# Zebra Crossing Spotter: Automatic Population of Spatial Databases for Increased Safety of Blind Travelers

Dragan Ahmetovic
Università degli Studi di Milano
dragan.ahmetovic@unimi.it

Roberto Manduchi
University of California Santa Cruz
manduchi@soe.ucsc.edu

James M. Coughlan
Smith-Kettlewell Eye Research Institute
coughlan@ski.org

Sergio Mascetti
Università degli Studi di Milano
sergio.mascetti@unimi.it

## ABSTRACT

In this paper we propose a computer vision-based technique that mines existing spatial image databases for discovery of *zebra crosswalks* in urban settings. Knowing the location of crosswalks is critical for a blind person planning a trip that includes street crossing. By augmenting existing spatial databases (such as Google Maps or OpenStreetMap) with this information, a blind traveler may make more informed routing decisions, resulting in greater safety during independent travel.

Our algorithm first searches for zebra crosswalks in satellite images; all candidates thus found are validated against spatially registered Google Street View images. This cascaded approach enables fast and reliable discovery and localization of zebra crosswalks in large image datasets. While fully automatic, our algorithm could also be complemented by a final crowdsourcing validation stage for increased accuracy.

## Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Spatial databases and GIS*; I.4.8 [**IMAGE PROCESSING AND COMPUTER VISION**]: Scene Analysis—*Object recognition*; K.4.2 [**COMPUTERS AND SOCIETY**]: Social Issues—*Assistive technologies for persons with disabilities*

## General Terms

Algorithms, Human Factors

## Keywords

Orientation and Mobility, Autonomous navigation, Visual impairments and blindness, Satellite and street-level imagery, Crowdsourcing


*ASSETS'15,* October 26–28, 2015, Lisbon, Portugal.

ACM 978-1-4503-3400-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2700648.2809847 .

## 1. INTRODUCTION

Independent travel can be extremely challenging without sight. Many blind persons learn (typically with the help of an Orientation and Mobility, or O&M, professional) the routes that they will traverse routinely [20], for example to go to work, school or church. Far fewer attempt independent trips to new locations: for example to visit a new friend or meet a date at a restaurant. To reach an unfamiliar location, a blind person needs to learn the best route to the destination (which may require taking public transportation); needs to follow the route safely while being aware of his or her location at all times; and needs to adapt to contingencies, for example if a sidewalk is undergoing repair and is not accessible. Each one of these tasks has challenges of its own. In particular, the lack of visual access to landmarks (for example, the location and layout of a bus stop or the presence of a pedestrian traffic light at an intersection) complicates the wayfinding process. Thus, a straightforward walk for a sighted person could become a complex, disorienting, and potentially hazardous endeavor for a blind traveler.

Technological solutions for the support of blind wayfinding exist. Outdoors, where GPS can be relied upon for approximate self-localization, a blind person can use accessible navigation apps. While these apps cannot substitute proper O&M training, they provide the traveler with relevant information on-the-go, or can be used to preview a route to be taken. A navigation tool, though, is only as good as the map it draws information from. Existing geographical information systems (GIS) lack many features that, while accessible by sight, are not available to a blind person. For example, Hara et al. found that knowing the detailed layout of a bus stop (e.g., the presence of features such as a bench or nearby trees) can be extremely useful for a blind person for figuring out where to wait for the bus [8]. Other relevant information lacking in GIS may include the presence of curb ramps (curb cuts) near intersections, or the location of an accessible pedestrian signal controlled by a push button.

We propose a novel technique to detect zebra crossings on satellite and street level images. Knowing the location of marked crosswalks is important for any traveler. A pedestrian, crossing a street outside of a marked crosswalk, has to yield the right-of-way to all vehicles[1], a difficult task for pedestrians with visual impairments. Conversely, a marked crosswalk is clearly visible by drivers, grants right-of-way to pedestrians, and is highly preferable as a location for street

[1] `http://mutcd.fhwa.dot.gov`

crossing in terms of safety. Pedestrians who are blind or visually impaired are taught sophisticated O&M strategies for orienting themselves to intersections and deciding when to cross, using audio and tactile cues and any remaining vision [3]. However, there may be no non-visual cues available to indicate the presence and location of crosswalk markings.

Blind travelers may benefit from information about the location of marked crosswalks in two main ways. First, ensuring that a route includes street crossing only on clearly marked crosswalks would increase safety during a trip. Second, this information can be used jointly with other technology that supports safe street crossing. For example, recent research [4, 1] shows that computer vision-based smartphone apps can assist visually impaired pedestrians in finding and aligning properly to crosswalks. This approach would be greatly enhanced by the ability to ask a GIS whether a crosswalk is present, even before arriving at the intersection. If a crosswalk is present, the geometric information contained in the GIS can then be used to help the user aim the camera towards the crosswalk, align to it and find other features of interest (e.g., walk lights and walk light push buttons).

## 2. RELATED WORK

Satellite and street-level imagery of urban areas in modern GIS are vast data sources. Computer vision methods can be used to extract geo-localized information about elements captured in these images (e.g., landmarks, vehicles, buildings). Satellite images have been used to detect urban areas and buildings [19], roads [14] and vehicles [13]. Senlet and Elgammal [18] propose sidewalk detection that corrects occlusion errors by interpolating available visual data. In street level images, Xiao and Quan [21] propose detection of buildings, persons and vehicles, while Zamir and Shah [22] tackle the issue of localizing user captured photographs.

