A Smartphone-Based Obstacle Detection and Classification System for Assisting Visually Impaired People

Ruxandra Tapu, Bogdan Mocanu, Andrei Bursuc, Titus Zaharia
IT/Télécom SudParis, ARTEMIS Department, UMR CNRS MAP5 8145, Evry, France
email: {ruxandra.tapu, titus.zaharia}@telecom-sudparis.eu

Abstract

In this paper we introduce a real-time obstacle detection and classification system designed to assist visually impaired people to navigate safely, in indoor and outdoor environments, by handling a smartphone device. We start by selecting a set of interest points extracted from an image grid and tracked using the multiscale Lucas - Kanade algorithm. Then, we estimate the camera and background motion through a set of homographic transforms. Other types of movements are identified using an agglomerative clustering technique. Obstacles are marked as urgent or normal based on their distance to the subject and the associated motion vector orientation. Following, the detected obstacles are fed/sent to an object classifier. We incorporate HOG descriptor into the Bag of Visual Words (BoVW) retrieval framework and demonstrate how this combination may be used for obstacle classification in video streams. The experimental results demonstrate that our approach is effective in image sequences with significant camera motion and achieves high accuracy rates, while being computational efficient.

1. Introduction

In the difficult context of day to day life of visually impaired/blind persons, the white cane still represents the most popular tool used for obstacle detection, provided that the encountered locations have been visited before and memorized. In an unfamiliar setting they completely depend on other humans to reach the desired destination [1]. The ability to detect obstacles in video streams representing highly dynamic urban scenes is a major requirement of any system designed to ensure the autonomous navigation [2].

In this paper we propose and validate a robust technology designed to alert users by the presence of static/dynamic obstacles few meters around them and to provide guidance/assistance in urban environments, in both outdoor and indoor spaces.

In the first phase, our system detects static and dynamic obstacles in video streams. First, only a sub-grid of interest points is selected and tracked using the multiscale Lucas-Kanade algorithm. Then, camera/background motion is identified by recursively applying the RANSAC algorithm on the set of matched keypoints. The outliers are merged into clusters using the associated motion vectors. Finally, the degree of risk of the obstacles is computed by classifying them as urgent or normal according to their movement and relative distance to the video camera. For validation we focused on challenging sequences that contain complex motions, possibly with high amplitude and sudden changes in the background caused mainly by the subject displacement.

In the second phase we considered the issue of obstacle classification. The image patches corresponding to the detected obstacles are now resized, while preserving the aspect ratio, and divided into 128 cells of 8 x 8 pixels. From each cell we extract an interest point that is characterized using an adaptation of HoG (*Histogram of Oriented Gradients*) descriptor that captures the object structure from its surrounding regions. Next, we generate a visual codebook using the k-means clustering algorithm and map the image patches to their nearest visual word.

At the system level the novelty lies into a real-time application working independently on a regular smartphone. In this context our framework provides a low-cost, non-intrusive and simple system for blind navigation.

2. Related work

Nowadays, most of the commercial devices and software products, designed to provide mobility assistance, rely mainly on the Global Positioning System (GPS). However, these solutions are not always reliable due to the low accuracy, signal loss and impossibility to work in indoor environments [3]. In urban areas with high density of buildings the accuracy error of GPS sensors can reach up to 40 metres, which makes it very difficult for blind and visually impaired persons to find their destination.

Computer vision-based approaches, referred as ETA, (*Electronic Travel Aids*) constitute a promising alternative to solve such problems.

One of the first papers was presented by Hesch *et al.* [4]. Authors propose a fusion of sensor modalities (*e.g.*

foot-mounted pedometer, 3-axis gyroscope and 2D laser scanner) to help indoor localization. By means of visual Simultaneous Localization and Mapping (SLAM) techniques [5], [6] it is possible to build an incremental map of the environment, providing simultaneously temporal location and spatial orientation of the user. In [7], a stereo vision system that estimates a 3D map of vicinity through a 6 degree of freedom egomotion algorithm is introduced. The resulted map is then used to detect head-level obstacles during the user navigation. A head-mounted stereo vision system is depicted in [6]. By combining the visual odometry and feature based metric-topological SLAM they create a 3D map representation around the vicinity in order to detect obstacles and establish a waking path. A dense optical flow for visual SLAM application is introduced in [8]. The proposed system is designed to work in real-world dynamic environments with various, independent moving objects. Also, in [9] using a SLAM system combined with a dense optical flow an obstacle detection algorithm is proposed. The authors claim that the technique helps improving performance when detecting moving obstacles and pedestrians in crowded areas.

