Efficient Plane Detection in Multilevel Surface Maps

V. Prieto-Marañón, J. Cabrera-Gámez, A. C. Domínguez-Brito,
D. Hernández-Sosa, J. Isern-González, E. Fernández-Perdomo

***Abstract*—An automatic system aimed at producing a compact tridimensional description of indoor environments using a mobile 3D laser scanner is described in this paper. The resulting description is made up of a Multi-Level Map (ML map) and a series of planar patches extracted from the map. We propose a novel plane detection algorithm, based on the efficient RANSAC algorithm, that operates directly over the data structures of an ML map and does not need to rely on the low level laser data cloud. The mobile 3D scanner is built from a Hokuyo laser range sensor attached to a 2DOF pan-tilt, which is installed on top of a 3DX Pioneer mobile robot. The 3D spatial information acquired by the laser sensor from different poses is used to build a large single map of the environment using the SLAM 6D library. Experimental results demonstrate that the system described is capable of efficiently building compact and accurate 3D representations of complex large indoor environments at multiple semantic levels.**

***Index Terms*—3D Maps, plane detection, multilevel surface maps, laser scanner, SLAM6D**

I. Introduction

EFFICIENT use of robots in a tridimensional environment, be it indoor or outdoor, requires identification of structures and objects present in the world. These objects and structures can often be described in terms of simpler forms or primitives at different semantic levels. For example, indoor primitives based on planes can be used to characterize most of the elements conforming the environment. Their descriptions and relative locations can be used to define the internal representation or map that is to be used by the robots acting in the environment.

Maps are built from information acquired from one or more sensors. Numerous different sensors can be used to capture information in a 3D scenario from cameras (monocular, stereo or time-of-flight) or range sensors (sonars and lasers). Maps may be topological or metric, but metric maps are the preferred option when geometrical features from the environment, such as distances, volumes or surfaces, are needed.

The problem of building maps in large and unknown environments while the system orients itself, widely known as the SLAM problem, has been intensively studied by the robotics community over the last ten years [1]. Basically, the problem lies in incrementally adding new information to a map whilst estimating the relative displacements between observations and recognizing areas that have already been explored and are present in the map. This problem in two dimensions has been largely studied and —in general terms— is nowadays considered solved. The latest achievements in SLAM, together with the availability of faster sensors and processors, have fostered an interest for extending the SLAM problem to 3D scenarios with 6DoF observers, a context formerly termed as unfeasible due to its high computational demands.

In the present work, we describe an approach that allows us to construct 3D maps for large indoor scenarios using a mobile robotic system equipped with a laser sensor mounted on a pan-tilt unit. The resulting tridimensional representation of the environment comprises two levels of description. At the lowest level, a Multi-Level map (ML map) [2] is built integrating 3D scans acquired from different poses. To offset odometry errors, we address the inherent SLAM problem using the SLAM 6D software developed by Nüchter et al. [3]. From the ML map the system can detect planar patches using an algorithm that is an optimized adaptation for plane detection of the efficient RANSAC (eRANSAC) method [4]. These planar patches will be used to define a second level of description in the 3D map of the environment from which structures of higher semantic level such as walls, doors, tables, etc, could be detected.

This paper is set out as follows: after discussing related works in section II, the data acquisition system will be described in section III. Section IV shows how the ML maps are built. The plane detection algorithm in ML maps will be explained in section V. Finally, several experimental results and conclusions are discused in sections VI and VII respectively.

II. Related work

Different approaches have been adopted to allow autonomous mobile robots to build tridimensional maps of the environment. Several methods use a tridimensional grid that splits space into portions called voxels, whose values reflect the occupancy of the corresponding space volume [5]. Other authors have proposed using elevation maps, due to their much lower memory requirements. In elevation maps, the environment is represented by using a two dimensional grid where each cell represents the elevation, i.e. terrain height, at the corresponding point. These maps enable us to model large

This work has been partially supported by the Research Project *TIN2008-06068* funded by the Ministerio de Ciencia e Innovación, Gobierno de España, Spain.

V. Prieto-Marañón, J. Cabrera-Gámez, A. C. Domínguez-Brito, D. Hernández-Sosa, J. Isern-González and E. Fernández-Perdomo are with the Instituto Universitario SIANI, Universidad de Las Palmas de Gran Canaria, Spain.

J. Cabrera-Gámez, A. C. Domínguez-Brito and D. Hernández-Sosa are with the Instituto Universitario SIANI, and the Departamento de Informática y Sistemas, Universidad de Las Palmas de Gran Canaria, Spain.

Corresponding author: vprieto@ono.com

Fig. 1. An example of scenario with several surfaces crossing at different heights.

environments as shown in [6]. However, elevation maps are ill-suited to modeling scenarios containing structures crossing at different heights over the vertical of a point (see figure 1). Two examples are a table indoors or a bridge outdoors. In order to avoid this limitation, Triebel et al. [2] propose multilevel surface maps as an extension to elevation maps. These multilevel maps include, at every cell of a bidimensional grid, a list of the traversable surfaces that exist in the corresponding vertical. An improvement of the multilevel surface maps can be found in [7] where they are formally described using a probabilistic approach.

