OBSTACLE DETECTION FOR PEDESTRIANS WITH A VISUAL IMPAIRMENT BASED ON 3D IMAGING

Michiel Vlaminck, Ljubomir Jovanov, Peter Van Hese, Bart Goossens, Wilfried Philips and Aleksandra Pižurica

Ghent University - TELIN - IPI - iMinds
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
ljj@telin.ugent.be

ABSTRACT

According to the World Health Organisation, 285 million people live with a visual impairment. Despite the fact that many efforts have been made recently, there is still no computer-guided system that is reliable, robust and practical enough to help these people to increase their mobility. Motivated by this shortcoming, we propose a novel obstacle detection system to assist the visually impaired. This work mainly focuses on indoor environments and performs classification of typical obstacles that emerge in these situations, using a 3D sensor. A total of four classes of obstacles are considered: walls, doors, stairs and a residual class (which covers loose obstacles and bumpy parts on the floor). The proposed system is very reliable in terms of the detection accuracy. In a realistic experiment, stairs are detected with 100% true positive rate and 8.6% false positive rate, while doors are detected with 86.4% true positive rate and 0% false positive rate.

Index Terms— Obstacle detection, 3D point clouds.

1. INTRODUCTION

Most of the existing systems for autonomous navigation of visually impaired people are based on the Global Position System (GPS). However, due to the inaccurate nature of GPS and its susceptibility to signal loss, these systems are not the most convenient way to assist the visually impaired. Moreover, such systems are not able to provide local information on the obstacles that are encountered. Another important drawback of such systems is their inability to operate indoors.

Motivated by these shortcomings, we propose a novel obstacle detection system based on the use of a 3D sensor. This task is far from trivial since the system has to satisfy a couple of strict conditions. The most important among these conditions is the time critical nature of our application. In addition, the proposed system has to be *robust* and maintain high performance level in varying environment conditions. Finally, the system should also be as *reliable* as possible in a sense that as little obstacles as possible should remain undetected.

Although our proposed system operates outdoors, we mainly focus on indoor environments. Obstacles emerging in such situations are nonetheless quite diverse. This fact encouraged us to perform a classification of the obstacles the pedestrian could face in typical indoor environments. More specifically, we will consider four classes of obstacles, including walls, doors, stairs and a residual class which covers loose obstacles and bumpy parts on the floor. The choice for this categorization is derived from the actions that the pedestrian will need to take in each situation. For example, in case that the pedestrian encounters a staircase, he does not need to avoid them but rather to go upstairs. In a different situation, when the obstacle is classified as a door, the right action to perform is to open it. Loose obstacles and bumpy parts of the floor, in turn, require no action other than simple avoidance.

2. RELATED WORK

Obstacle detection is one of the most active areas in computer vision. However, the majority of the proposed obstacle detection systems are developed for autonomous mobile robots and although there are some similarities, there are two major differences.

First difference is related to the fact that humans are generally more intelligent than mobile robots and as a result they can benefit from a classification as explained in the introduction. A second major difference is related to the orientation of the camera. In systems developed for mobile robots, the orientation of the camera remains more or less stable, while in systems for visually impaired people, the camera is usually subject to changes in orientation while the person is walking.

This last observation was already made by Molton *et al.* [5]. The authors used a stereo based system for the detection of the ground plane and obstacles. To overcome the problem of frequent changes in orientation of the camera, the authors present an algorithm for dynamic recalibration of the ground plane.

The obstacle detection systems developed for visually impaired people differ greatly from each other. A first point of

978-1-4799-3203-0/13/$31.00

difference can be found in the type of image data used as an input. Jie *et al.* [3] use intensity images to calculate a grayscale histogram, which is then used to detect obstacles. Once an obstacle is detected, the distance to it is calculated using the principles of stereo vision.

Rodriguez *et al.* [7] make use of data obtained from a stereo pair. A polar grid that delimits the region in which obstacles can occur was defined and estimated empirically to 4.5 meters. The first step in their detection process consists of estimating the ground plane by means of plane fitting technique based on RANSAC. The authors assume that the ground plane is the most dominant plane in the scene, which in some cases can lead to a wrong detection or even a non detection of the ground plane. In the next step, the polar grid is projected onto the detected ground plane and the bins of the grid are filled with the 3D points located within the borders of each considered bin. The obtained 2D histogram is then analysed to detect the real obstacles.