There are some approaches [10] and [11] on reactive obstacle avoidance, inspired from robotic platforms, that try to facilitate navigation by providing linear and angular velocity information. However, in the case of visual impaired users the path planning and navigation speed are provided by the person. So, the above techniques are not appropriate as warning systems to assist the user.

As we decided to use the video camera existent on a smartphone we review in the following part the methods focused on monocular cameras. 3D structure recognition from monocular images can be used to detect obstacles by computing the relative distance between the camera and an object. Among the existent approaches the most salient one is the structure from motion (SFM) method introduced in [12] that determines a correspondence between frames and develops a 3D scene using matchers that satisfy the epi-polar constraints. At the end other images are added to refine the scene using bundle adjustment. The technique is sensitive to important camera movement but also has a relatively high computational cost. In order to reduce the processing cost a real-time online SFM method is proposed in [13]. The system tries to determine the trajectory of a monocular camera by recovering the 3D structure of the landmarks in the scene. One limitation of the method is related to the required frame rate (30Hz), necessary for ensuring that the landmarks can be tracked within a small searching window. In order to achieve the real-time constraint the authors also limit the number of landmarks. When there are multiple moving objects in the scene, traditional SFM techniques usually fail. In such cases, some approaches [14] propose to cluster in different groups the correspondences between images and characterize them by different fundamental matrixes. However, this approach is still at preliminary stage and it is still impractical on light mobile devices.

A different approach for detecting static and moving obstacles is given by the two-dimensional motion analysis which does not involve 3D reconstruction. In [15], if the camera is moving smoothly, at constant speed and straight forward the authors can determine the time to contact and the divergence from the focus of expansion. When the movement is irregular it becomes difficult to determine the depth from the 2D flow and the technique fails.

To our very best knowledge, the only paper addressing the problem of obstacle detection by using monocular cameras embedded on smartphones is introduced in [16]. The system is able to detect on-floor objects with high precision by employing color information (RGB color histogram). However, the approach is evaluated solely in in-door scenarios. In addition, some constraints regarding the position of the video camera (*e.g.* 45 degrees tilt angle with respect to the floor) need to be imposed and the condition of hand-frees [17], which is an important constraint imposed by the user, is violated.

The main contribution of the present paper concerns a novel obstacle detection and classification technique that works in real time and helps the user in avoiding and recognizing static and dynamic objects. Our system is low-cost, ultra-portable and non-intrusive. In addition, is able to detect obstacles situated at random heights in the user proximity.

3. Proposed approach

The proposed obstacle detection and classification techniques are detailed in the following sections.

3.1. Obstacle/moving object detection

The obstacle detection algorithm consists of the following steps (Figure 1):

Step 1: *Interest points extraction* – As we do not consider any *a priori* information about the shape or texture of the obstacles, we decided to use a grid of points regularly sampling the video frame. Let us mention that initially, we have considered SIFT [18] and SURF [19] interest points for sampling the video scenes. However, we have observed that the number of interest points belonging to the obstacle is in general significantly lower than the one corresponding to the background Furthermore, for less textured regions or low resolution videos SIFT/SURF detectors usually extract none or few interest points. In addition, when the background changes fast, the number of interest points and their neighbors can be significantly different between two successive frames. Finally, the computational complexity of such interest point extractors/descriptors becomes important when developing real-time applications.