Detecting shapes in tridimensional data sets has been studied from different points of view. Starting from a 3D data point cloud, in [8], a 2 $\frac{1}{2}$ dimensional structure was built based on an incremental triangulation algorithm. Similarly, in [9], the authors developed a plane detection method using a more accurate range noise model for 3D sensors to derive from scratch the expressions for the optimum plane which best fits a point-cloud and for the combined covariance matrix of the plane's parameters. The parameters in question are the plane's normal and its distance from the origin. In other works, plane detection is addressed by using the information extracted from imaging sensors. A range imaging sensor is used in [10], with the aim of segmenting images of indoor environments in terms of horizontal and vertical planes by means of the Normalized-Cuts algorithm. An approach by Hähnel, Burgard and Thrun is presented in [11]. This work describes an algorithm for full 3D shape reconstruction of indoor and outdoor environments with mobile robots by approximating environments using flat surfaces. Other authors [12] present a method for obtaining the location, size and shape of main surfaces in an environment from points measured by a laser scanner onboard a mobile robot. The most likely orientation of the surface normal is first calculated at each point, from points in an adaptive-radius neighboring region. In other cases, stereo cameras are used, for example in [13], where an architecture for detection and estimation of planar surfaces in the scene from calibrated stereo images is presented.

III. Data acquisition

In this work, the data acquisition system is formed by a laser sensor coupled with a pan-tilt, both installed onboard a mobile robot. The laser sensor is a Hokuyo UTM-30LX with a scan width of $270°$ and $30\mathrm{m}$ detection range. The pan-tilt unit is a PTU 46-17.5 from Directed Perception. It has two degrees of freedom and it is used for scanning space in three dimensions. A Pioneer P3-DX has been used as a mobile platform and as the odometry data source.

The data acquisition system works in a move-and-stop way. The robot is moved to a new pose and then a 3D scan is taken. The pan-tilt is oriented with a pan angle α and, then, while the pan-tilt sweeps between the tilt start angle γ_s and the tilt end angle γ_e, the laser sensor takes range measurements from the environment. The laser sensor returns one scan every 25 msecs. The tilt angular speed is adjusted to obtain an angular separation between consecutive scans of ρ degrees at the maximal speed allowed by the hardware.

To integrate new laser measurements into the map we need to know the laser sensor orientation at all moment. The hardware used does not have a hardware synchronization system, so we have developed a software synchronization mechanism that allows us to acquire new laser measurements while the pan-tilt is moving between γ_s and γ_e. This synchronization system avoids having to stop the pan-tilt every time the laser initiates the acquisition of a new scan. The synchronization algorithm takes into account the pan-tilt's initial position when the laser scan starts and calculates the vertical elevation angle for every measurement returned by the sensor. The scan data timestamp t_s corresponds to the moment at which the laser sensor starts acquiring a new scan. The resolution of the Hokuyo UTM-30LX sensor is 1440 steps per revolution. The timestamp of range measurement m_p corresponding to step p is:

$$t_p = t_s + \frac{p}{1440f} \tag{1}$$

In this equation, f represents the laser beam rotation frequency. If t_0 is the instant when the pan-tilt began its tilt movement, then the time difference or delay l_p till the measurement m_p was taken is:

$$l_p = t_p - t_0 \tag{2}$$

The pan-tilt's tilt speed is adjusted so that tilt angle changes in ρ degrees while the laser beam completes a revolution. Thus:

$$\upsilon = \rho \cdot f \tag{3}$$

Thus, if l_p is known, then, by using the pan-tilt's trapezoidal acceleration scheme, we can calculate the tilt angle γ at which each measurement m_p was taken. The pan-tilt uses a trapezoidal acceleration scheme to achieve any velocity that is greater than the so called base speed v_b. The pan-tilt unit is considered to be able to accelerate instantaneously from zero to any speed up to v_b. Then γ is calculated by interpolation using this scheme.

The spatial coordinates $c = (c_x, c_y, c_z)$, corresponding to the 3D point where the laser beam impacts, must be calculated for each measurement returned by the laser sensor. Thus, it is necessary to look for a transformation function f such that:

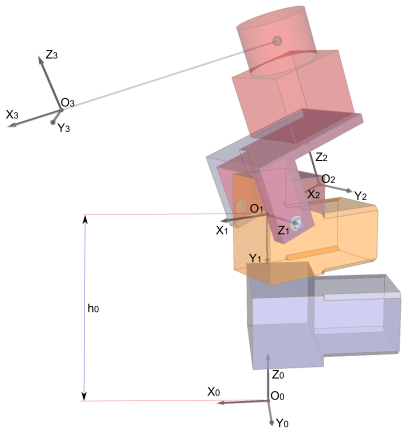


Fig. 2. Data acquisition system envisioned as a kinematic chain. The system is made up of a laser sensor (in red color, upper side) coupled over a two degrees of freedom pan-tilt. In the figure, the base reference system is showed.

$$c = f(\alpha, \gamma, \lambda, m_p, u) \tag{4}$$

If λ is the laser motor, i.e. beam, angle and $u = (u_x, u_y, u_z)$ are the coordinates from the pan-tilt and laser sensor localization. The transformation f can be easily found posing this problem as a direct kinematic problem. From this point of view, our system has three distinct joints. The first and second joints coincide with the pan-tilt's motors. The third joint corresponds to the laser sensor motor, considering the laser beam as another link of the chain (figure 2). The transformation f can then be determined by means of the Denavit-Hartenberg method [14].

A. SLAM

As equation 4 clearly shows, it is necessary to know the pan-tilt and laser sensor 3D orientation in order to merge the 3D scans taken from different places in a single map. The location of each pose is approximated by using the robot odometry. However, as is commonly accepted, odometry errors may grow without limit due to wheel sleepage or calibration errors. Specifically, one can expect odometry errors to increase rapidly with distance and turns. Hence, this error must be corrected in order to create a consistent map. To solve this issue, we first collected all the 3D scans and then the whole scan set was processed using the SLAM 6D package [3], [15]. The SLAM 6D project includes software to register 3D point clouds into a common coordinate frame. We used this registration software to correct the localization of the poses. This software matches 3D scans and it considers 6DoF for the robot pose: x, y and z coordinates and the roll, yaw and pitch angles. As a result, corrected poses are returned. We used the corrected poses to solve equation 4.

