2017 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 26-28, Coimbra, Portugal

3D Point Cloud Downsampling for 2D Indoor Scene Modelling in Mobile Robotics

Luís Garrote, José Rosa, João Paulo, Cristiano Premebida, Paulo Peixoto and Urbano J. Nunes

***Abstract*—Sensory perception and environment modelling are important for autonomous navigation in mobile robotics. 2D discrete grid representations such as the classic 2D occupancy grid maps are a widely used technique in scene representation because of the inherent simplicity and compact representation. In recent years, many 2.5D and 3D grid-based methods have been proposed however, as for the 2D case, a compromise between keeping a low computational bound and reliable sensor interpretation must be kept in order to perform real-world tasks. Assuming the input data in the form of a 3D point-cloud, in this paper we propose a 2D scene modelling approach which converts the 3D data to a 2.5D representation and then to a 2D grid map in an efficient and meaningful manner. The proposed approach incorporates a new rapidly exploring random tree inspired ground-plane detection (RRT-GPD), and an inverse sensor model (ISM) to correctly map 3D to 2.5D and then to 2D grid cells. Experiments were conducted in indoor scenarios with a robotic walker platform equipped with a Microsoft's Kinect One and a LeddarTech's Leddar IS16 sensor. Reported results show an improvement on the representation of non-trivial obstacles (stairs, floor outlets) over a ROS package solution, when applied to a 3D point cloud input.**

I. Introduction

The development of sensor-based perception systems is one of the main focuses of research in robotics. Considering the increasing introduction of robots in human populated environments, there is an heightened concern to ensure the safety of all agents who may interact with the robot. In particular, when considering the case where the robot is a physical support aid, such as a robotic walker [1], the safety of all agents is crucial. The robot must ensure that it follows all the user's instructions accordingly, while obeying safety rules taking into account the knowledge of the surrounding environment.

Despite the recent advances in the area, the development of efficient and robust maps of the robot's surroundings is still an open and challenging problem, mostly due to the uncertainty of the sensory data and unpredictability of the environments. One type of representation, widely used in mobile robotics, is based on occupancy grid maps, which makes it possible to determine, based on the probabilistic sensor models, the existence of free and occupied space in the surrounding environment. In this work, we review the 2D occupancy grid-map framework, and propose a new approach to the problem of mapping collision-able non-trivial obstacles from 3D point clouds into a 2.5D and then to a 2D environment representation for autonomous navigation. Such obstacles may trigger disastrous consequences for the user if not detected properly (e.g., stairs, gutters or floor outlets). The main contributions are: 1) an inverse sensor model (ISM) for 2.5D to 2D mapping, from 3D data input, incorporating the ground-plane and the concept of voxel density; 2) a new rapidly exploring random tree inspired ground-plane detection (RRT-GPD) algorithm; 3) real time execution ($\approx$ 30 frames per second).

An overview of the related work in environment representation is provided in Section II. In Section III, the proposed method is explained. Experimental results are reported and discussed in Section IV, and final conclusions are drawn in Section V.

II. Related Work

Environment representation and modelling have been the focus of many research work in mobile robotics [2], [3], with many representations being proposed in the last 30 years. We can categorize scene representations in 3 main classes: direct, topological or grid-based. In this paper, a grid-map based approach is proposed in which the environment is subdivided into a set of smaller units. Based on the unit structure, the grid-based representation can be further categorized into 2D cells, 2.5D cells (or voxels) and 3D voxels. In the literature, the 2D grid-map is a well-known representation, providing a probabilistic framework [4], with fast and constant-time access while being usually only applicable in planar environments. In this type of representation each cell contains the probability of occupancy, where values near zero correspond to free cells, values near one correspond to occupied cells and a middle value corresponds to an unexplored region. These probabilistic values depend on the successive readings from sensors, increasing or decreasing according to the sensors' readings and sensors' observation models. On the other hand, representations such as 2.5D grid-maps provide a framework to represent elevations and irregular terrain [5] instead of just occupancy, having properties similar to 2D maps but with information regarding the measured height of the environment for each cell, and are computationally lighter models than

Authors are with Institute of Systems and Robotics, Electrical and Computer Engineering Department, University of Coimbra, Coimbra, Portugal.

