equal width in three dimensions, which allows us to compute an ideal cross ratio. Specifically, consider a line drawn along the crosswalk plane that intersects any two adjacent stripes (or gaps); this line will intersect the stripe edges in four collinear points (see Fig. 4(a)), resulting in a cross ratio value of $1/4$. A similar cross ratio should be obtained for any line in the im-

age plane that intersects the crosswalk; the absolute value of the difference between the calculated cross ratio and the predicted value of $1/4$ is the arity-4 cue that we use.

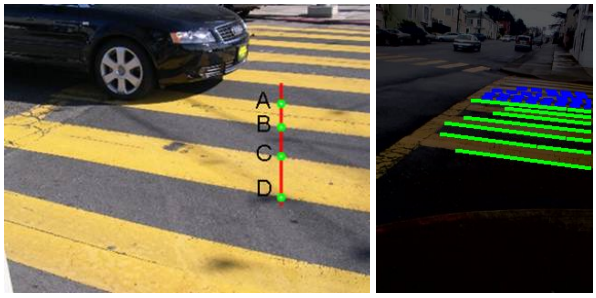

Figure 4. Left, illustration of cross ratio cue, computed with respect to a line that intersects four crosswalk stripe edges. Right, results of running full algorithm for the image shown in Fig. 2. Segments correctly classified as figure are shown in green; dashed blue indicates segments incorrectly classified as ground.

For both the arity-3 and arity-4 factors, we note that consecutive segments are constrained to have alternating polarities, which generalizes the opposite polarity requirement described for the arity-2 cues.

Next we describe how all the factors for each cue are constructed. First, all pairs of neighboring segments with opposite polarities are considered; only pairs with sufficient degrees of parallelism and overlap are chosen as candidate factors. For each candidate pair so obtained, candidate arity-3 factors are generated by choosing additional segments that satisfy the alternating polarity constraint and don't grossly violate the stripe width monotonicity constraint.

Each candidate arity-3 factor is subjected to an "anti-jump" test that maximizes the probability that the three segments are consecutive. This test samples RGB values in the two regions between the upper and lower segment pairs; if the RGB values are clustered tightly enough about a centroid in each region and there is minimal overlap between the two clusters, then the factor passes the test. Without such a test, too many factors with non-consecutive segments would be chosen (i.e. segments that could be separated by several stripes), which would contaminate some cues (such as the cross ratio) whose values become meaningless if the segments they are derived from are not consecutive.

Finally, the surviving arity-3 factors are combined with additional segments to form candidate arity-4 factors. For each candidate arity-4 factor (with segments i, j, k and l), both triplets (i, j, k) and (j, k, l) must survive the anti-jump test, and in addition the four segments must have a cross ratio sufficiently close to $1/4$, in order for the candidate to be accepted as a factor.

Note that all $P_{on}(.)$ and $P_{off}(.)$ distributions are measured using factors constructed from the above procedure, obtained from a set of training images. Thus the form of $P_{on}(.)$ and $P_{off}(.)$ depend not only on the training images but also on the exact procedure for constructing factors, and this procedure affects the range of training cue values that are represented in $P_{on}(.)$ and $P_{off}(.)$. Naturally, the same factor selection procedure is used for performing inference on images.

3.4. Factor Graph BP

The MAP estimate is the value of X that maximizes the log posterior, given in Eq. 3. We use factor BP [4], which is a generalization of standard BP [11] (typically formulated for graphical models with pairwise interactions, i.e. arity-2 factors) for use with factor graphs. The MAP is estimated by performing the max-product (max-sum in the log domain of Eq. 3) version of factor BP. A serial (asynchronous) message update schedule is used, with several sweeps of the entire factor graph to attain convergence.

The messages in max-sum factor BP determine the "belief" functions, which estimate the state (figure or ground) that is more likely for each segment in the graph. (These beliefs are not direct estimates of marginal probabilities, which would be provided by sum-product BP.) Fig. 4(b) shows the result of running factor BP, with segments correctly estimated to belong to figure shown in green. We obtained better results by ignoring the beliefs of any segments that were not part of any arity-4 factors, and classifying such segments as ground. The rationale for this simplification is that arity-1, -2 and -3 factors are much less reliable than arity-4 factors. (Even when a segment is "discarded" in this way, the lower arity factors still participate in BP.)

