sensors

MDPI

Article

Multibeam 3D Underwater SLAM with Probabilistic Registration

Albert Palomer *, Pere Ridao and David Ribas

Vicorob Research Institute, Universitat de Girona, c/Pic de Peguera 13-Parc Científic i Tecnològic de la UdG-CIRS Building, Girona l17003, Spain; pere@eia.udg.edu (P.R.); dribas@udg.edu (D.R.)
* Correspondence: apalomer@eia.udg.edu; Tel.: +34-972-419-651

Academic Editor: Jaime Lloret Mauri
Received: 15 January 2016; Accepted: 14 April 2016; Published: 20 April 2016

Abstract: This paper describes a pose-based underwater 3D Simultaneous Localization and Mapping (SLAM) using a multibeam echosounder to produce high consistency underwater maps. The proposed algorithm compounds swath profiles of the seafloor with dead reckoning localization to build surface patches (*i.e.*, point clouds). An Iterative Closest Point (ICP) with a probabilistic implementation is then used to register the point clouds, taking into account their uncertainties. The registration process is divided in two steps: (1) point-to-point association for coarse registration and (2) point-to-plane association for fine registration. The point clouds of the surfaces to be registered are sub-sampled in order to decrease both the computation time and also the potential of falling into local minima during the registration. In addition, a heuristic is used to decrease the complexity of the association step of the ICP from $O(n^2)$ to $O(n)$. The performance of the SLAM framework is tested using two real world datasets: First, a 2.5D bathymetric dataset obtained with the usual down-looking multibeam sonar configuration, and second, a full 3D underwater dataset acquired with a multibeam sonar mounted on a pan and tilt unit.

Keywords: AUV; multibeam; SLAM; 3D; bathymetry

1. Introduction

For Autonomous Underwater Vehicles (AUVs), addressing the navigation and mapping problems is crucial to achieve a fully operational status. Dead reckoning navigation systems suffer from an unbounded drift error, regardless of using high-end Internal Navigation Systems (INS) [1]. To avoid this, such systems are commonly aided with absolute positioning fixes. Using the measurements from a Global Positioning System (GPS) receiver is a typical solution during operations taking place on the surface. When the vehicle is submerged, Long Base Line (LBL) systems [2] can be used for the same purpose, although complex calibration of the acoustic beacon network is required prior to its operation. Using single beacon/transponder methods may reduce the calibration burden [3,4] or even eliminate it, at the cost of a reduced accuracy, when inverted LBL [5] or Ultra Short Base Line (USBL) [6] systems are used instead.

All those methods share the limitation of confining the robot operation to the area of coverage of the system. Terrain-based navigation (TBN) methods [7] can mitigate this limitation when an *a priori* Digital Terrain Map (DTM) is available on the target area. However, for an underwater vehicle to become truly autonomous, it should be able to localize itself using only on-board sensors and without the help of any external infrastructure. The Simultaneous Localization And Mapping (SLAM) concept aims to achieve that. Although more than 20 years of research have provided different approaches to solve the SLAM problem, mostly in land mobile robotics [8], there are still few solutions for underwater use, mainly due to the sensing limitations imposed by the medium and the complexity of the environment.

Underwater SLAM can be divided in two main categories: sonar and vision based SLAM. Although vision sensors may suffer from poor visibility in turbid waters, they provide fast refresh rates and high-resolution data at a fraction of the cost of a sonar sensor. Several noteworthy examples of underwater visual SLAM have been presented during recent years [9–12]. On the other hand, sonar sensors can work in bad visibility conditions, penetrating further (10–150 m) because of the low attenuation of sound in water. However, the refresh rate and resolution are medium to low and are generally expensive. Although the number of underwater SLAM examples using sonar is still reduced, they are promising.