This work was supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD grants SFRH/BD/88672/2012 and SFRH/BD/88459/2012 with funds from QREN − POPH and the European Social Fund from the European Union. It was also partially supported by the FCT projects UID048 and AMS-HM12: RECI/EEI-AUT/0181/2012.

Email: {garrote, joserosa, jpaulo, cpremebida, peixoto, urbano}@isr.uc.pt

3D grid-maps. However, a 2.5D representation can not fully represent vertical overlapping. Solutions to this problem have been proposed in the form of multi-layer maps. These grid-map approaches are easy to build, represent, maintain and facilitate computation of shortest paths but, on the other hand, they present downsides such as being memory consuming for large environments and presenting poor interfaces with most classic planning algorithms. More recently, 3D grid maps, and in particular solutions such as octomaps [3], provide a reliable 3D representation at the expense of variable access time, increased computational complexity and increased planning complexity for ground robots. In spite of the constant evolution, a 2D grid map is still an usefull and somehow efficient representation for indoor and outdoor scenarios but, due to its own limitations, fails to incorporate vertical elements that are in the robot's pathway, even if they can be trivially spotted by humans. A solution to this problem could be the use of a ground reference and/or a mapping strategy in order to distinguish between obstacles and drivable space. The ground reference could be provided by elevation thresholds or by a ground-plane detection algorithm. In [6], [7] 2D grid maps are built from 3D point clouds and stereo images where a fusion model incorporates vertical penalization based on elevation thresholds. In [8] a 3D voxel representation is built from stereo images and converted to 2D grid maps using a voxel observation model. Some solutions available in the Robot Operating System (ROS) provide a medium of converting 3D point clouds into 2D laser scans using elevation thresholds and can be trivially converted into 2D grid maps. In [9], 2D grid maps are computed from stereo sequences using an intermediary 2.5D representation to generate elevation thresholds. Solutions for multiple plane detection [10] have been proposed for indoor scenarios using depth images, that can be adapted for ground-plane detection. For ground-plane detection in point clouds, region growing methods [11] can provide good results, but their computational cost becomes prohibitive in dense point clouds.

In this paper, we propose an approach for 2D scene representation and modelling, from a 3D point-cloud input, to allow autonomous navigation in real-world (indoor) conditions. The approach takes advantage of a 2.5D representation to detect non-trivial obstacles. Moreover, a ground-plane detection approach, using an RRT algorithm, is also proposed.

III. PROPOSED METHOD

In this section we detail the proposed method and its functional modules. The key motivation here is to develop a reliable and computationally efficient method for mapping 3D point clouds into a 2D grid map and consequently, the correct mapping of collision-able non-trivial obstacles. Non-trivial obstacles are defined, in the context of this work, as obstacles that are in a given robot's pathway but are neither trivially detected nor efficiently mapped by common 2D mapping approaches (e.g., stairs, small boxes or electric wiring). The proposed method depends on a 2.5D representation, followed by the ground-plane detection and then the final (enhanced) 2D grid map is obtained using a 2.5-to-2D ISM approach.

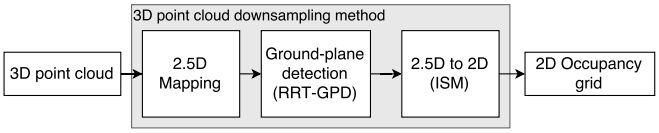

Fig. 1. Depiction of the proposed method including the modules: 2.5D mapping; ground-plane detection; 2.5D to 2D conversion.

A general overview of the proposed method is presented in Fig. 1 which includes 2.5D mapping, ground-plane detection and 2.5D map to 2D occupancy grid map conversion.

A. 2.5D Mapping

The first module builds on the construction of a 2.5D environment representation from a 3D point cloud. The 2.5D representation discards the concept of occupancy and provides an elevation measure for each cell. The proposed 2.5D grid is composed by m_x x m_y cells with constant resolution d_r. Each cell $c_{2.5D}$ is addressed by i and j indexes and has the elements $c_{2.5D} \leftarrow \{z^-,\ z^+, N_z\}$ where z^- is the minimum elevation value, z^+ the maximum elevation value and N_z the number of samples that contribute with information to the cell. Given a 3D point cloud composed by a set of 3D Cartesian points ($\mathbf{p}_k = (x_k, y_k, z_k)^T$, $k = 1, 2, ..., n$) the values of each cell are obtained by projecting the x and y components on the grid cell with consequent update of the maximum and minimum elevation (z^+ and z^-) and the number of samples (N_z). The projection consists on the conversion of the x and y components of $\mathbf{p}_k$ into grid indexes i, j with $i \leq m_x$, $j \leq m_y$, $m_x, m_y > 0$. The 3D to 2.5D maps conversion process is illustrated by an example in Fig. 2.

