ZebraLocalizer: identification and localization of pedestrian crossings

Dragan Ahmetovic
University of Milan
dragan.ahmetovic@studenti.unimi.it

Cristian Bernareggi
University of Milan
cristian.bernareggi@unimi.it

Sergio Mascetti
University of Milan
sergio.mascetti@unimi.it

ABSTRACT
Independent mobility in unfamiliar environments is a significant challenge for people with severe vision impairment. Among other problems, one specific issue concerns the identification of those road signs which can be recognized by sight only. In this paper we present *ZebraLocalizer*, an application for mobile devices that identifies zebra crossings and guides the user towards them. Two main problems are discussed in this contribution: the identification and localization of the crosswalks, performed by processing data acquired both from the camera and the accelerometers, and the design of an interaction paradigm specifically addressed to blind users. Experimental results, conducted both on a dataset of images and with blind users, validate the applicability of the proposed solution.

1. INTRODUCTION
Although mobile devices pose new problems to disabled users, they also provide new exciting opportunities. Indeed, on one side, new accessibility issues arise in the use of these devices; for example, there are difficulties related to the interaction based on touch-screens. However, on the other side, as these problems are being alleviated, for example by the use of screen readers, it becomes clear that mobile devices can support new assistive technologies that cannot be implemented on standard devices i.e., desktops and laptops. This is due to two main factors. Firstly, mobile devices can be used on the move, and hence they can support the user in a number of situations in which it is not practical to rely on a standard device. Secondly, many mobile devices are equipped with hardware sensors, like GPS receivers, accelerometers, and gyroscopes, that can be used to acquire information about the user's context and that are not generally available on standard computers.

In this paper we focus on the problem of supporting visually impaired users to identify and localize zebra crossings, that are one particular type of crosswalk characterized by a regular pattern of light parallel stripes[1]. The common problem of the existing solutions is that, even if the zebra crossing is correctly identified, no information is provided about the relative position of the user with respect to the zebra crossing [5, 8, 11, 10]. Consequently, none of the existing solutions is capable of guiding the user towards the crosswalk. Two difficulties arise while addressing this problem: (a) to compute the relative position of the user with respect to the zebra crossing and, (b) to define an effective user interaction paradigm capable of correctly conveying the alignment information to the blind users.

In order to tackle problems (a) and (b) mentioned above, we developed *ZebraRecognizer* and *ZebraLocalizer*, respectively. *ZebraRecognizer* is a software library that, by processing images, identifies zebra crossings and computes the relative position between the observer and the zebra crossing. *ZebraLocalizer* is an iPhone prototype application that acquires images from the camera, gives them in input to *ZebraRecognizer* and implements the interaction paradigm that enables blind users to identify a crosswalk, align to the best crossing position and safely cross the road. Note that, iPhone applications are accessible to visually impaired users through VoiceOver, a screen reader natively available in the iOS platforms. In a nutshell, the visually impaired user can touch the screen to explore the interface by speech, and can touch twice to activate an interface object (e.g., by touching an icon twice, the corresponding application is run). Hence, thanks to VoiceOver, the users are able to select and execute *ZebraLocalizer* as well as to perform the operations made available by its user interface (e.g. start/stop zebra recognition, have speech messages repeated, etc.).

The main contributions, with respect to the state of the art, are the following:

- differently from existing solutions, that only acquire data from the camera, *ZebraRecognizer* also uses data from the accelerometers to improve the recognition performance and to provide more accurate information about the relative position of the zebra crossing;
- *ZebraLocalizer* not only identifies the zebra crossings, but it also provides information that guide the user towards and over the zebra crossing.

[1] In the following we use the terms "zebra crossing" and "crosswalk" interchangeably

ZebraRecognizer has been designed with three main objectives: first, it should provide no false positive, as this could be hazardous for the user; second, it should have a low number of false negatives, hence correctly recognizing a zebra crossing most of the times; third, it should be efficient, in order to be capable of processing images acquired at a high frame rate from the camera. The objective of *ZebraLocalizer* is to provide helpful information about the relative position of the user with respect to the zebra crossing without providing too much information that would be confusing for the user. In order to evaluate if our solution achieves the above objectives, we conducted both automated experiments performed using a dataset of pictures and a human-driven evaluation performed with 5 blind users supported by a mobile device in a real world scenario. Our experimental results show that *ZebraRecognizer* and *ZebraLocalizer* achieve all of the above objectives. In particular, our results give evidence that *ZebraLocalizer* can effectively guide the user towards and over the zebra crossing.

The paper is organized as follows. Section 2 presents the related work, and in particular the existing solutions to the problem of zebra crossing identification. Section 3 first shows the technical description of the problem and then describes the recognition technique adopted in *ZebraRecognizer* that is based on image processing and spatial reasoning with data acquired from the accelerometers. In Section 4 *ZebraLocalizer* is described, while Section 5 presents the results of the experiments. Section 6 concludes the paper and highlights some future research directions.

2. RELATED WORK

Independent mobility in unfamiliar environments is a significant challenge for people with severe vision impairment. The main difficulties derive from the inability to efficiently obtain a holistic mental mapping of the surrounding area, while navigating. Such a mapping can be straightforwardly obtained by sighted individuals who can gather detailed information about close and distant objects through the visual channel. Instead, blind or visually impaired people can understand global features of an unknown environment by combining at conceptual level haptic information about objects in the immediate surroundings, acquired via tactile and kinesthetic perception (e.g. by touching objects with a white cane), and audio information about distant objects and events (e.g. moving cars). As shown in [2], nonvisual understanding of the environment is far more ineffective and inefficient as well as potentially dangerous than scanning the surroundings by sight.

