Stairs Detection with Odometry-aided Traversal From a Wearable RGB-D Camera

A. Perez-Yus*, D. Gutierrez-Gomez, G. Lopez-Nicolas, J. J. Guerrero

Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Spain

Abstract

Stairs are one of the most common structures present in human-made scenarios, but also one of the most dangerous for those with vision problems. In this work we propose a complete method to detect, locate and parametrise stairs with a wearable RGB-D camera. Our algorithm uses the depth data to determine if the horizontal planes in the scene are valid steps of a staircase judging their dimensions and relative positions. As a result we obtain a scaled model of the staircase with the spatial location and orientation with respect to the subject. The visual odometry is also estimated to continuously recover the current position and orientation of the user while moving. This enhances the system giving the ability to come back to previously detected features and providing location awareness of the user during the climb. Simultaneously, the detection of the staircase during the traversal is used to correct the drift of the visual odometry. A comparison of results of the stair detection with other state-of-the-art algorithms was performed using public dataset. Additional experiments have also been carried out, recording our own natural scenes with a chest-mounted RGB-D camera in indoor scenarios. The algorithm is robust enough to work in real-time and even under partial occlusions of the stair.

Keywords: Stair detection, visually impaired, RGB-D, visual odometry, wearable computing

1. Introduction

The inability to see properly affects many aspects of the life of the people who suffer from vision loss such as the ability to read text or interpret signs, the recognition of objects and people, or problems related to mobility. Giving assistance to the visually impaired has been a relevant topic for many years in the computer vision community, and many advances have been accomplished in some specific tasks. However, it still remains hard to integrate them naturally.

In this work, we deal with the mobility problems derived from visual impairment. Safely moving from one point to another poses some difficulties, *e.g.* presence of (moving) obstacles, traffic, orientation issues or the detection of curbs or doors. Some of these problems can be overcome by previous knowledge of the environment or with the help of mobility aids such as the white cane or guide dogs. Nevertheless, even when moving in familiar environments or using a mobility aid, visually impaired people are still prone to be involved in accidents. According to the survey performed in [1], 7% of the respondents experienced falls while walking at least once a month. It also mentions that the frequency of accidents has nothing to do with the type of mobility aid or the number of times going out along unfamiliar routes.

Although the reliability, feedback, simplicity and price of the white cane seems unbeatable, we believe with modern sensors and techniques the navigating experience can be enhanced comfortably. Some researchers studied how to make the cane smarter [2, 3]. In contrast, in our proposal we intend to attach the sensors to the user, using a wearable camera. The idea is to complement rather than replace, increasing and improving the amount of information they can already receive by other means.

One of the biggest challenges of using cameras as the main sensor in navigation is the geometric reconstruction of the environment, which is often solved using SLAM algorithms or camera systems that are able to retrieve depth information from the scene, e.g. stereo vision or Time-Of-Flight (TOF) cameras. These systems are either complex to use and calibrate (stereo) or heavy and expensive (TOF). However, since 2010, low cost domestic cameras providing both colour and depth information have been launched to the market, causing a great impact on the field. This type of sensors, called RGB-D, provides a fast and reliable geometric reconstruction in one single shot, which makes them interesting for navigation tasks. They are based on structured IR light, technology that currently prevents them from working properly in daylight. As a consequence we limited our experimental work to indoor scenarios.

Regarding the location of the camera, our experimental setup currently places the camera in a chest harness, pointing approximately 45° downwards (Fig. 1). We prefer this configuration to the head-mounted one as in this way the camera is more stable and is always pointing at the region in front of the user, causing safer travelling in terms of avoiding obstacles and, consequently, reducing falls or collisions. Also, as stated in [4], a chest-mounted configuration is likely the most social acceptable one as it interferes the least in social interaction. The sys-

*Corresponding author

Email addresses: alperez@unizar.es (A. Perez-Yus), danielgg@unizar.es (D. Gutierrez-Gomez), gonlopez@unizar.es (G. Lopez-Nicolas), josechu.guerrero@unizar.es (J. J. Guerrero)

Fig. 1. Photos of the current wearable camera system while detection (left) and traversal (right). The camera is placed on the chest, pointing downwards. The computations are performed in a laptop carried in a backpack.

tem is designed to work well with natural movements of a person, even though an inclination of the sensor of $\approx 45°$ is needed for initialization and suggested for security and functional reasons during the run. The processing has to be performed in a laptop at the moment, though new advances in both RGB-D cameras and processing units promise the possibility of developing more comfortable configurations in the near future.

In this work, we present a method which takes advantage of RGB-D cameras in the tasks of stair detection, modelling and traversal. Stairs are omnipresent structures in human made environments that allow humans to displace vertically between different floors, but also a source of potential accidents. RGB-D cameras can assist enormously, as stairs usually have a similar shape which can be perceived by these instruments. This stair detection method is thought to be part of a more general personal assistant based on computer vision which includes other applications such as obstacle detection and audio interface [5, 6]. In our current framework, we have included the estimation of the visual odometry from [7] during the navigation in order to maintain location awareness and to know the relative position to relevant features in the scene even when they are not in the current view.

A diagram summarizing how the system works is shown in Fig. 2. The sensor provides RGB and Depth images which are used to compute a 3D Point Cloud [8] and to feed the visual odometry estimation. With the 3D Point Clouds we perform a planar segmentation of the scene, compute the relative transformations of the user to the environment and obtain a classification of the planes according to their orientation. Unidentified vertical planes and clusters of points are classified as obstacles to be avoided in our system. The horizontal planes classified as step candidates are run through the stair detection and modelling algorithm. When the system detects that the user has reached the proximity of the staircase the system proceeds with the stair traversal algorithm, able to recover the position of the user along the stairway using the user pose estimated with the visual odometry module. Simultaneously, the live information from the camera is used to correct the visual odometry drift.

To summarize, our contributions are the following:

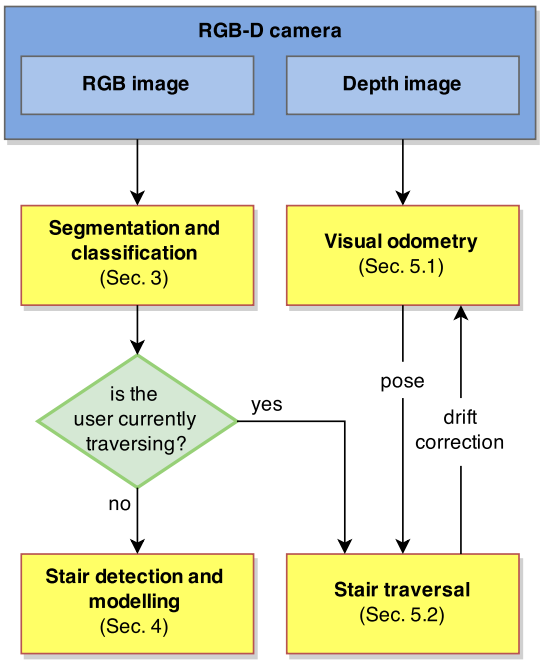


Fig. 2. Block diagram of one iteration of the main loop of the proposed method.

- Development a stair detection algorithm from point clouds integrated in a more general obstacle detection navigation framework. The algorithm models the stair dimensions and orientation with respect to the user which better fits the cloud for navigation and validation purposes.
- Development of a stair traversal algorithm which takes advantage of the visual odometry to complete the model of the full staircase, and to retrieve at any time the step in which the user stands.
- Reciprocally, the current view during the traversal is used to correct the drift of the visual odometry module.

This work extends what was presented in [9], where we addressed the stair detection and modelling part in a RGB-D navigation framework. In this paper we have added the visual odometry module and addressed the traversal, as well as enhanced the earlier approach with some inclusions. For instance, as we deal only with indoor scenes, we retrieve the Manhattan directions [10] to make a faster and simpler stair modelling, considering staircases most likely follow that convention in such type of environments. Experiments to test our system's detection ratio, quality of the model, temporal performance and drift correction during traversal have been made and included at the end of this work.

2. Related Work

The usage of cameras to detect obstacles for human navigation has been widely approached in the field of visually impaired aids. For instance, a combination of obstacle detection with a stereo-based SLAM for developing a visually impaired aid was performed in [11]. With the SLAM estimate of the

visual odometry and the 3D map, they perform a traversability analysis of the environment. A tactile interface situated on a vest steers the subjects away from obstacles along the path. A more recent version of the system presented in [12] using a RGB-D camera instead of a stereo system improved the results in robustness. Other approaches with different interfaces for blind navigation are [13, 14]. Aladren *et al.* [5] improved the way-finding by fusing data from the depth and RGB camera; the latter being used to extend the information recovered by the depth planar segmentation. In this case, the presence of obstacles or walls is communicated to the user with audio signals.

However, these works do not deal specifically with the detection of stairs. Different approaches with different types of sensors have been used for detecting stairs. For instance, conventional cameras were used by Se and Brady [15]. They use grayscale images to detect and estimate the orientation and slope of staircases in order to help partially sighted people. In the same vein, Hernandez *et al.* [16] propose to find the diagonal lines corresponding to the handrails to select stair candidates. Hesch *et al.* [17] focus on detecting descending staircases for small ground robots in both far and near distance. Stairs are usually characterized by a distinctive shape formed by a flight of steps, which makes the sensors capable of measure depth the most appropriate for the task, as they provide richer information. In [18], Ishiwata *et al.* develop a visually impaired assistant using a small laser range sensor. Lu *et al.* [19] combine the use of the geometry information provided by a stereo system with the RGB data to make the system less prone to error. Pradeep *et al.* [20] use stereo vision to estimate normals and planes of the scene. Gutmann *et al.* proposed a stair detection algorithm for humanoid robots in [21] where the stereo vision system segments the scene into planar surfaces.

The laser-based sensors are often expensive and heavy, whereas the stereo cameras have troubles in scenes with poor textures. The recent appearance in the consumer market of modern RGB-D cameras such as Microsoft Kinect or Asus Xtion Pro Live have become a new and powerful election to work on this topic. Using structured IR light technology, the RGB-D cameras provide a dense depth map in every frame right away. Some authors make use of these sensors applying machine learning algorithms to perform staircase detection. In the case of [22], Filipe *et al.* use neural networks to detect the presence of obstacles and classify scenes captured by the depth camera among ascending staircase, descending staircase or none. Wang and Tian used a similar approach in [23], where the groups of parallel concurrent lines in the RGB image detected by the Hough transform are classified between stairs and pedestrian crosswalks using the depth information.

Other authors preferred the usage of geometrical reasoning instead of machine learning to detect staircases with RGB-D [24, 25, 26]. This is the approach we also consider to solve this problem. The absence of the retrieval of the fully measured model of the staircase in [24, 25] leads to misdetections, as our comparison of results prove. They use a RANSAC approach for finding planes in the scene that outputs at each step a set of points at certain height not using any other shape constraint but the sum of sufficient points. Delmerico *et al.* in [26] proposed an ascending stairway localization and modelling with the goal of checking for traversability and enable autonomous multi-floor exploration. The stair edge detection, which is the starting point of their algorithm, is based on abrupt changes in depth that only appear in ascending staircases when the sensor is lower than the steps, requiring *e.g.* a small robot.