B. Rapidly exploring random tree based ground-plane detection (RRT-GPD)

One problem when converting an elevation map into an occupancy grid is the definition of free and occupied 2.5D cells. Given a 2.5D grid, the definition of what is an obstacle or what is navigable space is difficulty since there is no reference to the location of the ground plane. For a given frame, we can assume that a robot moves locally in a 2D plane, and based on this constraint we propose a new approach to the problem of ground-plane detection. This new approach is inspired by the rapidly-exploring random tree (RRT) algorithm, which has been widely applied in motion planning [12], and region growing [11], in the sense that it implements a rapid flood-like expansion towards similar normals. The inputs of the proposed RRT-GPD method include, the robot's pose, the plane's normal threshold, elevation threshold, the 2.5D map, the maximum number of iterations to find a suitable solution, and a node expansion distance. The RRT-GPD (detailed in **Algorithm 1**) starts by finding a valid seed to form the root of the RRT (**nearestValidSeed**). The use of an invalid seed may create a ill formed tree. This is due to the sparse nature of the 2.5D representation (see Fig. 2) meaning that not all cells of the

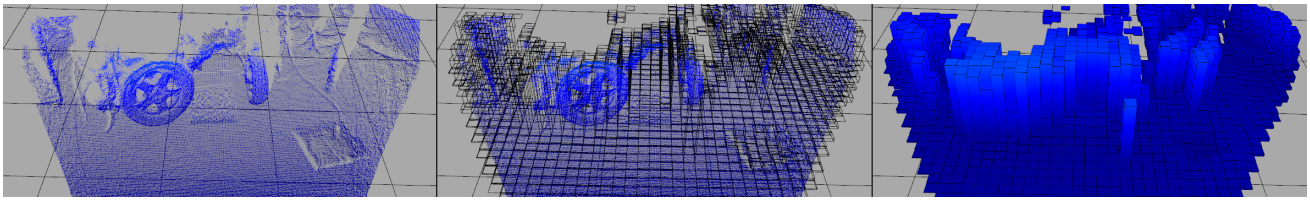

Fig. 2. Representation of the conversion from 3D point clouds to 2.5D environment representation. From left to right, the input 3D point cloud (given by a Microsoft's Kinect One), the projection step where each point in the point cloud is projected into the corresponding cell, and the final result where each cell resembles a voxel defined by the maximum and minimum elements projected onto the cell.

Algorithm 1: Rapidly exploring random tree based ground-plane detection (RRT-GPD) algorithm.

Input: Robot pose ($\mathbf{p}_{xyz}$), Plane normal threshold (N_{th}) , Elevation threshold (E_{th}), 2.5D Map ($M_{2.5D}$) , Maximum number of iterations (K) , Node expansion distance (d_n)