In order to make independent mobility for sight impaired effective, efficient and safe, research in assistive technology has addressed two major problems: to localize the position and orientation of the sight impaired person, and to acquire and convey information about surrounding objects (e.g. type, color, distance with respect to the observer, etc.). In this paper we tackle the latter problem. A widely used method to make objects detectable relies on RFID tagging. Among other solutions, Fukasawa et al. [7] use RFID tags for a nonvisual guiding system in railway stations. Alternatively, objects can be tagged by visual markers, which are recognizable through a camera. For example, Chan et al. [4] propose an indoor wayfinding system for visually impaired based on color markers placed on relevant elements (doors, stairs, etc.). Marked objects can be easily recognized, nonetheless the environment must be properly adapted and tags can be unwittingly damaged thus making nonvisual navigation unsafe.

The solutions based on RFID or color markers require that specialized physical objects are placed in the environment. This is not the case when the problem of object recognition is solved through computer vision techniques not applied to specifically adapted environments. Some solutions in this class aim at recognizing a broad class of objects by combining text reading (OCR), barcode reading, and other computer vision techniques. For example, the Looktel application is a comprehensive platform, specifically designed for visually impaired users, that recognizes many types of objects (e.g. doors, text labels, etc.) [9]. The Google Goggles application[2] uses a similar approach to extract information from an image and use it to perform a web search. To the best of our knowledge, none of these applications is capable of recognizing crosswalks.

Other solutions based on computer vision are specialized for the recognition of a single class of objects. Angin et al. [3] developed a system which performs fast recognition of traffic lights by exploiting the computational power of cloud computing. Unlike the work in [3], our solution aims to provide the user with real-time information about crosswalk position, therefore computation based on cloud computing cannot be exploited because of the delay due to network latency.

The zebra crossing recognition problem was first addressed by Stephen Se [8]. This solution can differentiate between zebra crossings and similar patterns that are sloped with respect to the ground plane (e.g., staircases). However, as observed by Uddin et al. [11], the solution proposed in [8] suffers from performance issues. In order to address this problem, Uddin et al. propose a more efficient detection algorithm based on bipolarity feature check and projective invariant. This work was extended in a later paper by Uddin et al. [10] that presents a technique also supporting the computation of the zebra crossing length. As recognized in [11], the techniques proposed in [11] and [10] can fail to recognize a crosswalk in case the zebra crossing is partially occluded, for example due to a car passing by. Vice versa, the recognition technique proposed in our paper can recognize a zebra crossing also in case of partial occlusion. Furthermore, *ZebraLocalizer* also guarantees that a crosswalk can be recognized in case the zebra crossing is totally occluded for a short timespan.

Ivanchenko et al. propose two techniques for crosswalk recognition: one is specific for zebra crossings [5], the other addresses the problem of detecting a different type of crosswalks, the two-stripes crossings [6]. In [5] the presence

[2] `http://www.google.com/mobile/goggles/`

of the crosswalk is announced to the user through simple acoustic signals. According to the experimental results presented in [5], this technique is reliable and efficient. However we suspect that the reliability of the method proposed by Ivanchenko et al. relies on the assumption that the device is kept in a precise position (portrait or landscape). This may not always be the case as it cannot be excluded that a visually impaired user accidentally rotates the device while walking. Vice versa, in our technique we use the information acquired through the accelerometers to render our technique *rotation independent* in the sense that the result of the recognition technique is not affected by the orientation of the device. The idea to use data acquired though the accelerometers to enhance the recognition was first adopted in [6] to recognize two-stripes crossings. In this solution, the user is informed about the presence of the crossing through acoustic signals coupled with vocal feedback notifying the user's lateral orientation with respect to the crossing. In our solution we apply a similar idea in the case of zebra crossings and we extend it by providing the user with three pieces of information: the lateral shift, the frontal distance and the rotation. The challenge, addressed by *ZebraLocalizer*, is to provide this information without distracting the user with overwhelming audio feedback.

To conclude, our solution improves the existing work in three directions. First, *ZebraRecognizer* relies on the accelerometer data to improve the performance both in terms of computation time and accuracy of the result. Second, our solution provides accurate information about the position and orientation of the sight impaired person with respect to the zebra crossing. Third, our solution introduces a novel feature concerning the human interaction paradigm: the user is not only informed about the existence of a crosswalk, but also guided towards it through speech messages that are carefully designed not to distract the user in the hazardous task of crossing a road.

3. THE *ZebraRecognizer* LIBRARY

Different road markings are used worldwide to identify pedestrian crossings. Despite this variety, the particular type of crosswalk called "zebra crossing" is probably the most common. Although a universally accepted definition for zebra crossings does not exist, many of the currently adopted zebra crossing standards are very similar. In this paper we use the definition provided by the Italian law [1] and we believe that our solution can be easily adapted to other standards.

A zebra crossing consists in a set of white *stripes*, generally painted on a dark background (see Figure 1). In this paper it is convenient to also define as "stripes" the gaps between two consecutive stripes. In this view, each stripe is a rectangle with a uniform light or dark color, having width of at least 250cm and height of exactly[3] 50cm. The long edges of two adjacent stripes are overlapping. Each crosswalk is composed by at least three consecutive stripes with alternating color, at least two of which are white stripes. The aim of *ZebraRecognizer* is to recognize the zebra crossings that have the long side of the stripes roughly perpendicular to the observer. Thanks to the regularity of this structure, an analytical recognition technique can be applied. In this section we describe the algorithm implemented in *ZebraRecognizer* that is composed of five main steps: horizon computation, feature extraction, line segment analysis, stripe analysis and computation of relative position. In this section we describe each step in detail.

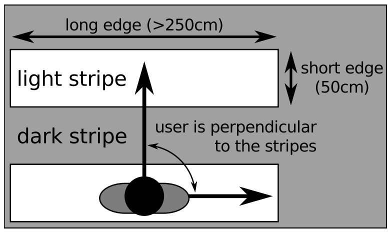


Figure 1. Zebra crossing example

The algorithm takes in input a gray-scale image and the values of the 3D accelerometers. The output contains a boolean value, indicating whether a crosswalk is recognized in the picture. If this is the case, the output also contains the information regarding the alignment of the observer with respect to the zebra crossing, described in details in Section 3.5.

