# HMAPs - Hybrid height-Voxel maps for environment representation

# HMAPs – Hybrid Height-Voxel Maps for Environment Representation

Luís Garrote, Cristiano Premebida, David Silva, Urbano J. Nunes

***Abstract*— This paper presents a hybrid 3D-like grid-based mapping approach, that we called HMAP, used as a reliable and efficient 3D representation of the environment surrounding a mobile robot. Considering 3D point-clouds as input data, the proposed mapping approach addresses the representation of height-voxel (HVoxel) elements inside the HMAP, where free and occupied space is modeled through HVoxels, resulting in a reliable method for 3D representation. The proposed method corrects some of the problems inherent to the representation of complex environments based on 2D and 2.5D representations, while keeping an updated grid representation. Additionally, we also propose a complete pipeline for SLAM based on HMAPs. Indoor and outdoor experiments were carried out to validate the proposed representation using data from a Microsoft Kinect One (indoor) and a Velodyne VLP-16 LiDAR (outdoor). The obtained results show that HMAPs can provide a more detailed view of complex elements in a scene when compared to a classic 2.5D representation. Moreover, validation of the proposed SLAM approach was carried out in an outdoor dataset with promising results, which lay a foundation for further research in the topic.**

## I. Introduction

In this work we propose a new representation method, we call Height-Voxel Map (HMAP), that encompasses characteristics from 3D and 2D representations (using a hybrid approach). The HMAP representation is composed by multiple Height-Voxels (HVoxels) aligned vertically in a 2D grid-map, as exemplified in Fig. 1 for an indoor environment. An HVoxel is a vertical voxel created from raw measurements and is defined by four parameters: number of samples ($|v|$); voxel density ($\rho$); minimum height ($z_{min}$) and maximum height ($z_{max}$). For each stack of vertically aligned HVoxels an occupancy probability is also provided. This particular representation presents some important characteristics, such as: no padding is added to the voxels, meaning that to some degree it is possible to recover a height upsampled version of the original 3D point-cloud; for the best case scenario (e.g., wall), this representation becomes a 2.5D representation O(1), and for the worst case a O(log(n)) search.

In terms of contributions, this work proposes:

- HVoxel representation (local representation) and HMAP (global representation) mapping pipeline - a 3D point-cloud is converted into a set of occupied HVoxels (the HVoxel representation), a ray tracing technique is applied in order to compute free-space HVoxels and the HMAP is updated accordingly.

The authors are with the Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Portugal. This work was supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD grant with reference SFRH/BD/88672/2012. E-mails: {garrote, cpremebida, urbano}@isr.uc.pt.

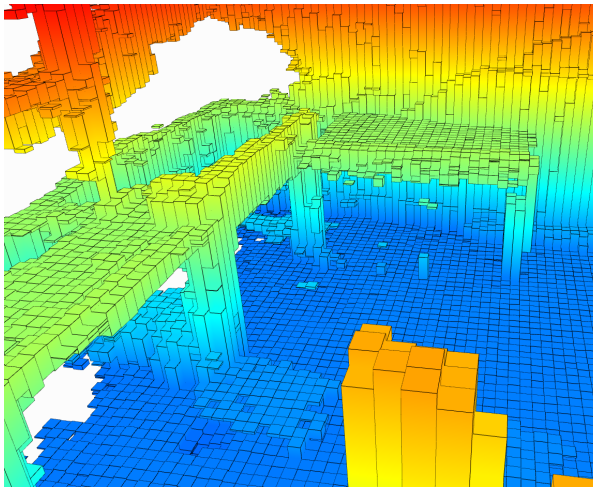

Fig. 1. A snapshot of an HMAP representation of an indoor environment (office setting).

- Hybrid HMAP representation - for each cell, a set of vertically aligned HVoxels is converted into an occupancy probability.
- SLAM based on HMAP - the method proposed in [1] for 2D grid-maps was adapted for HMAPs.

The remainder of this paper is organized as follows. An overview of the related work is provided in Section II. Section III details the proposed representation and mapping framework. Section IV describes the hybrid representation and the SLAM framework. The following section presents tests, performed in indoor and outdoor scenarios, and discussions. Conclusions are presented in Section VI.

## II. Related Work

This section gives a review of related works on environment representation in mobile robotics for indoor and outdoor scenarios. There are many ways to represent the environment surrounding a mobile robot and research in environment representation has been the focus of many research teams. Multiple solutions are already available and well field-tested through the Robot Operating System (ROS) platform [2]. A breakthrough was achieved with the 2D occupancy grid maps [3], a representation that is still widely used in many mobile platforms. Some of the key characteristics of this representation include a fast update of occupancy values and fast-constant cell access time. These characteristics mean that this representation can be deployed in a wide variety of mobile platforms including ones with low cost processing units. However, this representation presents some shortcomings when representing 3D objects and scaling to large areas.

In trying to provide some answers to these shortcomings, 2.5D and 3D representations have been proposed.

