

UNIVERSITÀ DEGLI STUDI DI MILANO

PhD School in Computer Science
Computer Science Department



PhD in Computer Science,
XXVII° Cycle

**Assistive technologies on mobile devices for people
with visual impairments**

INF/01

PhD candidate:
Dragan AHMETOVIC

Advisor:

Dr. Sergio MASCETTI

Co-advisor:

Dr. Cristian BERNAREGGI

School Director:

Prof. Ernesto DAMIANI

Academic Year 2013/14

Contents

1	Introduction	1
1.1	Context	1
1.2	Problem description	2
1.3	Contributions	3
1.4	Methodology	4
1.5	Outline	5
2	Related work	6
2.1	Spatial understanding in didactic assistive technologies	7
2.2	Cognitive mapping in smartphone-assisted mobility	10
3	Didactic assistive technologies	13
3.1	Interaction techniques	13
3.2	Math learning for children with visual impairments	15
3.2.1	Design challenges	18
3.2.2	Proposed solution	20
3.2.3	Evaluation	22
3.3	Analysis of functions through sonification	23
3.3.1	Function graph exploration techniques	24
3.3.2	<i>AudioFunctions</i> prototype	26
3.3.3	Evaluation	29
4	Urban navigation for visually impaired users	33
4.1	Problem definition	34
4.1.1	The zebra crossing pattern	34
4.1.2	Safe navigation in urban environments	35
4.1.3	Smartphone driven computer vision	36
4.2	System architecture	37
4.3	Computer vision based zebra crossing detection	38
4.3.1	Feature extraction	38
4.3.2	Line segment analysis	39
4.3.3	Stripes analysis	41
4.3.4	Relative position computation	42
4.3.5	Evaluation	43
4.4	Sensor fusion based zebra crossing detection	46
4.4.1	Image Pre-Processing	47
4.4.2	Ground plane reconstruction	48

4.4.3	Line segment detection	51
4.4.4	Line segments grouping	54
4.4.5	Stripes validation	55
4.4.6	Evaluation	57
4.5	The user interaction module	65
4.5.1	Step-by-step navigation	66
4.5.2	Output interaction modes	68
4.5.3	Thumbs-only input	71
4.5.4	Evaluation	72
4.6	Zebra crossing detection from satellite and street view imagery	81
4.6.1	Satellite imagery acquisition	81
4.6.2	Detection on satellite images	82
4.6.3	Street view imagery selection	82
4.6.4	Detection on street view images	84
4.6.5	Evaluation	85
5	Conclusions and future work	88
5.1	Didactic and educational tools	88
5.1.1	Elementary grade math and geometry learning	89
5.1.2	Function graph exploration	89
5.2	Unassisted urban navigation	90
5.2.1	Zebra crossing detection	90
5.2.2	Speech and sonification interaction	91
A	Horizon computation from gravity acceleration data	93
B	Comparison of line segment detection methods	95
B.1	Hough probabilistic line segment detector	95
B.1.1	Custom canny edge detection	95
B.1.2	Hough line segment detection	96
B.2	LSD	97
B.2.1	Line support regions	97
B.2.2	Segment identification	98
B.3	EDLines	98
B.3.1	Edge Drawing	98
B.3.2	Line fitting	101
B.3.3	Validation	102
B.4	Comparison and evaluation	103
	Bibliography	107

Chapter 1

Introduction

1.1 Context

Technological capabilities of modern smartphones evolved well above their original use as mobile telephony appliances both in terms of sheer computational power as well as in terms of versatility. Indeed, modern smartphones, and the closely related tablet devices, are equipped with many and various sensors, such as video cameras, accelerometers and GPS. They also have ubiquitous and persistent connectivity and their interactive capabilities are various, ranging from haptic to auditive and, of course, visual. Continuously, new and revolutionary capabilities are researched and commercialized (e.g., Google tango project that localizes the user and, at the same time, maps the surrounding 3D space). These devices are gradually becoming multi purpose platforms, akin to and in some cases much more than modern personal computers. Clearly, both problems and opportunities arise from the continuous improvements of these devices.

On one hand new accessibility issues arise in the use of these devices [1]. The interaction paradigms are often new and still not well established. Specifically for people with sensor or motor disabilities, the accessibility features can take years to get published and even more to consolidate. For example, the VoiceOver screen reader 3.2.2, a major touch screen accessibility feature on iOS devices for users with visual impairments, was released in 2009, two full years after the first iPhone model. The zoom functionality, available from the start on iOS devices, was radically changed with the recent introduction of iOS 8 in September 2014. While the frequent updates do require an effort from the users to get accustomed to, these devices are indeed accessible by visually impaired users.

On the other hand, the new functionalities of these devices can be used to tackle issues specific for people with disabilities and lead towards a greater independence without the need of specifically designed and often expensive devices. Indeed mobile devices have three key advantages over personal computers. First, they boast new interaction

paradigms, such as the touch screen interface. Second, they can be used on the move and support the user in situations when it is impractical to rely on bulky personal computers or proprietary and expensive hardware platforms. Third, mobile devices are equipped with various hardware sensors, generally not available on standard computers, that can be used to acquire unprecedented amounts of information about the user's context.

The goal of this work is to leverage these capabilities for assisting persons with visual impairments in understanding spatial relations. The novel interaction paradigms offer proprioceptive capabilities and thus can be used as an assistive educational tool for learning spatial and geometrical concepts. At the same time, the hardware capabilities and the use on the move make the mobile devices suitable for autonomous orientation in space.

1.2 Problem description

Spatial understanding and cognitive mapping are challenging tasks for people with visual impairments [2–5]. While the capabilities in learning and memorizing spatial information in blind persons is comparable to those of sighted people [4, 6], the spatial information conveyed at a time through haptic exploration is less than what can be inferred by sight [4]. Additionally, there is also a lack of indirect sources of information, such as accessible maps, for people with visual impairments [7, 8].

In particular, visually impaired children often lack experience to develop appropriate strategies for spatial information comprehension and memorization [9, 10]. Abstract spatial concepts (e.g., geometry) are also harder to grasp without their graphical representations [2, 11, 12].

Didactic software generally leverage visual medium for conveying spatiality information. In particular, educational tools for children immerse the teaching purpose within an entertaining environment, often in the form of a graphical game. Thus, most of the existing educational tools are inaccessible to visually impaired and, specifically, children.

Multi-modal stimuli on mobile devices can be used to interact with persons with visual impairments [13]. Existing solutions are often based on auditory feedback or tactile and haptic stimuli. Auditory interaction paradigms [14–16] require much attention, memory skills and the ability to recognize even slightly different sounds. For this reason acoustic isolation from the surroundings (e.g., with isolating headphones) is required, thus preventing the interaction with others. Haptic approach proves to be much more effective but relies on expensive and cumbersome specialized hardware [17–22].

Spatial comprehension during independent urban way-finding is another major issue for visually impaired users, even more so in unexplored surroundings. While a sighted user can grasp in details the complexity of the environment and possible hazards with a single

look, a visually impaired user has to constantly monitor the audio and haptic feedback from the environment, which is more stressful and slow [7, 23–26].

Some information critical during the navigation for people with visual impairments cannot be accessed neither through gps-based navigation software nor detected by tactile, kinesthetic or auditive means. For example, consider the issue of knowing whether it is possible to cross a street at a pedestrian crossing based on the information given by a traffic light. This task requires constant monitoring of a purely visual cue that also changes its state continuously. The cue is also spatially limited and, thus, cannot be approached by GPS navigation. This information can be accessed either by augmenting the environment with RFID or bluetooth beacons [27, 28] or through computer vision based detection on device’s video camera stream.

1.3 Contributions

The goal of this work is to leverage audio-haptic proprioceptive interaction paradigms through touch screen interface and audio capabilities of mobile devices for assisting people with visual impairments in spatial comprehension and memorization. The issue of spatial comprehension for people with visual impairments is tackled in two main contributions, the first contribution is to the field of assistive didactics and the second one in the field of unassisted way finding. In both cases the capability of the user in constructing a map of the explored space is investigated. A specifically defined user driven methodology was adopted for the design of proprioceptive spatial exploration assistive technologies for people with visual impairments.

In the field of assistive didactics, this research stemmed two key results, published in [29, 30]. The first result is an instrument for helping elementary school children in learning math [29]. Differently from existing edutainment solutions the software uses proprioceptive touch screen interaction accessible by users with visual impairments. 13 exercises and corresponding interaction paradigms are defined and designed with help from educators and visually impaired users. An entertaining story, backed by professional graphics, vocal and audio effects is used for capturing the attention of both visually impaired and sighted children. The advantages of the design methodology are not limited to the elementary education. Visually impaired students of any age can benefit from new interaction techniques proposed. Indeed, the second result is a tablet prototype that allows visually impaired students in higher education environments to explore function graphs [30]. The user inputs custom functions with a specifically engineered interface and explores the resulting function graph with the sonification and touch screen interaction, designed with three exploration modes in mind, two of which are based on proprioception by taking benefit from the direct interaction with the tablet touchscreen.

An extensive experimental study of the effectiveness of the proposed solutions is performed. The first solution is thoroughly evaluated in three sessions: an expert-based evaluation, a test conducted with the first prototype of the app and a more qualitative evaluation conducted on the commercial version of the application both with sighted and visually impaired children. Additional feedback which has been received after the publication of the software, is also presented as an additional evaluation step. The second solution is evaluated with 7 users and the results show that the solution allows the users to have a much better understanding of the the function graph than existing software. The results are better even when compared with classical tactile paper, with which the test subjects were all well acquainted, while the proposed solution was tested for only few minutes.

In the field of spatial comprehension during way-finding for people with visual impairments, an important contribution is a zebra pedestrian crossing navigation framework (discussed in the following publications: [31–34]). Two novel and accurate computer vision algorithms were designed for the recognition of the pattern in different conditions of illumination and visibility. One relies on data from video camera for the detection [32] while the other also leverages data from accelerometers and gyroscopes for a more precise detection and metric relative position computation [34]. The solutions were engineered to be accurate and efficiently executed in real time on mobile platforms. The interaction paradigm based on sonification and haptic input is engineered for being applicable for both blind users and users with limited sight residual. The interface conveys the information quickly and effectively yet without overwhelming the user with too much data [32]. The capability of integrating the detection with online data sources is engineered. In particular, a result consists in the detection of zebra crossings from online map services imagery (both satellite and streetview). This component helps to guide the user when the crossing is still too far to be detected with the video camera of the mobile device. The proposed solution is thoroughly evaluated both by visually impaired test subjects and through automated computer based evaluation.

1.4 Methodology

The methodology adopted for the work described in the following chapters is first and foremost user-centered. Both during the investigation stage and during the design of the solutions and of the interaction paradigms, the interaction with visually impaired users was of paramount importance. Indeed, the issues to be investigated are identified alongside visually impaired collaborators and an effective way to tackle these issues and the interaction paradigm to be adopted are defined jointly. After the definition of an initial prototype, the proposed solution is evaluated through extensive user based tests. Iteratively, the solution is improved based on feedbacks until a stable version is engineered and published. Differently from a purely academic approach, the publishing stage may

involve interaction with commercial entities such as university spin-offs, private research and contributors. Involving subjects with an economical interest can help in financing and thus accelerating the development of solutions ready for commercialization. The contributions do not end with publishing, however, and the users' feedback is constantly gathered (manually and with automated methods) for enhancing the considered solution and as a starting point for new solutions. For what concerns navigation assistive tools, given the inherent danger in the autonomous way-finding, both the accuracy and the reaction time of the solutions have to be tightly constrained. For evaluating these two parameters, appropriate data sources for the investigated solutions are identified and testing data sets are collected. Then, through automated evaluation, the proposed solutions are evaluated against the data sets and possibly compared to existing solutions.

1.5 Outline

Chapter 2 outlines the research on assistive technologies for visually impaired, with particular focus on solutions for mobile devices. Two novel education tools are described in Chapter 3. Chapter 4 focuses on independent navigation of zebra pedestrian crossings for people with visual impairments. Finally, Chapter 5 draws the conclusions of the work.

Chapter 2

Related work

Cognitive mapping is the process of creating a mental model of a space based on the perception of it. Early theories [35] questioned the ability of visually impaired to construct an effective cognitive map of a space. Following studies [36] confirmed that blind people (both congenitally and late) are indeed capable of cognitive mapping but the level of spatial understanding compared to sighted persons was unclear. Recent works show comparable cognitive mapping capabilities between sighted and blind people, but different levels of experience influence the expressed spatial understanding among the visually impaired [4, 6].

A sighted person can scan the space visually and immediately grasp both the position of the contained elements with respect to the personal coordinates and, at the same time, the relative positions between these elements [6]. Conversely a person with visual impairments can scan the space haptically with hands or white cane, easily localizing the objects in the personal coordinates space, but having to infer the relative positions between the elements through a sequence of explorations and position calculations [4]. If the target space is contained within the haptic area, that is the area at which the exploration can be performed without changing location, a comprehension of the personal coordinate space is often sufficient for the orientation [4]. However, at locomotor distances, movement is required for the haptic exploration. Consequently, while moving, the coordinates in personal space transform with each movement, thus leading to a slower exploration with a higher mental workload required for the understanding of the space [4, 7, 23]. The spatial understanding depends on the strategies adopted for the exploration and codification of the relative positions between the elements of the space and the orientation capability with respect to those elements [9, 37, 38]. Education and experience in autonomous navigation strategies are shown to impact the way-finding capability significantly [6, 9, 38].

2.1 Spatial understanding in didactic assistive technologies

Indeed, assistive tools can be used to convey the spatiality information and help in the cognitive mapping through sensory substitution mechanisms. The visual elements are substituted with other stimuli capable of conveying the information [39]. Frequently adopted interaction paradigms for spatial exploration include, in particular: auditory and haptic interaction. For example tactile maps, embossed models of areas or buildings, can help people with visual impairments to learn the layout of a site and the relative positions between the elements of the area [4]. Analogous methods have been successfully employed also for the study of abstract spaces. Techniques for embossing figures on paper by a tactile embosser or produced with pen and sheets are used for the study of mathematics and geometry. Sewell kit is a common “pen and sheets” tool for embossing graphs on paper. It consists in a rubber board on which a resistant embossing paper is positioned. By etching a drawing on the board with a pen, it is permanently embossed on the paper while the rubber surface returns to the initial flat shape and can be reused. These solutions have been improved over the years [12]. Indeed, at present, there are many tools to create and emboss high quality drawings on paper [40].

Nonetheless, some drawbacks still exist. First, tactile drawings cannot be edited once embossed, so the mistakes require to reprint the whole paper. Moreover, it is hard to add elements to an already embossed tactile drawing (both on tactile embosser and on swell paper) without damaging or overlapping the existing drawing. For example, given the tactile graph of a function, it is difficult to draw the symmetric function on the same drawing. In addition, also the Braille labels can easily overlap with lines or other labels, and impair the understanding of the underlying figure [41].

Computer software have also been used for assisting a visually impaired user in spatial exploration and understanding. Auditory feedback in form of sonification has been successfully employed to replace the sense of sight in software [42]. Sonification is the use of non-speech audio to convey information. A large variety of sonification techniques exist and are used in various applications [43]. One sonification technique, referred to as “parameter mapping” [44], has been implemented for the *Navigator* component of the *ZebraCrossing* software, described in Chapter 4. Parameter mapping sonification is based on creating a link between the data to be rendered and the parameters of a synthesizer (or of any other device which generates or plays back sound). In the following, Sound spatialization was also employed in order to allow the user to clearly perceive certain sounds as coming from the left or from the right, therefore to convey information using an additional cue. Considering that during the test the sound could be delivered using a pair of headphones, for a determined set of messages a binaural spatialization approach was chosen [45]. This was not implemented performing a full spatialization, but simply modifying the differences in level and time of arrival of the sound at the two ears (i.e. Interaural Level Differences - ILD and Interaural Time Differences - ITD).

Mendels [46] notes that recently many interactive games, both desktop and mobile have been developed. In particular, many auditory interfaces have been specifically designed for games on mobile devices [46, 47]. Roden et al. [14] illustrate a framework to generate audio game in a 3D audio environment. Vallejo et al. [16] investigate sonification techniques in point and click games. Miller et al. [15] propose audio cues in games inherently based on audio feedback (e.g. where the player is required to reproduce a rhythmic pattern), but that are inaccessible to sight impaired people because of visual instructions. Ramos [48] and Ng [49] evidence the advantages of informative sounds in interactive games.

Sonification has also been used to enable sight impaired people to access function graphs [12, 50, 51]. Audio Graphing Calculator [12] provides a description of the shape of a function in form of a sequence of sounds that convey the value of the function through frequency variations. These solutions enable blind users to understand the trend of the graph, but not the relevant points such as maxima, minima and intersections. To grasp these quantitative attributes of the function, in addition to the function sonification, an automatically generated value table is computed and can be requested by the user.

The issue of conveying map information is tackled with 3D sonification techniques by Heuten et. al [52]. The user moves in a 3D virtual space that models a real world area. Different objects are mapped with different sounds conveyed stereoscopically according to their relative position with respect to the user in the 3D space.

Audio interaction, however, requires attention and good memory skills [53]. Hence, especially in education (e.g. geometry), auditory feedback alone should be used sporadically. Also, no quantitative information is straightforwardly provided with a sonification approach and it is also difficult to convey information about complex attributes of a figure (e.g. asymptotic behavior of a graph, straight angles in a geometrical figure). One more drawback concerns the difficulty for a visually impaired user to understand mutual relations between distant points or between distinct portions of a figure (e.g. it is very hard to find out whether three points are on the same line or to find out symmetries). Finally, sonification based solutions require a quiet environment or isolating headphones that monopolize the auditory channel of the user and thus cannot be used on the move or when the interaction with other users is required.

Auditory interfaces can be extended with haptic or tactile feedback to reduce the cognitive load required to sight impaired people to construct a mental representation of the scene. Audio-haptic educational games have been proposed both with general purpose haptic devices [19] and with new hardware devices employing vibro-tactile feedback [20, 21], or haptic gloves [22], specifically designed to generate tactile stimuli which reinforce audio cues. Haptic and audio-haptic systems have also been proposed for non-visual geometry and graph exploration and manipulation [18, 54, 55]. The main advantage of these systems consists in the ability to touch and manipulate the graph. Moreover,

guided exploration is also possible: the hand of the student is guided by the arm of the haptic device along the curve. These solutions proved to be more usable by sight impaired people. Nonetheless, haptic devices are often expensive, proprietary solutions, frequently available as prototypes only and they are not designed to be used in mobility. Moreover, the workspace is limited (about 15 by 15 cm), so graphs with many details can be hardly understandable.

For example, the Stickgrip system, used by Evreinova et al. [55] for the study of curves, is a proprietary add-on hardware for Wacom tablets. While the evaluation of the solution shows promising results, the hardware is still not available off-the shelf, and the usage of a cumbersome augmented stilus for mobile usage has not been investigated. The Gravvitas system, proposed by Goncu et al. [54] for the study of geometrical shapes, tables and graphs, suffers from a similar drawback. It requires add-on vibration motors on fingers used for the exploration on a touchscreen of a tablet.

Gravvitas [54], however, also expands the haptic approach with the usage of stereo audio feedback for the spatial exploration. This approach, while allowing to convey more information at the same time, requires the use of headphones that isolate the user and avoid inclusion among students. The solution proposed in Section 3, instead, conveys the spatiality only through the position of the finger on the screen coupled with the pitch quality of the sound, thus requiring only a mono sound system and no cumbersome hardware add-ons.

Touch Tiles [17] is an interaction technique using a force feedback mouse for detecting shapes of figures, designed for teaching geometry to children. While Touch Tiles does not require custom hardware, it needs the use of a specific mouse hardware, which makes it unfeasible to be used in mobility.

The use of tablet devices for map exploration is tackled by Poppinga et al. [56] showing promising results in conveying the spatial information. The study, however, highlights issues in exploration when the elements are too small with respect to the user's precision in exploration and, thus, sometimes overlooked (The "Fat-finger" problem [57]).

Giudice et al, [58] use the vibro-tactile output on tablet device to efficiently convey shapes of simple figures or graphs. The method explored in the following extends this approach by substituting the vibro-tactile feedback with three different audio-based interaction techniques.

Tactison, by Burger et al. [59], is an audio-haptic computer interface designed for education of children with visual impairments. A grid of items is explored and accessed through a programmable 16x16 overlay keyboard enhanced with tactile overlays that define exploration and function areas. Audio and text-to-speech output is used as feedback for user's actions. The interaction technique adopted in *MathMelodies*, the educational

tool for 1st-3rd grade school children proposed in Section 3.1, expands the approach described in [59] in two crucial ways. First, instead of an overlay keyboard, a touchscreen on a tablet device is used. This hardware is widespread and has a higher resolution and sensitivity, thus it is more suitable for presenting arbitrary sized and shaped objects to the user. Second, by extending the interaction paradigm with a graphic interface presented through the touchscreen, it is possible to use the device in the education activities for sighted children and children with different levels of visual impairment, thus improving integration and inclusion between students. Additionally, the solution proposed in Section 3.1 comprehends 13 exercises integrated in the context of a story, specifically tailored for capturing the attention and teaching spatial and mathematical concepts to 1st-3rd grade children.

2.2 Cognitive mapping in smartphone-assisted mobility

The research in the field of independent mobility for people with visual impairments addresses the problem of providing the user with contextual information regarding the surrounding environment, in order to make the cognitive mapping of the explored space and the resulting autonomous way finding of the visually impaired effective, efficient and safe. Indeed, the unassisted navigation is a major issue and a potential hazard for people with visual impairments [24, 60]. In blind test subjects, Arditì et al. note a difficulty in estimating the distances [26] while Beggs [23] relates a lower walking speed with respect to the sighted subjects to a safety response to the stress associated with travel. Passini and Proulx [7] show how people with visual impairments rely on a higher number of data sources for the way-finding and often navigate based on substantially different information than sighted people [61, 62]. Thus, navigation issues for the visually impaired often involve retrieving specific information useful for guiding the user and conveying those information to the user in an efficient manner.

The aspect of the way-finding that deals with the retrieval of information about the remote locations and long range navigation, not in the immediately perceptible environment, is called Macronavigation [63]. Instead, Micronavigation deals with the short ranged orientation, navigation and obstacle avoidance [63]. The difference between these two aspects lies in the fact that, while during Macronavigation it is possible to leverage absolute positioning systems (E.g., GPS) for localizing and orienting the user, during Micronavigation the inherent imprecision of the absolute positioning systems requires to fall back to a relative orientation and navigation.

The data leveraged during Macronavigation can be easily related to a spatial position consistent in time and between different users. Thus, Macronavigation issues are often related either to gathering new information about the environment, or finding new and

more suitable ways for conveying the already gathered information to the user. An example of an outdoors Macronavigation solution, proposed by Hara et al. [64], gathers data related to the surroundings of bus stops through crowd sourcing. The user is afterwards notified when in GPS proximity of a stop and, by leveraging the knowledge of the presence of objects such as dust bins, poles, traffic signs, and their spatial positioning with respect to the bus stop, the visually impaired user is assisted during the identification and the approaching to the stop. The iMove¹ application, available for iOS devices, reads to the user the current address as well as the list of nearby points of interest (e.g., shops, museums, bus stops, etc.). Sudol et al. [65] propose Looktel, an application suite to assist visually impaired users with object recognition and navigation tasks. The GPS navigation tool, Breadcrumbs, supports geotagging, path creation and social sharing. However it does not deal with navigation planning or local navigation task (e.g., street crossing). MoBIC [66] is a toolset aimed at helping visually impaired people during route planning and navigation. Google Intersection Explorer² is an application for the mobility of visually impaired users. It does not have a function of specifying the routes, but provides information about the location of the intersection. Indoors solutions, that cannot use GPS localization due to the precision limits of the system and the attenuation of the signal in covered spaces, instead can use RFID tags or bluetooth beacons [67, 68].

The Micronavigation issues deal with short range way-finding, typically within 20m, characterized by the difficulty to reuse the gathered information due to the absence of an absolute positioning (Given the GPS imprecision) or due to the high mutability of the environment caused by the mobility of the elements involved (cars, pedestrians), periodical mutations of the environment (E.g., traffic lights change color), or unforeseen changes (E.g., roadworks). This happens in the outdoors environments when the navigation is at a scale too small to be positioned precisely with the GPS or, generally, when the absolute positioning infrastructure is unavailable or too expensive, given the cost and time requirements for adding and maintaining hardware augmentations in the environment.

Instead, relative positioning can be obtained by exploring the environment through computer vision techniques [69–71] or through position estimation techniques (dead reckoning) using onboard motion sensors [72, 73]. The usefulness of the gathered data in this case is related to the precision of the computed relative position with respect to the user and it diminishes once the positioning precision decays [74].

The issue is exacerbated by the intrinsic visual nature of many landmarks, such as traffic signs, that a visually impaired user cannot access through haptic interaction (touch or white cane).

Early work using computer vision in the field of assistive technologies for persons with visual impairments have been explored by Malek Adjouadi [75]. In particular, in the

¹itunes.apple.com/en/app/imove/id593874954

²play.google.com/store/apps/details?id=com.google.android.marvin.intersectionexplorer

field of way-finding, Adjouadi proposes a stereoscopic Micronavigation system capable of path computation and obstacle detection and avoidance [71].

For what concerns the detection of pedestrian crossings, the first solution proposed in literature (Stephen Se [76]) adopts the following approach: first, line segments and their vanishing points are detected through Hough transform. Another application of the Hough transform finds groups of parallel segments. The outliers are filtered out by a Random Sample Consensus algorithm. The result, expected to be a set of line segments belonging to the same zebra crossing, is then validated using cross ratio constraint. The problem of this solution is that it can recognize a zebra crossing only if the image contains the whole pattern, not covered even partially by objects (e.g., a car). Additionally, patterns that are not zebra crossings but have similar ly parallel lines, can be detected as crossings. A more effective approach (Uddin et al. [77–79]) first applies a bipolarity segmentation to detect areas of alternating black and white stripes and then validates the result through cross ratio invariant verification. This solution yields good results in terms of precision and recall, although the experimental evaluation has been conducted on a small data set (about 100 images), all with similar illumination conditions.

In particular, for the navigation of pedestrian crossings, the Crosswatch system [80, 81] has been proposed. Micronavigation tools proposed in it include, in particular, two techniques for detecting pedestrian crossings. One solution detects zebra crossings but does not compute their relative position with respect to the user that is a necessary step to guide the user towards the crossing [82]. The second solution detects the “two stripes” pedestrian crossings and adopts a rectification technique that, while not described in detail, appears to be similar to the one proposed in this work [83]. Another new contribution to the Crosswatch project, by Murali et al., presents an innovative approach to the related issue of estimating the user’s position in an intersection [84]. The idea is to acquire 360° image panoramas while turning in place on a sidewalk. The image panorama is then converted to an aerial (overhead) view of the nearby intersection, centered in the user’s location. The goal is to match this aerial view with a template of the intersection obtained from a satellite image and estimate the user’s location in the intersection more precisely than with GPS.

An issue tightly related to the pedestrian crossing detection is the traffic lights detection. Angin et al. [85] developed a system which performs fast recognition of traffic lights by exploiting the computational power of cloud computing. The response times of this approach have been tested over a high speed wi-fi networks, yielding acceptable results (660ms on average). However, the performance on different mobile network types (GPRS, 3G, LTE...) has not been tested. Since the solutions proposed in the following are executed on the mobile device, they provide the user with real-time information about crosswalk position regardless of the current connectivity status.

Chapter 3

Didactic assistive technologies

Section 3.1 explores two different interaction techniques considered for the touch-screen based learning on mobile devices: sonification and object-based interaction. The use of object-based interaction for assisting visually impaired grade school children in learning basic math has been previously published in [29] and it is described in Section 3.2. Section 3.3, instead, presents a study regarding the accessibility of math function graphs through sonification techniques (published in [30]).

3.1 Interaction techniques

Students with visual impairments face many difficulties while studying scientific subjects. This is due to the fact that the teaching of concepts requiring spatial understanding frequently leverages graphical representations for efficient learning. Touch screen interfaces can assist persons with visual impairments to understand spatiality through proprioceptive exploration of the screen. Indeed, the touch screen can be considered not only an input interface but also a source of information for the user.

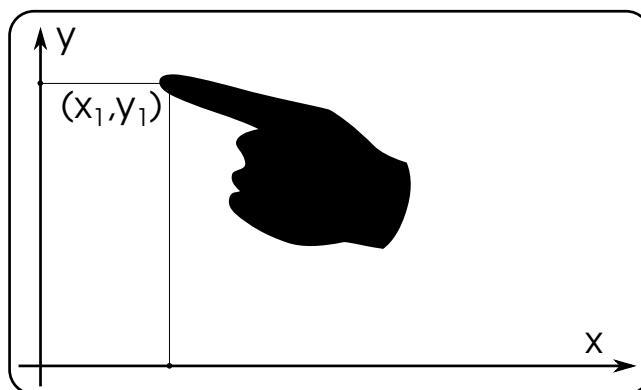


FIGURE 3.1: The knowledge of the touched position is in itself a source of spatiality information

The fact that the user is aware of the position of the area explored on the touch screen conveys spatial information that can then be used to form a cognitive map of the elements on the screen and their relative positions (See Figure 3.1). Besides the coordinates of the explored position, the information regarding the contents of the investigated area is also required for the understanding of the examined space. This information can be conveyed through different sensory substitution paradigms, two of which considered during this work: sonification and object-based interaction.

The sonification interaction allows to re-code the visual information through audio-feedback [86]. Indeed, a sound can carry information through its temporal, spatial, amplitude, and frequency qualities and therefore it is possible to convey the spatiality notion by manipulating these qualities. Coupled with tactile exploration, this approach has been used in literature for the detection of edges in figures [86] and for the exploration of pictures by transforming the color at the examined point into a sound conveying its luminosity and tonality qualities [87]. As an example of this method consider the Figure 3.2a. The light intensity can be conveyed through the volume of the sound. Thus, point *B* will be translated as a louder sound than point *A*.

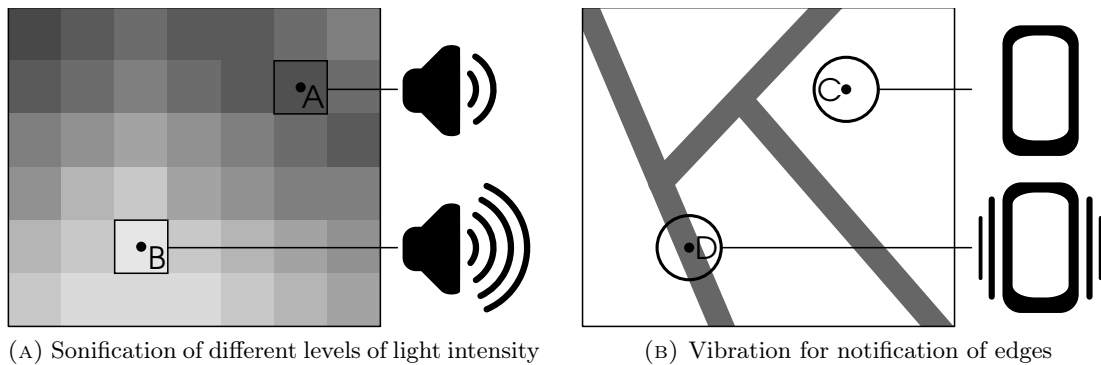


FIGURE 3.2: Sonification and Vibration based sensory substitution techniques

Note that vibro-tactile feedback, which will not be considered in the following, offers similar sensory substitution capabilities as the sonification (Giudice et al [58]). For example, for edge exploration, the presence of an edge at the explored point *D* can be notified by a vibration, absent when exploring a point *C* not on an edge (see Figure 3.2b). However, many existing devices have either simple on-off control or limited timing or intensity level control of the vibration. Therefore, this approach is not suitable when a higher expressiveness is needed (e.g., for conveying many levels of variation in light intensity, as in Figure 3.2a).

The object based interaction is conceptually similar to the interaction adopted by screen readers (e.g., VoiceOver on iOS). Screen readers are accessibility software for touch screen interfaces. The user slides the finger on the touch screen and explores the content of the interface while the screen reader vocally describes elements currently selected. In case of object based interaction, however, instead of long vocal descriptions, short and easy to remember sounds (often onomatopoeic) are adopted as descriptors. Each logical element

of the interface has a specific audio feedback that acts as its defining cue and the audio feedback is independent with respect to the position of the touch within the object. For example, as shown in Figure 3.3a, touching a point F that belongs to the figure of a dog, the test application plays a sound of a dog barking.