IV. ML MAP BUILDING

The first description level of the environment is based on Multilevel Surface Maps (MLSM) [2] and it is built using the 3D scans processed by the SLAM 6D package.

MLSM consists of a 2D grid where every cell $c_{i,j}$ stores a structure list. Each element of this list is represented as the mean $\mu^k_{i,j}$ and the variance $\sigma^k_{i,j}$ of the measured heights at the position of the cell in the map. Triebel et al.'s work [2] is aimed at obtaining an environmental representation that allows for robot navigation in tridimensional environments with several traversable surfaces at different overlapped heights. So, in that work, each list element (called surface patches) represents whether the space at the height indicated by the mean $\mu^k_{i,j}$, with an uncertainty equal to the variance $\sigma^k_{i,j}$, is traversable or not. Our objective, however, is to obtain a map that allows us to model and identify the objects present in the environment. Accordingly, in our map each list element, called block, represents a section of an object surface. This enables us to obtain a map that represents a compact discretization of the environment. This new approach introduces some differences during map building.

Within our ML maps, each cell $c_{i,j}$ stores a list of blocks $b^k_{i,j}$. Each measure $p = (p_x, p_y, p_z)$ returned by laser sensor is incorporated in a block so $p_x \geq j \cdot \text{cell\_size}$ and $p_x < (j+1) \cdot \text{cell\_size}$ and $p_y \geq i \cdot \text{cell\_size}$ and $p_y < (i+1) \cdot \text{cell\_size}$. The cell_size parameter expresses the map resolution. Each block is represented by a tuple (h, σ, d, π), where h is the height, σ the variance, d the depth and π the plane containing it (this last parameter will be explained in the next section). There are two block types:

1) Horizontal blocks represent a section of the external upper or lower surface of an object, for example: a floor section or a ceil part, a table board, etc. This kind of blocks has a depth equal to zero.
2) Vertical blocks, in turn, represent sections of vertical surfaces of objects like walls or wardrobes.

When new measures are acquired, the height and variance of horizontal blocks are updated using the Kalman update rule. In vertical blocks, in turn, the height and the variance are the height and variance of the highest measurement assigned to the block. The depth of a vertical block is the difference between the upper and lower measurements which fit in the block. When new measures are acquired, the map is updated as follows (see figure 3):

- Every time a new measurement (p, σ_m) is added, where $p = (p_x, p_y, p_z)$ are the coordinates and σ_m is the variance corresponding to the measurement, the cell $c_{i,j}$ where the measurement fits is selected.
- In the block list of the cell $c_{i,j}$ we look for a block (h, σ, d, π) that collects the new measurement. A block collects a measurement if $|p_z - h| < cell\_size$ and $|(h-d) - p_z| < cell\_size$.
- If there is a block that collects the measurement and this block is horizontal and $|p_z - h| < 3 \cdot \sigma$, then the height and variance of the block is updated using Kalman's update rule. In this case, the block remains horizontal. If, in turn, the block is horizontal but we obtain that $|p_z - h| \geq 3 \cdot \sigma$, then the block becomes a vertical one with $h = max(p_z, h)$ and $d = |p_z - h|$
- If the block that collects the measurement is vertical then we simply update the block height or depth as needed.

- If the new measurement is simultaneously collected by two blocks (h_1, σ_1, d_1) and (h_2, σ_2, d_2), then both blocks will be joined into a single vertical block and the old blocks are removed.
- If the measurement is not collected by any block, or the block list of the cell is empty, then a new horizontal block will be created with $h = p_z$ and $\sigma = \sigma_m$, and added to the list of cell $c_{i,j}$.

V. Plane detection

In this project, we have developed an algorithm called efficient RANSAC in Multilevel Surface Maps (eRMSM), as a modification of the efficient RANSAC (eRANSAC) algorithm [4]. While eRANSAC works in point clouds, eRMSM works directly over the block structures of an ML map and it focuses on detecting just planes.

If M is an ML map that collects a set of blocks $b^k_{i,j}$, (i, j) is the cell index pair where the block falls in and k is the block index in the cell's list, then the eRMSM algorithm detects and returns a set of planes $\Pi = \{\Pi_1, \cdots, \Pi_n\}$ in the map. Furthermore, each block is labeled with an index i which indicates that the block matches plane Π_i. Matching between a block and a plane implies that the block is close enough to the plane and that the block is part of a block setting with a similar orientation to the plane. When the algorithm stops, each block $b^k_{i,j}$ will be represented as (h, σ, d, π) where π is the index of the matching plane. A block that does not match any plane will have $\pi = 0$.

Iteratively, the algorithm produces candidate planes that are hypothesis of real planes. Each candidate plane (CP) obtains a score that is defined as a function of the blocks matching the plane. As in eRANSAC, at the end of each iteration the CP with the highest score is accepted as a valid plane only if the probability of not overlooking a better candidate is high enough. However, in the eRMSM algorithm we have changed the estimation of this probability. In our algorithm, the number of CP needed to accept a plane as valid is significantly reduced as we will demonstrate in the sequel. When a CP Π is accepted as a valid plane, each block that matches the plane is labeled with the index i of the plane. Once a CP has been accepted, any other CP that matches the accepted plane is removed from the CP list.