The 2.5D representation, in its initial implementations, solved the problem of representing height in 2D maps by providing a representation of elevation as "height" information on each cell. In the category of 2.5D representations, as in [4], 2.5D polar grid maps were proposed for 3D point-cloud inputs where the representation was adapted to the sensor's measurement model (Velodyne HDL-32E), and interpolation methods such as inverse distance weighting (IDW) and Kriging were employed to generate denser representations. The main limitation of 2.5D mapping is the representation of vertical overlapping elements (e.g., space between trees and pavement could be marked as an obstacle). However, multi-layer or multi-level approaches [5] provided a solution to this limitation by layering the environment with multiple 2D grid-like maps.

Finally, 3D mapping approaches have been proposed in order to provide a complete solution to environment representation. A multiresolution cuboid representation, based on the incremental fusion of grid cells, is proposed in [6]-[7]. The Octomap approach proposed by Hornung et al. [8], a probabilistic representation based on octrees, allows 3D multiresolution representations and querying. This representation was widely tested and is fully integrated in ROS.

From a computational point of view, the 2D and 2.5D representations provide simpler and faster representations that can have constant access time while 3D representations may present bigger memory footprints constraints on the access time depending on the depth or resolution of the desired region on the representation. Most motion planning approaches still require 2D grid-maps or 2D costmaps to compute collision free paths since planning in 3D can prove to be a computationally heavy task, and depending on the scenarios can be considered an over-engineered approach (e.g., 2D planning of a wheelchair's motion). However, when approaching to interact with objects or to capture more detailed characteristics of an environment, precise and 3D-like representations may be crucial to provide optimal and safer planning solutions.

In the context of mobile robots, converting a 3D or 2.5D representation into a 2D representation requires the definition of a ground plane reference or a direct mapping strategy. Although many strategies have been proposed to solve this problem, only two solutions are more related to the work presented here. In [9], a 3D point-cloud to 2D grid-map pipeline was proposed featuring good performance in the representation of non-trivial obstacles (e.g., gutters, stairs). The pipeline is composed by a 2.5D intermediary representation, a ground plane detection algorithm and an inverse sensor model. In [10] a fusion model incorporates vertical penalization based on elevation thresholds to convert a 3D point-cloud to a 2D grid-map.

In order to extend the proposed representation into a SLAM approach, we build on the work presented in [1] for 2D grid-maps, known in the ROS community as *Hector_SLAM*[1], where a Gauss-Newton minimization and bilinear interpolation are used in order to minimize the matching error between a map and a laser-scan 2D point-cloud.

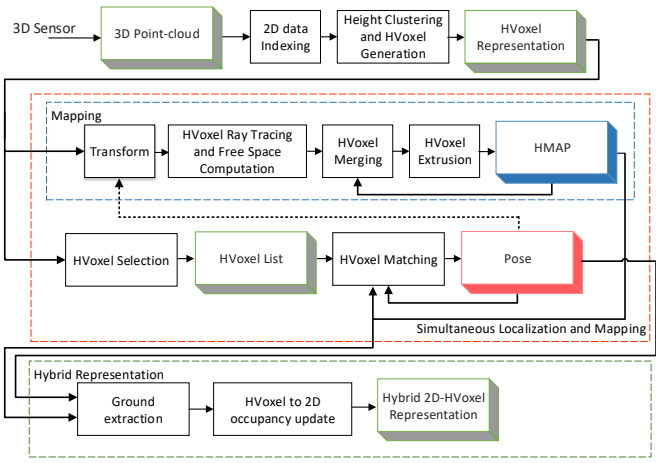


Fig. 2. Pipeline of the proposed method including the hybrid representation (green dotted rectangle), the mapping pipeline (blue dotted rectangle) and the SLAM framework (red dotted rectangle).

## III. HMAPs Mapping Framework

In this Section we present the pipeline to generate HMAPs. The overall proposed approach for mapping, SLAM and hybrid 2D-HVoxel representation is presented in Fig. 2. The first step, comprised by two modules (*2D Data Indexing* and *Height Clustering and HVoxel Generation*), consists in generating an HVoxel representation. The representation is then transformed, given an external pose, and an HMAP is initialized or updated (*HVoxel Ray Tracing and Free Space Computation*, *HVoxel Merging* and *HVoxel Extrusion*).

### A. 2D Data Indexing

The first step on this process requires the downsampling of the 3D point-cloud into a 2D grid map ($R$) representation. $R$ is defined as a two-dimensional environment representation bounded between ($[0,\ n_{rows}], [0,\ n_{cols}]$) and composed by a set of cells $c_{ij}$, with row $i$ and column $j$. Each cell $c$ is defined by constant size and contains a set of elevation measurements. The data structure used in the HMAP ($M$) follows the same structure presented for the representation $R$. Given a point-cloud ($P$) composed by a set of 3D Cartesian points ($\mathbf{p}_k = (x_k, y_k, z_k)^T,\ k = 1, 2, ..., n$), each cell is updated by projecting the $x$ and $y$ components on the grid cell. The algorithm used to generate a 2D grid-map from a 3D point-cloud is summarized in Algorithm 1: the **convertToMapIndex** procedure provides the mapping between the coordinates in a Cartesian frame $(x, y)$ to the grid coordinates $(i, j)$; while the procedure **withinMapBounds** checks if a provided point $p^{ij}$ belongs to the current 2D map.