3.5. Experimental Results

Our algorithm was implemented in Matlab (the cell phone implementation is described in the next section). The dataset of training images consisted of 25 images of urban scenes containing crosswalks (whose segments were manually labeled as figure or ground using a mouse) and 25 images of urban scenes containing no crosswalk (all segments in these images were labeled as ground). Each image was photographed by the N95 cell phone and had resolution 320x240. Some sample results are shown in Fig. 5, including one result for a white crosswalk.

The algorithm was tested on a set of 90 images not contained in the training set, 30 of which contained crosswalks and 60 of which did not (this proportion is typical of images obtained by a subject searching for a crosswalk). Some of the images were photographed by blind subjects and some by sighted users. We evaluated the performance of the algorithm in two ways.

The first was to measure the true positive and false positive rates, which are computed in terms of the segments extracted from each image, and which were determined to be 72% and 0.5%, respectively. (In this measure, each segment

in the entire image test set has equal weight regardless of length.) Note that imperfect extraction of segments means that some segments that should be extracted are missing; our true positive and false positive rates do not reflect this problem.

Naturally, with a different choice of belief threshold we could increase the true positive rate at the expense of also increasing the false positive rate; however, for the purposes of finding a crosswalk it is important to avoid false positives, especially since the cell phone user will be taking many images of each crosswalk. Conversely, a less-than-perfect true positive rate is acceptable when multiple images are taken of a crosswalk, since the crosswalk is likely to be detected in at least several of these images.

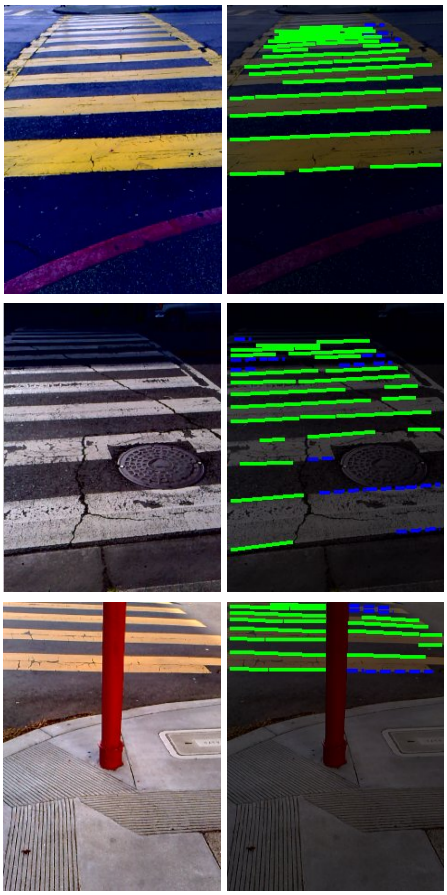

Figure 5. Results of running algorithm: input images on left, results on right. Segments correctly classified as figure are shown in green; dashed blue indicates segments incorrectly classified as ground. Note that one crosswalk is white and the other are yellow.

Next, we implemented the following rule to decide whether or not a crosswalk was present in each image. Each segment (if any) classified by factor BP as figure contributes a number equal to the length of the segment multiplied by an estimate of the probability that the segment is in the figure state. (This estimate, which would ordinarily be directly calculated by the beliefs calculated in sum-product BP, is an approximation derived from the non-probabilistic beliefs produced by the max-product version of BP that we are using.) The numbers are totaled to arrive at a final score, and a decision that a crosswalk is present is issued if and only if the score is larger than some (empirically chosen) threshold. Note that a crosswalk can be detected in this way even when many segments are missed by the algorithm (either because the factors containing them were noisy or because they were not extracted as features from the image).

Given the test set of 90 images, only one image was misclassified (the crosswalk was not detected in one of the 30 images containing crosswalks), shown in Fig 6(a). Fig 6(b) shows the only image in the test set without a crosswalk that generated any false positive segments. This performance measure is more subjective than the previous one since it is unclear how an image containing only a very small amount of visible crosswalk should be classified. Nevertheless, this measure more directly measures the success of the algorithm in the context of guiding a user to a crosswalk, and helping align him/her to the crosswalk.