Regarding imaging sonar mosaicking, [13] presented a feature-based registration method for two-dimensional forward-looking sonar images, while [14] developed a Fourier-based registration method to build large-scale mosaics. Moreover, several feature-based methods have been reported using: (1) point features extracted from mechanically/electronically scanned imaging sonars [15–17] or using a synthetic aperture imaging sonar [18]; and (2) line features extracted from a Mechanical Scanning Imaging Sonar (MSIS) in a man-made environment [19]. However, it is extremely difficult to extract features robustly in a natural underwater environment. Therefore, some researchers have focused on using featureless methods such as scan matching or occupancy grids. The work presented in [20] proposed a SLAM algorithm using a Particle Filter (PF) and range measurements from multiple pencil-beam sonars to generate an occupancy grid of a sinkhole. The method was time and computationally efficient because of the use of an octree structure to represent the environment. Although bathymetric (elevation 2.5D) maps are commonly used in the context of TBN, there have been few studies reporting successful SLAM implementations using bathymetric maps generated with data from a multibeam profiler. The pioneering work in [21] used cross correlation and Iterative Closest Point (ICP) for coarse and fine registration of bathymetric surfaces. More recently, [22] presented the bathymetric distributed particle SLAM (BPSLAM), an algorithm based on the distributed particle SLAM (DPSLAM) [23], which used a PF similar to the one proposed in [20] but representing the environment as a bathymetric map distributed across the ancestry of a given particle. It is worth mentioning that those methods were specifically designed for 2.5D elevation data, and, therefore, they are not suited for full 3D underwater environments.

This paper presents the extension to 3D of the work previously presented in [24]. The registration algorithm is a 3D-capable evolution of the 2D MSIS probabilistic Iterative Correspondence (MSISpIC) algorithm [25], which has been already applied to 2D SLAM in underwater man-made [26,27] and natural [28] environments. Our method is similar to the previous work of [21] but takes advantage of recent results obtained using the probabilistic ICP algorithms mentioned above, which are better suited to dealing with the uncertainty inherent in sonar data. Moreover, our method is not restricted to using solely 2.5D bathymetric data, and, hence, new results obtained with full 3D data are also reported here.

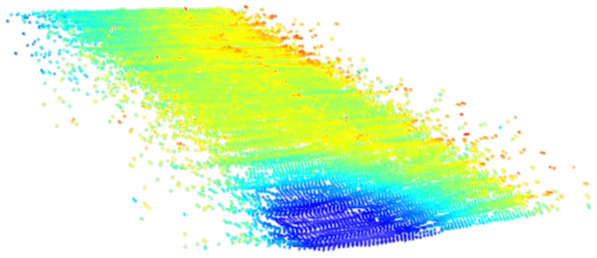
The 3D underwater SLAM framework presented here corrects the robot trajectory in order to produce high consistency underwater maps. The algorithm, like other state of the art SLAM techniques [21], divides the mission into a set of submaps, or surface patches, created by combining multibeam data and an estimate of the navigation until certain criteria are fulfilled. Every time a submap is created, possible overlaps with other existing patches are checked to look for loop closures. If any are found, the registration process takes place between the patches in order to refine the robot navigation. One of the novelties of the proposed method is the implementation of a two-step Probabilistic ICP (pICP) with point-to-point and point-to-plane for rough and fine registration, respectively. The improved registration method also incorporates a point cloud subsampling strategy to decrease the number of involved points as well as a novel method to decrease the complexity in the association step of the pICP from $O(n^2)$ to $O(n)$.

The rest of the paper is organized as follows. First, Section 2 focuses on the submap creation. In Section 3, the registration algorithm is explained, followed by the SLAM algorithm in Section 4.

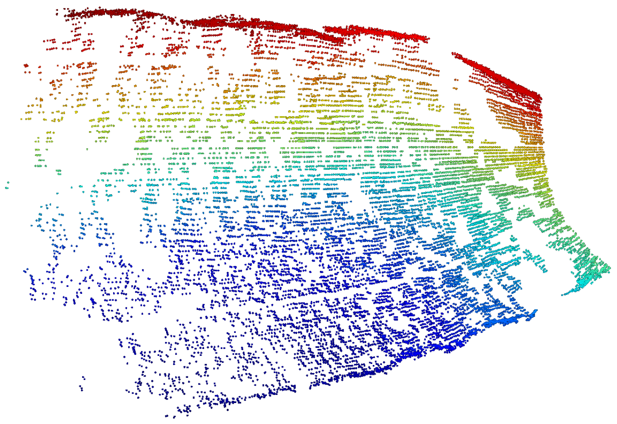
Section 5 presents the experiments and results, and, finally, Section 6 presents the conclusions and future work.

2. Submap Creation

For bathymetric mapping, multibeam sonars are generally fixed to the vehicle so that the 2D swath profiles are generated perpendicular to the surge direction. In that way, 2.5D surfaces are built by composing the multibeam data with the displacement of the vehicle. Alternatively, more complex environments can be inspected by sweeping a multibeam sonar mounted on a pan and tilt unit, so it is the rotation of the sonar head, and not solely the vehicle motion, that leads to the coverage of the surfaces. The point clouds resulting from the collection of multibeam data (Figure 1), along with other information such as boundaries or position with respect to the world, are what we refer to in this work as patches or submaps. This section describes the process of building these submaps during a mission.