[1]Hector_SLAM – http://wiki.ros.org/hector_slam

**Algorithm 1:** 2D data indexing for a 3D point-cloud $P$: the correspondent 2D grid map is generated; where each cell contains a set of elements corresponding to the height.

**Input**: 3D point-cloud ($P$)

**Initialization**:
Create empty 2D Representation $n_{rows} \times n_{cols}$ ($R$)
**foreach** $p$ *in* $P$ **do**
    $p^{ij} \leftarrow$ convertToMapIndex($p$)
    **if** *withinMapBounds*($p^{ij}$) **then**
        $R(p^{ij}) \leftarrow R(p^{ij}) \bigcup p_z$

**Output**: $R$

**Algorithm 2:** Height Clustering and HVoxel Generation: Converts a set of height elements into an HVoxel.

**Input**: Representation map $n_{rows} \times n_{cols}$ ($R$)

**foreach** $c$ *in* $R$ **do**
    **if** $\neg$ *Empty(cell)* **then**
        $set \leftarrow$ HeightClustering($cell$)
        **foreach** $segment$ *in* $set$ **do**
            $c \leftarrow c \cup createHVoxel(segment)$
    **else**
        $c \leftarrow \varnothing$

**Output**: $R$

### B. Height Clustering and HVoxel Generation

In our approach, each cell in $R$ contains its corresponding height values therefore, in order to generate HVoxels, we apply an height clustering point-distance algorithm (see the survey [11]) with a point-distance threshold on the z-axis (**HeightClustering**); this is summarized in Algorithm 2. This approach allows, to a certain degree, to recover an upsampled 3D point-cloud, with the HVoxels representing height boundaries. In the context of this work, an HVoxel is defined as a fixed-size voxel in the boundaries of a 2D grid-map cell, and having a maximum ($z_{max}$) and minimum ($z_{min}$) heights. Unlike 2.5D grid-maps, in HMAPs each cell can contain multiple HVoxels. Another characteristic of this HVoxel is the normalized voxel density ($\rho(v)$), as proposed in [9], which is given by:

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

where $V_{voxel}$ denotes the volume of the HVoxel ($V_{voxel} = hA$), $|v|$ is the number of points inside the HVoxel, $K_m$ denotes an amplitude gain, $K_n$ is a sample normalization factor, and $d_{min}$ represents the minimum number of points inside the HVoxel.

### C. HVoxel Ray Tracing and Free Space Computation

A 3D point-cloud does not represent *per se* free space but, knowing the 3D sensor field-of-view (FOV), a 3D ray tracing algorithm for HVoxels can be used (as shown in Algorithm 3). For each HVoxel in the representation $R$, a 3D point-cloud is projected (from the HVoxel to the 3D sensor's frame) into a 2D grid map plane (see Fig. 3). On the 2D plane we compute the cells corresponding to the 3D line using the Bresenham's algorithm [12] (**ProjectLineTo2D**). Since the maximum and minimum height values per HVoxel are available, we define two line models from the maximum and minimum heights to the sensor's origin (**lineModel**), and for each cell in the 2D grid map we compute a free space HVoxel (**createFreeHVoxel**). If the normalized density of an HVoxel ($\rho$) is bellow a threshold, the computation of free space HVoxels for that particular HVoxel is ignored.

**Algorithm 3:** HVoxel ray tracing and free space computation: Given sensor origin and an HVoxel we compute the free space HVoxel pathway.

**Input**: HVoxel representation $n_{rows} \times n_{cols}$ ($R$)
Sensor origin $p_b \leftarrow (x_i, y_i, z)$

**foreach** $cell$ *in* $R$ **do**
    **if** $\neg$ *Empty(cell)* **then**
        **foreach** *HVoxel in cell* **do**
            **if** *HVoxel*($\rho$) $\geq n_{th}$ **then**
                $cellIDs \leftarrow$ ProjectLineTo2D($p_b$, HVoxel)
                $ulm \leftarrow$ lineModel($p_b$, HVoxel($z_{max}$))
                $llm \leftarrow$ lineModel($p_b$, HVoxel($z_{min}$))
                **foreach** $c_{ID}$ *in* $cellIDs$ **do**
                    $R \leftarrow$ createFreeHVoxel($c_{ID}$, $llm$, $ulm$)

**Output**: $R$

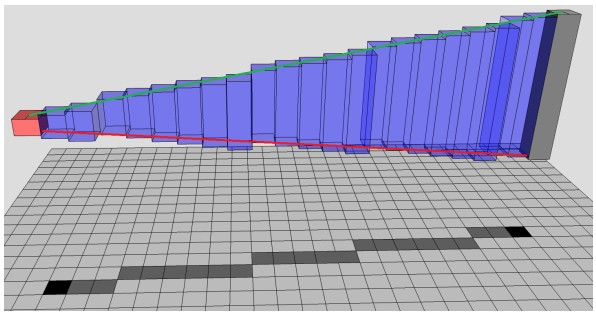

Fig. 3. Representation of the HVoxel ray tracing method for one HVoxel on a 2D grid map. Considering the HVovels, we have: the base point (in light red), the free HVoxels (in light blue), and the lines connecting the maximum and minimum values of $z$, in red and green respectively. Considering the 2D grid, the dark-grey corresponds to the cells where the free-space HVoxel will be added and, in black, the base point and the occupied HVoxel projected in the 2D grid map.