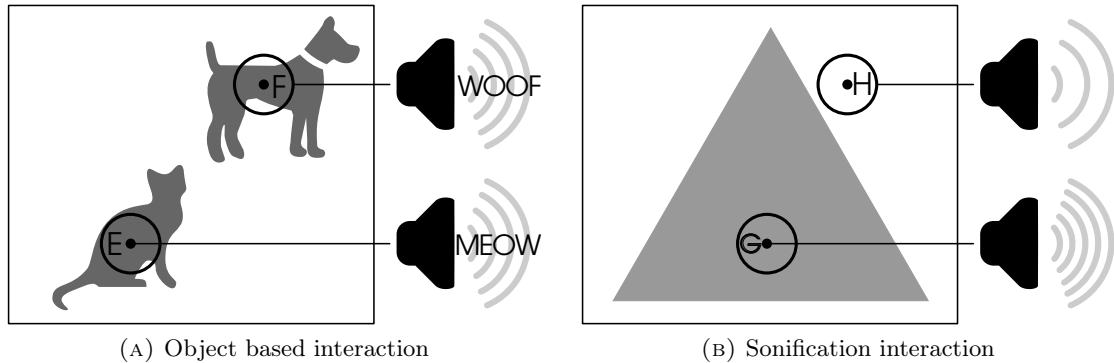


FIGURE 3.3: Object-based and Sonification interaction

3.2 Math learning for children with visual impairments

The two touch-screen based interaction paradigms defined in the previous section, sonification and object-based interaction, were evaluated. The goal of the evaluation was to define which approach to use in the context of a didactic tool for the teaching of mathematical and geometrical concepts to 1st-3rd grade school students with visual impairments. Teachers expert in education of blind students were involved for evaluating the feasibility of the two approaches and the prototype test software implementing the two techniques was tested by three blind children. The first test focused on the evaluation of the sonification approach and involved the recognition of simple black and white figures. The second test considered the object-based interaction and focused on three exercises: “counting”, “position in a table” and “spelling”.

In the test involving the sonification interaction the goal was to recognize simple geometrical figures depicted on binary black and white images. The user scans the touch screen interface with a finger and the presence or absence of the image underneath the finger is conveyed through sound pitch. As shown in Figure 3.3b, low pitch sound is constantly played while the user is exploring the touch screen without being near or touching the figure. The pitch of the sound gradually increases as the user’s finger nears the figure and it is highest when the user is exploring inside the figure.

Specifically the pitch of the sound is computed as follows. Given a figure G and a touched point p , a circle A with center in p and of radius r is computed. The intersection I between the circle A and the explored figure F is computed and a sound with the pitch $P = P_{min} + (P_{max} - P_{min}) \cdot \frac{I}{A}$ is played, where P_{min} and P_{max} are parameters defining the minimum and maximum pitch respectively. This means that, when the user is exploring

far from the figure, the intersection $I = 0$ and the sound played has pitch $P = P_{min}$, and when the user is exploring completely inside the figure, the intersection $I = A$, and therefore the sound played has pitch $P = P_{max}$. For example, in Figure 3.4, the area I' is smaller than area I , thus, the sound yielded when exploring p' will have a lower pitch than the sound played when exploring the point p .

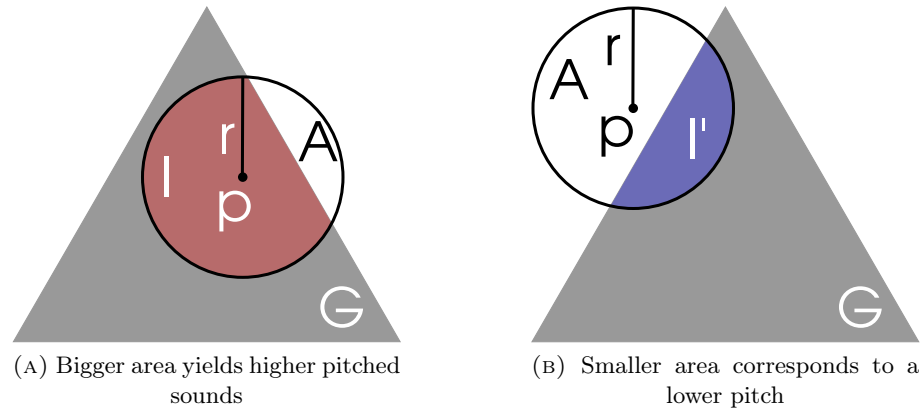


FIGURE 3.4: Sonification of geometrical figures

The “counting” exercise consists in exploring the interface in which different types of objects are contained and, afterwards, recall the number of objects of a certain type. Each object type corresponds to a specific sound which is played when the user explores an element belonging to the typology. For example, Figure 3.5 contains cats, dogs and horses. When, while exploring the interface, the user encounters one animal, the call of the animal is played. At the end of the exercise the user is asked to specify the number of one type of animals in the picture.



FIGURE 3.5: “Counting” exercise

The “position in a table” exercise consists in exploring a table containing different elements (e.g., animals, whose call is played when the user explores the corresponding cell)

and finding the position of the cell of a specific element in terms of its coordinates. The coordinates are specified as tuples of column and row indexes where column indexes are lexically ordered letters starting from *A* and row indexes are ordered numbers starting from 1. Figure 3.6 shows an example of this type of exercise.





	A	B
1		
2		

FIGURE 3.6: “Position in a table” exercise

The “spelling” exercise, shown in Figure 3.7, consists in correcting a misspelled word. The call of an animal is played and a sequence of letters shows the misspelled name of the animal. The user explores the sequence and each letter is read when encountered. An empty sequence is proposed underneath and the user can select the cells of the sequence and insert the letters with an input keyboard at the bottom of the screen.

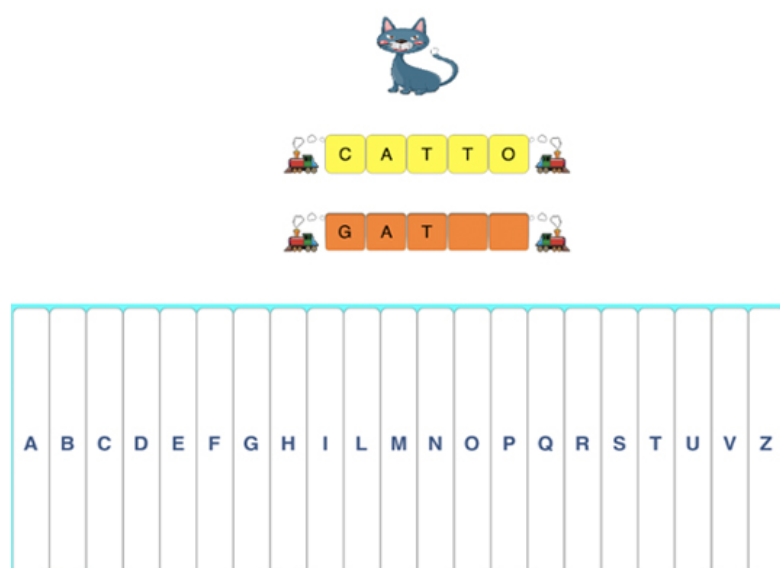


FIGURE 3.7: “Spelling” exercise

The experimental evaluation considered the mental load of the two proposed interaction paradigms and has shown that the object-based interaction is less cognitively demanding, hence resulting more enjoyable for the users and more suitable for didactic tools aimed at children. For details of the evaluation and results see Chapter 3.2.3.

Afterwards, in accordance to the obtained results, a set of 13 different types of exercises relying on the object-based interaction paradigm were designed. The developed solution, named *MathMelodies*, is an iPad application that supports primary school children in learning Mathematics. Thanks to a crowd-funding campaign, the software has been engineered and distributed as a commercial application¹.

3.2.1 Design challenges

During the design and development of *MathMelodies* there were three main challenges. First, the application has to present exercises that are accessible to visually impaired children. After the testing of different interaction techniques, the object-based interaction was chosen for the design and development of the user interface. In order to further simplify the interaction model, the objects were organized into a grid layout that, as observed during the evaluation, helps reducing the time and mental workload required to explore the entire screen. Another choice driven by the need of simplifying the interaction consisted in the definition of two input techniques: a simplified on-screen keyboard to insert the digits only (e.g., for the addition exercises, see Figure 3.8) and a multiple choice dialog (e.g., to answer an exercise like the one shown in Figure 3.9).

Finally, it appeared that the interaction with the exercises without any preliminary explanation is not intuitive for some children. Also, for some exercises, the eyes-free exploration of the screen can be time consuming. To address these two problems, a short explanation was added that is automatically read when a new type of exercise is presented.

The second design challenge is to stimulate children to play the exercises several times hence taking benefit from reinforcement learning. To address this challenge, the exercises were designed to have up to 6 difficulty levels. For example, in the “easy” addition exercise the child is asked to add two single-digit numbers, while at a harder level (designed for 3rd grade students) the aim is to add three numbers, each one with up to three digits as in Figure 3.8. For the same reason, the exercises are defined in terms of their type and difficulty level and not according to their actual content that is randomly generated each time the exercise is presented to the student.

Another important aspect to stimulate children to play the same exercise several times is to entertain them. This objective was pursued by presenting, in most of the exercises,

¹ itunes.apple.com/us/app/math-melodies/id713705958

Sum these numbers

H	T	U	
1	2	9	+
	1	1	+
4	1	4	=

1 2 3 4 5 6 7 8 9 0 C X

The interface features a grid for the math problem and a simplified keyboard at the bottom with buttons for digits 1-9, 0, a clear button (C), and an exit button (X). In the top right corner, there are three circular icons: an information icon (i), a red 'X' for incorrect, and a green checkmark for correct.

FIGURE 3.8: Addition exercise with simplified keyboard

“audio-icons”: amusing drawings, each one associated with an easy-to-recognize and funny sound. Also, the application gives a reward to the child in the form of a short piece of music when a correct answer is provided. As a future work a more sophisticated reward mechanism will be explored, so that also the number of wrong answers the child provided before giving the right one are taken into account. For example, zero mistakes can be rewarded with 3 “golden stars”.

Which animal appears more times?

The interface shows two rows of animal illustrations on a grid. The top row contains four identical brown rabbits, each holding a bone. The bottom row contains three identical orange cats sitting and eating. In the top right corner, there are three circular icons: an information icon (i), a red 'X' for incorrect, and a green checkmark for correct.

FIGURE 3.9: Counting exercise

The last design challenge is to immerse the educational activities in an accessible entertaining environment that also links the exercises together and motivates children to keep on playing and practicing. This challenge was addressed through a tale, divided into 6 chapters, organized in increasing difficulty levels (two chapters for each grade).

Each chapter is further divided into “pages”, each one comprising a background image, some text (read by a speech synthesizer) and some “audio-icons” i.e, colorful drawings associated with a funny sound (see Figure 3.10). Pages are intertwined with the exercises and there are about 30 exercises in each chapter. Overall, the tale and the audio-icons have also the objective of triggering children’s interest. This is similar to the approach adopted in most textbooks that heavily rely on colorful images. The difference is clearly that in *MathMelodies* this solution works for visually impaired children too.



FIGURE 3.10: Two pages of *MathMelodies* story

3.2.2 Proposed solution

The effort to develop *MathMelodies* can be divided into two main activities: the actual app implementation (i.e., the code writing) and the creation of the content: the story text, images (backgrounds and icons), and audio (sounds and music).

The story is composed by six chapters, each one including about 50 pages similar to the ones depicted in Figure 3.10. Since the app has been localized into two languages (Italian and English), 600 pages of content in total were produced. For what concerns the images, there are 25 backgrounds (e.g., the theater curtains in Figure 3.10) and about 100 audio-icons (like the dog in Figure 3.9 or the piano in Figure 3.10). Also, each chapter has an associated music that is played at the end of the chapter and that is also divided into small parts, each one played after each exercise for a total of about 200 (short) pieces of music.

During the app implementation two major technical issues were faced. The first issue deals with the large amount of app content. Indeed, it is clearly impractical to define the app by hard-coding the content into the program. Instead a format for the content package, that describes, for example, the structure of each chapter, each page, etc, was engineered. A “content engine” in *MathMelodies* reads this package and presents the content to the user, in the form of exercises, pages, etc... Thanks to this approach, it is possible to define the app content independently from the app implementation. Also,

through a user friendly tool, it is possible to edit the content packages. Thus, it is also possible for non-technicians to define the app content.

The second issue relates to the implementation of the object based interaction paradigm built on top of the system accessibility tools. On iOS devices, there are two sets of system accessibility tools specifically aimed at visually impaired users. One set of tools is designed for low-visioned users and includes the “zoom” screen magnifier, font adjustment and color inversion. Ad-hoc gestures are defined to use the zoom functions but the overall interaction paradigm is analogous to the one for sighted users. The second set of tools is aimed at blind users and it is globally called “VoiceOver”. VoiceOver proposes a totally different interaction paradigm, based on screen reader approach. The overall idea is that, when the user touches a graphical object on the screen (e.g., a button), VoiceOver gives it the focus and describes it both with a speech synthesizer and an external Braille display (if connected). To activate a focused object (e.g., to press a button), the user double taps anywhere on the screen. In addition to this basic behavior, VoiceOver has several additional gestures to make the interaction more efficient.

In order to enhance the app usability for visually impaired users that rely on residual sight, large fonts and high contrast between the foreground objects and the background was used. Although this solution was not evaluated sufficiently, the app is expected to be accessible to most low-visioned students by using the default accessibility tools.

For what concerns blind users or low-visioned users that cannot totally rely on residual sight, some issues arose in the implementation of the object-based interaction paradigm. Indeed, the simplest solution to implement this paradigm would be to fully rely on VoiceOver (i.e., not implementing any custom behavior for app accessibility). This approach would make it possible to develop an app that is totally consistent with the system-wide interaction paradigm. However, this solution suffers from a major drawback, as it is not suitable to address all design challenges. For example, without defining any custom behavior it is not possible to develop the audio icons that, when VoiceOver is active, play the associated sound upon getting the focus. Other features that call for a custom behavior are multi-tap exploration and automatic reading when a new page is shown.

Clearly, to achieve a deeper customization of the interaction paradigm a larger coding effort is required and it is quite involved to mimic VoiceOver standard behavior as well as to guarantee the consistency with the system-wide interaction paradigm. For example, current version of *MathMelodies* (1.0) uses some custom objects in the story view: to enable the automatic reading of a page, story text is “hidden” to VoiceOver and “played” automatically with iOS 7’s integrated text-to-speech synthesizer. This approach was chosen for the app to be of immediate use also to users that are still not acquainted to standard VoiceOver gestures. Preliminary tests, presented in Section 3.2.3, validated this approach. However, feedback from users highlighted two problems: first, the text is not

shown on external assistive devices such as Braille displays. Second, the text-to-speech synthesizer does not use VoiceOver settings, hence the talking speed is not the one chosen by the user through VoiceOver preferences. To address these issues, as a future work, a new version of the software will minimize the use of objects with custom behavior.

3.2.3 Evaluation

The whole design and development process benefited from the feedback obtained from one of the designers who is blind and experienced in education for blind persons. In addition to this constant feedback, three main evaluation sessions were organized.

The first session was organized with four teachers expert in education for blind students². The evaluation was divided in two steps. The former consisted in a list of the exercises derived from Italian educational directives and integrated with workbooks and online resources. For each exercise the experts were asked to evaluate the importance of the exercise in the education of a blind person and to rate how difficult it is to practice it with existing solutions. In the second step of the evaluation the preliminary prototype of *MathMelodies*, implementing sonification-based and object-based interaction paradigms, was presented. All four experts independently agreed on the fact that the object-based paradigm would be quicker to learn and also more adaptable to a larger variety of exercises.

The second session was conducted as a test with three blind users. Table 3.1 lists the information about the test subjects of this test. After a short training with the prototype, each user was asked to solve three exercises with object-based interaction and one exercise with sonification-based interaction.

User	Age	Sex	Type of vision loss
1	7	female	Total blindness (right), color and light perception (left)
2	8	male	Total blindness
3	10	male	Color perception, visual residue 1/100

TABLE 3.1: Information about the users of the second evaluation session

All students have been able to complete and correctly answer exercise 1 (counting) and 2 (position in a table). Vice versa, one student has not been able to complete (and hence to provide an answer to) exercise 3, a spelling exercise, and exercise 4 consisting in recognizing a triangle by a sonification-based interaction. Overall, all students reported that the object-based interaction is easier to understand and two of them also highlighted that it is funnier.

The third evaluation session was conducted with three primary school blind children and with two primary school sighted children. The information about the testers is shown

²From the center for the blind people in Brescia, Italy.

in Table 3.2. All children were required to complete all the exercises in the first chapter consisting in counting exercises, sums, etc.

User	Age	Sex	Type of vision loss
1	6	female	visual residue 1/100
2	8	male	Total blindness
3	9	male	Total blindness
4	9	female	
5	8	male	

TABLE 3.2: Information about the users of the third evaluation session

All blind children were enthusiast while using the application. Two out of three reported that they were entertained and engaged especially by the sounds used (e.g. the call of animals and the rewarding melodies). All of them experienced some difficulties in the early exploration of the tables, and needed some help by a sighted supervisor. However, after at most 2 minutes of supervised training, all children got familiar with the application and were able to solve the exercises autonomously and, most of the times, providing a correct answer at the first attempt. The two sighted children enjoyed the application as well. One of the two children experienced some difficulties, at the beginning, in understanding how to answer. This was partially due to the fact that the child didn't pay much attention to the exercise explanation. After explaining how to answer, no more help was needed. To solve this problem, a future work will be to create introductory exercises in which the focus is not on the exercise itself rather to explain how to use the application.

As a future work, *MathMelodies* will also be evaluated against other didactic solutions for children such as Tactison [59] and Touch tiles [17] (See Section 2.1).

3.3 Analysis of functions through sonification

Differently from the didactic tools for children, proposed in Chapter 3.2, the issue of function analysis requires an interaction method that allows fine resolution exploration with a faster responsiveness than the object-based interface. The object-based interaction is suitable for the purpose of conveying information related to a small number of objects described by complex sounds or speech cues. For example, if an object represents a car, it is possible to use the sound of a car for notifying the user when the object is touched on the touch screen. However, for describing the shape of a function in an efficient manner, during the exploration, every point of the function has to immediately yield an output while no output has to be given as soon as the user stops exploring within the boundaries of the function. While the responsiveness of the interaction method is a priority, in this case, given the lower amount of information to be conveyed at a time, the expressiveness of the interaction technique is not required to be high. Indeed, for each

point, it is sufficient to communicate the presence of the function and, in some cases, as seen in the following, the value of the function. The sonification interaction, described in Section 3.1, is suitable for this type of exploration.

Section 3.3.1 describes the three sonification based interaction paradigms designed for the study of function graphs while Section 3.3.2 details the functionalities of the developed prototype software. Section 3.3.3 discusses the experimental evaluation of the solution.

3.3.1 Function graph exploration techniques

Three interaction techniques were designed and developed for the purpose of spatially describing function graphs. The first interaction technique, called “Non-interactive sonification” is similar to techniques existing in literature [12, 50]. Purely sonification based, this approach does not offer proprioceptive interaction. Instead, it describes the function graph by dividing the function domain in a sequence of intuitively small intervals (e.g. the rectangle r in Figure 3.11) which are then converted to sounds and played as follows. For each interval, the app computes the value of $y = f(x)$ where x is the minimum value of the interval. The software then assigns to each interval the “value-sonification” for y , i.e., a sound whose pitch is proportional to the value of y with respect to the range of y values. The computed sounds are finally played in sequence, starting from the interval having the lowest x coordinate and up to the highest x coordinate.

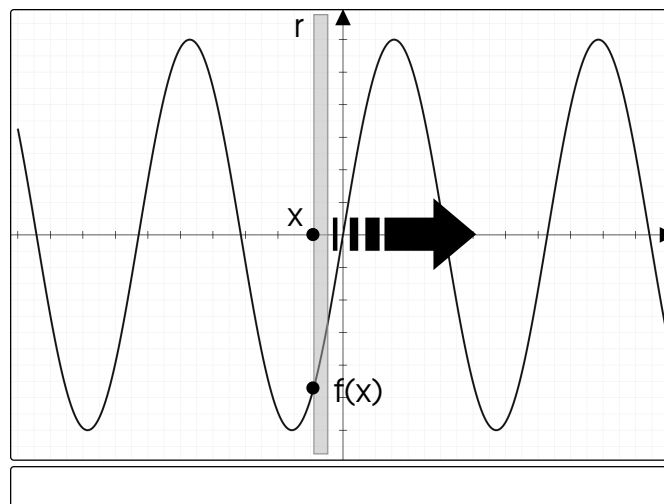


FIGURE 3.11: Non interactive sonification

The second technique, which is called “mono-dimensional interactive sonification” (See Figure 3.12), adopts a proprioceptive approach. The proposed interaction paradigm is the following: the user explores an area of the touch screen with a finger and only the horizontal movement is considered. While sliding the finger, the current value of the function $y = f(x)$ is computed, where x corresponds to the current finger position. The computed value is then transformed to sound in a similar fashion as in the previous

technique. That is, lower values yield lower pitched sounds while to higher values corresponds a higher pitched sound. The clear advantage of this exploration mode is that, thanks to proprioception, the user can perceive the current x position and thus grasp the shape of the function graph. Also, the user can move forward and backward along the x axis, at the desired speed, hence, for example, focusing more on some parts of the functions that are more relevant for the user (e.g., a minimum point).

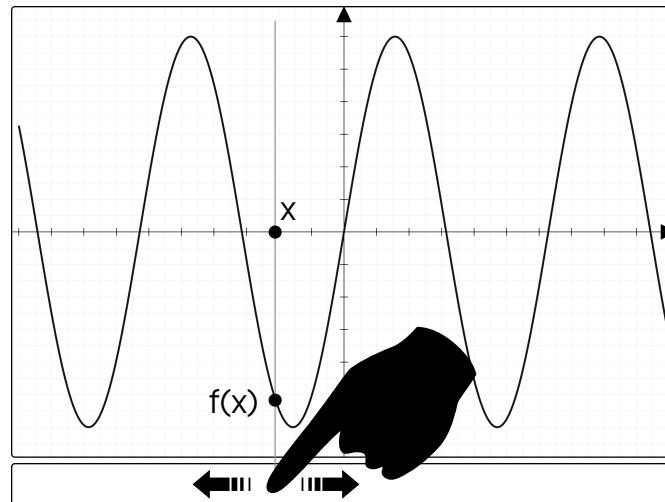


FIGURE 3.12: Mono-dimensional sonification

The third exploration mode, shown in Figure 3.13, is called “bi-dimensional interactive sonification”. The overall idea is to make it possible for the user to follow the shape of the function graph with one finger on the touch screen. The proprioceptive approach facilitates the user’s cognitive mapping of the graph’s form. This mode adopts a different sonification, called “position-sonification”, for signaling the presence of the curve in the investigated area, since the aim is not to encode the y value, but rather to guide the user while following the plotted line. Conceptually, this approach is similar to the sonification technique explored in Section 3.2. When the user touches the graph line, the position-sonification plays a sound with the highest pitch. Instead, touching outside the line, the pitch diminishes as the distance between the touched position and the line increases.

In an early stage a simpler technique, in which a sound was played only when the user was on the line of the function, was adopted. This approach had two significant issues for which the development of the “position-sonification” was needed. First, if the line was too thin, it was very difficult for the user to find the line during the exploration, and if the line was too thick, it was hard to understand the shape of the curve. Second, without a way to understand when the user was about to exit the area of the line of the graph, the exploration consisted in constant exiting and reentering the curve, which penalized the user’s understanding of the function graph.

3.3.2 *AudioFunctions* prototype

AudioFunctions is an iPad prototype that helps people with visual impairments in study and comprehension of the functions graphs by implementing the three navigation methods proposed in the previous section. It benefits from the non-mediated interaction, typical of touchscreen devices, to implement the three techniques for the exploration of function graphs described in the previous section. Differently from existing software solutions, two of the interaction techniques proposed by *AudioFunctions* highly rely on proprioception.

The use of *AudioFunctions* can be divided in two main activities: the specification of the function expression as well as of its drawing properties (Section 3.3.2.1) and the function graph exploration (Section 3.3.2.2).

3.3.2.1 Specification of function expression and drawing properties

The dialog for the specification of function expressions uses the VoiceOver screen reader (Section 3.1) for interaction, thus allowing visually impaired users to access this interface. However, given the complexity of the function formulas, an initial training by an instructor is useful for introducing the function expression syntax to a student learning about functions. As a future work, a tutorial module will be designed for this goal.

To specify a function expression, a user can choose a template and then edit it (see Figure 3.14). The template can be chosen from the list of “default” expressions (i.e., a pre-defined set of common functions, like $y = x$, $y = x^2$ or $y = \sin(x)$) or from the list of recently used expressions, as they were edited by the user (in Figure 3.14 the list of recently used expressions is hidden by the keyboard). To edit the function expression, *AudioFunctions* presents an ad-hoc keyboard, that is similar to a calculator keyboard and

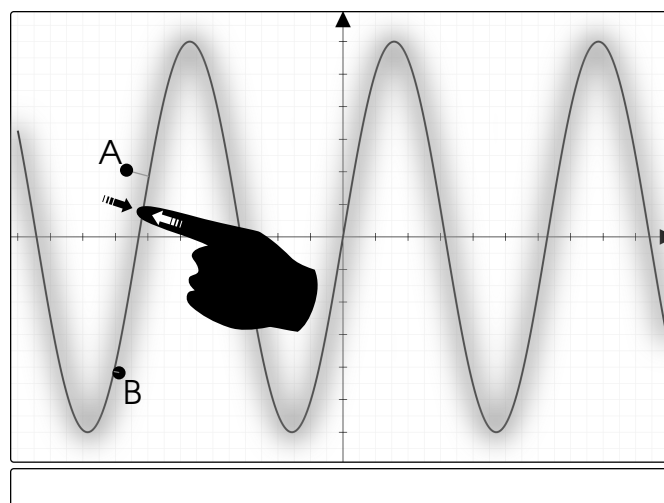


FIGURE 3.13: Bi-dimensional sonification

that contains keys for the digits and for the most common arithmetic and trigonometric operators.

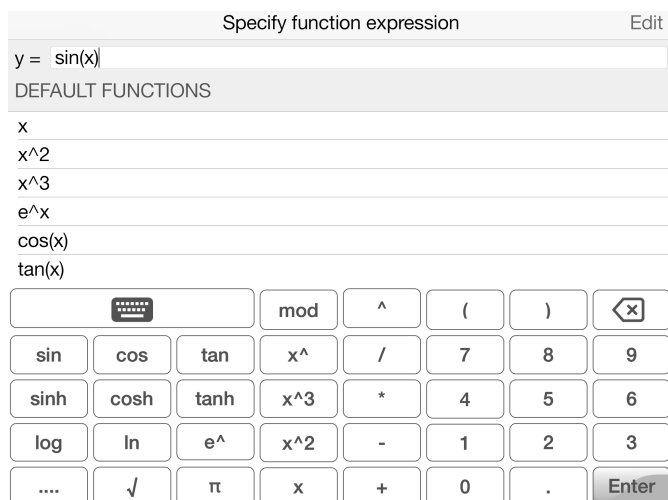


FIGURE 3.14: Specification of function expression

The function drawing properties (see Figure 3.15) include options to define the domain and the scale on the two axes. With the first option, the user can set the function domain in terms of the minimum and maximum values of x to be represented. The second property is a boolean value indicating if the y axis should have the same scale as the x axis. If this property is set to “true” (the default value) then the next two options are disabled. In case this property is set to “false”, with the third option the user can choose to automatically scale the y axis, which means that *AudioFunctions* chooses the largest scale for the y axis such that the function graph fits in the screen. If “automatic scale” is disabled, then the user can manually choose the scale for the y axis, by indicating the minimum and maximum values to represent on the y axis.

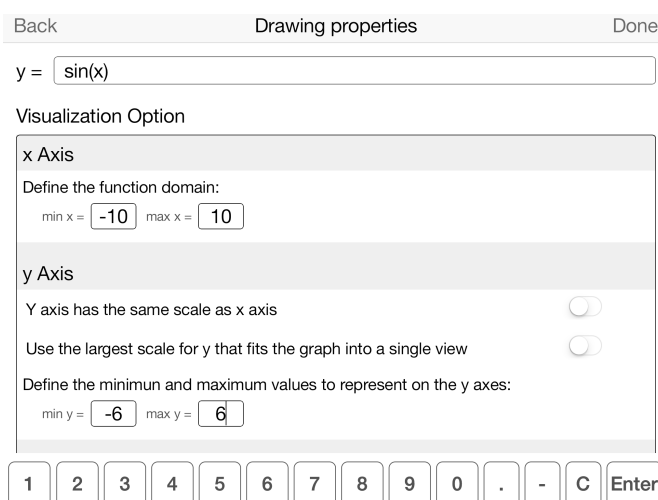


FIGURE 3.15: Specification of drawing properties

3.3.2.2 Function graph exploration techniques

The exploration of the function graph supports the three “exploration modes” described in Section 3.3.1. The three interaction techniques complement each other and all of them can be accessed directly from the main exploration interface.

The implementation of the “non interactive sonification” is analogous to the solution proposed in Audio Graph Calculator software³, discussed in Section 2.1: the software accepts as input the equation of a function, shows its graphical representation and also outputs the function as a sequence of pure sounds representing the curve shape. The y axis scale chosen during the function input (See Section 3.3.2.1) defines the value-sonification. That is, the minimum and maximum pitch are assigned respectfully to the lowest and highest y coordinates represented. For example, as shown in Figure 3.11, with the function $y = \sin(x)$, for $x \in [-10, 10]$, when $x = \pi/2$ then $y = 1$ which is also the maximum value for y and hence the value-sonification for $x = \pi/2$ has the highest pitch. Vice versa, if $y = x$, for $x \in [1, 10]$, the sonification for $x = 1$ has the lowest pitch, because 1 is the smallest value represented for y . A “double two finger tap” gesture⁴ anywhere on the exploration interface causes *AudioFunctions* to start playing the function sonification.

The second exploration mode, “mono-dimensional interactive sonification” (Figure 3.12), is accessed from the white bar at the bottom of the interface. The position of the bar helps the user in finding the area swiftly. Afterwards, as explained in Section 3.3.1, the user can investigate the function by sliding the finger laterally along the white bar which represents the x axis of the function graph. The corresponding $y = f(x)$ value is computed given the current position of the user’s finger on the bar as the x coordinate and it is interpreted as sound in the same manner as in the “non-interactive” technique.

The “bi-dimensional interactive sonification” is accessed from the main area of the touchscreen interface which contains the visual representation of the function graph. This area covers all the interface minus the white bar at the bottom of the screen used for the “mono-dimensional interactive sonification”. The user can explore the whole area while a sound notifies when the function graph figure is in proximity. As explained in Section 3.3.1, the sound will have a low pitch at higher distances from the graph while the highest pitched sound will be played when the user explores exactly on the function graph. For example, in Figure 3.13, point A is more distant from $f(x)$ than point B . Therefore, when the user touches A a low pitch sound will be played while touching B will yield a high pitch sound.

The two interactive modes have some additional features. First, while exploring, *AudioFunctions* plays some additional sounds in case the function intersects some “points

³www.viewplus.eu/products/software/math

⁴This is the gesture that on iOS devices is associated, for example, to start and pause the music.

of interest”, like intersections with axes, local minima and maxima and changes in the concavity. Second, interaction with two fingers is supported. This is very useful, for example, when it is necessary to maintain a reference point in the exploration. To achieve this, when a second finger touches the screen, *AudioFunctions* starts playing the sound associated to that finger, ignoring the first one. Third, by double tapping, *AudioFunctions* reads details on the current position, including: the values of x and $f(x)$ and the function concavity in that point. This is useful because function concavity is easily understandable by sight, but hard to figure out with these sonification techniques.

Through extensive experimental evaluation, presented in Section 3.3.3, it is shown that *AudioFunctions* is more effective and expressive with respect to existing paper-based and software solutions.

3.3.3 Evaluation

AudioFunctions tremendously improves, with respect to the other software solutions, the effectiveness of the application in terms of how clearly the user understands a function graph. Actually, *AudioFunctions* is so accurate that the testing users better recognized the function’s properties with it (after five minutes of training only) rather than with tactile drawings (that every tester was well trained to use). Clearly *AudioFunctions* also has the great advantage, with respect to tactile drawings, to allow the user to study function graphs in total autonomy.

The main objective of *AudioFunctions* is to let the user perceive the shape of a function graph. Therefore the focus of the experiments was to determine how precisely a user can recognize the function properties from the exploration. The experiment was conducted with 7 blind users, all with some education in Mathematics (at least high school) and acquainted with tactile drawings. Table 3.3 lists the information about the test subjects of this test.

User	Age	Sex	knowledge of functions	Type of vision loss
1	28	male	high	color and light perception
2	27	male	low	color and light perception
3	43	male	medium	total blindness
4	29	female	high	total blindness
5	35	female	medium	total blindness
6	18	female	low	color and light perception
7	33	female	low	light perception

TABLE 3.3: Information about the users of the second evaluation session

During each test session *AudioFunctions* was first described in about 2 minutes and then the test subjects were left 3 minutes to get familiar with the app. To measure the level of understanding of a function, each user was asked to explore (without time limitations) three different graphs with the three tools: *AudioFunctions* (See Section 3.3), tactile

drawing on embossed paper (see Figure 3.16) and AGC software (Both described in details in Section 2.1). The order of the three steps was random.