The traversal of staircases has not been treated thoroughly regarding visually impaired aids, but there have been some approaches in humanoid robotics. Oßwald *et al.* in [27] studied two planar segmentation approaches to model staircases using the laser range data acquired by tilting the head of a humanoid robot. In [28] they show how the model obtained is used as input for the traversal of a spiral staircase combined with the laser data, the joint encoders, and an IMU to localise and retrieve the pose of the robot during the climb. The information is fused with the edge detection of the images from a camera pointing downwards to help the robot refine his pose for stair climbing. Our work present some similarities, although only the RGB-D sensor is used to compute the model, the pose, and the close-range refinement.

3. Scene segmentation and classification

In a visual assistant, in order to perform any complex task it is necessary to recognise the features in the surroundings. The starting point in this work is the partition of the environment in clusters of points by a segmentation process (Section 3.1). To give context to these segments it is necessary to calculate how the scene is oriented with respect to the user (Section 3.2). Once we have calculated the main transformations of the scene it is possible to classify the planar segments according to their orientation and localise the step candidates (Section 3.3).

3.1. Segmentation

In most human-made scenarios, the basic structure of the scene is a combination of planes at different orientations. Range sensors have proven to be extremely helpful for planar segmentation, and many algorithms have been developed through the years [29, 30, 31, 32]. We use the algorithm from [30], integrated in the Point Cloud Library (PCL) [8]. This algorithm outputs delimited regions of points with similar normal orientations and spatially lying on a unique 3D plane. Prior to this phase the point cloud is filtered to reduce the amount of data and then a normal extraction algorithm is applied with the following stages:

Downsampling. Each point cloud has a large quantity of points (640×480) which provides redundant information and makes further computations highly time-consuming (Fig. 3 (a)). Thus, the first operation is downsampling. We apply the 3D voxel grid filter from [8] to the point cloud, *i.e.* a 3D division of the space in a grid of 3D boxes (voxels) inside of which there is only one point (the centroid) instead of the initial set of points contained. The size of the edges of the voxels is determined by balancing time consumption and accuracy. Big voxels improve the performance rate but reduces the accuracy of the models. Typically,

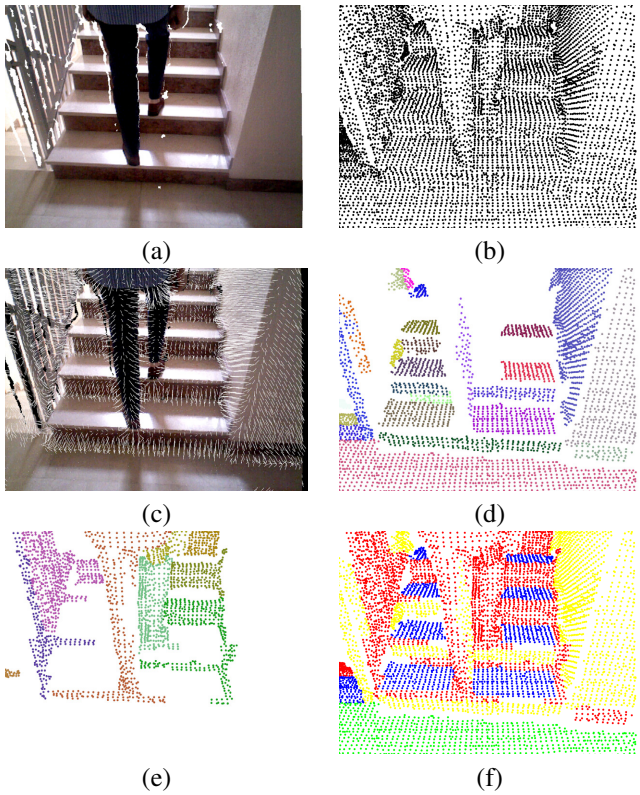


Fig. 3. Example of the segmentation process used in this work: (a) Initial coloured point cloud as retrieved by the camera. (b) Cloud after applying a voxel grid filter. (c) Normal extraction. (d) After the region-growing algorithm, where the planar regions are coloured in random colours. (e) Non-planar clusters from the Euclidian Cluster Extraction algorithm coloured randomly. (f) Classification of planes (yellow = vertical, blue = horizontal, green = floor, red = others). Best viewed in colour.

a size of voxels of about 3 − 4cm worked well for us. This is a common algorithm widely used for downsampling point clouds, which also helps removing noise and smoothing the surfaces. As a result of the downsampling we define the point cloud $P^C = \left\{\mathbf{p}_1^C, \mathbf{p}_2^C, ..., \mathbf{p}_i^C, ...\mathbf{p}_{n-1}^C, \mathbf{p}_n^C\right\}$, where $\mathbf{p}_i^C = \left(x_i^C, y_i^C, z_i^C\right)$ represents each of the n points in the scene in the camera reference C (Fig. 3 (b)).

Normal estimation. The surface normal estimation is based on the Principal Component Analysis (PCA) [8], consisting in the analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbours of every point $\mathbf{p}_i$. The eigenvector associated with the smallest eigenvalue corresponds to the normal direction $\mathbf{n}_i$ (Fig. 3 (c)). We considered the neighbours in a small radius (5cm around the points) to reduce the computation time and be able to detect sharper edges. In this process the curvature of the surfaces c_i is also computed to feed the following stage.

Region-growing. This algorithm [30] starts from a seed, which is the point with minimum curvature, and then expands the region towards the neighbouring points that have small normal deviation and similar curvature value. The neighbouring points which satisfy the normal and curvature threshold become the new seeds and the process is repeated until the region cannot expand any more. Then, a new initial seed is chosen among the remaining points, and the process starts over until the regions found are smaller than a certain pre-established threshold. The thresholds we set in normal deviation and curvature values are respectively 6° and 0.5. The minimum size of a region is set at 50 points.

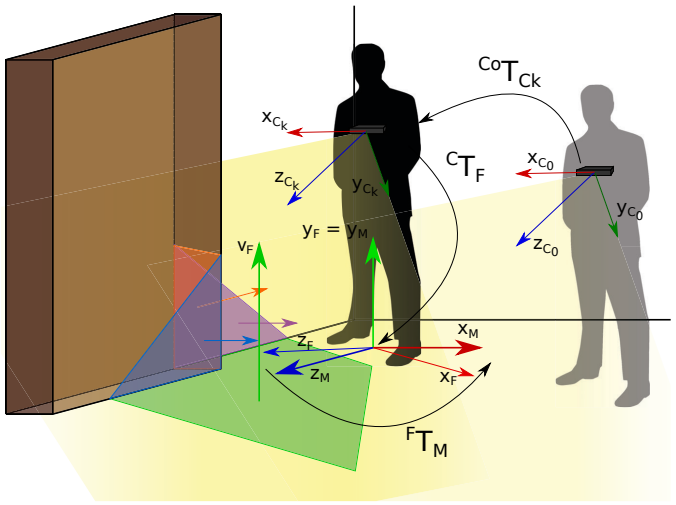


Fig. 4. Main transformations of the algorithm. ${}^{C_0}\mathbf{T}_{C_k}$: from the initial camera reference frame ($C_{t=0} = C_0$) to the camera in $t = k$ (C_k). ${}^{C}\mathbf{T}_{F}$: from the camera reference frame (C) to the floor (F). ${}^{F}\mathbf{T}_{M}$: Transformation from the floor reference frame (F) to match the Manhattan directions (M). Best viewed in colour.

We get as output a set of regions $R^C = \{R_k^C\}$ where $R_k^C \subseteq P^C$ (Fig. 3 (d)). Usually all $\mathbf{p}_i^k = \left(x_i^k, y_i^k, z_i^k\right) \in R_k^C$ lie on a plane $A^k x_i^k + B^k y_i^k + C^k z_i^k + D^k = 0$ where (A^k, B^k, C^k, D^k) are the plane coefficients of R_k^C, but they can also form a curved surface with smooth transitions. As the ground, walls, doors or steps are all planes, it is important to check this condition. A RANSAC algorithm seeks for the biggest plane in each region. If most of the points are inliers (we set more than 80%), it is considered a planar surface with the plane equation obtained and therefore normal vector $\mathbf{n}_k^C = (A^k, B^k, C^k)$ and distance to the origin D^k.

Euclidean cluster extraction. The points still not belonging to any region go through a cluster extraction algorithm which establishes connections and forms separate entities just by looking at their Euclidean position in the scene and ignoring normal orientations. In this operation we group in the same instance all the points that form isolated objects in the scene in a set of clusters $C^C = \{C_k^C\}$ where $C_k^C \nsubseteq R^C$ (Fig. 3 (e)).

3.2. Scene orientation

As the points of the cloud are referenced to the camera, a change of the reference system is necessary to determine the position with respect to the user in order to provide a more natural way to understand the environment and the movements of the person. Besides, the fact that most human-made indoor scenarios are composed by planes situated in three dominant orientations can be used in our benefit.

3.2.1. Camera to floor

The information from the camera is not very useful on its own when reasoning about the scene if the relative position of the camera to the world is unknown. We move the reference frame from the camera to the floor by computing the transformation ${}^{C}\mathbf{T}_{F}$ as shown in Fig. 4. That way all the planes will be properly oriented and the height of the points with respect to the floor allows to draw conclusions about the nature of the objects in the scene.

The computation of this transformation requires to find the floor plane (in Fig. 4 the green plane with normal $\mathbf{v}_F$). As no other sensor has been used for this task, the only previous knowledge is the approximate location of the camera on the chest. A RANSAC procedure is used to find the biggest planes one by one, and the relative distance and orientation of each plane with respect to the camera are analysed to determine whether it is floor or not. The rules to verify this are:

- The orientation of the normal must be coherent with the orientation of the camera in the chest. For example, in Fig. 4 it would be an approximate rotation of $\approx 180°$ in $\mathbf{z}_C$ and of $\approx 45°$ in $\mathbf{x}_C$ so the $\mathbf{y}_F$ matches the floor normal $\mathbf{v}_F$.
- The distance of the plane to the camera should be within a provided valid range which depends on the height of the user.
- It is very likely that the floor appears close to the subject, as the camera points down. So we can consider for floor detection points closer than a certain threshold in $\mathbf{z}_C$.

This computation requires relaxed thresholds to not discard valid portions, as conditions can vary due to the movement and the height of the subject; but they should quickly discard planes belonging to instances such as walls or tables. In our implementation the inclination of the camera is not restricted to 45°, but within a valid range of $20° - 70°$ and the height of the camera from the floor within $1 - 1.6$m. The distance to consider points as inliers in the RANSAC is the voxel edge size. The last condition could be useful to discard the planes which are extremely close to the floor, such as steps, which could deceive the algorithm. The algorithm typically start searching within $\mathbf{z}_C = 1$m and progressively increasing the threshold until a valid plane is found.