### D. HVoxel Merging

Given a previous HMAP, $M_{k-1}$, and an HVoxel representation from the last 3D point-cloud $R^*$, an update on the previous HMAP is required in order to obtain the current HMAP $M$. Assuming that all maps are referenced to the same global coordinate system (in mapping or SLAM contexts a transformation is applied to $R^*$ in order to align $R^*$ in $M_{k-1}$), for each overlapping HVoxel we update the minimum and maximum height values, the sum of the number of points and also the corresponding normalized

HVoxel density. This final step generates the HMAP, $M$, without the integration of free-space HVoxels.

### E. HVoxel Extrusion

On a new HMAP, $M$, we apply the concept of extrusion to HVoxels. For a set of HVoxels and free-space HVoxels we apply the following rules:

- If no free-space HVoxels overlap HVoxels then, no changes are made to the representation.
- If a free-space HVoxel overlaps HVoxels but has a smaller normalized density then, the overlapped HVoxel is updated accordingly to the overlapped area ($|v| = |v| - d * |v_f|$); furthermore, if the final normalized density is below a given threshold the overlapping area is removed.
- If a free space HVoxel overlaps but has bigger normalized density then, the overlapping area is removed.
- If a free space HVoxel is inside an HVoxel and the normalized density is smaller then, the HVoxel is subdivided, otherwise the HVoxel is updated.

After applying these rules, all HVoxels with small density are removed.

## IV. Hybrid HMAPs and SLAM

In the previous section the HMAP framework was introduced to deal with 3D environments, on the other hand in this section we present a 2D occupancy grid-map that can be stored alongside the proposed HMAP (each cell contains the occupancy probability) and is updated using data from the representation. An overview of the proposed Hybrid Representation pipeline is presented in Fig. 2 (green dotted rectangle).

### A. Ground Extraction

In [9] a conversion pipeline was proposed to convert from 3D point-clouds to 2D grid-maps using a Rapidly-exploring Random Tree (RRT) inspired ground plane detection algorithm (RRT-GPD), which is suitable for indoor environment representation. Also in [9] the approach was particularly developed for assistive robotics contexts (i.e., mapping floor outlets, gutters or steps as obstacles). In this subsection, we present a new version of the approach conveniently adapted for HMAPs.

Proposing a solution simultaneously suitable for indoor and outdoor scenarios is a complex task. Although for most indoor scenarios a ground plane detection algorithm will suffice, outdoor scenarios have different requirements and detecting a single plane may not be enough. Therefore, we propose to solve this problem by detecting drivable patches using a RRT inspired algorithm (see Algorithm 4). Patches in the context of this work are small local planes defined by position, normal vector, height and width dimensions. We choose to represent the tree in a Kd-tree for fast patch queries (search complexity is O(log n)).

Given an initial pose, the first step is to compute an histogram of the neighborhood heights in order to select an acceptable candidate ($\mathbf{p}_0$) to start the tree (**NeighborhoodHistogram**), then the tree is initialized (**initializeTree**) with the initial patch centered in $\mathbf{p}_0$. The patch is computed using local interpolation and a plane fitting algorithm and, until a maximum number of iterations $K$ is met, the algorithm follows a RRT-like exploration approach. Initially, the search space is restricted to a neighboring region but increases and moves (**updateSearchSpace**) to the average of all valid sampled points reducing in this way the number of samples needed to provide a minimal set of patches.

The sampling procedure **sampleRandomDirection** generates 2D points ($x_{rand}$) uniformly sampled in the search region. A nearest node approach is then performed to obtain the nearest patch in the tree. Since the distance between the sampled points and the nearest patch can be higher than a defined threshold, meaning that a new patch might not be valid depending on the patches in between (e.g., a rail in a highway). The **projectTo** procedure is used to provide a patch center point candidate which, for indoor and outdoor 2D scenes, is given by :

$$x_{expansion} = \begin{bmatrix} x_{near}(x) + d_n \cos \measuredangle(x_{near}, x_{rand}) \\ x_{near}(y) + d_n \sin \measuredangle(x_{near}, x_{rand}) \end{bmatrix} \quad (2)$$