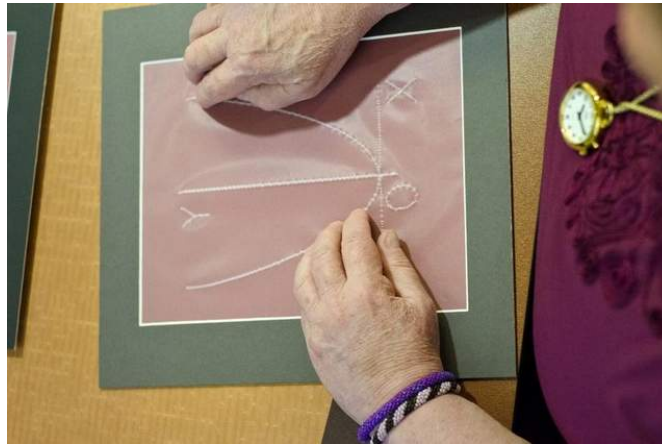


FIGURE 3.16: Example of tactile drawings used for the evaluation

During the usage of *AudioFunctions*, the user could freely access all three exploration modes described in Section 3.3.1. The embossed paper drawings were etched on a Sewell kit rubber clipboard and some supporting elements such as axes and numbered ticks on the axes were drawn for enabling the user to answer to the questions about the function's properties. In case of AGC, the user could tweak the sonification speed and loudness and access the value table that associates to different x values the corresponding y value of the function. During each step a random function expression was chosen from a set of pre-defined functions (See Table 3.4) presenting the corresponding graphs to the user.

N	Function	Domain
1	$5 \sin(x)$	6.28
2	$5 \sin(x) - 1$	6.28
3	$5 \sin(x) + 1.5$	6.28
4	$-4 \sin(x)$	6.28
5	$-4 \sin(x) + 2$	6.28
6	$-x^2 + 3$	6
7	$-x^2 + 5$	6
8	$-0.5x^2 + 4$	8
9	$ x^2 - 3 $	6
10	$ 12 \sin(x) $	6.28
11	$ 12 \sin(x) + 2$	6.28
12	$ 14 \sin(x) - 3$	6.28
13	$4 \sin(x) + 4 \cos(x)$	7
14	$4 \sin(x) + 4 \cos(x) + 1$	7
15	$-4 \sin(x) - 2 \cos(x)$	7

TABLE 3.4: List of functions used for the test

The user was then asked to answer 8 questions about the explored functions. While answering each question the user was free to interact with the exploration tool. The questions asked were the following:

- What's the value of the function in $x = a$, where a is a random integer value within the function domain?
- What are the domain intervals in which the function is increasing/decreasing?
- In which points does the function intersect the x axis?
- Does the point $p = (x, y)$ belong to the function, where x and y are random integer values within the function domain/codomain respectively?
- What are the domain intervals in which the function is positive/negative?
- What are the coordinates of the minimum and maximum points?
- Is the function convex or concave in $x = a$, where a is a random integer value within the function domain?
- Is the function a line, a parabola or a sinusoid function?

The answers and the time needed to provide them was recorded. Each answer was scored with a mark of 0 (totally wrong answer), 1 (no answer), 2 (partially correct answer) or 3 (correct answer). For questions requiring a numerical value as answer, the correct answer was considered rounded up to the closest integer. Figure 3.17a shows, for each user and technique, the sum of the scores obtained in all the questions (maximum is 24). Intuitively this metric represents the overall understanding of the function obtained by each user with each technique. Note that every user obtained much better results by using *AudioFunctions* ($AVG = 22.00$, $SD = 2.58$) with respect to AGC ($AVG = 8.29$, $SD = 2.98$). *AudioFunctions* also proved to be more effective also compared with the tactile drawings ($AVG = 17.14$, $SD = 4.63$) that all the users were acquainted with. Indeed, every user, except user 7, obtained better results with *AudioFunctions* than with tactile drawing and for most of the users the results with *AudioFunctions* are much better than with tactile drawings. This result can be explained by the fact that *AudioFunctions* not only offers a proprioceptive interaction mode that is very similar to the tactile drawings (i.e., the Two-dimensional sonification) and therefore easily accessed by users acquainted with the tactile paper, but it enhances the interaction with audio and vocal cues about interesting attributes of the function such as maxima/minima and intersections with axes. Additionally, the other interaction modes allow the user to grasp the general shape of the function and its global properties (e.g., convexity, type). The thorough exploration of the three different interaction modes is also the reason why *AudioFunctions* was consistently explored more than the other two tools.

Figure 3.17b compares the total time required by each user to answer the 8 questions by using each technique. Results show that, by using AGC, users provided answers more quickly (about 2 minutes on average) than with tactile drawings (about 5 minutes on average) and *AudioFunctions* (about 9 minutes on average).

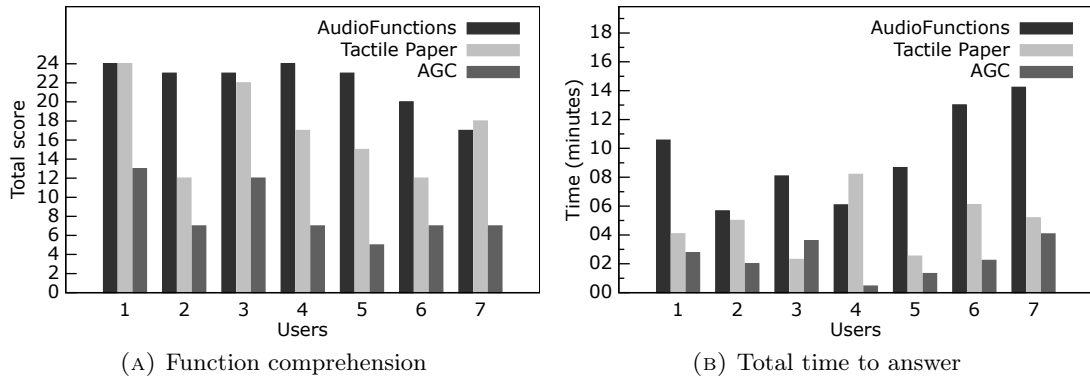


FIGURE 3.17: Results of the experimental testing

Statistical significance of the obtained results was computed with one-way analysis of variance (ANOVA). Based on the result of the test ($F(2, 18) = 27.443$, $p < 0.001$), we conclude that the difference in score between the three considered tools that was detected during the tests is statistically significant. The one-way ANOVA [88] results are shown in Table 3.5.

Source	SS	df	MS	F	p
Between:	676.448	2	338.224	27.443	< 0.001
Within:	221.842	18	12.325		
Total:	898.290	20			

TABLE 3.5: ANOVA results table

A follow up Tukey's HSD test [89] (See Table 3.6) confirms the significance in case of all pairs of considered tools.

	Tactile Drawings	AGC
<i>AudioFunctions</i>	$p < 0.05$	$p < 0.01$
Tactile Drawings		$p < 0.01$

TABLE 3.6: Tukey's HSD post-hoc test results

Chapter 4

Urban navigation for visually impaired users

The cognitive mapping of abstract spaces considered in the previous chapter revolves around the accessibility issues that arise in exploring an already mapped multidimensional space without a visual representation. In this aspect, the previously discussed issues present similarities to the macronavigation [63] spatial exploration (See Section 2).

Instead, the zebra crossing detection solution presented in the following, involves both Micronavigation and Macronavigation aspects. It assists a person with visual impairments in autonomous exploration and short range navigating of an unknown three dimensional space in absence of a precise localization through micronavigation techniques. At the same time the coarser satellite detection and gps-based macronavigation is used for long distance routing.

As a future work, the interaction techniques presented in the previous chapter will be integrated to the proposed solution to be used for the presentation of the spatial information during the macronavigation aspects of the navigation.

Section 4.1 defines the zebra crossing recognition problem, while Section 4.2 outlines the architecture of the proposed solution, called *ZebraCrossing*. Section 4.3 describes the computer vision based zebra crossing detection algorithm. Section 4.4 focuses on the sensor fusion approach to the zebra crossing detection problem. The interaction paradigm is described in Section 4.5 and satellite imagery based zebra crossing detection is explored in Section 4.6.

4.1 Problem definition

Section 4.1.1 defines in detail the zebra crossing pattern that the solution will detect. Section 4.1.2 deals with the issue of the safety for people with visual impairments in urban environment and Section 4.1.3 defines the problem of the hardware platform choice for the detection.

4.1.1 The zebra crossing pattern

Different horizontal traffic signs are used across the world to define pedestrian crossings. One of the most widely adopted pedestrian crossing patterns is the “zebra crossing” pattern. The definition of zebra crossings adopted in this work is extracted from the Italian traffic regulations [90], however the solution can be easily adapted to most definitions used worldwide.

According to Italian traffic laws, the zebra crossing is a horizontal traffic sign, positioned on the ground plane, consisting in a set of white stripes painted on a darker background (see Figure 4.1a). In the following, it is convenient to define as “stripes” also the dark gaps between two consecutive white stripes. The dark stripes are therefore of the same color of the underlying road while the light stripes may be white or, in case of road works, yellow. Each stripe is a uniformly colored rectangle or less frequently, in case of diagonal crossings, parallelogram, having a width of at least 250cm and a height of exactly 50cm. Each crosswalk is itself a rectangle or a parallelogram composed by at least four consecutive stripes with alternating color.

In United States¹, different pedestrian crossing markings are available. The transverse marking, also frequent in the United Kingdom (see Figure 4.1d), is commonly used to mark the pedestrian crossings. It consists in two white lines, perpendicular to the road direction and thick between 6in (15cm) and 24in (60cm). The distance between the two lines is at least 6ft (180cm). The zebra crossing pattern (officially designated as “continental crossing”) is used frequently when the visibility of the crossing is paramount for the pedestrians’ safety, for example near schools and hospitals. Indeed, the distance at which this type of marking is detected in different illumination conditions is consistently higher than for the other types of markings common in the US [91]. Differently from the Italian definition, the crossing has to be at least 6ft (180cm) wide and stripes’ thickness can range from 15cm to 60cm. There is no enforcement on the thickness of the gaps between the stripes and the lines can be either longitudinal (parallel to the road) or diagonal.

Given the similarity between Italian zebra crossings (Figure 4.1b) and the US version (see Figure 4.1c), it is possible to reconfigure the proposed solution to recognize the US zebra

¹See the Manual on Uniform Traffic Control Devices mutcd.fhwa.dot.gov

crossings with very limited effort, by setting and re-tuning the detection parameters. Clearly, while the solution proposed in this chapter does not directly apply to other types of pedestrian crossings, like the transverse crossings (see Figure 4.1d). However, the adopted methodology and the proposed technique can indeed be used to design similar solutions for other pedestrian crossings or other kinds of geometrically well known horizontal traffic signs.

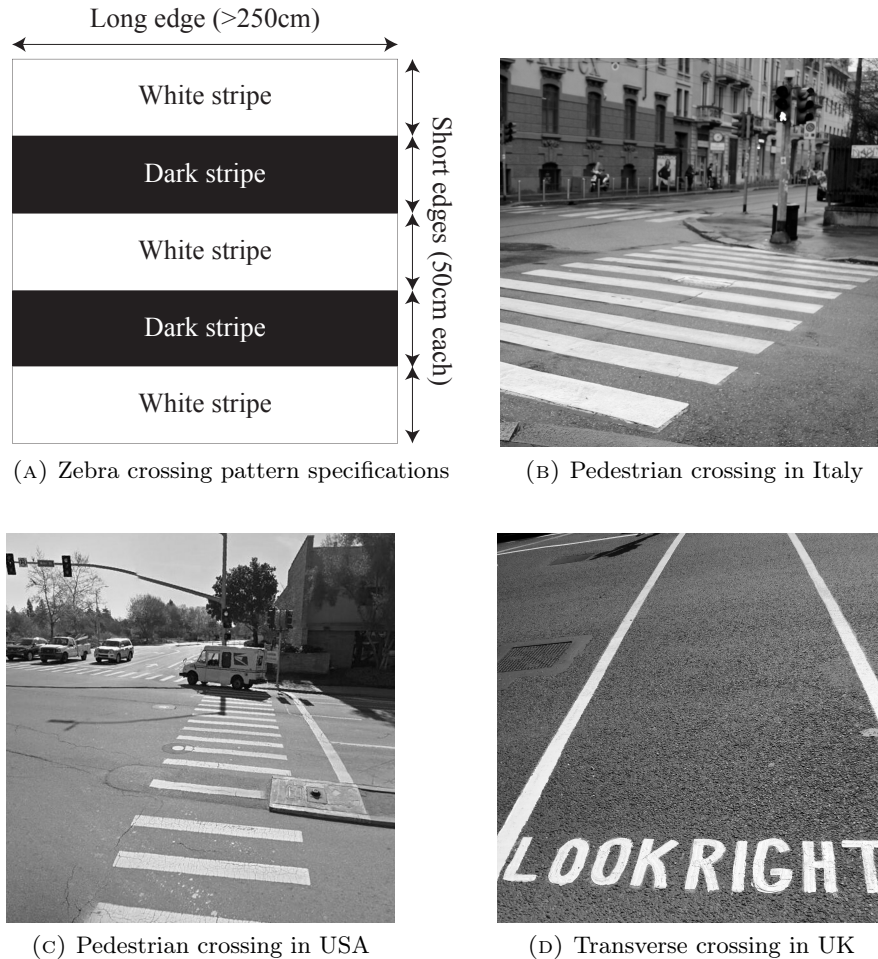


FIGURE 4.1: Zebra crossing definition and examples

4.1.2 Safe navigation in urban environments

Experimental evaluations show that some blind people do not even try to find a zebra crossing, while, for those who try, 6% of attempts to cross are judged dangerous (in contrast, no attempt by sighted person is dangerous). Indeed orientation in unfamiliar environment is a dangerous task for an unassisted blind user [24, 25].

For designing efficient and safe assistive mobility tools for people with visual impairments we consider the following:

- During mobility, a blind user often relies on auditive stimuli in order to avoid perils.

- Often a visually impaired person uses a white cane or has a guide dog.
- The assistive tool must not endanger the user.

Five observations stem from the previous notions:

First, the effectiveness of audio-based interaction is limited in noisy environments, and even more so since the visually impaired user's hearing sense is crucial for danger avoidance, obstacles detection and for understanding the surroundings during movement. Diverting the user's attention from this task is potentially hazardous and in particular overwhelming the user by too much information should be avoided. On the other hand, purely haptic interaction techniques, such as touch input coupled with vibration output have a limited output bandwidth. Thus, the amount of data that needs to be conveyed to the user has to be kept low and the output should use available senses but be such to avoid distracting the user from surroundings.

Second, two-handed interaction methods have to be avoided since, most likely, one of the user's hand is likely occupied either by holding the leash of a guide dog or by maneuvering the white cane, thus only one hand of the user is available for interacting. This consideration involves carefully planning the interaction technique used for input. Specifically no complex hand gestures can be used and, most probably, the thumb is the only finger that can be used for the input.

Third, no erroneous information can be given to the user. Particular attention is required with the kind of information that can expose the user to hazards. For example, telling the user to cross the street in a dangerous position can lead the user to perilous situations. Therefore it is fundamental to be sure of what needs to be communicated to the user and to be able to relay that information effectively

And fourth, at the same time the information about the surroundings must be delivered to the user in a timely manner. That is, the assistive solution has to be able to notify the user immediately of variations in the surroundings. For example, the fact that a pedestrian traffic light is informing the user to stop has to be detected and conveyed immediately. A difference of just a few second can prove to be hazardous.

4.1.3 Smartphone driven computer vision

It's worth noting that, given that the considered issue is encountered on the move, the target platforms for tackling them have to be mobile devices. More specifically, off-the-shelf smartphones have been selected as the hardware of choice for the development of *ZebraCrossing*. There are several advantages with this approach. First, the user can have a single device running both mass market applications and assistive software. This is clearly cheaper than having several hardware devices, it is more convenient while on the

move, it simplifies software acquisition through online stores and reduces the user's effort required to get acquainted with each application, as all applications adopt the standard system-wide interaction paradigm. The second advantage is that mass market devices have powerful hardware including multicore CPUs and GPUs, large main memory and several sensors, like accelerometer, gyroscope, camera, etc. Operating systems are reliable, highly optimized and regularly maintained. Also, developers can benefit from a number of available libraries (either within the OS or from third parties). Finally, from the business development point of view, it is much cheaper and less risky to develop and distribute software for mobile devices rather than produce ad-hoc hardware, also considering that the relative small number of potential users does not activate the economies of scales that drastically reduce production costs.

For the detection of zebra crossings, data sources available on off-the-shelf smartphones are leveraged. Specifically, the considered sources are video camera, accelerometer and gyroscope. The former captures image frames that can then be analyzed with computer vision techniques in order to detect zebra crossings, if present. Accelerometer and gyroscope, instead, can be used to extract the orientation of the device with respect to the ground plane and the detected crossings.

It is worth noting that, since it's difficult for a person with visual impairments to focus the video camera of the device and take a picture, it's preferable to capture the images through camera's video stream. This approach will be used in the following.

4.2 System architecture

ZebraCrossing navigation system deals with the issue of crossing a pedestrian crossing in a safe and efficient way. It contains both Macronavigation and Micronavigation aspects and it is divided into three main modules, as depicted in Figure 4.2.

The *Recognizer* module, deals with the computer vision-based detection of the zebra crossing on the mobile device. This Micronavigation solution has been tackled with two different approaches. Section 4.3 shows the *VideoRecognizer* version that relies on computer vision based detection on images captured by the camera of the mobile device. Instead, Section 4.4 explores the *FusionRecognizer* version of the module that bases the detection on sensor fusion-based approach that results in ground plane reconstruction through the usage of accelerometer and gyroscope of the device.

The *Navigator* module, described in Section 4.5 is in charge of interacting with the user by acquiring inputs through the touchscreen and giving audio feedback.

The *Satellite* module considers an orthogonal Macronavigation issue. That is capability to guide a user towards a crossing that is too distant to be detected with the proposed Micronavigation techniques. This module is defined in Section 4.6.

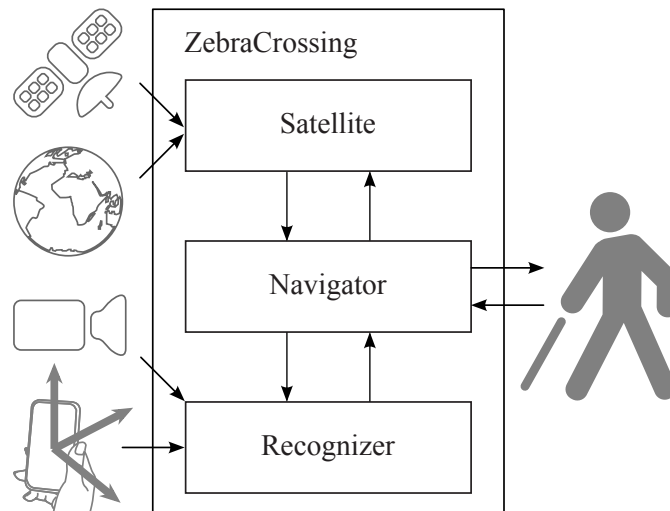


FIGURE 4.2: System architecture

4.3 Computer vision based zebra crossing detection

VideoRecognizer module leverages computer vision techniques to detect zebra pedestrian crossings from video camera input. The detection algorithm takes in input a gray-scale image from the video camera stream 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.

The first step of the *VideoRecognizer* detection is the horizon computation, common to the *FusionRecognizer* technique presented in Section 4.4. It will be presented in Appendix A. The algorithm can also be executed without this step. In this case, however, the quality of the detection degrades with the camera rotation and execution time optimizations that leverage the knowledge of the horizon cannot be applied. The next step is the feature extraction (Section 4.3.1) using the LSD algorithm detailed in Appendix B.2. Section 4.3.2 deals with the line segment analysis while the stripe analysis is discussed in Section 4.3.3. Finally, the relative position is computed in Section 4.3.4. Section 4.3.5 shows the evaluation of the *VideoRecognizer* module.

4.3.1 Feature extraction

In the considered object recognition problem, the features used for the detection 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 [92] is used. The implemented algorithm, described in details and evaluated in Appendix B, differs with respect to the one proposed in [92]: it ignores the points above the horizon and the line segments whose angle with respect to the horizon is bigger than $\pi/6$ (only the stripes that are roughly perpendicular to the observer are considered). Figure 4.3 shows the result of the original LSD algorithm compared with the result of the modified version.



FIGURE 4.3: Extraction of line segments (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 the line segments that are approximately parallel and very close to each other are merged into a single segment if they have the same gradient orientation. Otherwise both line segments are discarded. Figure 4.4 shows the result of the application of this sub-step.

2.c) In the last sub-step the line-segments whose length is below a threshold value are pruned. Indeed, since the stripes are formed by long line segments, dropping short segments does not discard any useful feature.

4.3.2 Line segment 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

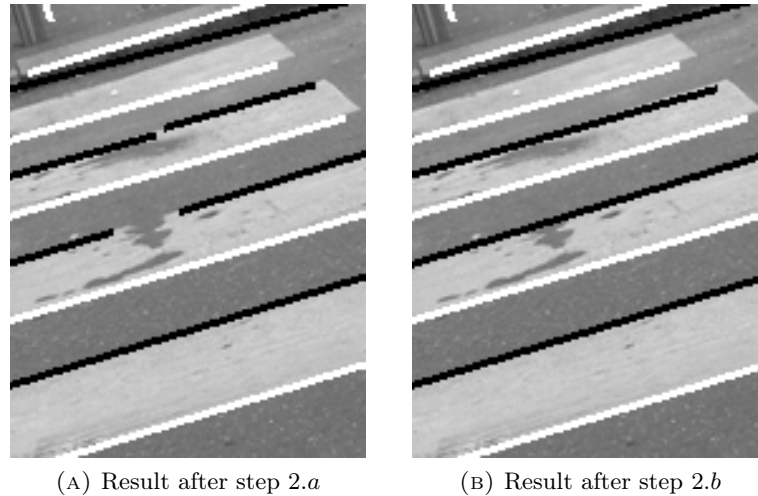


FIGURE 4.4: Line segments merging

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.

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 4.5a 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. The latter property is more involved during the detection. 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, the 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 4.5b 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

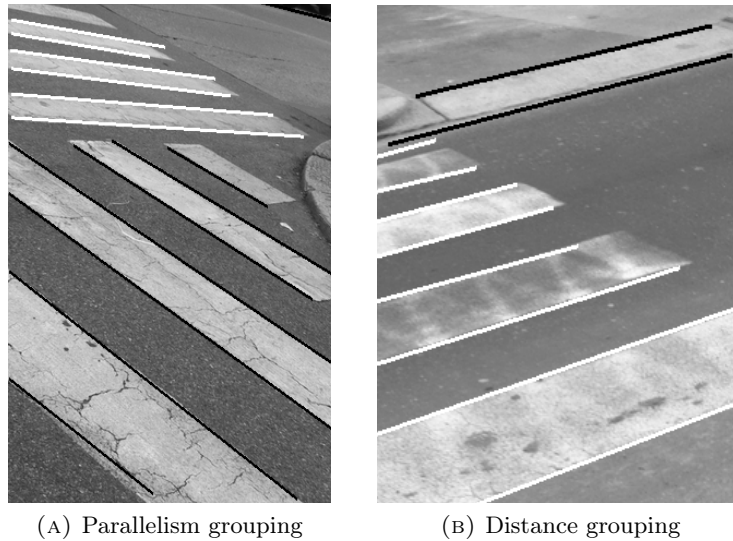


FIGURE 4.5: Grouping of line segments

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.

4.3.3 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. The validation proceeds as follows. First, considering some sample points, the average (*avg*), minimum (*min*) and maximum (*max*) color intensity of the half-plane below the horizon (i.e., the background) is computed. Then, again considering some sample points, the average color intensity inside a stripe is determined. 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 . This approach has been previously proposed for the zebra crossing validation by Uddin and Shioyama [79]. Considering the points A, B, C, D depicted in Figure 4.6, 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 solution 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.

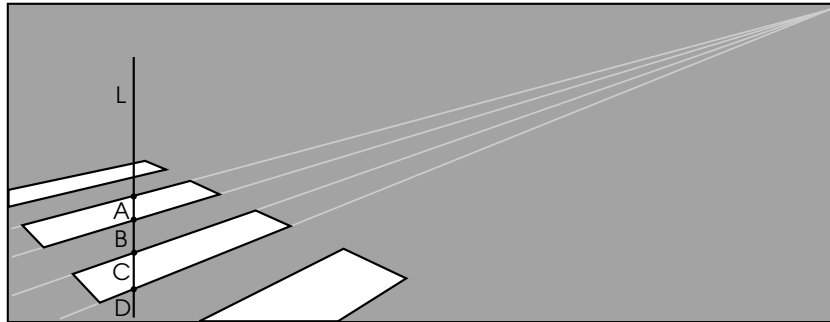


FIGURE 4.6: Cross ratio

4.3.4 Relative position computation

If a crosswalk is identified, the last step of algorithm computes its relative position of the detected crossing with respect to the observer. In absence of ground plane rectification it is unfeasible to compute the precise position. However, if the user is instructed to keep the device always roughly in the same position, it is possible to estimate whether to adjust the user's position or not based on where the crossing is found inside the captured image.

The position is computed as a set of three values: the *rotation*, the *lateral shift*, and the *distance*.

The *rotation* is derived from the slope of the stripe closest to the observer. A threshold is computed in such a way that, if the stripes have an inclination with respect to the user of 30 deg or more, the absolute value of the slope is greater than the threshold. In that case, the *rotation* is set to "left" or "right" if the slope is positive or negative respectively,

and the user is instructed to rotate accordingly. In other cases the *rotation* is set to “aligned”

The *lateral shift* signals if a movement is required from the user to cover the lateral distance from the center of the visible part of the crosswalk. A value is computed as the distance, in pixels and along the x -axis (or the horizon, when available), between the center of the line segments closest to the observer and the center of the picture. As in the previous case, a threshold is set such that lateral distances greater than $1m$ result in *lateral shift* being set to “left” or “right” as needed. In other cases *lateral shift* is set to “centered”

Finally, the *distance* between the user and the crosswalk is computed. First, the distance, in pixel, between the horizon and the line segment closest to the user is found. A threshold is set such that frontal distances above $1m$ result in *distance* being set to “far”, otherwise it is set to “near”

4.3.5 Evaluation

The computer-driven evaluation was performed by repeatedly running the *VideoRecognizer* 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 in different light and weather conditions, including the natural light of a bright sunny day, a cloudy 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 specifically developed that inserts the values of the accelerometers within the picture file as EXIF data. This makes it possible to use the accelerometer data during the recognition phase of the tests conducted on a PC platform.

The quantitative experiments highlighted three aspects of the developed algorithm:

1. 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. Given the time efficiency of the algorithm, it can be embedded in applications running on off-the-shelf smartphones.

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

To evaluate the accuracy of the algorithm, precision and recall metrics are used. 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 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, a variation of the *VideoRecognizer* library that does not compute the horizon and that does not rely on accelerometer information was evaluated.

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 4.3.3) 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 4.3.2) to predict the next stripe’s maximum allowed width given the width of the last recognized stripe.

Figure 4.7 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 *VideoRecognizer* is to have no false positive, 0.075 was used 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.

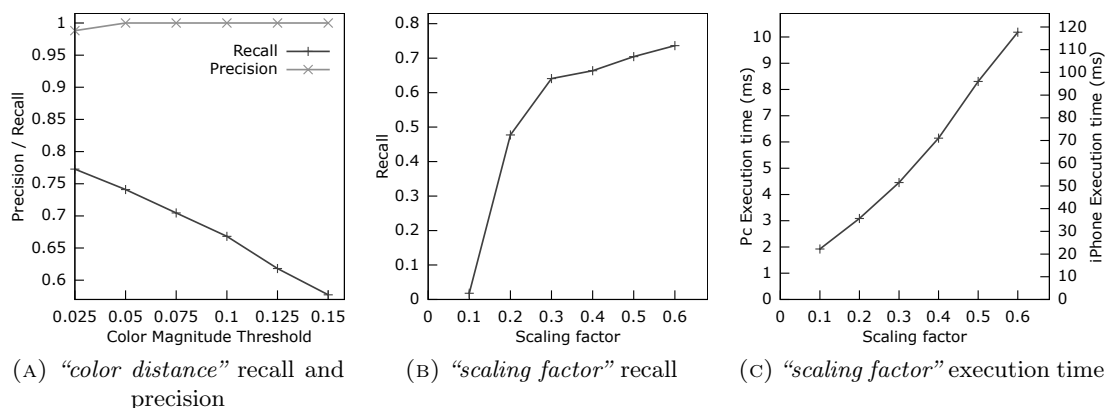


FIGURE 4.7: Impact of the parameters “*color distance*” and “*scaling factor*”

In the experiments it emerges that the size of the image is the parameter that most significantly affects the computation time. Figures 4.7c and 4.7b 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 0.6 and 0.1, the recall is not significantly affected for values of “*scaling factor*” between 0.6 and 0.3, while it rapidly decreases for smaller values. Vice versa, for larger images, i.e., “*scaling factor*” larger than 0.6 (not represented in the figures), the recall does not significantly improve, while the computation time still grows almost linearly. Figure 4.7c 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 0.6 and 0.3. In this implementation of *Navigator*, the objective was to process about ten images per second and hence the choice was to use 0.5 as the default value for “*scaling factor*”.

One set of experiments conducted was devoted to measure the improvement introduced in the recognition process by use of the accelerometers. To achieve this, the dataset of images representing zebra crossings was divided into those taken in an almost portrait position and the others (called “sloped” in the following). Our results (see Figure 4.8) 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.

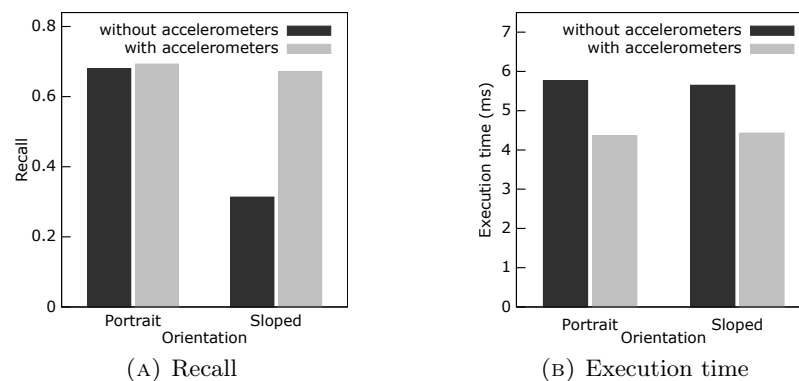


FIGURE 4.8: Impact of accelerometer data on recall and execution time

4.4 Sensor fusion based zebra crossing detection

Differently from the approach explored in the previous section, the *FusionRecognizer* module, described in the following, heavily leverages the spatial orientation data for enhancing the detection procedure, mainly through the reconstruction of the ground plane on the camera input.

The input of the algorithm is the grayscale image from the device's video camera stream and the gravity acceleration data from the accelerometers. Note that accelerometers actually measure the combined acceleration due to user's movement and gravity. Since, for the purpose of this work, only the gravity component of the acceleration is required, sensor fusion techniques that leverage gyroscope and accelerometer data are used to compute the gravity acceleration vector by subtracting the component of the acceleration caused by the user movements from the accelerometer measurements. At the moment, the sensor fusion algorithm adopted in this work is the one adopted by the iOS 8.0 API. In the considered API, the gravity component of the acceleration, as well as the device orientation in space, are accessible through the *CMDeviceMotion* system class ². This approach is possible on modern iOS and Android devices (iOS 4.0 and up, android 2.3 and up). As a future work, the Kalman filter sensor fusion algorithm [93] will be implemented, both for enabling the use of the detection algorithm on other mobile platforms and for increasing the precision of the gravity vector computation.

The gravity acceleration with respect to the device is expressed as a three dimensional unit vector $a = \langle a_x, a_y, a_z \rangle$. The gravity acceleration components a_x, a_y and a_z are measured in $g = 9.80665m/s^2$ and take values in $[-1, 1]$. Since the gravity force is normal to the ground plane, they represent, respectively, the portion of the gravity that is applied on the device x, y and z axes (see Figure 4.9a). These values correspond to the sine of the inclination with respect to the ground plane of the corresponding axes of the device.

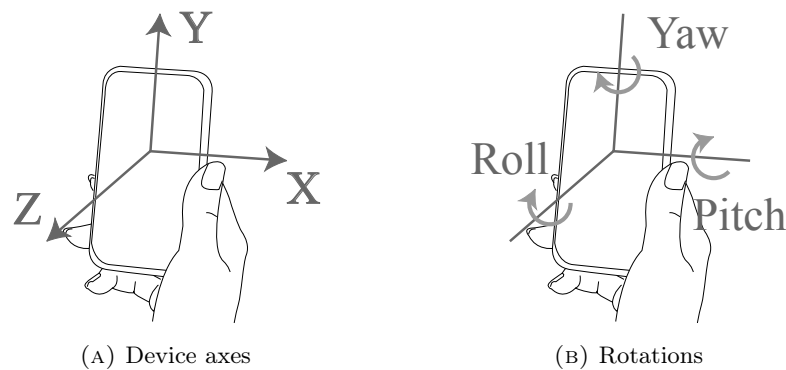


FIGURE 4.9: Device orientation.