3.2.2. Floor to Manhattan

We assume that most indoor scenes satisfy the Manhattan World assumption [10], *i.e.* most planes have normals in three mutual orthogonal directions. This reduces a lot of reasoning about the environment. In our case of study we are going to use it to retrieve the directions of the stairs, as most certainly lie in that convention. The stair must be properly oriented to get the model that fits the points better.

To acquire the Manhattan directions $(\mathbf{i}_M, \mathbf{j}_M, \mathbf{k}_M)$ in the scene we can take advantage of the previous transformation floor ${}^{C}\mathbf{T}_{F}$, as it already matches the vertical direction ($\mathbf{j}_M = \mathbf{y}_F$). The problem is then reduced to find $\mathbf{i}_M$ and $\mathbf{k}_M$. After the segmentation we have a set of planes with their normal directions and their number of points. The two orthogonal directions which satisfy the greater number of points are the $\mathbf{i}_M$ and $\mathbf{k}_M$. To solve the ambiguity of these directions we choose as $\mathbf{k}_M$ the one that have less angle with the vector pointing out to the front of the user ($\mathbf{z}_F$). In Fig. 4 the $\mathbf{x}_M$ share directions with the blue and purple planes and $\mathbf{z}_M$ with the orange one. Throughout iterations the ${}^{F}\mathbf{T}_{M}$ is recalculated but maintaining the orientation convention assumed.

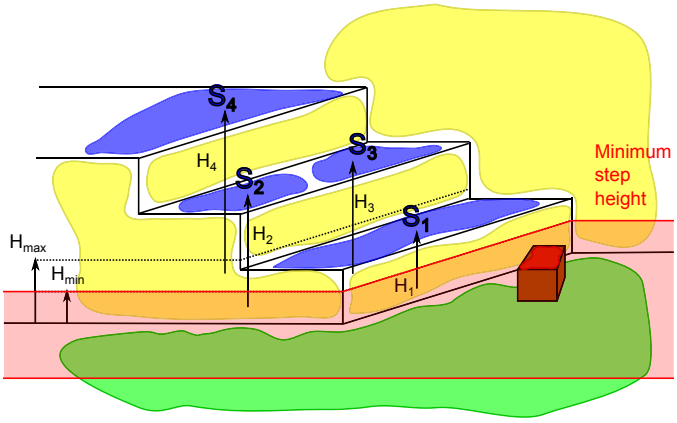


Fig. 5. Classification of planes regarding orientation and height. Horizontal planes are classified among Floor (green), Obstacles (red) or Step candidates (blue) depending their distance to the floor. Vertical planes (yellow) are used to compute ${}^{F}\mathbf{T}_{M}$. Best viewed in colour.

3.3. Classification of regions

Once we have the planar regions with their normals and the transformations to the Manhattan directions, we can transform the regions R^C to the new reference frame R^M. The regions are then classified regarding their orientation and relative position. To consider a plane parallel or perpendicular to another a threshold of 10° is considered in every case.

The planes which are perpendicular to the floor are classified as vertical (*e.g.* walls, doors, risers, furniture). These are the planes used to obtain the Manhattan directions of the environment. As we have these directions, extended reasoning about the orientation of the walls can be done and new subcategories of vertical planes can be added (left, right, frontal).

The planes which are parallel to the floor are horizontal. The distance of the planes to the floor (H_k) is considered for further analysis, considering the dimensions regulated by the Technical Edification Code[1] (Fig. 5). According to the Code the vertical distance between two consecutive steps ranges from a minimum $H_{min} = 13$cm to a maximum $H_{max} = 18.5$cm. Horizontal regions are considered as step candidates if they are situated above (in ascending stairways) or below (in descending ones) H_{min} from the floor. The floor must have height zero given the transformation ${}^{F}\mathbf{T}_{M}$ and other planes whose height does not fit above descriptions are considered obstacle. To evaluate this condition it is necessary to add a threshold as

[1] Código Técnico de la Edificación (2006). CTE, DB-SUA, section 1.4.2., http://www.codigotecnico.org/

the measurements can be noisy. Other size and shape restrictions are kept to a minimum at this point because they could discard valid portions of steps which might be useful for a better modelling of staircases. From this operation we get a set of step candidates $S = \{S_1, S_2, ..., S_i, ...S_{n_s-1}, S_{n_s}\}$ where $S_i = \{R_k \mid \|\mathbf{n}_k \times \mathbf{y}_F\| \approx 0, |H_k| \geq H_{min}\}$ and n_s is the number of step candidates. The existence of a set of at least one step candidate activates the stair detection algorithm.

The planes which are not perpendicular nor parallel to the ground are kept also as obstacles and removed from further analysis. In Fig. 3 (f) there is an example of the classification.

4. Stair detection and modelling

The step candidates obtained in Section 3.3 are the input of the stair detection and modelling algorithm, whose output consists in the detection and retrieval of a scaled model of the staircase. At this moment, the algorithm is functional with both ascending and descending staircases. Isolated single steps can also be detected. The algorithm is able to overcome partial occlusions of a stairway splitting steps in more than one region in the detection. Spiral staircases can be detected but the modelling part has not been addressed yet.

4.1. Stair detection

The detection consists in determining whether the n_s step candidates form a staircase or not. The algorithm establishes connections among the candidates to discard the ones that do not belong to the staircase and to group the regions that do belong in *levels* according to the distance in steps to the floor. The connectivity between step candidate regions S_A and S_B has been computed considering there is at least a minimum pre-established number of points from S_A inside a valid range of distances from S_B. In this case, we set a radius of 0.5m for the neighbour search and a minimum amount of 10 points to set a valid connection. The candidates are analysed one by one starting from the closest to the floor, and verifying the connections to the previously established levels every time (Fig. 6 (a)).

We define a set of levels $L = \{L_j\}$, where $j = 0..n_L$ and n_L is the highest level detected. The level zero is occupied by the floor, so initially $n_L = 0$. The candidates whose centroid is between H_{min} and H_{max} to the ground constitute the first step candidates (Fig. 6 (b)). The first step must be connected to the floor if it is present in the image (Fig. 6 (c)). If no first step candidate satisfies neighbouring conditions, the algorithm determines there is no staircase. Otherwise, $L_1 = S_1$ (where $H_{min} \leq H_1 \leq H_{max}$) is established, and $n_L = 1$.

The algorithm takes the remaining step candidates by height and starts testing connectivity and height conditions to determine whether they belong to a new (Fig. 6 (d)) or to the current level (Fig. 6 (e)). In case they belong to the current level (a step candidate is considered the same height if it is within ±3cm), the step candidate regions fuse in one single point cloud forming the level. If they have no connection to previous levels (*e.g.* a horizontal plane correspondent to a table) they are classified as obstacles (Fig. 6 (f)). As a result, a set of connected regions

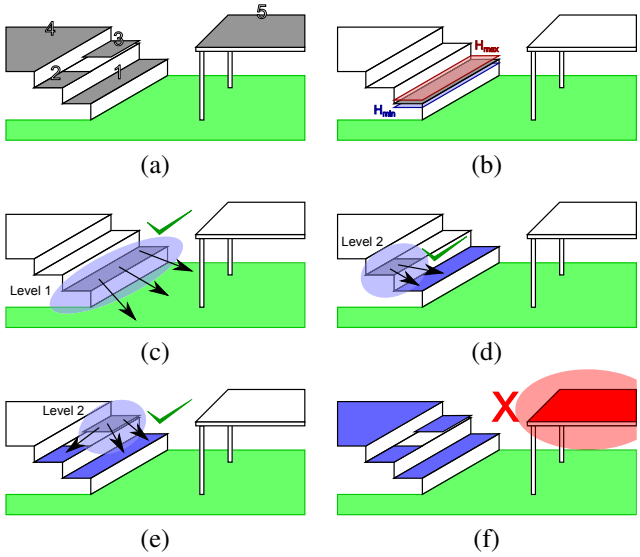


Fig. 6. Explicative sketches for the stair detection algorithm. (a) Select the candidates in order. (b) First step must be in the valid range of heights. (c) First step must be connected to the floor if it is visible. (d) and (e) The connectivity to previous levels is checked doing a neighbour search. (f) If the candidate is not connected to the previous level, it is not part of the staircase.

Algorithm 1 Stair detection algorithm

```
Step candidates list S = {S_1, S_2, ..., S_i, ...S_{n_s-1}, S_{n_s}}
Levels list L = {Ø}
L_0 = R_floor;
n_L = 0;
S = sortByHeight(S);
if (S_1.height > H_max) then
    stair = false;
else
    stair = true;
    for i = 1 : n_s do
        S_i.is_connected = false;
        for j = 0 : n_L do
            if (areConnected(S_i, L_j)) then
                S_i.is_connected = true;
            end if
        end for
        if S_i.is_connected then
            if (S_i.height ≈ L_{n_L}.height) then
                L_{n_L}.points ∪ S_i.points;
            else
                n_L = n_L + 1;
                L_{n_L} = S_i;
            end if
        end if
    end for
end if
return stair
```

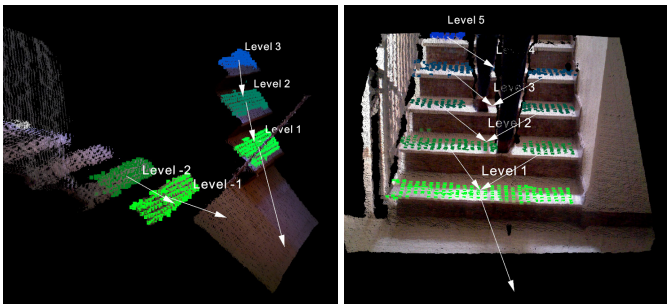


Fig. 7. Example of stair detection with both ascending and descending stairs (left) and with more than one region per level (right). The connectivity is traced with white arrows.

corresponding to different levels is obtained (Fig. 7). The algorithm is summarized in Algorithm 1. When all the candidates have been checked and the number of levels is greater than one, the system proceeds with the modelling of the staircase.

A special case occurs when there is only one step candidate. It might either actually be the first step of a staircase, or be just a single curb on the way. Curbs are a singular case, sometimes omitted by stair detection algorithms, but dangerous as well. Here, more strict area and shape analysis can be applied in order to determine in which case we are, as we know the measurements of the steps according to the regulations. For example, if the candidate is within the valid range of heights but does not satisfy size conditions because is too small then it is an obstacle. If it is too big then it is floor at another level and it is a single curb.