(**a**) Fixed multibeam



(**b**) Sweeping multibeam

Figure 1. Two different submaps colored according to depth. In (**a**), the multibeam was mounted in a fixed downward-looking configuration, typically from bathymetric mapping; In (**b**), the sonar head was mounted on a pan and tilt unit and swept vertically to cover a portion of steep terrain.

2.1. Dead Reckoning

To be able to construct the submaps, regardless of whether the sonar is being swept or mounted on a fixed position, it is necessary to estimate the AUV position at the time each multibeam reading was acquired. As will be later detailed in Section 2.2, our procedure uses the relative displacements made by the vehicle between consecutive multibeam swaths to compound the point clouds. Moreover, given the probabilistic nature of the proposed registration algorithm, it is also necessary to estimate how the uncertainty evolves during these motions. To obtain this information, an Extended Kalman Filter (EKF) is used.

The state vector of the filter (see Equations (1) and (2)) contains 12 elements representing the current six Degrees of Freedom (DoF) vehicle position and velocity, as well as two more elements corresponding with the stored x and y position of the vehicle at the time when the last multibeam reading was obtained (x_{mb}, y_{mb}):

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r & x_{mb} & y_{mb} \end{bmatrix}_k^T \tag{1}$$

$$\boldsymbol{P}_{\boldsymbol{x}_k} = E\left([\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k] \, [\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k]^T \right)^T \tag{2}$$

A constant velocity kinematic model is used for prediction of the vehicle states, while those regarding the stored previous vehicle position are assumed static. In the correction stage, updates are performed asynchronously with the measurements coming from an Attitude and Heading Reference System (AHRS), a Doppler Velocity Log (DVL), and a pressure sensor. The filter iterates normally until a new multibeam reading is received. When this occurs, one last prediction is made to get an updated estimation of the vehicle's position before calculating $\boldsymbol{o}_k = N(\hat{\boldsymbol{o}}_k, \boldsymbol{P}_{\boldsymbol{o}_k})$, a new vector containing the displacement executed by the vehicle in the horizontal plane during the period of time between the current and the previous multibeam readings, as well as the z position and orientation of the vehicle at the current time:

$$\hat{\boldsymbol{o}}_k = \begin{bmatrix} x - x_{mb}, & y - y_{mb}, & z, & \phi, & \theta, & \psi \end{bmatrix}_k^T \tag{3}$$

$$\boldsymbol{P}_{\boldsymbol{o}_k} = \boldsymbol{J}_o \boldsymbol{P}_{\boldsymbol{x}_k} \boldsymbol{J}_o^T; \;\; \boldsymbol{J}_o = \begin{bmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times4} & \boldsymbol{0}_{2\times6} & -\boldsymbol{I}_{2\times2} \\ \boldsymbol{0}_{4\times2} & \boldsymbol{I}_{4\times4} & \boldsymbol{0}_{4\times6} & \boldsymbol{0}_{4\times2} \end{bmatrix} \tag{4}$$

Note that the two first elements of $\boldsymbol{o}_k$ correspond to incremental values, while the other four are absolute with respect to the base reference frame used for the dead reckoning. The calculation of those increments is motivated by the cumulative drift that affects the motion in the horizontal plane. Since those states are only estimated indirectly by the velocity measurements from the DVL, the uncertainty in the xy position grows without bound. As will be introduced in the following section, working with those increments allows for a better distribution of the uncertainties within the point cloud. On the other hand, the remaining states in $\boldsymbol{o}_k$ are observed directly by other sensors (the z position is observed by the pressure sensor, and the orientation by the AHRS), and therefore their uncertainties are bounded.

Once $\boldsymbol{o}_k$ has been calculated, it is stored until the current submap is finalized. To continue with the execution of the dead reckoning filter, and to keep track of the displacements from the current position to that of the next multibeam measurement, it is necessary to replace the last two elements of the state vector (x_{mb}, y_{mb}) with the current position of the vehicle x and y:

$$\hat{\boldsymbol{x}}_k^* = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r & x & y \end{bmatrix}_k^T \tag{5}$$

$$\boldsymbol{P}_{\boldsymbol{x}_k^*} = \boldsymbol{J}_o^* \boldsymbol{P}_{\boldsymbol{x}_k} \boldsymbol{J}_o^{*T}; \;\; \boldsymbol{J}_o^* = \begin{bmatrix} \boldsymbol{I}_{12\times12} & \boldsymbol{0}_{12\times2} \\ \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times12} \end{bmatrix} \tag{6}$$

Given that, the execution of the filter can continue by replicating the procedure we have just described.