²developer.apple.com/library/ios/documentation/coremotion/reference/CMDeviceMotion_Class

The output of the algorithm is the most suitable detected zebra crossing, if any. It is characterized by a list of stripes, each defined by its top and bottom line segments and its color (i.e., black or white). The position of each line segment is represented both in the source image (e.g., Figure 4.10a) and on the rectified ground plane (e.g., Figure 4.10b). The result also includes four compact and easy-to-use distance measurements (see Figure 4.10c): frontal distance, rotation angle, lateral distance from the left and right borders of the crossing.

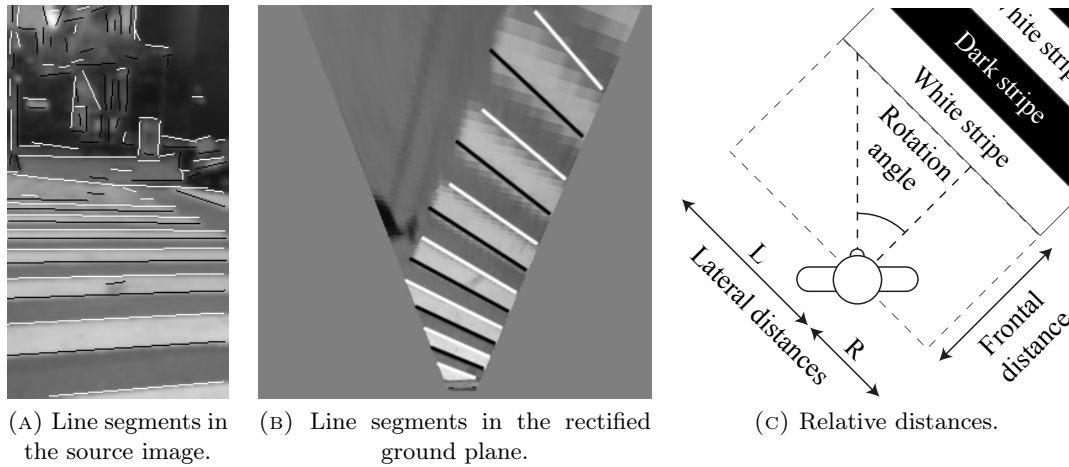


FIGURE 4.10: *Recognizer* output

It is internally divided into 7 steps (See Figure 4.11), the results of which are shown in Figure 4.12.

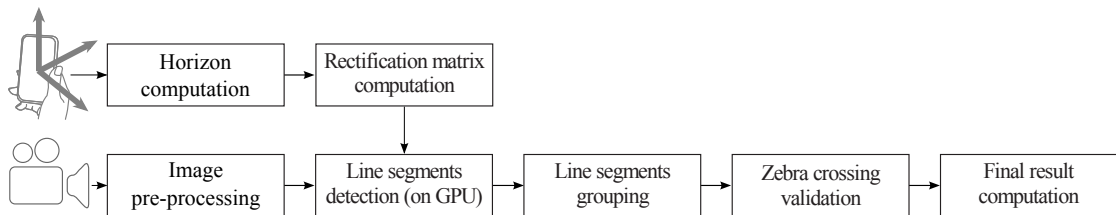


FIGURE 4.11: *Recognizer* flowchart

The image preprocessing step (Section 4.4.1) prepares the image for the segment detection. The horizon computation step is common to the technique presented in Section 4.3 and thus, will be presented in Appendix A. The image reconstruction technique that computes the rectification matrix is described in Section 4.4.2 while the segment detection step, explored in Appendix B.3, is outlined in Section 4.4.3. Section 4.4.4 and Section 4.4.5 present the line segments grouping and stripes validation respectively. Finally, Section 4.4.5.1 describe the final result computation.

4.4.1 Image Pre-Processing

As observed in Section 4.1.1, zebra crossings can be painted with different colors and therefore only the light intensity components of the image is used for the detection.

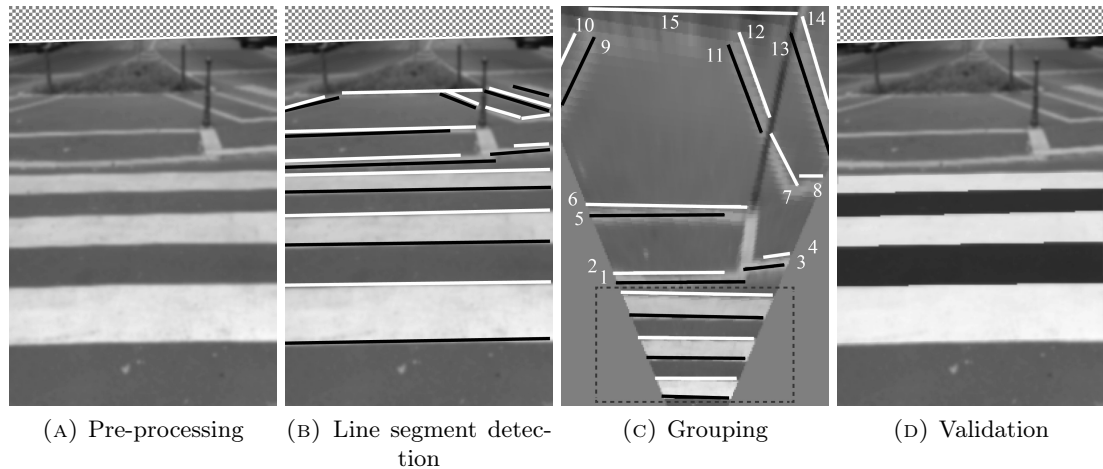


FIGURE 4.12: Main steps of the sensor fusion based zebra crossing detection

Hence, the images are acquired in grayscale. Clearly, the use of single-channel images also helps improving the computation performance and reduces the memory footprint.

The acquired images contain many small details and uninteresting details, such as cracks, paint imperfections, leaves and dirt. These imperfections may actually impair detection, therefore resampling and blurring is used to filter them out. The first method rescales the image until small details become undetectable while the second technique is applied along with the segment detection algorithm (see Appendix B.2). Also, it reduces the image size and thus diminishes the execution time of following per-pixel operations. However, the size still has to be sufficient for a correct detection. As highlighted in the experiments (see Section 4.4.6), the best results of the recognition, without significant loss of information in terms of detected crossings, can be obtained with an image resolution that is much smaller than the maximum camera resolution of modern smartphones.

The images in the test-sets were recorded at the resolution of 1080×1920 and resized, with a linear interpolation filter, before running each test so that *Recognizer* can be run with images at different resolutions.

4.4.2 Ground plane reconstruction

The planar reconstruction is useful during the zebra crossing recognition since it allows to validate the crossings by examining the undistorted geometrical features, obtained by rectifying the line segments obtained from the previous step. This differs from the technique presented in Section 4.3 where geometrical reasoning is applied directly on the detected line segments.

Planar rectification is a homography, represented by a 3×3 rectification matrix, that removes the projective distortions from the image of a planar surface and returns a view

of the same plane in which the camera's axis is perpendicular to the plane. For the ground plane, the rectified image is a view from directly above it, as seen in Figure 4.13b.

Once the rectification matrix is known, it can be selectively applied to some elements (i.e., line segment end points) instead of the whole image, thus reducing the execution time. Also, it is possible to switch back from the rectified image to the original one by applying the inverse of the rectification matrix to the desired elements.

In a previous work ([34]) the rectification matrix was computed from the accelerometer data as the product of two matrices: the affine rectification matrix, obtained by using the technique proposed by Liebowitz and Zisserman ([94]), and the metric rectification matrix, computed through a custom technique.

Lefler et al. [95] propose a technique for computing the rectification matrix from three symmetric vanishing points. We experimentally observed that this technique yields better performance (mainly in terms of recall) with respect to the one previously used. As seen in Figure 4.13c, symmetric vanishing points are 3 points v_x , v_y and v_z that create a right angle with the camera point C . This technique can be applied easily since it leverages already computed information: the gravity vector a (See Section 4.4) and the image horizon hl (See Appendix A).

A previous attempt to use the edges of the zebra crossing for the computation of the vanishing points yielded less precise results (due to the imprecision in the computation of the vanishing point derived from the short edges of the stripes). This approach will not be used in the following. However, as per reviewers' suggestions, as a future work the rectification computed by leveraging both the long edges of the detected stripes and the accelerometer data will be evaluated.

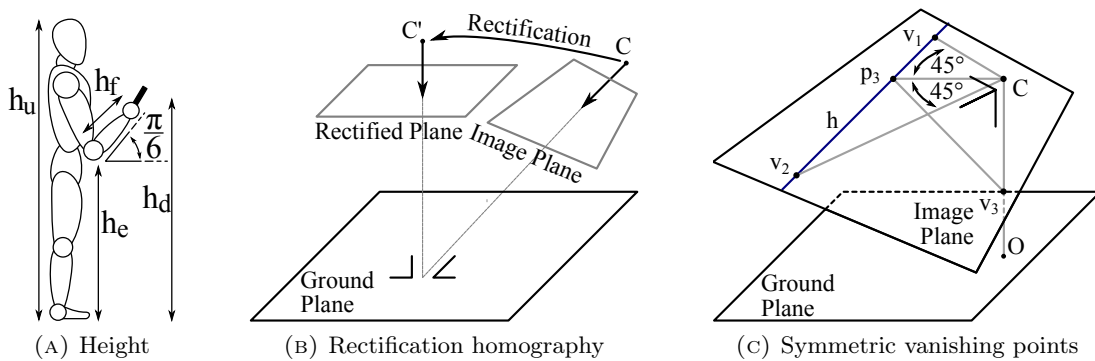


FIGURE 4.13: Height extraction and rectification matrix computation.

The three symmetric points are defined as follows. Point C is the camera, O is the projection of C on the ground plane and v_3 is the point where CO intercepts the image plane. To specify v_1 and v_2 let first define p_3 as the projection of v_3 on the vanishing line. Then, v_1 and v_2 are the points laying on the vanishing line such that angles $\widehat{v_1 C p_3}$ and $\widehat{v_2 C p_3}$ have a span of 45° . The position of the three points in the image can

straightforwardly be obtained. For example, the distance $\overline{Pv_3}$ between v_3 and the image principal point P can be computed with the following proportion:

$$\sin(\widehat{PCv_3}) : \overline{Pv_3} = \sin(\beta_y) : i_h$$

where $\widehat{PCv_3}$ is the device pitch, β_y is the vertical field of view (can be derived from the camera intrinsic parameters) and i_h is the image height. The computation is analogous for v_1 and v_2 .

The application of the rectification matrix to the image yields a “rectified plane” in which the distances are proportional to those on the ground plane. More specifically, the distance between any two points on the ground plane is equal to the distance of the corresponding points on the rectified plane multiplied by a *zoom factor*.

The zoom factor is defined as the ratio $d_r(A_r, B_r)/d_g(A_g, B_g)$ where $d_r(A_r, B_r)$ is the distance of two arbitrary points in the rectified plane and $d_g(A_g, B_g)$ is the distance of the corresponding points on the ground plane.

To compute the zoom factor *Recognizer* takes into account two points A_i and B_i on the image plane and their projections A_r, B_r, A_g and B_g on the rectified plane and ground plane, respectively. See Figure 4.14.

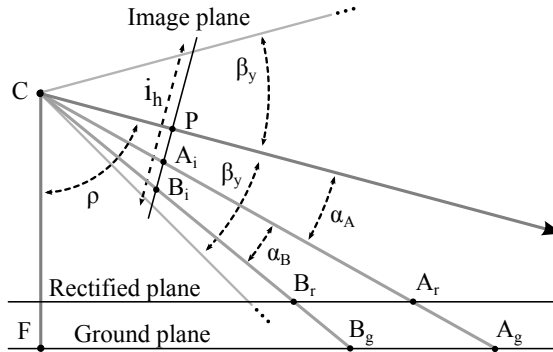


FIGURE 4.14: Zoom factor computation

A_i and B_i are arbitrary points below the horizon and that lie on line vl that is perpendicular to the horizon and that passes through the image principal point P . The distance $\overline{A_r B_r}$ is computed by applying the rectification to A_i and B_i and computing the euclidean distance on the resulting points A_r and B_r . The distance $\overline{A_g B_g}$ on the ground plane is computed as $\overline{A_g B_g} = \overline{FA_g} - \overline{FB_g}$.

To compute $\overline{FA_g}$ consider right triangle CFA_g . Angle $\widehat{FCA_g}$ is equal to the device pitch ρ minus α_A . To compute angle α_A , consider the following proportion:

$$\overline{AP} : i_h = 2\sin(\beta_y) : \sin(\alpha_A)$$

where i_h is the image height and β_y is camera half vertical field of view. Consequently $\alpha_A = \text{asin}(\overline{AP} \cdot 2\sin(\beta_y)/i_h)$. By knowing angle $\widehat{FCA_g}$ and $h_d = \overline{CF}$ (the height of the

device), $\overline{FA_g}$ can be easily computed as:

$$\overline{FA_g} = h_d \cdot \tan(\rho - \text{asin}(\overline{AP} \cdot 2\sin(\beta_y)/i_h))$$

Analogously, $\overline{FB_g}$ is computed as:

$$\overline{FB_g} = h_d \cdot \tan(\rho - \text{asin}(\overline{BP} \cdot 2\sin(\beta_y)/i_h))$$

As observed above, the computation of the zoom factor relies on the height h_d of the device from the ground. To estimate this value, *Recognizer* assumes that the user is holding the device in a position like the one depicted in Figure 4.13a in which the elbow is close to the hip and the forearm has an inclination of about $\pi/6$ with respect to the ground plane. By considering the proportions of the human body [96], the device height can be derived from the user's height h_u (either estimated or asked to the user). Indeed, on average the height at elbow is $0.615 \cdot h_u$ and the forearm length is $0.205 \cdot h_u$. Consequently, the device height from the ground is estimated as:

$$h_d = 0.615 \cdot h_u + \sin(\pi/6) \cdot 0.205 \cdot h_u$$

Clearly the above computation is subject to some approximation. However, the error is practically not significant. For example, considering a 175cm tall person the technique estimates that the device is held at 125cm from the ground. Even if the device is actually kept at the height of the shoulders (a situation never observed during the experiments with the users), at the height of 142cm, a zebra crossing at a distance of 2m is computed as being 2.33m from the user. This does not significantly affect, for example, the number of steps required to reach the start of the zebra crossing.

4.4.3 Line segment detection

The line segments detection relies on a modified version of the original EDLines algorithm [97] described in Appendix B.3. The input is composed by the input image, the horizon line and the rectification matrix. The output is a set of detected segments in the rectified coordinate system. There are four main differences with respect to the original algorithm.

First, the proposed technique ignores the portion of the image above the horizon (See Figure 4.12a) since no zebra crossings will ever be found there. This approach significantly reduces the computation time for two different reasons: it speeds up the line segments detection process itself and it reduces the number of detected segments, hence reducing the computation time of successive processing steps. This solution also helps improving the recognition accuracy as it prevents false positives (i.e., a false crossing recognized above the horizon).

The second difference with respect to the original EDLines algorithm is that the implemented solution computes additional information about the detected line segments. First, as shown in Appendix B.3, in addition to gradient direction, i.e., an angle in $[0, \pi)$, this solution also computes the gradient orientation of the detected segments, so, in practice, the angle of the gradient is computed in $[0, 2\pi)$. Hence, it is possible to distinguish between a light-to-dark gradient and a dark-to-light one. This is a useful information in the following steps since the direction of the gradient can differentiate between segments on top and on the bottom of each stripe. The line segment's orientation is represented by the order of its end points. Thus, two line segments $s = AB$ and $s' = BA$ have the same direction but opposite orientation. The second additional information computed by this version of EDLines is whether each end point of each line segment lies on the image boundary. This is useful, in the following computation, to distinguish between stripes that terminate in the end point position and those that, instead, can potentially continue but are not visible in the image.

The third difference is that the presented technique also merges close segments. Two segments having slope distance and spatial distance both lower than specified thresholds are merged. This is useful, for example, when two or more portions of a line segment have been recognized as different line segments due to minor imperfections in the image, noise, flawed coloration of the stripes or objects between the observer and stripes. Figure 4.15 shows an example. The line segment s resulting from the merging of two line segments s_1 and s_2 is computed as follows: first, the lines l_1 and l_2 on which the two line segments lay are calculated. Then, a new line l (equation in general form: $ax + by + c = 0$) is computed with parameters a , b and c being weighted averages (based on the two segments' lengths) of the corresponding parameters of lines l_1 and l_2 . Finally, the segment s is computed as the union of the two line segments' projections on l .

In Section 4.3.1 this merging operation was computed using line segments in their representation on the image and hence were subject to projection distortion. Vice versa, in the current solution, line segments are rectified before being merged.

The fourth difference, described in Appendix B.3, is that, during line segment computation, The orthogonal regression is used instead of least squares line fitting for the purpose of determining the equation of the line on which each line segment lays. Orthogonal regression computes the orthogonal distance between each point and the candidate line, differently from the line fitting algorithm that computes the vertical distance. Orthogonal regression is needed in this case since vertical line segments also need to be computed.

As a final step, after lines segments merging, the segments that are too short to possibly represent a stripe edge are pruned. Figure 4.12b shows an example of application of the customized version of EDLines.

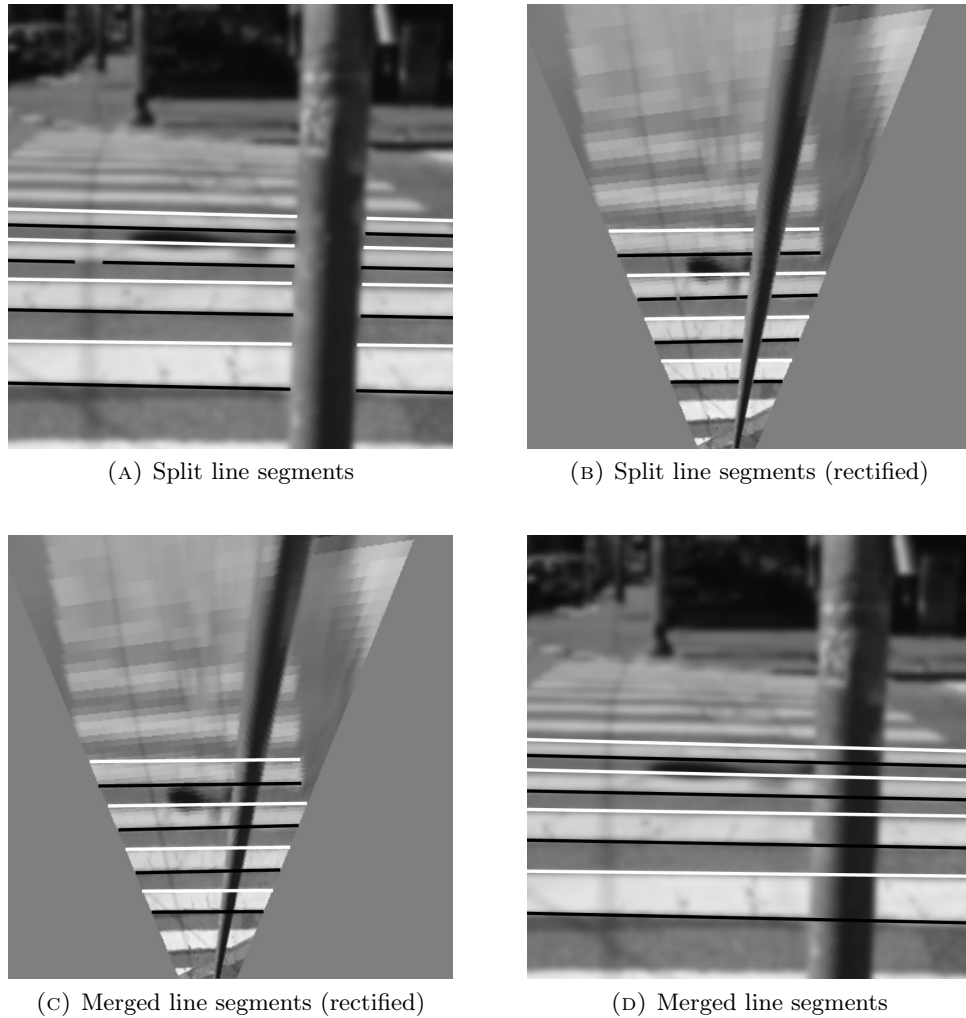


FIGURE 4.15: Example of line segments merging

While the custom implementation of EDLines has been highly optimized, it is still the most expensive operation of the detection procedure and it takes about 45% of the entire computation time. The reason is that three operations required by EDLines have a time complexity linear in the number of pixels in the image. These three operations consist in the computation of: the gradient magnitude, the gradient direction and the anchors. Since the aim of these three operations is to extract the so called “anchors”, they are referred to as “anchors extraction”.

To reduce the computation time of “anchors extraction”, it has been implemented through two fragment shaders, so that the computation can be run by the GPU highly parallel architecture. Indeed, while the general purpose GPU computation frameworks like CUDA and OPENCL are still not available on mobile devices, it is possible to use programmable fragment and vertex shaders that are actually available in mobile GPUs. The core idea behind a fragment shader is that it defines how to compute each pixel of an output image. To achieve a highly parallel computation, each pixel in the output image

must be computed independently from all the others in the sense that it is not possible to use, in the computation of a pixel, the result of the computation of a different one.

In this solution one fragment shader is used to compute gradient magnitude and direction. These two operations can be computed in a single shader as both depend on the input image only. Vice versa, anchors computation depends on the result of the other two operations, hence it is implemented in a separate shader. The result of each operation is stored in a different channel of an RGB image.

Experimental results, run on an iPhone 5s with the methodology presented in Section 4.4.6, show that, on average, anchors extraction is more than 4 times faster when run on GPU. In absolute terms, the average time required to compute these operations on a single frame is about 8.5ms when computed in CPU and less than 2ms when computed in GPU.

4.4.4 Line segments grouping

Starting from the lines segments extracted from the image, the solution rectifies them with the procedure described in Section 4.4.3 and then groups them forming candidate crossings. Each candidate crossing is characterized by a set of stripes, that, in turn, are composed by a pair of line segments each. Line segments grouping processes the rectified line segments, so that it is possible, for example, to straightforwardly check geometrical properties (e.g., parallelism) and to compute quantified measurements (e.g., the width of each stripe).

All line segments are first assumed to be part of a single set that is then partitioned according to three criteria: “slope”, “horizontal overlapping” and “vertical distance”. Each grouping criterion is enforced by using an agglomerative hierarchical clustering technique with single linkage. The first criterion (“slope”) is applied to the entire set of line segments (considered as a single set) and results in a set of blocks, each one used as input for the iterative application of the following criteria.

The idea behind the “slope” criterion is that the line segments in the same crossing are mutually parallel. Thus, line segments having roughly the same direction are grouped together.

The “Horizontal overlapping” criterion captures the fact that, in a zebra crossing, there must be an overlap among the projections of stripes long edges on a line parallel to them. This computation should not discard stripes that are partially outside the field of view and, because of that, have a small overlap. It is possible to distinguish this cases because it is known, due to the optimizations to the EDLines algorithm introduced in Section 4.4.2, for each end-point of each line segment, if it lies on the image boundary.

Finally, the “vertical distances” criterion is enforced. This validation guarantees that, in each group, two consecutive line segments must have opposite gradient directions and a distance of about 50cm (this is specific for Italian regulation, see Section 4.1.1). The computation proceeds as follows. First, the distances between each pair of line segments in the same group are computed. Then, the pair that has the distance closest to 50cm is linked. Subsequently remaining pairs with the best distances are iteratively linked until there are no free pairs or the distances are shorter or longer than specified thresholds

Figure 4.12c shows an example of the application of the three criteria above. All the numbered line segments are not grouped with the line segments in the dashed box due to the following reasons: line segments 7, 9, 10, 11, 12, 13 and 14 due to the slope criteria; line segments 3 and 4 due to the horizontal overlapping criteria; line segments 1, 2, 5, 6 and 15 due to the vertical distance criteria. As explained in Section 4.4.5, all numbered line segments are pruned due to the fact that they form groups with too few line segments.

4.4.5 Stripes validation

After line segments grouping each resulting block is validated two criteria: “color consistency” and “number of edges”.

The “Color consistency” criterion checks if each light (or dark) stripe has a color consistently lighter (darker, respectively) than the average color of the candidate crossing. By considering the average crossing’s color, minor imperfections due to shadows are tolerated. If, however, the contrast between the shadowed area and the rest is too high, most likely the smaller area will be discarded by the color consistency check. Clearly the expected color (light or dark) of a stripe is known due to the fact that the gradient of its two edges is defined. The minimum required difference between the stripe color and the crossing’s average color is specified by the “color consistency magnitude threshold” parameter. Thanks to the “color consistency” criterion, structures that are geometrically similar to stripes but without consistent dark/light alternating colors are discarded. An example of application of the “color consistency” criterion is shown in Figure 4.16. After the grouping phase, some line segments are grouped in a single block and hence are marked as a candidate crossing (see Figure 4.16a). However, as can be observed in Figure 4.16b, there is a small difference in the coloration of the identified stripes. This is captured by “color consistency” that correctly discards the crossing.

The second validation criterion, is “number of edges”. It defines that a valid zebra crossing should be composed by a minimum number of edges. In the experimental settings, this value is set to 5, hence guaranteeing that each crossing contains at least two white stripes, as required by Italian regulation. Consequently, blocks that contain a smaller number of

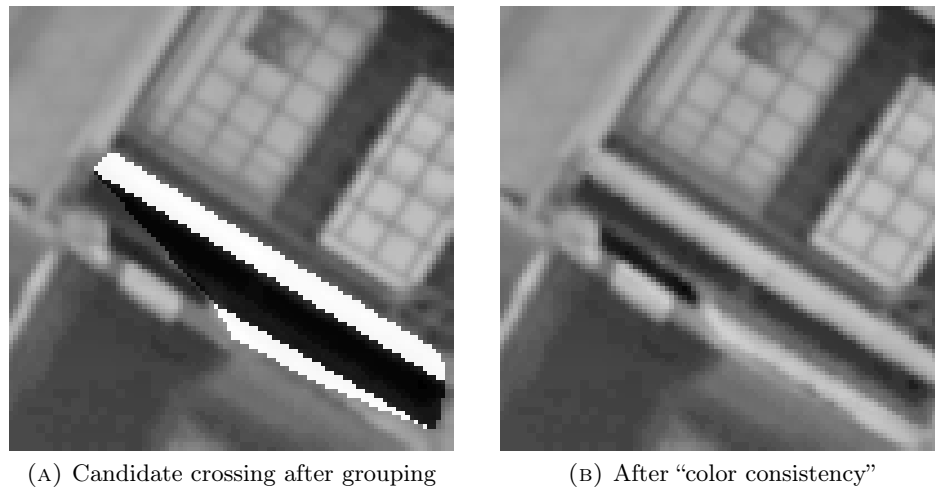


FIGURE 4.16: Example of “color consistency”.

line segments are pruned. A candidate crossing that passed “color consistency” and still has a sufficient number of line segments is returned as “validated crossing”.

In theory, the “number of edges” criterion could be only checked as the last step of the recognition procedure (i.e., after “color consistency”). However, “number of edges” validation is very efficient. For this reason it is enforced after each step of grouping and validation in order to reduce the number of line segments to process, hence improving the overall computation time of solution.

4.4.5.1 Final result computation

In many cases either none or a single validated crossing is returned by the validation phase. However, it is possible that two or more crossings are returned. This happens, for example, when the camera is viewing two distinct crossings or a single crossing is erroneously split in two parts. To decide which is the “most relevant” crossing for the user the following methodology is adopted. The cases in which two or more validated crossings were present in the same image were isolated. It was empirically observed that the most relevant crossing is always the one closest to the user among those having roughly the same direction as the user. Consequently the solution first checks if any detected crossing has an orientation angle within a threshold from the user’s orientation. If favorable crossings are available, then all other crossings are discarded. Among the remaining ones, the closest one to the user is selected as the most relevant.

The four distance measurements used to position the most relevant crossing are shown in Figure 4.10c. “Frontal distance” is defined as the distance between the user and the closest line segment (called *CLS* in the following). “Rotation angle” is the (oriented) angular distance between the user’s heading and the crossing. In the figures the user direction is represented as pointing upwards, so the rotation angle corresponds to the

stripes angle. In theory, since the line segments should be mutually parallel, the angle is the same for all line segments. However, in practice, there can be some approximation and hence the rotation angle is computed as the average angle of all line segments. The third and fourth distance measurements are “lateral distance left” and “lateral distance right”. The definition of the former is given in the following, while the latter is computed in an analogous fashion. “Lateral distance left” intuitively represents the distance between the user and the left border of the crossing measured on *CLS*. More formally, it is the (directed) distance between the left border of *CLS* and the projection of the user’s position on *CLS*.

There is an issue arising in the computation of “lateral distance left” (the same holds for “lateral distance right”). Indeed, it is possible that the edge of the first detected stripe is not entirely contained in the image. In this case the left end-point of *CLS* does not necessarily represent the left border of the closest stripe. Let’s consider two examples. In Figure 4.17a the left end-point of *CLS* (point *B*) actually represents the left end of the stripe (point *A*). Figure 4.17b shows the rectified view. Differently, in Figures 4.17c and 4.17d the first stripe is not fully contained in the image and the left end-point of *CLS* (i.e., point *B'*) is not the left end of the stripe (i.e., point *A'*). In the first case (Figures 4.17a and 4.17b) it is clear that the user is close to the left border and hence she/he should be instructed to stripe right before crossing. In the second case, instead, the same instruction is not given. Indeed, by observing the stripes that are farther from the user, it is possible to infer that the first stripe extends on the left of the user hence, intuitively, it is safe to start crossing in the current position. To capture this intuitive reasoning, The left end-points that are marked as not-being on the image boundary (see Section 4.4.2) are considered. If there are too few of these points, the “lateral distance left” is marked as *not quantifiable*. Vice versa, an orthogonal regression algorithm is used to find the stripe “border” i.e., the line that passes through these points. Then the intersection *A'* of this line with the line where *CLS* lies is computed. The “lateral distance left” is expressed as the length of $\overline{A'C'}$.

4.4.6 Evaluation

Extensive experimental evaluation of the *FusionRecognizer* application was conducted with the aim of tuning the system parameters, debugging the code and evaluating the performances.

The *FusionRecognizer* solution was compared with *VideoRecognizer* ([32], [34]). A direct comparison with other solutions is unfeasible because the implementation and the data used for their evaluation are not public. Vice versa, as explained in Section 4.4.6.1, the data used for the tests is public, so that a direct comparison of a future work with the proposed solution is possible.

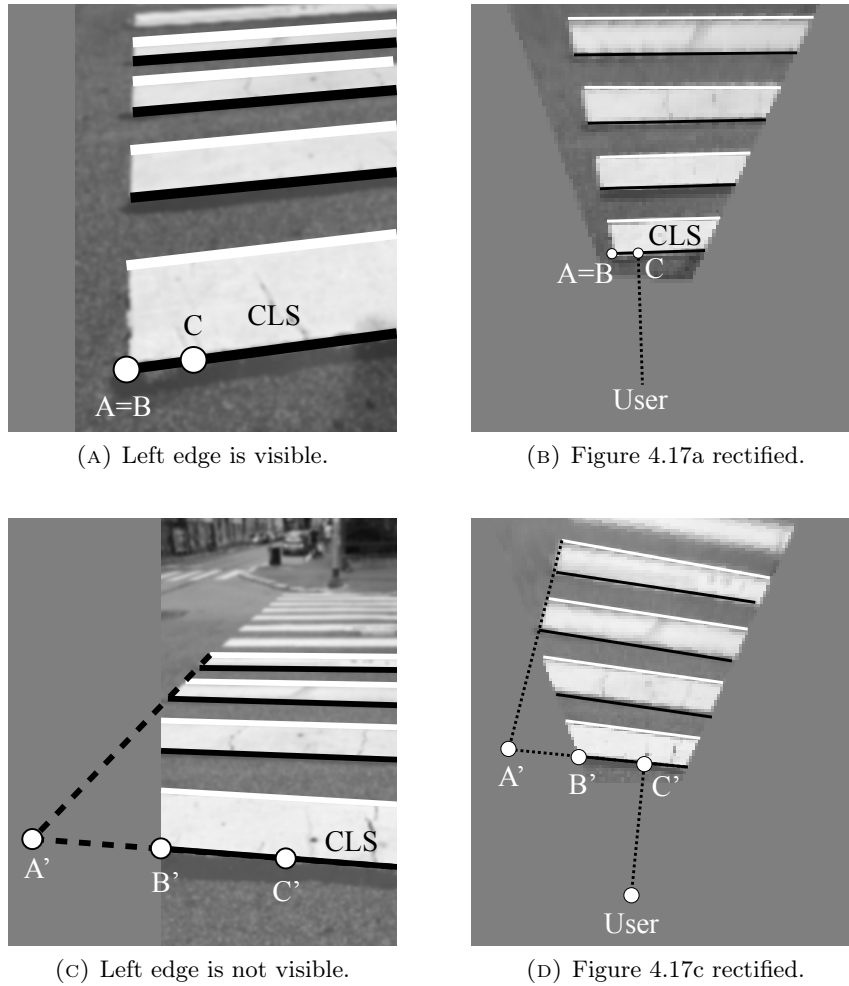


FIGURE 4.17: Distance from the left border computation.