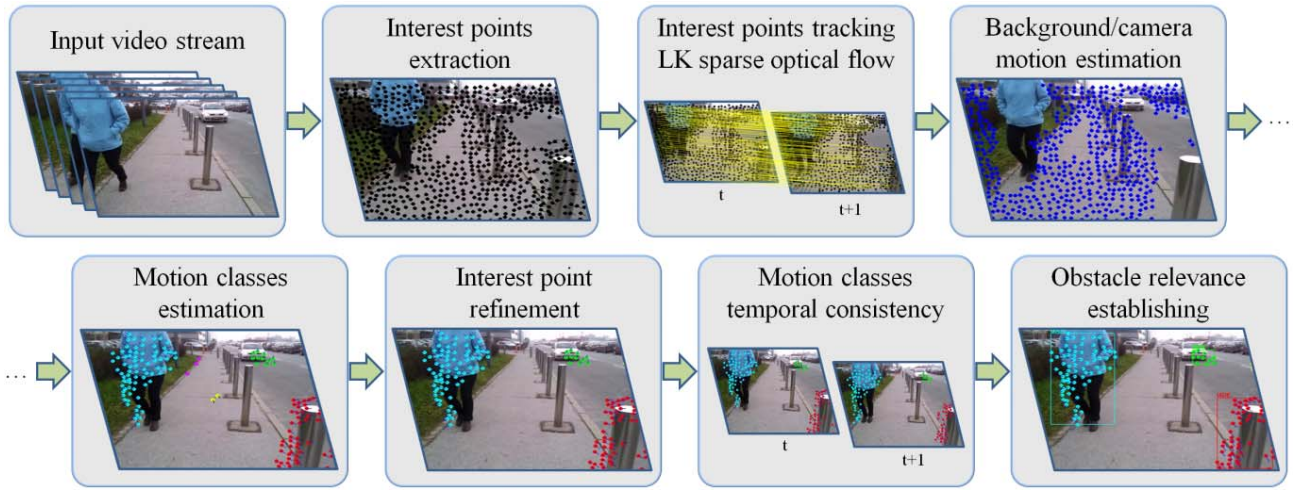

Figure 1: The proposed obstacle detection framework.

For all these reasons, we have adopted a regular grid sampling strategy.

In our algorithm the grid step is defined as: $\Gamma = \left\{\frac{W \cdot H}{N}\right\}$, where W and H are the dimensions of the image and N is the maximum number of points. The value of parameter N is important because it controls the balance between the processing time and detection accuracy. So, in order to limit the computational cost and obtain high precision we have set N to 1000.

Step 2: *Interest points tracking* - When detecting obstacles an essential feature is to determine the displacement of the interest points (motion vectors). We used here the multiscale Lucas-Kanade algorithm (LKA) [20]. The LKA method has some drawbacks: it does not ensure spatial or brightness consistency. We could have applied a significantly more powerful technique such as [21] that obtains better estimations and is robust to abrupt changes of illumination. However, this algorithm features higher computational cost and our goal is to obtain a relatively good estimation of the motion in the video sequence, rather than a highly accurate but also costly one.

The feature point selection method presented in *Step1* is applied on the first frame of a video sequence in order to initialize the LKA tracking procedure. The same procedure can be used locally to reinitialize the tracker in areas where such an action is required (*e.g.* when obstacles disappear or other/new objects appear).

Let $p_{1i}(x_{1i}, y_{1i})$ be the i^{th} keypoint in the first image and $p_{2i}(x_{2i}, y_{2i})$ be its correspondent in the successive frame. The associated motion vectors (v_{ix}, v_{iy}), expressed in polar coordinates with magnitude $(D_{i(1,2)})$ and angle of motion $(\theta_{i(1,2)})$ are also computed in this step:

$$v_{ix} = x_{2i} - x_{1i}; v_{iy} = y_{2i} - y_{1i} \quad (1)$$

$$D_{i(1,2)} = \sqrt{{v_{ix}}^2 + {v_{iy}}^2}, i = \overline{1,n} \quad (2)$$

$$\theta_{i(1,2)} = arcos \frac{v_{ix}}{D_{i(1,2)}}, \theta \in [0,2\pi] \quad (3)$$

where n is the total number of tracked points.

Step 3: *Camera/background motion estimation* – The set of tracked interest points is used to determine the global geometric transform, between two successive frames by considering a homographic motion model. A mapping from $IP^2 \rightarrow IP^2$ is a projectivity if and only if there exists a non-singular 3 x 3 matrix **H** such that any point $\boldsymbol{x}$ in IP^2 is mapped into $\boldsymbol{H} \cdot \boldsymbol{x}$. With the help of the RANSAC (*Random Sample Consensus*) [22] algorithm we determined the optimal homographic matrix **H**.

Based on the matrix **H**, for a given point $p_{1i}[x_{1i}, y_{1i}, 1]^T$ expressed in homogenous coordinates, its estimated position $p_{2i}^{est}[x_{2i}^{est}, y_{2i}^{est}, 1]^T$ is determined as:

$$\begin{bmatrix} x_{2i}^{est} \\ y_{2i}^{est} \\ w \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \cdot \begin{bmatrix} x_{1i} \\ y_{1i} \\ 1 \end{bmatrix} \quad (4)$$

where:

$$w = 1/(h_{20} \cdot x_{2i}^{est} + h_{21} \cdot y_{2i}^{est} + h_{22}) \quad (5)$$

The estimation error is defined as the difference between the estimated and the actual position of the considered interest point, as described in equation (6):

$$\in (p_{1i}, \boldsymbol{H}) = \|p_{2i}^{est} - p_{2i}\| \quad (6)$$

Ideally $p_{2i}^{est}[x_{2i}^{est}, y_{2i}^{est}, 1]^T$ should be as closely as possible to$p_{2i}[x_{2i}, y_{2i}, 1]^T$. In the case where the estimation error $\in (p_{1i}, \boldsymbol{H})$ is inferior to a predefined threshold E, the corresponding keypoints are marked as belonging to camera/background motion. The outliers, *i.e.* keypoints with an estimation error $\in (p_{1i}, \boldsymbol{H})$ exceeding E are considered to belong to foreground objects.