3.1 Horizon computation

In the first step of the algorithm, the data acquired through the accelerometers is used to compute the position of the horizon with respect to the input image (see Figure 2). The computed horizon is used during the following stages and allows our algorithm to have the following fundamental properties:

- The feature search space is limited to the semi-plane below the horizon and to the roughly horizontal lines, hence reducing the computation time and limiting the possibility of false positive results;
- Parallel lines are recognized with high precision since, in perspective geometry, two parallel lines lying on the ground plane are either parallel to the horizon or they meet on the horizon;
- The relative position of the crosswalk with respect to the user is computed in order to allow the alignment of the user with respect to the crossing.

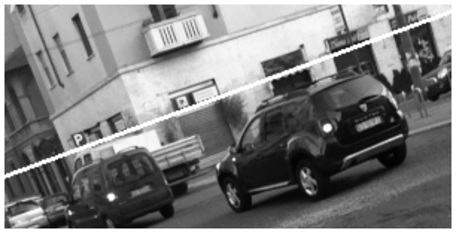


Figure 2. The horizon, as it is computed by *ZebraLocalizer*.

[3] Note that, according to some existing standards, the height of the dark stripes can be less than the height of the white stripes.

3.2 Feature extraction

In the considered object recognition problem, the features that we are interested to detect are the straight lines representing the long edges of each stripe. For this reason, in the second step, the image is processed to detect and isolate straight line segments. This is performed in three sub-steps.

$2.a$) The first sub-step is the line segment detection. For this purpose a modified version of the Line Segment Detector (LSD) algorithm [12] is used. The implemented algorithm differs with respect to the one proposed in [12]: it ignores both the points above the horizon and the line segments whose angle with respect to the horizon is bigger than $\pi/6$ (we recall that we are only interested in the stripes that are roughly perpendicular to the observer). Figure 3 shows the result of the original LSD algorithm compared with the result of our modified version.

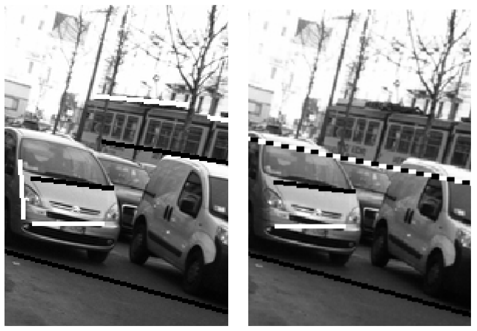

(a) Original LSD (b) Modified LSD (the dotted line represents the horizon)

Figure 3. Extraction of line segments (represented in white or black).

$2.b$) The second sub-step consists in merging segments that approximately lie on the same line and whose distance is below a given threshold value. This is useful because the LSD procedure may recognize parts of the same line segment as individual line segments due to noise in the image, imprecise coloration of the stripes or objects between the stripes and the observer (e.g.: a pole positioned between the user and the stripe). Similarly, in this sub-step we deal with the line segments that are approximately parallel and very close to each other: in this case we merge them into a single segment if they have the same gradient orientation or we drop both line segments otherwise. Figure 4 shows the result of the application of this sub-step.

$2.c$) In the last sub-step we drop the line-segments whose length is below a threshold value. Indeed, since we are interested in the recognition of the long edges of each stripe, dropping short segments does not discard any useful feature.

3.3 Line segments analysis

In the third step of the algorithm, the line segments are analyzed in order to group them into sets, each one representing a potential crosswalk. Since each zebra crossing is composed by at least two white stripes, a set of line segments that contains less than four elements cannot correspond to a crosswalk. For this reason, after each of the following sub-steps, the groups containing less than four elements are pruned.

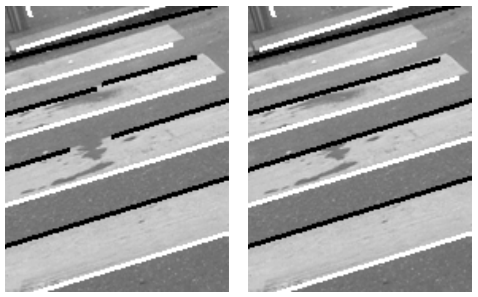
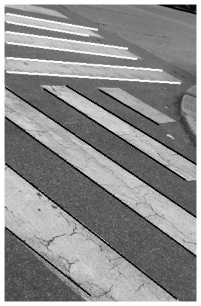

(a) Result after step $2.a$ (b) Result after step $2.b$

Figure 4. Line segments merging.

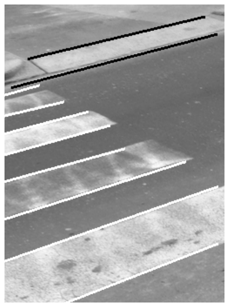
$3.a$) As observed above, the long edges of the stripes are parallel. For this reason, the line segments are grouped according to their slope. The computation is based on the observation that, in projective geometry, two parallel lines laying on the ground plane are either parallel to the horizon or they meet on the horizon. Exploiting this property, the identification of the parallelism among line segments is conceptually straightforward. Figure 5(a) shows two groups of line segments, the line segments belonging to one group are colored in white the others in black.

$3.b$) In the second sub-step, each group is partitioned into blocks according to the distances among the line segments. The idea is to exploit two geometrical properties of crosswalks: the height of the stripes is constant and the centers of the stripes lie on the same line. The former property can be checked by ordering the line segments according to their distance from the observer and then iteratively computing the distances between pairs of consecutive line segments. Since the crosswalk is observed in perspective geometry, the height of stripes must decrease as the considered pairs are farther from the observer. We empirically observed that the direct evaluation of the latter property is more involved. Indeed, the fact that stripes can be partially covered by obstructions (e.g. a car passing by) or not totally included in the picture prevents the LSD algorithm from recognizing the entire line segment. For this reason, our algorithm has been designed to tolerate the case in which the line segments are not perfectly aligned and to only exclude from the group the line segments whose horizontal distance from the group barycenter is significant. Figure 5(b) shows two groups of line segments grouped according to their distances. The group represented in black will be pruned as it contains two line segments only.

