*2. Manuscript

Robust Traffic Lights Detection on Mobile Devices for Pedestrians with Visual Impairment.

Sergio Mascetti[a,b,*], Dragan Ahmetovic[a], Andrea Gerino[a,b], Cristian Bernareggi[a,b], Mario Busso[a], Alessandro Rizzi[a]

[a] *Università degli Studi di Milano, Dept. of Computer Science, Milan, Italy*
[b] *EveryWare Technologies, Milan, Italy*

Abstract

Independent mobility involves a number of challenges for people with visual impairment or blindness. In particular, in many countries the majority of traffic lights are still not equipped with acoustic signals. Recognizing traffic lights through the analysis of images acquired by a mobile device camera is a viable solution already experimented in scientific literature. However, there is a major issue: the recognition techniques should be robust under different illumination conditions.

This contribution addresses the above problem with an effective solution: besides image processing and recognition, it proposes a robust setup for image capture that makes it possible to acquire clearly visible traffic light images regardless of daylight variability due to time and weather. The proposed recognition technique that adopts this approach is reliable (full precision and high recall), robust (works in different illumination conditions) and efficient (it can run several times a second on commercial smartphones). The experimental evaluation conducted with visual impaired subjects shows that the technique is also practical in supporting road crossing.

Keywords: Assistive technologies; computer vision; visual impairment; traffic lights; mobile devices.

*Corresponding author
Email: sergio.mascetti@unimi.it
Phone: +39 02 50316336
Fax: +39 02 87153726

1. Introduction

Most mobile devices are accessible to people with visual impairment or blindness (VIB)[1]. This makes it possible to use these devices as platforms for the development of assistive technologies. Indeed, applications specifically designed for people with VIB are already available in online stores. For example, *iMove* supports independent mobility in urban environment by "reading aloud" the current address and nearby points of interest[2]. Other solutions proposed in the scientific literature adopt computer vision techniques to extract contextual information from the images acquired through the device camera. In particular, this paper focuses on the problem of recognizing traffic lights with the aim of supporting a user with VIB in safely crossing a road.

A number of solutions have been proposed in the scientific literature to recognize traffic lights. Existing solutions have a common problem: they use images acquired through the device camera with automatic exposure. With this approach, in conditions of low ambient light (e.g., at night) traffic lights result overexposed (see Figure 1) while in conditions of high ambient light (e.g., direct sunlight) traffic lights are underexposed (see Figure 2).

This paper presents *TL-recognizer*, a traffic light recognition system that solves the above problem with a robust image acquisition method, designed to enhance the subsequent recognition process. Experimental results show that *TL-recognizer* is reliable (full precision and high recall) and robust (works in different illumination conditions). *TL-recognizer* has also been optimized for efficiency, as it can run several times a second on commercial smartphones. The evaluation conducted on subjects with VIB confirms that *TL-recognizer* is a practical solution.

This paper is organized as follows: Section 2 discusses the related work and defines the objectives of this contribution. The basic acquisition and recognition technique is presented in Section 3, while improvements are described in Section 4. Section 5 reports the results of the extensive experimental evaluation and finally Section 6 concludes the paper.

---

[1] In case the reader is unfamiliar with accessibility tools for people with VIB, a short introduction video is available at `http://goo.gl/mEI6Uz`.

[2] At the time of writing, iMove is available for free download from AppStore: `https://itunes.apple.com/en/app/imove/id593874954?mt=8`.

Figure 1: Pedestrian traffic light is overexposed.

Figure 2: Pedestrian traffic light is underexposed.

2. Detecting traffic lights for people with VIB

Independent mobility is a challenge for people with sight impairments, in particular for what concerns crossing a road at a traffic light. A solution to this problem consists in the use of acoustic traffic lights. There are many different models of acoustic traffic lights. For example, in Italy, there are acoustic traffic lights that produce sound on demand by pushing a button placed on the pole. The sound signals to the person with VIB when the light is green. In Germany, there are models that always produce a sound when the light is green (no button has to be pushed) and they adapt the intensity of the sound according to the background noise.

Nonetheless, as reported by many associations for blind and visually impaired persons, in most industrial countries (e.g., Italy, Austria, France, Germany, etc.), acoustic traffic lights are not ubiquitous; they are present in some urban areas but may be absent in small towns. Furthermore, acoustic traffic lights are not always working properly because damages often take a long time to be reported and fixed. The situation can be even worse in developing countries.

2.1. Related work

One of the first contributions on traffic light recognition was presented by Kim et al. [1]. This solution is aimed at assisting drivers with color deficiency. Images are acquired through a digital video camera and processed by a notebook. The main limitation of this solution is that it works correctly only when there is a uniform background (e.g., the sky). Consequently this solution cannot be applied to the purpose of detecting pedestrian traffic lights, because they are located in urban environments where the background contains, for example, shop lights and trees.

Several other solutions proposed in the literature are specifically designed for smart vehicles [2, 3, 4, 5, 6]. These techniques cannot be directly used to guide people with VIB because they are specifically optimized for circular or elliptical lights, while pedestrian traffic lights have different shapes.

Differently, other solutions, while designed for smart vehicles, are not specialized for circular or elliptical traffic lights and hence can be adapted to recognize pedestrian traffic lights. The solution by Wang et al. [7] aims at recognizing traffic lights in a complex urban environment. The proposed technique first computes color segmentation in the HSI color space, then identifies candidate regions and finally uses a template-matching function to validate a traffic light. The solution by Cai et al. [8] is aimed at recognizing 'arrow-shaped' traffic lights. In this solution, the dark regions of the images are singled out. Then, the regions that are either to small or too big are discarded. Subsequently, a color filter for green, red and yellow is applied to the candidate regions. Eventually, the arrow is recognized through Gabor transform and 2D independent component analysis. The solution by Almagambetov et al. [9] discusses a technique aimed at guaranteeing recognition of traffic lights from large distances (this is clearly an important feature for smart vehicles) and tackles the problem of recognizing 'arrow-shaped' traffic lights through a template-matching technique. The solution proposed by Charette et Nashashibi [10] detects, with a template-matching technique, the optical unit, the signal head as well as the traffic light pole.

Other solutions have been specifically proposed to support detection of pedestrian traffic lights with the aim of supporting users with VIB. Ivanchenko et al. [11] present a recognition algorithm for smartphones designed for traffic lights in U.S.. The status of the traffic light is represented by the white shape of a pedestrian together with a circular light that can become red, yellow or green. In the first step, the algorithm uses smartphone sensors to determine the position of the smartphone with respect to the horizon and it analyzes only the upper part of the image. Secondly, it detects the circular light and the shape of the pedestrian. This algorithm also searches for a pedestrian walk to validate the result.

Roters et al. in [12] investigate an algorithm consisting in three stages:

identification, *video analysis* and *time-based verification.* In the identification stage, the algorithm recognizes the traffic light in front of the pedestrian. The video analysis stage tracks the candidate traffic light in different frames of the video. Finally, during the time-based verification stage, the results of the identification stage are double-checked with those of the video analysis. Our contribution focuses on the first stage only; the other two forms of reasoning are important in the final application, and in fact the proposed architecture implements them in the *TL-logic* module (see Section 2.3). This contribution improves the identification stage by proposing a technique that is rotation invariant and that also takes into account the shape of the pedestrian traffic light.

Most of the techniques mentioned above have a common problem: the images are processed *after* their acquisition with the aim of guaranteeing robust recognition under different lighting conditions. The problem has been explicitly highlighted by Diaz-Cabrera et al. [5] that proposes a method for smart vehicles for detecting and determining the distance of Italian suspended vehicle traffic lights. The approach uses normalized RGB color space to obtain a consistent accuracy in different illumination conditions. However, experimental results are still unsatisfactory in bright days or at night.

A follow-up publication by Diaz-Cabrera et al. [6] argues that it is impossible to reconstruct information with high precision from overexposed or underexposed images like the ones in Figures 1 and 2. Thus, the authors propose dynamic exposure adjustment based on sky pixels segmentation and luminosity evaluation. The paper also proposes an enhanced fuzzy-based color clustering and improves the previous solution with a faster, parallelized detection and a higher accuracy detection and distance computation. In our approach we also propose a dynamic method for exposure adjustment based on external luminosity that makes it possible to acquire suitable images in all illumination conditions at the desired distances. Differently from Diaz-Cabrera et al. [6], our approach also uses shape matching to identify pedestrian traffic lights. Also, due to the fact that the device is held by the user, we leverage accelerometers and gyroscopes to compute the device's position in space and correctly detect and measure the distances between the user and the pedestrian traffic light.