In our experiments, the background/foreground separation threshold E was set to 2 pixels (Figure 1).

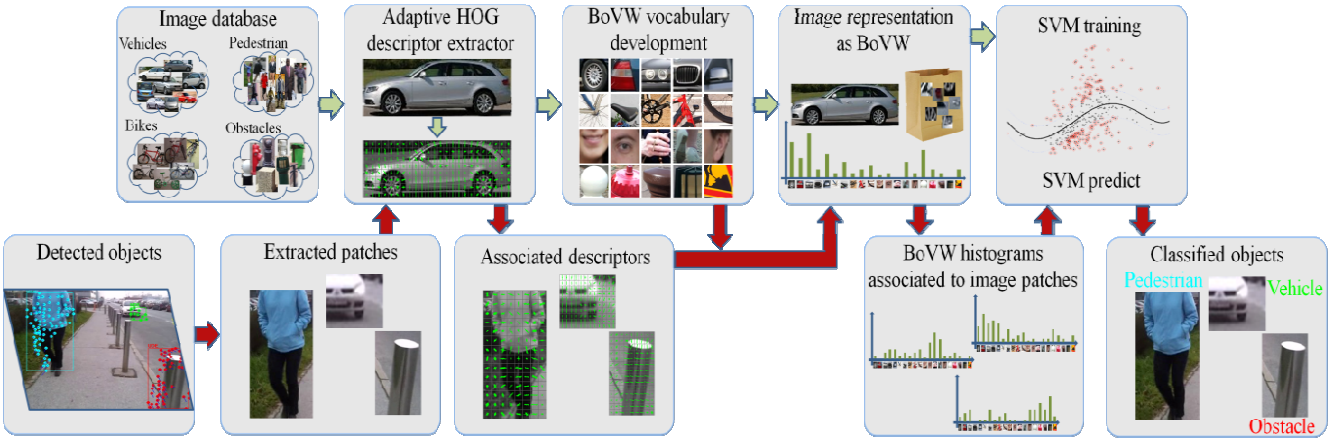

Figure 2: Object classification framework.

Step 4: *Motion classes estimation* - In practice, multiple moving objects are present in the scene. Also, even the static obstacles situated in the foreground of a moving camera act as objects in motion relatively to the background, due to the phenomenon of foreground apparent motion. In this case, we determine a new subset of points composed by all the outliers obtained at *Step 3* that we want to group into clusters in an agglomerative manner.

The basic principle behind agglomerative techniques is to consider each individual point as a cluster and then successively reduce the number of classes by merging the two closest clusters until all points are assigned to a category [23]. The key operation of the proposed algorithm is the proximity computation between two interest points that are classified into clusters in the following phases:

- *Phase I* - The motion vectors are sorted in descending order based on the number of occurrences of the corresponding motion vector angle. For the first interest point, of the current list, a new cluster is formed (MC_i) having as centroid its motion vector angular value (θ_c);

- *Phase II* - For all the other interest points not assigned to any motion class, we compute the angular deviation:

$$Dev(\theta_i, \theta_c) = |\theta_i - \theta_c| \quad (7)$$

If $Dev(\theta_i, \theta_c)$ is beyond a predefined threshold Th_θ and the motion magnitude is equal with the cluster centroid then the current point will be grouped into the MC_i cluster. Otherwise, a new motion cluster is created. For the remaining outliers, the process is applied recursively until all points belong to a motion class. In our work, the threshold Th_θ has been set to 15 degrees.

Step 5: *Interest point refinement* – For all the interest points included in the motion classes we applied next the *k-NN* algorithm [24], in order to verify that their assignment to the current class is not caused by an erroneous clustering. For the current point we determine its *k* nearest neighbors based on the Euclidian distance. If at least half of the detected points do not belong to the same motion class then this point is eliminated from the motion cluster.