4.2. Stair modelling

A staircase is basically a flight of steps between two floors. The shape of the staircases can have some differences regarding the presence of absence of the riser or the inclination it might have. We are going to consider a unified model for all the possible cases, where the steps are defined by a horizontal rectangular plane of $l_{width} \times l_{length}$ and a vertical rectangular plane of $l_{width} \times l_{height}$ which links the horizontal plane to the previous level. The line where two planes intersect is called the *edge* of the step. Every staircase is also oriented according to three orthogonal directions whose pose with respect to the user is relevant to guide the subject towards it. In the modelling phase we are going to retrieve the l_{width}, l_{length}, l_{height} and ${}^{C}\mathbf{T}_{S}$ as depicted in Fig. 8. The model can be then drawn for the number of levels detected in the previous stage (n_L). If the traversal of the staircase is then performed, the final number of levels can be obtained, with the procedure explained in Section 5.

The extraction of the measurements is correlated with the extraction of the ${}^{C}\mathbf{T}_{S}$, for which first we need to define the three main directions of the stair, $(\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S)$. We have developed two ways to do this:

- Considering stairs are oriented according to the Manhattan assumption (*i.e.* $(\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S) = (\mathbf{x}_M, \mathbf{y}_M, \mathbf{z}_M)$).
- Method based on the Principal Component Analysis of the step candidates.

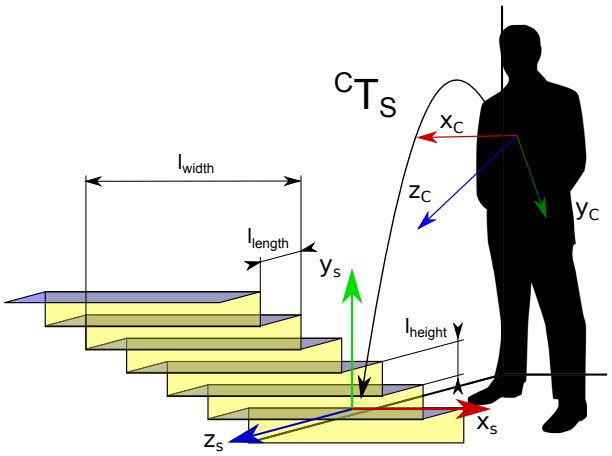


Fig. 8. Parameters to compute for the modelling: l_{width}, l_{length}, l_{height} and ${}^{C}\mathbf{T}_{S}$.

The better option of the two depends on the type of scene: scenes populated with big structural planes (coming from a side wall or the risers, for instance) work better with the first one, whereas scenes with mainly horizontal planes (e.g. isolated stairs, no risers) work better with the second. The Manhattan World option is used when there are enough planes with normals in horizontal directions that allow us to call the scene "Manhattan World scene". We determine this circumstance by a threshold empirically set at a percentage of "Manhattan points" of the total number of points. The PCA option is more time consuming, so only is used when there is not enough evidence of the Manhattan directions of the scene. This method can always be applied, as it is based on the step candidates already detected (i.e. if there is no step candidates the modelling does not even start). However, when the observation of the steps is partial due to occlusions it could lead to erroneous solutions (hence, it is the second choice). From here on the PCA method is detailed as follows:

From the last stage we have a level-organised set of points corresponding to each visible step. We can perform a Principal Component Analysis (PCA) in every set of points to retrieve their main directions and initial estimate of their measurements by defining the bounding rectangle which encloses all the points in these directions. From this procedure we get a collection of principal components and measurements which presents high variance (Fig. 9 (a)). To solve this we choose one step as best initial guess: the one with greater extent. We define *extent* as the ratio of the area of the concave hull and the area of the rectangle as defined in Fig. 9 (b). The principal components of this step are rotated to match the normal of the floor (if visible) and then the other axes are rotated until the sum of areas of the bounding rectangles using these directions in all steps is minimised. With this last operation you make sure that the axis of the model fits the points of the stair in the best way (Fig. 9 (c)).

Once the directions have been computed, the bounding rectangles of each step present different dimensions, so the final stage consists in defining the global dimensions of the staircase. The l_{width} can be chosen as the largest value of all, whereas the

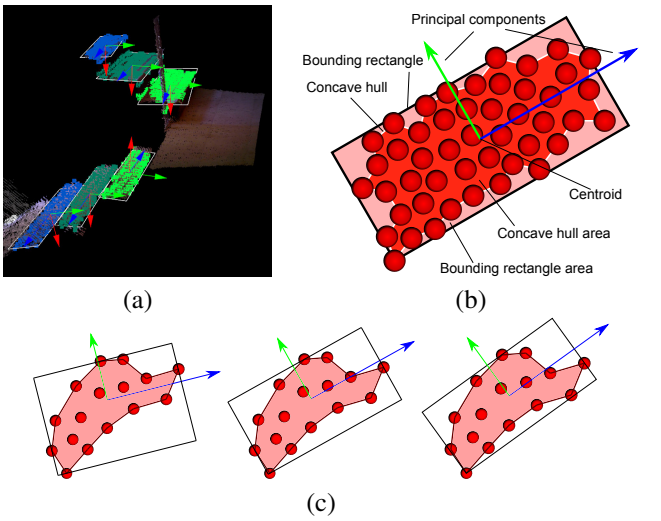


Fig. 9. (a) Principal components for each step coloured in order (blue-green-red) and bounding rectangle in white. (b) Illustrative sketch of the different components involved in the standalone algorithm. (c) Example of the rotation of the selected axis to minimise the area of the bounding box with the points of one single step. It needs to be done with all the steps at the same time in order to obtain the final direction that fits best the staircase.

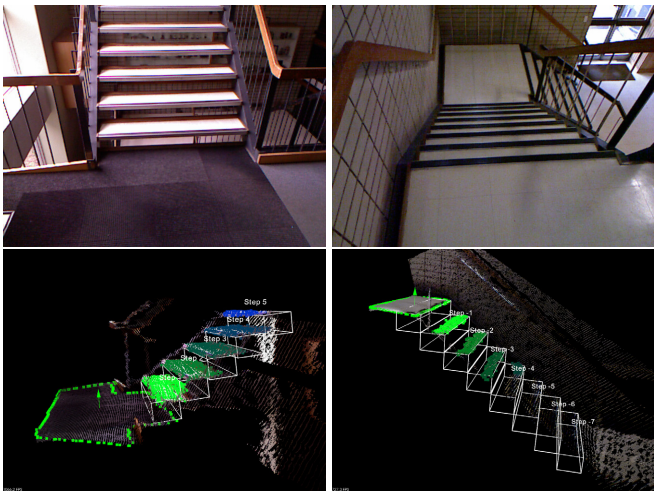


Fig. 10. The length of the steps for our model does not usually match the length of the bounding rectangles of the steps as detected. For instance, non-existent risers in ascending staircases (top-left) or any descending staircase (top-right). The step length is shortened (bottom-left) or extended (bottom-right) until the edge of the step above.

l_{height} is considered the average of vertical distances between the centroids of consecutive steps. The l_{length} is the average horizontal distance between the edge of every two consecutive steps. The definitive length of the steps is computed this way because the vertical projection of the bounding rectangles of two consecutive steps usually overlaps in ascending staircases due to inclining or non-existent risers, or leaves a gap in descending staircases due to self occlusions (Fig. 10). It causes that the length you see in ascending staircases is more than what we use for our model, whereas in descending staircases some valid portion of the step is hidden.

Once we have all the parameters, we can use them to validate the staircase detection or discard false positives, as we know the appropriate dimensions steps must have by regulations. The ${}^{S}\mathbf{T}_{C}$ can be obtained using as rotation matrix the three stair directions as described, and as translation vector the coordinate in the camera reference frame of the centre of the edge of the first step of the staircase. Alternatively, the translation part can direct the subject to one of the ends of the edge of the first step, where the handrails are expected to be.

5. Odometry-aided stair traversal

Odometry is the process of using the data from sensors to retrieve the change of position and orientation over time. When the sensors being used are cameras it is called visual odometry. In this work we estimate the visual odometry using the information from both RGB and depth cameras. This information does not need to be used all the time, but it can be of great help as it adds some kind of memory to the system.

Calling the initial reference frame of the camera C_0, and the reference frame in the instant k as C_k, we define the transformation provided by the visual odometry as ${}^{C_0}\mathbf{T}_{C_k}$ (Fig. 4). In theory, this transformation could be used to displace any feature captured at any given time to a common geometrical reference. In practice, the sensor localization has some drift, which increases through iterations. As our goal is far away from mapping, we use the odometry as a rough estimate of the position of the stairs when they are no longer visible.

However, when there are known structures in the image, we can use that information to correct the drift. We do that during staircase traversal, where we can use the currently seen positions of the steps to correct that drift.

5.1. Visual odometry module

For the visual odometry from RGB-D there are many approaches [33, 34]. We use the method presented by Gutierrez-Gomez *et al.* [7], where visual odometry is obtained in real time from the dense RGB and inverse depth maps by establishing pixel-wise constraints through the flow equations.

Let us denote two camera frames as A and B, at instants t and $t + \Delta t$ respectively. Given the intensity images $\mathcal{I}_A$ and $\mathcal{I}_B$, and inverse depth maps $\mathcal{W}_A$ and $\mathcal{W}_B$ defined over the image domain $\Omega \subset \mathbb{P}^2$, for an image point $\mathbf{p} = (u,\ v,\ 1)^T \in \Omega$ in frame A, the following photometric and geometric constraints hold:

$$\mathcal{I}_B(\mathbf{p} + \mathbf{\Delta p}) = \mathcal{I}_A(\mathbf{p}) \tag{1}$$

$$\mathcal{W}_B(\mathbf{p} + \mathbf{\Delta p}) = \frac{1}{\mathbf{e_z}^T \mathbf{X}_B}, \tag{2}$$

where $\mathbf{X}_B$ is the 3D point lifted from pixel $\mathbf{p} + \mathbf{\Delta p}$ in frame B, $\mathbf{\Delta p} = (\Delta u,\ \Delta v,\ 0)^T$ is the displacement of one point from frame A to B, and $\mathbf{e}_\mathbf{z}^T = (0,\ 0,\ 1)$. The photometric constraint assumes constant illumination of one scene point over time. The geometric constraint is the measurement model of the depth sensor at frame B in inverse depth parametrisation.

Assuming small pixel displacements between frames we compute the flow equations from (1) and (2):

$$\nabla \mathcal{I}_A(\mathbf{p})\Delta\mathbf{p} + \mathcal{I}_B(\mathbf{p}) = \mathcal{I}_A(\mathbf{p}) \tag{3}$$

$$\nabla \mathcal{W}_A(\mathbf{p})\Delta\mathbf{p} + \mathcal{W}_B(\mathbf{p}) = \frac{1}{\mathbf{e}_\mathbf{z}^T\mathbf{X}_B}, \tag{4}$$

where the gradient operators $\nabla\mathcal{I} = \left(\frac{\partial\mathcal{I}}{\partial u}, \frac{\partial\mathcal{I}}{\partial v}, 0\right)$ and $\nabla\mathcal{W} = \left(\frac{\partial\mathcal{W}}{\partial u}, \frac{\partial\mathcal{W}}{\partial v}, 0\right)$.