There has been an increasing interest in adding specific spatial information to existing GIS through crowdsourcing. The impact of this phenomenon, called volunteered geographic information (VGI), is stressed by Goodchild [6]. Wheelmap[2] allows accessibility issues to be marked on OpenStreetMap, the largest entirely crowdsourced GIS. Hochmair et al. [11] assess bicycle trails quality in OpenStreetMap while Kubásek et al. [12] propose a platform for reporting illegal dump sites.

Safe and autonomous urban navigation is difficult for people with visual impairments. Smartphone apps that address this issue have been proposed. iMove[3] informs the user about the current address and nearby points of interest. Crosswatch [4] and ZebraLocalizer [1] allow pedestrians with visual impairments to detect crosswalks with smartphone camera.

Crowdsourcing is also used to assist users with visual impairments during navigation. BlindSquare[4] is a navigation app that relies on Foursquare social network for points of interest data and OpenStreetMap for street info. VizWiz[5] allows one to ask help of a remote assistant and attach a picture to the request. BeMyEyes[6] extends this approach to allow assistance through video feed. Rice et al. [17] gather information on temporary road accessibility issues (e.g., roadworks, potholes). StopFinder [16] helps people with visual impairments to locate bus stops by gathering data about non-visual landmarks near bus stops. Hara et al. [8] improve on this approach by performing crowdsourcing on street-level imagery, without the need to explore an area of interest in person. Guy and Truong [7] propose an app to gather information on the structure and position of nearby crossings through crowdsourcing of street-level images and to assist users with visual impairments in crossing streets.

Computer vision techniques use satellite and street-level images to assist persons with visual impairments. Hara et al. [10] propose the detection of inaccessible sidewalks in Google Street View images. Murali and Coughlan [15] match 360° panoramas captured by smartphone to satellite images of the surroundings for estimating the user's position in the intersection more precisely than with GPS.

Hybrid approaches using automated computer vision techniques for menial work and human "Turkers" for more complex tasks have also been proposed. Hara et al. [9] extend their previous work to combine crowdsourcing and computer vision detection of street accessibility problems, such as obstacles or damaged roads.

To the best of our knowledge, our technique is the first to use satellite and street-level imagery for automated detection of zebra crossings. The detection can be extended with crowdsourcing for a final validation step, for adding missed crosswalks and for gathering other information on the surroundings of the detected crossings that can be of use to visually impaired pedestrians in finding and aligning to the crossings. Detected crosswalks can be added to a crowdsourced GIS and used by travelers with visual impairments during navigation planning. Solutions that detect crosswalks using the smartphone video camera [1, 4] can benefit from this information to assist the user in finding crossings at long distances that cannot be captured by the camera.

## 3. ZEBRA CROSSING IDENTIFICATION

Multiple types of road surface markings are used across the world to define pedestrian crossings. In the United States, at least two different types of pedestrian crossing markings are available[1]. The *transverse marking* consists of two white lines, perpendicular to the road direction, with width between 6in (15cm) and 24in (60cm). The separation between the two lines is at least 6ft (180cm). *Zebra crossings*, known as "continental crossings" in USA, can be visually detected at larger distances than other crosswalk markings in the same illumination [5]. Thus, zebra crossings are common when the crossing visibility is paramount for the pedestrians' safety, for example near schools and hospitals. As such, they also inform drivers to pay more attention to the crosswalk, which is desirable for pedestrians with visual impairments who cannot rely on sight for noticing incoming vehicles.

While in this contribution we focus on U.S. zebra crossings, the parameters of the detection can be tuned for other types of zebra crossings with different geometric characteristics. Other pedestrian crossing types, such as transverse markings, can be detected with the same approach but they require significant changes to the detection algorithm and therefore they will be considered as future work.

[2] http://wheelmap.org
[3] http://www.everywaretechnologies.com/apps/imove
[4] http://blindsquare.com/
[5] http://vizwiz.org
[6] http://www.bemyeyes.org