Step 6: *Motion classes' temporal consistency:* It is very common in the object detection/tracking tasks that interest points cannot be tracked during the entire sequence. In particular, partial occlusion, noise and image distortion can lead to incomplete trajectories. Pose, illumination or appearance changes can also result in losses of point trackers. In order to avoid errors in keypoints classification and obtain coherent objects over the entire video stream we propose the reinforcement of the process of estimating the motion classes by adopting a multi-frame fusion scheme. Thus, by memorizing the trajectory of a motion class centroid over a number of *N* past frames, we predict the novel position of the object using the average speed, compensated with the current camera/background movement.

$$p_i(x_i, y_i, t_i) = p_i(x_i, y_i, t_{i-1}) + \sum_{j=1}^{N} v_j(v_{jx}, v_{jy}, t_j)/N - \\ - p_i^{est}(x_i^{est}, y_i^{est}, t_i) \quad (8)$$

Step 7: *Obstacle relevance establishing* – Once the obstacles are identified, we need to determine their degree of danger to the user and classify by their position and relative direction to the video camera. An object is labeled as approaching (AP) if its associated direction points to the camera focus, otherwise it is considered as de*parting (DE)*. Using a polar grid projected on the image, an obstacle is marked *urgent (U)* if it is situated in the proximity of the blind person otherwise it is categorized as *normal (N)*.

3.2. Obstacle/moving object classification

This section gives an overview of our object recognition (Figure 2) technique which can be summarized in the following steps:

Step 1: *Adapted HOG descriptor extraction* – The standard version of the HOG descriptor [25] is computed by dividing the image (*I(x,y)*) into cells that accumulate in local 1-D histogram of gradient directions over the considered pixels of the cell. In order to offer invariance to illumination changes, the local histogram energy is computed over image blocks and used to normalize all cells in the block. As the HOG descriptor was developed for human detection, the authors propose to use an analysis window normalized to 64 x 128 pixels. For other types of objects different image sizes are required (*e.g.,* for cars: 104 x 56 pixels; for bicycles: 120 x 80) [26].

Since our system is designed to work as a real-time application, we cannot afford to test multiple different sizes of the detected object (patch) in order to verify which classifier offers the best results and then make a decision. In order to avoid these problems we propose not constraining the patch size but to limit the total number of cells for which we compute the HOG descriptor. In this case, the image is resized but the aspect ratio is preserved and the descriptor needs to be extracted only once.

In our experiments we have considered the cell dimension of 8 x 8 pixels with 18 orientation bins. The maximum number of cells is fixed to 128.

Step 2: *BoVW vocabulary development (Visual vocabulary computation):* In a typical BoVW framework the interest points from a set of images are first detected and represented using a visual descriptor [27]. Following, an unsupervised grouping is performed over the entire set of descriptors to generate *k* clusters. The corresponding centroid for each such cluster is called a visual word and the codebook of visual words represents the visual vocabulary.

In the case of our framework, the pixel situated in the center of a cell is considered as an interest point for which the HOG features are extracted. However, a visual word computed only over one cell (18 bins) has a low discriminative power. In order to create larger visual words that capture bigger regions of an image (blocks), the features extracted from adjacent cells are concatenated within a more global descriptor (constructed using an overlapping sliding window of *n* x *n* cells).

Features from all images are clustered off-line to form a single BoVW codebook $W = \{w_1, w_2, \dots w_k\}$ by using the *k*-means clustering algorithm.

Step 3: *BOVW Image representation:* Now, the images from the training dataset are divided into blocks out of which an adaptive HOG descriptor is created. Each block is mapped to the nearest visual word according to the following equation:

$$w(d_i) = \arg\min_{w \in W} Dist(w, d_i) \qquad (9)$$

where $w(d_i)$ denotes the visual word assigned to the i^{th} descriptor d_i and $Dist(w, d_i)$ is the distance between the visual word w and the descriptor d_i. Now every image is represented as a collection of blocks that are each mapped to a visual word.

The final representation of the image is given as a histogram of visual words. The number of bins of the histogram is equal to the dictionary dimension (*k*).

Step 4: *SVM training:* Once the feature vectors are quantized to clusters, we propose next a multi-class supervised learning phase. During the training, the labeled data is sent to the classifier and used to adapt a statistical decision procedure in order to distinguish from categories. We employ the SVM classifier [28] that determines a separation hyperplane between two data with maximal margin:

$$f(x) = sign\left(\sum_i y_i \alpha_i K(x, x_i) + b\right) \qquad (10)$$

where x_i are the training features from the data space, y_i is the label of x_i, α_i are some parameters typically set to zero, b is the hyperplane free term, while K is the SVM kernel. In order to apply the SVM to a multi-class problem we selected the one-against all approach.