Using the camera projection and inverse projection models, $\mathbf{p} = \pi(\mathbf{X}) = \mathbf{K}\frac{\mathbf{X}}{\mathbf{e}_\mathbf{z}^T\mathbf{X}}$ and $\mathbf{X} = \pi^{-1}(\mathbf{p}) = \frac{1}{\mathcal{W}(\mathbf{p})}\mathbf{K}^{-1}\mathbf{p}$, and with the same assumption of small pixel displacement we get:

$$\begin{aligned}\frac{1}{\mathbf{e}_\mathbf{z}^T\mathbf{X}_B} &= \frac{1}{\mathbf{e}_\mathbf{z}^T\mathbf{X}_A} - \frac{1}{(\mathbf{e}_\mathbf{z}^T\mathbf{X}_A)^2}\mathbf{e}_\mathbf{z}^T\Delta\mathbf{X}_\mathbf{p} + O\left(\left\|\mathbf{e}_\mathbf{z}^T\Delta\mathbf{X}_\mathbf{p}\right\|^2\right)\\ &\approx \mathcal{W}_A(\mathbf{p}) - \mathcal{W}_A^2(\mathbf{p})\mathbf{e}_\mathbf{z}^T\Delta\mathbf{X}_\mathbf{p}.\end{aligned} \tag{5}$$

$$\begin{aligned}\Delta\mathbf{p} &= \mathbf{K}\frac{\mathbf{X}_B}{\mathbf{e}_\mathbf{z}^T\mathbf{X}_B} - \mathbf{K}\frac{\mathbf{X}_A}{\mathbf{e}_\mathbf{z}^T\mathbf{X}_A}\\ &= \mathbf{K}\mathbf{X}_B\left(\mathcal{W}_A(\mathbf{p}) - \mathcal{W}_A^2(\mathbf{p})\mathbf{e}_\mathbf{z}^T\Delta\mathbf{X}_\mathbf{p}\right) - \mathbf{K}\mathbf{X}_A\mathcal{W}_A(\mathbf{p})\\ &= \mathcal{W}_A(\mathbf{p})\left(\mathbf{K} - \mathbf{p}\mathbf{e}_\mathbf{z}^T\right)\Delta\mathbf{X}_\mathbf{p}.\end{aligned} \tag{6}$$

where $\Delta\mathbf{X}_\mathbf{p}$ is the 3D flow associated to each pixel in frame A. Substituting in (3) and (4), we get the linear constraints on this pixel-wise 3D flow.

$$\mathcal{W}_A(\mathbf{p})\nabla\mathcal{I}_A(\mathbf{p})\left(\mathbf{K} - \mathbf{p}\mathbf{e}_\mathbf{z}^T\right)\Delta\mathbf{X}_\mathbf{p} + \mathcal{I}_B(\mathbf{p}) - \mathcal{I}_A(\mathbf{p}) = 0 \tag{7}$$

$$\begin{aligned}\mathcal{W}_A(\mathbf{p})\Big(\nabla\mathcal{W}_A(\mathbf{p})\left(\mathbf{K} - \mathbf{p}\mathbf{e}_\mathbf{z}^T\right) + \mathcal{W}_A(\mathbf{p})\mathbf{e}_\mathbf{z}^T\Big)\Delta\mathbf{X}_\mathbf{p} +\\ +\mathcal{W}_B(\mathbf{p}) - \mathcal{W}_A(\mathbf{p}) = 0.\end{aligned} \tag{8}$$

Assuming that the scene is rigid, the 3D flow map $\Delta\mathbf{X}_\mathbf{p}$ is produced only by a small interframe camera motion, described by the rotation translation pair $({}^B\mathbf{R}_A, \mathbf{r}_B^A) \in \mathbb{SE}(3)$:

$$\begin{aligned}\Delta\mathbf{X}_\mathbf{p} &= {}^B\mathbf{R}_A\mathbf{X}_A + \mathbf{r}_B^A - \mathbf{X}_A\\ &= \left(\mathbf{I} + [\boldsymbol{\theta}_B^A]_\times\right)\mathbf{X}_A + \mathbf{r}_B^A - \mathbf{X}_A + O\left(\left\|[\boldsymbol{\theta}_B^A]_\times^2\mathbf{X}_A\right\|\right)\\ &\approx \mathbf{r}_B^A - [\pi^{-1}(\mathbf{p})]_\times\boldsymbol{\theta}_B^A = \mathbf{M}(\mathbf{p})\boldsymbol{\xi}_B^A.\end{aligned} \tag{9}$$

Eq. (9) leads to a well-posed problem with 6 unknowns corresponding to the camera motion parameters for nearly $W_{im}H_{im}$ constraints (where W_{im} and H_{im} are the width and height of the image respectively), excluding pixels without depth measurements, with the following residuals:

$$\begin{aligned}r_\mathcal{I}(\mathbf{p},\boldsymbol{\xi}) = \mathcal{W}_A(\mathbf{p})\nabla\mathcal{I}_A(\mathbf{p})(\mathbf{K}-\mathbf{p}\mathbf{e}_\mathbf{z}^T)\mathbf{M}(\mathbf{p})\boldsymbol{\xi} +\\ + \mathcal{I}_B(\mathbf{p}) - \mathcal{I}_A(\mathbf{p})\end{aligned} \tag{10}$$

$$\begin{aligned}r_\mathcal{W}(\mathbf{p},\boldsymbol{\xi}) = \mathcal{W}_A(\mathbf{p})\Big(\nabla\mathcal{W}_A(\mathbf{p})(\mathbf{K}-\mathbf{p}\mathbf{e}_\mathbf{z}^T)+\mathcal{W}_A(\mathbf{p})\mathbf{e}_\mathbf{z}^T\Big)\mathbf{M}(\mathbf{p})\boldsymbol{\xi} +\\ + \mathcal{W}_B(\mathbf{p}) - \mathcal{W}_A(\mathbf{p}),\end{aligned} \tag{11}$$

which can be straightforwardly minimised by standard Gauss-Newton least squares.

In practice we do not apply conventional least squares. Instead we use a robust cost function by applying iteratively reweighted least squares algorithm [35]. We also follow a coarse-to-fine approach using a 3 level image pyramid, performing a number of 10 iterations on a pyramid level before stepping down to the next finer level. The incremental motion estimate at each iteration γ is computed as:

$$\boldsymbol{\xi}_B^{A(\gamma)} = \underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{\mathbf{p}\in\Omega} \omega\left(\frac{\check{r}_\mathcal{I}(\mathbf{p})}{\sigma_{r_\mathcal{I}}}\right)\frac{r_\mathcal{I}^2(\mathbf{p},\boldsymbol{\xi})}{\sigma_{r_\mathcal{I}}^2} + \omega\left(\frac{\check{r}_\mathcal{W}(\mathbf{p})}{\sigma_{r_\mathcal{W}}}\right)\frac{r_\mathcal{W}^2(\mathbf{p},\boldsymbol{\xi})}{\sigma_{r_\mathcal{W}}^2}, \tag{12}$$

where $\check{r}_\mathcal{I}(\mathbf{p})$ and $\check{r}_\mathcal{W}(\mathbf{p})$ denote the initial residuals computed after warping intensity and inverse depth maps in frame B towards frame A with the estimated camera motion up to current iteration ${}^k\mathbf{T}_{k+1}^{(\gamma+1)}$. $\omega(x) = \frac{6}{5+x^2}$, since we use an estimator based on the Student's t-distribution with $\nu = 5$ as in [36], which shows in general better performance than other candidates. The scaling parameters are fixed to $\sigma_{r_\mathcal{I}} = 5$ and $\sigma_{r_\mathcal{W}} = 0.0025m^{-1}$ based on tests on static sequences and the disparity measurement model of RGB-D sensors [37].

After each iteration the motion estimate between frames k and $k+1$ is updated by the current incremental estimate:

$${}^k\mathbf{T}_{k+1}^{(\gamma+1)} = \begin{pmatrix} \exp([\boldsymbol{\theta}_B^{A(\gamma)}]_\times) & \mathbf{r}_B^A \\ 0 & 1 \end{pmatrix}^{-1} {}^k\mathbf{T}_{k+1}^{(\gamma)}. \tag{13}$$

Camera motion at first iteration ${}^k\mathbf{T}_{k+1}^{(0)}$ is initialised assuming a constant velocity, *i.e.*, ${}^k\mathbf{T}_{k+1}^{(0)} = {}^{k-1}\mathbf{T}_k$.

5.2. Stair traversal

In the stair detection and modelling stage, most of the computations are done on one single image at a time. However, in some circumstances, the way of proceeding must change to perform contextually appropriate operations. For instance, while the user is traversing the staircase instead of searching for stairs, the application should provide information about the current state of the traversal, *e.g.* in which step the user stands, which and where is the next step or how many steps are left to reach the other end. It is impossible to recover this kind of information in one single image analysis because all the steps usually look the same.

When the user stands so close to the staircase that the edge from the first step is no longer visible, the transformation ${}^C\mathbf{T}_S$ cannot be computed as described, and is then estimated using the visual odometry transformation. Calling k the last iteration when the ${}^{C_k}\mathbf{T}_S$ could be properly computed, it is possible to retrieve the ${}^{C_{k+n}}\mathbf{T}_S$ in the iteration $k+n$ if the transformations ${}^{C_k}\mathbf{T}_S$ and ${}^{C_0}\mathbf{T}_{C_k}$ have been kept:

$${}^{C_{k+n}}\mathbf{T}_S = ({}^{C_0}\mathbf{T}_{C_{k+n}})^{-1}\ {}^{C_0}\mathbf{T}_{C_k}\ {}^{C_k}\mathbf{T}_S \tag{14}$$

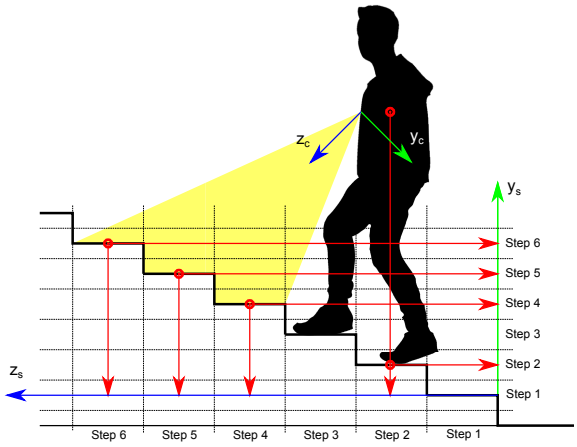


Fig. 11. The centroids of the steps and the estimated centroid of the body can be used to retrieve the numbers of the steps and the step the user is on by transforming the points to the stair reference frame.