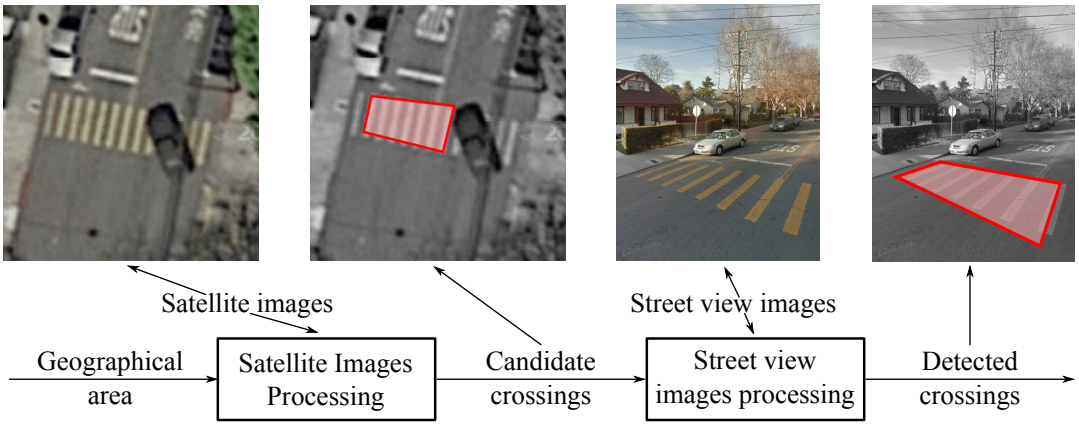



**Figure 1: Technique overview**

A zebra crossing is a set of parallel, uniformly painted, white stripes on a dark background. The gaps separating the white stripes are "dark stripes". Each stripe is a rectangle or, in case of diagonal crossings, a parallelogram. United States regulation dictates that zebra crossings be at least 6ft (180cm) wide, with white stripes 6in (15cm) to 24in (60cm) thick. The thickness of the dark stripes is not regulated.

## 3.1 Technique Overview

The proposed cascade classifier is composed of two main steps, as shown in Figure 1. The overall idea is to extract candidate zebra crossings from satellite images and then to validate them through street view images. The input of the procedure is an area $A$ in which to detect the zebra crossings.

In the first step (see Section 3.2), the satellite images covering the input area $A$ are downloaded from an online GIS and candidate zebra crossings are extracted from them through a computer vision technique. While the technique proposed in our solution can be applied to images from different providers, the system we developed gathers images from Google Maps.

The second step (see Section 3.3) acquires, for each candidate zebra crossing, nearby street view panoramas (if available) that might contain the zebra crossing. Zebra crossings are detected in street view panoramas and matched with the candidate crossings extracted from satellite images. In case of a match, the validated zebra crossing is returned and the result can be cached for future use.

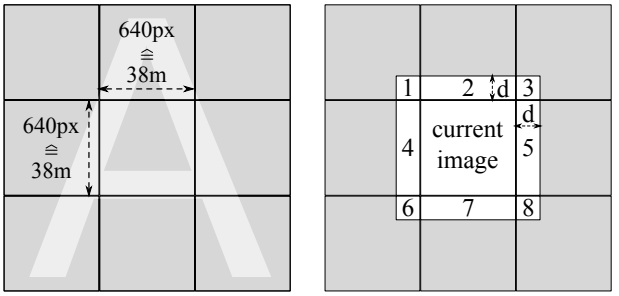
It is also possible to extract candidate zebra crossings from all street view panoramas in the input area $A$ and then validate them with satellite images. However, this approach would require downloading all panoramas in $A$, which is time consuming given the Google download limits[7] and the large amount of data. In our experiments, for an input area of 1.6km$^2$, starting first with satellite images required downloading $\approx$ 23MB, followed by 16MB for the street view panoramas validation. Conversely, starting from street view panoramas would have required downloading 637MB, followed by less than 1MB for the subsequent validation through satellite images. Overall, the cost (in terms of data transfer) of starting with satellite images is only about 6% of the reverse procedure that begins with street view images.

[7] `https://developers.google.com/maps/licensing`

## 3.2 Satellite Image Processing

Algorithm 1 describes the procedure to acquire the satellite images[8] and to process them. To acquire satellite images, we rely on the *Google Static Maps API*[9] that allows satellite images to be downloaded through HTTP calls. Each call specifies the GPS coordinates of the image center, the zoom factor and the image resolution. The maximum image resolution is constrained so, for large input areas, several HTTP requests must be sent.

We reconstruct the area $A$ by downloading the images whose union yields $A$. Indeed, given a zoom factor and the maximum image resolution, it is straightforward to compute the size (in meters) of the region covered by each image. Thus, $A$ can be partitioned in sub-regions with this maximum size (Line 2). Figure 2(a) shows an example of the partitioning step.




(a) Area $A$ partitioned (b) Extended image

**Figure 2: Area partitioning and extended image**

Since there is a maximum daily number of requests that can be submitted to Google Maps, we only download images that do not contain roads. This can be achieved through the *Google Maps Javascript API*[10]. These APIs expose a method to compute, given an input coordinates, the closest position on a road. Before downloading an image centered at a point $p$, we compute the distance from $p$ to the closest

[8] Satellite and street view imagery courtesy of Google ©

[9] `https://developers.google.com/maps/documentation/staticmaps/`

[10] `https://developers.google.com/maps/documentation/javascript/`

**Algorithm 1** Satellite images acquisition and processing

**Input:** Rectangular geographical area $A$.
**Output:** a set $Z$ of zebra crossings, each one represented by its position and direction.
**Method:**

```
1: Z ← ∅ {algorithm result}
2: partition A in a set R of sub-regions
3: for all (sub-region r ∈ R) do
4:    if (r does not contain a road) then continue
5:    download satellite image i of area r
6:    generate extended image i'
7:    L ← detect line segments in i'
8:    S ← group line segments in L in candidate crossings
9:    for all (candidate crossing s ∈ S) do
10:      if (s is not a valid crossing) then continue
11:      z ← position and direction of zebra crossing s
12:      merge and add z to Z
13:   end for
14: end for
15: return Z
```