It is not possible to fairly compare the solution proposed in this contribution with previous ones, based on quantitative experimental results. Indeed, many existing contributions only present qualitative evaluations and, among those presenting quantitative results, very few are based on a publicly available dataset of images. Also, the few public datasets contain images that had not been acquired with the proposed solution for dynamic exposure adjustment and, in most of the cases, they do not include accelerometer measurements for each frame. Hence, it is only possible to compare the experimental results presented in this contribution with other ones obtained with different datasets of images, which leads to possibly biased outcomes. Another important difference is that, in some existing solutions, precision and recall are computed on streams of images, rather than on single images, hence applying a sort of "high level reasoning" to aggregate results from different successive frames. Roters et al. [12] experimentally show that the analysis of video yields better results (in term of precision and recall) than the analysis of single frames. Still, the solution by Roters et al. has a precision of 1 and a recall of about 0.5, while our solution has a precision of 1 and a recall of 0.81 (see Section 5). Conversely, the solution by Almagambetov et al. [9] has a higher detection rate (up to 100% for certain illumination conditions), but it incurs into false positives and precision is as low as 0.8, which is unacceptable for the application considered in this contribution.

Finally, a set of papers address the problem of traffic light detection with a solution based on machine learning ([13, 14]). A comparison between recognition of traffic lights though analytic image processing and learning-based processing was proposed by De Charette and Nashashibi [15]. The authors conclude that analytic image processing guarantees better performances in terms of precision and recall. For this reason, our contribution focuses on this approach.

2.2. User story description

Many people with VIB learn (typically with the help of an Orientation and Mobility professional) the routes that they will be undertaking daily, for example to go to work, school or church [16]. It is less common that a person with VIB independently attempts trips to new locations. The recognition technique described in this contribution enables the development of a mobile application that supports people in both cases, as described in the following two user stories that have been designed with the support of a blind person, with a user-centered design approach.

User story 1. A person with VIB that is moving along a known path keeps track of his/her approximate position and heading with respect to many points of reference that can be perceived through touch (e.g., with the white cane), hearing or possibly through any residual sight. Upon reaching a road crossing with a traffic light, the person takes his/her mobile device and runs the application that automatically starts acquiring images from the camera. Then, he/she points the camera towards the traffic light. The person knows the direction (both horizontal and vertical), that he/she learned while practicing on the route. It should be observed that the camera field of view is generally larger than about $\pi/4$ on both dimensions[3], even if the person points with an error of about $\pi/8$, the traffic light will still be in the field of view.

As soon as the application detects the traffic light, it gives a feedback (e.g., a vibration) and reads the current color or provides an instruction (like "stop" or "go") with a text-to-speech message or through a vibration pattern. To guarantee a safe crossing, if the application first detects a green light, it still instructs the person not to cross: the traffic light needs first to turn red and then, when it turns green again, the user is instructed to cross. Note that this is the same approach used in many acoustic traffic lights.

User story 2. A person with VIB that is walking along an unknown route incurs into two additional problems. First, he/she might be unaware whether the road intersection has traffic lights. Second, he/she might be unaware of where to point the device camera to frame the traffic light. To support the user in solving these two problems, the application, by using the accelerometer, instructs the person on how to point the camera along the vertical direction. Indeed, since traffic lights are above the horizon, the device should be held with an angle such that the lower border of the captured image is approximately on the horizon. This guarantees that the upper edge of the image is above a traffic light, if any are present.

To "find" the traffic light along the horizontal direction, the person can rely on the fact that traffic lights are oriented towards the direction where the pedestrian is coming from. So the person has an approximate knowledge of the angular range where he/she should point the camera. Then, starting from one edge of this range, the person can scan towards the other range while the application processes the images. By using the device gyroscopes it is possible to detect if the user is rotating too fast and, in this case, to inform him/her. This guarantees that a traffic light is detected with high likelihood, if one is actually present.

[3]The exact value depends on the specific device.

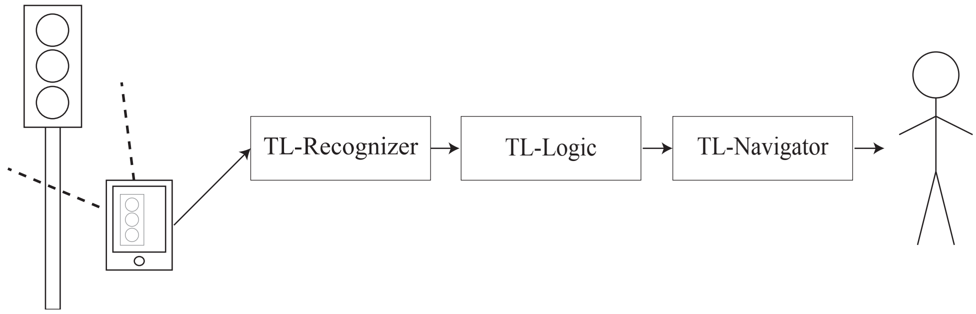

Figure 3: Structure of the main application modules.

2.3. System Modules

This paper focuses on the *TL-recognizer* module that computes the position and color of a pedestrian traffic light in a given image. For the detection of traffic lights *TL-recognizer* relies on data sources available on off-the-shelf smartphones: video camera, accelerometer and gyroscope. The first captures image frames that can then be analyzed with computer vision techniques. Accelerometer and gyroscope, on the other hand, can be used to extract the orientation of the device with respect to the ground plane. As shown in the following, this information has an important role in the proposed technique.

In addition to processing frames, an application that supports people with VIB in road crossing should implement at least two other functionalities, which are designed as other two modules: *TL-logic* and *TL-Navigation* (see Figure 3).

The *TL-logic* module is in charge of combining different results of *TL-recognizer* and computing messages to guide the user. Example 1 shows a simple form of reasoning.

Example 1. *One run of* TL-recognizer *detects a red traffic light in a certain position.* TL-logic *computes a 'wait' message to instruct the user not to cross. After the recognition,* TL-logic *uses accelerometer and gyroscope data to estimate how the device is being moved and hence where the traffic light is expected to be in the next frame. Indeed, the following run of* TL-recognizer *identifies a green traffic light in the expected position. Consequently* TL-logic *can conclude that the traffic light has now turned green and therefore generates a 'cross' message for the user.*

The *TL-Navigation* is in charge of conveying the messages to the user through audio, haptic (vibration) and graphical information. The main challenge in using audio information is that it should not divert the user's attention from the surrounding audio scenario, which is essential to acquire indispensable information (e.g., an approaching car, a person walking by, etc.). Indeed, as remarked by Ullman et al., blind people run into difficulty when guided by verbose speech messages [17]. In the field of pedestrian crossings, the problem of guiding people with VIB has been specifically addressed by Mascetti et at. [18].

2.4. The target to detect

This paper considers traffic lights currently used in Italy, which adhere to European Standard 12368 [19]. This standard specifies a number of physical properties of the traffic lights, including, for example, their size, luminous intensities and colors that have to be consistent in all European countries.

Luminous intensities are specified in two classes, with a common minimum and two maxima according to the class. Values are different according to the color and are reported in Table 1.

	red	yellow	green
min	$100cd$	$200cd$	$400cd$
Max Class 1	$400cd$	$800cd$	$1000cd$
Max Class 2	$1100cd$	$2000cd$	$2500cd$

Table 1: Luminous intensities range in the reference axis according to European Standard 12368 [19].

Chromaticities are delimited in the CIE XYZ space according to the values reported in Table 2.

In Italy, as in many other countries, differently shaped lights are used to transfer messages to different classes of road users. For example, the rounded light is used for drivers, while the "body-shaped" light is used for pedestrians. Two different shapes are used in Italy for pedestrians lights: one for green light, the other for yellow and red lights (see Figures 7, 8 and 9). While the actual shape of the figure appearing through the lens can vary from country to country (in some cases even within the same country), the proposed solution can be easily adapted to most existing standards by simply re-tuning the detection parameters and by using different template images

	chromaticity boundaries	boundary
red	$y = 0.290$	red
	$y = 0.980 - x$	purple
	$y = 0.320$	yellow
yellow	$y = 0.387$	red
	$y = 0.980 - x$	white
	$y = 0.727x + 0.054$	green
green	$y = 0.726 - 0.726x$	yellow
	$x = 0.625y - 0.041$	white
	$y = 0.400$	blue

Table 2: Chromaticities range according to European Standard 12368 [19].