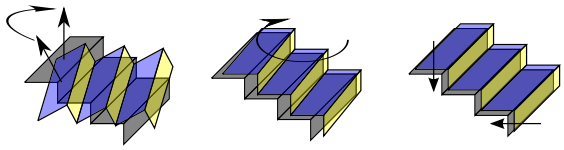


Fig. 12. To correct the stair pose estimated by the odometry we use the live depth information, drawn as the grey stair. (a) Correction of the orientation to match the vertical normal. (b) Correction of the orientation to match the edges of the steps in sight. (c) Correction in the position to match the centroids and edges of the steps. Best viewed in colour.

The current pose of the camera with respect to the stair can be computed at any time and it shows the translation with respect to the initial point. Since l_{height} and l_{length} of the current staircase are known from previous stages, the 3D position of the centroid of every step in sight transformed to the stair reference frame would reveal in which step it is (Fig. 11). With this information, every time a step of a level higher than the current n_L is detected, the number of levels of the staircase model is updated ($n_L = n_L + 1$). Similarly, transforming the estimated centroid of the body to the stair reference frame can be used to know the step in which the user is. In practice, it does not work as well as expected, because the visual odometry has a noticeable drift, specially in situations like this where a few centimetres can cause mismatches. The transformation ${}^{C_{k+n}}\mathbf{T}_S$ reveals an estimation of how the user has moved, or looking at it the other way, an estimation of how the stair is posed with respect to the user. The pose correction problem lies in computing the rotation and translation needed to match where the stair really is, *i.e.* as it is seen by the camera. We can use the live information of the depth camera to correct the drift.

It requires some slight changes of some previously commented algorithms. To compute the vertical direction, instead of looking for the floor all step planes are used to compute the resulting normal as we can certainly say they are horizontal planes (Fig. 12 (a)). The rotation in $\mathbf{y}_s$ can be computed as described, either trusting Manhattan estimation or the directions from the PCA (Fig. 12(b)). The rotation needed to move the estimated stair axes to the new ones is the correction in orientation. The translation part can be calculated by looking at the height of the centroids of the steps in $\mathbf{y}_s$ and the edge-points of the steps can reveal the translation in $\mathbf{z}_s$ (Fig. 12 (c)). In the $\mathbf{x}_s$ direction the translation cannot be retrieved and the odometry needs to be trusted.

Calling ${}^{S}\mathbf{T}_{S'}$ the drift correction transformation, the final transformation from the current camera reference frame to the stair reference frame is:

$$ {}^{C_{k+n}}\mathbf{T}_{S'} = {}^{C_{k+n}}\mathbf{T}_{S} \; {}^{S}\mathbf{T}_{S'} \tag{15}$$

Using the correction, the computation of the position of the subject is more reliable. When the step position estimation using the height is higher than the one provided by the length, it means that the user is currently climbing the step. The dimensions of the last step found are computed in order to determine when they are significantly larger than the step length, because that would mean that it is the last step. Once these last step is detected, the model of the staircase is updated with the final number of steps of the stair. When the user finally stands on the floor at another level, the stair traversal algorithm stops and the algorithm proceeds as usual by performing the detection and modelling.

6. Experiments

The experiments were carried out in a 3.4Ghz computer with a GPU Nvidia GeForce GT730 running Ubuntu 12.04, ROS Hydro and the library PCL version 1.8. In this work we include the experimental evaluation of the perception part of the system. No experiments with visually impaired people have been performed as we have not updated our human-computer interface [5] including stair navigation interaction yet. Although we already had our own recordings from previous research, new scenarios including stair traversal were also recorded to conduct specific experiments.

Tang et al. compiled a dataset in [24] which includes 148 captures made with a Microsoft Kinect sensor. 90 of them include RGB and depth snapshots of a set of staircases from different poses and the other 58 are normal indoor scenes to test for false positives. A sample of the results are shown in (Fig. 13). We tested for false positives and false negatives using this dataset and compared our results with the ones from [24] and [25] (Table 1). We achieve better results with the 0% of false negatives as in [25] but also reaching a 0% of false positives. The main reason of being completely successful was that although sometimes a bad floor detection or structures composed by parallel planes such as shelves caused the detection of a false positive, as we retrieve the measurements we could discard invalid staircases for being too narrow, or having too small or too big steps. Some discarded examples are shown in Fig. 14 (a).

We studied the step detection ratio according to the position of the step in the staircase using Tang's dataset (Fig. 15). The behaviour changes when we are facing an ascending staircase or

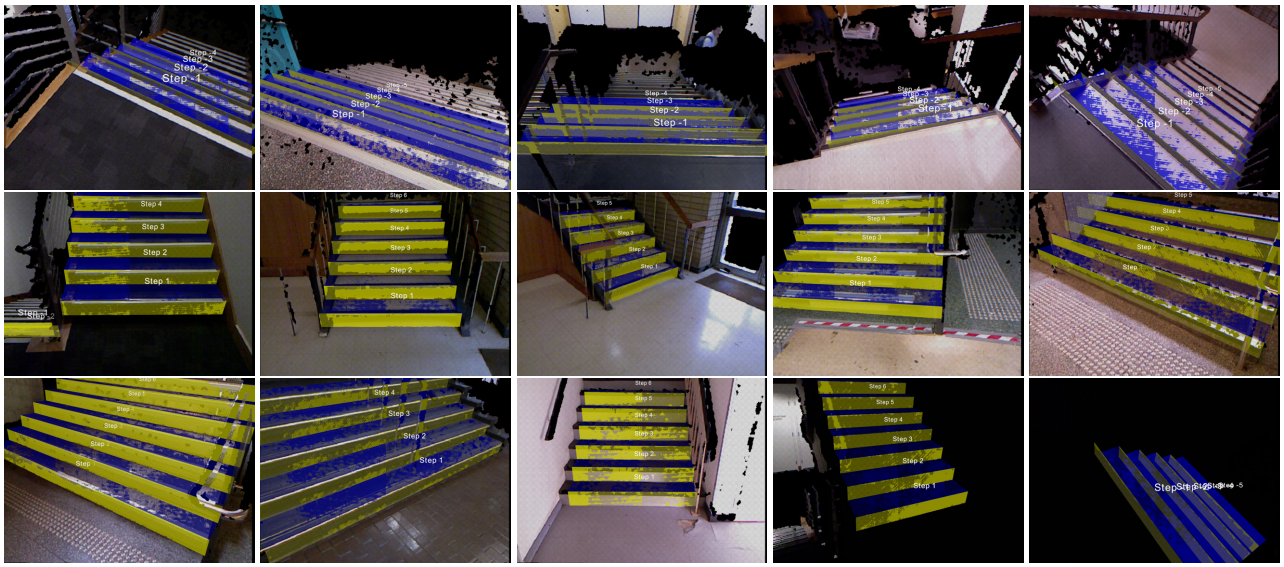


Fig. 13. Several examples of results obtained with Tang's dataset. The model of the staircases retrieved is superimposed in the point cloud for visual verification.

Table 1
Comparison of false negatives and false positives between our work and the one presented in [24, 25]

Evaluator	False negative	False positive
Tang [24]	5.07%	1.02%
Vlaminck QVGA [25]	0.00%	8.62%
Vlaminck VGA [25]	0.00%	3.45%
Our work	0.00%	0.00%

Table 2
Average and standard deviation (in centimetres) of the length and height measured with and without obstacles.

	No obstacles		Obstacles		Real
	$\bar{\mathbf{x}}$	σ	$\bar{\mathbf{x}}$	σ	$\mathbf{x_r}$
Length	29.003	2.013	29.389	1.887	30
Height	15.399	1.364	15.561	0.593	17

a descending one. Due to the orientation of the chest-mounted sensor, standing before a descending staircase allow us to see the whole staircase. However, the self occlusion of consecutive steps and the quality of the measurements decreasing with the distance, harm the detection of steps farther than the third position. In ascending staircases the ratio of detection diminishes in a less prominent way, because the steps remain almost as close to the subject as they rise, although with the penalty of having less and less visual angle. Steps higher than the seventh position are out of the field of view of the camera.

The modelling usually provides qualitatively good results with rectangular staircases unless the detection have severe obstructions, there is a strong influence of the sun or the stairs present atypical constructions (*e.g.* Fig. 14 (b)). We have quantitatively analysed the resemblance of the model to the real staircase. We have excluded the width from the analysis as the view of the stairs may be partial and it is not as relevant as the other measurements. After computing the height and length of a staircases, in both ascending and descending perspectives, from different viewing angles, the results were compared to the real measurements, as shown in the Table 2. As we can observe, the values do not have strong deviation even though the model is computed with one single frame. Several frames capturing the same staircase could be potentially used to improve the retrieved dimensions, or even for an online update of these dimensions during traversal. However, we decided not to include these feature given the good estimations from one frame to keep the problem simple and avoid the extra computational cost. Half of the experiments were conducted with real people going up and down the stairs. The presence of obstacles partially occluding the view of the staircase does not adversely affect the quality of the model and we get similar results in terms of average measurements. Our experiment from Table 2 show slighly better standard deviation in the presence of occluding obstacles. This unexpected result is due to the variability of the images in the set and not because of the method itself. Some pictures of the experiments with people climbing up/down the staircase can be seen in Fig. 17.

The computation time was also tested to analyse the performance of the system in the current state of development in the computer described above. The four main parts in which the algorithm is divided are the Visual Odometry estimation (VO), the Segmentation and Classification (SC), the Stair Detection and Modelling (SDM) and the Stair Traversal (ST); the yellow blocks from Fig. 2. The VO and SC part run every iteration. When the SC raises the existence of a set of step candidates, the SDM part is executed. The ST part runs when the user is climbing the stairs, instead of SDM (they both never run in the same iteration).

The VO stage has a detailed explanation about the computation time depending on the configuration used in [7]. In this work we have removed the dense volumetric mapping, and we manage to estimate the visual odometry in an average time of

Fig. 14. (a) Four examples of images without staircases including deceiving parallel planes which are detected and then discarded by our algorithm due to impossible stair models. (b) Two examples of complex stairs where the modelling fails because of atypical shapes (left) or non-Manhattan directions of the edges with respect to the wall (right). Best viewed in colour.

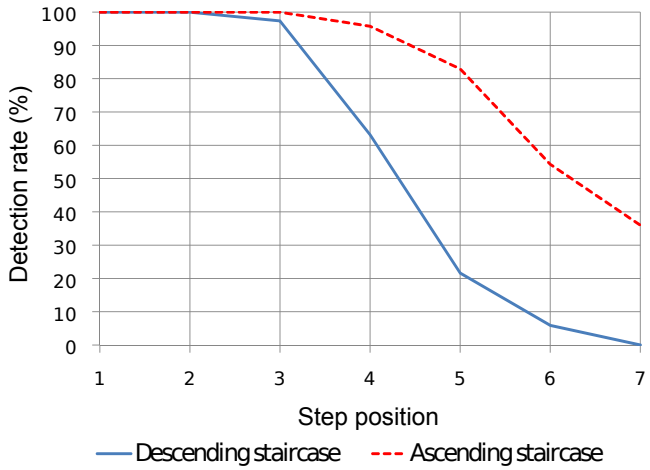


Fig. 15. Step detection rate with the step position in the staircase.