where $d_n$ is the node expansion distance. Given the center point $x_{expansion}$, a neighborhood window of points is extracted (**getHeightPoints**). If a cell has multiple HVoxels, all maximum height values are returned. If a cell is empty, a search in the neighborhood nodes is performed to find a valid HVoxel candidate ($\Delta h < d_{th}$) and an interpolation step is applied to the candidate's maximum height. The interpolation uses the IDW algorithm, and helps the generation of smoother patches. If no valid points are detected in the patch then the patch is discarded. The patch's normal is computed by **patchGeneration** using a least squares plane fitting algorithm. Once a valid patch is found, and depending on the requirements of a particular application, a set of conditions must be verified to ensure that a connection between two patches (the patch represented in $x_{near}$ and $p_{local}$) is valid. This algorithm ensures that a connection between two patches is valid by verifying the displacement of the normals of both patches and the elevation between them. If one of the conditions is not met the candidate can still be connected to neighbor patches otherwise the patch is added to the tree and the search space is updated. If neither the nearest patch nor the neighboring patches meet the conditions, the candidate is rejected. After $K$ iterations, the output of this algorithm is a tree composed by patches that cover the searchable space.

### B. HVoxel to 2D Occupancy Update

To convert an HVoxel into an occupancy probability it is necessary, as the first step, to query a patch. The nearest patch to an HVoxel represents the local ground plane. HVoxels with $z_{min}$ and $z_{max}$ close to the local ground plane mark driveable space which translates into a small probability of a cell being occupied. Also HVoxels higher or lower than a

**Algorithm 4:** RRT based patch exploration algorithm.

**Input**: Initial Hint ($\mathbf{p}_{xyz}$), Plane normal threshold ($N_{th}$), Elevation threshold ($E_{th}$), HMAP ($M$), Maximum number of iterations ($K$), Node expansion distance ($d_n$)

**Initialization**:
$\mathbf{p}_0 \leftarrow$ NeighborhoodHistogram($\mathbf{p}_{xyz}$, $M$);
$G \leftarrow$ initializeTree($\mathbf{p}_0$);
**for** *k=1 to K* **do**
  $x_{rand} \leftarrow$ sampleRandomDirection();
  $x_{near} \leftarrow$ nearestNode($G$,$x_{rand}$);
  $x_{expansion} \leftarrow$ projectTo($x_{near}$,$x_{rand}$);
  $P \leftarrow$ getHeightPoints($M$, $x_{expansion}$);
  $p_{local} \leftarrow$ patchGeneration($P$);
  **if** $|$ *normal*($p_{local}$) $\cdot$ *normal*($x_{near}$) $| \leq N_{th}$ **then**
    **if** $| p_{local}(z)$ - $x_{near}(z) | \leq E_{th}$ **then**
      $G \leftarrow G \bigcup \{ x_{near}, p_{local} \}$;
      updateSearchSpace($x_{expansion}$);
  **else**
    $x_{neighbours} \leftarrow$ nearestNodes($G$,$x_{expansion}$,$d_n$);
    **foreach** *node* ***in*** $x_{neighbours}$ **do**
      **if** $|$ *normal*($p_{local}$) $\cdot$ *normal*(node) $| \leq N_{th}$ **then**
        **if** $| p_{local}(z)$ - $node(z) | \leq E_{th}$ **then**
          $G \leftarrow G \bigcup \{ node, p_{local} \}$;
          updateSearchSpace($x_{expansion}$);

**Output**: $G$

predefined threshold are truncated or ignored (i.e., to ignore the detection of the ceiling in indoor environments or trees in outdoor environments as obstacles). A log-odds probability that a cell $c$ is occupied given the observations $z_{1:t}$ (see [13]) is given by:

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

where $p(c)$ is the prior probability, $p(c|z_{1:t-1})$ the previous estimate and $p(c|z_t)$ the probability that a cell $c$ is occupied given the measurement $z$ and it is computed using an inverse sensor model. The proposed inverse sensor model to convert vertically stacked HVoxels into an occupancy probability is given by:

$$p(c|z_t) = \max_{i=1}^{N_H} \begin{cases} 0.5, if\ v_i(z_{min}, z_{max}) > z^+ \cup v_i(z_{min}, z_{max}) < z^- \\ \begin{cases} \max(v_i(\rho), 0.5)\ e^{-\frac{(|c|-|v_i|)^2}{2\sigma^2}}, if ||p_{local} - v_i|| > d_{pth} \\ K_g + \frac{0.5-K_g}{1+e^{-(|c|-|v_i|)}}, if ||p_{local} - v_i|| \leq d_{pth} \end{cases} \\ , otherwise \end{cases} \tag{4}$$

where $\sigma^2$ denotes the variance, $K_g \in [0, 0.5]$ a bias gain, $d_{pth}$ a distance threshold, $p_{local}$ a local patch, $N_H$ the number of vertical HVoxels, $z^+$ and $z^-$ the detection limits. The inverse sensor model is valid for $c \in [v_o, v]$. $v_o$ is an HVoxel representing the origin of the sensor's frame and the path between $v_o$ and $v$ is computed using a ray tracing algorithm.