(see Section 3.5). Also, if the proposed technique is used in countries with very particular light conditions (e.g., a bright sunny day in the desert) it could be necessary to accordingly tune the acquisition parameters with the methodology presented in the following.

Among other physical properties of the traffic light, its position with respect to the observer is particularly relevant. Indeed, given the application, only traffic lights with bounded distance from the observer should be detected. For example, considering the width of urban roads, in the experiments the minimum and maximum horizontal distances adopted are 2.5m and 20m, respectively. Analogously the signal head should not be too high or too low with respect to the observer. Hence the vertical distance is bounded. For example, in the experiments the minimum and maximum vertical distances adopted are 0.5m and 4m, respectively. Finally, the user is interested only in the traffic lights that 'point' towards him/her. Consider for example Figure 4: the direction of the red traffic light (red circle) is roughly the same angle as the line passing through the traffic light and the user (black circle). Hence, that traffic light should be detected. Vice versa, the green traffic light (green circle) is pointing away from the user and hence it should not be detected. The 'maximum rotation distance' is the parameter defining the angular distance between the direction of the traffic light and the direction from the traffic light towards the user. In the experiments a 'maximum rotation distance' of 45° is adopted. In a typical crossroad like the one in Figure 4, this value prevents the identification of a diagonally opposite traffic

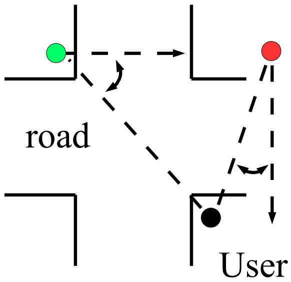

Figure 4: Example of 'maximum rotation angle'.

light that, generally, shows an opposite color with respect to the one shown by the traffic light the user is interested in.

Henceforth some of the terms defined in European Standard 12368 [19] are used. In particular, the *signal head* (see Figure 5) is the device composed by different *optical units* (see Figure 6), each one with its *lens*. For example, in Italy, there are three optical units in each signal head. The *background screen* is the opaque and dark board placed around the optical units to increase the contrast. Also, the term *active optical unit* ('AOU' in the following) refers to the optical unit that is lighted in a given instant (as in Figure 6). Finally, "optical unit color" is the color of an optical unit when it is active. Examples of different visual appearances of the AOU are shown in Figures 5 to 9.

Figure 5: Signal head

Figure 6: (Active) optical unit

Figure 7: Green AOU

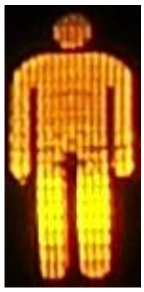

Figure 8: Yellow AOU

Figure 9: Red AOU

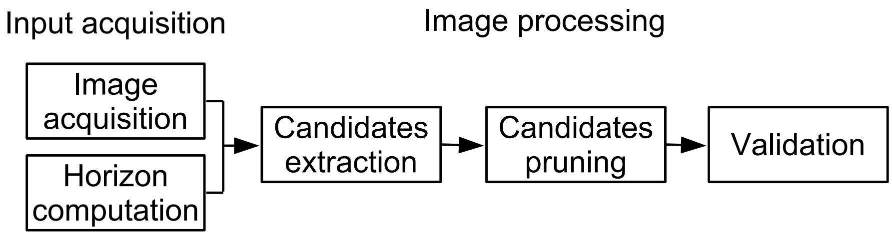

Figure 10: Organization of the recognition process

3. Recognizing traffic lights

3.1. Technique overview

The recognition process is organized in two main phases: 'input-acquisition' and 'image-processing' (see Figure 10). Input-acquisition is composed of two main steps: 'image acquisition' and 'horizon computation'. During image acquisition a frame is captured by the device camera using specifically designed exposure parameters. This is presented in Section 3.2. The horizon computation step uses accelerometer and gyroscope data to compute the equation of the horizon line in the image reference system. The horizon computation is based on Property 1 (proofs of formal results are in Appendix A).

Property 1. *Let ρ and θ be the device pitch and roll angles respectively, $C = \langle C_x, C_y \rangle$ is the center of the image and f is the focal distance of the camera (in pixels). Then, the equation of the horizon line h inside the acquired image is:*

$$\sin(\theta)x - \cos(\theta)y - \sin(\theta)(C_x + \tan(\rho)\sin(\theta)f) + \cos(\theta)(C_y + \tan(\rho)\cos(\theta)f) = 0 \tag{1}$$

The image-processing phase is aimed at identifying the AOUs that appear in the image. The overall computation is presented in Algorithm 1 and can be logically divided into three steps: extraction of candidate AOUs, pruning of candidate AOUs and validation of AOUs (see Sections 3.3, 3.4 and 3.5, respectively).

The image-processing algorithm takes in input the results of the acquisition phase: an image i (encoded in the HSV color space) and the horizon line equation h. There are other system parameters that form the algorithm input: three range filters f_g, f_y and f_r, one for each optical unit color; three

Algorithm 1: Image processing (non optimized version)

Input: image i; horizon line equation h; range filters f_g, f_y and f_r; template images t_g, t_y and t_r; threshold value $T \in (0,1)$.

Output: a set R of active optical units. Each element of R is a pair $\langle o, c \rangle$ where o is the AOU contour and c the color.

Constants: g, y and r represent the three optical unit colors (i.e., green, yellow and red).

Method:

$R \leftarrow \emptyset$ {algorithm result}
for all (color $c \in \{g, y, r\}$) **do**
    {Extraction of candidate AOU}
    $i' \leftarrow$ apply f_c to i {i' is a binary image}
    $O \leftarrow$ extract the set of contours from i'
    for all (contour $o \in O$) **do**
        {Pruning of candidate AOU}
        $o' \leftarrow$ rotate o by the inverse of the inclination of h
        if (o' does not satisfy "distance" or "width" properties) **then**
            continue {prune o}
        end if
        {Validation}
        $p \leftarrow$ image patch, extract from i, corresponding to the MBR of o'
        $p \leftarrow$ resize p to have the same size of t_c
        α is the result of normalized cross correlation between t_c and p
        if ($\alpha > T$) **then** add $\langle o, c \rangle$ to R
    end for
end for

template images t_g, t_y and t_r, each one representing the three lenses and, finally, a threshold value $T \in (0, 1)$ used in the validation step. The output of the algorithm is a set of identified AOUs, each one represented by its color and its contour in the input image.

3.2. Image acquisition

The exposure of the image to be acquired is a key point. Light conditions during day and night are extremely variable, while luminance coming from traffic lights is pretty stable. Since smartphone camera automatic exposure balances the mean luminance of every point in the entire image, its use can result in underexposed or overexposed AOUs (see Figures 1 and 2). For this reason, the proposed solution disables the automatic exposure feature of the mobile device and sets a fixed exposition value (EV) chosen among a small group of EVs pre-computed to encompass the luminance variations. These variations are mainly due to traffic light class (see Section 2.4), and acquisition noise due to distance, misalignment, veiling glare, pixel saturation etc.

Before selecting candidate EV values, the intensity and chromaticity of light coming from a set of traffic lights were empirically verified. Table 3 reports the values measured for four of them, as an example of the high variability.

Although the standard for traffic light luminous intensity is clearly defined, variability in the real world (i.e., in the streets) can be very high, both in terms of illuminance and chromaticity. The reasons are many: class (see Section 2.4), technology of light bulbs, dirt on the lens, aging, etc.

To identify the correct EV, a series of pictures were taken at different times of the day and distances, starting from the theoretical EV computed from the European Standard luminous intensity ([19]) on a ± 5 stops bracketing, with step 1. From this set of shots, a subset of EVs were selected to cover the major part of the variance of correctly exposed lenses, in four light conditions.