Initialization:
$\mathbf{p}_0 \leftarrow$ nearestValidSeed($\mathbf{p}_{xyz}$, $M_{2.5D}$);
$G \leftarrow$ initializeTree($\mathbf{p}_0$);
for *k=1 to K* **do**
  $x_{rand} \leftarrow$ sampleRandomDirection();
  $x_{near} \leftarrow$ nearestNode(G,x_{rand});
  $\theta \leftarrow$ angleBetween(x_{near},x_{rand});
  $x_{expansion}(x) \leftarrow x_{near}(x) + d_n \cos(\theta)$;
  $x_{expansion}(y) \leftarrow x_{near}(y) + d_n \sin(\theta)$;
  $x_{expansion}(z) \leftarrow$ elevation($M_{2.5D}$, $x_{expansion}$) ;
  $P \leftarrow$ interpolation($M_{2.5D}$, $x_{expansion}$);
  $plane_{local} \leftarrow$ fitPlane(P);
  if $|$ *normal*($plane_{local}$) $\cdot$ *normal*(x_{near}) $| \leq N_{th}$ **then**
    if $|$ $plane_{local}(z)$ - $x_{near}(z)$ $| \leq E_{th}$ **then**
      $G \leftarrow G \bigcup \{$ x_{near},$x_{expansion}$, normal($plane_{local}$) $\}$;
      updateSearchSpace($x_{expansion}$);
  else
    $x_{neighbors} \leftarrow$ nearestNodes(G,$x_{expansion}$,d_n);
    foreach *node* ***in*** $x_{neighbors}$ **do**
      if $|$ *normal*($plane_{local}$) $\cdot$ *normal*(*node*) $| \leq N_{th}$ **then**
        if $|$ $plane_{local}(z)$ - *node*(z) $| \leq E_{th}$ **then**
          $G \leftarrow G \bigcup \{$ *node*,$x_{expansion}$, normal($plane_{local}$) $\}$;
          updateSearchSpace($x_{expansion}$);
$g_{plane} \leftarrow RANSAC(Points(G))$
Output: g_{plane}

provided 2.5D map, $M_{2.5D}$, will contain information. Besides, a seed far from a valid cell may be unable to expand, a problem which can be solved using a variable node expansion distance. In the process of finding a valid seed, the robot's pose provides an initial guess, and since no assumptions are made on the sensors' configuration every direction is explored to find a valid candidate. Although it is important to find a valid candidate to initialize exploration, the final solution may discard the initial seed point (i.e., it is not guaranteed that the final solution will contain the initial point). Those considerations are based on the assumption that the 2.5D map is computed with the sensor referenced on the robot's frame and that a transformation between the robot's frame and the world frame is available. For local maps where the mobile robot frame (or sensor frame) is at the map's center ($\mathbf{p}_{xyz} = (0,0,0)^T$) the same constraints apply. The concept of valid seed implies in this particular case, a valid local plane computed using **fitPlane**. After finding a valid seed and a correspondent local plane, a new tree is generated by **initialiseTree**, where the seed, which corresponds to the center of the plane, and the plane normal define the base unit (or node) of the RRT tree. If a valid node is created, the algorithm enters an iterative process to expand the RRT until it reaches K iterations. For each iteration the first step involves sampling in a search space window (**sampleRandomDirection**) resulting in a point x_{rand}. For a given sample node x_{rand}, the nearest node x_{near} already present in the tree is retrieved (**nearestNode**) and a new node $x_{expansion}$ is created based on the direction between x_{near} and x_{rand} (provided that they do not overlap). The direction is given by (**angleBetween**),

$$\theta = \arctan\left(\frac{x_{rand}(y) - x_{near}(y)}{x_{rand}(x) - x_{near}(x)}\right) \tag{1}$$

and the (x, y) component is given by,

$$x_{expansion}(x,y) = \begin{cases} x_{near}(x) + d_n \cos(\theta) \\ x_{near}(y) + d_n \sin(\theta) \end{cases} \tag{2}$$

where d_n denotes the node expansion distance. For the new candidate node $x_{expansion}$, the z component is given by the procedure **elevation**. As stated earlier, given the sparse nature of the 2.5D grid, not every $x_{expansion}$ will correspond to a valid cell when projected on the 2.5D map. To provide a valid z component we apply a spatial interpolation method on the node neighborhood (a region of interest with dimensions (s_x,s_y) centered in $x_{expansion}$) and if an invalid z value is retrieved (no valid neighbors) the current iteration is ended. With a valid $x_{expansion}$, and applying a similar spatial interpolation method, a set of neighborhood points P is extracted (**interpolation**) where empty points missing the z component are interpolated and discarded if the neighborhood does not contain valid elements. If the neighborhood P contains at least three non-collinear points, a least squares regression is performed to find the best planar fit to the points (**fitPlane**) of the form $ax + by + cz + d = 0$. Given a valid plane $plane_{local}$, the inner product of the normal, stored in x_{near},