Before the algorithm begins, each block $b^k_{i,j}$ receives a direction vector ν. This direction vector will be used so that only blocks with a similar direction vector will produce a new CP. This vector is the normal vector to a hypothetic surface formed by the block $b^k_{i,j}$ and all the same kind of vertical or horizontal blocks in a r radious neighborhood of the block (see figure 4). To speed the process up, in eRMSM we use the Chebyshev distance as the selected distance because it does not change the result. Vector ν is calculated by using the principal component analysis (PCA) [16]. As eRMSM does not work over spatial coordinates, but over map blocks, we must supply, from each block, some coordinates that allow to obtain a vector $\nu \in \mathbb{R}^3$. Two cases must be differentiated:

- Case 1: the block $b^k_{i,j}$ is horizontal. The horizontal blocks are part of the upper or lower surface of an object such as the board in a table, or even the oblique surface of an object like a ramp. So, the direction vector that we are looking for can have any orientation in space. In this case, from the vertical blocks set $B_V = \{b^{k_1}_{i_1,j_1}, \cdots b^{k_n}_{i_n,j_n}\}$ that exist in a setting with radius r of $b^k_{i,j}$ we can obtain a point set $P_V = \{p_1, \cdots, p_n\}$ where $p_i \in \mathbb{R}^3$. Let $b^{k_l}_{i_l,j_l} = (h_l, \sigma_l, d_l, \pi_l)$, then the corresponding point p_l is $(i_l \cdot \text{cell\_size}, j_l \cdot \text{cell\_size}, h_l)$. PCA is applied to P_V to compute the normal vector to the surface that has the P_V elements.
- Case 2: the block $b^k_{i,j}$ is vertical. This block must be part of a vertical object: a wall, a chair back, etc. Hence, the direction vector in this block must be a vector parallel to the ground then. In this case, from the horizontal blocks set $B_H = \{b^{k_1}_{i_1,j_1}, \cdots b^{k_n}_{i_n,j_n}\}$ that exist in a setting with radius r of $b^k_{i,j}$ we can obtain a point set $P_H = \{p_1, \cdots, p_n\}$ where $p_i \in \mathbb{R}^2$. If $b^{k_l}_{i_l,j_l} = (h_l, \sigma_l, d_l, \pi_l)$, then the corresponding point $p_l = (i_l \cdot \text{cell\_size}, j_l \cdot \text{cell\_size})$. PCA is applied to P_H to compute the normal vector (vn_x, vn_y) to the surface that has the P_V elements. Using this two dimensional vector we get the vector $V_N = (vn_x, vn_y, 0)$ which is parallel to the ground.

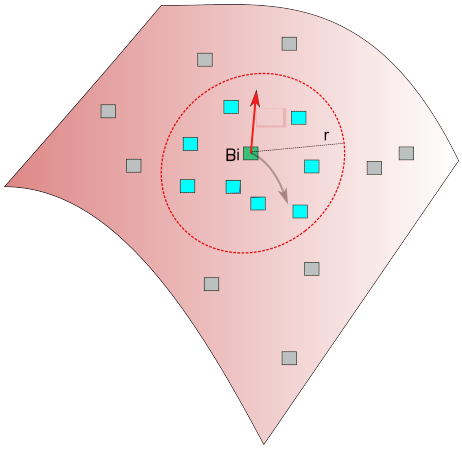


Fig. 4. The direction vector attached to block B_i is normal to a hypothetic surface formed by the block and all blocks of the same class (vertical or horizontal) in a neighbourhood radius r.

Once each block has a direction vector assigned, Algorithm 1 is executed. The candidate plane list obtains the hypotheses about planes in the environment. The detected plane list obtains the hypotheses that have been positively tested. The M variable represents the blocks list. Finally, p_t is the lowest probability considered valid to accept a hypothesis as a true plane.

The candidate planes are generated randomly selecting a block $b^{k_1}_{i_1,j_1}$ and two other blocks $b^{k_2}_{i_2,j_2}$ and $b^{k_3}_{i_3,j_3}$ close to the first that have not been matched to any other accepted plane. The neighborhood radius r is an algorithm parameter that affects the algorithm's behavior. If r is small, then the three blocks may be part of the same surface, but the plane's orientation will be affected by errors of measurement. On the other hand, if r is big, then the possibility of selecting blocks that do not match the same surface increases, but if

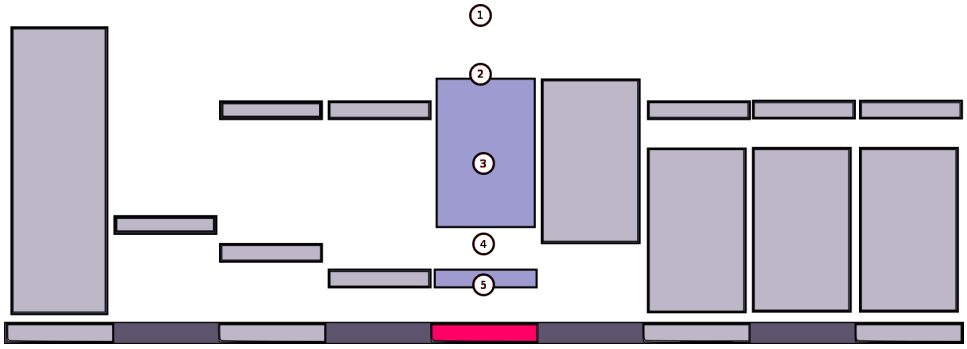


Fig. 3. Different cases when a new observation (numbered circles) is added to ML. For simplicity, the map is presented in 2D. (1) The new observation is far from registered blocks, so a new block is created. (2) The new is close to the top or bottom of a vertical block. Then, its height or depth is updated. (3) If the new observation falls inside a vertical block there are no changes. (4) The new observation is between and close enough to two registered blocks. In this case, both blocks are merged into a single one. (5) The new observation is close to a horizontal block. If the new measure is very close to the block, the height of the block is updated using the Kalman's filter updated rule. If the new observation is further than 3σ wrt. the block's, the block changes to vertical.

the blocks match the same surface, the increased distance will compensate the measurement error. The three selected blocks will generate a CP only if the angles between their direction vectors are lower than a threshold θ.