2.2. Submap Forming

During the execution of the mission, the information required for the generation of the patches is stored in a temporal data structure $\boldsymbol{S}_{temp}$:

$$\boldsymbol{S}_{temp} = \left\{ \boldsymbol{O} \quad \boldsymbol{M} \quad \boldsymbol{R} \right\} \tag{7}$$

where $\boldsymbol{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_n | \boldsymbol{o}_i = N(\hat{\boldsymbol{o}}_i, \boldsymbol{P}_{\boldsymbol{o}_i})\}$ is the set of displacements and positions as computed in Equations (3) and (4), while $\boldsymbol{M} = \{\boldsymbol{m}_1, ..., \boldsymbol{m}_n\}$, with $\boldsymbol{m}_i = \{\delta_1, ..., \delta_m | \delta_i = N(\hat{\delta}_i, \boldsymbol{P}_{\delta_i})\}$, is the set of all the multibeam swaths $\boldsymbol{m}$, each one containing the corresponding polar range measurements δ. Finally, $\boldsymbol{R} = \{\boldsymbol{r}_1, ..., \boldsymbol{r}_n | \boldsymbol{r}_i = N(\hat{\boldsymbol{r}}_i, \boldsymbol{P}_{r_i})\}$ is the set of transformations required to represent the multibeam data with respect to the vehicle frame. This is particularly relevant in the case of a multibeam sonar mounted on a pan and tilt unit, since the transformations will change continuously because of the sweeping motion.

When the amount of accumulated data is deemed sufficient (see the conditions below), the current patch is closed and the contents of $\boldsymbol{S}_{temp}$ are used to generate the point cloud and other information that will be necessary later during the registration process. In addition, the position of the recently terminated patch is stored in the state vector of the pose-based EKF in charge of the SLAM process (see Section 4). Before beginning a new patch, the $\boldsymbol{S}_{temp}$ is reset to store a new batch of data.

The criteria to close a patch depend on which scenario we are dealing with. If the sonar is scanning a tri-dimensionally rich environment by means of a pan and tilt unit, each complete sweep is taken as an independent submap because, unless a very fast vehicle is used, successive scans will re-visit the same area, which only contributes to increasing the number of points without incorporating significant new information. On the other hand, the situation with typical bathymetric survey missions where the multibeam is fixed on the vehicle is substantially different. Scanned areas are generally not re-visited (not in the same transect), and the seabed is often scarce in features, which may make the successful matching of surface patches difficult. In this case, a combination of three criteria is used to determine when to close a patch:

- **Minimum size:** A minimum size is defined to avoid handling a large number of tiny patches augmenting unnecessarily the length of the SLAM state vector and reducing the overlapping.
- **Maximum size:** The maximum size is bounded to avoid handling huge patches with a high uncertainty in the surface points due to the accumulated dead reckoning error.
- **Normal occupancy:** The surface relief is analyzed to determine when the patch is rich enough to be successfully matched. The procedure basically consists in finding surface normals for each point on the cloud and representing their parametrization on a histogram. If the histogram is sufficiently occupied, the submap is closed.

Once sufficient data has been acquired and the submap is closed, all the stored data is processed to generate the point cloud and the information required for the potential registration with other submaps. In [21], the reference frame for each submap was defined as the position of the robot when the patch was started. Here, the point cloud is generated with respect to a new reference frame $\mathbb{I}$, which is placed on top of the central position of the trajectory executed during the creation of the patch, but oriented like the base frame $\mathbb{B}$ used for the dead reckoning navigation. By placing this frame in the center of the submap, the uncertainty of the points grows from the center of the patch to the edges (see Figure 2b) instead of growing from the beginning to the end (see Figure 2a). This gives a more convenient distribution of the uncertainty in the point cloud which improves the registration [29].