Lee *et al.* [4] present an obstacle detection system that makes use of data produced by a Time of Flight (ToF) camera. The approach followed there differs fundamentally from the approach used in [7] in a sense that the algorithm does not start from the detection of the ground plane. Instead, a segmentation of the entire scene is made on a pixel level. This segmentation is guided by the discontinuities in the depth map and performed using a region growing method. Because this segmentation is dependent of the initial seeds, the authors present an algorithm to guarantee that the seeds are equally spread among the objects. In the next stage, the obstacles are extracted out of the segmented image.

The approach followed in [1] is again completely different from the one in [4] in a sense that it does not work directly on the depth map, but rather constructs a 3D representation of the environment in front of the person. For this representation, a transformation is defined to map the reference system of a Kinect to a reference system centered at the feet of the pedestrian. This latter approach will also be used in the proposed work.

3. THE PROPOSED OBSTACLE DETECTION METHOD

An overview of the proposed obstacle detection algorithm is given in Figure 1. As can be seen, the proposed algorithm is divided in three stages of which the first stage deals with the reconstruction of a point cloud using both depth and color data. The second stage subsequently filters the obtained point cloud and feed it to the segmentation stage. This latter consists of a plane based segmentation followed by the segmentation of the remaining objects.

3.1. Reconstruction of the 3D point cloud

During the reconstruction stage, depth and color are combined to create a 3D point cloud with color information. Extensive research on the Kinect calibration was done in [9], resulting in the calibration parameters which we use in the proposed approach. Once the point cloud is created, we transform it in order to align the ground plane with the xz-plane in our reference system (see Figure 2). The importance of the alignment to a reference system is twofold. Firstly, by the alignment, the detection of the ground plane is greatly simplified. Secondly, we do not have to assume that the ground plane is the dominant plane in the scene as in [4] and [7]. Moreover, the alignment will enable us to make use of the absolute position of the objects in the scene, in particular of the absolute height of the obstacles. Figure 2 illustrates the idea of the performed transformation.

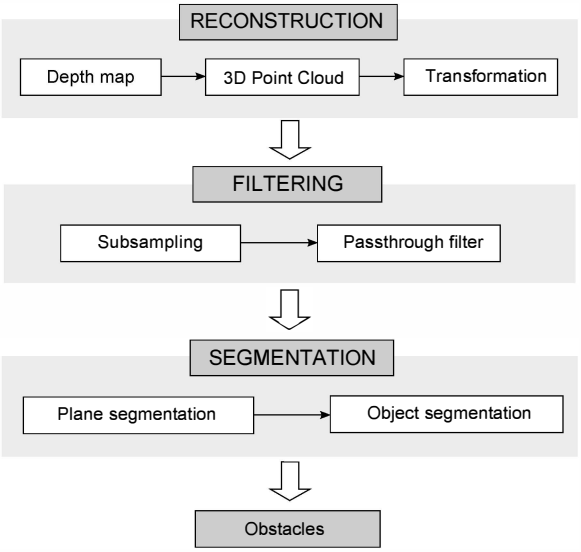

Fig. 1. Overview of the obstacle detection proces.

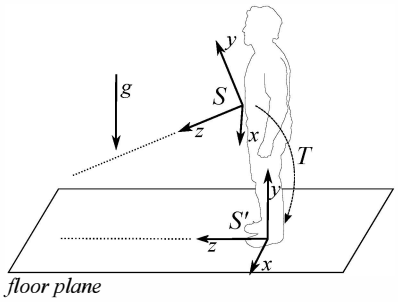

Fig. 2. Illustration of the performed transformation, used in order to align the point cloud to the reference system. The xz-plane is aligned with the floor plane and the origin is centered at the feet of the pedestrian.

3.2. Filtering to reduce the computational complexity

Since the computational complexity of the algorithms performed on point clouds is a function of the number of points

they contain, we can reduce the execution time by reducing the number of points. To achieve this we apply two different techniques. First we downsample the point cloud, from 640×480 to 320×240 points, by calculating a mean value of a sliding 2×2 block. Next, we remove all points located at a position further than 75 cm in the x-direction centered at the position of the person. Assuming that the person is walking along his viewing direction, this last filter will not jeopardize the correct detection of obstacles.

3.3. Segmentation for obstacle detection

The segmentation stage consists of both plane-based segmentation and object-based segmentation (see Figure 1). During the plane-based segmentation, all the dominant planes in the scene are extracted. The segmentation is performed by RANdom SAmple Consensus (RANSAC), an iterative method to estimate the parameters of a model, using data that contains outliers. In each iteration a plane is estimated using a set of N randomly selected hypothetical inliers from the point cloud.