In Section 4.4.6.1 the adopted evaluation methodology of *FusionRecognizer* and the experimental setting are described. Then, four sets of experiments are presented. The first set is aimed at measuring the impact of GPU computation (Section 4.4.6.2). The second set studies the impact of the most relevant system parameters (Section 4.4.6.3). The third set compares *FusionRecognizer* with *VideoRecognizer* (Section 4.4.6.4). Finally, the objective of the fourth set of experiments is to evaluate how precisely the position of the crossing is computed by *FusionRecognizer* (Section 4.4.6.5).

4.4.6.1 Experimental methodology and setting

When *FusionRecognizer* is run in *ZebraCrossing*, the input image is taken directly from the camera and this makes it impossible to run the recognition procedure twice with the same input. To overcome this problem, two software applications were designed. The former, called *zRecorder*, is a mobile application that records the stream of images and motor sensors data (i.e., accelerometer and gyroscope). The latter, called *zSimulator*, reads the data stored by *zRecorder*, uses it as an input to run *FusionRecognizer* and

to measure its performance. *zSimulator* can be run both on traditional devices (i.e., desktops and laptops) and on mobile devices and supports both the sequential reading of input data as well as the parallel one, that simulates the concurrent behavior of *ZebraCrossing*. Thanks to this solution it is possible to record a video and then to use it several times as an input for *FusionRecognizer*. This significantly eases the debugging process and enables regression tests, parameters tuning and reproducible experimental tests.

Two sets of images (with corresponding motion data) were recorded at 1080×1920 resolution. Both sets are publicly available³. The first set, called *TestSet1*, consists of 40 videos and 4015 frames. All frames have been manually annotated to distinguish those containing a zebra crossing (1877) from the remaining ones (2138). Videos and gravity measurements were captured on an iPad 2 device in different illumination conditions (sunny, cloudy and night).

The second set, called *TestSet2*, includes 6 videos with a total of 206 frames. In this case, for each frame also the relative position of the crossing was annotated. To minimize the approximation while collecting this information, the videos were recorded by using a tripod positioned at a given frontal and left/right distance from the crossing. Since the tripod is stationary, the frontal and lateral distances are fixed for each video. While recording each video the camera (an iPhone 5, in this case) was rotated. To measure the rotation, before starting the recording, the device was calibrated to be perfectly perpendicular with the stripes and then, for each frame, the rotation angle was measured by using gyroscopes information. Indeed it was observed that the error introduced by the gyroscopes is negligible, also considering that the video duration is of few seconds and that the device is not subject to sudden movements (since it is on a tripod).

A desktop pc was used for computationally intensive evaluations (e.g., parameters tuning) and an iPhone 5s smartphone for evaluating the execution time.

Four indicators are taken into consideration: precision, recall, execution time and the positioning accuracy. The precision, calculated as the ratio of the correctly detected crossings and all the detected crossings, measures the amount of false positives. A precision score of 1.0 means that each detection corresponds to a crossings in the examined image, conversely a lower ratio implies that some crossings were detected where none was present. The recall metric is computed as the ratio between the detected crossings and all the correct crossings in the dataset. While a score of 1.0 means that all the crossings were correctly detected, lower values indicate that some of the crossings were not. Given the safety concerns for the navigation of users with visual impairments in a dangerous environment, anything less than a perfect precision score is unacceptable, while a high recall score, although important, is less critical. In the following, unless differently stated, in the results the precision is always equal to one.

³<http://webmind.di.unimi.it/ZebraRecognizerTestSet/>

The execution time defines the average time needed to run *FusionRecognizer*. It does not take into account the time required to load the image from the hard drive nor the time required to resize the input image. Indeed, when *FusionRecognizer* is used in *ZebraCrossing*, the input image is already acquired at the necessary resolution and no resizing is needed. Clearly, lower execution time allows higher frame rates, increasing the responsiveness of the detection with respect to the user's movements. Also, it means that the procedure is less computationally intensive, with a lower power consumption.

Finally, the positioning accuracy indicates how precise is the relative position returned by *FusionRecognizer*. The positioning accuracy in a given frame is characterized by four values, one for each distance measurements. Each value is the difference between the distance computed by *FusionRecognizer* and the expected (actual) value. Clearly, positioning accuracy can only be computed if the expected relative distance is known and hence only using *TestSet2*.

4.4.6.2 Impact of GPU computation

The first set of experiments is aimed at assessing the improvements of the GPU implementation of anchors extraction (see Section 4.4.3). Figure 4.18 shows the comparison between the CPU and the GPU implementations for different values of the “resolution” parameter. As expected, this parameter significantly influences the execution time of anchors computation as this is an operation with time complexity linear in the number of pixels. Experimental results confirm this theoretical expectation. Indeed, the computation time of the CPU implementation is 2.5ms for images with resolution 90×160 , while it is almost exactly four times larger (i.e., 9.67ms) for images with four times the number of pixels (i.e., 180×320). The same holds for images with resolution 360×640 . Differently, with the GPU implementation, the total computation time is composed by a constant-time overhead (estimated to be about 1.5ms) and the actual computation, whose cost is indeed linear in the number of pixels and about 10 times faster than with the CPU implementation. So, overall, while the computation on the GPU leads to an improvement of about 30% for 90×160 images, the improvement is much larger with 180×320 images (the default resolution value) where the GPU implementation is more than 4 times faster. For larger images the benefits are even higher (e.g., for 360×640 images the GPU implementation is about 8 times faster).

One question is how the GPU implementation of anchors extraction impacts on the overall computation time of *FusionRecognizer*. Figure 4.19 provides an answer by showing, at the default resolution, how the entire computation time of *FusionRecognizer* is divided between anchors extraction and all other operations. When anchors extraction is computed in CPU, it requires almost the same time as all the other operations (precisely, anchors extraction takes 44% of the entire computation time). Vice versa, with the GPU

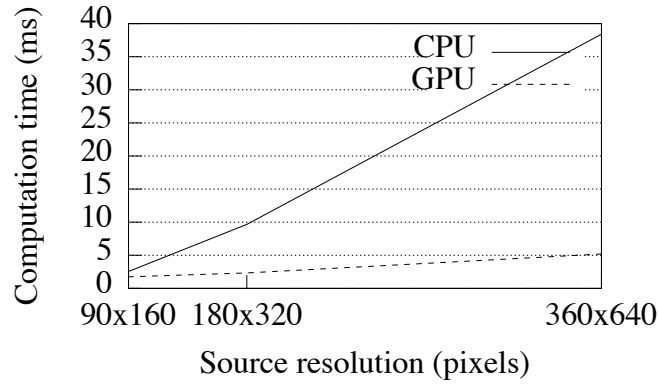
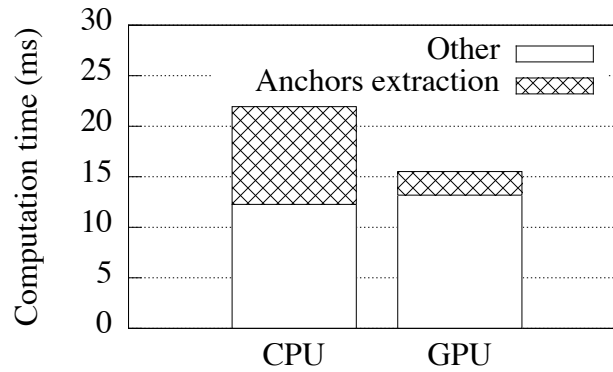


FIGURE 4.18: Computation time of anchors extraction

implementation, anchors extraction requires 15% of the entire computation time. Overall, since the GPU implementation is about 4 times faster, the overall *FusionRecognizer* computation time improved by about 30%.

FIGURE 4.19: Impact of anchors computation time in *FusionRecognizer*

4.4.6.3 Parameters tuning

While a large set of system parameters was evaluated and tuned, the impact of three of them appears to be most significant: “resolution”, “group-angle threshold” and “color consistency magnitude threshold”. The “resolution” parameter specifies the size of the image on which the detection is run. The “group-angle threshold” defines the maximum angular distance between two line segments that are grouped together (see Section 4.4.4). The “color consistency magnitude threshold” defines the minimum difference in color intensity (value range between 0 and 255) between a stripe and the average color of the crossing (see Section 4.4.5). These parameters are listed in Table 4.1 together with their minimum, maximum and default values.

Figure 4.20 shows that with a very low resolution (below 90×160) recall diminishes drastically. This is due to the fact that in these cases the features are hard to detect. For high resolutions (above 180×320) there is also a reduction in recall due to the

Parameter	Min	Default	Max
Resolution	90×160	180×320	720×1280
Goup-angle thres.	1.5	3	7.5
Color magnitude thres.	1	5	9

TABLE 4.1: Most influential parameters and their values

fact that noise and imperfections are more visible and impair drastically the segment detection stage. This behavior can be offset by using stronger smoothing during the preprocessing step (see Section 4.4.1). The best results can therefore be achieved with a resolution of 180×320 .

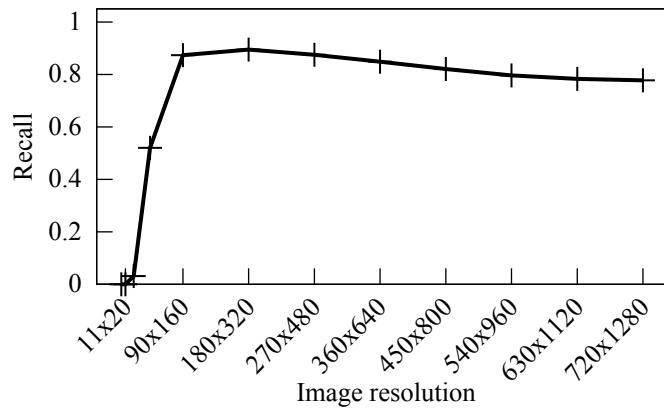


FIGURE 4.20: Tuning of "resolution" parameter

For "group-angle threshold" observe (see Figure 4.21a) that, for larger values of this parameter, recall is higher due to the fact that larger blocks of line segments are generated with the application of the "slope" criterion hence it is less likely that they are pruned by the "number of edges" criterion. However, for values larger than 3° , some false positives can be introduced and hence precision diminishes, although very slowly. For this reason, the default value is 3.

The analysis of "color magnitude threshold" is similar: for smaller values of this parameter the "color consistency" criterion is easier to satisfy hence there is a higher recall. However, for values smaller than 5 precision is less than 1. Hence, the default value chosen is 5.

4.4.6.4 Comparison with *VideoRecognizer*

FusionRecognizer is compared with with the *VideoRecognizer* solution proposed in [32] and described detailedly in Section 4.3, and a previous version of *FusionRecognizer*, proposed in [34]. These are called "Version 1" and "Version 2", respectively. The three solutions are compared according to two metrics: recall and computation time. For each version the corresponding default system parameters, which, as previously stated, are tuned to yield a precision equal to 1 are used.

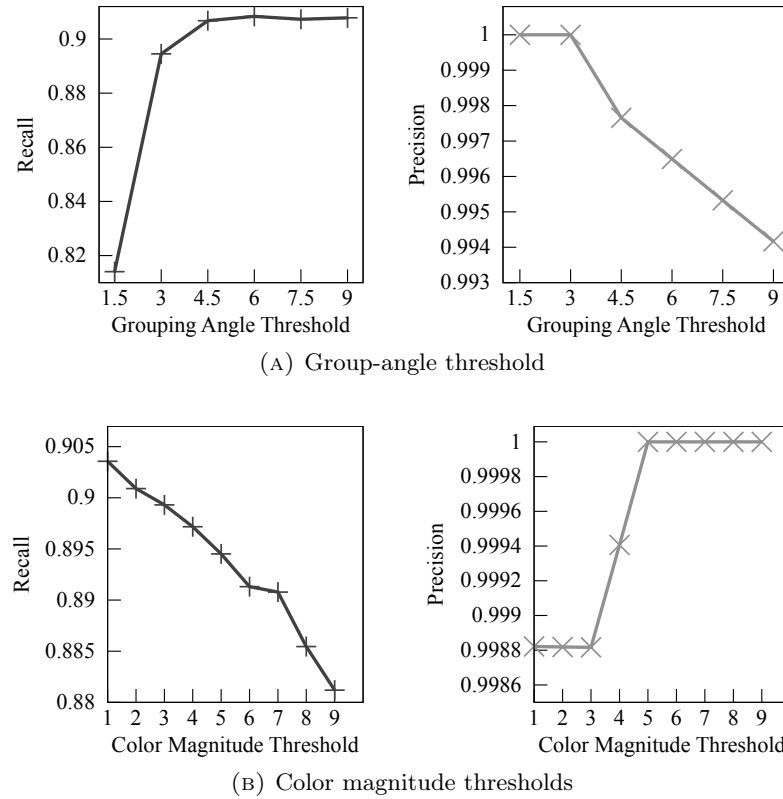


FIGURE 4.21: System parameters tuning.

For what concerns the recall, Figure 4.22 shows that it improved from .69 in Version 1 to .78 in Version 2 up to .89 in the current version of *Recognizer*. The improvement from Version 1 to Version 2 is mainly due to the fact that in Version 2 the geometrical properties are checked on the rectified image. The improvements from Version 2 to the current version is due to the number of improvements described in Sections 4.4.3 and 4.4.4.

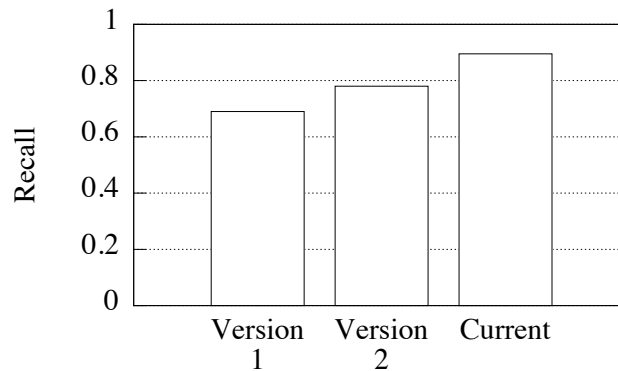


FIGURE 4.22: Recall comparison

For what concerns the computation time, in Version 1 the average time to process each frame is 74ms, while in Version 2 it is 23ms. In the current version of *Recognizer* the average time is 22ms with the CPU implementation of anchors extraction while it is

16 with the GPU implementation. The small improvement between Version 2 and the current CPU implementation is due to two contrasting factors: on one side, the code was engineered and optimized, hence improving the computation time by about 20%. On the other side, a bug was fixed in the line segments merging algorithm (see Section 4.4.3). The effect of the bug was to erroneously terminate the merging algorithm, hence resulting in partially incorrect result but faster computation. After fixing that bug all line segments are now correctly merged, but the improvement in computation time from Version 2 to the current version is negligible. Still, the GPU implementation of anchors extraction guarantees an improvement of about 30%.

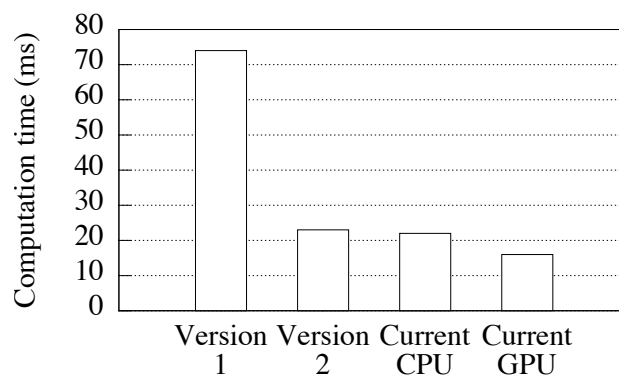


FIGURE 4.23: Computation time comparison

4.4.6.5 Positioning accuracy

The results of a set of experiments aimed at asserting the positioning accuracy of *FusionRecognizer* in terms of the four distance measurements is reported. As noted in Section 4.4.6.1, the dataset is publicly available⁴ and it is composed of 6 short videos totaling 206 frames, captured at well known positions and angles. The position of the stripes computed by *FusionRecognizer* is evaluated against the known positions. In the following “error” indicates the absolute value of the difference between the distance (frontal, angular or left/right) returned by *Recognizer* and the expected (correct) distance.

For what concerns the frontal distance, the average error is 0.22m. While an error of 20 centimeters does not significantly impact on the overall effectiveness of *ZebraCrossing*, in some cases the error is much larger due to the fact that the visible stripe closer to the user is not recognized. In current tests this problem introduces an error up to 1m (i.e., when the two closer stripes are not recognized). Fortunately the incorrect recognition of the closer stripes happens in few frames (less than 3%) that are generally non-consecutive (the longest sequence measured is composed by two consecutive frames). This makes it possible to identify the occurrence of this problem in the *Logic* module by checking for

⁴<http://webmind.di.unimi.it/ZebraRecognizerTestSet/>

sudden changes in the frontal distance. Indeed, since the temporal distance between two consecutive frames is less than 0.1s (frequency is more than 10 frames per second), a change in the frontal distance larger than 0.5m clearly indicates that the closer stripes have not been recognized.

For what concerns the rotation angle, the average error is about 2.2° . In some sporadic cases it is possible that the error is up to 9.5° (in 93% of the cases the error is less than 7°). However, in this case it is not possible to check for sudden changes, as a user can possibly rotate very quickly. However, values can be easily smoothed by using a moving average in the *Logic* module. For example with a moving average of length 3, the average error in the rotation angle is 1.0° and the maximum error is 3.0° .

During the experiments it was observed that the computation of the lateral distance is subject to a non-negligible approximation caused by two factors: first, EDLines frequently does not recognize the entire stripe edge, but just a portion of it and consequently the computation of the border is not always precise. Second, the projection of the user's position on *CLS* (the line segment closest to the user) can be imprecise due to approximations in the computation of the *CLS* angle. To address the former issue, the proposed solution excludes, from the border computation, the points that introduce an error above a given threshold. This is useful, for example, when there are few line segments that are much shorter than the actual stripe edge. To address the latter issue, when computing the projection on *CLS*, instead of using the angle of *CLS*, the average angle computed among all the stripes was used. The resulting technique always correctly identifies a lateral distance as *not quantifiable* (i.e., the border is out of the field of view, see Section 4.4.5.1). In some rare cases it happens that, even if the border is visible, it is still identified as *not quantifiable*. This is often due to the fact that the border is only visible in the stripes that are far away from the user and these stripes are not recognized. In any case, in 87% of the cases, if a border is visible, than it is identified by the technique and, in these cases the average error is 0.25m.

4.5 The user interaction module

The *Navigator* component implements the multi-modal user interaction paradigm. From the application-design point of view, this module faces a number of challenges. First of all, it is necessary to provide the user with a large amount of information (i.e., the position of the crosswalk) mainly through audio but, at the same time, it is important not to overwhelm the user with too much audio.

Another challenge is due to the usage context of the application. Recall that the users of *ZebraCrossing* are on the move. In this context, blind persons often use one hand to hold the white cane or the guide dog, hence it is important to be able to hold the device firmly with one hand only. In addition, since the user is interested in nearby zebra

crossings, the device's camera has to be pointed towards the user's immediate vicinity with a suitable angle for a correct detection of possible close crossings.

To face these challenges, the interface has been designed based on four main principles:

- **Quiet navigation** - The system should distract the user as little as possible, all the while communicating information necessary for guiding the user correctly. Especially while the user is crossing the street, the distraction should be kept at minimum.
- **Step by step** - In order to limit the quantity of information conveyed to the user at a time, the navigation is divided in steps, each having a small set of messages that can be given to the user one at a time.
- **Thumb input** - One handed input is limited to the usage of only the thumb finger. This allows to keep a firm grip on the device while using simple gestures for interacting with the device
- **Camera pointing assistance** - The user is assisted in pointing the device towards the ground in such way to be able to detect nearby zebra crossings correctly.

The "Step-by-step" message generation logic is described in Section 4.5.1. The output modes ("speech", "mono sonification" and "stereo sonification"), that were designed to convey the computed messages, will be discussed in Section 4.5.2. The "Thumb-only" input technique will be explored in Section 4.5.3. Finally, the experimental evaluation of *Navigator* will be described in Section 4.5.4

4.5.1 Step-by-step navigation

The navigation is divided into three operational modes that correspond to the three main activities involved in the act of crossing: searching for the crosswalk (operational mode "Search"), reaching a good starting position to cross (operational mode "Align") and actually crossing (operational mode "Cross"). Each operational mode is associated with different feedback messages that are optimized for that specific situation and with different parameters adopted in the crosswalk recognition process. Some feedback messages, when triggered, also switch the system from one operational mode to another (e.g., when correctly aligned, the application triggers the "Cross" message and switches from the "Align" to the "Cross" mode).

The "Search" operational mode sets the system parameters in such way to favor the detection of distant crossings and yield the fastest possible detection rate. The "Align" and "Cross" modes trigger the detection algorithm only after a certain amount of time passed or when a significant rotation has been detected from the device's gyroscope (since

rotations have a huge impact on the differences in scenery captured by the camera), thus reducing the detection rate and battery consumption. Additionally, the “Cross” operational mode also reduces the audio volume of the feedback messages in order to reduce the distraction for the user and raise the awareness to the surrounding sounds. Initially we considered also the absence of feedback during the crossing. However, a preliminary test showed how, in absence of feedback, users tend to drift laterally during straight walking. In particular, in one test, one user would consistently drift left so much to approach the border of the stripes every meter and thus require software’s feedback to readjust his position.

“Not found” message is activated each time the device does not detect any crossings. When the system is in other operational modes, this message triggers the “Search” operational mode. The “Approach” message is triggered when a crossing is detected and the user’s distance from the detected crossing is higher than a specified threshold. It also causes the switch from “Search” and “Cross” operational modes to “Align” mode. When the difference between the user’s direction and the direction of the stripes is above a threshold, the message “Rotate Left” or “Rotate Right” are triggered according to the rotation direction. When the user’s projection on the lowest line of the stripes is outside of the left or the right border of the stripes, respectively the “Step Left” or “Step Right” feedback is triggered. The last four messages will induce a switch from the “Search” mode to the “Align” mode. The “Cross” message signals that the user is aligned with respect to the stripes and can cross the street. Thus, it also causes a switch from “Search” and “Align” to “Cross” operational mode.

To assist the user in pointing the device’s camera correctly, two mode independent feedback messages (“Incline Up”, “Incline Down”) are provided. At any point during the interaction, the device’s inclination with respect to the ground plane is computed from the gravity measurements (See Section 4.4) and the user is notified if the device’s inclination is not optimal. The inclination is considered correct when the device can capture the ground plane between the minimum distance d_{min} and the maximum distance d_{max} from the user. These values are set based on the current operational mode, and the device’s correct inclination range is computed starting from them as follows.

Given the user’s height h and the camera’s vertical field of view β_y (See Section 4.4.2), if we want to capture the ground plane at a minimum distance of d_{min} , the inclination has to be at most $\theta_{max} = atan(d_{min}/h) + \beta_y$. Instead if we want to capture the ground plane at a maximum distance of d_{max} , the inclination has to be at least $\theta_{min} = atan(d_{max}/h) - \beta_y$.

Table 4.2 lists the feedback messages and possible operation mode changes they cause.

The output modes described in Section 4.5.2 define the ways to convey the listed feedback messages to the user.

Message	Description	States
Not found	No crosswalks found, search	Align → Search Cross → Search
Approach	Crosswalk is distant, approach	Search → Align Cross → Align
Rotate left	Rotate in place to the left	Search → Align
Rotate right	Rotate in place to the right	
Step left	Do a lateral step to the left	
Step right	Do a lateral step to the right	
Cross	Crosswalk is in front of you, cross	Search → Cross Align → Cross
Incline up	Incline the device up	-
Incline down	Incline the device down	

TABLE 4.2: Messages and state changes

4.5.2 Output interaction modes

In order to deliver the guidance messages to the user, the *Navigator* module employs auditory feedback that can be conveyed through two separated interaction techniques: speech and sonification (See Section 2.1). The latter can further be split in two categories: mono and stereo. Three output modes are therefore available to the user: “speech” (Section 4.5.2.1), “mono sonification” (Section 4.5.2.2) and “stereo sonification” (Section 4.5.2.3).

4.5.2.1 Speech output

Considering the computed feedback messages, the *Navigator* module delivers to the user a set of instructions using the OS’s text-to-speech API (present on both iOS and Android). Considering the fact that the subjects who participated to the evaluation were all Italian mother-tongue, the instructions were delivered in Italian (an English translation is available between brackets).

Table 4.3 list the feedback messages, the corresponding speech output in Italian and the translation in English.

The speech message is reproduced only once and the user should continue with the last suggested action until *Navigator* changes to another suggested action. The user can replay the last message with a specific thumb gesture (See Section 4.5.3).

One of the main problems with speech guiding mode is that it does not convey quantified information about the relative position between the user and the crossing. For example, if the user is instructed to rotate on the right, he/she does not know how much rotation is required in order to be aligned with the crossing. In theory, it could be possible to design a speech guiding mode in which the quantity is reproduced (e.g., “rotate right

Message	Output	Translation
Not found	“Non trovato”	“Not found”
Approach	“Strisce davanti”	“Crossing ahead”
Rotate left	“Ruota a sinistra”	“Rotate left”
Rotate right	“Ruota a destra”	“Rotate right”
Step left	“Passo a sinistra”	“Step left”
Step right	“Passo a destra”	“Step right”
Cross	“Attraversa”	“Cross”
Incline up	“Alza il dispositivo”	“Raise the phone”
Incline down	“Abbassa il dispositivo”	“Lower the phone”

TABLE 4.3: Speech messages in Italian and English

- 20 degrees”). However, this guiding mode would be much more verbose and, most importantly, it is clearly impractical to update the quantity associated to the message (i.e., the rotation angle in the above example) while the user is moving.

To overcome this problem, the sonification guiding modes, described in the following sections, inform the user about the *quantity* associated with the instruction.

4.5.2.2 Mono sonification output

The process of user-centric analysis of the system raised another important requirement that has a direct impact on the sound design: some people with VIB are not willing to wear any type of headphones while walking, not even bone-conducting headphones⁵. For that reason, a mono sonification output has been designed and evaluated. It delivers the audio signal exactly in the same way to the two ears and hence is suitable to be played by the device speaker.

In order to deliver left-right-type messages without relying on sound spatialization, low pitch sounds were associated to a rotation/step towards the left, and high pitch sounds towards the right. This choice can be intuitively explained considering the keyboard of the piano from the point of view of the player (high-pitch notes on the right).

Considering the list of speech messages in Table 4.2, the following mono sonifications have been designed and implemented:

- *Rise/lower the phone*. Impulsive sound with fast transients and harmonic spectrum (similar to a short beep). Two quick repetitions with no pause. High pitch (800 Hz) for the ‘rise’ message and low pitch (200 Hz) for the ‘lower’ message. The signal is repeated increasing linearly the rate (from 1 Hz to 2.5 Hz) the closer the user gets to the right inclination.

⁵Bone-conducting headphones do not occlude the ear canal, therefore do not impede the perception of the sounds from the surrounding.

- *Rotate left/right.* Impulsive sound with fast transients and in-harmonic spectrum (similar to a percussive sound on metal). The left-right information is delivered modifying the frequency of the stimulus; 300 Hz for the left rotation and 1200 Hz for the right rotation. The repetition rate of the sound is modified linearly from 1.6 Hz (large rotation) to 3.3 Hz (small rotation), varying continuously until the user reaches the target angle.
- *Step left/right.* Impulsive sound with fast transients and in-harmonic spectrum (similar to a percussive sound on wood). Two fast (200 ms) repetitions. The left-right information is delivered modifying the frequency of the stimulus; 300 Hz for the left step, and 1200 Hz for the right step.
- *Not found.* Low frequency (200 Hz) in-harmonic sound, slow transients, two repetitions (300 ms the first and 500 ms the second).
- *Crossing ahead.* Pure-tone (single frequency with no harmonic components) impulsive sound. A rising scale of 6 notes (between 800 and 1700 Hz, one each 100 ms) for a required 10 m advance, 5 notes for 8 m, 4 notes for 6 m, 3 notes for 4 m and 2 notes for 2 m. The scale is repeated every 1000 ms, modifying the message as the person gets closer to the target.
- *Cross.* Impulsive sound with fast transients and in-harmonic spectrum (similar to a percussive sound on wood). A group of three notes (one note every 150 ms) with fundamentals at 500-800-1000 Hz is repeated every 1200 ms. If the user is required to proceed towards the right, the frequency of the fundamentals is divided by 0.33 (lower pitch), while if towards the right is multiplied by 2 (higher pitch). The level of the sound is rather low, but it becomes louder (up to +20 dB) the more the user needs to modify the path towards the left or the right. When the user is at less than 4 meters from the target, the delay between repetitions is decreased linearly (down to 700 ms).

4.5.2.3 Stereo sonification output

Differently from the mono sonification, in the stereo sonification mode the audio signal is delivered differently to the two ears through sound spatialization. Thus, the user can clearly perceive certain sounds as coming from the left or from the right. This sonification requires the user to wear a pair of headphones, and employs, for a determined set of messages, a binaural spatialization approach [45]. Considering the low resolution of bone-conducting headphones in terms of high frequencies (above 10 kHz) and the individual-related features of a full Head Related Transfer Function (HRTF) simulation, this was not implemented performing a full spatialization, but simply modifying the differences in level and time of arrival of the sound at the two ears (i.e. Interaural Level Differences - ILD and Interaural Time Differences - ITD). In the following, the spatialization was

performed employing ILD (from 0 to 10 dB) and ITD (from 0 to 0.5 ms). For a precise spatialization, the sound had to feature a large and dense spectrum. In order to satisfy this constraint, it was conceived to create a set of impulsive sounds of short duration, using an approach similar to the one employed by [98].

The following stereo sonifications have been designed and implemented:

- *Rise/lower the phone.* Same as mono mode.
- *Rotate left/right.* Same sound as mono mode, frequency 500 Hz. The impulse is continuously repeated every 400 ms, and is spatialised on the left if the user needs to turn left, and vice-versa if the user needs to turn right. The repetition continues until the user can centre the sound on the front (therefore when reaching the target angle).
- *Step left/right.* Same sound as mono mode, frequency 500 Hz. Sound spatialized on the left or on the right (depending on the required direction)
- *Not found.* Same as mono mode.
- *Crossing ahead.* Same as mono mode.
- *Cross.* Same sound as mono mode, with frequencies 500-800-1000 Hz. The left-right direction is given by gradually spatializing the sound on the left or on the right, so that the task of the user is to rotate in order to keep the sound central.

4.5.3 Thumbs-only input

As previously noted, all the application input needs to be inserted with the thumb only and with easy-to-do gestures, which is possible since only limited input from the user is required.

Overall, the entire application requires 5 commands from the user, listed in Table 4.4

Gesture	Command
Tap	Repeat the last message with speech
Swipe left right	Previous Next output mode
Swipe up	Toggle detection
Swipe down	Extended information

TABLE 4.4: Thumb gestures and associated commands

The description of the specified gestures is described in the following.

Repeat the last message with speech. The *Navigator* automatically provides instructions only when strictly needed in terms of short speech output or with the sonification techniques described previously. However, in presence of background noise when the output is not perceived correctly or, in case of sonifications, if the user is unsure of the meaning of the played sound, a speech confirmation can be requested by the user with a tap on the screen. As seen in the experimental results, this functionality is often requested by the users during the learning of the sonification interaction techniques designed.

Previous | Next output mode. Experimental results show how there is not a single guiding mode that is the best for every visually impaired user. Thus, *Navigator* allows the user to switch between the three guiding modes with horizontal swipe gestures on the screen. Swiping to the left will change the mode to the previous one while swiping right will switch to the next one. The order of the guiding modes is: Speech, mono sonification, stereo sonification.

Toggle detection. The user is interested in the detection of zebra pedestrian crossings only when interested in crossing the street. In order to reduce the battery consumption when the user does not desire to run the detection, a gesture is available to halt or start *ZebraCrossing*. Swiping up with the thumb on the screen will trigger the detection or turn it off.

Extended information. The user can also be interested in knowing the relative position of the detected stripes in details so to have a more precise idea on how to align correctly. With a swipe down with the thumb on the device's touchscreen, the following information is conveyed with the speech output: rotation distance between the user's direction and the crossing's direction (e.g., 10 deg), the lateral distance between the user and the stripes (e.g., 1.5m to the left), the frontal distance between the user and the crossing (e.g., distant 5m) and the length of the visible crossing (e.g., 10m long).

4.5.4 Evaluation

We conducted three sets of empirical evaluations: a qualitative evaluation with blind-folded sighted subjects (Section 4.5.4.1), a qualitative evaluation with blind subjects (Section 4.5.4.2) and, finally, a quantitative and qualitative evaluation conducted with three visually impaired subjects (Section 4.5.4.3). In Section 4.5.4.4 we report a discussion of the empirical results.

The trials were conducted with an iPhone 5s and wireless bone-conducting headphones⁶.