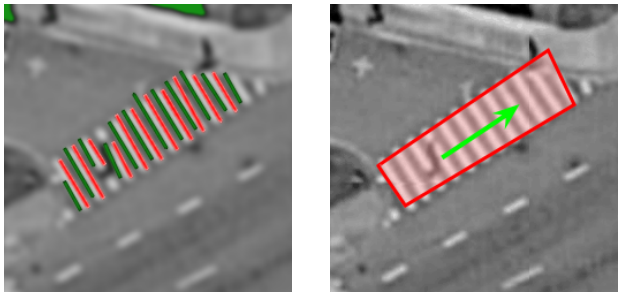
road. If this distance is larger than half the diagonal length of the image, we can infer that the image does not include any road, and hence we can avoid downloading it (Line 4). Note that this approach has a secondary advantage: it limits the number of false positives, i.e., patterns that are similar to zebra crossings that are erroneously recognized as such.

Dividing $A$ in subregions implies that a zebra crossing spanning multiple adjacent images may be recognized in some of them or may not be recognized in any. Consider the example in Figure 3, where the white dashed line is the boundary between two adjacent images. To tackle this problem, we construct an *extended image* by merging a downloaded image with the borders of the 8 surrounding images, as shown in Figure 2(b) (see Algorithm 1, Line 6). The width of each border is chosen to guarantee the recognition of a zebra crossing, even if it is on the border between adjacent images. Still, this approach can result in a crossing being recognized in more than one image. We show in the following how to merge candidate zebra crossings that correspond to the same crossing and appear in different images.

**Figure 3: Zebra crossing on two adjacent images**

The detection technique extracts the line segments corresponding to the long edges of stripes from the image (see Figure 4(a)) using a customized version of the EDLines algorithm [2] (Line 7). The line segments are grouped into sets of stripes based on horizontal distance, vertical distance and parallelism criteria (Line 8). The resulting candidate crossings (see Figure 4(b)) are validated based on the stripes number and color intensity and the candidate crossings that are not validated are discarded (Line 10).

Our technique is adapted from the ZebraLocalizer algorithm for zebra crossing recognition on smartphones [1]. Differently from ZebraLocalizer, our approach does not require reconstructing the ground plane since satellite images are not subject to perspective distortion, being acquired from above the ground plane.



(a) Line segments (b) Crossing

**Figure 4: Satellite detection steps**

If a set of stripes is not discarded, it is assumed to be a zebra crossing and it is characterized by its direction and position. Its direction is easily derived as the angle of the line perpendicular to the stripes, which should all share nearly the same angle, due to the parallelism criterion (Line 8). The zebra crossing's position is represented as the quadrilateral bounding the detected set of stripes, as depicted in Figure 4(b).

Finally, the detected zebra crossing is added to the set of results $Z$. As mentioned above, the same zebra crossing may be recognized in two or more different images. Hence, when inserting $z$ in $Z$ (Line 12), we first check if $Z$ already contains a zebra crossing with approximately the same position and direction as $z$. If a similar crossing is found, the two crossings are merged.

## 3.3 Street View Image Processing

Algorithm 2 describes the procedure to validate a single zebra crossing through the acquisition and processing of street view panoramas. The procedure is iterated for all zebra crossings detected during satellite image processing. In *Google Maps*, street view panoramas are spherical images (i.e., they span 360° horizontally and 180° vertically) positioned at discrete coordinates distributed non-uniformly in space and they are structured in a graph that closely follows the road graph. Through the *Google Maps Javascript API* it is possible to request the panorama closest to a point in space and to retrieve the coordinates of panoramas directly linked to a given one.

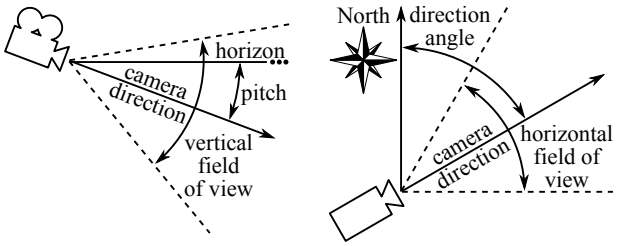
For an input zebra crossing $z$ the algorithm first identifies the coordinates $c_0$ of the panorama closest to it. The panorama centered in $c_0$ in some cases does not contain $z$ due to occlusion by the car used to take the pictures, or by other objects and vehicles. Thus, our algorithm processes $c_0$ as well as other nearby panoramas until the crossing has been validated or all nearby panoramas have been processed. The set of coordinates of panoramas still to be processed is represented by $C$, initially containing only $c_0$.