### C. Simultaneous Localization and Mapping

In this subsection we present a SLAM approach using the HMAP representation (see red dotted rectangle highlighted in Fig. 2). Following a similar approach as in [1], the goal is to find a rigid transformation $\xi = (p_x, p_y, \varphi)$ that minimizes the error between a local representation (HVoxel representation) and a global representation (HMAP)

$$\xi^* = \underset{\xi}{argmin} \sum_{i=1}^{n} [1 - M_H(S_i(\xi)))]^2 \tag{5}$$

where $M_H(S_i(\xi))$ is a function that returns an HVoxel related metric and $S_i(\xi)$, which denotes a 2D transformation to an HVoxel representation in the HMAP ($v_x$,$v_y$), is expressed as

$$S_i(\xi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \tag{6}$$

The Gauss-Newton equation for the minimization problem is given by:

$$\Delta\xi = H^{-1} \sum_{i=1}^{n} ((1 - M_H(S_i(\xi))) (\bigtriangledown M_H(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi})^T \tag{7}$$

where $\bigtriangledown M_H(S_i(\xi))$ is the map representation gradient and $H$ is the Hessian matrix given by:

$$H = \sum_{i=1}^{n} \left[ \bigtriangledown M_H(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi} \right]^T \left[ \bigtriangledown M_H(S_i(\xi)) \frac{\delta S_i(\xi)}{\delta \xi} \right] \tag{8}$$

For further details, the full derivation is provided in [1]. Some of the contributions in [1] include the use of a bilinear interpolation to approximate the representation gradient and a multiresolution representation of the map to avoid local minimum. For the HMAP representation, and depending on the scenario, generating multiresolution maps can be a memory expensive task, and preliminary results showed that in this particular case, decreasing the resolution meant the degradation of the final results. Also, the bilinear interpolation was applied to a 2D grid map, meaning that for the HMAP representation and, in particular for an HVoxel, even if the base of the representation is a 2D grid-map, it is not possible to define multiple 2D grid maps for each HVoxel. The proposed solution is shown in Fig. 4 and consists in the definition of regions of interest around an HVoxel. The bilinear interpolation is applied only after the region of interest has been interpolated by an IDW method adapted for HVoxels.

Each HVoxel contains its normalized density that, to some degree, defines if the HVoxel is valid or not. Even if an HVoxel has high density, this value is not enough to compare two HVoxels since they can have similar densities and correspond to different characteristics on the environment. To compare HVoxels we propose the Intersection over Union $IoU$ criterion (also known as Jaccard index) for HVoxels,

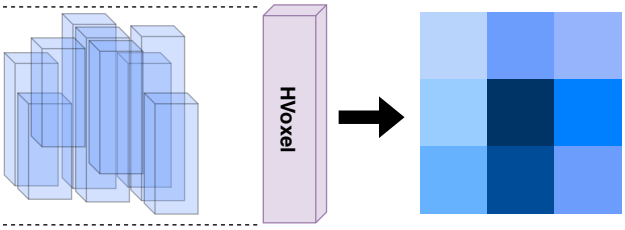


Fig. 4. Layered interpolation of HVoxels in a 3x3 window - For a given HVoxel candidate, the $IoU$ metric is computed inside the boundaries of the HVoxel.

$IoU(v) = \frac{V_{intersection}}{V_{union}}$. The HVoxel volume is given by $V = hA$. Since the area of the base is the same for each HVoxel and based on the vertical stacking nature of the representation, the $IoU$ criterion can be expressed as follows:

$$IoU(v) = \frac{\sum_{i=1}^{N} (v(h) \cap v_i(h))}{\sum_{i=1}^{N} (v(h) \cup v_i(h))} \tag{9}$$

where $N$ is the number of HVoxels present in the layer defined by the HVoxel $v$, and the HVoxel height is given by $v(h) = v(z_{max}) - v(z_{min})$.

A selection of HVoxels is performed in the "Height-Voxel Selection" step (see Fig. 2). The selection is performed by taking into account the normalized density. If an HVoxel density is above a given threshold, that HVoxel is used in the SLAM. The $M_H(v)$ is obtained by applying the bilinear interpolation,

$$\begin{aligned} M_H(v) \approx \frac{y - y_0}{y_1 - y_0} \left( \frac{x - x_0}{x_1 - x_0} F_{11} + \frac{x_1 - x}{x_1 - x_0} F_{01} \right) \\ + \frac{y_1 - y}{y_1 - y_0} \left( \frac{x - x_0}{x_1 - x_0} F_{10} + \frac{x_1 - x}{x_1 - x_0} F_{00} \right) \end{aligned} \tag{10}$$

while its partial derivatives that approximate the map representation gradient are calculated as follows:

$$\begin{aligned} \frac{\partial M_H}{\partial x}(v) &\approx \frac{y - y_0}{y_1 - y_0} (F_{11} - F_{01}) + \frac{y_1 - y}{y_1 - y_0} (F_{10} - F_{00}) \\ \frac{\partial M_H}{\partial y}(v) &\approx \frac{x - x_0}{x_1 - x_0} (F_{11} - F_{01}) + \frac{x_1 - x}{x_1 - x_0} (F_{10} - F_{00}) \end{aligned} \tag{11}$$