and the planes' normal (**normal**($plane_{local}$)) is computed and if its value is less than a given threshold (N_{th}) the plane is considered to be similar in orientation given the node x_{near}. In order to analyze whether the new plane can connect or not with the nearest node, the variation in height is also checked (using the elevation threshold (E_{th})). If the normals are not similar, the nearest nodes within a radius d_n from $x_{expansion}$ are retrieved (**nearestNodes**) and the same thresholds are applied to validate further node connections. If a pair plane-node is considered valid (i.e., is valid for each threshold) a new node is added to the tree, containing the parent connection (x_{near}), the new center point from the plane midpoint ($x_{expansion}$) and the correspondent plane normal (**normal**($plane_{local}$)). The last step, after adding a node to the tree, is to update the search space window (**updateSearchSpace**). The search space window starts centered on the seed node but with each added node and each iteration the search window grows and shifts towards the average value (geometric center) of the explored nodes, slightly biasing the search process to areas with similar properties where the expansion advances more rapidly, but without leaving out unexplored areas. After K iterations, the center points of each node on the RRT tree are extracted (a Kd-tree is used at the algorithm's core to store each discovered node) and a new plane fitting is performed. In this final step, we apply the random sample consensus (RANSAC) algorithm [13] due to its robust estimation even in the presence of outliers. The RRT-GPD algorithm is summarized in the pseudocode of **Algorithm 1**.

C. 2.5D to 2D conversion

The conversion from 2.5D to 2D follows the same principles applied in the integration of sensor readings in occupancy grid mapping [4], [14]. In this case, our observations are the elevation voxels present in the 2.5D grid map, turning the representation into a virtual sensor. The probability that a cell c is occupied given the observations $z_{1:t}$ is given in log odds by:

$$l(c|z_{1:t}) = \log \frac{p(c|z_t)}{1-p(c|z_t)} - \underbrace{\log \frac{p(c)}{1-p(c)}}_{=0,\text{ if } p(c)\,=\,0.5} + \log \frac{p(c|z_{1:t-1})}{1-p(c|z_{1:t-1})} \tag{3}$$

with $p(c)$ the prior probability, $p(c|z_{1:t-1})$ the previous estimate and $p(c|z_t)$ denotes the probability that cell c be occupied given the measurement z. The log odds representation is used here due to its numerical stability. To solve the 2.5D to 2D conversion problem we propose an ISM that converts an observation in the form of an elevation voxel, c_v, to the probability that, given the actual observation, the cell of the 2D grid is occupied $p(c|z_t)$. Each elevation voxel can be defined as being in a valid state if it contains more than one measurement ($N_z \geq 1$). In order to determine the influence of each voxel we rely on the concept of voxel density explored in [15]. An elevation voxel c_v in the 2.5D map (see Fig. 2) occupies the volume given by $V_{voxel} = hA$, where h is the height value of the voxel ($h = \Delta z = z^+ - z^-$) and A the base area of the voxel (i.e., based on the cell resolution). The voxel density is given by $\rho_{voxel} = \frac{m}{V_{voxel}}$, where m is the voxel mass. The mass of a voxel in this context can be defined as the amount of data the voxel contains and can be extrapolated using the number of samples N_z of an elevation voxel. To represent a normalized voxel density, the following sigmoidal function is proposed,

$$\rho_{voxel}(c_v) = \frac{K_m}{1 + e^{-(\frac{K_n(c_v(N_z) - d_{min})}{V_{voxel}})}} \tag{4}$$

where K_m denotes an amplitude gain, K_n a sample normalization factor, and d_{min} the minimum number of points. The voxel is composed by 3 explicit parameters and an implicit one related to the distance $|c_v|$ defined by the voxel position in relation to the base frame (e.g., sensor frame, robot frame or local frame).

1) Inverse sensor model (2.5D to 2D): The proposed ISM approach takes into account the voxel distance to the robot base frame, decreasing the elevation voxel occupancy probability as voxels move away from the base frame. Knowing a valid ground-plane allows for the definition of "free" or "occupied" values in the sense that a specific cell contains a high or low probability of being occupied, for instance, if a elevation voxel is near the detected ground-plane it may be considered as part of the ground and thus contribute to decrease the 2D map cell's probability. On the other hand, if an elevation voxel is above the ground, it may be considered to be an obstacle and thus contributes to increase the cell's probability. The ISM is mathematically expressed by,