Step 5: *Object labeling:* For each object detected as presented in Section 3.1 we construct a frequency histogram by quantizing the adapted HOG features with the same codebook of visual words obtained at *Step 3*. Let us note that in our case, we do not perform an exhaustive sliding window search over the current frame in order to identify the possible object categories. We leverage on the results of the detection module and use the detected objects as queries. Thus, the classifier receives information about the localization and size of the object to be classified. Images are ranked according to the histogram distance computed using the *L2* distance.

4. Experimental results

The proposed vision based aid system for visually impaired persons consists only of a smartphone that is used to record and process the images. The system is attached to the user with the help of a chest mounted harness (Figure 3). Although blind users have learned not to sway much their bodies when walking, if the recording camera is attached with strings to the person the smartphone will not be very stable over the time *leading to* cyclic pan and tilt oscillation.

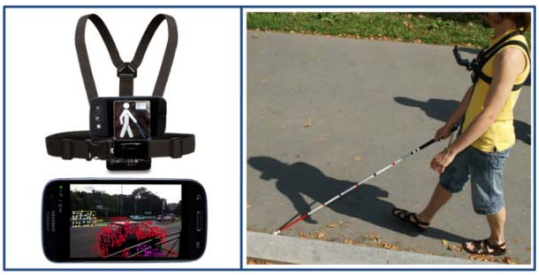

Figure 3: The proposed vision based system for partially sighted users.

4.1. Obstacle detection system evaluation

We conducted multiple experiments in real life urban environments such as Latin Quarter and Saint Michel Boulevard in Paris, France. The image sequences were also recorded and used for build a database of 20 videos at a resolution of 320 x 240 pixels and with an average duration of 5 minutes. The videos may include multiple independent moving objects such as pedestrians, bicycles or cars. The videos are noisy and trembling, include dark, cluttered and highly dynamic scenes which make them very challenging for an automatic obstacle extraction system. In addition, various types of both camera and multiple object motions are present.

Existent methods developed for obstacle detection are limited to small or regular motions of the camera or of the detected object. In order to emphasize the strength of our method we show in Figure 4 five challenging sequences for which these constraints do not necessary hold. Different colors are used to represent the various moving objects existent in the scene. Due to our temporal consistency step, the object color remains constant during the action development.

For the videos *Evry* and *Saint Michel* we present four frames that are very close in time. The system is able to detect static obstacles (*e.g.,* fences, pillars, trees and phone booths) situated either down on the foot area or high on the head level and label them at a considerable distance from the user. We can notice important camera motion which is due mostly to the subject displacement. The other three video sequences *Porte Maillot*, *Haussman* and *Latin Quarter* contain static obstacles and also dynamic objects (*e.g.,* vehicles, pedestrians, bicycles) correctly detected and classified even when situated at larger distances (10 meters). If we consider the case of the video *Latin Quarter* our system detects early enough a pedestrian that is approaching to the visually impaired person. In this case the user can change his/her direction of walking in order to avoid the obstacle.

Figure 4: Experimental results obtained using the proposed obstacle detection system.

Regarding the computational complexity, the processing time required by a desktop PC[1] for obstacle detection is 18 ms/frame. When running it on a Smartphone Samsung Galaxy S3 the average processing time is 102 ms/frame.

4.2. Obstacle classification system evaluation

In this section we present a comprehensive evaluation of the adapted HOG descriptor with BoVW. First we constructed a training dataset that contains 4500 images selected from the PASCAL database [29] that were divided into four classes depending on their relevance to a blind person: vehicles (1700 pictures), bicycles (500 pictures), pedestrians (1100 pictures) and static obstacles (fences, pylons, trees, garbage cans… 1200 pictures).

The considered categories were according to the responses given by 40 partially impaired persons on a questionnaire [30] regarding their major problems and preferences concerning their navigation in indoor and outdoor spaces. From these we determined that a blind user is the most frightened by the collision with a vehicle or a bike. In the context of the proposed application it is not important to differentiate between cars or buses, but to determine that the detected object is in fact a vehicle that is approaching the user. Regarding the class containing the static obstacles this is characterized by a high variability of the instances.

The tests are conducted on a set of 924 image patches, extracted from the video database using the algorithm presented in Section 3.1. In Table 1 we illustrate the performance of our system for each category along with the confusion matrix for a vocabulary of 4000 words.

As it can be noticed from Table 1 we introduced an extra category called *Outliers*. We added it in order to make sure that our system does not classify a patch in a category just because it is required but due to its high resemblance to a class.