To detect the ground plane, we check in each iteration if the obtained plane has a normal *perpendicular* to the xz-plane. Subsequently, all the points located within a fixed distance to the plane are added to the set of inliers. Finally, if the number of inliers belonging to the plane is larger than the maximum number of inliers from the previous iteration, we will select this plane as our current ground plane. This procedure is repeated for a fixed number of iterations.

Similarly, we seek for the walls in the scene. To detect walls, in each iteration we check if the normal of the obtained plane is *parallel* to the xz-plane.

To accelerate the plane-based segmentation, we first calculate local surface normals of each point in the point cloud. The calculation of the surface normals is based on the average change in depth value. Before applying the RANSAC procedure, we perform clustering of the points in accordance to their surface normal. In order to perform the detection of the ground plane, we associate all the points with normals deviating less than a threshold from the direction of the y-axis in the same cluster. To detect of walls, we form a cluster consisting of points whose normal is orthogonal to the y-axis.

Once the dominant planes in the scene are segmented, we move on to the segmentation of the remaining objects. The easiest way to do this is to cluster the remaining points based on their distance to each other. Each point that has a distance smaller than a certain threshold (which we set to 5 cm) to its neighboring pixel will be added to same cluster of this neighboring pixel.

4. DETECTION OF STEPS

The detection of steps in the scene is solely based on depth information. As a result, the algorithm can operate in situations subject to different light conditions. In [2], the authors attempt to detect the horizontal edges of stairs using the discontinuities in the depth map, an approach which works well for straight staircases. However, due to the frequent occurrence of spiral staircases, we will not adopt this approach. Instead, we will use a piecewise planar model, just like the authors in [6] and [10]. The algorithm can be summarized as follows:

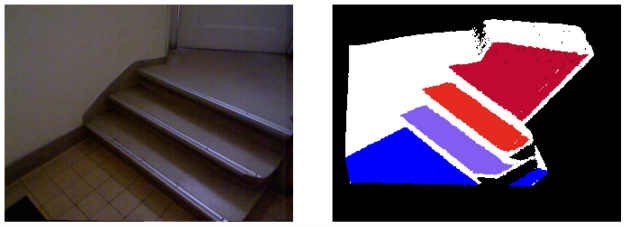

Fig. 3. Example of detected steps.

1. Estimate the local surface normals $\mathbf{n}_i$ of the points $\mathbf{p}_i$ belonging to a point cloud P.
2. Filter $\mathbf{p}_i$ based on $\mathbf{n}_i \perp xz$-plane.
3. Initialise $N = 1$.
4. Determine $P_N = \{\mathbf{p}_n\}$, the set of points located at height $h = H_N \pm H_{\text{tol}}$. Herein, $H_N = H_{\text{step}} \times N$ and H_{step} and H_{tol} are respectively the height of a step and the tolerance measure for this height.
5. Estimate a plane V_N using the set of points P_N in a RANSAC based manner.
6. Determine the number of inliers of V_N. If $|V_N| < \frac{T_{\text{step}}}{d}$ no step is found and the procedure quits. Herein, $\frac{T_{\text{step}}}{d}$ is a threshold that defines the minimum number of inliers in order to recognize the plane as a step and d is the average depth value of a step.
7. In the other case, increment N by 1 en restart the procedure from step 4.

The values of H_{step} and H_{tol} are the parameters that we can adapt. However, to make the detection as robust as possible, it is recommended to select $H_{\text{tol}} = \frac{H_{\text{step}}}{2}$ to make sure not to miss a step.

5. DOOR DETECTION

Due to the great diversity of doors and door handles, their detection in a 3D point cloud is far from trivial. For this reason our proposed door detection algorithm will focus on doors that satisfy certain conditions. According to the International Building Code (IBC), doors should have a minimum width of 32 inch (813 mm) and the height of the door handle should

be between 34 inch (864 mm) and 48 inch (1219 mm). However, after some experimentation in the "Technicum" building of Ghent University, we observed that plenty of doors do not satisfy these conditions. Typical example of such doors are dual doors, because none of these have of 813 mm per side. Because of this observation, we decided to relax the condition on a door width to a minimum width of 670 mm and a maximal width of 1200 mm.

In contrast to the proposed algorithm for detecting steps, the detection of doors will also use color information. The motivation for this is threefold. First, we need color to segment the different planes of interest. In addition, we also need color in order to be able to segment the two door surfaces of dual doors. The last reason is related to the nature of the door handle, i.e., its thin structure. Since we downsample a point cloud with a factor two, a very small number of points will belong to the door handle. In this respect, color can be useful to determine if the points are the result of noise at that position.