- If $|c| \in [\ 0,\ |c_v|\ ]$:

$$p(c|z_t) = \begin{cases} \max(\rho_{voxel}, 0.5)\ e^{-\frac{(|c|-|c_v|)^2}{2\sigma^2}} & ,if\ d > d_{pth} \\ K_g + \frac{0.5 - K_g}{1+e^{-(|c|-|c_v|)}} & ,if\ d \leq d_{pth} \end{cases} \tag{5}$$

- If $|c| \in]\ |c_v|,\ |c_v|^{max}\ ]$:

$$p(c|z_t) = \begin{cases} \max(\rho_{voxel}\ e^{-\frac{(|c|-|c_v|)^2}{2\sigma^2}}, 0.5) & ,if\ d > d_{pth} \\ 0.5 & ,if\ d \leq d_{pth} \end{cases} \tag{6}$$

where σ^2 denotes the Gaussian variance, K_g an amplitude gain with $0 \leq K_g \leq 0.5$, d_{pth} a distance threshold and d the distance between the plane g_{plane} and the voxel c_v. The distance d is computed using the maximum plane distance to the points $\mathbf{p}^+ = (x, y, z^+)^T$ and $\mathbf{p}^- = (x, y, z^-)^T$ with $d = max(\frac{|ax+by+cz^+ + d|}{\sqrt{a^2+b^2+c^2}}, \frac{|ax+by+cz^- + d|}{\sqrt{a^2+b^2+c^2}})$ where x, y represent the voxel position.

IV. EXPERIMENTAL RESULTS

Two experiments, using the mobile robot shown in Fig. 3, were carried out with the purpose of validating the proposed method. The first experiment was performed on two static indoor scenarios using a Microsoft's Kinect One sensor mounted onboard the robot and consisted on a qualitative analysis of a grid map obtained by the proposed method (without outlier removal, with outlier removal and with 2.5D grid map interpolation) and a grid-map obtained by an approach based in [6] (see Section A. Lidar sensor model) for a z-axis point

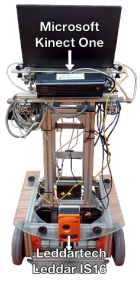

Fig. 3. The ISR-AIWALKER experimental platform.

TABLE I
LIST OF PARAMETERS

Variable	Value	Variable	Value
K	600	d_{pth}	0.03
N_{th}	0.2	K_g	0.3
E_{th}	0.05	σ^2	0.02
s_x,s_y	3	K_n	0.0005
m_x,m_y	200	d_{min}	3
d_r	0.05	K_m	0.8
d_n	0.4		

cloud layer (0.2 m to 1.5 m). The second experiment consisted on an office traversal using three approaches integrated in a local map framework: proposed method using the Kinect One sensor; an approach using the Leddar IS16; and a ROS based solution henceforth named `ROS local_map`, using the Kinect One sensor.

A. Experimental Setup

As experimental platform, we used the differential drive robotic walker ISR-AIWalker [1] shown in Fig. 3. It has two grips interfaced with Leap Motion sensors, a gait perception module aided by a Intel F200 RGB-D camera, and two sensors (a Kinect One and a Leddar IS16) for environment perception, assessment of hazardous situations and safety purposes. The Kinect One outputs a 512 x 424 point cloud and the Leddar IS16 delivers a 16-channel distance array (detection range of 0-50 meters, 45° field of view and 2.8° angular resolution). Throughout the experimental evaluation of the method, the detection and removal of outliers was an important step (as illustrated in Fig. 4). The detection and removal of outlier points belonging to the point cloud were performed at the 2.5D elevation voxel map. For a given voxel, the 3σ method was applied to the set of projected elevations and then all voxel elements were recomputed. The spatial interpolation applied on the 2.5D map, for candidate plane computation, was performed by the inverse distance weighting (IDW). The IDW algorithm was also applied to interpolate the 2.5D representation in order to obtain a more dense representation. For invalid elevation voxels in the IDW algorithm, the number of samples, N_z, was updated from the number of valid neighbor voxels. The ISR-AIWALKER perception modules run in ROS nodes, and all experiments reported in this section were carried out in the same environment with C++ implementations. The visualization of results were provided by a in-house QT/C++ application. The experiments were carried out in a mid-range laptop with Kinect and Leddar data acquisition frequencies of 10Hz; the average time per frame for the proposed method was less than 30 ms, and for the ground-plane detection less than 3 ms. Table I shows the parameter values used in this work.