**Algorithm 2** Street view images acquisition and processing

**Input:** candidate zebra crossing $z \in Z$ represented by its position and direction.
**Output:** a validated zebra crossing $z'$ represented by its position and direction or **null** if not validated.
**Method:**

1: $c_0 \leftarrow$ get the coordinates of panorama closest to $z$
2: $C \leftarrow \{c_0\}$ {Set of panoramas to be processed}
3: **while** $(C \neq \emptyset)$ **do**
4: $c \leftarrow$ pop element from $C$
5: $\alpha \leftarrow$ direction angle from $c$ to $z$
6: $i \leftarrow$ image at coordinates $c$ with direction *alpha*
7: $L \leftarrow$ detect line segments in $i$
8: $L' \leftarrow$ rectify line segments in $L$
9: $S \leftarrow$ group line segments in $L'$ in candidate crossings
10: **for all** $(s \in S)$ **do**
11: **if** ($s$ is a valid crossing) **then**
12: $z' \leftarrow$ position and direction of zebra crossing $s$
13: **if** ($z$ matches $z'$) **then return** $z'$
14: **end if**
15: **end for**
16: push in $C$ coordinates of panoramas directly linked to $c$ and close to $z$
17: **end while**
18: **return null**

For each panorama centered in coordinates $c \in C$, the algorithm identifies a small portion of the spherical image that actually needs to be acquired. On the vertical axis, we use a fixed pitch ($-30°$) and vertical field of view ($60°$) in order to exclude the portion of the panorama occluded by the car taking the picture and the portion of the panorama above the horizon, which clearly does not contain crossings. For capturing the correct horizontal portion of the panorama we first compute the direction angle $\alpha$ that, starting from coordinates $c$, points towards the center of $z$ (see Line 5). This angle represents the center, along the horizontal axis, of the acquired image. A horizontal field of view of $60°$, given the pitch, vertical field of view, and a camera height of $2.5m$, guarantees to include, at minimum distance, a span of $3.5m$. Considering the United States regulation, this is sufficient to include from 5 to 23 stripes of a crossing, sufficient for a correct detection. See Figures 5(a) and 5(b).




(a) Vertical field of view (b) Horizontal field of view

**Figure 5: Horizontal and vertical field of view**

The usage of fixed parameters simplifies a number of formulae, including those for image rectification (see below). As future work these parameters could be derived from the relative position of $c$ and $z$. For example smaller pitch values can be used if $z$ is closer to $c$.

The image $i$ containing a portion of the panorama at coordinates $c$ and with direction $\alpha$ is acquired and then processed with an adapted version of ZebraLocalizer [1] algorithm to reconstruct the coordinates of line segments on the ground plane (Lines 7 to 11). The difference with ZebraLocalizer is that, since camera height and pitch are fixed, we can precompute the homography to reconstruct the ground plane in all acquired images. Figures 6(b) and 6(c) show an example of ground plane reconstruction. Rectified line segments are then grouped and validated based on the same criteria described in Section 3.2.

If the set $S$ of identified stripes represents a valid crossing $z'$, its position and orientation are compared with those of $z$. Clearly, even if $z'$ actually represents the same crossing as $z$, the bounding quadrilaterals of the two crossings rarely have exactly the same coordinates, due to a number of approximations introduced in the computation, including GPS imprecision and the fact that not all stripes of a zebra crossing are always identified. For example, the same crossing detected in a satellite image (Figure 6(a)) and detected in a street view panorama (Figure 6(b)) has different GPS coordinates, as shown in Figure 6(d). Similarly, the orientation of $z$ and $z'$ can differ.

Thus, there is a tolerance in the comparison of $z$ with $z'$ and a system parameter defines the maximum distance such that $z$ and $z'$ are considered the same crossing. Analogously, a system parameter defines the maximum angular distance between the orientation of $z$ and $z'$. If $z$ and $z'$ represent the same crossing, then $z'$ is returned as a valid crossing.

If the set of stripes $S$ does not represent a valid crossing or if $z'$ does not represent the same crossing as $z$, the algorithm populates $C$ with the coordinates of other panoramas close to $c$ (if any). The algorithm adds to $C$ the coordinates of panoramas directly linked to $c$ that have not already been processed during the validation of $z$ and that are close to $z$ (the maximum distance is bounded by a system parameter). Eventually either $z$ is validated or all nearby street view panoramas are processed (i.e.: $C = \emptyset$); in this case **null** is returned.

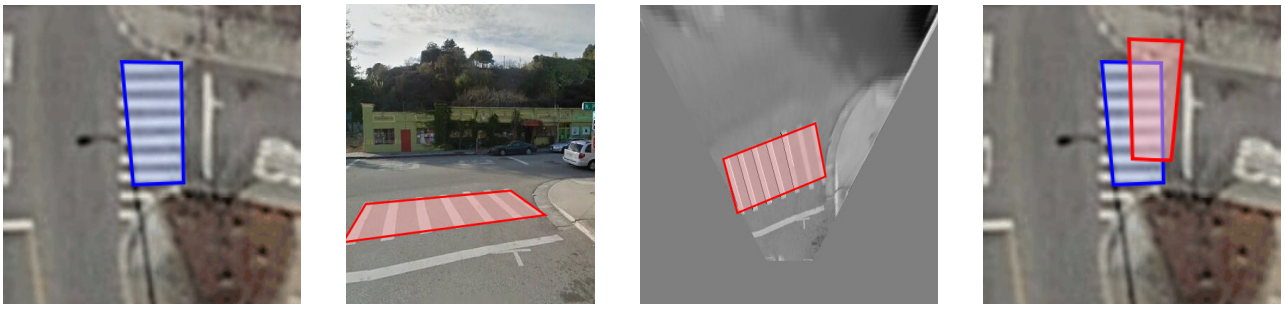
# 4. EXPERIMENTAL EVALUATION

This section reports the results of the experiments conducted to evaluate our technique over a dense urban region of San Francisco. These results are quantified in terms of precision (the fraction of detected zebra crossings that are true detections) and recall (the fraction of true zebra crossings that are correctly detected). We demonstrate that our cascade classifier is powerful enough to identify nearly all zebra crossings, with only a small number of false positive candidates that need to be eliminated in a subsequent stage of crowdsourcing-based image inspection.