The four light conditions are: very high light intensity (e.g., a sunny day at noon), high light intensity (e.g., a partially cloudy day at noon, or a clear day when the Sun is not high in the sky), mid light intensity (e.g., a cloudy day, or a clear day at dawn or dusk), low light intensity (e.g., night). Note that, for our purposes, light condition is highly influenced by the time of day and by weather conditions (e.g., sunny, cloudy, etc...), while other meteorological conditions (like rain) do not affect light intensity. To

traffic light number	AOU color	Lux	x	y
1	green	2671	0.0875	0.6075
	yellow	1138	0.5839	0.4155
	red	740	0.7068	0.293
2	green	491	0.2785	0.495
	yellow	1199	0.5676	0.4471
	red	723	0.6568	0.3425
3	green	754	0.2193	0.5025
	yellow	1502	0.5755	0.4129
	red	955	0.6854	0.3142
4	green	1941	0.0727	0.5091
	yellow	2065	0.587	0.4121
	red	1082	0.7048	0.2951

Table 3: Intensity and chromaticity of four sample traffic lights.

automatically identify the light condition, the following approach is adopted: before starting recognition, a picture is taken with fixed camera parameters (ISO 100, aperture F8.0, shutter speed 1/125). Then, value M is computed as the mean, for each pixel, of the V channel. This value characterizes the light condition. Table 4 shows how light conditions are specified as well as the camera parameters that yield best shots in each of them. It may appear counterintuitive but at night time the exposition is shorter; this reduces the optical veiling glare on the edges of the body shaped lens.

Light intensity	M	ISO	Aperture	Shutter speed
Very High	$120 < M$	100	F8.0	1/160
High	$60 < M \leq 120$	100	F8.0	1/200
Mid	$5 < M \leq 60$	100	F8.0	1/250
Low	$M \leq 5$	100	F8.0	1/500

Table 4: EV parameters.

Image acquisition with fixed EV was implemented on both Android 4.x and Android 5.x. With Android 4.x it is possible to set the values for ISO,

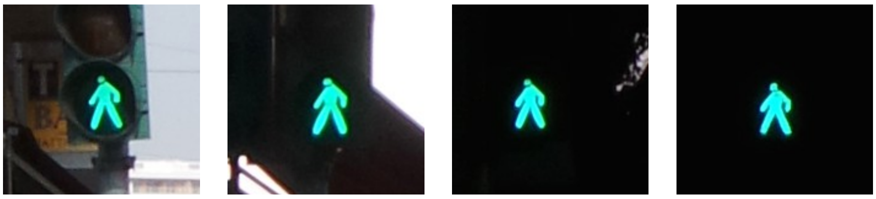

Figure 11: Details of four pictures taken in different illumination conditions.

shutter speed and aperture through the `Camera.Parameters` object[4]. It should be observed that, while the `Camera.Parameters` object is defined for all Android APIs up to level 21 (excluded), not all of its methods produce effects on all devices. Indeed, on most devices the methods to manually set ISO, shutter speed and aperture do not produce any effect and do not disable auto exposure. To the best of our knowledge, the only device that fully supports these APIs is the 'Samsung Galaxy Camera', which was used to collect the images used in the experiments (see Section 5).

Android 5.x offers a totally renewed set of APIs to access the camera and its parameters. The package containing the classes is called `Camera2` [5]. These classes offer several new APIs to control camera parameters and, based on our experience, these APIs are actually supported by most devices, including Nexus 5, which was used for the experiments.

A final comment on gamut spaces. The high variability in terms of both European standard ranges and actual measured chromaticities of the AOUs (see Table 3) turned out to be wider than the average image variance due to possible changes of gamut space in the acquisition device. Thus, varying the parameter settings (see Section 5) is sufficient to compensate this variance.

Figure 11 shows details of four pictures, each one representing a green AOU in a different illumination condition. The pictures were taken with the camera parameters described above. From left to right, the four light intensities are: very high, high, mid, and low. These results are examples of the stable acquisition (see Figures 1 and 2 for a visual comparison with automatic exposure).

[4] `http://developer.android.com/reference/android/hardware/Camera.Parameters.html`

[5] `https://developer.android.com/reference/android/hardware/camera2/package-summary.html`

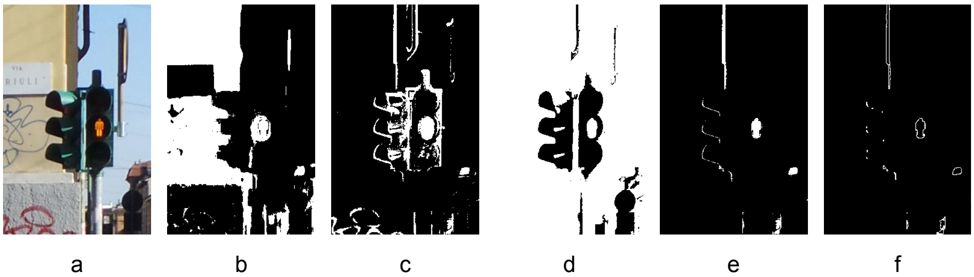

Figure 12: Extraction of candidates AOUs. (a) Portion of original image, (b) filter on H, (c) filter on S, (d) filter on V, (e) conjunction of filter results, (f) extracted contours.

3.3. Extraction of candidate active optical units.

After image acquisition, for each optical unit color c (i.e., green, yellow and red), *TL-recognizer* identifies a set of image portions, each one representing a candidate AOU. To achieve this, the proposed technique first applies a range filter and then groups contiguous pixels. This approach relies on the fact that AOUs have high luminosity values and are surrounded by regions with low luminosity values (i.e., the optical unit background).

The range filter is defined over the HSV image representation and is used to identify the pixels with high luminosity values (see Line 4 in Algorithm 1). A different filter is defined for each optical unit color c. The result of the application of the range filter is a binary image whose white pixels are segmented into blocks of contiguous pixels (see Line 5). This is obtained through the technique proposed by Suzuki and Abe [20]. The result is a list of contours, each one composed of a set of points.

Example 2. *Consider the portion of image shown in Figure 12a. Figure 12b shows the application of the range filter for the yellow optical unit color on the H channel. Figures 12c and 12d shows the same filter for the S and V channels, respectively. Details on the filter ranges are provided in Section 5. Figure 12e shows the logical conjunction of the previous three figures, i.e., the result of the range filter. Finally, Figure 12f shows the contours extracted from the image.*

3.4. Pruning of candidate active optical units.

After extracting the contours from the source image, the algorithm removes the contours whose geometrical properties are not compatible with

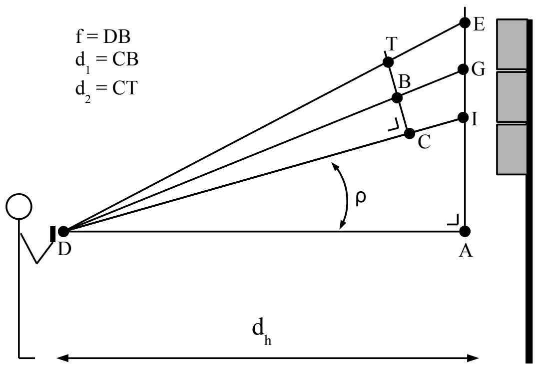

Figure 13: "Distance"

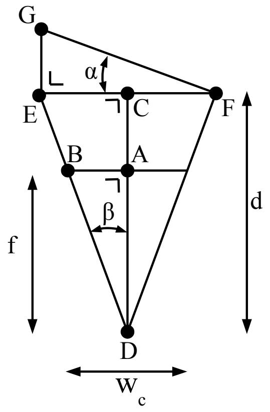

Figure 14: "Width"

those of an AOU. This pruning phase helps prevent false positives and it also improves computational efficiency, as it reduces the number of times the validation process needs to be run. Pruning is based on two properties: "distance" and "width".

The "distance" property is based on the idea that the optical units to be recognized should not be too far or too close from the user (see Section 2.4). To capture this intuition, each contour is assumed to be an AOU (whose size is known). Then, its distance along the horizontal and vertical axes from the device camera is computed. These distances are then compared with threshold values and the contour is discarded if the AOU is too close or too far away along any of the two axes. Property 2 shows how to compute the horizontal and vertical distances.

Property 2. *Let ρ be the device pitch angle, d_1 and d_2 the directed minimum and maximum vertical distances between the contour and the center of the image (in pixel), f the focal distance (in pixel), l_h the height of the optical unit lens (see Figure 13 for a graphical representation). The horizontal and vertical distances (d_h and d_v, respectively) between the device and the optical unit are:*

$$d_h = \frac{l_h \cdot \cos(\arctan(d_2/f) + \rho) \cdot \cos(\arctan(d_1/f) + \rho)}{\sin(\arctan(d_2/f) - \arctan(d_1/f))} \tag{2}$$

$$d_v = d_h \cdot \tan(\arctan(d_1/f) + \rho) \tag{3}$$

There are two aspects related to the "distance" property that are worth observing. First, the formulae are based on the contour height, which is computed after rotating the contour by the inverse of the horizon inclination. This makes the proposed technique 'rotation invariant' in the sense that it is not affected by accidental rotation of the device. The reason for using the height as the reference length is that, by using the device accelerometer, it is possible to compute the device pitch (i.e., the inclination with respect to the ground) that is then used to compensate for projection distortion. The second aspect is that, in practice, "distance" property checks the vertical size of the contour and discards the contours that are too small or too big. Indeed, small contours correspond to potential AOUs that are too distant from the user, hence not relevant for the recognition. Analogous reasoning can be applied for contours that are very large.