where $F_{00}$, $F_{01}$, $F_{10}$ and $F_{11}$ are locally interpolated values of the nearest grid cells by an interpolation function $F$. For the virtual layer defined by the boundaries of an HVoxel ($z_{max}$ and $z_{min}$) the interpolation function $F$ interpolates a new HVoxel if one is missing, using the IDW algorithm. Finally, having the interpolated HVoxel then the $IoU$ is computed. The IDW interpolation technique to obtain $v'(z_{min}, z_{max})$, was implemented according to the following expression:

$$v' = \left( \frac{\sum_{i=1}^{N} \frac{1}{||v - v_i||^p} (v_i(z_{min}))}{\sum_{i=1}^{N} \frac{1}{||v - v_i||^p}}, \frac{\sum_{i=1}^{N} \frac{1}{||v - v_i||^p} (v_i(z_{max}))}{\sum_{i=1}^{N} \frac{1}{||v - v_i||^p}} \right) \tag{12}$$

where $N$ corresponds to the number of neighbor cells with valid HVoxels in the $z_{max}$ and $z_{min}$ boundaries. Using these two stages of interpolation, a solution for the local minima problem often encountered in Gauss-Newton minimization is provided. In particular, this approach behaves as a dynamic multiresolution grid, in which, if a HVoxel has a match, only the bilinear filter will be used; otherwise an interpolated HVoxel contributes to the direction of the minimization. Interpolation values can be cached locally to speed up the calculation.

## V. EXPERIMENTAL RESULTS

Experimental tests where carried out with datasets acquired in indoor and outdoor settings. In the indoor setting, a Microsoft Kinect One mounted in a differential drive mobile platform performed an office traversal. For the outdoor setting, using a Velodyne VLP-16 LiDAR mounted in an electric vehicle, an exit from a underground parking-lot with a small route inside the campus was performed. No changes or special measures where taken while making these tests in order to make them as realistic as possible, meaning that dynamic obstacles such as humans or vehicles (in the outdoor setting) are present in the datasets. Both datasets where acquired using ROS and stored in the ROS BAG file format for offline processing. All the results presented in this section were generated by an in-house QT/C++ 3D environment built to handle ROS BAG files and to study environment representations.

The validation of the proposed representation was performed based on two tests. The first was a comparison between the HMAP and the pipeline proposed in [9] (2.5D and 2D representations). The second test consisted in applying SLAM to the outdoor dataset. A grid size of 600x600 was used with a cell size of 0.05 m for the indoor scenarios and 0.25 for the outdoor scenarios. The point-distance threshold was 0.15 m (indoors) and 0.4 m (outdoors). The 3D point-cloud was limited between 0.8 and 5 m for the indoor dataset and 1.2 to 25 m in the outdoor case. Other parameters were set as : $K_m = 1$, $K_n = 0.0005$, $d_{min} = 5$, $\sigma = 0.02$, $K_g = 0.3$, $K = 400$, $N_{th} = 0.2$, $E_{th} = 0.05$ (indoor), $E_{th} = 0.2$ (outdoor), $d_n = 0.4$ (indoor), $d_n = 0.9$ (outdoor), $d_{pth} = 0.03$ (indoor), $d_{pth} = 0.1$ (outdoor) and interpolation window size was 3 cells.

Figures 5 and 6 show five scenarios (three indoor and two outdoor) and corresponding results. For each scenario, five figures were generated to contextualize the results and to qualitatively evaluate the method. The first column shows the input 3D point-cloud, columns $II$ and $III$ correspond to the hybrid HMAP (3D and 2D representations) and columns $IV$ and $V$ to the 2.5D and 2D grid-map proposed in [9]. By comparing columns $II$ and $IV$ (in a detailed viewing), it is noticeable the increase in the detail in the scenario by observing the tables, and the human where it is almost perceptible that the human's arms are holding the joystick that is controlling the mobile platform: that level of detail is lost in the 2.5D representation. It is also important to note that the walls are represented similarly in both 2.5D and HMAP representations. On the indoor scenarios the 2D representation presents quite similar results, while in outdoors the 2D occupancy grids show more detail with the new hybrid representation: that is due to the method used to extract the ground plane, also proposed here, which better

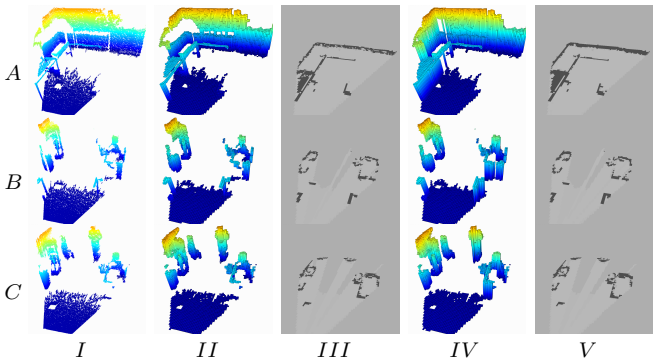


Fig. 5. Results obtained with the indoor dataset (office setting): first row corresponds to three tables ($A$), second row to a scenario with a cabinet and chairs ($B$) and third row scenario contains a human ($C$). From left to right - 3D point-cloud ($I$), HMAP representation ($II$), 2D representation from HMAP ($III$), 2.5D representation and 2D grid-map [9] ($IV$ and $V$).