Next, in Figure 5 we studied the impact that the size of the vocabulary has over the performance of object classification system. We used as evaluation metrics the traditional Recall (R), Precision (P) and F1 norm ($F1$) measures, defined as follows:

$$R = \frac{D}{D + MD};\ P = \frac{D}{D + FA};\ F1 = \frac{2 \cdot P \cdot R}{P + R} \quad (11)$$

where D is the number of correctly classified objects, MD is the number of missed classified, and FA is the number of false alarms (patches that are erroneous included in a category).

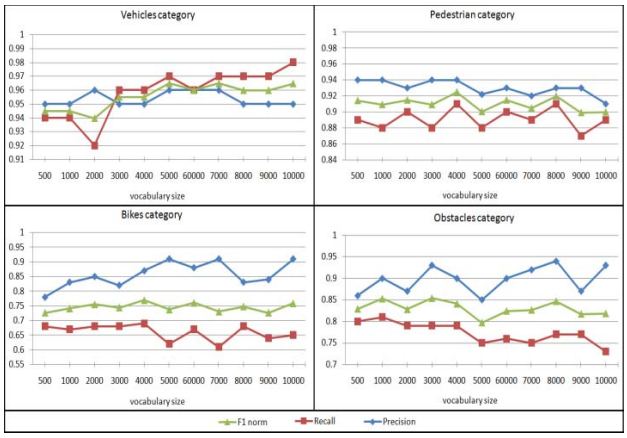

Figure 5: System's performance evaluation when varying the vocabulary size.

Even if the best results are obtained when using a vocabulary with 4000 words we have to consider that our system is designed as a real-time application for which the classification speed is very important because it controls how quickly the user will receive a response.

In Figure 6 we present the variation of the computational time per image patch (when testing it on a desktop PC[1]) with the increase of the vocabulary size.

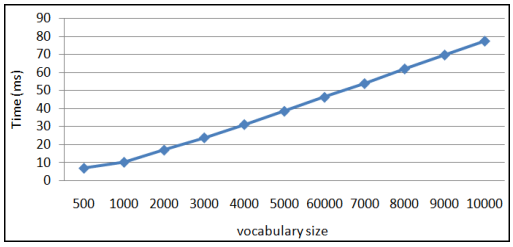

Figure 6: Computational time variation when changing the vocabulary size.

Even if the best results regarding the classification efficiency were obtained for a vocabulary of 4000 words due to the real-time constraints of our application we selected a vocabulary of size 1000. In this case, the average processing time for the proposed technique when running it on a Samsung Galaxy S3 smartphone is around 30ms/frame. The total time of the entire framework (object detection and classification) is 132 ms/frame which leads to a processing speed around 7 frames/second.

Table 1. Confusion matrix for the best vocabulary (4000 words).

	Cars	Bikes	People	Obstacles	Outliers	Ground Truth	False Alarm	Missed Detected	Precision	Recall	F1 norm
Cars	294	3	0	4	4	305	15	11	0,95	0,96	0,95
Bikes	3	79	15	3	15	115	12	36	0,87	0,69	0,77
People	4	8	342	4	19	377	23	35	0,94	0,91	0,92
Obstacles	8	1	8	100	10	127	11	27	0,90	0,79	0,84

5. Conclusions and perspectives

In this paper we have introduced a novel framework for obstacle detection and classification in order to assist partially sighted persons in navigating autonomously. The proposed obstacle detection module can function as an independent application, since it can detect dangerous moving objects very fast in complex environments without any *a priori* information about the size, shape or position of the obstacles to be detected.

The method works in real-time and is implemented on a regular smartphone as a mobility tool attached to the user with the help of a chest mounted harness. This technique is able to detect both static/dynamic obstacles and to classify them based on the relevance and degree of danger to a blind person. We tested our framework in different environments and prove its consistency and robustness.

For further work we plan to extend the system and improve it with a advanced alerting functionalities. In this respect, we want to make use of bond induction headphones which allow the users to hear the both the sounds from the device and the environment sounds. An alerting system based on the vibration functions of such headphones can be designed for this application. On the other hand, a wider study with more blind users and further questioners about the system will offer us some further guidelines.

6. Acknowledgement

This work has been carried out within the framework of the European ALICE project (AAL-2011-4-099), supported by the AAL (*Ambient Assisted Living*) program.