The process of generating the point cloud begins by selecting the central position that will be associated to $\mathbb{I}$, and that will be referenced hereafter with the mp subindex. Then, the $\boldsymbol{q}_k$ vector relating a given k position in which a multibeam reading was acquired, and the $\mathbb{I}$ reference frame can be computed from the corresponding $\boldsymbol{o}_k$ and $\boldsymbol{o}_{mp}$ (both pertaining to $\boldsymbol{O}$ and stored in $\boldsymbol{S}_{temp}$) as:

$$\hat{\boldsymbol{q}}_k = \begin{cases} \begin{bmatrix} x_{q_{k-1}} + x_{o_k}, & y_{q_{k-1}} + y_{o_k}, & z_{o_k} - z_{o_{mp}}, & \phi_{o_k}, & \theta_{o_k}, & \psi_{o_k} \end{bmatrix}^T & k > mp \\ \begin{bmatrix} 0, & 0, & 0, & \phi_{o_k}, & \theta_{o_k}, & \psi_{o_k} \end{bmatrix}^T & k = mp \\ \begin{bmatrix} x_{q_{k+1}} - x_{o_{k+1}}, & y_{q_{k+1}} - y_{o_{k+1}}, & z_{o_k} - z_{o_{mp}}, & \phi_{o_k}, & \theta_{o_k}, & \psi_{o_k} \end{bmatrix}^T & k < mp \end{cases} \quad (8)$$

where $x_{\boldsymbol{a}}$ should be read here as the element x contained in the vector $\boldsymbol{a}$. Note that for computing $\boldsymbol{q}_k$, the vectors $\boldsymbol{q}_{k-1}$ (if $k > mp$) or $\boldsymbol{q}_{k+1}$ (if $k < mp$) also need to be known. This means that the calculation needs to be done sequentially starting by the mp position and then moving towards both ends of the submap (1 and n). The uncertainty of $\boldsymbol{q}_k$ is then computed as:

$$\boldsymbol{P}_{\boldsymbol{q}_k} = \begin{cases} \boldsymbol{J}_1 \boldsymbol{P}_{\boldsymbol{q}_{k-1}} \boldsymbol{J}_1^T + \boldsymbol{J}_2 \boldsymbol{P}_{\boldsymbol{o}_k} \boldsymbol{J}_2^T + \boldsymbol{J}_3 \boldsymbol{P}_{\boldsymbol{o}_{mp}} \boldsymbol{J}_3^T & k > mp \\ \boldsymbol{J}_4 \boldsymbol{P}_{\boldsymbol{o}_k} \boldsymbol{J}_4^T & k = mp \\ \boldsymbol{J}_1 \boldsymbol{P}_{\boldsymbol{q}_{k+1}} \boldsymbol{J}_1^T + \boldsymbol{J}_5 \boldsymbol{P}_{\boldsymbol{o}_{k+1}} \boldsymbol{J}_5^T + \boldsymbol{J}_3 \boldsymbol{P}_{\boldsymbol{o}_{mp}} \boldsymbol{J}_3^T + \boldsymbol{J}_4 P_{\boldsymbol{o}_k} \boldsymbol{J}_4^T & k < mp \end{cases} \tag{9}$$

being $\boldsymbol{J}_j$ the Jacobians of the function:

$$\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times4} \\ \boldsymbol{0}_{4\times2} & \boldsymbol{0}_{4\times4} \end{bmatrix} \tag{10}$$

$$\boldsymbol{J}_2 = \boldsymbol{I}_{6\times6} \tag{11}$$

$$\boldsymbol{J}_3 = \begin{bmatrix} \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times1} & \boldsymbol{0}_{2\times3} \\ \boldsymbol{0}_{1\times2} & -1 & \boldsymbol{0}_{1\times3} \\ \boldsymbol{0}_{3\times2} & \boldsymbol{0}_{3\times1} & \boldsymbol{0}_{3\times3} \end{bmatrix} \tag{12}$$

$$\boldsymbol{J}_4 = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{13}$$

$$\boldsymbol{J}_5 = -\boldsymbol{J}_1 \tag{14}$$

It is worth noting that $\boldsymbol{q}_k$ is composed of both relative (first three elements representing a displacement) and absolute (last three elements being the orientation) measurements. Assuming that correlations in attitude estimates are negligible, computing the relative increment of the orientation would end up adding uncertainty in these 3 DoF artificially. This would apply also to the z displacement. However, we observed that if the same approach is taken in this DoF, the lever-arm effect in the registration process (the depth is referenced to the water surface) makes it much more prone to error. Therefore, we decided to reference the z position to the actual depth of the AUV regardless of the increment of uncertainty in order to make the registration process more stable.