## 4.1 Experimental setting

To evaluate the proposed technique we considered a rectangular urban area $A$ in San Francisco, with sides of length 1529m and 1025m and an area of 1.6km$^2$. The area coordinates[11], along with detected crossing portions, true positives (green pins) and false positives (red pins) have been published[12].

[11] `http://webmind.di.unimi.it/satzebra/satzebra.kml`
[12] `http://webmind.di.unimi.it/satzebra`



(a) Satellite detection (b) Street view detection (c) Reconstructed street view (d) imprecision in detection

**Figure 6: Imprecision in GPS coordinates between satellite and street view detected crossings**

A total of 141 zebra crossings and 152 transverse pedestrian crossings have been detected in the satellite and street view images of the area by a human operator. In the following, a zebra crossing is considered to be detected correctly if at least 4 consecutive stripes of the crossing have been detected correctly.

Concerning satellite images, with the maximum zoom level available in *Google maps* for the considered area, each image with maximum resolution of 640 × 640 pixels covers $38m \times 38$m. Thus, a total of 1149 satellite images are required to cover $A$. Since the size of each image is approximately 46KB, the size of all images covering $A$ is 52MB. The total number of street view panoramas available in $A$ is 1425. As we show in the following, we acquired only a small portion of these panoramas, each having a resolution of 640 × 640 pixels and, on average, a size of 51KB.

The tests were conducted on a laptop computer with *Intel core i7 4500u* 1.8GHz CPU and 8GB RAM.

## 4.2 Satellite image processing evaluation

As reported in Algorithm 1, only images containing streets are actually considered. With this approach a total of 791 images actually need to be acquired with a total size of 35MB. This means that, in $A$, our technique avoids downloading about one third of the images that would be otherwise required. In areas where the density of the road network is lower (like in suburban or rural areas), we can expect that an even higher percentage of images can be omitted.

The recognition process described in Algorithm 1 (Lines 7-8) detects a total of 773 zebra crossing portions. Often a single zebra crossing is detected as two or more zebra crossing portions. For example this can happen when a vehicle is visible in the middle of the crossing, causing a partial occlusion. By merging these crossing portions (Algorithm 1, Line 12) our technique identifies 199 candidate crossings.

Out of 199 detected candidate crossings, 137 correspond to actual zebra crossings. Since the number of actual zebra crossings in $A$ is 141, recall is 0.97. A few zebra crossings are not detected due to discolored or faded paint (see Figure 7(a)) while others are almost totally covered by trees, shadows or vehicles (see Figure 7(b)). The process also yields 62 false positives, hence the precision is 0.69. In many cases, false positives correspond to rooftops (Figure 7(c)) or other parts of buildings (Figure 7(d)).

These recall and precision scores refer to parameter settings tuned for the highest possible recall so that almost no crosswalks are missed by the algorithm. Naturally, perfect recall is difficult to reach and comes at the expense of a greater number of false positives, i.e., a smaller precision. However, considering that a final crowdsourcing validation step is possible, it is much easier for crowdworkers to rule out false positives than it is to find false negatives, which requires scrutinizing the entire area of interest to identify crosswalks that have not been detected by the algorithm. Parameters can be tuned for different trade-off levels between precision and recall. The Pareto frontier shown in Figure 8 lists the best precision and recall trade-offs obtained during the tuning of the parameters.

Regarding computation time, we consider the CPU-bound process only and we ignore the time to acquire images, which mainly depends on the quality of the network connection. The CPU-bound computation required for the extraction of candidate crossings in a single image is 180ms. Running the algorithm sequentially on the 791 images acquired for $A$ requires a total of 142s. However, note that the process can be easily parallelized and thus it would be straightforward to further reduce computation time.

## 4.3 Street view image processing evaluation

For each candidate crossing in $Z$ (the set of candidate crossings computed with satellite images), there are on average 5.7 nearby street view panoramas. By considering true positives only (i.e., candidate crossings that represent actual crossings), the average number of nearby street view panoramas increases to 7.3 and only 2 candidate crossings that represent actual crossings have no street view panoramas in their vicinity (less than 1.5%). Conversely, false positive candidate crossings have a much lower number of nearby street view panoramas (on average 2.6). Indeed, 19 false positives (30% of the total) do not have any nearby street view panorama and 46 false positives have 3 or fewer surrounding panoramas (74%). This is caused by the fact that, as observed previously, many false positives are located on rooftops or other areas that are not in the immediate vicinity of streets.