Algorithm 1 Plane detection in a ML map M

```
L_p ← ∅                                   ▷ detected plane list
L_c ← ∅                                   ▷ candidate plane list
for i = 0 to Max_cp − 1 do
    L_c ← L_c ∪ newCandidates(r, θ)
    b ← bestCandidate(L_c)
    s_c ← SimilarOrientationSurface(b)
    if P(surface(b), s_c) > p_t then
                                          ▷ matching blocks are removed:
        M ← M − M_b
        L_p ← L_p ∪ b                     ▷ CP that matches b are removed
        L_c ← L_c − C_m
    end if
end for
```

In earlier algorithms, the CP is determined as the plane that includes the three selected points (see figure 5 (a) and (b)). By contrast, to filter the surface localization error due to measurement errors, our method determines the CP in another way. The plane generated from the three blocks CP cp_i is determined as a point o and a normal vector to plane V_N. The point o is selected as the barycenter of the polygon with the three blocks as vertex and the normal vector V_N as the mean between the corresponding direction vectors. This CP represents a better hypothesis of a real plane (figure 5 c).

The way a score is assigned to each CP in eRMSM algorithm also varies in relation to previous works. Since our algorithm works with blocks, rather than point clouds, it is not possible to assign the number of matching points to CP as a score, so we propose a new score function. We will now give a definition of matching between a block and a plane. It is said that a block $b^k_{i,j}$ with a direction vector V_D matches a plane $\Pi = (o, V_N)$ if:

- The distance from the block to the plane is $d = dist(b^k_{i,j}, \Pi) < \varepsilon$.

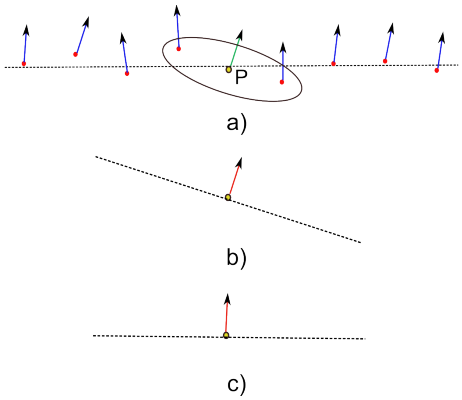


Fig. 5. Different ways to determine a candidate plane. For simplicity, in the figure the problem is depicted in two dimensions. a) Point observations obtained for a measures set of a straight line. b) Determining a candidate line as the line that best fit to three points. c) Candidate line defined as the line with a normal vector that is the mean of the direction vectors of the three points.

- The angle between the block's direction vector and the normal vector to the plane is $\beta = \arg(V_D, V_N) < \kappa$.

The thresholds ε and κ are system parameters that adjust the goodness of the accepted CP as valid planes.

Each CP receives a score depending of the area of the surface that is represented by the blocks that match that plane. To normalize the probabilistic computations, the area is measured in "surface units" s_u, where $s_u = \text{cell\_size} \times \text{cell\_size}\ \text{mm}^2$. Then the CP score is $S = \sum s_{b_i}$, where s_{b_i} is the area of the surface represented by the blocks matching the plane. The block surface depends on whether the blocks are vertical or horizontal. In a vertical block b_v the corresponding surface is $s_v = \frac{d}{\text{cell\_size}}$. When the block is horizontal it has a surface $s_h = 1$. This score method, instead of counting the number of matching blocks, as in the original method, has the advantage of being based on a real indicator of the importance of the plane in the real world. Hence, a CP that corresponds to a large surface has more possibilities of being found early on.

The CP generated in this way is likely to have some deviation in orientation with respect to the corresponding real plane. This deviation is explained by measurement errors

or due to the inclusion of neighboring blocks that in fact do not belong to that plane when computing the block's direction vector. Thus, we apply a refitness process to each CP. Whenever a new CP is created and scored we use the set of blocks that match the CP to adjust the CP's orientation using PCA. In this case, we apply PCA to the set of blocks matched by the CP to obtain the normal vector to this set. Subsequently, this vector is used as the new CP's normal vector. Finally, the modified CP is scored again. This process is recursively applied a predetermined number of iterations or when the score enhancement, from one iteration to the next, is lower than a given threshold (see algorithm 2).

Algorithm 2 Plane refitness procedure

Require: CP_i, the new plane candidate
Ensure: The optimized CP: CP_i
1: $score = score\_function(CP_i)$
2: **repeat**
3: $old\_score = score$
4: $normal = get\_normal(CP_i \rightarrow matched\_blocks)$
5: $modify\_normal\_CP(CP_i, normal)$
6: $score = score\_function(CP_i)$
7: **until** $score - old\_score < threshold$

A CP is accepted as valid only if the probability of not overlooking a better candidate is high enough. As we can see in [4], if $\wp$ is a cloud of N data points and Ψ a shape comprising n points, then the probability of detecting Ψ in a single iteration is

$$P(n) = \binom{n}{k} / \binom{N}{k} \approx \left(\frac{n}{N}\right)^k \tag{5}$$

If k is the minimum number of elements needed to define a shape —$k = 3$ for planes— thus, the probability $P(n, s)$ of successfully detecting a shape after s new candidates have been generated is

$$P(n, s) = 1 - (1 - P(n))^s \tag{6}$$

Finally, the number T of candidates needed to detect a shape of a size n with a probability $P(n, T) \geq p_t$, where p_t is the minimum desired probability, is

$$T \geq \frac{\ln(1 - p_t)}{\ln(1 - P(n))} \tag{7}$$

Applying equations 5, 6 and 7, and assuming that we have as an environment a corner formed by a ground section and two walls, if the number of points in the cloud is equally spread over the three planes, each plane has a third of the total points. Then, as 5 shows, the probability of detecting the ground in a single pass is:

$$P(n) \approx \left(\frac{1}{3}\right)^3 \approx 0.037 \tag{8}$$

Hence, according to equation 7, the number of CP that we need to detect the ground with a probability greater or equal to 0.99 is:

$$T \geq \frac{\ln(1 - 0.99)}{\ln(1 - 0.037)} > 122 \tag{9}$$

Clearly, with other shapes that represent less than a third part of the total information, the number of candidates increase significnatly as usually happens in realistic environments, where most surface planes represent a small portion of the total map. In the eRMSM algorithm, we have introduced changes to estimate the probability of not overlooking a better candidate. These changes considerably reduce the number of CP that is necessary to generate before a plane is accepted as valid.