$3.c$) One obvious property of the line segments corresponding to a crosswalk is that, considering them in their order from the observer, the gradient of two consecutive elements must have an opposite sign. The last sub-step checks if this property holds and, if this is not the case, partitions the group accordingly.



(a) Grouping according to parallelism.



(b) Grouping according to distance.

Figure 5. Grouping of line segments.

3.4 Stripes analysis

During the stripe analysis step, each set of line segments is processed in order to prune from the set the elements that are recognized as not representing a crosswalk. This step is conceptually different from the previous one as line segments are not considered in pairs, but in groups of 3 (or 4) elements, representing 2 consecutive stripes (or 3 consecutive stripes, respectively). If, at the end of this step, a set still contains at least four elements, then it can be safely regarded as a crosswalk. For the validation to be positive, the following two criteria must be simultaneously satisfied: color consistency and cross ratio.

Color consistency aims to capture the difference in the color between a stripe and the background. Intuitively the dark (light) stripes should be darker (lighter, respectively) than the average color of the background. We compute it as follows: first, considering some sample points, we compute the average (avg), minimum (min) and maximum (max) color intensity of the half-plane below the horizon (i.e., the background). Then, again considering some sample points, we compute the average color intensity inside a stripe. If the color intensity of a stripe is within a threshold range from the min in case of light stripes or max in case of dark stripes the stripe is considered valid. Note that this approach to color consistency is preferable with respect to the absolute color computation because it is independent of the actual color of stripes which may be affected by lighting conditions.

The cross ratio is a projective invariant (a ratio preserved by the projective transformations) of an ordered quadruple of distinct points which lie on a straight line L. Considering the points A, B, C, D depicted in Figure 6, it holds that any choice of origin or scale does not influence the value of

$$\frac{AC \cdot BD}{BC \cdot AD}$$

Since the stripes of a crosswalk have all the same width, it holds that

$$\frac{AC \cdot BD}{BC \cdot AD} = \frac{4}{3}$$

The *ZebraRecognizer* library checks, for any set of three consecutive stripes, if the cross ratio holds, also taking into account that some approximation, expressed in terms of a threshold value, should be tolerated.

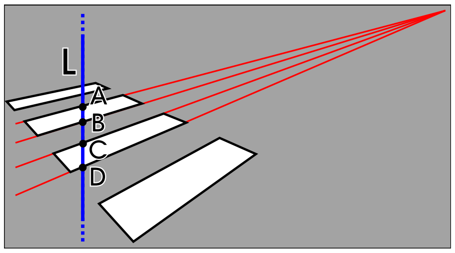


Figure 6. Cross ratio

3.5 Computation of relative position

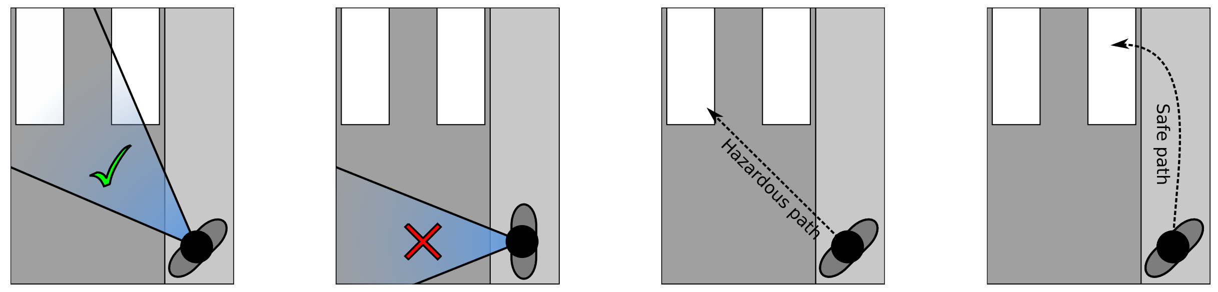
If a crosswalk is identified, the last step of algorithm computes its relative position with respect to the observer. The position is characterized by three values: the *rotation angle*, the *lateral shift*, and the *distance*.

The *rotation angle* is the angle the user needs to rotate in order to be orthogonal to the crosswalk, in the sense that the shoulders of the user are be parallel to the longer edge of the stripes. This value corresponds to the slope of the stripe closest to the observer with respect to the horizon. The *lateral shift* represents the movement that is required by the user to be at the center of the visible part of the crosswalk. This value is computed as the distance, in pixels and along the x-axis (i.e., the horizon), between the center of the line segments closest to the observer and the center of the picture. Finally, the *distance* between the user and the crosswalk is computed as the distance, in pixel, between the horizon and the line segment closest to the user. Given this value, and knowing the calibration information of the camera, as well as the position of the device (i.e., its orientation and height from the ground), it is possible to estimate the ground distance, in meters, between the user and the center of the line segment closer to the user.

4. THE *ZebraLocalizer* APPLICATION

ZebraLocalizer is an application developed for iOS and specifically optimized and tested with iPhone 4. This device was chosen both because it embeds a suitable hardware equipment (i.e. high performance processor, high quality camera and accelerometers) and since it natively makes available accessibility features required for blind users (i.e. the screen reader called "Voice Over").

Conceptually, *ZebraLocalizer* performs a simple task: it collects the input from the camera and the accelerometers, provides these information to *ZebraRecognizer* and returns the output to the user. However, during the preliminary evaluation of the application performed with one blind user, we realized that there is a number of issues that should be considered. In particular, in this section we describe the solutions implemented in *ZebraLocalizer* to tackle the problems related to the interaction paradigm and the data management.