⁶Headphones are *Aftershoks bluez 2*

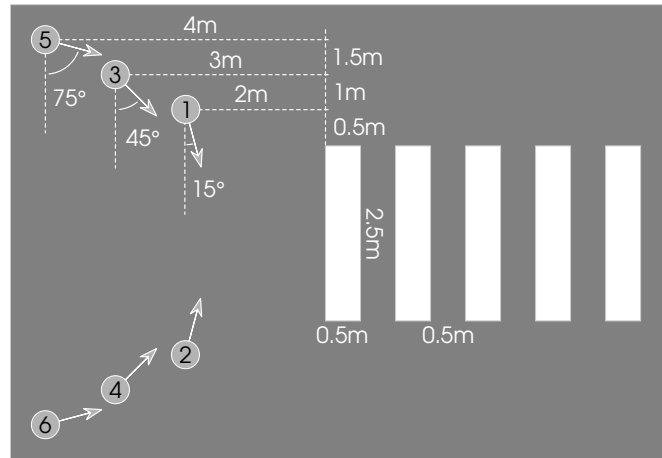


FIGURE 4.24: Layout of the plastic sheeting where tests were conducted. Numbers and arrows represent starting points and starting directions, respectively.

4.5.4.1 Quantitative evaluation with sighted subjects

The quantitative evaluation was conducted with 11 blindfolded sighted subjects.

Tests were conducted in an outdoor environment where real-size zebra crossing was represented on a large plastic sheeting. The zebra crossing used during the experiments are compliant with Italian traffic regulations: they are composed by five light stripes over a dark background and each stripe is 2.5m large and 0.5m wide⁷ (see Figure 4.24).

The outdoor environment was chosen in order to give a more realistic setting to the tests. In order to reduce the subjects' ability to orientate by means of environmental sounds and audio feedback and to minimize hazards, it was decided to carry out the test in a large courtyard. Sound of traffic and other environmental noises were audible, but particularly diffuse in the environment, and generally not usable for orientation purposes. For the same reason, the plastic sheeting was moved or rotated after each test, so that it was impossible for the subjects to predict the position of the zebra crossing based on previous trials. Also, in order to avoid that tactile or audio feedback coming from the ground surface could give clues to help orientation, during each test, the subjects walk or stand over the plastic sheeting, that is actually much larger than the zebra crossing itself.

Each test was organized into three phases: learning, practice and measurements. During the learning phase each subject had access to a document describing the test structure, introducing *ZebraCrossing* and the three different auditory guiding modes. The document was presented in the form of an HTML page, so that subjects could listen to sonification examples⁸.

⁷Italian regulation defines zebra crossings that are similar to those used in most countries worldwide

⁸The document was presented in Italian. Its English translation is available here: <http://webmind.di.unimi.it/zebraexplanation/>

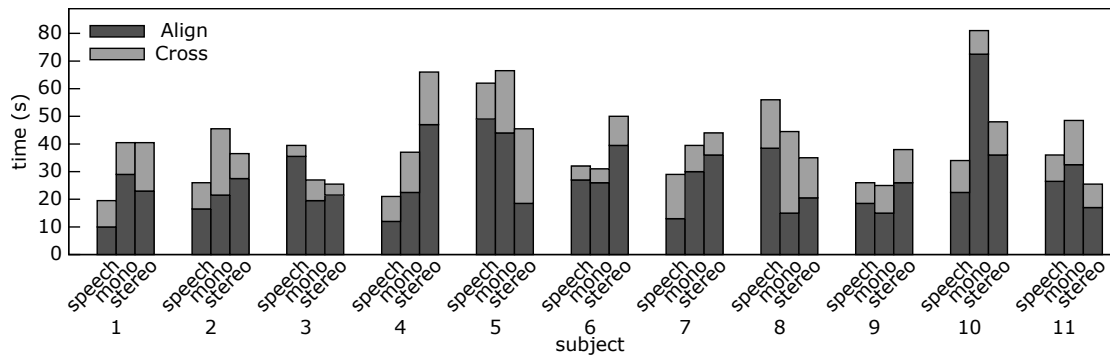


FIGURE 4.25: Average alignment and crossing time in the two rounds.

During the practice phase each subject could try *ZebraCrossing* with the three auditory guiding modes. No time constraint was enforced: each subject could freely decide how long to practice with each guiding mode until he/she felt comfortable with it. On average, subjects tested the speech guiding mode for about 1 minute, and the other two modes for about 2 minutes each.

During the measurement phase each subject was asked to autonomously align with the zebra crossing and to actually cross it. These two operations were repeated for two “rounds” of tests. During each round, three trials are conducted, one for each guiding mode in the order: speech, mono, stereo. Each trial starts from a different starting point with an associated starting direction. The choice of the starting points was guided by the idea that the time and effort required to cross should be almost the same for all starting points. So, after some informal trials, we decided to use the 6 starting points depicted in Figure 4.24.

During the measurement phase, the *ZebraCrossing* app recorded a number of parameters for each trial and in particular: the time to align (i.e., to reach the first stripe), the time to cross (i.e., from the first stripe to the end of the crossing), the complete list of messages and the number of tap on the screen to repeat/clarify the message.

During the measurement phase all subjects have been able to successfully complete all crossings. The only exception is subject 6 that, during the trial with the mono guiding mode, second round, misinterpreted a “rotate left” message and walked straight. Since the subject was going to hit a parked car, the supervisor had to stop the test.

Figure 4.25 shows, for each subject and each guiding mode, the average time required in the two rounds to align and cross. We can observe that 5 subjects have been able to align and cross faster with speech guiding mode, 2 subjects with mono and 4 with stereo. Mean alignment time is 24s, 29s and 28s with speech, mono and stereo modes, respectively while mean crossing time is 10s, 14s and 12s, respectively. Overall, the mean time to align and cross is 34s, 44s and 41s.

Above results seem to suggest that there is not a clear difference in crossing time for the three guiding modes. These results can be also graphically observed in the boxplot shown in Figure 4.26a. This chart also highlights that, differently from what expected, there is no learning effect between the first and second round. Indeed, on average, the crossing time in the second round is slightly lower for the mono guiding mode and slightly higher for the other two guiding modes.

Another metric that can help understanding the performance of the three guiding modes is the total number of changes in the message to be conveyed during a crossing (in the following we call this metric “number of messages”). Clearly, a smaller value indicates higher performance. In this case it emerges that speech and stereo guiding modes yield very similar results, while mono sonification requires a slightly larger number of instructions, on average (see box plot in Figure 4.26b).

Inferential statistics have been performed to identify whether the differences between guiding mode groups are statistically significant.

Considering the time to align and cross, the data sets are normally distributed, therefore a one-way ANOVA was conducted. The results show that there are no statistically significant differences between the three groups ($F(2, 63) = 1.178, p = 0.314$).

Considering the number of messages, the data sets are not normally distributed, therefore a Kruskal-Wallis test was conducted. As for the time measurements, no statistical difference was found between the three groups ($\chi^2 = 0.164, p = 0.921$).

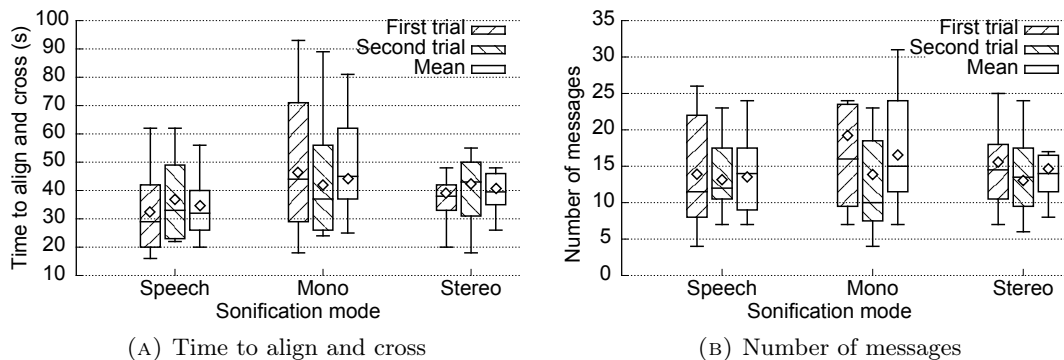


FIGURE 4.26: Boxplot representation (\diamond symbol represents mean)

During the tests with the two sonifications, we observed that some subjects were frequently requiring *ZebraCrossing* to read the current instruction by tapping on the device screen. We believe that these subjects did not get acquainted with the sonification technique and hence they needed to get constant speech feedback in addition to the sonification. For example, during the second round with the stereo guiding mode, subject 4 tapped on the device almost three times for each new message received (67 taps and 24 messages). Vice-versa, other subjects used the tap gesture only sporadically. For example, subject 5 tapped only 2 times in the second round with the mono guiding mode during which he received 15 messages in total.

4.5.4.2 Qualitative evaluation with blind subjects

The qualitative evaluation was conducted in an indoor environment during an exhibition of assistive technologies⁹. The evaluation was conducted by 12 blind subjects and it was divided into three phases: learning, practice and questionnaire. The learning and practice phases were conducted with the same methodology as for the quantitative evaluation. The questionnaire is aimed at providing an answer to the following questions.

- How much demanding is it to follow the instructions with each guiding mode?
- Which is the preferred guiding mode?
- Which is the level of satisfaction in the use of *ZebraCrossing* with the preferred guiding mode.

To provide an answer to the questions above, we designed two lists of sentences. The former is derived from the System Usability Scale and is composed of 7 sentences, related to the ease of use of the three auditory guiding modes. Subjects could evaluate each sentence with a rate from 1 ('totally disagree') to 5 ('totally agree').

The latter list of sentences is derived from IBM Computer Usability Satisfaction Questionnaire (CSUQ) and is preceded by a single question, asking which is the preferred auditory guiding mode. Then, 8 sentences follow, all referring to *ZebraCrossing* with the preferred auditory guiding mode. Subjects could rate each sentence from 1 ('totally disagree') to 7 ('totally agree').

All subjects agree that instructions provided with the speech guiding mode are simple to follow (see Figure 4.27). There is not the same consensus about the easiness of following instructions with the mono guiding mode: 8 subjects argue they are easy to follow (with a rate of 4 or 5) while 4 subjects argue they are not so easy (with a rate of 3). Very similar result for the stereo guiding mode: 8 subjects argue that instructions provided with the stereo guiding mode are easy to follow. Interestingly, only one subjects found the instructions provided with both mono and stereo guiding modes hard to follow. Instead, 6 users found one of the two guiding modes is hard to follow, while the other is not. This suggests that subjects have clear and contrasting preferences. To confirm this, 50% of the subjects argue that mono guiding mode is more intuitive than stereo, while 50% argue the opposite. None of the subjects believes that the two sonifications have the same level of intuitiveness.

There is another statement in which there is not a consensus among the subjects: 'Hearing sounds from the environment is more difficult in the speech mode than in mono or stereo'. 6 subjects agree with this sentence (rate of 4 or 5), 4 subjects do not have a clear position (rate of 3), while 2 users disagree (rate of 1 or 2).

⁹HANDImatica 2014, held in Bologna, Italy.

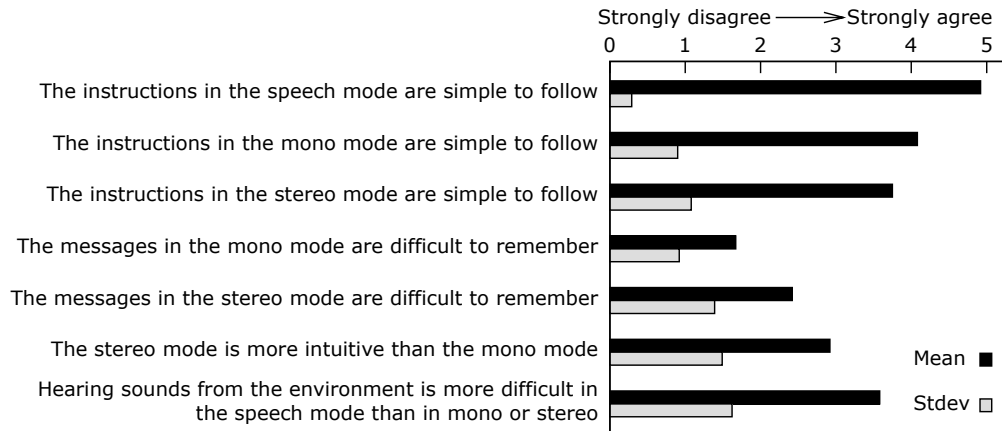


FIGURE 4.27: Questionnaire and results, first part.

The difference feelings of the subjects reflect in their preferred guiding mode. Three subjects prefer the speech guiding mode, 4 prefer the mono and 5 prefer the stereo one.

In the second part of the questionnaire, devoted to establish the overall level of satisfaction, subjects converge more towards a positive view of *ZebraCrossing* (see Figure 4.28). Indeed, subjects argue to be satisfied by the ease of use of the application (rates 7, 6 and 5 given by 5, 5 and 2 subjects, respectively) and that they have been able to complete the crossing using *ZebraCrossing* (rate 7 from 9 subjects). Vice versa, subjects mainly agree on the fact that learning to use the application requires some effort. The sentence “It was easy to learn to use this system” got an average rate of 5.3. Finally, most subjects believe that the auditory interface is pleasant (9 subjects give a rate of 6 or 7).

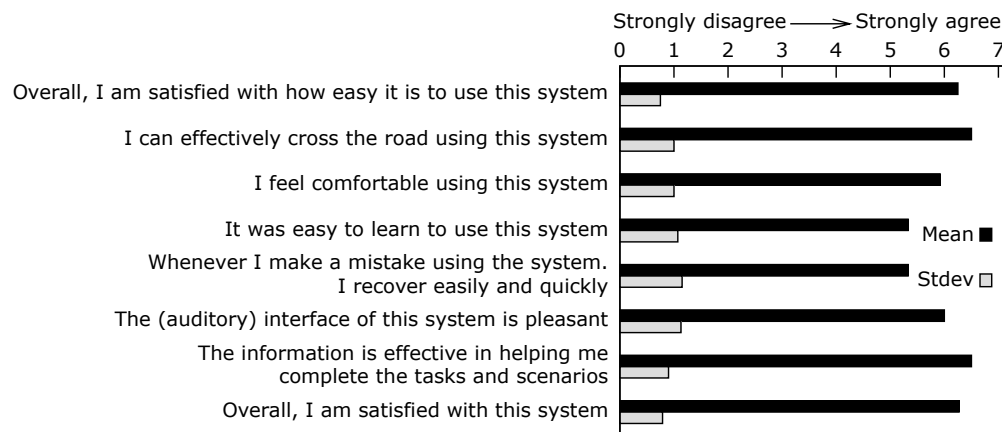


FIGURE 4.28: Questionnaire and results, second part.

4.5.4.3 Qualitative and quantitative evaluation with visually impaired subjects

The third set of tests consisted in a quantitative and qualitative evaluation conducted with three subjects with severe visual impairments. The evaluation was conducted with

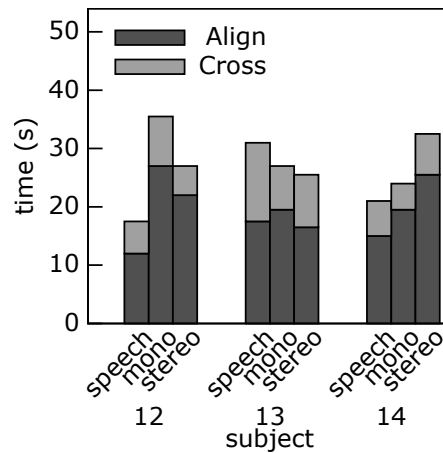


FIGURE 4.29: Crossing time in the 6 tests conducted by each of the 3 subjects with visual impairments.

three subjects: one of them was blind, the other two were low visioned and not able to recognize zebra crossing through residual sight.

The evaluation consisted in three phases. The first phase is analogous to the quantitative evaluation described in Section 4.5.4.1. Note that the two low visioned subjects were blindfolded.

The second phase was conducted in a urban crossroad and consisted in a set of about 10 crossings attempts. A supervisor was constantly taking care of the subject, avoiding any hazard. At each crossing attempt the supervisor was guiding the subject to the crossing vicinity and then asked the subject to align with the crosswalk. Once aligned, the subject had to wait for the traffic light to turn green (this information was provided by the supervisor) and then was asked to cross. In case the crossing did not complete before the traffic light turns yellow, the supervisor was instructed to guide the subject towards the sidewalk. No formal measurements were collected during this phase. The objective is to allow the subjects to use *ZebraCrossing* in a real environment.

The third phase consisted in the qualitative evaluation described in Section 4.5.4.2 with an additional set of open questions.

In phase one, all subjects have been able to successfully complete the crossing in all attempts. Figure 4.29 shows the time to align and cross measured in phase one. For what concerns the comparison among the three guiding modes, results are not dissimilar to those presented in Figure 4.25. One difference is that, in the case of subjects with visual impairments, the average crossing time is about 27s with the three guiding modes, hence it is more than 10s faster than for blindfolded sighted user. The number of messages is also similar: mean values are 20, 11 and 14 for the three guiding modes. In this regard we have to underline that Subject 12 (the visually impaired subject) had some problems, at the beginning, finding the correct inclination of the device and this caused a large number of ‘raise’ and ‘lower’ messages in the two runs with the speech guiding mode.

In phase two, all users have completed the crossing before the traffic light turned yellow. The subjects conducted at least one test with each guiding mode, but they were left free to choose how to conduct the majority of test and all of them choose to use their preferred guiding mode (that is listed below).

In phase three, it emerges that the three subjects agree on the fact that the instructions provided in the speech and the mono guiding modes are easy to follow while the same holds only partially with the stereo guiding mode (two subjects rated 4 and the other rated 3). Vice versa, there is not a consensus about how hard it is to remember the sonifications; two subjects argue they are hard to remember, while Subject 14 agree the opposite.

Meaningfully, each of the three subjects prefer a different guiding mode. Subject 11 prefers stereo guiding mode and justify the choice by clarifying that the stereo guiding mode “provides both the spatial references and the clearness of the speech messages that can be activated by tapping”¹⁰. Subject 12 declares to prefer the speech guiding mode because it is less cognitive demanding. This subject comments that “you need to get used to this app, because when you are crossing you need to pay attention to the surrounding. With the stereo [and mono] guiding mode[s], you need to concentrate to remember the sounds [i.e., the association between the sounds and the instruction] and this may distract you”. Finally, Subject 13 prefers the mono guiding with these motivations: “I like the other two [guiding modes] as well. Still, stereo [guiding mode] requires me to concentrate, while speech messages can get confused with other sounds in the environment”.

Finally, the last questions about the overall satisfaction denote high satisfaction by all three subjects.

4.5.4.4 Discussion

A number of discussion points emerge from the analysis of the experimental results and from the experience derived by the observation of the trials.

First, it is quite clear that there is not a guiding mode which is best suited for all subjects. Indeed, while speech guiding mode allows the subjects to align and cross more quickly on average, looking at the majority of the individual subjects (6 out of 11) it is faster to align and cross with one or both the sonification guiding modes. More importantly, subjects distribute their preferences for the best guiding mode almost uniformly among the three solutions.

This result is, at least partially, consequence of the different abilities and leaning rate of the subjects. Indeed, we observed that some subjects have been able to get acquainted

¹⁰Interview was conducted in Italian, we report here and in the following the translation of the subjects' comments.

with one of the two sonifications (or both) in the few minutes of practicing, hence taking actual benefit from them (e.g., in terms of time to align and cross) and preferring these guiding modes with respect to the speech one. Vice-versa, for other subjects the effort required to understand the messages in the mono and stereo guiding mode was so high that they simply preferred to constantly rely on the speech messages they could obtain by tapping on the device. Clearly, these subjects prefer the speech guiding mode.

To judge the applicability of the two sonifications, we should consider that, while they are less intuitive (all subjects believes that at least one of the two sonifications is harder to understand than the speech guiding mode), subjects still express their appreciation for them even after a short practice (11 out of 14 subjects prefer the mono or stereo guiding mode). There is also another consideration: during the test no learning effect emerged. We argue that this is due to the short duration of the trials. We also suspect that there can be a sort of ‘tiring’ effect because the subjects are required to keep concentrated for about 20 minutes of evaluation. Indeed, 5 of the 11 blindfolded subjects required a longer average time to align and cross during the second round than during the first one. We speculate that, since the sonifications are less immediate, they should take larger benefit from the learning effect derived by frequent use of *ZebraCrossing*.

Another aspect is that the tests conducted on the plastic sheeting are more challenging than those on the real environment. Indeed, when testing the app on the plastic sheeting, no haptic or audio clue is available. Vice-versa, when crossing on the road there are a number of hints that can help a person with VIB to orientate with the crossing, including, for example, the sidewalk and traffic noise.

One final remark is related to the unexpected high dispersion of the quantitative results with respect to the mean. Indeed, relative standard deviation is 41%, 47% and 40% for speech, mono and stereo guiding modes, respectively. Combining these data with the experience derived from the observation of the experiments, we can highlight two important facts. First, some users are more confident and hence move faster (e.g., subject 9) while others are more cautious (e.g., subject 5) and tend to move and rotate more slowly. Second, there are some human errors that can lead one subject to have different results in two tests with the same guiding mode. For example, subject 4 completed the two trials with stereo guiding mode in 28s and 104s, respectively: in the second round the subject misinterpreted a message so he believed that the crossing was on his right while actually it was on his left. This caused the align process to take much longer (78s in total) than in the other test with the same guiding mode.

4.6 Zebra crossing detection from satellite and street view imagery

In the following, the *Satellite* Macronavigation module is described. It detects zebra crossings from satellite and street view imagery gathered from on line map services. Currently Google maps and Google street view imagery is used, but the work can be extended to consider other data sources. The developed solution complements the smart-phone based detection described in the previous sections with mainly three goals in mind.

First, to signal to the user, during the navigation, the presence of crossings in proximity and to guide the user towards a crossing still too distant to be detected. Also, the knowledge that a crossing is nearby can be used to confirm that the visual detection by the *Recognizer* step is correct.

Second, to mark the crossings on map services and share the information with the users. The knowledge of the existence of crossings can be useful to people with visual impairments during the route planing.

And third, similarly to the work of Hara et al. [64], as a first step in the design of a crowd-sourcing platform for describing the surroundings of each detected zebra crossing in terms of nearby objects and features. This information can be very helpful to a visually impaired person for finding the crossing and navigate towards it.

Google maps and street view API are usable only through web browsers so parts of the algorithm are javascript snippets executed on web browser while detection components are developed in c++.

The algorithm has 4 steps: Satellite imagery acquisition (Section 4.6.1), satellite detection (Section 4.6.2), street view selection (Section 4.6.3) and street view detection (Section 4.6.4). The evaluation of the *Satellite* module is described in Section 4.6.5

4.6.1 Satellite imagery acquisition

This step identifies areas on which to apply the detection. First, the satellite imagery from Google maps are downloaded given an area. The resolution of the downloaded images is the maximum available for a given area. The images of neighboring areas have an overlap in order to detect also crossings that would otherwise be on the border between two images. The download is limited to street areas (as in Figure 4.30b), which lessens disk usage, avoids possible false positives and reduces execution time.

The step that limits the research to street areas works as follows: First, the algorithm computes the distance d of the image center point from the closest street (information gained from Google maps API). If d is less than half the diagonal of the image i_d , then

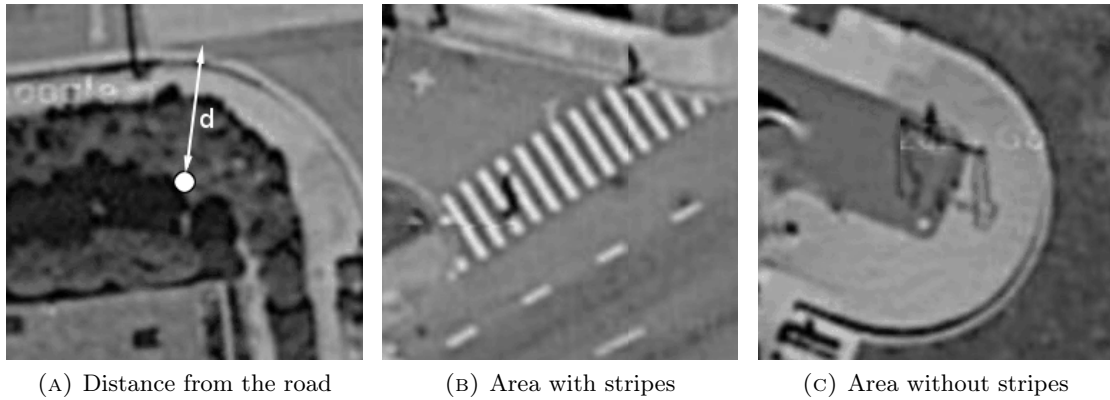


FIGURE 4.30: Downloaded satellite images

the image will have a street in it and thus it is downloaded. Figure 4.30a shows an image of a crossing whose center is at $d < i_d$ from the closest street while Figure 4.30c shows an area discarded by the validation procedure.

4.6.2 Detection on satellite images

In the second step, a cascade classifier recognizes the zebra crossings in an input satellite image (Figure 4.31a). The detection algorithm first detects line segments in the image through a custom version of EDLines line segment detector [97]. The EDLines detection proceeds as follows:

First, the image gradient and magnitude is computed (Figures 4.31b, 4.31c) and local maxima are extracted as anchor points. Then, shown in Figure 4.31d, starting from anchors, nearby points are linked to form pixel chains, which are then fitted to line segments through least squares line fitting (Figure 4.31e) and validated perceptually by Helmholtz principle [99] (Figure 4.31f).

The segments are then clustered according to slope, horizontal and vertical distances in the same way as in Section 4.4.4 and then they are grouped in candidate crossings (Figure 4.31g). In Figure 4.31h, the crossings are validated based on the color difference between white and black stripes, following the procedure described in Section 4.4.5. Finally, the area of the crossing, shown in Figure 4.31i, is computed.

4.6.3 Street view imagery selection

In defining the street view to use for the detection, often the closest street view is not the most suitable one. If the position of the panorama is exactly on the crossing, it is possible that the area of the crossing is covered by the image of the Google car during the capture of the panorama. The imagery interpolates the surrounding data to cover the missing area but, as seen in Figure 4.32a, the results are not usable for the detection.

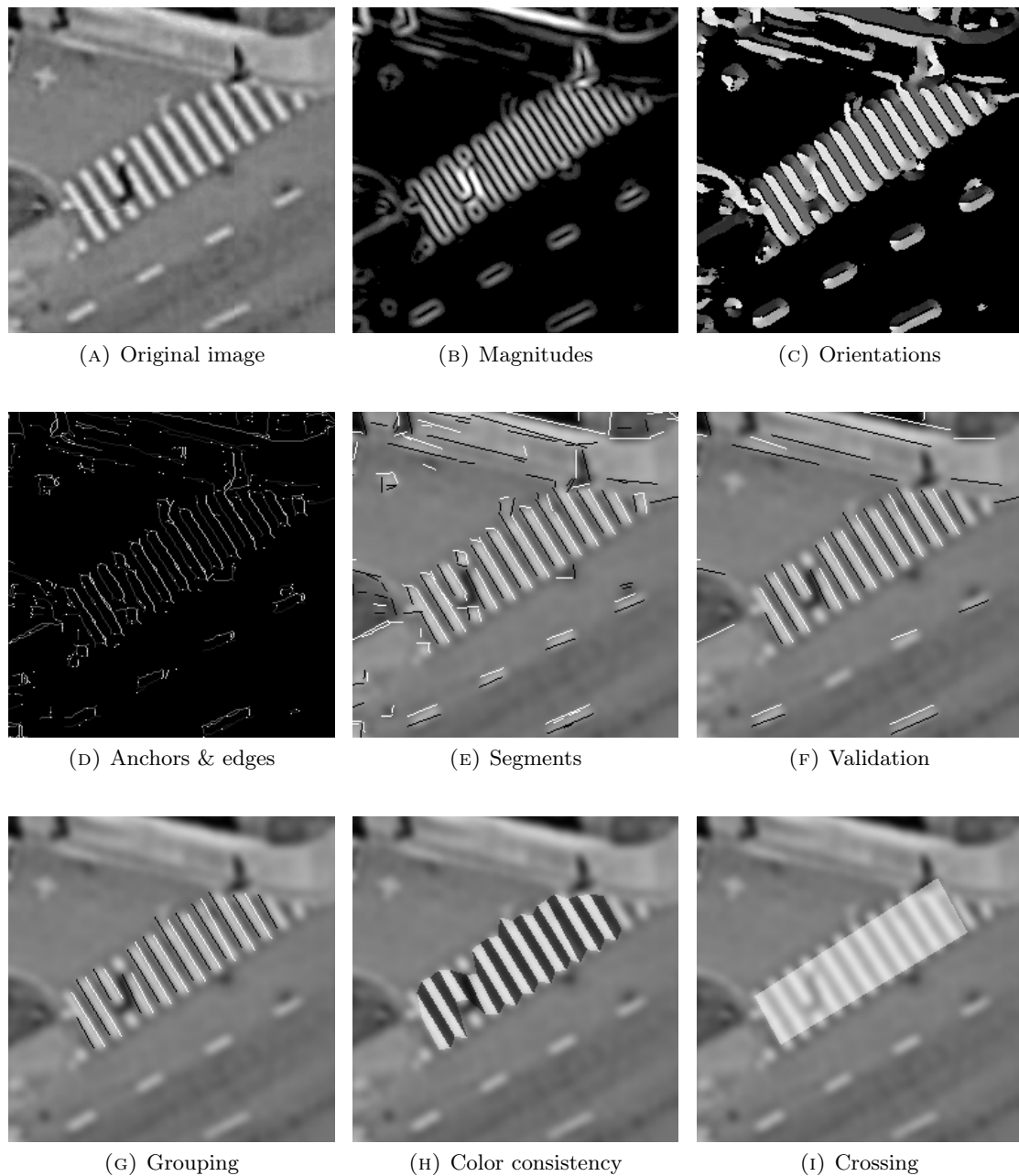


FIGURE 4.31: Steps of the EDLines algorithm and satellite detection

Thus, the solution is to view the crossing from streetview panoramas at a certain distance. The solution uses the Google street view javascript api, through which, given the id of a panorama, it is possible to retrieve the links of the available neighboring panoramas (Figure 4.32b). So both the pictures at the closest panorama to the desired coordinates and at the neighboring panoramas are downloaded for the following detection. The heading of the panorama image is computed from the GPS coordinates of the panorama and the target crossing detected during the previous stage.

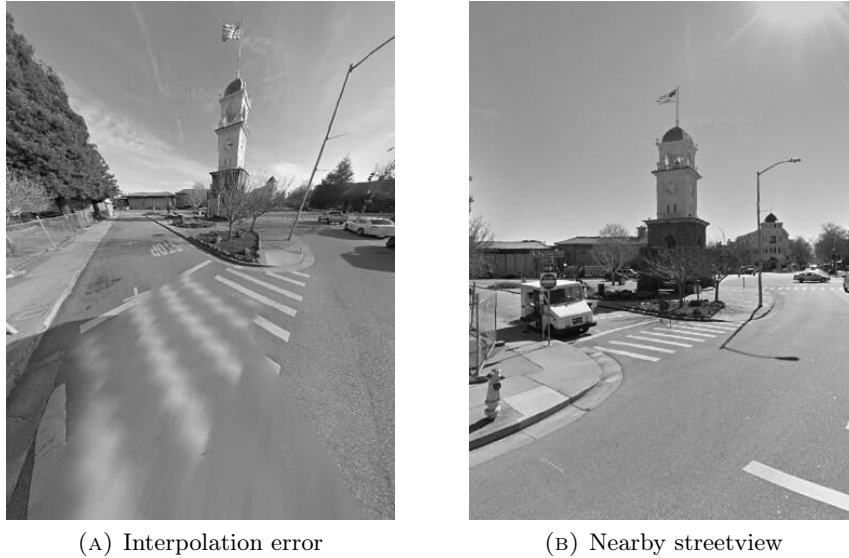


FIGURE 4.32: Closest street view has an interpolation error, a nearby street view doesn't

4.6.4 Detection on street view images

The last step detects the crossings in street view images in the areas suggested by the satellite detection. First, the ground plane in the street view image is reconstructed. Given that the camera parameters do not change, once the reconstruction is computed, it can be used also in other images. The technique used for the metric reconstruction (Liebowitz and Zisserman [94]) requires the knowledge of the plane's vanishing line (the horizon) for the affine rectification and two known angles or proportions in the image for the metric rectification.

The horizon is always horizontal in the street view imagery and it can be positioned on the center of the image by enforcing the pitch parameter of the panorama to 0. For the other constraints it is sufficient to have the knowledge of a square on the ground plane. Indeed, in this case it is possible to choose a square of known dimensions, as in Figure 4.33a, thus knowing that the sides of the squares are equal and that the angle between any of the sides is π gives the constraints required for the rectification and knowing exactly the length of the sides to be $4m$ allows to scale the reconstructed ground plane to known distance proportions. With this information the computation follows straightforwardly the suggested method [94] and Figure 4.33b is obtained.

Once the ground plane is reconstructed, it is possible to map on the image the crossings found on the satellite imagery during the step 2). Indeed, since the coordinates of the street view camera and of the crossing are known, with a simple scaling and translation transformations, it is possible to map the detected crossing on the street view image (Figures 4.34c, 4.34b).