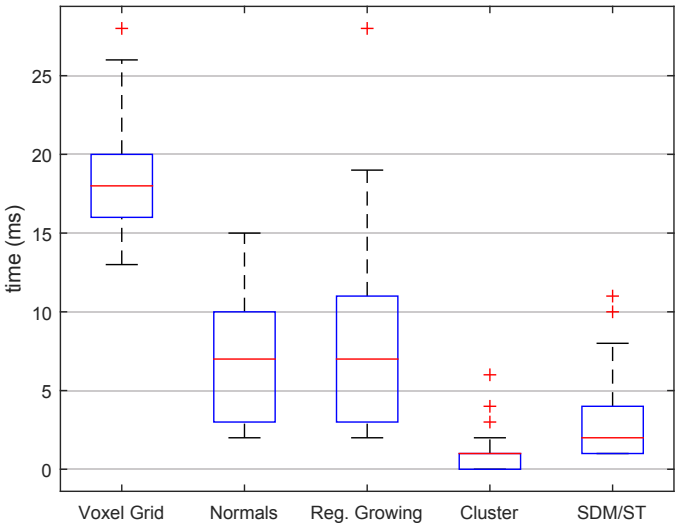


Fig. 16. Box plot of the time consumption of the most relevant parts of our algorithm. For each column, the box is limited by the 25th and 75th quartile with the median inside. The whiskers reach the most extreme points and the outliers are marked with a cross.

15.397ms per iteration. Unlike the VO part, SC stage's runtime is scene dependent. It takes longer to compute when the scene is bigger or more complex, as both the normal estimation and region-growing algorithms need to iterate in a larger amount of points. The stair-related part is also scene dependent, as the SDM stage is only executed when there is stair in the image and the ST when the user is traversing. To cover all situations and provide results from this part, we have performed an experiment where the user approaches the staircase from far away (no visible stair) until he reaches the first step (during half of the time) and then move upstairs it until the other end of the stair is in sight (the other half of the time).

The following numeric results come from using a voxel grid of size 4cm, which provides a good compromise between accuracy and speed. Excluding the VO part from the computation, each iteration takes a median time of 39ms (25Hz), with 46ms during the first half and 29ms during the second half; and a maximum of 77ms. The second half takes less time due to the simpler scene (points are close to the camera and among themselves so the voxel grid returns less points to deal with) and to the execution of the ST instead of the SDM algorithm (ST takes a median time of 1ms whereas SDM takes 7ms). In Fig. 16 there is a box plot showing the median and quartiles of the different stages. The segmentation part appears broken down in the four biggest time consumers (voxel grid, normal estimation, region-growing and cluster extraction) discarding stages which takes less than 1ms. Voxel grid is the slowest part but it presents lesser variability than the others, where it is more noticeable. Cluster extraction is usually small as most points of the scene belong to planes already segmented in the region-growing stage. The stair-related times are only represented when a stair is visible (otherwise time is zero).

In general, this timing should be considered fast enough for indoor navigation assuming walking speeds around $1 - 1.5$m/s. Modern laptops or even smartphones or tablets should have

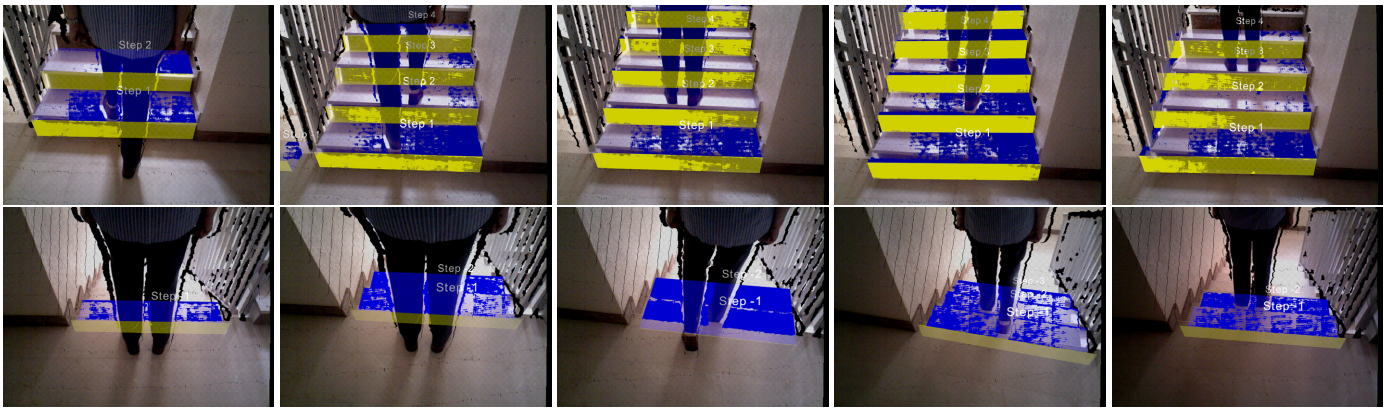
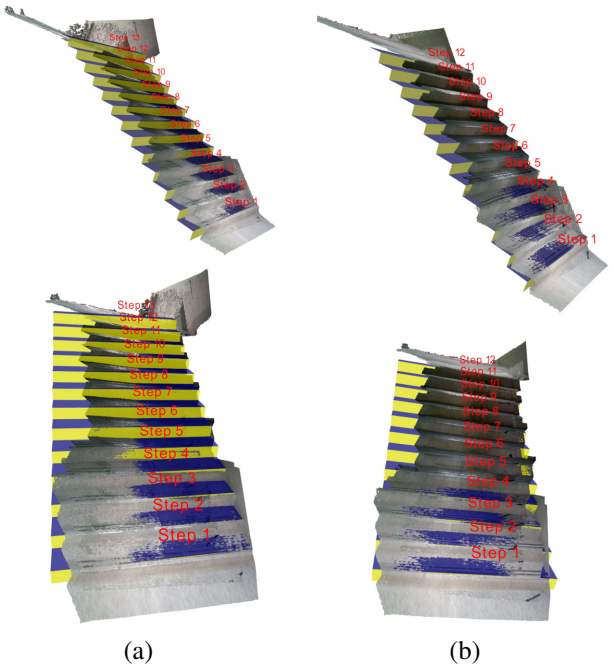


Fig. 17. Example of a person partially blocking the view of the staircase during ascent or descent. Best viewed in colour.

nowadays enough processing power to run this system. In case it were necessary to improve the performance rates, some pre-processing parts could be optimized by using more efficient algorithms (we used some standard implementations included in widely common open source libraries) or by running them in GPU (at this point only the visual odometry takes advantage of the GPU), but the optimization of the system has not been subject of our research at this point.

For the stair traversal algorithm, we have tested several video sequences with and without the drift correction implemented. A qualitative visual analysis consisted in looking at the 3D mapping from some intermediate key frames compared to the stair template generated in the modelling. An example is shown in Fig. 18. In (a) we can see that, although the first step matches the model perfectly in both cases, as the camera rises up the stair the 3D map diverges from the template, due to the drift. It is more prominent in the last few steps, where the drift is big enough to cause the detection of one step more than the staircase actually has. With the drift correction the 3D map succeed in matching the stair template, recovering the correct number of steps of the staircase.

A more quantitative analysis leads to the graphs in Fig. 19, Fig. 20 and Fig. 21. In Fig. 19 there is the $y_s - z_s$ trajectory of the user in the first seven steps and the corresponding stair profile. The trajectories have a wave form during the traversal, where each peak corresponds with the instant the user reaches the height to walk on the following step. With the drift correction every peak is consistently paired with the edge of every step. However, without the correction the peak is reached increasingly farther, being more than a half step in the seventh step. In Fig. 20 a closer look to this gap can be observed in both cases compared to the ground truth. In the seventh step the trajectory without odometry correction reaches the step with 16.6cm of gap, whereas with the correction there is almost no deviation with respect to the expected trajectory. When the stair is large enough that gap can provoke the misdetection of more steps than the stair has. For instance, in Fig. 18 the algorithm detects an extra 13th step non existent in the real staircase. In Fig. 21 it is displayed the three rotation angles of the camera reference frame during the traversal with and without the correction. As it occurs with the translation, the orientation drift also increases through iterations, but it is corrected with our approach.



(a) (b)

Fig. 18. Visual comparison of the 3D map composed by key frames obtained during the traversal with the stair model without (a) and with the correction (b).

7. Conclusion

In this paper we have presented a stair-aimed perception module of a wearable personal assistant oriented to visually impaired people, although it may have applications in other fields such as robotics, specially in the case of humanoids. For this we have developed an algorithm covering operations such as the detection of stairs, the retrieval of the location and measurements of the stairs and the continuous self-localization during the traversal. Our algorirthm includes a visual odometry module which provides location awareness to the system, enabling the possibility of going back to places not currently visible and

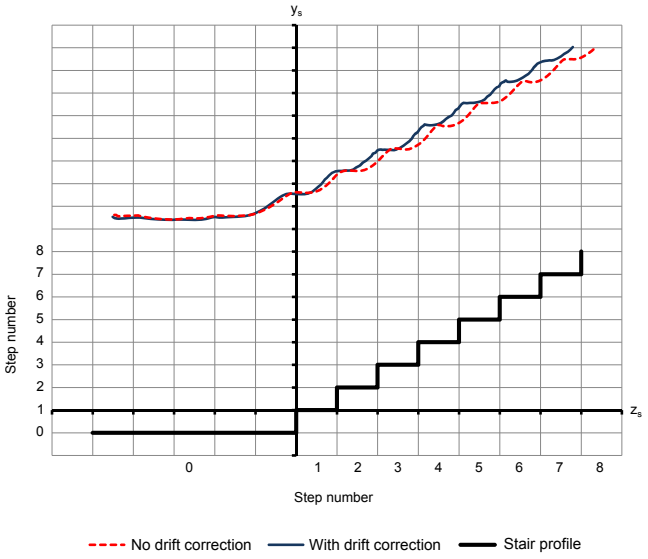


Fig. 19. Trajectory of the person during the climb with and without drift correction.