(a) Crosswalk is recognized
(b) Crosswalk is not recognized
(c) Hazardous path
(d) Safe path

Figure 7. Problems arising when instructing the user to rotate, (Figures (a) and (b)) or when guiding the user to the crosswalk (Figures (c) and (d)).

4.1 The interaction paradigm

The interaction paradigm of an assistive technology like *ZebraLocalizer* should take into account two main issues. First, a blind person who is walking autonomously, aided by a white cane, usually focuses the attention both on haptic stimuli (e.g. touching the wall to go ahead along a sidewalk) and on sound signals or speech messages from the surrounding environment. Particularly, the auditive stimuli are essential for blind people to acquire information about distant objects and potentially dangerous events (e.g. an approaching car). Therefore, the interaction paradigm provided by a guiding aid such as *ZebraLocalizer*, should not divert the user's attention from haptic and especially auditive stimuli and it should also aim not to increase the mental workload with overmuch information.

The second issue concerns how to safely guide a blind user from the position at which a crosswalk is detected to the best position to start crossing (in front of the zebra crossing, roughly at the midpoint of the first stripe) and then to the other side of the crossing. Several problems can arise, consider the following as an example. Assume the zebra crossing is on the left side of the user at a certain distance (see Figure 7(a)); if the user rotates before getting closer to the crosswalk, the zebra crossing will be out of the camera field of view, and hence *ZebraLocalizer* will not be able to detect the crosswalk anymore (see Figure 7(b)). In the same situation, if the user takes the shortest path between her/his current position and the crosswalk, she/he has to get off the sidewalk, which can be potentially hazardous (see Figure 7(c)). The safe path is to first move along the sidewalk until the user gets closer to the zebra crossing, then rotate and, finally, shift on the left or right to get to a good position to start crossing (see Figure 7(d)).

To address the two problems above, *ZebraLocalizer* informs the user through very short speech messages, which can be easily retained in the user's short-term memory. These messages are read only when the "alignment status" changes: Table 1 defines the possible statuses and the corresponding messages that are read when the status is reached. The idea is that, when the crosswalk is identified and the user is still far from it, then *ZebraLocalizer* informs the user to proceed walking (see the second problem highlighted above) telling the user the position of the stripes (ahead, on the left or on the right). Otherwise, when the user is close to the zebra crossing, the alignment information should also consider the lateral shift, in order to find a good crossing position. Note that, when the user is crossing, it is particularly important to have short messages as they need to be quickly decoded and interpreted by the user, who needs to decide in a short time the action to take (e.g. to go straight on, to rotate slightly while crossing over a zebra crossing, etc.). In addition, each message is preceeded by a vibrational effect coupled with a very short sound, which aim to temporarily have the user's attention focused on the upcoming message. In case a user misses a message, she/he can have the latest message read again by touching the screen device. Thanks to this approach, *ZebraLocalizer* does not divert the user's attention from the surrounding environment, except for very short time slots.

Another problem related to the interaction paradigm is that the position in which the device is held by the user may dramatically affect the results of the recognition. For example, if the user is close to the crosswalk and holds the device pointing to the horizon the stripes closer to the user are out of the camera field of view, hence *ZebraRecognizer* returns a relatively high distance between the user and the closest stripe. Analogously, if the user is far from a zebra crossing and points the device camera down on the ground, it is likely that no stripe is identified. For this reason, there are two additional messages read by *ZebraLocalizer*: "point device downwards" and "point device ahead". The former is read when *ZebraLocalizer* detects that the device is pointing towards the horizon and that the distance from the closest stripe rapidly increases between two consecutive executions of *ZebraRecognizer*. Indeed, this generally means that the stripe closer to the user that was visible in the previous execution of *ZebraRecognizer* is not visible anymore, probably due to the fact that the user has moved close to it without rotating the device towards the ground. The latter message is read when no crosswalk is detected and the device is pointing towards the ground.

4.2 The data management

From the data management point of view, *ZebraLocalizer* needs to take into account that the information acquired from

Crosswalk detected	Distance δ (in meters)	Angle α (in degrees)	Shift s (in pixels)	Message
No	-	-	-	"zebra not detected"
Yes	$\delta > 1.5$	$\lvert\alpha\rvert \leq 10$	-	"proceed, zebra ahead"
Yes	$\delta > 1.5$	$\alpha > 10$	-	"proceed, zebra on the right"
Yes	$\delta > 1.5$	$\alpha < -10$	-	"proceed, zebra on the left"
Yes	$\delta \leq 1.5$	$\lvert\alpha\rvert \leq 10$	$\lvert s\rvert \leq 50$	"you can cross"
Yes	$\delta \leq 1.5$	$\lvert\alpha\rvert \leq 10$	$s > 50$	"shift left"
Yes	$\delta \leq 1.5$	$\lvert\alpha\rvert \leq 10$	$s < -50$	"shift right"
Yes	$\delta \leq 1.5$	$\alpha > 10$	-	"rotate right"
Yes	$\delta \leq 1.5$	$\alpha < -10$	-	"rotate left"

Table 1. Messages defined in the *ZebraLocalizer* application

the sensors are subject to noise, that should be properly filtered. Indeed, the instantaneous value of the accelerometers is not always accurate as it may be affected by the movement of the user. Our current implementation of *ZebraLocalizer* uses a moving average filter to attenuate this problem.

The issue is more involved when the data acquired through the camera are taken into account. Indeed, although *ZebraRecognizer* has a high detection rate (see Section 5.2), it can happen that even if the user does not significantly move, two consecutive executions of the recognition algorithm return significantly different results. Consider the case in which one user is pointing a crosswalk and a car passes over it: this can result in the stripes suddenly disappearing. According to the interaction paradigm defined above, the consequence is that *ZebraLocalizer* reads two messages to the user (one when the zebra crossing disappears and another when it is visible again).