The algorithm can be summarized as follows:

1. Estimate the local surface normals $\mathbf{n}_i$ of the points $\mathbf{p}_i$ belonging to a point cloud P.
2. Segment the set of planes $V = \{V_1, V_2, ..., V_N\}$ of potential doors using both the surface normals and the color of the points.
3. Estimate the parameters of each plane as in Equation 1 using RANSAC.

$$a_k x + b_k y + c_k z + d = 0, \tag{1}$$

   where $\mathbf{n}_k = (a_k, b_k, c_k)$ is the normal to the plane V_k.
4. Determine the angle between $\mathbf{n}_k$ and the normal of the ground plane. This angle should approximate 90 degrees, since doors are perpendicular to the floor.
5. Determine the dimensions of each plane.
6. For each plane $V_k \in V$ check if its width satisfies the conditions. Remove V_k if this is not the case.
7. Detect the potential door handle.

In order to segmentation the different planes, we make use of a method based on connected components. Using this method, neighboring pixels are compared to each other in a 4-connected neighborhood. To account for both color, depth and curvature information, we make use of three comparison functions. Only in the case where the results of all three functions are positive, the considered point is added to the segment of its neighbor.

The first comparison function checks if the angle between 3D points belonging to two neighboring pixels is smaller than a threshold θ, whereas the second comparison function checks if the distance of these 3D points is smaller than a threshold d. Finally, the third comparison function checks if the colors of the two neighboring pixels are sufficiently similar. This comparison of colors is done in the RGB color space.

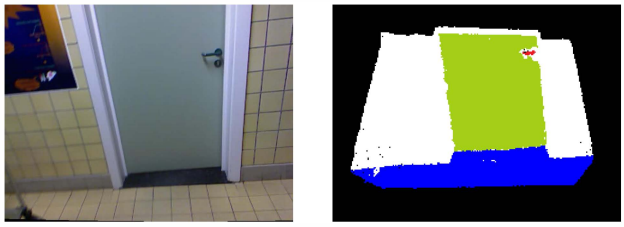

Fig. 4. Example of a detected door.

We determine the plane width by using a convex hull of the set of points V_k. For this, we use the Quick Hull algorithm implemented in the QHull library [11] which has a time complexity of $O(n \log v)$, n being the number of input points and v the number of output vertices.

The procedure to detect the door handle can further be summarized as follows:

1. Filter the remaining points $\mathbf{p}_j$, i.e., points that do not belong to any plane, based on their height. This height should be in the interval of 864 mm and 1219 mm.
2. Check for every door candidate $V_k \in V$ if point $\mathbf{p}_j$ is located in the predetermined handle zone.
3. In case the point $\mathbf{p}_j$ is located in the handle zone, add it to the set H of candidates for the door handle.
4. Select H as door handle and V_k as door plane when H contains sufficient points.

The predetermined handle zone consists of two virtual cuboids located at the left and right side of the door. The dimensions of these cuboids are set to $18 \times 36 \times 10$ cm. The position of these cuboids is defined by the minimum and the maximum height of the door handle and their distance to the door surface is set to 3 cm. Thus the distance of a point $\mathbf{p}_j$ to a door surface V_k has to satisfy Equation 2, where d_{thresh} is set to 3 cm:

$$d(\mathbf{p}_j, V_k) = |a_k x + b_k y + c_k z + d_k| < d_{\text{thresh}} \tag{2}$$

When a plane and accompanying handle found satisfy the conditions previously described, we classify the detected object as a door.

6. IMPLEMENTATION DETAILS

The system was developed in C++ using Visual Studio 2010 and some external libraries including PCL, OpenCV, Eigen, Boost, FLANN, QHull and VTK. It was tested on a computer system with an Intel Core 2 Duo 2.53 GHz processor

and 4 GB RAM. We have evaluated three types of depth sensors namely, SoftKinetic DepthSense 325, PMD CamCube 2.0 and Microsoft Kinect. With regard to DepthSense 325, we have concluded that its SNR on distances up to 1 meter makes it inconvenient to use for the purpose of obstacle detection. CamCube 2.0 camera performs much better in this respect, but in contrast to the Kinect, it is not able to capture color information. These reasons made us decide to mainly work on data captured using Kinect, despite the fact that it needs an external power source and that it cannot operate outdoors. The implementation relies on the use of the *Point Cloud Library* (PCL) [8], an open source, platform-independent library, which provides some useful data types and algorithms for 3D point cloud processing. We also needed a framework to provide us the color images and depth maps produced by the Kinect. With a view to interoperability, we decided to work with the Open Natural Interaction (OpenNI) library, for which PCL provides an interface.