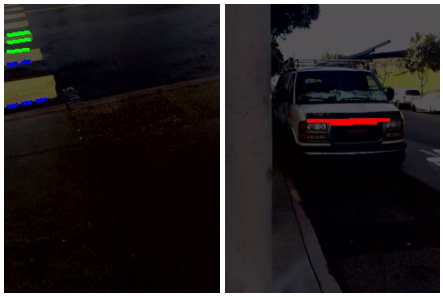

Figure 6. Left, the only image in the test set that was misclassified as not containing a crosswalk. Right, the only image in the test set without a crosswalk that generated any false positive segments (but too few for the entire image to be misclassified), shown in red.

Both performance measures can be improved in the future by implementing a high-level model verification stage. Such a process would fit the detected segments to a global model of a crosswalk that explicitly expresses the camera viewing angle (and perhaps metric properties such as distances if we assume known crosswalk dimensions) relative to the crosswalk. This would compensate for false positive and false negative segments to arrive at a more reliable crosswalk/no-crosswalk classification. The model could also estimate the location of the border(s) of the crosswalk (if visible) and give the user quantitative orientation information (e.g. “the crosswalk is at 11 o’clock”).

4. Cell Phone Implementation

A prototype version of our algorithm was implemented in Symbian C++ on the Nokia N95 cell phone (see Fig. 7), which we call “Crosswatch.” The camera is run in video

mode at 320 x 240 resolution, and approximately three frames are processed by the algorithm per second. A brief audio tone is sounded for each frame in which a crosswalk is detected. A simplified version of factor BP was used that is approximately equivalent to doing one sweep of BP; as our experimental results (described below) with blind subjects demonstrate, the system worked successfully even with this simplification.

Figure 7. Photograph of Nokia N95 cell phone running crosswalk detection algorithm. The green and red lines superimposed on the crosswalk stripe edges in the cell phone display denote segments classified as figure and ground, respectively.

4.1. Choice of Cell Phone Platform

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, Symbian C++ is the fastest language available for Symbian cell phones, whereas slower languages such as Java would be too inefficient for running computer vision algorithms on the cell phone. Third, 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 from our website and installed on any Symbian camera phone.

The N95 is a powerful Nokia cell phone model that offers several advantages, including high camera quality and a large amount of RAM, making it a natural choice for our Crosswatch system.

4.2. Implementation Details

The computational resources of the cell phone platform are limited compared to that of the typical desktop computer used in computer vision. The cell phone we used, the Nokia N95, has a dual ARM 11 CPU (only one of which is used by our algorithms), with a CPU clock rate of 332 MHz, and a total of approximately 81 MB free executable RAM memory. Symbian C++ is the language of choice for our algorithms since it is much faster than Java or Python (two popular languages that are supported on the Symbian platform).

In our experience, integer arithmetic calculations are approximately one order of magnitude slower on the cell phone (running Symbian C++). Floating-point calculations are even slower since they are performed in software rather than using an FPU. In some computationally intensive portions of our code we avoided floating-point calculations where possible in favor of integer operations; in future work we plan to make similar improvements throughout the code.

Another important software design strategy was to avoid repetitive calculations that dynamically allocate memory, as naturally occurs when growing and shrinking dynamic lists. A more efficient alternative was to substitute static arrays, pre-dimensioned at compile time to be large enough under all runtime conditions.

We note that Symbian C++ is challenging to program in because it is a non-standard dialect of C++. Newcomers to Symbian C++ are advised to consult a standard Symbian book [8], and a helpful wiki is at `http://wiki.research.nokia.com/index.php/N93HackingPages`.

4.3. Experiments with Blind Subjects

We performed two tests with blind subjects, which we briefly summarize (see [3] for a full report of the experiments). The outcome of the experiments proves the feasibility of the system, and shows that the system is sufficiently reliable to help a blind person detect and orient him/herself to a crosswalk. Both experiments were performed after a brief training session with the subjects.

The first experiment was designed to test the usability of the system and to test the ability of a subject to determine whether or not there was a crosswalk visible at a given traffic intersection. 15 different traffic intersections were chosen in advance, such that there was a 50% chance that a zebra crosswalk was present at each intersection. Intersections without a zebra crosswalk either had no crosswalks or had one or more non-zebra (i.e. two-stripe) crosswalks. The experiment was designed so that there were no cues available to the blind subject (e.g. ambient environmental sounds or tactile cues from the user's white cane) about the presence

or absence of a crosswalk except for those furnished by the cell phone system. The subject answered correctly whether a zebra crosswalk was present or absent at all 15 intersections.