In this case the filtering of the information is more complicated due to two main reasons. First, the number of samples (i.e., the results of the recognition) is limited to around ten per seconds, while the sample rate of the hardware sensors like the accelerometers is much higher. Second, the aggregation function is more complicated: each sample is composed by several values, and a different aggregation function is required for each of them. In our current solution *ZebraLocalizer* runs the recognition library a number of times and then aggregates the result before providing it to the user. Clearly, the higher is the number of executions, the better is the precision of the result, but the delay is higher. In our current implementation, we run the recognition procedure 3 times before aggregating the results and giving it to the user.

In practice, our solution is working properly, as it is accurate in the computation of the horizon (see, for example, Figure 2) and it is capable of frequently computing useful information that can actually guide the user towards the zebra crossings (see Section 5.3). Nevertheless, we believe that the data management technique can still be improved along two main directions: first, we shall evaluate the impact of different filtering technique, like the Kalman filter. Second, we are planning to combine the accelerometers information with the data acquired from other sensors, like the gyroscopes. This would be particularly interesting as it would make it possible to recognize the user's movements (e.g, walking straight, rotating, etc.). This information can be used, after a zebra crossing has been identified, to estimate the relative position of the same crosswalk after a period of time. Clearly, this knowledge can then be used to enhance both the recognition procedure and the aggregation of different recognition results.

5. EXPERIMENTAL RESULTS

In this section we report the results of our experimental evaluation. We conducted one set of tests, using a computer-based *quantitative* measurement, to measure the performance of the recognition technique while, in order to evaluate the usability of the *ZebraLocalizer* application, we conducted a preliminary *qualitative* evaluation of usability with blind users.

5.1 Experimental setting

The quantitative evaluation was performed by repeatedly running the *ZebraRecognizer* library on two sample sets of images, each one containing 200 items. One set contains pictures of crosswalks, while the other contains pictures of common urban environments (e.g. tramlines, shadows of trees and buildings, etc.) without zebra crossings. It should be observed that the latter set of picture was specifically designed in order to contain pictures of items that could be erroneously recognized as zebra crossings, like tramlines, stairs. etc. Also, both set of images were taken is different light and weather conditions, including the natural light of a bright sunny day, a cloudy rainy day and the artificial light in the night. All the images in the test sets were captured by the iPhone 4 camera with the low quality streaming video presets corresponding to a 192x144 resolution. These pictures were taken with an application we specifically developed that inserts the values of the accelerometers as a comment within the picture file. This makes it possible to use the accelerometer data during the recognition phase of the tests conducted on a PC platform.

The quantitative experiments aimed at achieving four major goals:

1. To prove that the recognition is accurate, in the sense that if there are crosswalks in the picture, they are properly recognized, otherwise no crosswalk is detected. Note that,

for the algorithm to guarantee safe road crossing, the number of false positives must be equal to zero, namely areas on the road background which are not crosswalks must not be erroneously recognized as crosswalks.

2. To evaluate the time efficiency in order to show that the algorithm can be embedded in applications running on off-the-shelf smartphones.

3. To show that the use of data acquired through accelerometers highly improves both the accuracy and the efficiency of crosswalk recognition.

4. To tune the internal parameters of the application.

In this section we show the results obtained for the first three points above. To evaluate how much accurate is the algorithm (point 1. above), we use two indicators: precision and recall. Precision is the ratio between the number of properly recognized crosswalks and the number of all the detected crosswalks, including erroneously detected ones. Recall is the ratio between the number of properly recognized crosswalks and the number of all pictures containing crosswalks. The time efficiency (point 2. above) was expressed as the average execution time per image in milliseconds. The tests were performed both on an iPhone 4 and on an IBM ThinkPad T60 with Gentoo Linux, with a 1,6GHz Intel Core solo processor and 4GB of RAM. For what concerns the impact of the accelerometer data (point 3. above), we implemented a variation of the *ZebraRecognizer* library that does not compute the horizon and that does not rely on accelerometer information.

The qualitative test was conducted by five blind users and aims to assess the usability of the *ZebraLocalizer* application in real life scenarios. Users were required to perform two tasks while walking on the sidewalk along a road, about 20 meters far from a crosswalk, aided by a white cane and by *ZebraLocalizer*. The first task required to detect a crosswalk located in front of the user. In this case the blind users were told to move as close as possible to the middle of the zebra crossing and then cross the road. In the second task users were told to walk straight on the sidewalk and cross the road in the middle of the first zebra crossing located on the left side. Note that, in both cases it was not possible for the blind user to predict the position of the crosswalk by any other mean than *ZebraLocalizer*. In particular, in both cases there was no physical marker on the sidewalk to denote the presence of the zebra crossing. In the latter case, the zebra crossing was not even located near a crossroads. Also note that none of the users have ever explored the area where the test took place.

For what concerns the experimental environment, we chose two zebra crossings in different light conditions. The former, in front of the user, was partially covered by the shadow of a building; the latter, on the left side of the user, was almost totally covered by the shadow of tree branches. Note that, ragged edges of the shadow of tree branches may affect the recall of zebra crossing recognition. The evaluation was conducted in a partly cloudy day with weak light conditions.