References

[1] C. Shah, M. Bouzit, M. Youssef, and L. Vasquez, Evaluation of RU-Netra Tactile Feedback Navigation System for the Visually Impaired, International Workshop on Virtual Rehabilitation, 72-77, 2006.

[2] L. Chen, B.L. Guo and W. Sun, Obstacle Detection System for Visually Impaired People Based on Stereo Vision, Fourth International Conference Genetic and Evolutionary Computing, 723-726, 2010.

[3] A. Rodriguez, Assisting the Visually Impaired: Obstacle Detection and Warning System by Acoustic Feedback, 17476 – 17496, Sensors 2012.

[4] J. A. Hesch, S.I. Roumeliotis, Design and analysis of a portable indoor localization aid for the visually impaired, Journal of Robotics Research, 1400–1415, 2010.

[5] J. M. Sáez, F. Escolano, and A. Peñalver, First steps towards stereo based 6DOF SLAM for the visually impaired, in IEEE CVPR, 2005.

[6] V. Pradeep, G. Medioni, J. Weiland, Robot Vision for the Visually Impaired, 15-22, CVPR, 2010

[7] J. M. Saez, F. Escolano, Stereo-Based Aerial Obstacle Detection for the Visually Impaired, In W. on Computer Vision Applications for the Visually Impaired, 2008.

[8] P. F. Alcantarilla, On Combining Visual SLAM and Dense Scene Flow to Increase the Robustness of Localization and Mapping in Dynamic Environments, ICRA, 1290–1297, 2010.

[9] S. Vedula, S. Baker, P. Rander, R. Collins, T. Kanade, Three-Dimensional scene flow, Trans. Pattern Analysis Mach. Intellect, (27) 475–480, 2005.

[10] J. W. Durham, F. Bullo, Smooth Nearness-Diagram Navigation, International Conference on Intelligent Robots and Systems, 690–695, 2008.

[11] S. Kumar, Binocular Stereo Vision Based Obstacle Avoidance Algorithm for Autonomous Mobile, Int. Conf. on Advance Computing, 254–259, 2009.

[12] N. Snavely, S. M. Seitz, R. Szeliski, Modeling the world from Internet photo collections, International Journal of Computer Vision, 80(2): 189–210, 2008.

[13] A. J. Davison, I.D. Reid, N. D. Molton, O. Stasse, MonoSLAM: real-time single camera SLAM, IEEE PAMI, 29(6): 1052–1067, 2007.

[14] E. Tola, S. Knorr, E. Imre, A. Alatan, T. Sikora, Structure from motion in dynamic scenes with multiple motions, Workshop on Immersive Communication, 2005.

[15] K. Souhila, A. Karim, Optical flow based robot obstacle avoidance, International Journal of Advanced Robotic Systems, 4(1): 13–16, 2007.

[16] E. Peng, P. Peursum, L. Li, S. Venkatesh, A smartphone-based obstacle sensor for the visually impaired, Lect. Note Comput. Sci., 590–604, 2010.

[17] R. Manduchi, Mobile vision as assistive technology for the blind: An experimental study, Note Comp. Science, (7383) 9–16, 2012.

[18] D. Lowe, Distinctive image features from scale-invariant keypoints, IJCV, 1-28, 2004.

[19] H. Bay, T. Tuytelaars, L. Gool, Surf: Speeded up robust features, In Proc. ECCV, 2006.

[20] B. Lucas, T. Kanade, An iterative technique of image registration and its application to stereo, ICAI, 1981.

[21] M. Black, P. Anandan, A frame work for robust estimation of optical flow, CVPR 1993.

[22] J. Lee, G. Kim, Robust estimation of camera homography using fuzzy RANSAC, ICCSIA, 2007.

[23] P. Cimiano, A. Hotho, S. Staab, Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text, ECAI, 435–439, 2004.

[24] M. Zhang, Z. Zhou, A k-nearest neighbor based algorithm for multilabel classification, International Conference on Granular Computing, 2005.

[25] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 2005.

[26] N. Dalal, B. Triggs, Object detection using histograms of oriented gradients, ECCV, 2006.

[27] C. Dance, J. Willamowski, L. Fan, C. Bray, Visual categorization with bags of keypoints, in: ECCV, 2004.

[28] Tong, S., Chang, E., "Support Vector Machine Active Learning for Image Retrieval," Proc. ACM Multimedia Conf., pp. 107-118, 2001.

[29] http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases

[30] http://www.alice-project.eu/

[i] The algorithm was run on an Intel Xeon Machine 3.6 GHz, 16 GB with a NVIDIA Quadro 4000 video board, under Windows 7 platform.