7. EVALUATION

In this section we evaluate the performance of the proposed algorithm using multiple sets of recorded 3D sequences corresponding to different types of obstacles. We consider two different user scenarios, depending on the position of a Kinect sensor during recording. In the first scenario Kinect sensor was attached to the helmet of a user, while in the second it was attached to the user's jacket at the height of approximately 120cm. In both cases, walking speed of approximately 1 m/s was maintained during the recording.

7.1. Initial segmentation

An important issue in this application is real-time operation. We have determined that the segmentation of the ground plane takes about 15 to 20 ms whereas the segmentation of the remaining planes takes 5 to 10 ms. The clustering based on a distance, takes only 3 ms to execute, since we are exploiting the adjacency in the pixel domain.

Regarding the robustness, the most critical part of our system is the detection of the ground plane. Although the proposed detection algorithm is very robust, there are two situations in which it could fail to produce sensible results. In the first of such situations, one or more steps are causing a change of the ground plane, which sometimes causes wrong detection of the future ground plane as a current ground plane, even when the pedestrian has not taken the steps yet. Second such situation occurs when an obstacle consists of a large horizontal plane. In this situation, the horizontal plane of the obstacle could be wrongly identified as the ground plane.

7.2. Step detection

In order to evaluate the robustness and the speed of the proposed step detection algorithm, we have used the dataset collected by Tang *et al.* and presented in [10]. This dataset consists of 90 positive images of staircases together with their corresponding depth maps and accelerometer data. In addition, a negative collection of 58 entries is provided. The results are summarized in Table 1. In this Table, FN and FP denote false negatives and false positives respectively. The values of t_{NE} and t_{SD} denote the processing times for calculation of the surface normals and the step detection respectively.

resolution	FN (/90)	FP (/58)	t_{NE} (ms)	t_{SD} (ms)
QVGA	0	5	18	16
VGA	0	2	74	66
Tang [10]	5	1	N/A	13

Table 1. Performance of the step detection algorithm on the dataset presented in [10]. FN = false negatives, FP = false positives, t_{NE} = time for calculation of the surface normals, t_{SD} = time for detection of one step.

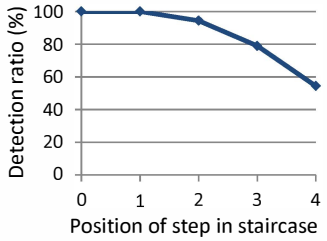

Fig. 5. Plot of the detection ratio (i.e., the true positive rate) as a function of the position of the staircase step. The farther away from the sensor, the lower the chance the step is detected.

As we can see in Figure 3, no stairs remain undetected. For a QVGA resolution a total of 5 stairs are wrongly detected, whereas for a VGA resolution only 2 false positives occur. The algorithm is thus quite effective. With regard to the processing speed, 18 ms are needed to estimate the normals in the point cloud. Normal estimation is based on the average depth change in the pixel domain. An additional 16 ms are needed to detect one step. Therefore the proposed algorithm is fast enough to be used in practice.

The results shown in Table 1 are obtained using binary evaluation, i.e., the detection is considered successful when one step was detected. To evaluate how the detection of subsequent steps deteriorate due to the increasing distance, an additional test was performed on the positive dataset. The results are shown in Figure 5. As we can see in the graph, the second step is detected in 95% of the cases, whereas the third and fourth step are detected in respectively 80% and 55% of the cases.

resolution	FN (/44)	FP (/58)	t_{NE} (ms)	t_{D} (ms)
QVGA	6	0	19	48

Table 2. Performance of the door detection algorithm on our own collected dataset. FN = false negatives, FP = false positives, t_{NE} = time for calculation of the surface normals, t_{D} = time for door detection.

7.3. Door detection

In order to evaluate the robustness and speed of our proposed door detection algorithm, we have collected our own small dataset in the "Technicum" building of Ghent University. This dataset contains 44 images of stairs together with their corresponding depths map and accelerometer data. The results are summarized in Table 2. For the evaluation of false positives, the negative dataset of [10] was used.