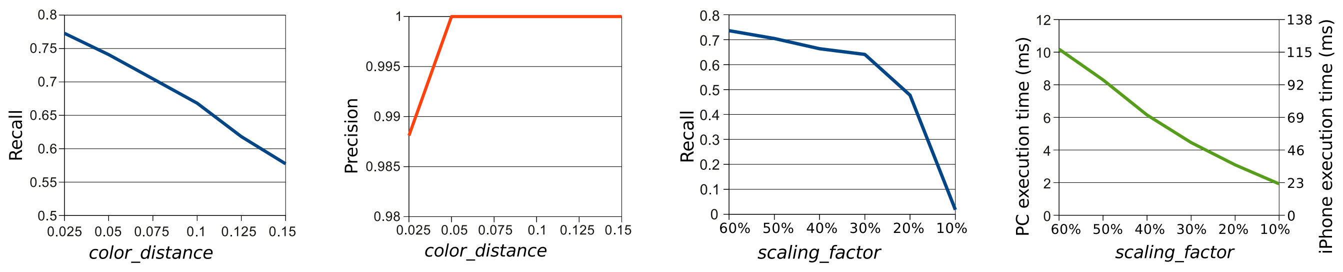
The following indicators were measured: (a) the user position and orientation before crossing the road, (b) the distance from the crosswalk at which *ZebraLocalizer* performs the first successful recognition, and (c) number of shift/rotation messages given to the user before finding the best crossing point. User position and orientation suggest how suitable are the alignment information given to the user; the distance at which a successful recognition is achieved helps the designers define use cases and, finally, the number of alignment messages measures the effort required to find the crossing point.

5.2 Quantitative evaluation

Our experiments highlight that two parameters have a major impact on precision and recall. One parameter, called *color_distance* is used during the stripe analysis (see Section 3.4) to define the minimum allowed difference between the average color of a given stripe and the average image color. The other parameter called *width_expansion* is a multiplier coefficient used during the line segment analysis (see Section 3.3) to predict the next stripe's maximum allowed width given the width of the last recognized stripe.

Figure 8 shows that the recall monotonically decreases with increasing values of *color_distance*. Vice versa, when this parameter is set to values smaller than 0.05 some false positive results are returned. Consequently, it is necessary to define a trade-off between precision and recall. Given that the aim of the *ZebraRecognizer* is to have no false positive, we used 0.075 as the default value for this parameters so that no false positive results are returned while about 70% of the zebra crossings are actually recognized. Analogous considerations were conducted for the *width_expansion* parameter.

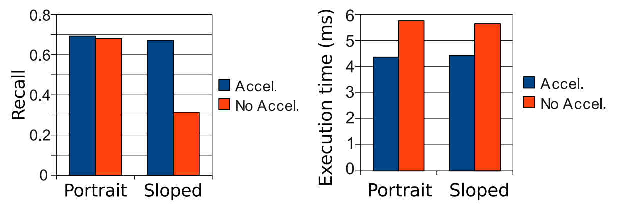
In our experiments it emerges that the size of the image is the parameter that most significantly affects the computation time. Figures 8(d) and 8(c) show the computation time and the recall, respectively, for different values of the *scaling_factor* parameter that defines how much the original 192x144 image should be scaled before processing. Independently from the value of this parameter the precision is equal to 1. As expected, both the recall and the execution time monotonically decrease with smaller images. However, while the execution time decreases almost linearly between 60% and 10%, the recall is not significantly affected for values of *scaling_factor* between 60% and 30%, while it rapidly decreases for smaller values. Vice versa, for larger images, i.e, *scaling_factor* larger than 60% (not represented in the figures), the recall does not significantly improves, while the computation time still grows almost linearly. Figure 8(d) shows the computation time both on a PC and on an iPhone 4. It can be observed that the computation on the iPhone 4 is about one order of magnitude slower but the recognition process can still be performed in less than 0.1 seconds. Thanks to this analysis, it is possible to conclude that a good trade-off between computation time and recall is obtained with values of *scaling_factor* between 60% and 30%. In our implementation of *ZebraLocalizer*, we had the objective to process about ten images per second and hence we chose to use 50% as the default value for *scaling_factor*.



(a) Recall (b) Precision (c) Recall (d) Execution time

Figure 8. Impact of the parameters *color_distance* (Figures (a) and (b)) and *scaling_factor* (Figures (c) and (d).

One set of experiments we conducted is devoted to measure the improvement introduced in the recognition process by use of the accelerometers. To achieve this, we divided the dataset of images representing zebra crossings into those taken in an almost portrait position and the others (called "sloped" in the following). Our results (see Figure 9) show that the use of accelerometers increases the recall by more than 100% with sloped images. Moreover, the accelerometer data can reduce the computation time of about 25%, for both sloped and portrait pictures.



(a) Recall (b) Execution time

Figure 9. Impact of accelerometer data on recall and execution time.

5.3 Qualitative evaluation

The usability test mostly confirmed the expectations of the designers and it also provided further suggestions for future development. The results can be grouped according to three stages for each task: **approaching**, while the user has detected a crosswalk and is getting close to the start point of the first stripe, **aligning**, while the user is rotating or shifting in order to find the best crossing position and **crossing**, while the user is walking over the zebra crossing.

In the approaching stage, all the users were able to detect the zebra crossing in both tasks. We observed that the recognition distance ranges from about seven meters in task 1 to about four meters in task 2. In the first task, two users remarked the difficulty to rotate the device toward the first stripe of the zebra crossing aided by the discrete audio feedback of *ZebraLocalizer*. Both of them reported that a continuous sound signal would probably better inform the the user about the correct device rotation. In the second task, one user run into difficulty while walking along the sidewalk searching with the device for the zebra crossing. Indeed, this user relied on the wall on his right to orientate. However, while searching for the zebra crossing, the user was induce to rotate on his left, hence moving away from wall and loosing his orientation. After incurring into this problem twice, the user autonomously learned to point the device for the detection on his left side while continuing to walk along the wall.

After the aligning stage, all the users were in a safe crossing position, less than one meter far from the first stripe and in front of the zebra crossing. In task 1, four users out of five were roughly at the midpoint of the first stripe before crossing. One of them started crossing at the right half of the first stripe. In task 2, three users out of five were at the left half of the first stripe before crossing, while two of them started crossing roughly at the midpoint. In task 2, four users corrected more than three times and up to seven times their position while rotating to align with the zebra crossing. This is due to the fact that the users rotated left or right more than needed. This suggests to provide the user with more detailed speech messages during the rotation movement. Two users suggested to provide more detailed messages about rotation, for example indicating how much the user needs to rotate, expressing this value in degrees or as on the clock face.