B. Results and Discussion

Figure 4 shows two indoor scenarios, an uncovered floor outlet (row A) and downward stairs (row B), that were considered in the first experiment. For each scenario, qualitative results are shown in Fig. 4 where column I gives the input raw data and column II shows the 2.5D grid-map, which was generated without any preprocessing or outlier removal. As it is noticeable on the first scenario, only a reduced number of light-blue elevation voxels correspond to erroneously generated voxels due to noise. The 2.5D to 2D map conversion was performed and the ground-plane was correctly detected, as well as the floor gutter, but the correspondent outlier voxels were mapped (column III) and produced a non-traversable map (i.e. on the context of motion planning and considering only this map, the robot would be unable to move). Applying the 3σ method to the 2.5D representation (column IV) yields a cleaner 2D representation with only a small portion of noise at the end of the traversable path. The following result (column V) was obtained directly from the input point cloud for a z-axis layer (0.2 m to 1.5 m) using an approach based on the lidar sensor model proposed in [6]. Finally, in the result shown in column VI, the 3σ and IDW were applied to the 2.5D map resulting in a dense representation, but with inflated cells. By comparing the columns III, IV, and VI, for both scenarios, the most satisfactory results were achieved for the output with removal of outliers (IV), as in this case the floor outlet is correctly mapped and the portion of floor before the stairs is detected.

As regards the second experiment, with corresponding results shown in Fig. 5, three 600x600 2D local maps are presented by using: the proposed method with outlier removal, the classic sensor beam model using the Leddar scan as input and a `ROS local_map` package solution (Kinect One acquisition $\rightarrow$ point cloud to laser scan[1] $\rightarrow$ local map framework[2]). The scenario used for this experiment included three floor outlets (on the left side of the scenario), office chairs and tables (on the right side of the scenario). Results show that the local map from Kinect One, along with the proposed method, provided a reliable environment model by detecting the general scenario outline and the considered hazards (correctly mapped three floor outlets). The `ROS local_map` solution provided only an outline of the scenario since the point cloud was converted to a laser-like scan and important points relevant to the detection of near ground objects were discarded. The Leddar local map provided a rough outline representation of the scenario. A comparison between the methods applied to both sensors becomes somehow unfair, since the sensors have different fields of view and the quantity/quality of information is very distinct. However, in a safety-oriented perspective, the Kinect One, when compared to the Leddar, has a minimum depth distance of 0.5 m which can become problematic for close obstacles, while on the other hand, the Leddar not having

[1]pointcloud_to_laserscan Package – http://wiki.ros.org/pointcloud_to_laserscan
[2]local_map Package – http://wiki.ros.org/local_map