As can be seen in Table 2, for a QVGA resolution 6 out of 44 doors remain undetected. The reasons for these errors are diverse. First of all, since our algorithm relies on color information, the lack of sufficient ambient light could cause the detection failure. In most cases however, the concave nature of the door, i.e., exhibiting an indentation with respect to the wall, will compensate these shortcomings.

Even when there is enough ambient light, problems can occur at the detection of dual doors in situations where the two door surfaces are not separable in color space.

Another possible reason for wrong detection can be an acute angle in situations where the concave nature of a door causes the door handle or a considerable part of the door surface to be invisible.

The last possible cause of failure, which has not encountered in our dataset, can occur in the rare situation when a door is rich in texture. In these situations a non detection could be caused by oversegmentation, when none of the segments satisfy the conditions.

Although there are many possible reasons which might cause missed detections of the door surfaces, this is not the main cause of failure. Due to the amount of noise and the thin structure, door handles are quite difficult to detect, which in the most cases prevents the door detection.

The results of our measurements show that average door detection time is 48ms, which makes the proposed algorithm suitable for the practical use.

7.4. Entire system

Local surface normals are used for the initial segmentation and step and door detection, so we can calculate them only once. As stated before, this estimation only takes 18 ms. Initial segmentation is performed in 30 ms on average. In the situation where steps are present, the step detection process barely needs 16 ms per detected step. Finally, an average time of 48 ms is needed for the door detection. These four parts finally lead to an average time of 100 ms per frame or a processing speed of 10 Hz. Although the proposed system performs obstacle detection frame per frame, obstacle detection results show good consistency over time. Detection performance could be additionally improved by observing temporal buffer containing segmentation results from previous frames.

Achieved speed of 10-15 Hz can be considered as sufficient for practical indoor applications if we assume walking speeds up to 1.3 m/s. By using more recent multicore CPU and GPU acceleration, processing time can be significantly reduced. The application of the proposed algorithm in outdoor conditions will become possible once depth sensors become able to produce images of sufficient quality in the presence of strong ambient illumination.

8. CONCLUSION

In this work, an obstacle detection system for visually impaired people was presented. The system operates at an average speed of 10 Hz, which is considered fast enough to be used in practice. In terms of reliability we conclude that steps will be detected in almost all cases as we obtained 0 false negatives on the dataset of [10]. Moreover, only 5 false positives occur on a set of 58 images. Due to the noise and the thin structure of door handles, the detection of doors is less reliable than the detection of steps. However, only 6 out of 44 doors remained undetected in our own dataset, with no false positives.

9. REFERENCES

[1] D. Bernabei, F. Ganovelli, M. Di Benedetto, M. Dellepiane, and R. Scopigno. A low-cost time-critical obstacle avoidance system for the visually impaired. In *International conference on indoor positioning and indoor navigation*, Guimarães, Portugal, 2011.

[2] J. A. Delmerico, D. Baran, P. David, J. Ryde, and J. J. Corso. Ascending stairway modeling from dense depth imagery for traversability analysis. In *Proceedings of IEEE Internation Conference on Robotics and Automation*, 2013.

[3] Y. Jie and S. Yanbint. Obstacle detection of a novel travel aid for visual impaired people. In *Intelligent Human-Machine Systems and Cybernetics*, pages 362–364, 2012.

[4] C. H. Lee, Y. C. Su, and L. G. Chen. An intelligent depth-based obstacle detection system for visually-impaired aid applications. In *WIAMIS*, pages 1–4, 2012.

[5] N. Molton, S. Se, J. M. Brady, D. Lee, and P. Probert. A stereo vision-based aid for the visually impaired. In *Image and Vision Computing*, pages 251–263, 1998.

[6] V. Pradeep, G. Medioni, and J. Weil. Piecewise planar modeling for step detection using stereo vision. In *Workshop on Computer Vision Applications for the Visually Impaired*, Marseille, France, 2008. James Coughlan and Roberto Manduchi.

[7] A. Rodríguez, J. J. Yebes, P. F. Alcantarilla, L. M. Bergasa, J. Almazán, and A. Cela. Assisting the visually impaired: Obstacle detection and warning system by acoustic feedback. *Sensors*, 12(12):17476–17496, 2012.

[8] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011.

[9] Nicolas Burrus Kinect calibration parameters http://nicolas.burrus.name

[10] T.J.J. Tang, W.L.D. Lui, and W.H. Li. Plane-based detection of staircases using inverse depth. In *Australasian Conference on Robotics and Automation*, Wellington, New Zealand, 2012.

[11] Quick Hull `http://www.qhull.org`