The "width" property is used to prune the contours whose width is not compatible with the width of an optical unit lens. Property 3 shows how to compute the width of the object represented by the contour. Note that distance d between the camera and the traffic light is easily computed from d_h and d_v.

Property 3. *Let w_c be the contour width, f the camera focal distance (in pixel), α the angular distance between the image plane and the plane of the optical unit lens and d the distance between the camera and the optical unit. The width of the object represented by the contour is:*

$$w = \frac{d \cdot w_c}{f \cdot \cos(\alpha)} \tag{4}$$

There is a major difference with respect to the computation of the "distance" property: the relative angle α between the image plane and the plane of the optical unit lens (see Figure 14) is not known. Consequently it is not possible to compute the exact width of the contour, but it is possible to bind it in a range. The minimum value of the range represents the case in which α is zero (i.e., the device camera is pointing directly towards the traffic light), while the maximum value represents the situation in which α is equal to the 'maximum rotation distance' (see Section 2.4). If the width of the optical unit lens (which is known) is not contained in the range, the contour is pruned.

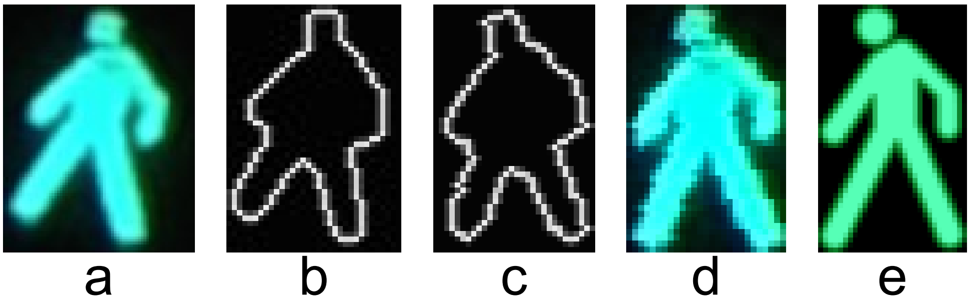

Figure 15: Validation of candidates AOUs. (a) Portion of original image, (b) Contour, (c) Rotated Contour, (d) Image Patch (rotated), (e) Template image.

3.5. Validation of active optical units.

Each contour that passes the pruning step has geometrical properties compatible with an AOU; still, it is not guaranteed that it actually represents an AOU. To validate a contour, the proposed solution extracts from the input image the image portion (called 'patch', in the following) corresponding to the contour minimum-bounding rectangle (MBR).

Note that the contour is rotated (see Algorithm 1 Line 8). For this reason, in theory, it should be necessary to apply the same rotation to the original image before extracting the patch. Since it is computationally expensive to rotate the entire image, the patch is rotated on-the-fly when it is constructed.

The patch is then resized to the same size as the template, which is a system parameter. Finally, the two figures (patch and template) are compared with the fast normalized cross-correlation technique [21], chosen as the technique to evaluate the similarity between two images. The patch is considered to be an active optical unit if the result of the comparison is larger than a given threshold T (see Line 16 in Algorithm 1). The methodology to select the threshold is described in Section 5.

Example 3. *Figure 15a shows a portion of an original image. Figures 15b shows the contour, as extracted during the extraction step, while 15c shows the rotated contour computed during the pruning step. Figure 15d shows the extracted patch. Note that the extracted patch is smaller than the template shown in Figure 15e (in the figure they are shown with the same size, but the patch has a smaller resolution). For this reason the patch is first resized to have the same size as the template and then the two images are compared. In this example, fast normalized cross-correlation returns a value of* 0.82 *that,* *as shown in Section 5 is larger than T, hence the contour is recognized as a green AOU.*

4. Algorithm improvements

In addition to the core recognition procedure described in Section 3, the proposed technique implements a number of improvements aimed at increasing the reliability of the results and computational performances.

4.1. Improving recognition of red and yellow AOUs

As shown in Section 5, the boundaries of the range filters for the red and yellow colors overlap. As a consequence, it is relatively frequent that a red AOU is confused with a yellow one, and vice versa.

To avoid this problem, the following optimization is introduced. The main loop starting at Line 2 (see Algorithm 1) is iterated for two colors only (instead of three): green and 'yellowRed', i.e., a single color representing both red and yellow AOUs. To distinguish between red and yellow AOUs, a procedure is run during the validation phase, after extracting the patch p (Line 13). This procedure counts, in the patch p, the number of pixels with a purely red hue ($160 \leq h \leq 179$) and those with a purely yellow hue ($10 \leq h \leq 30$)[6]. If the number of red pixels is larger than the number of yellow ones, the patch is then assumed to be red and is compared with the red template. Otherwise the patch is assumed to be yellow.

As shown in Section 5, this approach helps reducing the number of cases in which yellow and red AOUs are confused.

4.2. Improving computational performance

As shown in Section 5, the computation time of the base recognition algorithm is about 1s on a modern smartphone (with maximum image resolution). While a delay of about 1 second in the notification of the current traffic light color could be tolerable, an additional problem arose during preliminary experiments: it is challenging, for people with VIB, to point the device camera towards the traffic light. To find the correct position, users needs to rotate the device left and right while paying attention to the device feedback (audio or vibration). This requires a responsive system and a delay

[6]Henceforth hue scale is reported in $[0, 180)$.

of 1 second is not tolerable as it does not allow the user to find the traffic light position.

To speed up the computation, two different techniques are adopted: multi-resolution processing and parallel computation. Multi-resolution is based on the idea that the validation step requires the processing of images at a high resolution, while extraction and pruning steps are reliable (in terms of precision and recall) even when images are processed at a smaller resolution. Running these two steps with images at a smaller resolution significantly improves the performances. For this reason, a resized version of the acquired image is processed during the extraction and pruning steps. Then, during the validation step, the image patch p is extracted (see Line 13) from the acquired high-definition image. 'Resize factor' is the parameter that defines to what extent the original images is resized. Technically, the number of pixels on both sides of the original image is divided by 'resize factor'. As shown in Section 5, this optimization drastically reduces the computation time. However, large values of the resize factor negatively affect algorithm recall, so the value of the resize factor should be carefully tuned.

Since modern smartphones have multi-core CPUs, a natural approach to improve the performance of computational intensive operations is to adopt parallel computation. In particular, two pools of threads are used: one aimed at parallelizing the extraction process (Algorithm 1, Line 2), the other aimed at parallelizing the contours' processing (Algorithm 1, Line 6). The former pool has a number of threads equal to the number of colors, while the latter has a number of threads equal to the number of CPU cores.

5. Parameters tuning and experimental evaluation

Two main sets of experiments were conducted: one set, called 'computational-based' is aimed at tuning the system parameters and at quantitatively measuring the performances of *TL-recognizer*. The second set, called 'human-based' is aimed at qualitatively asserting the effectiveness of the proposed technique.

5.1. Experimental evaluation methodology and setting.

In order to ease the development of *TL-recognizer* and to guarantee reproducibility of the computational-based experiments, the following methodology was adopted: images of urban scenarios were recorded, each one with its associated information representing device orientation[7]. Each image was manually annotated with the position and the color of AOUs (if any). Finally, an Android app was implemented to read the stored images and to use them as input for *TL-recognizer*.

Two datasets of images each were created. The exposition of all the collected images has been chosen according to the methodology described in Section 3.2. The 'tuning' dataset (501 images), was used for debugging and parameters tuning, while the 'evaluation' dataset ($1,252$ images) was used for performance measurement. Both datasets are divided into four subsets, one for each of the illumination conditions defined in Section 3.2. Details are reported in Table 5. The two datasets of images are publicly available[8]. Note that some of the pictures (in particular with mid and low illumination conditions) were taken while it was raining and results are not affected by this weather condition.