As reported in Algorithm 2, our solution acquires nearby panoramas iteratively, until the candidate crossing is validated. On average, considering true positives, 1.8 street view panoramas are acquired for each candidate crossing. In 76% of the cases a true positive crossing is validated with at most two panoramas, and up to 56% are validated by processing a single street view panorama (see Figure 9). Conversely, filtering out false positives requires processing all available nearby street view panoramas. Overall, the tech-

(a) FN - discoloration (b) FN - hidden by trees (c) FP - roof pattern (d) FP - stairs pattern

**Figure 7: False negatives (FN) and false positives (FP) in satellite detection and street view validation**

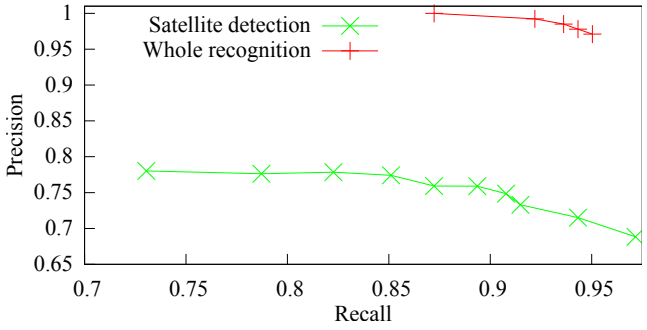



**Figure 8: Pareto frontier of the detection procedure**

nique requires acquiring and processing a total of 406 street view panoramas for $A$ ($\approx$ 2 for each candidate crossing). The total size of these images is 20.3MB.

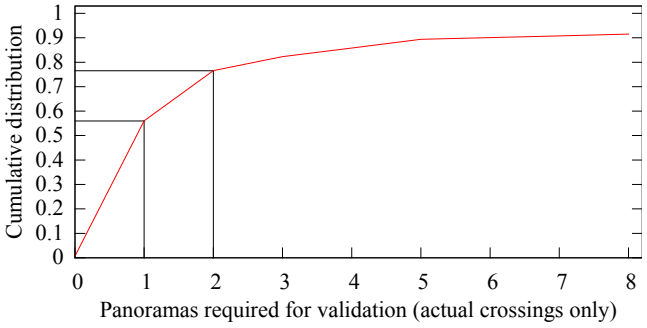



**Figure 9: Used panoramas cumulative distribution**

The street view-based validation (tuned for the best recall score) filters out 58 out of 62 false positives identified in the previous step, yielding a precision score of 0.97. The few false positives still present are caused by patterns very similar to zebra crossings, like the stairs in Figure 7(d). Of 137 true positives in $Z$, 134 are validated, resulting in a recall score of 0.98. Overall, the recall score of the whole procedure (including both satellite and street view detection) is 0.95.

As with the satellite detection, different parameter settings yield different precision and recall scores during the validation. Figure 8 shows the settings that yield the best precision and recall trade-offs during the validation.

Regarding computation time, each street view image can be processed in 46ms and hence the total computation time is 18.5s. Overall, considering the two detection steps (from satellite images and street view panoramas), the total CPU-bound computation time to process $A$ is 161s.

## 5. DISCUSSION AND FUTURE WORK

While the accuracy of the detection is high, since no computer vision algorithm is perfectly accurate, we envision a subsequent stage of processing based on online crowdsourcing (as in [10]) to rule out false positives and identify false negatives. A web service will be offered to allow users to submit crossings that have been missed (possibly also integrated in a future navigation software). For pruning false positives, instead, we intend to leverage crowdsourcing services such as Amazon Mechanical Turk[13] and propose that zebra crossings detected by our computer vision algorithm be referred to crowdworkers, who will decide whether zebra crossings are indeed present in the images. The crossings database could also be augmented with auxiliary information such as the presence and location of important features such as walk lights (which could be monitored in real time by the app) and walk push buttons. These could be also added by users or automatically by a visual detection app.

The resulting crosswalk database could be accessed by visually impaired pedestrians through the use of a GPS navigation smartphone app. The app could identify the user's current coordinates and look up information about nearby zebras, such as their number, placement and orientation. Existing apps such as Intersection Explorer and Nearby Explorer (for Android) and Sendero GPS LookAround and Intersection (for iPhone) could be modified to incorporate information from the crosswalk database. Computer vision-based detection apps on smartphones, such as Zebralocalizer [1] and Crosswatch [4], could additionally use the database to help users to approach and align properly to crosswalks even when they are not yet detected by the app.

Another important way in which the crosswalk database could be used by blind and visually impaired pedestrians is for help with offline route planning, which could be conducted by a traveler on his/her smartphone or computer from home, work or other indoor location before embarking on a trip. For example, a route-planning algorithm may weigh several criteria in determining an optimal route, including the number of non-zebra crossings encountered on the route (which are less desirable to traverse than zebra crossings) as well as standard criteria such as total distance traversed. The crosswalk database could also be augmented with additional information, including temporary hazards or barriers due to road construction, etc.

[13] `https://www.mturk.com`

## 6. CONCLUSIONS

Our contribution presents a technique that uses computer vision algorithms to detect and localize zebra crosswalks with high accuracy in existing spatial images databases such as Google satellite and Street View. To the best of our knowledge, these features are not already marked in existing GIS services and we argue that knowing the position of existing crosswalks could be useful to travelers with visual impairments for planning the navigation, finding pedestrian crosswalks and crossing roads.