This information can be used to limit the research area inside the street view image in order to reduce the execution time and to improve the precision of the solution by

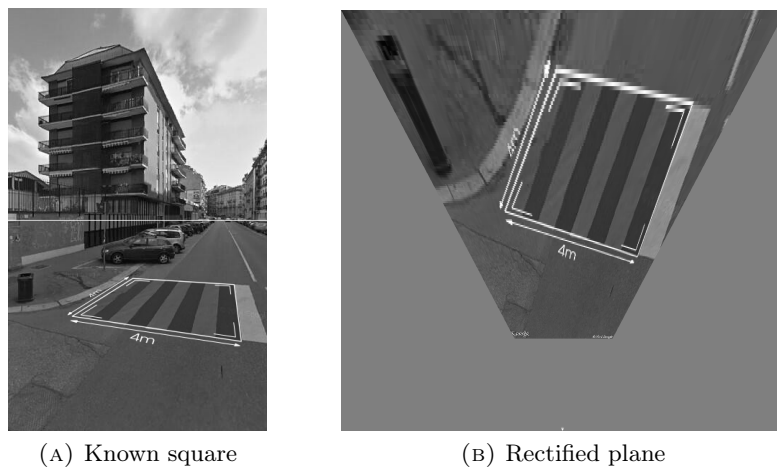


FIGURE 4.33: Rectification is calculated by knowing a square in the street view image

discarding areas that should not contain crossings (Figure 4.34d). Finally, in Figure 4.34e, the crossing is detected in the street view image with a cascade classifier similar to the one used for the mobile device-based detection of zebra crossings, described in Section 4.6.2.

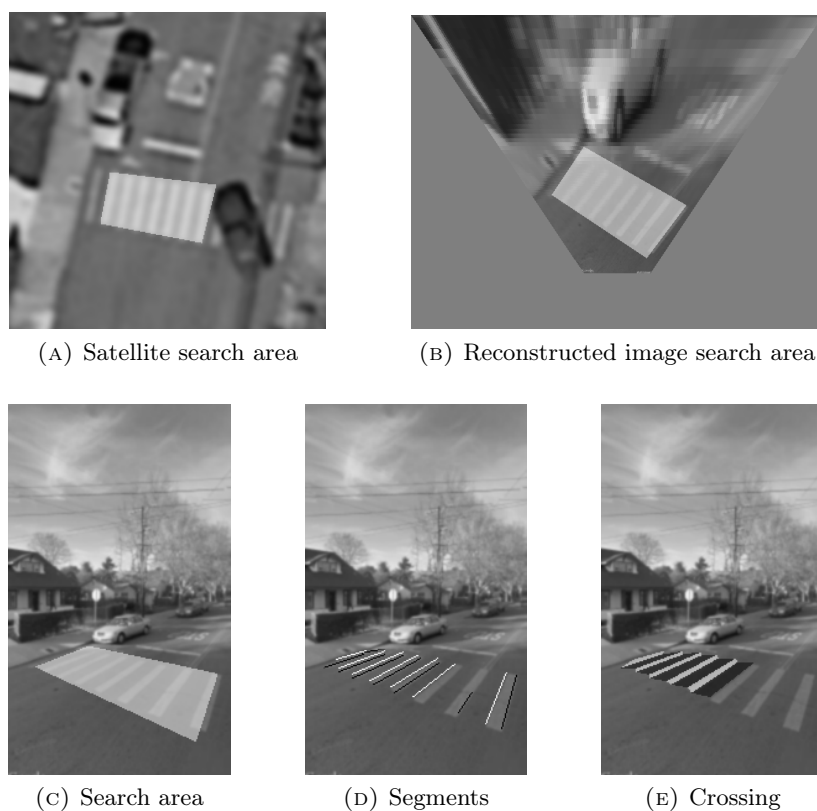


FIGURE 4.34: Steps of the detection on street view imagery

4.6.5 Evaluation

While generally the satellite image detection results in clean, complete crossings, as in Figure 4.31, in some cases the detection can be penalized by damaged or covered stripes.

Figure 4.35 shows some of the common defects in the detection.

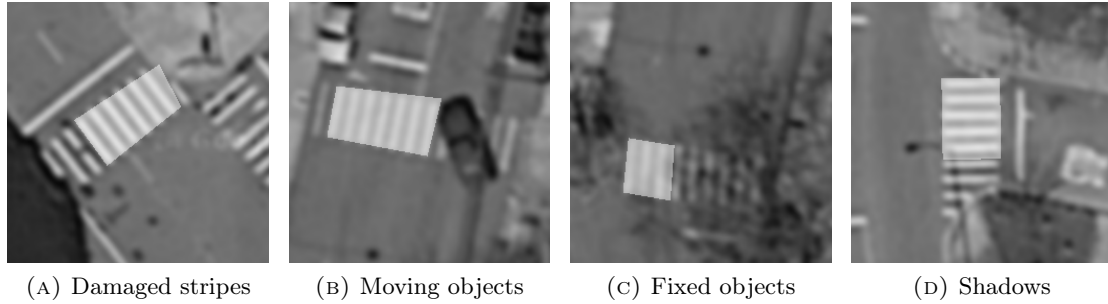


FIGURE 4.35: Errors in detection from satellite images

Another common problem is the incorrect mapping between the crossing's coordinates in the satellite image and in the street view (Figure 4.36). The reason behind this error is that the coordinates of street view panoramas are clipped to the closest street by the google maps API. Thus, there is an error between the actual coordinates of the street view and the one exposed by the API. In 19 panoramas manually positioned at the correct coordinates, the distance from the shown coordinates is of $3m$ in average and up to $6m$.

In this set it is worth noting that out of 19 images, 10 preserve an overlap between the area of the crossing in the image and the estimated area that is over 50%, 5 have an overlap that is less than 50% and 4 have no overlap.

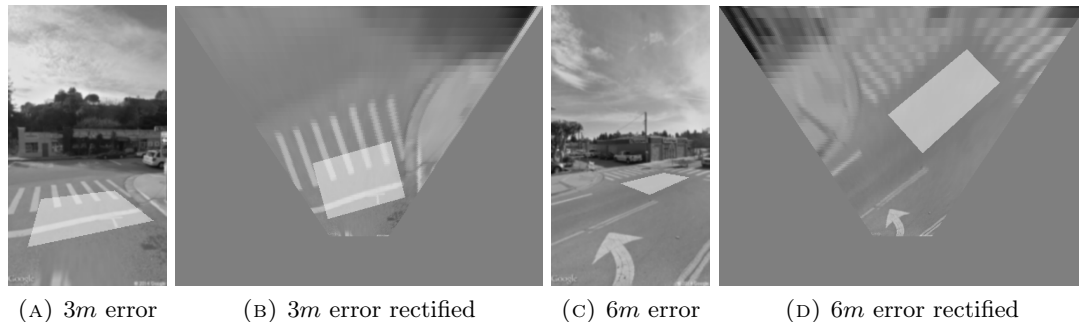
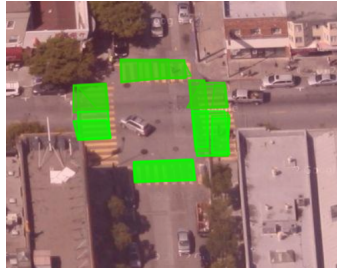


FIGURE 4.36: Errors in positioning of $3m$ and $6m$

A more extensive evaluation of the satellite imagery detection module considered an area of $1.71km^2$ in San Francisco, CA, United States. The imagery corresponding to the area was captured and comprised a dataset of 2350 images¹¹. The satellite detection was tuned in such way to favor the recall score. Thus, of the 138 zebra crossings in the area, 135 were correctly detected (For example, see Figure 4.37a), while 3 crossings were not, yielding a recall score of 0.978. A significant amount of false positives were to be expected. Indeed, there were 45 false positive detections, yielding a precision score of 0.75. However these corresponded to patterns that either were positioned in unlikely locations or were due to the particular satellite point of view. For example, in

¹¹The area and the detection results can be seen at webmind.di.unimi.it/satzebra/

Figure 4.37b a pattern was detected on the roof of a building, while in Figure 4.37c the pattern was found on a window of a house. These errors were easily filtered out by a following street view detection step yielding no actual false positives (i.e., a precision score of 1).



(A) Correct detection



(B) False positive on the roof



(C) False positive on a window of a house

FIGURE 4.37: Examples of correct and erroneous detections of zebra crossings from satellite imagery

Chapter 5

Conclusions and future work

The research presented in this dissertation tackles the issue of spatial understanding and cognitive mapping for persons with visual impairments through the usage of assistive technologies developed on smartphones and mobile devices. In particular, two research fields are explored. Section 5.1 outlines the research conducted in the field of exploration of abstract bi-dimensional spaces, spatial concepts and spatial relations in didactic assistive technologies. Section 5.2, instead, draws the conclusions of the research in the field of assistive technologies used to improve the cognitive mapping during the independent mobility in urban environments.

5.1 Didactic and educational tools

In the field of assistive didactics, the research focused on techniques to convey spatial information to visually impaired students through audio channel. Two different audio-based sensory substitution interaction paradigms were explored. The first one, called “sonification interaction”, is based on sonification techniques and leverages different qualities of the sound, such as loudness and pitch, to convey the spatial information (e.g., distance or size of objects) about the explored space to the user. The second one, dubbed “object-based interaction” adopts more complex, evocative sounds or speech as tags to identify objects on a two-dimensional interface, through an exploration technique akin to screen readers commonly used by persons with visual impairments for the accessibility of touch screens on mobile devices.

The two techniques were evaluated through user-based testing and it was shown that each technique applies to different use cases. The “sonification interaction” approach is more suitable for the recognition of a small number of complex shapes, when the position information has to be relayed with precision and when the resolution of the investigated space is high. Another typical use of sonification is to convey the relative position of

objects in space with respect to the user. Vice-versa, the “object-based interaction” has been successfully adopted when the spatial resolution of the objects is lower, but objects themselves have intrinsically different properties that are easily conveyed vocally or through sound cues.

5.1.1 Elementary grade math and geometry learning

The “object-based interaction” approach has been designed into an interaction technique which has been implemented in *MathMelodies*, a commercial software for teaching mathematics and geometry to elementary school children with and without visual impairments. The solution has been evaluated with three visually impaired and two sighted children which, after roughly 2 minutes of supervised training, were able to complete all the exercises proposed successfully.

Currently, the software is freely available on the AppStore for iOS devices and it has been downloaded more than 700 times in the first two months after the English version has been released. Clearly, a commercial release allows to reach a higher numbers of users and testers than a similar research limited only to the academic field. As a future work we intend to take advantage from the software distribution to the public and automatically collect usage data for the evaluation of the proposed solution.

Another future improvement consists in developing a collaborative system that allows the final user (or the teachers) to directly collaborate in the development of the app content. The definition of this crowdsourcing system can drastically reduce the development costs of the next versions of *MathMelodies* and ease the scalability of this solution.

5.1.2 Function graph exploration

The “sonification interaction” approach has been developed in the context of a tablet device prototype that allows visually impaired students to explore function graphs. In particular three interaction modes based on sonification have been designed, two of which use proprioception by taking benefit from the direct interaction with the tablet touchscreen.

The experimental evaluation was conducted with 7 visually impaired users and compares the proposed interaction paradigm with alternative approaches for the function graph exploration: tactile paper and another sonification software, “AGC”. The evaluation shows that the proposed solution, after only 5 minutes of training, allows the users to have a much better understanding of the the function graph than existing solutions. This is the case even with tactile paper, with which the users were all well acquainted.

As a future work we intend to focus on each sonification technique proposed, compare them to other solutions and identify the approach that best suites each exploration

mode. We also plan to engineer and distribute the solution as a commercial application. Experience with other didactic software show how this would allow a larger distribution of the solution, which in turn can have positive effects on future research. Indeed, by remotely collecting usage data, it could be possible to evaluate the solution with a much larger number of users.

5.2 Unassisted urban navigation

The cognitive mapping of the surrounding space during the independent mobility of persons with visual impairments faces important issues. First, purely visual elements, such as writings and signs, cannot be accessed through haptic or acoustic means. Thus, these cues are not considered by a visually impaired person during way-finding. Also, objects outside the haptic vicinity of the visually impaired person (that is the area that can be explored by touch or with the white cane) are also ignored during exploration and can only be accessed after approaching them. Additionally, in case of mobility in urban environments, approaching distant objects is not a trivial task and, in areas with vehicular traffic, poses danger to the visually impaired.

The first goal of the research therefore consisted in identifying distant, visual cues in order to assist the user in the correct cognitive mapping of the surroundings and navigation. In particular, the issue of zebra pedestrian crossing detection was tackled. The approach leveraged the satellite imagery and the camera and spatial sensors present on smartphones and mobile devices to identify the stripes of the zebra crossing visually.

The second goal focused on conveying the information related to the detected crossings in such a way to guide the user in a safe, efficient and unassisted way. Different speech and sonification based interaction paradigms were designed, developed and evaluated for this goal. The user evaluation shows that the techniques proposed are indeed feasible for assisting a visually impaired user in autonomous urban way-finding.

5.2.1 Zebra crossing detection

For the purpose of detecting zebra pedestrian crossings, three different detection techniques were designed and implemented. The first technique consists in a purely computer vision-based detection on images captured by a mobile device's camera. An alternative approach that also leverages accelerometer and gyroscope data was engineered and implemented.

The two approaches were thoroughly evaluated on a dataset of 4015 frames consisting in camera images (1877 with and 2138 without crossings), accelerometer and gyroscope data. The evaluation shows how the proposed solutions compare favorably with other

solutions available in literature. The evaluation also highlighted how the inclusion of accelerometer and gyroscope data both enhances the efficiency and the quality of the detection. In particular, a significant improvement in recall was observed (from 0.7 to 0.9). Also, the execution time of the procedure was drastically improved (from 74ms to 16ms).

A third detection technique, leveraging satellite and street view imagery for the zebra crossings detection, was engineered, developed and evaluated. This approach is orthogonal to the other two techniques as it does not allow to detect the crossings in the user's immediate vicinity (under 20m). Instead, it is leveraged to notify the user of the distant crossings (20m to 100m) if the user is interested in crossing the street and no nearby crossings are available. This solution was evaluated with a dataset of 2350 satellite images of a 1.71km² area in San Francisco, CA, United States yielding promising results.

As a future work, the detection of other visual cues, such as traffic lights (its position and its current color), will be investigated. Additionally, the solutions proposed will be integrated in a comprehensive urban navigation tool for visually impaired users.

5.2.2 Speech and sonification interaction

For the purpose of guiding the users towards and over zebra pedestrian crossings, three interaction paradigms were designed: one speech based and two sonification based interaction techniques, which were evaluated with visually impaired and blindfolded sighted subjects. Experimental results show that *ZebraCrossing* prototype can effectively guide people with VIB in road crossing with any of the three auditory guiding modes. Most subjects (75%) declare to prefer one of the two sonification modes with respect to the speech mode. This result supports the usefulness of the two sonifications, also considering that subjects prefer the sonifications despite being less immediate to use. At the same time, results show that there is not a single guiding mode that is the best for every user. Indeed, some users prefer the immediate understanding of the information received with speech output while others favor the minimal intrusiveness of the sonification approaches. Finally, while some users are comfortable with bone conducting headphones used during the tests, speech and mono sonification can be also used by individuals are not willing to wear headphones while walking autonomously.

Another promising guiding mode is being currently investigated. Each new message is conveyed with a speech message or distinct sound; additionally, the quantity associated with the message is conveyed to the user through sonification. So, for example, it would be possible to have a 'rotate right' message followed by a sound that informs the users about the quantity of rotation and that changes dynamically while the user rotates. The main difference with respect to the sonifications presented in this contribution is that

there is only one sonification for all messages, and this should simplify the process for each user to get acquainted with the new guiding mode.

The results presented in this contribution open several directions in research and development. The three guiding modes presented in this paper find their direct application in commercial software to support road crossing. Once the software is available on the market, it will be possible to remotely collect usage statistics from a large population. For example it will be possible to monitor how many people use each guiding mode, to collect statistical data about crossing performance or how many people use the application with headphones. Even more significant, remote monitoring of real usage will make it possible to collect data about long term learning effect, which is a very important aspect that was not practically possible to take into account in this contribution. In turn, information collected remotely will guide the design of future sonifications.

Appendix A

Horizon computation from gravity acceleration data

The horizon is the ground plane vanishing line and, as such, it is used in two ways by the zebra crossing detection techniques described in Chapter 4.

First, during the segment detection on the images with perspective distorted ground plane (i.e. From smartphone or street view), the horizon is used for speeding up the computation by ignoring the area above it since, clearly, no zebra crossings will ever be found there.

Second, in Section 4.3.2, the horizon is also used during the ground plane reconstruction step for computing the rectification matrix.

To compute the horizon line equation inside the image we need the slope θ of the horizon line and a point through which the horizon passes. To obtain both, our technique requires the device's attitude in space, the image size (width i_w and height i_h), the image principal point p and the half horizontal and vertical angles of view, β_x and β_y . While the image size is known and the attitude is derived from the gravity acceleration measurements, the other three values can be obtained from the camera sensor size (sensor width S_w and height S_h) and the camera's intrinsic parameters that are fixed for each device and can be obtained from the camera producer or through camera calibration. Table A.1 shows the matrix of intrinsic parameters.

Note that, for the camera we used in our experiments, it holds that the skew coefficient γ is zero and that the focal length f is the same on both axes (i.e., $f = f_x = f_y$). From these values, we can easily obtain $p = [u_0, v_0]$ (where u_0 and v_0 are specified in the intrinsic parameters) and $\beta_x = \text{atan}(S_w/2f)$ and $\beta_y = \text{atan}(S_h/2f)$.

$$\begin{vmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{vmatrix}$$

TABLE A.1: Intrinsic camera parameters matrix.

The gravity acceleration components a_x and a_y measure the roll inclination of the device (see Figure 4.9b) with respect to the ground plane. This value corresponds to the inclination of the horizon on the device's screen. Consequently: $\theta = \text{atan2}(a_x, -a_y)$.

The pitch inclination of the device (see Figure 4.9b) with respect to the ground plane, captured by the gravity acceleration component a_z , is used to find the point through which the horizon passes. In case of a perfectly vertically held device, the horizon passes through the point:

$$hp = \left(u_0, \frac{i_h \cdot a_z}{2 \cdot \sin(\beta_y)} + v_0 \right)$$

For example, considering the principal point to be roughly at the center of the image, when $a_z = 0$ the horizon is at the center of the screen. Similarly, when $a_z = \sin(\beta_y)$ (i.e: the inclination is the same as the field of view of the camera) the horizon is at the bottom of the screen and when $a_z = -\sin(\beta_y)$ it is at the top of the screen.

Generalizing for any orientation of the device we obtain:

$$hp = \left[\frac{\sin(\theta) \cdot i_w \cdot a_z}{2 \cdot \sin(\beta_x)} + u_0, \frac{\cos(\theta) \cdot i_h \cdot a_z}{2 \cdot \sin(\beta_y)} + v_0 \right]$$

Given the inclination θ and the point hp , the horizon line equation is:

$$hl = \sin(\theta) \cdot x + \cos(\theta) \cdot y - \sin(\theta) \cdot hp_x - \cos(\theta) \cdot hp_y$$

Appendix B

Comparison of line segment detection methods

Three segment detection algorithms were implemented, aiming for a fast and accurate detection: The well known Hough transform based probabilistic line segment detector (Appendix B.1), LSD (Appendix B.2) and EDLines (Appendix B.3).

A detailed evaluation of the implementations of the three detectors can be found in Appendix B.4.

B.1 Hough probabilistic line segment detector

The Hough line segment detection is divided in two steps. In the first step, described in Section B.1.1, the edge is computed with a custom version of the canny edge detection. In the last step (Section B.1.2, the Hough probabilistic line segment detector is applied on the edge map.

As a preprocessing step, the input picture is smoothed for filtering out imperfections in the crossing's paint or shape.

B.1.1 Custom canny edge detection

The OpenCV implementation of the Canny edge detection is used: non-maxima suppression and hysteresis thresholding are applied on a sum-image of horizontal and vertical Sobel edge detections. However, the adopted implementation differs from the original Canny algorithm: it differentiates between the orientation of the intensity gradient in order to distinguish the starting of a stripe from the ending of a stripe.

This is obtained by using the first order sobel operator for differentiating between the increasing and the decreasing gradients instead of the second order sobel. The canny edge mapping is then applied on high and low values of the sobel image generating two different images based on the gradient orientation.

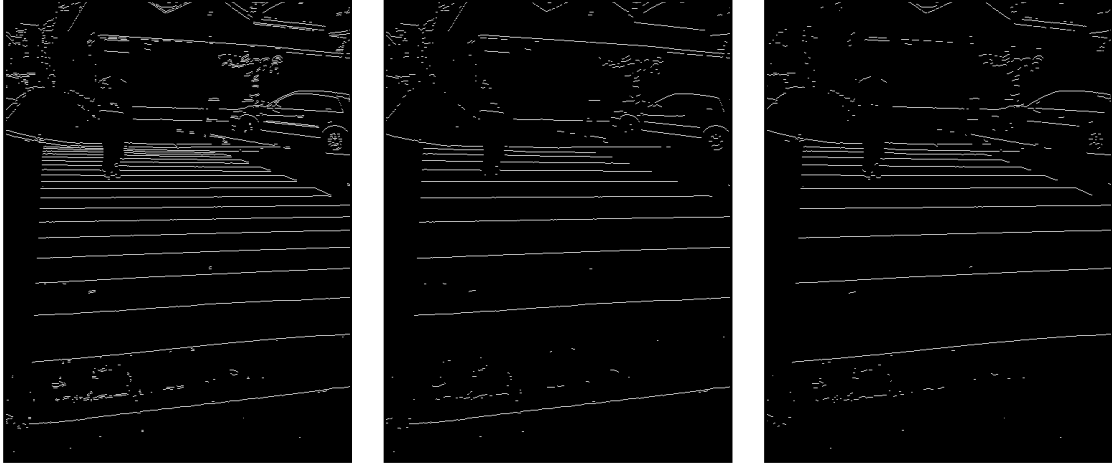


FIGURE B.1: Custom canny edge detection isolates increasing and decreasing gradients

B.1.2 Hough line segment detection

The Hough transform-based line segment detection [100] conducts the detection through the use of a voting procedure executed for each pixel of the input binary image.

The voting procedure is conducted in a parameter space with the number of dimensions equal to the number of unknown parameters of the feature to be detected. In case of lines, given the slope-intercept equation form

$$y = \theta x + \rho$$

the unknown parameters are the slope θ (Theta) and the y-intercept ρ (Rho) of the line. Each point on the input image specifies the most probable parameters (or a set of the most probable parameters) of the line it belongs to by exploring its neighboring points. Different points casting the same parameter vote possibly lay on the same line. During the hough step, the parameter space is computed in range $\theta \in (-\pi, \pi]$ instead of $[0, \pi)$ so to accomodate the distinction between line orientations introduced in the previous section. The group of segments that were obtained from the increasing gradient cast votes in $[0, \pi)$ range while the decreasing gradient segments vote in $[\pi, 0)$.

The local minima and maxima in the parameter space (high number of votes for the same parameter pairs), exceeding a specified threshold, are extracted from the image as lines as per Figure B.2. Afterwards, the extracted lines are cut into line segments corresponding to the edges of the edge map with the usage of a minimum segment length and a maximum gap threshold.

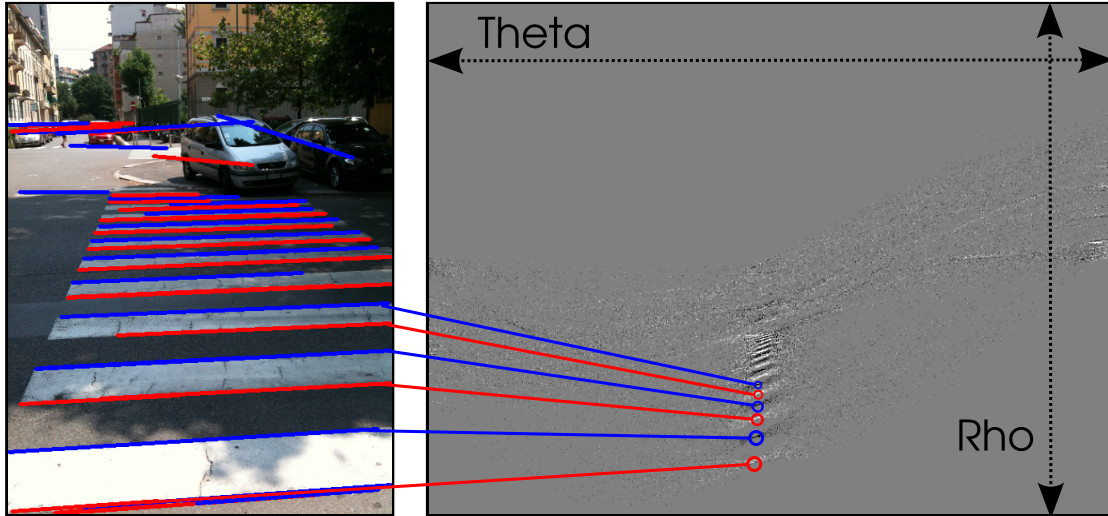


FIGURE B.2: Hough parameter space peaks correspond to the detected line segments.

B.2 LSD

A linear-time (with respect to the number of pixels) line segment detection algorithm named “LSD: A Line Segment Detector”, using only the gradient orientation (without the gradient magnitude information), has been studied by Grompone von Gioi et al. [92]. The algorithm does not depend on an initial edge detection and requires no parameter tuning. Regarding the quality of the detection, Grompone von Gioi et al. state that on average there is one false positive segment per image and that the discarded line segments are likely to appear in noise and therefore can be discarded safely.

The detector is structured in 3 stages: partitioning of the picture into line-support regions (Section B.2.1), individuation of the line segments (Section B.2.2) and validation of the found line segments. Since conceptually, the validation is identical to the one used in EDLines algorithm, it will be described along with the definition of the EDLines algorithm (Section B.3.3).

B.2.1 Line support regions

The first step detects the gradient orientation for each pixel of the picture and groups adjacent areas that share the same gradient orientation up to a certain tolerance. This process is executed by starting from pixels with strong edge intensity and greedily connects close regions (Figure B.3).

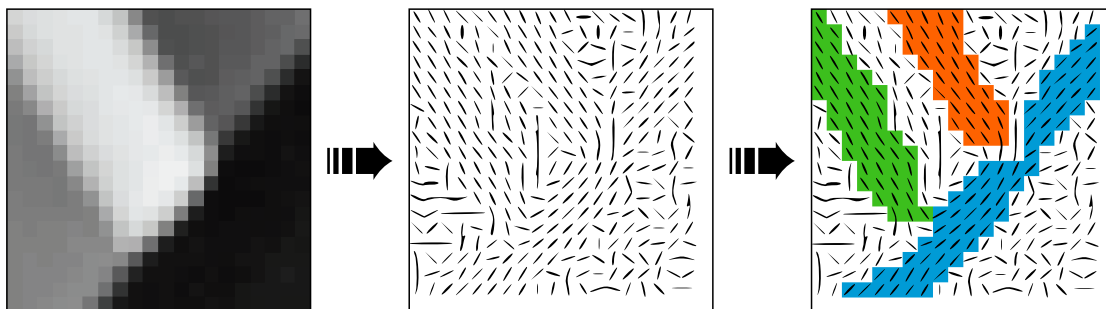


FIGURE B.3: Gradient computation, support regions extraction

B.2.2 Segment identification

Afterwards, the detected regions are approximated to rectangles according to the global gradient direction and distribution of the region. Finally, the obtained rectangles are extracted as line segments (Figure B.4).

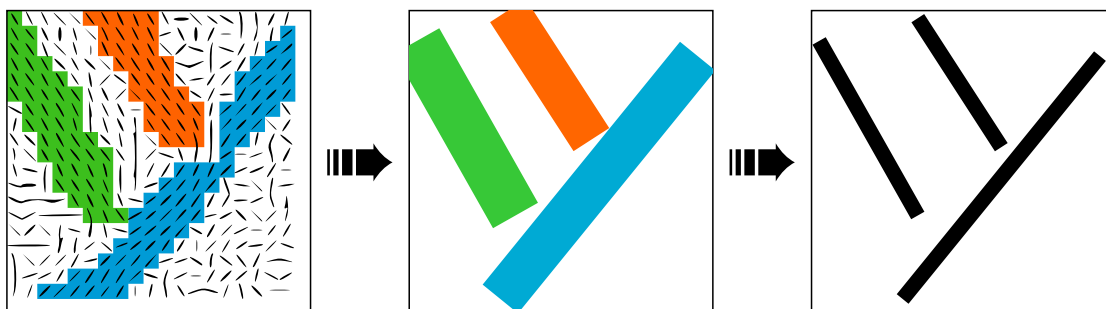


FIGURE B.4: Rectangle definition, segment extraction

B.3 EDLines

The implementation of the EDLines algorithm adopted in Section 4.3.3, follows the description by Akinlar and Topal [97] but also introduces new optimizations. It is divided in three steps. In the first step, described in Section B.3.1, the edge map is computed as chains of pixels. The chains are then processed to extract straight line segments (Section B.3.2). Finally, Section B.3.3 describes the segments validation based on the Helmholtz principle.

B.3.1 Edge Drawing

The Edge drawing step computes an array of chains of pixels starting from the input image. Differently from previous pixel chain based methods [101], which compute the edge map of the image and extract the chains of pixels as two steps, EDLines computes the edges directly as pixel chains, resulting in clean and contiguous edges. The algorithm,



FIGURE B.5: The steps of the EDLines algorithm

called “Edge Drawing” [102], can execute most operations in one pass, thus having lower computation time. The algorithm is implemented as following steps:

0 - preprocessing. A Gaussian filter is applied to the input image for noise suppression. Differently from [97], during the parameter tuning for the zebra crossing detection, the best results were obtained with a kernel size $k = 9$ and $\sigma = 1$

1 - gradient magnitude and direction computation. For each pixel (x, y) of the image, having the intensity $I(x, y)$, the gradient direction $d(x, y)$ (Figure B.5c) and magnitude $m(x, y)$ (Figure B.5b) are calculated. For the gradient magnitude, the same

operator as in the original definition of the algorithm is used [97]:

$$m(x, y) = \sqrt{m_x(x, y)^2 + m_y(x, y)^2}$$

Where

$$m_x(x, y) = \frac{I(x+1, y) - I(x, y) + I(x+1, y+1) - I(x, y+1)}{2}$$

$$m_y(x, y) = \frac{I(x, y+1) - I(x, y) + I(x+1, y+1) - I(x+1, y)}{2}$$

Instead, for the gradient direction, the gradient is computed as:

$$d(x, y) = \text{atan2}(m_x(x, y), m_y(x, y))$$

where `atan2` is the two argument arctangent function, available in the standard C `math.h` library. This way, the gradient directions range in $[0, 2\pi)$ instead of $[0, \pi)$, thus distinguishing between the gradients having the same direction and opposite orientations. This allows to differentiate when the color changes from dark to light (e.g., from dark stripes to light) and vice-versa. Also, given that the `atan2` (and `atan`) operations are computationally expensive, a 16-value LUT approximation has also been implemented. As shown in Section B.4, the speed increase is significant and the detection does not deteriorate.

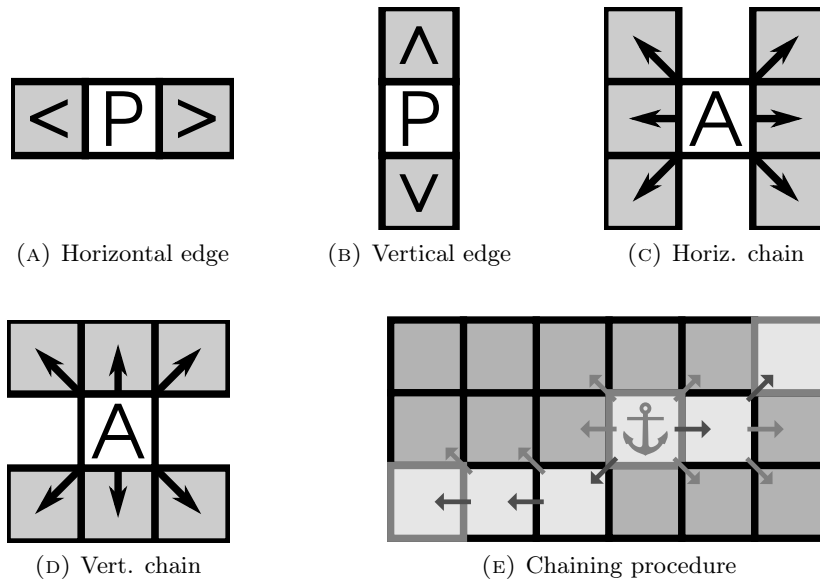


FIGURE B.6: Edges, chains and chaining procedure

2 - anchors computation. Local maxima in the gradient magnitude map are marked as anchors. First, the edge orientation is calculated. If $m_x(x, y) > m_y(x, y)$ then the edge is considered vertical, otherwise it is horizontal. For a point $P = (x, y)$ on a horizontal edge the neighbors are $(x, y - 1)$ and $(x, y + 1)$ (Figure B.6a), whereas on a vertical edge the neighbors are $(x + 1, y)$ and $(x - 1, y)$ (Figure B.6b). Then, if the examined point has a gradient stronger by a threshold than both neighbor points, it is considered an anchor.