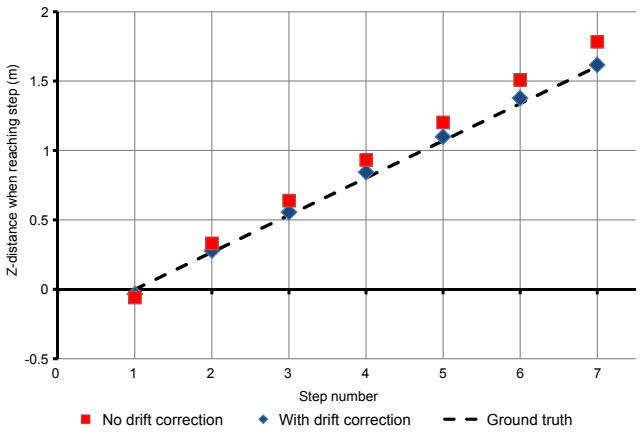


Fig. 20. Distance in z_S the moment the user reaches the first seven steps.

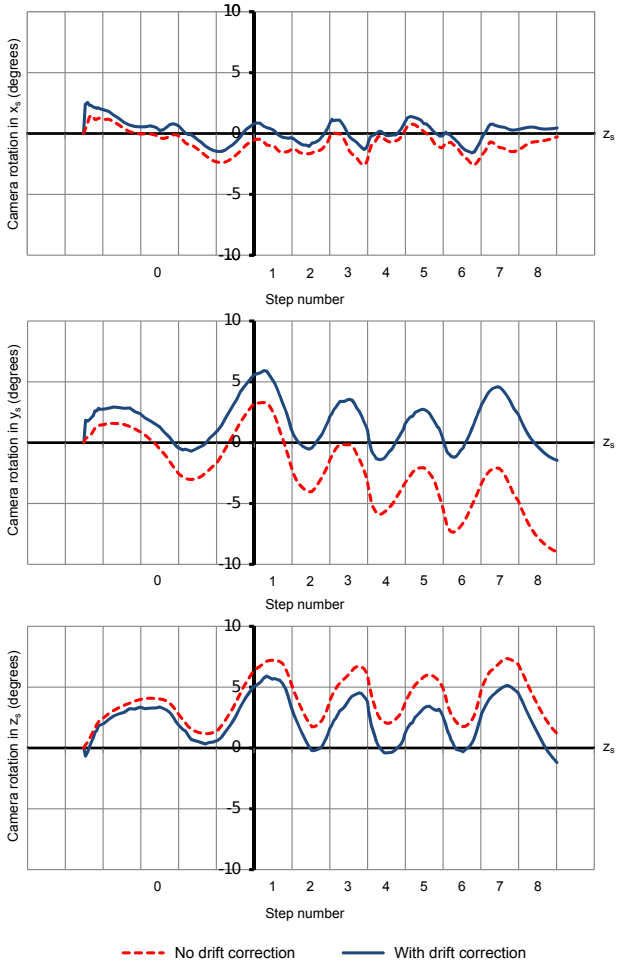


Fig. 21. Rotation angles of the person with respect to the stair reference frame in the three stair directions. Best viewed in colour.

the traversal of staircases. Moreover, we use the information of the camera to correct the drift that the visual odometry have. The experiments prove that the model quality and the computing time are good enough to be used in real-time. The algorithm overcomes some limitations existing in related works, such as the possibility of single step detection or full modelling with partial occlusions caused mainly by other people traversing the staircases.

In the near future we would like to extend the possibilities that a RGB-D sensor can bring to stair detection by combining the depth information with colour images. RGB data would help improving the model, counting the steps to extend the staircase model, detecting possible staircases from farther distances where depth measurements are not reliable or when the sun rays affect negatively the depth sensing. In addition, besides simply extracting planes and clusters and consider them obstacles, performing more extended reasoning about the scene and the objects in it is also a line of research we want to explore in future works. Another interesting way to continue the work would be to extend the experiments to outdoor environments, trying other currently suitable sensors such as stereo cameras.

Acknowledgments

This work was supported by Ministerio de Economía y Competitividad and European Union under FPI grant BES-2013-065834 and projects DPI2014-61792-EXP and DPI2015-65962-R.

References

[1] R. Manduchi, S. Kurniawan, Mobility-related accidents experienced by people with visual impairment, Research and Practice in Visual Impairment and Blindness 4 (2) (2011) 44–54.

[2] J. A. Hesch, S. I. Roumeliotis, Design and analysis of a portable indoor localization aid for the visually impaired, The International Journal of Robotics Research 29 (11) (2010) 1400–1415.

[3] X. Qian, C. Ye, NCC-RANSAC: A fast plane extraction method for navigating a smart cane for the visually impaired, in: IEEE International Conference on Automation Science and Engineering (CASE), 2013, pp. 261–267.

[4] W. W. Mayol-Cuevas, B. J. Tordoff, D. W. Murray, On the choice and placement of wearable vision sensors, IEEE Transactions on Systems,

Man and Cybernetics, Part A: Systems and Humans 39 (2) (2009) 414–425.

[5] A. Aladren, G. Lopez-Nicolas, L. Puig, J.J. Guerrero, Navigation assistance for the visually impaired using RGB-D sensor with range expansion, IEEE Systems Journal, Special Issue on Robotics & Automation for Human Health PP (99) (2014) 1–11.

[6] J.J. Guerrero, A. Perez-Yus, D. Gutierrez-Gomez, A. Rituerto, G. Lopez-Nicolas, Human navigation assistance with a RGB-D sensor, in: ACTAS V Congreso Internacional de Turismo para Todos: VI Congreso Internacional de Diseño, Redes de Investigacion y Tecnologia para todos DRT4ALL, 2015, pp. 285–312.

[7] D. Gutiérrez-Gómez, W. Mayol-Cuevas, J. J. Guerrero, Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry, in: IEEE International Conference on Robotics and Automation (ICRA), 2015.

[8] R. B. Rusu, S. Cousins, 3D is here: Point cloud library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 1–4.

[9] A. Pérez-Yus, G. López-Nicolás, J.J. Guerrero, Detection and Modelling of Staircases Using a Wearable Depth Sensor, in: ECCV 2014 Workshops, Part III, Vol. LNCS 8927, Springer, 2015, pp. 449–463.

[10] J. M. Coughlan, A. L. Yuille, Manhattan world: Compass direction from a single image by bayesian inference, in: IEEE International Conference on Computer Vision, Vol. 2, 1999, pp. 941–947.

[11] V. Pradeep, G. Medioni, J. Weiland, Robot vision for the visually impaired, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010, pp. 15–22.

[12] Y. H. Lee, G. Medioni, A RGB-D camera based navigation for the visually impaired, in: RSS 2011 RGBD: Advanced Reasoning with Depth Camera Workshop, pp. 1–6.

[13] S. Mann, J. Huang, R. Janzen, R. Lo, V. Rampersad, A. Chen, T. Doha, Blind navigation with a wearable range camera and vibrotactile helmet, in: ACM International Conference on Multimedia, 2011, pp. 1325–1328.

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

[15] S. Se, M. Brady, Vision-based detection of staircases, in: Asian Conference on Computer Vision (ACCV), Vol. 1, 2000, pp. 535–540.

[16] D. C. Hernandez, K.-H. Jo, Outdoor stairway segmentation using vertical vanishing point and directional filter, in: IEEE International Forum on Strategic Technology (IFOST), 2010, pp. 82–86.

[17] J. A. Hesch, G. L. Mariottini, S. I. Roumeliotis, Descending-stair detection, approach, and traversal with an autonomous tracked vehicle, in: IEEE International Conference on Intelligent Robots and Systems (IROS), 2010, pp. 5525–5531.

[18] K. Ishiwata, M. Sekiguchi, M. Fuchida, A. Nakamura, Basic study on step detection system for the visually impaired, in: IEEE International Conference on Mechatronics and Automation (ICMA), 2013.

[19] X. Lu, R. Manduchi, Detection and localization of curbs and stairways using stereo vision, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 4, 2005, p. 4648.

[20] V. Pradeep, G. Medioni, J. Weiland, et al., Piecewise planar modeling for step detection using stereo vision, in: Workshop on Computer Vision Applications for the Visually Impaired, 2008.

[21] J.-S. Gutmann, M. Fukuchi, M. Fujita, Stair climbing for humanoid robots using stereo vision, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 2, 2004, pp. 1407–1413.

[22] V. Filipe, F. Fernandes, H. Fernandes, A. Sousa, H. Paredes, J. Barroso, Blind navigation support system based on Microsoft Kinect, Procedia Computer Science 14 (2012) 94–101.

[23] S. Wang, Y. Tian, Detecting stairs and pedestrian crosswalks for the blind by RGBD camera, in: International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), 2012, pp. 732–739.

[24] T. J. J. Tang, W. L. D. Lui, W. H. Li, Plane-based detection of staircases using inverse depth, Australasian Conference on Robotics and Automation (ACRA), 2012.

[25] M. Vlaminck, L. Jovanov, P. Van Hese, B. Goossens, W. Philips, A. Pizurica, Obstacle detection for pedestrians with a visual impairment based on 3D imaging, in: IEEE International Conference on 3D Imaging (IC3D), 2013.

[26] J. A. Delmerico, D. Baran, P. David, J. Ryde, J. J. Corso, Ascending stairway modeling from dense depth imagery for traversability analysis, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 2283–2290.

[27] S. Oßwald, J.-S. Gutmann, A. Hornung, M. Bennewitz, From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids, in: 11th IEEE-RAS International Conference on Humanoid Robots, 2011, pp. 93–98.

[28] S. Oßwald, A. Gorog, A. Hornung, M. Bennewitz, Autonomous climbing of spiral staircases with humanoids, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011, pp. 4844–4849.

[29] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, R. B. Fisher, An experimental comparison of range image segmentation algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (7) (1996) 673–689.

[30] T. Rabbani, F. van den Heuvel, G. Vosselmann, Segmentation of point clouds using smoothness constraint, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (5) (2006) 248–253.

[31] R. Triebel, J. Shin, R. Siegwart, Segmentation and unsupervised part-based discovery of repetitive objects, in: Robotics: Science and Systems, Vol. 2, 2010.

[32] A. Karpathy, S. Miller, L. Fei-Fei, Object discovery in 3d scenes via shape analysis, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 2088–2095.

[33] C. Raposo, M. Lourenço, M. Antunes, J. P. Barreto, Plane-based odometry using an RGB-D camera, in: British Machine Vision Conference, 2013.

[34] Y. Taguchi, Y.-D. Jian, S. Ramalingam, C. Feng, Point-plane SLAM for hand-held 3D sensors, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 5182–5189.

[35] P. W. Holland, R. E. Welsch, Robust Regression Using Iteratively Reweighted Least-Squares, Communications in Statistics: Theory and Methods A6 (1977) 813–827.

[36] C. Kerl, J. Sturm, D. Cremers, Robust odometry estimation for rgb-d cameras, in: IEEE International Conference on Robotics and Automation (ICRA), 2013.

[37] K. Konolige, P. Mihelich, Technical description of kinect calibration, http://wiki.ros.org/kinect_calibration/technical (2010, [Online; accessed 14-April-2015]).