In our approach, CP are not generated from any three blocks of the map. On the contrary, each CP is exclusively generated from three neighboring blocks with a similar orientation and therefore similar to the orientation of the plane itself. Exploiting that fact, in eRMSM algorithm, if Π is a CP where s_c is the surface of the blocks matching the plane and let s_o be the total surface of all blocks with a similar orientation to Π, we can calculate the probability of finding the plane in a single pass as

$$P(s_c) = \binom{s_c}{3} / \binom{s_o}{3} \approx \left(\frac{s_c}{s_o}\right)^3 \tag{10}$$

In the example of three planes forming a corner, the probability of finding the plane corresponding to the ground in a single pass is 1, since $s_c = s_o$ and then

$$P(s_c) \approx \left(\frac{s_c}{s_o}\right)^3 = (1)^3 = 1 \tag{11}$$

In this case, we have enough with only a single generated CP against the 123 candidates needed using the previous approach. This method can validate CP spurious planes or planes with little significance, i. e. with a small total surface if the blocks matching the plane represent a high percentage of all blocks with an orientation equal or similar to the generated CP. To avoid this, a threshold accepting candidate planes only with a score greater than a value s_m and hence with a minimal surface suffices.

The algorithm exit condition is reached when a given number of candidates is generated.

VI. Results

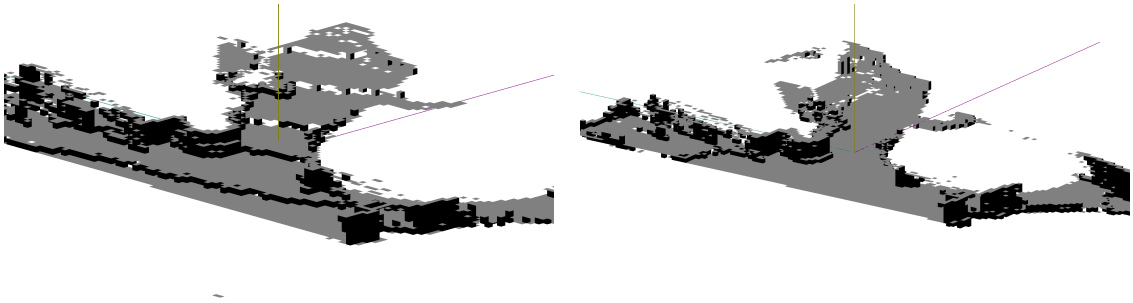
The system presented in this paper has been tested in several locations of the main building of the University of Las Palmas de Gran Canaria's Technological Park.

In the first test, we steered the robot through the basement and took 24 3D scans of the corridor (see figure 6(a)). The corridor's estimated dimensions are $40.4\mathrm{m}$ long and $4.75\mathrm{m}$ wide. The corridor has perpendicular subcorridors $11\mathrm{m}$ long.

No matter how carefully the acquisition system is placed on board the robot, it will not be parallel with respect to the floor. This inclination or deviation from horizontal causes a "step effect" in horizontal and vertical planes. The system solves this issue by self-calibration. At the beginning, a scan of the surrounding floor is adquired (see figure 7(a)). We then use the eRMSM algorithm to detect the floor plane. The normal vector to this plane is used to fix the system's inclination.

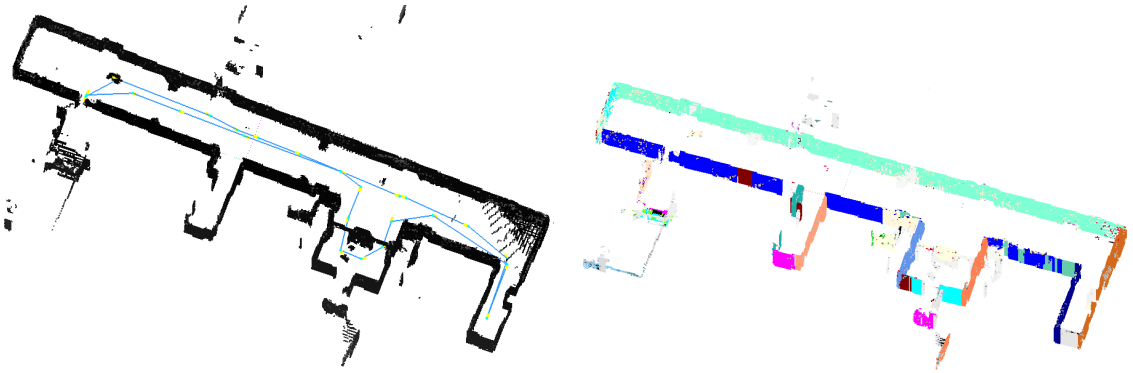
(a) (b)

Fig. 6. (a) Test scenario #1: Corridor of the Technological Park. ULPGC. (b) Test scenario #2: Robotics Laboratory of the Technological Park. ULPGC.



(a) (b)

Fig. 7. (a) "Step effect" in an horizontal plane (Robotics Laboratory's floor). (b) "Step effect" fixed using an estimate of the acquisition system's inclination.



(a) (b)

Fig. 8. (a) Upper oblique view of ML maps generated from 24 poses at the corridor. The line represents the robot path and the points over the line are the poses at which the scans were taken. (b) Planes detected in the corridor map. Grey zones represent blocks that do not match any plane. Each color represents different planes.