As an optimization, the anchors are saved as an array. This way, when computing the edges, the anchor array is iterated, instead of scanning the whole image again.

3 - chain linking. Starting from the anchors, the algorithm searches neighbor pixels with the same edge orientation, and links the one with the strongest gradient magnitude. For example, in Figure B.6c, the neighbor points are searched horizontally from the anchor *A*, while in Figure B.6d they are searched vertically. The new pixel becomes the new anchor and the search continues from there. As seen in Figure B.6e, starting from an anchor point, the procedure iterates until there are no valid pixels on the path or the path reaches another path or image border. In Figure B.5d the resulting edges and their initiating anchors (a small subset of initial anchors) are shown.

The operations 0) to 2) are applied on all the pixels of the image and can actually be executed efficiently in one pass, while the operation 3) is executed only on a small subset of points. Thus, the computation time is linear in the number of pixels of the image and, as shown in Section B.4, lower than in the other examined algorithms.

B.3.2 Line fitting

The line fitting algorithm computes line segments from chains of pixels obtained in the previous step (See Figure B.5e). The original EDLines algorithm uses the least squares line fitting technique (Figure B.7a). This approach first computes a line such that the sum of the squares of the vertical distances of the points to the line is minimized. In this implementation, instead, the orthogonal regression method is used (Figure B.7b).

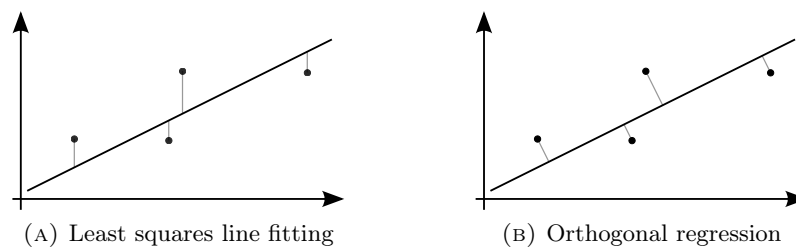


FIGURE B.7: Least squares and orthogonal line fitting procedures

This approach minimizes the sums of the squares of the orthogonal distances of the points to the line. Thus, it is suitable also for the fitting of vertical line segments, where the vertical distances are imprecise or unavailable. Afterwards the line is cut to obtain the line segment.

Precisely, from the initial chain of pixels, a small initial valid line segment is searched and then expanded until the orthogonal regression error is lower than a threshold. In the original algorithm the expansion is done incrementally one pixel at a time. Instead, this implementation can also expand the line segment dichotomically in a greedy fashion.

That is, given a chain of pixels (Figure B.8a), the algorithm tries to obtain a good initial line segment starting from the beginning of the chain. If the starting pixels are not aligned (Figure B.8b), they are discarded until an initial strong line segment is found (Figure B.8c). The algorithm tries to match as much as possible of the remaining chain immediately (Figure B.8d). If the expansion fails, the number of points for the expansion is halved and the expansion is tried again (Figure B.8e). The procedure iterates until the expansion fails with a step of 1 pixel and the final good line segment is returned (Figure B.8f). This way, small imperfections, which might halt the expansion in the beginning, have a lower impact. Thus, longer segments are created while the cost of the operation is small with respect to the total execution time of the algorithm.

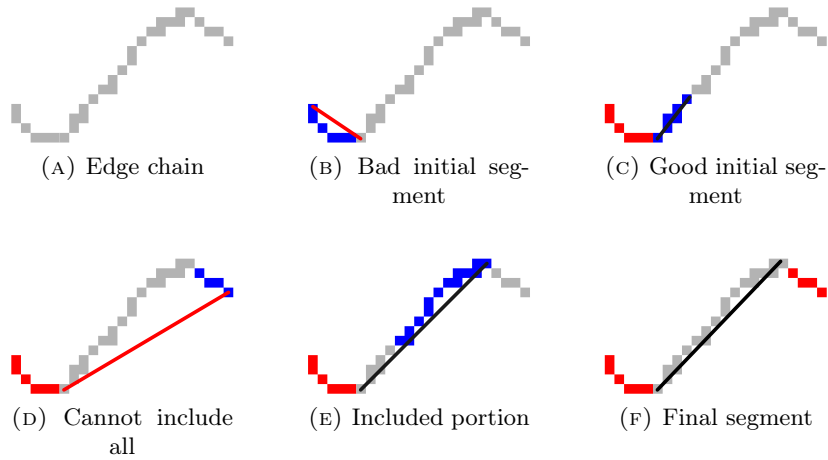


FIGURE B.8: Greedy expansion

B.3.3 Validation

The validation procedure follows the technique suggested by Desolneux et al. [99]. In this technique, line segment is considered valid if its probability in a random noise model is low. For this, the concept of the “Number of False Alarms” (NFA) is used. A segment of length n and having k aligned points is valid only if $NFA(n, k) \leq \epsilon$ where ϵ defines the maximum number of false detections of the given segments in the image. This value, in accordance to Desolneux et al. [99], is set to 1, which corresponds to one false detection per image. NFA is computed as follows:

$$NFA(n, k) = N^4 \cdot \sum_{i=k}^n \binom{n}{i} p^i (1-p)^{n-i}$$

where N^4 defines the number of possible segments in a $N \times N$ image, and the probability of two points being perceptually aligned $p = 1/8 = 0.125$, since, perceptually, two points are aligned if their angles are within $\pi/8$ of each other.

Akinlar and Topal [97] use this formula also to compute the minimum length a segment in an image must have for being perceptually valid, even if all its points are aligned.

Indeed, by setting $n = k$, it is possible to find n given N :

$$n \geq -4\log(N)/\log(p)$$

Once n has been computed, it is possible to discard segments shorter than n , even without computing the alignment of their pixels. In Figure B.5f, valid segments are drawn.

B.4 Comparison and evaluation

Figure B.9a shows the execution times and Figure B.9b the average number of segments detected per image on the ZebraRecognizerTestSet¹ run on an Intel i5 2450M 2-core (4-thread) machine with Gentoo linux 3.14.14 kernel. The execution times of this implementation of the EDLines algorithm compare favorably to the openCV implementation of the Hough probabilistic line segment detector and even more so with respect to the previously adopted LSD algorithm. EDLines is 2 to 3 times faster than LSD at any resolution, while the difference with Hough is constantly from 2ms to 8ms (avg. 4ms).

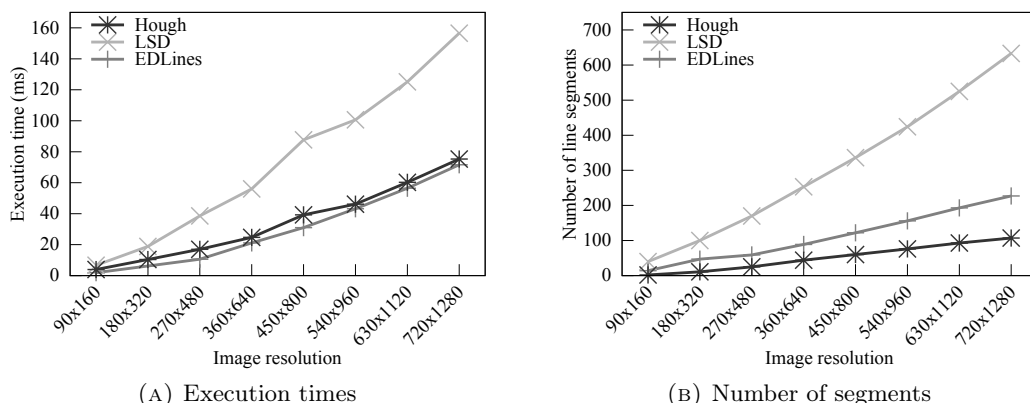


FIGURE B.9: Comparison between EDLines, LSD and Hough

At 320×180 resolution, most of the computation time (73%, 4.39ms) is spent in gradients and anchors computation (See Figure B.10). Pixel chaining requires 18%, or 1.16ms while orthogonal line fitting takes 7% of the execution time, corresponding to 0.46ms. The validation step weights only for 3.6% on the computation time (0.22ms). Figure B.10 also shows the impact of the optimizations on the execution time of the detector. Specifically, the use of the anchor array instead of the anchor image and the use of the LUT table for the atan2 computation save about 1.1ms each.

For all three algorithms the number of detected segments is linearly proportional to the image resolution (Figure B.9b). In particular, EDLines detects half the segments that LSD detects at a given resolution. There are two reasons for this. 1) EDLines does not detect very short segments, often detected by LSD (Figure B.11) and 2) the greedy line

¹<http://webmind.di.unimi.it/ZebraRecognizerTestSet/>

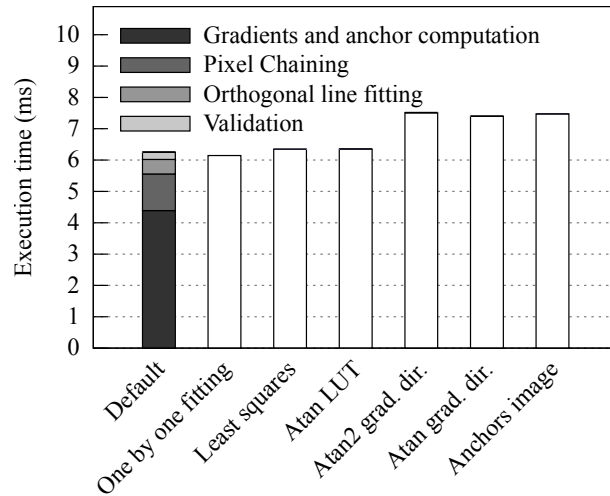


FIGURE B.10: Cost of different operations of the segment detection

fitting of this implementation often identifies single longer segments rather than many smaller ones (Figure B.13b). Hough, having a global notion of the structure of the image, finds the intersecting segments easily and generally tends to produce 2 to 3 times less line segments than EDLines and 5 to 6 times less segments than LSD.



(A) Image with many intersecting lines in foreground



(B) Hough detects foreground segments better but not the background structures



(C) Lsd finds small portions of foreground segments and the background structures



(D) EDLines finds none of the foreground structure but finds some background segments

FIGURE B.11: The effect of foreground intersecting lines on Hough, Lsd and EDLines

On the other hand, the Hough based detector often detects many false positive segments

and fails to grasp the structure of the underlying figure. In Figure B.12 an anisotropic structure such as a tree causes the Hough detector to identify many false positive segments, whereas EDLines detects much less even with respect to LSD.

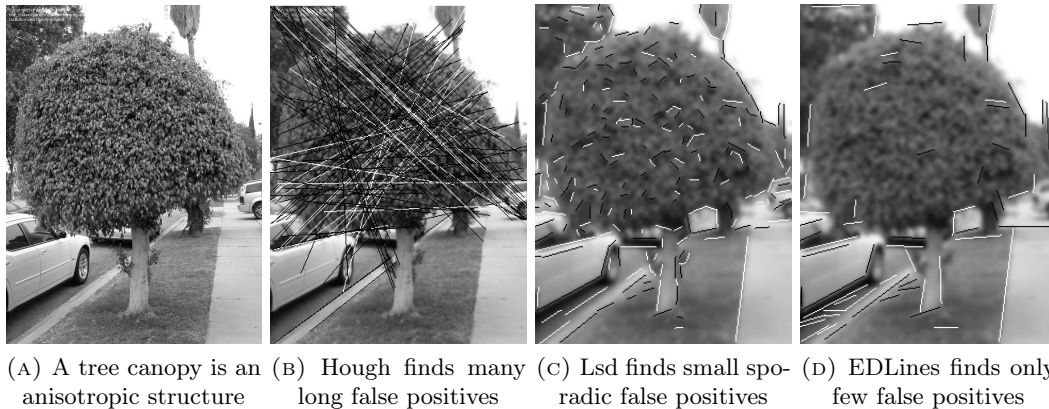


FIGURE B.12: Anisotropic structures cause some false positives in LSD and many in Hough

As seen in Figure B.10, the cost of the Orthogonal regression and Greedy chaining is negligible, while solving two problems. The Orthogonal regression (Figure B.13d) detects vertical segments lost during Least squares line fitting (Figure B.13c), while the Greedy chaining detects as one (Figure B.13b) otherwise fragmented segments (Figure B.13a).

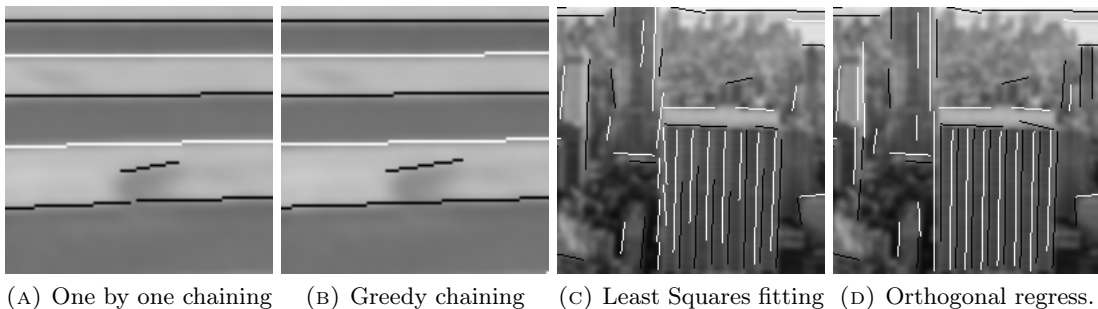


FIGURE B.13: Effects of Greedy chaining and Orthogonal regression optimizations

The substitution of the atan2 function with a LUT approximation achieved an enhancement in the computational costs (Figure B.10) without significant deterioration in the detection. Indeed, most of the segments are detected with both methods. For example, out of the 36 line segments detected by the atan2 version (Figure B.14a) and 45 line segments found by the LUT version (Figure B.14b), the 30 longest segments correspond in both images. The same holds for the other examples in Figure B.14.

An interesting evaluation method, proposed by Zhang et al. [103] will be considered as a future work. This approach compares EDLines, LSD and other detectors according to the idea of the repeatability, defined as the capability of detecting the same line segments in different conditions of the same scene (e.g., for better stereo matching). Namely the conditions examined are the change of view point, scale, blur, lighting and lossy compression. For the target use case, that is detection of a crossing while the user



FIGURE B.14: Effects of the LUT atan2 approximation

is on the move, this measure is indeed interesting and it is worth noting that, in the author's implementation, in all cases LSD and EDLines performed better than other detectors and similarly between them.

Bibliography

- [1] McGookin, D., Brewster, S., and Jiang, W. Investigating touchscreen accessibility for people with visual impairments. In *Proceedings of Nordic Conference on Human Computer Interaction*. ACM, 2008.
- [2] Robert M Kitchin, Mark Blades, and Reginald G Golledge. Understanding spatial concepts at the geographic scale without the use of vision. *Progress in Human Geography*, 1997.
- [3] RM Kitchin and RD Jacobson. Techniques to collect and analyze the cognitive map knowledge of persons with visual impairment or blindness: Issues of validity. *Journal of Visual Impairment and Blindness*, 1997.
- [4] Simon Ungar. Cognitive mapping without visual experience. *Cognitive mapping: past, present, and future*, 2000.
- [5] R Dan Jacobson. Cognitive mapping without sight: Four preliminary studies of spatial learning. *Journal of Environmental Psychology*, 1998.
- [6] Susanna Millar. *Understanding and representing space: Theory and evidence from studies with blind and sighted children*. Clarendon Press/Oxford University Press, 1994.
- [7] Passini, R. and Proulx, G. Way finding without vision: an experiment with congenitally blind people. In *Environment and behavior*. Kluwer, 1988.
- [8] Steven M Casey. Cognitive mapping by the blind. *Journal of Visual Impairment & Blindness*, 1978.
- [9] Simon Ungar, Mark Blades, and Christopher Spencer. Visually impaired children's strategies for memorising a map. *British Journal of Visual Impairment*, 1995.
- [10] Linda Pring. Psychological characteristics of children with visual impairments: learning, memory and imagery. *British Journal of Visual Impairment*, 2008.
- [11] Thomas Dick and Evelyn Kubiak. Issues and aids for teaching mathematics to the blind. *The Mathematics Teacher*, 1997.

- [12] John A. Gardner. Access by blind students and professionals to mainstream math and science. In *Proceedings of the 8th International Conference on Computers Helping People with Special Needs*. Springer, 2002.
- [13] Thomas Westin, Kevin Bierre, Dimitris Gramenos, and Michelle Hinn. Advances in game accessibility from 2005 to 2010. In *Proceedings of the 6th International Conference on Universal Access in Human-computer Interaction*. Springer, 2011.
- [14] Timothy Roden and Ian Parberry. Designing a narrative-based audio only 3d game engine. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*. ACM, 2005.
- [15] Daniel Miller, Aaron Parecki, and Sarah A. Douglas. Finger dance: A sound game for blind people. In *Proceedings of the 9th International Conference on Computers and Accessibility*. ACM, 2007.
- [16] José Ángel Vallejo-Pinto, Javier Torrente, Baltasar Fernández-Manjón, and Manuel Ortega-Moral. Applying sonification to improve accessibility of point-and-click computer games for people with limited vision. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. British, 2011.
- [17] Linda Bussell. Touch tiles: Elementary geometry software with a haptic and auditory interface for visually impaired children. In *EuroHaptics 2003*, 2003.
- [18] Rameshsharma Ramloll, Wai Yu, Stephen Brewster, Beate Riedel, Mike Burton, and Gisela Dimigen. Constructing sonified haptic line graphs for the blind student: First steps. In *Proceedings of the 4th International Conference on Assistive Technologies*. ACM, 2000.
- [19] René Gutschmidt, Maria Schiewe, Francis Zinke, and Helmut Jürgensen. Haptic emulation of games: Haptic sudoku for the blind. In *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. ACM, 2010.
- [20] Roope Raisamo, Saija Patomäki, Matias Hasu, and Virpi Pasto. Design and evaluation of a tactile memory game for visually impaired children. *Interact. Comput.*, 2007.
- [21] Ravi Kuber, Matthew Tretter, and Emma Murphy. Developing and evaluating a non-visual memory game. In *Proceedings of the 13th International Conference on Human-computer Interaction*. Springer, 2011.
- [22] Bei Yuan and Eelke Folmer. Blind hero: Enabling guitar hero for the visually impaired. In *Proceedings of the 10th International Conference on Computers and Accessibility*. ACM, 2008.

- [23] Beggs, W. D. A. Psychological correlates of walking speed in the visually impaired. *Ergonomics*, 1991.
- [24] Guth, D., Ashmead, D., Long, R., Wall, R., and Ponchillia, P. Blind and sighted pedestrians' judgments of gaps in traffic at roundabouts. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 2005.
- [25] Schroeder, B. J., Roupail, N. M., and Emerson, R. S. W. Exploratory analysis of crossing difficulties for blind and sighted pedestrians at channelized turn lanes. *Transportation Research Board of the National Academies*, 2007.
- [26] Arditi, A., Holtzman, J.D., and Kosslyn, S.M. Mental imagery and sensory experience in congenital blindness. In *Neuropsychologia*. Elsevier, 1988.
- [27] Fukasawa, N., Matsubara, H., Myojo, S., and Tsuchiya, R. Guiding passengers in railway stations by ubiquitous computing technologies. In *Proceedings of IASTED Human-Computer Interaction*. ACM, 2005.
- [28] Li, B., Ramsey-Stewart, E., Johar, K., Woo, D., and Rizos, C. More freedom for the blind and vision impaired—a proposed navigation and information system. In *IGNSS Symposium*. ACM, 2009.
- [29] Gerino, A., Alabastro, N., Bernareggi, C., Ahmetovic, D., and Mascetti, S. Math-melodies: inclusive design of a didactic game to practice mathematics. In *Proceedings of International Conference on Computers Helping People with Special Needs*. Springer, 2014.
- [30] Taibbi, M., Bernareggi, C., Gerino, A., Ahmetovic, D., and Mascetti, S. Audio-functions: Eyes-free exploration of mathematical functions on tablets. In *Proceedings of International Conference on Computers Helping People with Special Needs*. Springer, 2014.
- [31] Ahmetovic, D. Independent way-finding for visually impaired users through multi-sensorial data analysis on mobile devices. In *Proceedings of International Conference on Pervasive Computing and Communications (PhD Forum)*. IEEE, 2012.
- [32] Ahmetovic, D., Bernareggi, C., and Mascetti, S. ZebraLocalizer: identification and localization of pedestrian crossings. In *Proceedings of 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2011.
- [33] Ahmetovic, D. Smartphone-assisted mobility in urban environments for visually impaired users through computer vision and sensor fusion. In *Proceedings of International Conference on Mobile Data Management (PhD Forum)*. IEEE, 2013.
- [34] Ahmetovic, D., Bernareggi, C., Gerino, A., and Mascetti, S. ZebraRecognizer: efficient and precise localization of pedestrian crossings. In *Proceedings of the 22nd International Conference on Pattern Recognition*. IEEE, 2014.

- [35] Marius Von Senden. Space and sight: the perception of space and shape in the congenitally blind before and after operation. 1960.
- [36] Janet F Fletcher. Spatial representation in blind children. 1: Development compared to sighted children. *Journal of Visual Impairment and Blindness*, 1980.
- [37] Everett W Hill, John J Rieser, Mary-Maureen Hill, Marc Hill, et al. How persons with visual impairments explore novel spaces: Strategies of good and poor performers. *Journal of visual impairment & blindness*, 1993.
- [38] Florence Gaunet, Catherine Thinus-Blanc, et al. Early-blind subjects' spatial abilities in the locomotor space: Exploratory strategies and reaction-to-change performance. *PERCEPTION*, 1996.
- [39] Paul Bach-Y-Rita, Carter C Collins, Frank A Saunders, Benjamin White, and Lawrence Scadden. Vision substitution by tactile image projection. 1969.
- [40] Stephen E. Krufka and Kenneth E. Barner. Automatic production of tactile graphics from scalable vector graphics. In *Proceedings of the 7th International Conference on Computers and Accessibility*. ACM, 2005.
- [41] John A. Gardner and Vladimir Bulatov. Scientific diagrams made easy with iveotm. In *Proceedings of the 10th International Conference on Computers Helping People with Special Needs*. Springer, 2006.
- [42] Andreas Stefik, Christopher Hundhausen, and Robert Patterson. An empirical investigation into the design of auditory cues to enhance computer program comprehension. *International Journal on Hum.-Comput. Stud.*, 2011.
- [43] B.N. Walker and D.M. Lane. *Auditory display: sonification, audification, and auditory interfaces*. Westview Press, Boulder, CO, USA, 1994.
- [44] T. Hermann and H. Ritter. Listen to your data: model-based sonification for data analysis. *Advances in intelligent computing and multimedia systems*, 1999.
- [45] Dorte Hammershøi and Henrik Møller. Methods for binaural recording and reproduction. *Acta Acustica united with Acustica*, 2002.
- [46] Philip Mendels and Joep Frens. The audio adventurer: Design of a portable audio adventure game. In *Proceedings of the 2nd International Conference on Fun and Games*. Springer, 2008.
- [47] Joy Kim and Jonathan Ricaurte. Tapbeats: Accessible and mobile casual gaming. In *Proceedings of the 13th International Conference on Computers and Accessibility*. ACM, 2011.

- [48] Daniel Ramos and Eelke Folmer. Supplemental sonification of a bingo game. In *Proceedings of the 6th International Conference on Foundations of Digital Games*. ACM, 2011.
- [49] Patrick Ng and Keith Nesbitt. Informative sound design in video games. In *Proceedings of the 9th Australasian Conference on Interactive Entertainment: Matters of Life and Death*. ACM, 2013.
- [50] J.T. Cothroan B.N. Walker. Sonification sandbox: A graphical toolkit for auditory graph. In *Proceedings of the 9th Meeting of International Community for Auditory Display*. ACM, 2003.
- [51] Stephen H. Choi and Bruce N. Walker. Digitizer auditory graph: Making graphs accessible to the visually impaired. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*. ACM, 2010.
- [52] Heuten, W., Wichmann, D., and Boll, S. Interactive 3d sonification for the exploration of city maps. In *Proceedings of the 4th Nordic Conference on Human-computer interaction*. ACM, 2006.
- [53] Paul Stanley. Assessing the mathematics related communication requirements of the blind in education and career. In *Proceedings of the 11th International Conference on Computers Helping People with Special Needs*. Springer, 2008.
- [54] Cagatay Goncu and Kim Marriott. Gravvitas: generic multi-touch presentation of accessible graphics. In *Human-computer interaction—INTERACT 2011*. Springer, 2011.
- [55] Tatiana V Evreinova, Grigori Evreinov, and Roope Raisamo. An evaluation of the virtual curvature with the stickgrip haptic device: a case study. *Universal access in the information society*, 2013.
- [56] Poppinga, B., Pielot, M., Magnusson, C., and Rasmus-Gröhn, K. Touchover map: Audio-tactile exploration of interactive maps. In *Proceedings of 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2011.
- [57] Daniel Vogel and Patrick Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 657–666. ACM, 2007.
- [58] Nicholas A Giudice, Hari Prasath Palani, Eric Brenner, and Kevin M Kramer. Learning non-visual graphical information using a touch-based vibro-audio interface. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2012.

- [59] Burger Dominique, Bouraoui Amina, Mazurier Christian, Cesarano Serge, and Sagot Jack. Tactison: a multimedia learning tool for blind children. *Computers for Handicapped Persons*, 1994.
- [60] Ullman, B.R. and Trout, N.D. Accommodating pedestrians with visual impairments in and around work zones. In *Journal of the Transportation Research Board*. TRB, 2009.
- [61] Reginald G Golledge. Geography and the disabled: a survey with special reference to vision impaired and blind populations. *Transactions of the Institute of British Geographers*, 1993.
- [62] Nicholas A Bradley and Mark D Dunlop. An experimental investigation into wayfinding directions for visually impaired people. *Personal and Ubiquitous Computing*, 2005.
- [63] H Petrie. User requirements for a gps-based travel aid for blind people. *Orientation and navigation systems for blind persons*, 1995.
- [64] Kotaro Hara, Shiri Azenkot, Megan Campbell, Cynthia L Bennett, Vicki Le, Sean Pannella, Robert Moore, Kelly Minckler, Rochelle H Ng, and Jon E Froehlich. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view. In *Proceedings of the 15th International Conference on Computers and Accessibility*. ACM, 2013.
- [65] J. Sudol, O. Dialameh, C. Blanchard, and T. Dorcey. Looktel, a comprehensive platform for computer-aided visual assistance. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (Workshop)*. IEEE, 2010.
- [66] H. Petrie, V. Johnson, T. Strothotte, A. Raab, S. Fritz, and R. Michel. Mobic: Designing a travel aid for blind and elderly people. *Journal of Navigation*, 1996.
- [67] Rosen Ivanov. Indoor navigation system for visually impaired. In *Proceedings of the 11th International Conference on Computer Systems and Technologies (workshop)*. ACM, 2010.
- [68] Wu, H., Marshall, A., and Yu, W. Path planning and following algorithms in an indoor navigation model for visually impaired. In *Proceedings of the 2nd International Conference on Internet Monitoring and Protection*. IEEE, 2007.
- [69] James Coughlan and Roberto Manduchi. Functional assessment of a camera phone-based wayfinding system operated by blind users. In *Proceedings of the International Symposium on Research on Assistive Technology*. IEEE, 2007.
- [70] Chan, K.Y., Manduchi, R., and Coughlan, J. Accessible spaces: navigating through a marked environment with a camera phone. In *Proceedings of the 9th International Conference on Computers and accessibility*. ACM, 2007.

- [71] Malek Adjouadi. A man-machine vision interface for sensing the environment. *Journal of rehabilitation research and development*, 1991.
- [72] Masakatsu Kourogi, Tomoya Ishikawa, Yoshinari Kameda, Jun Ishikawa, Kyota Aoki, and Takeshi Kurata. Pedestrian dead reckoning and its applications. In *Proceedings of the International Symposium on Mixed and Augmented Reality (Workshop)*, 2009.
- [73] Iwan Ulrich and Johann Borenstein. The guidecane-applying mobile robot technologies to assist the visually impaired. *Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 2001.
- [74] AR Jimenez, F Seco, C Prieto, and J Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 37–42. IEEE, 2009.
- [75] Malek Adjouadi. Computer vision techniques to aid the blind. 1985.
- [76] Se, S. Zebra-crossing detection for the partially sighted. In *Proceedings of the conference on Computer Vision and Pattern Recognition*. IEEE, 2000.
- [77] Uddin, M.S. and Shioyama, T. Detection of pedestrian crossing using bipolarity feature-an image-based technique. In *Tran. on Intelligent Transportation Systems*. IEEE, 2005.
- [78] Uddin, M.S. and Shioyama, T. Detection of pedestrian crossing and measurement of crossing length - an image-based navigational aid for blind people. In *Transactions on Intelligent Transportation Systems*. IEEE, 2005.
- [79] Uddin, M.S. and Shioyama, T. Bipolarity and projective invariant-based zebra-crossing detection for the visually impaired. In *Workshop on Computer Vision Applications for the Visually Impaired*. IEEE, 2005.
- [80] Ivanchenko, V., Coughlan, J., and Shen, H. Crosswatch: a camera phone system for orienting visually impaired pedestrians at traffic intersections. In *Proceedings of 11th International Conference on Computers Helping People with Special Needs*. Springer, 2008.
- [81] J.M. Coughlan and H. Shen. Crosswatch: a system for providing guidance to visually impaired travelers at traffic intersection. *Journal of assistive technologies*, 2013.
- [82] Volodymyr Ivanchenko, James Coughlan, and Huiying Shen. Detecting and locating crosswalks using a camera phone. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (Workshop)*. IEEE, 2008.

- [83] Ivanchenko, V., Coughlan, J., and Shen, H. Staying in the crosswalk: A system for guiding visually impaired pedestrians at traffic intersections. In *Assist technol Res Ser.* IOS, 2009.
- [84] Murali, V. and Coughlan, J. M. Smartphone-based crosswalk detection and localization for visually impaired pedestrians. In *International Conference on Multimedia and Expo (workshop)*. IEEE, 2013.
- [85] Angin, P., Bhargava, B., and Helal, S. A mobile-cloud collaborative traffic lights detector for blind navigation. In *Proceedings of International Conference on Mobile Data Management*. IEEE, 2010.
- [86] Tsubasa Yoshida, Kris M. Kitani, Hideki Koike, Serge Belongie, and Kevin Schlei. Edgesonic: Image feature sonification for the visually impaired. In *Proceedings of the 2nd International Conference on Augmented Human*. ACM, 2011.
- [87] Ivan Kopecek and Radek Oslejsek. Hybrid approach to sonification of color images. In *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*. IEEE, 2008.
- [88] Ronald Aylmer Fisher. *Statistical methods for research workers*. 1934.
- [89] John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.
- [90] 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.
- [91] Kay Fitzpatrick, Susan T Chrysler, Vichika Iragavarapu, and Eun Sug Park. Crosswalk marking field visibility study. Technical report, 2010.
- [92] Von Gioi, R. G., Jakubowicz, J., Morel, J.M., and Randall, G. Lsd: A fast line segment detector with a false detection control. *Transactions on Pattern Analysis and Machine Intelligence*, 2010.
- [93] Kalman, R.E. A new approach to linear filtering and prediction problems. In *Transactions of the Journal of Basic Engineering*. ASME, 1960.
- [94] Liebowitz, D. and Zisserman, A. Metric rectification for perspective images of planes. In *Proceedings of Computer Vision and Pattern Recognition*. IEEE, 1998.
- [95] Lefler, M., Hel-Or, H., and Hel-Or, Y. Metric plane rectification using symmetric vanishing points. In *Proceedings of the 20th International Conference on Image Processing*. IEEE, 2013.
- [96] Huston, R. *Principles of biomechanics*. CRC press, 2008.

-
- [97] Akinlar, C. and Topal, C. Edlines: Real-time line segment detection by edge drawing (ed). In *Proceedings of 18th International Conference on Image Processing*. IEEE, 2011.
- [98] Brian FG Katz, Emmanuel Rio, Lorenzo Picinali, and Olivier Warusfel. The effect of spatialization in a data sonification exploration task. 2008.
- [99] Desolneux, A., Moisan, L., and Morel, J.M. Meaningful alignments. *International Journal of Computer Vision*, 2000.
- [100] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. In *Communications of ACM*. ACM, 1972.
- [101] A. Etemadi. Robust segmentation of edge data. In *Proceedings of the International Conference on Image Processing and its Applications*. IEEE, 1992.
- [102] Topal, C., Akinlar, C., and Genç, Y. Edge drawing: a heuristic approach to robust real-time edge detection. In *International Conference on Pattern Recognition*. IEEE, 2010.
- [103] Zhang, Y., Liu, Y., and Zou, Z. Comparative study of line extraction method based on repeatability. *Journal of Computational Information Systems*, 2012.