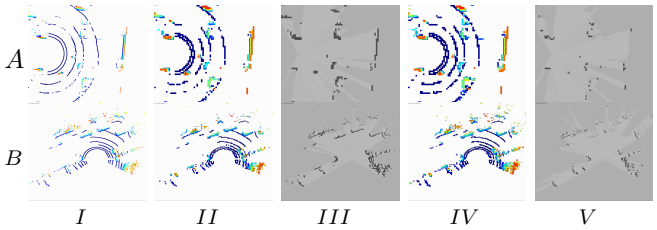


Fig. 6. Results obtained with the outdoor dataset: first row corresponds to the vehicle leaving its parking space ($A$) and second row a T intersection scenario ($B$). From left to right - 3D point-cloud ($I$), HMAP representation ($II$), 2D representation from HMAP ($III$), 2.5D representation and 2D grid-map [9] ($IV$ and $V$).

represents ground surfaces in outdoor scenarios. Overall, the proposed HMAP approach provides a detailed representation of the surrounding environment.

A second test (see Fig. 7, grid size of 2000x2000) was performed with the purpose of validation of the proposed SLAM approach. The method successfully mapped the visible parked vehicles and structural elements, both inside the parking lot and outside on the street. It was also able to correctly match those elements with the map resulting in an accurate representation of the environment, and the path generated (in red) closely matched the path performed by the vehicle.

## VI. Conclusion

In this paper we presented the HMAP framework, a consistent 3D-based environment representation suitable for indoor and outdoor scenarios. Mapping and SLAM approaches were also explored based on the HMAP representation. The proposed approach takes as input 3D point-clouds and provides rich hybrid representations, suitable for planning problems with multiple layers of decision (e.g., general and fine planning). Experiments in indoor and outdoor scenarios using moving robotic platforms, with 3D sensors mounted onboard, showed increased details obtained by the HMAP in the scenarios representation. In addition, the HMAP-based SLAM implementation achieved promising results in a complex outdoor scenario.

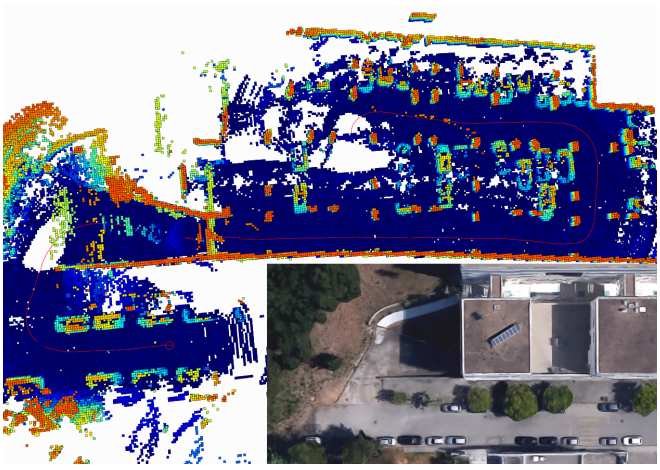

Fig. 7. SLAM results obtained in the outdoor dataset. In red the estimated localization of the electric vehicle, on the bottom right a Google Maps view of the map.

## Acknowledgment

Part of this work has been supported by UID/EEA/00048/2013, AGVPOSYS (CENTRO-01-0247-FEDER-003503) and MATIS (CENTRO-01-0145-FEDER-000014) projects, with FEDER funding, programs PT2020 and CENTRO2020.

## References

[1] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[3] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE ICRA*, 1985.

[4] 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.

[5] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ IROS Conference*, Oct 2006.

[6] S. Khan, A. Dometios, C. Verginis, C. Tzafestas, D. Wollherr, and M. Buss, "RMAP: a rectangular cuboid approximation framework for 3D environment mapping," *Autonomous Robots*, vol. 37, no. 3, pp. 261–277, 2014.

[7] S. Khan, D. Wollherr, and M. Buss, "Adaptive rectangular cuboids for 3D mapping," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015.

[8] 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.

[9] L. Garrote, J. Rosa, J. Paulo, C. Premebida, P. Peixoto, and U. Nunes, "3D point cloud downsampling for 2D indoor scene modelling in mobile robotics," in *2017 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2017.

[10] J. D. Adarve, M. Perrollaz, A. Makris, and C. Laugier, "Computing occupancy grids from multiple sensors using linear opinion pools," *IEEE ICRA*, 2012.

[11] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2D laser range data for mobile robot applications," *Robótica 2005 - Actas do Encontro Científico*, 2005.

[12] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[13] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," 2005.