We can appreciate in figure 7(b) how the "step effect" has disappeared.

We used the Nüchter et al. SLAM 6D library [3] to correct the odometer location information returned by the robot regarding the robot's pose where the 24 3D scans were taken. Once the poses are corrected, we build a map from the set of measures taken in the 3D scans. A 3D visualization software was developed to make spatial zooms and rotations of the map. In figure 8(a) we can see an upper oblique view from a map of the corridor generated using a 100mm cell size. For improved visualization we have removed the floor and the ceiling from the map. In addition, we can see the poses where the 3D scans were taken from. This map allocates 44732 blocks. Once the map has been generated, the eRMSM algorithm is executed. With an implementation of the algorithm optimized for a 2.4GHz quad-core processor, it is possible to identify 12 planes in 7.8 seconds.

In figure 8(b) we can see the corridor map where the blocks that match any detected plane are depicted using different colors. The largest planes in this map match about 1650 blocks. Using the eRANSAC test it would be necessary to generate 1132 CP (see equation 7) to accept the first plane with a probability greater than 0.9. Using our probability estimation, the first plane hypothesis is confirmed after generating 50 CP in 600ms. At the same scenary, we can see how the different steps of a stair are intified by the system (see figure 9). It is important that the lower steps are detected better than the upper steps. The reason is that upper steps are parallel to laser

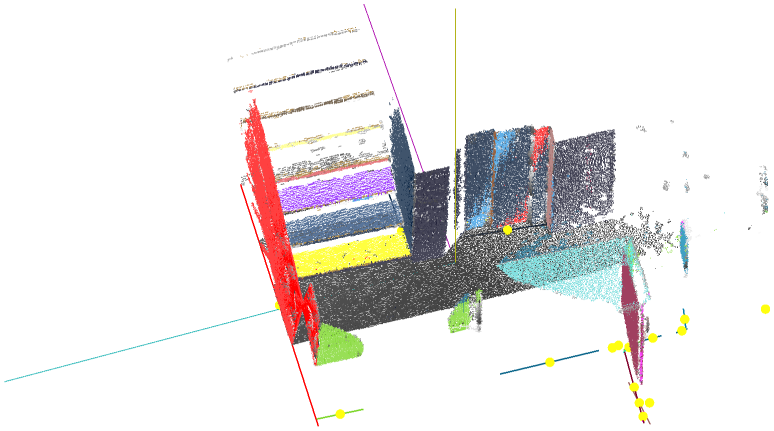
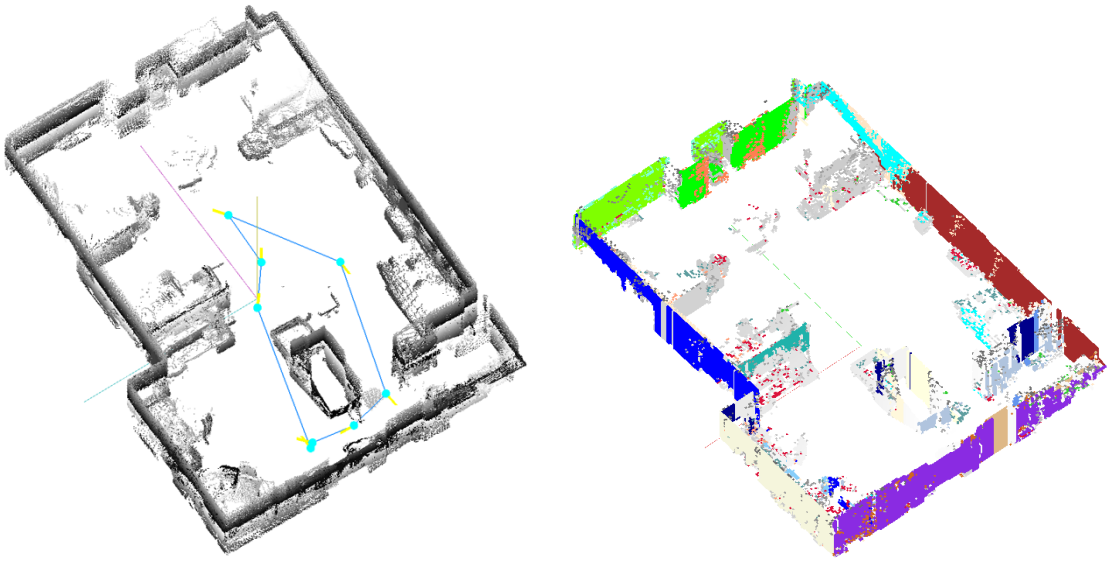


Fig. 9. An example of stairs detection in the middle of the corridor depicted in figure 6(a). The bottom steps are closer to perpendicular to the laser beams than the top steps, and thus they are detected better.



(a) (b)

Fig. 10. (a) Upper oblique view of ML maps generated from 8 3D scans taken at different poses in the Robotic's Laboratory. The line represents the robot path and the points over the line are the poses at which the scans were taken. (b) Planes detected in the Robotics Laboratory's map. Grey zones represent blocks that do not match any plane. Each color represents different planes.

beams, so the sensor does not receive an echo from these steps.

Figure 6(b) shows a new test scene. In this case, the scenario is a laboratory $8.3\mathrm{m}$ wide and $11.4\mathrm{m}$ long. Figure 10(a) shows a map generated using a cell size of $20\mathrm{mm}$. This map collects 196385 blocks. Fourteen different planes were detected in 4.6 seconds. As in the previous test, different colors in figure 10(b) correspond to blocks that match different detected planes.