Set	Light intensity	Number of images with			
		no AOU	green AOU	red AOU	yellow AOU
Tuning	Very High	62	21	22	22
	High	62	21	21	19
	Mid	62	21	21	22
	Low	62	21	21	21
Evaluation	Very High	75	62	45	37
	High	105	96	104	52
	Mid	64	78	109	59
	Low	120	51	118	77

Table 5: Composition of the two sets of images.

During the computer-based experiments, a number of parameters were measured, including: precision, recall, computation time and number of "R-Y errors", i.e., the number of times a yellow AOU is confused with a red AOU or vice versa. Note that, from the point of view of a person with VIB that is about to cross a road, a yellow AOU has the same semantic as a

[7]Henceforth, the term 'image' refers to the actual image with the associated device orientation information.

[8]`http://webmind.di.unimi.it/CVIU-TrafficLightsDataset`

red AOU i.e., the person should not start crossing. For this reason, when computing precision and recall, a R-Y error is still considered a true positive result. Note that, unless otherwise specified, **precision is always equal to one**, meaning that no traffic light is erroneously detected. Finally, note that computation time is measured excluding the time needed to acquire the input image.

To conduct human-based experiments *TL-recognizer* was implemented into a mobile application that collects live input from the camera and the accelerometer and that implements basic versions of the *TL-logic* and *TL-Navigation* modules. The application continuously runs *TL-recognizer* with the acquired frames and creates three messages for the user: 'not found', 'stop' and 'go': the first indicates that no traffic light was found, the second indicates that a red or yellow AOU was detected and the third one indicates that a green AOU was detected. To convey these messages, the application uses spoken messages (through the system text-to-speech synthesizer), two clearly distinguishable vibration patterns (for 'stop' and 'go' messages) and a visual message for subjects that are partially sighted (the entire screen becomes black, red or green).

The experiment involved 2 blind subjects and 2 low-visioned subjects (unable to see the traffic lights involved in the experiment). The experiments took place in different illumination conditions. All subjects have been trained for about one minute on how to use the application. Then, in a real urban intersection, subjects were asked to walk towards a crossroad and to determine when it was safe to start crossing in a given direction (straight, left or right) i.e., when a green traffic light appears right after a red one. For each attempt, a supervisor recorded whether the task was successfully completed and took note of any problem or delay in the process. Each subject repeated this task five times. Finally, the subjects were asked to answer a questionnaire.

For what concerns the devices used during the experiments, the images were collected with a Samsung Galaxy Camera with Android 4.1. Computer-based and human-based experiments were conducted with a Nexus 5 device with Android 5, which, with respect to a Galaxy Camera, has a faster CPU and is also more ergonomic for the subjects involved in the human-based tests[9].

[9]The choice of using a Galaxy Camera to collect images was driven by the fact that,

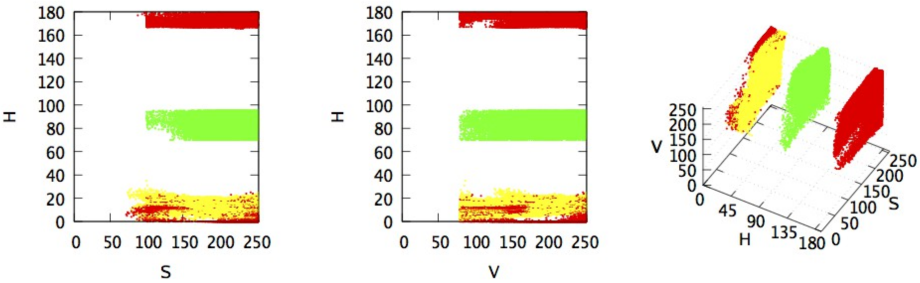

Figure 16: Pixels composing AOUs

5.2. Parameters tuning.

The recognition technique presented in Section 3 uses several system parameters that need to be tuned. Section 3.2 describes the tuning process of the parameters used for image acquisition. The tuning of other parameters that mainly affect system performance is described in the following.

One set of parameters defines the boundaries of the range filters (see Algorithm 1). To tune these values each pixel composing AOUs (if present) was sampled in the 501 pictures composing the tuning dataset. This was obtained with a semi-automated process: first, a few pixels were manually sampled, hence defining broad ranges. Then, by running the algorithm with these ranges, a set of contours representing the AOUs were extracted, together with contours representing other objects. Thanks to picture annotations, the contours representing AOUs were automatically identified and the values of all pixels included in these contours were stored. From this set of pixels white pixels (i.e., $v = 255$) and dark pixels were excluded.

The selected pixels are shown in Figure 16 where green, red and yellow dots represent a pixel for a green, red and yellow AOU, respectively. Given these results, the smallest ranges to include all pixels were defined . Results are shown in Table 6. Note that, since the yellowRed color lies on both sides of the hue circular axis, two range filters are defined and their disjunction yields the result.

Threshold T is another important parameter that requires to be tuned. The following methodology was adopted: the image processing algorithm was run for each image in the tuning dataset. For each extracted patch

at that time, this was the only available device supporting manual EV settings.

Optical unit color	H min	H max	S min	S max	V min	V max
Green	70	95	100	255	80	255
yellowRed (first)	0	25	100	255	80	255
yellowRed (second)	166	180	100	255	80	255

Table 6: Range filters boundaries.

(see Algorithm 1) the value of the normalized cross correlation was stored, together with a boolean value representing whether the patch is actually an AOU or not (this is derived from the annotations). Among all patches in all images in the tuning dataset, the larger cross correlation value for a patch that does not represent an AOU is 0.586. Threshold T is set to this value, hence guaranteeing, in the tuning dataset, a precision of 1.

Figure 17 shows the impact of the resolution on both recall and computation time. As expected, computation time decreases almost linearly, since most of the costly operations are linear in the number of pixels in the image. At the same time, recall slowly decreases when using images with up to 3 times less pixels (i.e., 1413×1884) that guarantee a recall of 0.887. With smaller images, recall decreases at a faster rate. For these reasons, images with a resolution of 1413×1884 were used in the tests. Note that, while in the tests the images are resized from their original size to 1413×1884, in the *TL-recognizer* prototype this operation is not necessary: indeed images are directly acquired at a similar resolution (i.e., 1536×2048) and this also significantly speeds-up the image acquisition process.

5.3. Impact of the algorithm improvements

With the basic version of the algorithm, the proposed technique incurs in the 'R-Y error' in 20 cases in the images in the tuning set. This means that, considering only the 168 images containing red and yellow AOU, the frequency of this error is above 10%. By using the improvement described in Section 4.1, the number of these errors is reduced by 75% with 5 errors and a frequency of less than 3%.

Figure 18 shows computation time and recall for different values of the resize-factor parameter. As expected, there is a trade-off between computation time and recall (this is very similar to what was observed for the resolution parameter). By observing the results shown in Figure 18 it is possible to conclude that value 3 is a good trade-off: computation time is halved

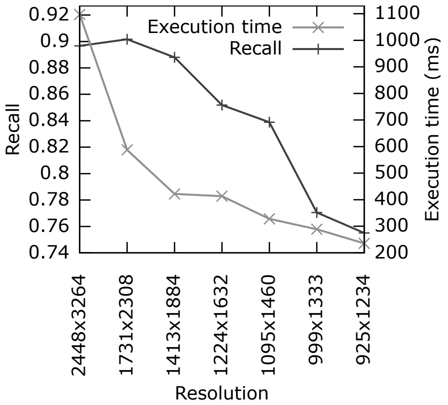

Figure 17: Impact of image resolution on computation time and recall.

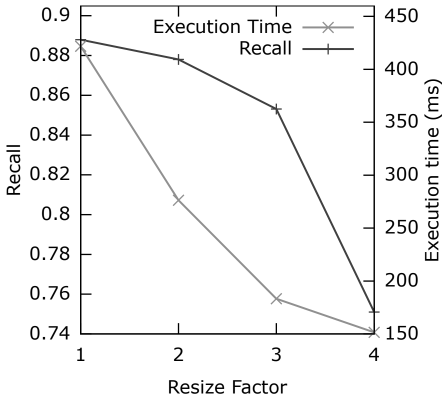

Figure 18: Impact of resize factor on computation time and recall.

(with respect to value 1), while recall decreases only by 0.03. For larger values (e.g., 4) there is no substantial improvement in the computation time, while recall decreases by more than 0.1.