For the detection we propose a cascade classifier derived from the algorithm we previously proposed for zebra crossing recognition on smartphones [1]. The proposed solution identifies potential crossings in street areas in satellite images and validates potential crossings with nearby Street View panoramic images.

We evaluated the proposed solution on a $1.6\text{km}^2$ area in San Francisco in which the pedestrian crossings were previously manually labeled. The dataset contained of 791 satellite images limited to street areas and 406 portions of Street View images referring to potential crossings. The technique achieved a precision of 0.97 and a recall of 0.95 with a computation time of $161s$ for the whole area.

The proposed approach is also complementary to existing solutions [1, 4] that leverage computer vision techniques for detecting pedestrian crossings using video cameras on mobile devices. A technique leveraging both data sources would be a helpful tool for assisting people with visual impairments to align to and safely and independently to cross roads.

## 7. ACKNOWLEDGMENTS

James M. Coughlan acknowledges support by the National Institutes of Health from grant No. 2 R01EY018345-06 and by the Administration for Community Living's National Institute on Disability, Independent Living and Rehabilitation Research, grant No. 90RE5008-01-00.

## 8. REFERENCES

[1] D. Ahmetovic, C. Bernareggi, A. Gerino, and S. Mascetti. Zebrarecognizer: Efficient and precise localization of pedestrian crossings. In *Int. Conf. on Pattern Recognition*. IEEE, 2014.

[2] C. Akinlar and C. Topal. Edlines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 2011.

[3] J. Barlow, B. Bentzen, D. Sauerburger, and L. Franck. Teaching travel at complex intersections. *Foundations of Orientation and Mobility*, 2010.

[4] J. Coughlan and H. Shen. Crosswatch: a system for providing guidance to visually impaired travelers at traffic intersection. *Jour. of Assistive Technologies*, 2013.

[5] K. Fitzpatrick, S. T. Chrysler, V. Iragavarapu, and E. S. Park. Crosswalk marking field visibility study. Technical report, 2010.

[6] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 2007.

[7] R. Guy and K. Truong. Crossingguard: exploring information content in navigation aids for visually impaired pedestrians. In *Conf. on Human Factors in Computing Systems*. ACM, 2012.

[8] K. Hara, S. Azenkot, M. Campbell, C. L. Bennett, V. Le, S. Pannella, R. Moore, K. Minckler, R. H. Ng, and J. E. Froehlich. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view. In *Int. Conf. on Computers and Accessibility*. ACM, 2013.

[9] K. Hara, V. Le, and J. Froehlich. Combining crowdsourcing and google street view to identify street-level accessibility problems. In *Conf. on Human Factors in Computing Systems*. ACM, 2013.

[10] K. Hara, V. Le, J. Sun, D. Jacobs, and J. Froehlich. Exploring early solutions for automatically identifying inaccessible sidewalks in the physical world using google street view. *HCI Consortium*, 2013.

[11] H. H. Hochmair, D. Zielstra, and P. Neis. Assessing the completeness of bicycle trails and designated lane features in openstreetmap for the united states and europe. In *Transportation Research Board Annual Meeting*, 2013.

[12] M. Kubásek, J. Hřebíček, et al. Crowdsource approach for mapping of illegal dumps in the czech republic. *Int. Jour. of Spatial Data Infrastructures Research*, 2013.

[13] J. Leitloff, S. Hinz, and U. Stilla. Vehicle detection in very high resolution satellite images of city areas. *Trans. on Geoscience and Remote Sensing*, 2010.

[14] M. Mokhtarzade and M. V. Zoej. Road detection from high-resolution satellite images using artificial neural networks. *Int. jour. of applied earth observation and geoinformation*, 2007.

[15] V. Murali and J. M. Coughlan. Smartphone-based crosswalk detection and localization for visually impaired pedestrians. In *Int. Conf. on Multimedia and Expo (workshop)*. IEEE, 2013.

[16] S. Prasain. Stopfinder: improving the experience of blind public transit riders with crowdsourcing. In *Int. Conf. on Computers and Accessibility*. ACM, 2011.

[17] M. T. Rice, A. O. Aburizaiza, R. D. Jacobson, B. M. Shore, and F. I. Paez. Supporting accessibility for blind and vision-impaired people with a localized gazetteer and open source geotechnology. *Transactions in GIS*, 2012.

[18] T. Senlet and A. Elgammal. Segmentation of occluded sidewalks in satellite images. In *Int. Conf. on Pattern Recognition*. IEEE, 2012.

[19] B. Sirmacek and C. Unsalan. Urban-area and building detection using sift keypoints and graph theory. *Trans. on Geoscience and Remote Sensing*, 2009.

[20] W. R. Wiener, R. L. Welsh, and B. B. Blasch. *Foundations of orientation and mobility*. American Foundation for the Blind, 2010.

[21] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *Int. Conf. on Computer Vision*. IEEE, 2009.

[22] A. R. Zamir and M. Shah. Accurate image localization based on google maps street view. In *Oroc. of. European Conf. on Computer Vision*. Springer, 2010.