ML maps, as we have generated them, easily allow the joining or fusion of different partial maps of adjacent spaces. The laboratory shown in figure 6(b) and the corridor of figure 6(a) are contiguous rooms in the same building. Both spaces were independently mapped using our approach, with the results depicted in figure 10(a) and 8(a). We have been able to generate a single map from the two data sets after the poses were corrected using the SLAM 6D software. We can see the resulting map in figure 11.

VII. Conclusions

This paper has described an approach to building compact 3D maps of indoor environments based on multilevel surface maps. This kind of space representation allows us to describe the scene with detail and balances spatial resolution and memory cost appropriately. These multilevel maps are easily scalable and versatile enough to provide sophisticated spatial information without having to rely on low level data, i.e. clouds of laser data points.

In addition, an efficient algorithm for detecting planes using the multilevel surface maps (eRSMS algorithm) has

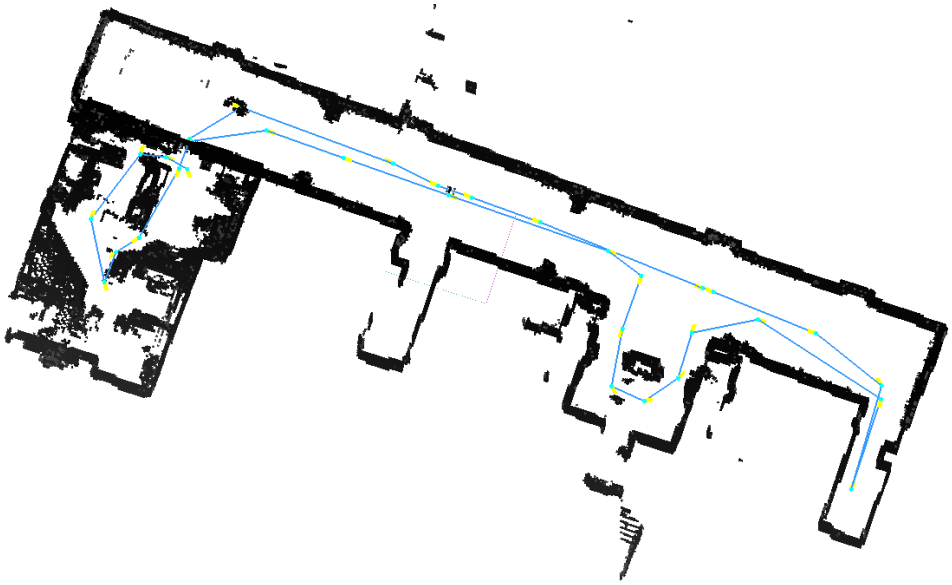


Fig. 11. Single map of the corridor and the Robotics Laboratory after merging the 3D scans independently taken at both scenarios.

been proposed. A key feature of the eRSMS algorithm that distinguishes it from the original eRANSAC algorithm is that it does not need to generate a high number of hypotheses in order to identify candidate planes with high probability. Moreover, eRSMS is easily parallelizable, an attractive feature that may be exploited on multicore processors.

While the system described in this paper has proved reliable, there is considerable room for improvement. Future work will be directed towards alleviating the off-line 6D SLAM pre-processing and the associated computational cost by using geometrical features instead of the scan point clouds. Moreover, the multilevel maps offer interesting possibilities to attempt to label an indoor space semantically.

VIII. Acknowledgements

The authors would like to thank Heather Adams, ULPGC, for checking the English version of this manuscript.

References

[1] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The SLAM problem: a survey. In *Proceeding of the 2008 conference on Artificial Intelligence Research and Development*, pages 363–371, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[2] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[3] Andreas Nüchter, Kai Lingemann, and Joachim Hertzberg. 6D SLAM with cached K-D tree search. In *RA'07: Proceedings of the 13th IASTED International Conference on Robotics and Applications*, pages 101–106, Anaheim, CA, USA, 2007. ACTA Press.

[4] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[5] Julian Ryde and Michael Brünig. Non-cubic occupied voxel lists for robot maps. In *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 4771–4776, Piscataway, NJ, USA, 2009. IEEE Press.

[6] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian process modeling of large-scale terrain. *J. Field Robot.*, 26(10):812–840, 2009.

[7] Cesar Rivadeneyra, Isaac Miller, Jonathan R. Schoenberg, and Mark Campbell. Probabilistic estimation of multi-level terrain maps. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 3709–3714, Piscataway, NJ, USA, 2009. IEEE Press.

[8] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2829–2834, Piscataway, NJ, USA, 2009. IEEE Press.

[9] Kaustubh Pathak, Narunas Vaskevicius, and Andreas Birk. Revisiting uncertainty analysis for optimum planes extracted from 3D range sensor point-clouds. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2035–2040, Piscataway, NJ, USA, 2009. IEEE Press.

[10] GuruPrasad M. Hegde and Cang Ye. Extraction of planar features from swissranger sr-3000 range images by a clustering method using normalized cuts. In *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 4034–4039, Piscataway, NJ, USA, 2009. IEEE Press.

[11] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[12] Mariano Martin Nevado, Jaime Gomez Garcia-Bermejo, and Eduardo Zalama Casanova. Obtaining 3d models of indoor environments with a mobile robot by estimating local surface directions. In *Robotics and Autonomous Systems*, pages 131–143, 2004.

[13] Martin Heracles, Bram Bolder, and Christian Goerick. Fast detection of arbitrary planar surfaces from unreliable 3D data. In *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 5717–5724, Piscataway, NJ, USA, 2009. IEEE Press.

[14] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech*, 23:215–221, 1955.

[15] Andreas Nüchter and K. Lingemann. 6D Simultaneous Localization and Mapping, 2010. http://slam6d.sourceforge.net.

[16] I.T. Jolliffe. *Principal Component Analysis*. Springer, New York, NY, 1986.