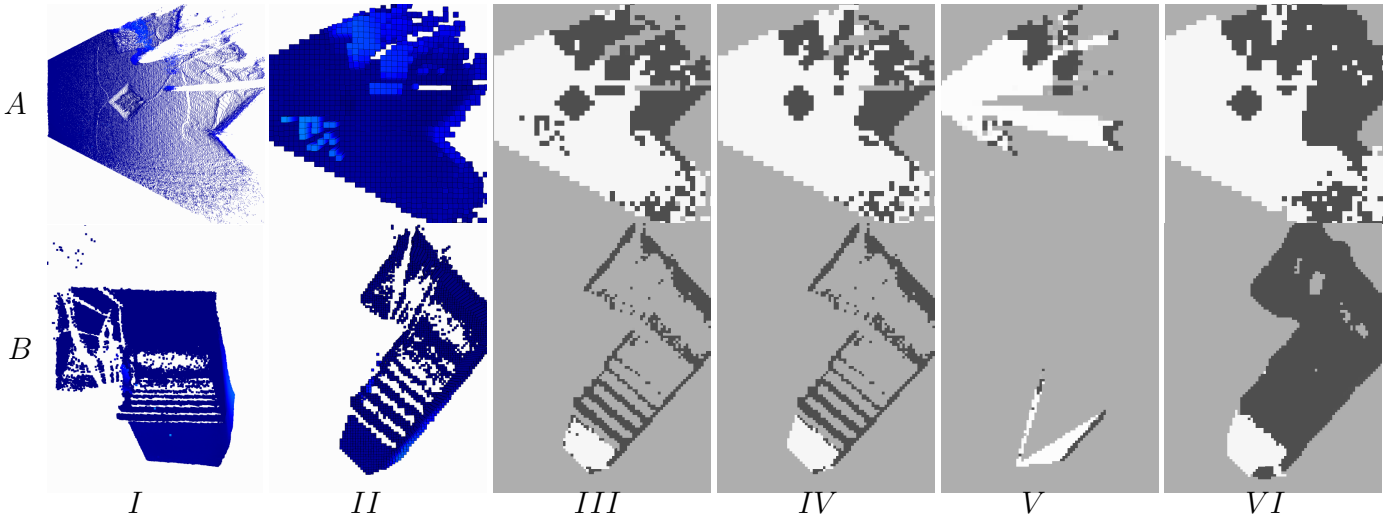

Fig. 4. Results obtained from experiment 1: first row corresponds to a scenario with an uncovered floor outlet (A), second row to a scenario with a downward stairs (B). From left to right - point cloud (I), 2.5D representation with outliers (II), 2D representation with outliers (III), 2D representation without outliers (IV), 2D representation from a point cloud z-axis layer (0.2 m to 1.5 m) using an approach based in [6] (V) and 2D representation interpolated without outliers (VI).

Fig. 5. Results obtained from the office traversal in experiment 2 (from left to right): Kinect One using the proposed method, Leddar IS16 using local map and `ROS local_map` package solution using Kinect One.

the precision of the Kinect One, provides a reliable local free space mapping.

V. Conclusion

In this paper, a novel approach that maps 3D point cloud data into an enhanced 2.5D and then to a 2D grid map, is proposed. The approach encompasses a new RRT-based ground-plane detection algorithm (RRT-GPD) and a new way to model the concept of elevation voxel density. Preliminary experimental tests give indications that the proposed method, when compared to the `ROS local_map` package solution, can provide a richer representation and a solution to map non-trivial obstacles with more realistic contour information. The proposed algorithm was tested in a robotic walker-assisted scenario, successfully validating the proposed algorithm in scenarios that are pitfalls on the handling of walkers. For future work, we plan to research on a multisensory approach for local mapping, incorporating the Kinect One and the Leddar IS16 sensor, aiming to validate the framework in terms of safe navigation in mobile robotics applications.

References

[1] J. Paulo, P. Peixoto, and U. Nunes, "A novel vision-based human-machine interface for a robotic walker framework," in *IEEE RO-MAN*, 2015.
[2] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *IEEE/RSJ IROS*, Oct 2006.
[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.
[4] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," 2005.
[5] C. Premebida, J. Sousa, L. Garrote, and U. Nunes, "Polar-grid representation and kriging-based 2.5D interpolation for urban environment modelling," in *IEEE ITSC*, 2015.
[6] J. D. Adarve, M. Perrollaz, A. Makris, and C. Laugier, "Computing occupancy grids from multiple sensors using linear opinion pools," *IEEE ICRA*, 2012.
[7] T. Rakotovao, J. Mottin, D. Puschini, and C. Laugier, "Multi-sensor fusion of occupancy grids based on integer arithmetic," *IEEE ICRA*, 2016.
[8] H. Ghazouani, M. Tagina, and R. Zapata, "Robot navigation map building using stereo vision based 3D occupancy grid," *Journal of Artificial Intelligence: Theory and Application*, vol. 1, no. 3, 2011.
[9] H. Lategahn, W. Derendarz, T. Graf, B. Kitt, and J. Effertz, "Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications," in *IEEE IV*, 2010.
[10] J. Biswas and M. Veloso, "Planar polygon extraction and merging from depth images," in *IEEE/RSJ IROS*, Oct 2012.
[11] T. Rabbani, F. A. van den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," in *IEVM06*, 2006.
[12] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep. 98-11, Oct 1998.
[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, Jun. 1981.
[14] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE ICRA*, 1985.
[15] I. Dryanovski, W. Morris, and J. Xiao, "Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles," in *IEEE/RSJ IROS*, Oct 2010.