The purpose of the second experiment was to test the ability of another blind subject to align him/herself with a zebra crosswalk. We led the subject near a traffic intersection and told him/her to approach the crosswalk, try to orient properly to it and press a button on the cell phone to take a picture of the crosswalk. In half the trials, the audio feedback from the Crosswatch system was turned off, and in the other half the feedback was turned on as usual. This procedure was repeated at all four corners of two separate crosswalks. The images captured during the experiment demonstrated that the subject was better able to align him/herself using the feedback from the system (though we emphasize that this measure is qualitative).

5. Conclusion

Crosswatch is a novel camera phone-based system for helping blind and visually impaired pedestrians find crosswalks and align themselves properly to them before crossing. A prototype system has been implemented on the Nokia N95 camera phone, which searches for crosswalk stripes a few times per second and provides audio feedback whenever a crosswalk is detected. We have conducted preliminary experiments with blind volunteers to test the system, demonstrating its feasibility.

In the future we will improve the robustness of our system by adding a high-level model verification stage to reject false positives and provide detailed geometric information about the user's orientation relative to the crosswalk. We will also optimize our code to increase the number of frames per second that can be processed. Ultimately we will expand the capabilities of the Crosswatch system to include detecting and reading traffic signal lights (such as "Walk" signals) and detecting and locating two-stripe crosswalks. We will also investigate the possibility of extending the functionality of the system to evening/low-light conditions, if possible.

Naturally, many more experiments will have to be conducted, and many improvements made to the functionality and user interface, before the Crosswatch system can become a commercial product. Government safety standards will be needed to establish appropriate performance parameters (such as acceptable maximum system error rates), and to determine appropriate training procedures for users. Most important, the blind and visually impaired community must be consulted at all phases of Crosswatch research and development to ensure that the system is useful, safe and easy to use.

6. Acknowledgments

The authors were supported by The National Institute on Disability and Rehabilitation Research (NIDRR) grant H133G030080, The National Institute of Health grants 1 R21 EY015187-01A2 and 1 R01 EY018345-01, and the Claire Giannini Fund.

References

[1] S. Bohonos, A. Lee, A. Malik, C. Thai, and R. Manduchi. Cellphone accessible information via bluetooth beaconing for the visually impaired. In *11th International Conference on Computers Helping People with Special Needs (ICCHP '08)*, Linz, Austria, July 2008.

[2] J. Coughlan and H. Shen. A fast algorithm for finding crosswalks using figure-ground segmentation. In *Proc. 2nd Workshop on Applications of Computer Vision, in conjunction with ECCV 2006*, 2006.

[3] V. Ivanchenko, J. Coughlan, and H. Shen. Crosswatch: a camera phone system for orienting visually impaired pedestrians at traffic intersections. In *11th International Conference on Computers Helping People with Special Needs (ICCHP '08)*, Linz, Austria, July 2008.

[4] Kschischang, Frey, and Loeliger. Factor graphs and the sum-product algorithm. *IEEETIT: IEEE Transactions on Information Theory*, 47, 2001.

[5] S. Kumar and M. Hebert. Man-made structure detection in natural images using a causal multiscale random field.

[6] S. Se. Zebra-crossing detection for the partially sighted. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2000 Volume 2*, pages 211–217, South Carolina, June 2000.

[7] S. Se and M. Brady. Road feature detection and estimation. *Machine Vision and Applications Journal*, 14(3):157–165, 2003.

[8] J. Stichbury. *Symbian OS Explained: Effective C++ Programming for Smartphones*. Wiley, 2005.

[9] M. S. Uddin and T. Shioyama. Bipolarity- and projective invariant-based zebra-crossing detection for the visually impaired. In *1st IEEE Workshop on Computer Vision Applications for the Visually Impaired*, San Diego, June 2005.

[10] S. Utcke. Grouping based on projective geometry constraints and uncertainty. In *International Conference on Computer Vision (ICCR)*, 1998.

[11] J. Yedidia, W. Freeman, and Y. Weiss. Bethe free energies, kikuchi approximations, and belief propagation algorithms. *MERL Cambridge Research Technical Report*, 2001.

[12] S. X. Yu and J. Shi. Object-specific figure-ground segregation. In *Computer Vision and Pattern Recognition (CVPR)*, 2003.

[13] A. Yuille. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.