Finally, it has been measured that with parallel processing computation time diminishes by about 40%: from an average computation time of 183ms to 113ms. Table 7 shows the system performance measured on the tuning dataset after having tuned the system parameters and adopting the algorithm improvements.

Testset	Precision	Recall	Computation time
Tuning	1	0.85	113ms
Evaluation	1	0.81	107ms

Table 7: Performances of *TL-recognizer*

5.4. Results with the evaluation testset

Table 7 shows the results obtained with the evaluation dataset. Performance results are very similar to those obtained with the tuning dataset.

While conducting the evaluation with the testset it has been observed that computation time is influenced by the total number of contours that are processed. For example, images with an irregular background (like Figure 19) take much longer to compute than average images. For example, Figure 20

Figure 19: Frame in a sunny day.

Figure 20: Contours extracted from Figure 19

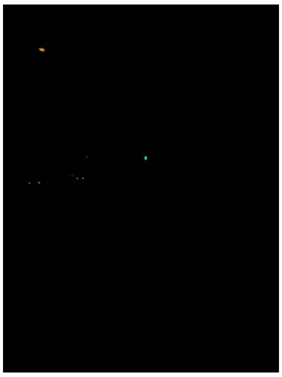

Figure 21: Frame during night.

shows the contours extracted from Figure 19: the bright background behind the trees results in more than 8000 contours to be processed. Clearly the great majority is discarded thanks to 'distance' and 'width' constraints, but still 80 of them need to be validated. While the overall result is correct (no traffic light is detected), the computation time for this frame is more than 500ms, about 5 times higher than the average time.

The above observation raises a more general question: how does computation time vary in the different illumination conditions? In sunny days it is more likely to have bright surfaces that generate a high number of contours, like in Figure 19. Indeed, the average computation time with high light intensity is 196ms. Vice versa, with low light intensity (e.g., at night), since fixed camera parameters are used, the input image is almost entirely black, with the exception of traffic lights and other sources of light, like street lamps and car beacon lights. For example, in Figure 21 a single contour is extracted for the green color (there is a small green AOU in the center of the image) and 5 contours are extracted for the 'yellowRed' color (in the figure, in addition to the green AOU, there are 5 small bright dots corresponding to two car beacon lights and a street lamp). Hence, with low light intensity, the computation time is 52ms, on average. In the two intermediate illumination conditions i.e., high and mid light intensities, the average computation times are 124ms and 90ms, respectively.

5.5. Results of the human-based evaluation

Overall, all subjects have been able to successfully complete the assigned tasks. The only exception was with the first attempt made by the first subject: since he was pointing the camera too high up and almost towards the sky, the traffic light was always out of the camera field of view. The problem was solved by simply explaining to the subject how to correctly point the camera. In the following experiments with the other subjects this was explained during the training phase. Note that this problem could also be solved by monitoring the pitch angle and by warning the user if the he/she is pointing too high or too low.

During this experiment it has been observed that the two blind subjects needed a slightly longer time (up to about 5 seconds) to find the traffic light. This is due to the fact that they could not precisely predict where the traffic light was and hence needed to rotate left and right until the traffic light entered the camera field of view. On the contrary, the two partially sighted subjects managed to find the traffic light almost instantaneously even if they could not see it. One possible motivation is that the two partially sighted subjects had a better understanding of their current position with respect to the crossroad and a more developed ability to predict the position of the traffic light.

For what concerns the questionnaire, all subjects agree that the application is easy to use and useful. There are some comments that are worth reporting. One subject observes that this application would be very useful because some traffic lights are still not equipped with acoustic signal and, even if they are, in some cases they are not working properly and in other cases it takes some time to find the button to activate the signal (in Milan acoustic traffic lights need to be activated by a button positioned on the traffic light pole). Another subject observes that he would use this application only when an acoustic traffic light is not available, because it is not convenient to hold the device in one hand while holding the white cane on the other one. All subjects agree on the fact that the vibration pattern is the best way to get the message. Indeed, audio messages can be hard to listen due to traffic noise, as observed by one subject. Visual instructions are also not practical, according to both low-visioned subjects, as they are not always clearly visible.

6. Conclusions and future work

This paper presents *TL-recognizer*, a system to recognize pedestrian traffic lights aimed at supporting people with visual impairments. The proposed technique, in addition to the pure computer vision algorithms, implements a robust method to acquire images with proper exposure. The aim is to guarantee robust recognition in different illumination conditions. Experimental results show that *TL-recognizer* actually achieves this objective and is also efficient, as it can run several times a second on existing smartphones. Positive results were also obtained with a preliminary evaluation conducted on subjects with VIB: they were able to detect traffic lights in different illumination conditions.

In future work it would be interesting to integrate *TL-recognizer* with a video tracking system, possibly based on the use of accelerometer and gyroscope. Also, user interaction should be carefully studied, with the aim of providing all the required information without distracting the user from its surrounding environment. The design of effective user interfaces will become even more challenging if *TL-recognizer* is integrated with other solutions that collect and convey to the user contextual information, for example, the current address or the presence of pedestrian crossings.

Regarding exposure robustness, improvements could be derived from the adoption of HDR techniques to extend the acquisition dynamic range. In this case tests should be performed to verify the trade-off between reliability gains and computational costs.

In order to ease the adoption of the proposed technique in different countries, a (semi) automated technique can be implemented to tune the parameters. This could be possibly based on a learning technique that gradually tunes the parameters in order to adapt to different contexts.

An effort will also be devoted to the development of a commercial product based on *TL-recognizer*. Indeed, it could be possible to integrate this software with *iMove*, a commercial application that supports orientation of people with VIB developed by EveryWare Technologies. This will require tuning the system in order to detect pedestrian traffic lights in countries other than Italy. Also, in the near future it will be possible to implement *TL-recognizer* as an application for wearable devices (e.g., glasses). This will solve one of the main design issues: the fact that the user needs to hold the device in one hand.

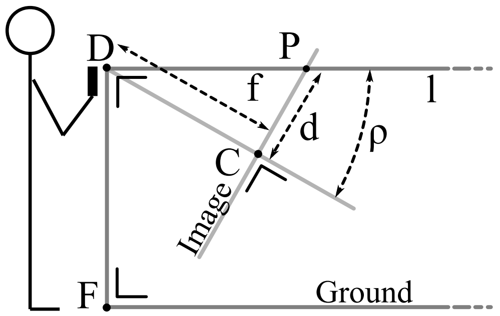

Figure A.22: Horizon computation, lateral view.

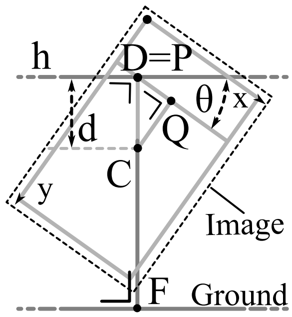

Figure A.23: Horizon computation, frontal view.

Appendix A. Proof of formal results

Appendix A.1. Proof of Property 1

The notation used in the proof refers to Figures A.22 and A.23.

Proof. The ground is approximated to an infinite plane. Thus, line l, which points from the device camera to the horizon, is parallel to the ground plane and angle $\widehat{FDP}$ is $\pi/2$.

We define h through its angle θ and a point P where h passes. The general form is:

$$\sin(\theta)x + \cos(\theta)y + (\sin(\theta)P_x + \cos(\theta)P_y) = 0 \tag{A.1}$$

We now show how to compute θ and P.

Consider Figure A.22. Let P be the point where the image plane intersects line l. Thus, point P lies on the horizon line h and P is inside the image. Also, since point D is the device, segment $\overline{DC}$ is perpendicular to $\overline{CP}$. Hence PCD is a right triangle. Since CD is the focal distance f and angle PDC is the device pitch angle ρ, the distance (in pixel) between the image center C and point P is $d = f \cdot \tan(\rho)$.

In the image plane, the device roll θ is the inclination of the device's x axis with respect to the ground plane. Since the horizon line h is parallel to the ground plane, θ is also the inclination of the horizon in the image. Consider Figure A.23. Let Q be the projection of C on the line parallel to the x axis (in the device reference system) that passes through P. Since $\widehat{CPQ} + \theta = \pi/2$, it follows that $\widehat{PCQ} = \theta$. Since the distance d is known, then the distance between point P and point C along the x axis is $d_x = \overline{PQ} = d \cdot \sin(\theta)$. Analogously, the distance between point P and point C along the y axis is $d_y = \overline{CQ} = d \cdot \cos(\theta)$. Thus, the coordinates of point P are $P =< C_x - \sin(\theta)d, C_y - \cos(\theta)d >$.