As for the measurement of the mental workload required in the aligning stage, it can be noted that, on average, four lateral shift messages are required in task 2 while only three messages are required in task 1. The difference between the two tasks is even larger for what concerns the time required to reach the correct crossing point: 10 seconds are required, on average, in task 1, while about 40 seconds are necessary on average, in task 2. The different results between task 1 and task 2 suggest that rotating and shifting turns out to be more imprecise and attention demanding than left/right shifting only.

For the crossing stage to be effective, the user must always be walking over the stripes and reach the opposite side of the road in a short time. All the users were able to cross the road (about 6m wide) in a short time ranging from three to five seconds. Nonetheless, one of them crossed the road very close to the right end of the stripes. While crossing he was holding the device on his left side hence he did not receive any information about the stripes on his right side. This suggests that the user should be better informed, before using *ZebraLocalizer*, on how to hold the device while crossing. One more remark from the users concerns the speech messages while crossing. Most of them (4 out of 5) pointed out that, although the messages are short, during the crossing phase (i.e., an hazardous situation) they are still a possible source of distraction. One possible solution could be to have even shorter speech messages or audio clues such as those provided by audible traffic lights.

As a final remark, it is worth noting that all the users were able to detect the crosswalks both in task 1 and 2 in a rather short time, if compared to the average walking time of a blind person aided by a white cane, and that all of them started crossing in a safe position and crossed the road on the crosswalk.

6. CONCLUSIONS AND FUTURE WORKS

Based on the experimental results illustrated in Section 5, we can draw some conclusions concerning both the effectiveness and efficiency of the recognition algorithm and the usability of the *ZebraLocalizer* application. Firstly, the recognition algorithm does not incur into false positives and recognizes most crosswalks. Also, the computation time is low enough that it is possible to execute the current implementation on off-the-shelf mobile devices.

The usability test gives evidence that, thanks to *ZebraLocalizer*, users can independently, quickly and safely discover crosswalks both on their left/right side and along their walking direction. The use of *ZebraLocalizer* is immediate, as all the users only took a few seconds to learn how to interact with this application. In addition, users can easily follow the instructions to be guided towards a good crossing position.

The test also highlighted two problems that we intend to solve: first, the user's rotation necessary to align with a crosswalk on his/her side is a task that requires some time and effort; second, the application should be even less intrusive when the user is crossing. We plan to alleviate the former problem by exploiting data acquired through more specialized sensors (e.g. gyroscopes) that, as mentioned in Section 4.2, can also be used to improve the data management involved in *ZebraLocalizer*. For what concerns the latter problem, we are investigating the following possible solution: the user notifies the application (through a simple gesture on the screen, like a double tap) when he/she starts crossing. While in "crossing mode", the application can switch to audio clues and can only give information about the presence of the zebra crossing.

We are also planning to further develop *ZebraLocalizer* along other two directions. First, it would be interesting to consider the extension of the user interface to support speech input that can increase the number of hands-free input commands the user can give to the device, thus improving the application usability. Second, we are considering to combine the recognition technique with other data retrieved from web-oriented services (e.g. geo-localized data sources). The idea is that each time a zebra crossing is identified, this geo-localized information can be shared with the other users. This knowledge can then be exploited, by a different user, to enhance the recognition procedure.

Acknowledgments

The authors would like to thanks the anonymous reviewers for their valuable suggestions, Andrea Gerino for his indispensable help with the iOS implementation and all the volunteer users involved in the experiments.

7. REFERENCES

1. Art. 145, d.p.r. 16/12/1993, n. 495, in materia di 'regolamento di esecuzione e di attuazione del nuovo codice della strada'; relativo all' art. 40 c.s.
2. J. D. Holtzman A. Arditi and S. M. Kosslyn. Mental imagery and sensor experience in congenital blindness. In *Neuropsychologia*, 1988.
3. P. Angin, B. Bhargava, and S. Helal. A mobile-cloud collaborative traffic lights detector for blind navigation. In *In Proc. of 11th Int. Conf. on Mobile Data Management*, pages 396–401. IEEE, 2010.
4. K. Y. Chan, R. Manduchi, and J. Coughlan. Accessible spaces: navigating through a marked environment with a camera phone. In *Proc. of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, pages 229–230, New York, NY, USA, 2007. ACM.
5. V. Ivanchenko, J. Coughlan, and H. Shen. Detecting and locating crosswalks using a camera phone. In *Computer Vision and Pattern Recognition Workshop*, pages 1–8. IEEE, 2008.
6. V. Ivanchenko, J. Coughlan, and H. Shen. Staying in the crosswalk: A system for guiding visually impaired pedestrians at traffic intersections. In *Assistive technology research series*, 2009.
7. S. Myojo N. Fukasawa, H. Matsubara and R. Tsuchiya. Guiding passengers in railway stations by ubiquitous computing technologies. In *Proc. of IASTED Human-Computer Interaction*. ACM, 2005.
8. S. Se. Zebra-crossing detection for the partially sighted. In *Proc. of the conference on Computer Vision and Pattern Recognition*. IEEE, 2000.
9. J. Sudol, O. Dialameh, C. Blanchard, and T. Dorcey. Looktel, a comprehensive platform for computer-aided visual assistance. In *Proc. of Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2010.
10. M.S. Uddin and T. Shioyama. Detection of pedestrian crossing and measurement of crossing length - an image-based navigational aid for blind people. In *Trans. on Intelligent Transportation Systems*. IEEE, 2005.
11. M.S. Uddin and T. Shioyama. Detection of pedestrian crossing using bipolarity feature-an image-based technique. In *Tran. on Intelligent Transportation Systems*. IEEE, 2005.
12. Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:722–732, 2010.