Finally, substituting d and P in Equation A.1 we obtain:

$$\sin(\theta)x - \cos(\theta)y - \sin(\theta)(C_x + \tan(\rho)\sin(\theta)f) + \cos(\theta)(C_y + \tan(\rho)\cos(\theta)f) = 0 \tag{A.2}$$

$\square$

Appendix A.2. Proof of Property 2.

To ease the reading of the proof, please refer to Figure 13. Note that, in the figure, points B and T are above point C. Since d_1 is defined as the *directed* vertical distances between C and B, in case B is below C, the value of d_1 is negative. The same holds for d_2. Under this consideration, it is easily seen that the following proof holds when both B and T are below C and also when B is below C and T is above C.

Proof. Since $d_h = DA$, by considering triangle DAG, it holds that

$$d_h = DA = GD \cdot \cos(\widehat{GDA}) \tag{A.3}$$

Thesis easily follows by showing that

$$GD = \frac{l_h \cdot \sin(\pi/2 - \arctan(\frac{d_2}{f}) - \rho)}{\sin(\arctan(\frac{d_2}{f}) - \arctan(\frac{d_1}{f}))} = \frac{l_h \cdot \cos(\arctan(\frac{d_2}{f}) + \rho)}{\sin(\arctan(\frac{d_2}{f}) - \arctan(\frac{d_1}{f}))} \tag{A.4}$$

and

$$\widehat{GDA} = \arctan(\frac{d_1}{f}) + \rho \tag{A.5}$$

For what concerns GD, by considering triangle GED we have:

$$GD = \frac{EG \cdot \sin(\widehat{GED})}{\sin(\widehat{EDG})} \tag{A.6}$$

EG is the lens height l_h given in input.

$\widehat{EDG}$ is equal to $\widehat{TDB}$ that, in turn, is equal to $\widehat{TDC} - \widehat{BDC}$. Since TDC and BDC are right triangles, it holds that $\widehat{TDC} = \arctan(\frac{CT}{CD})$ and $\widehat{BDC} = \arctan(\frac{BC}{DC})$ where $CT = d_2$, $BC = d_1$ and $CD = f$. Hence:

$$\widehat{EDG} = \widehat{TDB} = \arctan(\frac{d_2}{f}) - \arctan(\frac{d_1}{f}) \tag{A.7}$$

For what concerns $\widehat{GED}$, by considering right triangle EDA we have that:

$$\widehat{GED} = \widehat{AED} = \pi/2 - \widehat{EDA} = \pi/2 - (\widehat{EDI} + \widehat{IDA}) \tag{A.8}$$

where $\widehat{EDI} = \widehat{TDC}$ and $\widehat{IDA}$ is the device pitch ρ. So, it follows:

$$\widehat{GED} = \widehat{AED} = \pi/2 - \arctan(\frac{d_2}{f}) - \rho \tag{A.9}$$

For what concerns $\widehat{GDA}$, it is equal to $\widehat{GDI} + \widehat{IDA}$ where $\widehat{GDI} = \widehat{BDC}$ and $\widehat{IDA}$ is the device pitch ρ. Hence:

$$\widehat{GDA} = \arctan(\frac{d_1}{f}) + \rho \tag{A.10}$$

Finally, we show the value of $d_v = AG$. Consider the right triangle ADG where $AD = d_h$ and $\widehat{GDA}$ is known (see above). Consequently,

$$d_v = AG = AD \cdot \tan(\widehat{GDA}) = d_h \cdot \tan(\arctan(\frac{d_1}{f}) + \rho) \tag{A.11}$$

□

Appendix A.3. Proof of Property 3.

Notation used in the following proof refers to Figure 14.

Proof. Consider right triangle ADB: $\widehat{ADB} = \arctan(AB/AD)$ where $AB = w_c/2$ and $AD = f$.

Now consider right triangle CDE: $CE = CD \cdot \tan(\widehat{CDE})$ where $CD = d$ and $\widehat{CDE} = \widehat{ADB}$. Hence:

$$EF = 2 \cdot CE = \frac{d \cdot w_c}{f} \tag{A.12}$$

Finally, consider right triangle FEG:

$$w = GF = EF/\cos(\alpha) = \frac{d \cdot w_c}{f \cdot \cos(\alpha)} \tag{A.13}$$

□

[1] Y. Kim, K. Kim, X. Yang, Real time traffic light recognition system for color vision deficiencies, in: Proceedings of the Fourth International Conference of Mechatronics and Automation, IEEE Computer Society, 2007, pp. 76–81.

[2] M. Omachi, S. Omachi, Traffic light detection with color and edge information, in: Proceedings of the Second International Conference on Computer Science and Information Technology, IEEE, 2009, pp. 284–287.

[3] C. C. Chiang, M. C. Ho, H. S. Liao, A. Pratama, W. C. Syu, Detecting and recognizing traffic lights by genetic approximate ellipse detection and spatial texture layouts, International Journal of Innovative Computing, Information and Control 7 (2011) 6919–6934.

[4] S. Sooksatra, T. Kondo, Red traffic light detection using fast radial symmetry transform, in: Proccedings of the International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, IEEE, 2014, pp. 1–6.

[5] M. Diaz-Cabrera, P. Cerri, J. Sanchez-Medina, Suspended traffic lights detection and distance estimation using color features, in: Proceedings of the Fifteenth International Conference on Intelligent Transportation Systems, IEEE, 2012, pp. 1315–1320.

[6] M. Diaz-Cabrera, P. Cerri, P. Medici, Robust real-time traffic light detection and distance estimation using a single camera, International Journal on Expert Systems with Applications 42 (2015) 3911–3923.

[7] C. Wang, T. Jin, M. Yang, B. Wang, Robust and real-time traffic lights recognition in complex urban environments, International Journal of Computational Intelligence Systems 4 (2011) 1383–1390.

[8] Z. Cai, M. Gu, Y. Li, Real-time arrow traffic light recognition system for intelligent vehicle, in: Proceedings of the Sixteenth International Conference on Image Processing, Computer Vision and Pattern Recognition, IEEE, 2012, pp. 848–854.

[9] A. Almagambetov, S. Velipasalar, A. Baitassova, Mobile standards-based traffic light detection in assistive devices for individuals with color-

vision deficiency, Transactions on Intelligent Transportation Systems 16 (2015) 1305–1320.

[10] R. de Charette, F. Nashashibi, Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates, in: Proceedings of the International Symposium on Intelligent Vehicles, IEEE, 2009, pp. 358–363.

[11] V. Ivanchenko, J. Coughlan, H. Shen, Real-time walk light detection with a mobile phone, in: Proceedings of the Twelfth International Conference on Computers Helping People with Special Needs, Springer-Verlag, 2010, pp. 229–234.

[12] J. Roters, X. Jiang, K. Rothaus, Recognition of traffic lights in live video streams on mobile devices, Transactions on Circuits and Systems for Video Technology 21 (2011) 1497–1511.

[13] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, H. Chen, The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles, in: Proceedings of the International Symposium on Intelligent Vehicles, IEEE, 2010, pp. 431–435.

[14] P. Angin, B. Bhargava, S. Helal, A mobile-cloud collaborative traffic lights detector for blind navigation, in: Proceedings of the Eleventh International Conference on Mobile Data Management, IEEE Computer Society, 2010, pp. 396–401.

[15] R. de Charette, F. Nashashibi, Traffic light recognition using image processing compared to learning processes, in: Proceedings of the 22nd International Conference on Intelligent Robots and Systems, IEEE, 2009, pp. 333–338.

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

[17] B. Ullman, N. Trout, Accommodating pedestrians with visual impairments in and around work zones, 2140, Transportation Research Board of the National Academies, 2009, pp. 96–102.

[18] S. Mascetti, L. Picinali, A. Gerino, D. Ahmetovic, C. Bernareggi, Sonification of guidance data during road crossing for people with visual

impairments or blindness, International Journal of Human-Computer Studies (2015).

[19] European Standard EN 12368:2006 on "traffic control equipment - signal head", Technical Committee CEN/TC 226 "Road equipment", 2006.

[20] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, International Journal on Computer Vision, Graphics, and Image Processing 30 (1985) 32–46.

[21] J. P. Lewis, Fast normalized cross-correlation, International Journal on Vision interface 10 (1995) 120–123.