With all the $\boldsymbol{q}_k$ computed, the point cloud can now be generated. The first step is to transform all the polar range measurements $\boldsymbol{\delta}_i = N(\hat{\boldsymbol{\delta}}_i, \boldsymbol{P}_{\boldsymbol{\delta}_i})$ which are represented in the sensor frame to that of the vehicle using the $\boldsymbol{r}_k$ transformations stored in $\boldsymbol{R}$:

$$\hat{\boldsymbol{p}}_i^{\times} = \hat{\boldsymbol{r}}_k \oplus g(\hat{\boldsymbol{\delta}}_i) \tag{15}$$

$$\boldsymbol{P}_{\boldsymbol{p}_i^{\times}} = \boldsymbol{J}_{1\oplus} \boldsymbol{P}_{\boldsymbol{r}_k} \boldsymbol{J}_{1\oplus}^T + \boldsymbol{J}_{2\oplus} (\boldsymbol{J}_g \boldsymbol{P}_{\boldsymbol{\delta}} \boldsymbol{J}_g^T) \boldsymbol{J}_{2\oplus}^T \tag{16}$$

where $g(.)$ is the polar to Cartesian conversion function, $\boldsymbol{J}_g$ is its corresponding Jacobian and $\oplus$ is the compounding operator with Jacobians $\boldsymbol{J}_{1\oplus}$ and $\boldsymbol{J}_{2\oplus}$ as defined in [30]. With the point $\boldsymbol{p}_i^{\times}$ referenced to the vehicle frame and $\boldsymbol{q}_k$ being the vehicle position referenced to $\mathbb{I}$, we can calculate the position of a point $\boldsymbol{p}_i$ as:

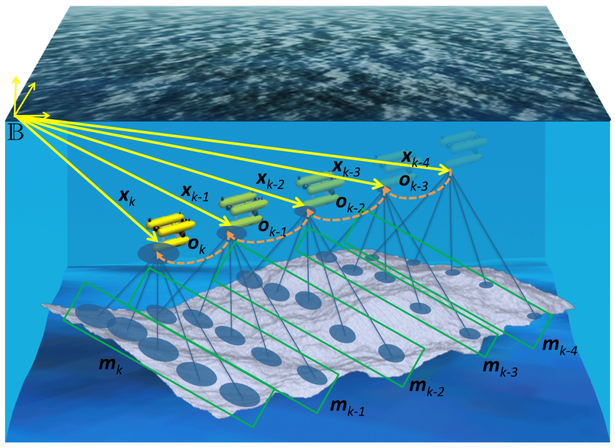
$$\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{q}}_k \oplus \hat{\boldsymbol{p}}_i^{\times} \tag{17}$$

$$\boldsymbol{P}_{\boldsymbol{p}_i} = \boldsymbol{J}_{1\oplus} \boldsymbol{P}_{\boldsymbol{q}_k} \boldsymbol{J}_{1\oplus}^T + \boldsymbol{J}_{2\oplus} \boldsymbol{P}_{\boldsymbol{p}_i^{\times}} \boldsymbol{J}_{2\oplus}^T \tag{18}$$

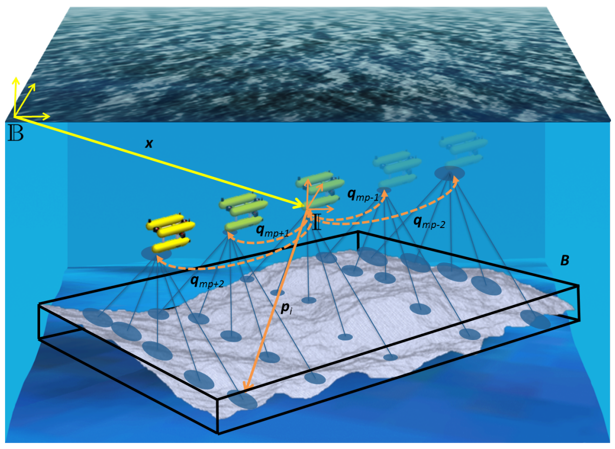
After all these calculations, the information regarding the patch is saved in a new structure $\boldsymbol{S}$ as:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{O} & \boldsymbol{W} & \boldsymbol{B} \end{bmatrix} \tag{19}$$

where $\boldsymbol{x} = N(\hat{\boldsymbol{x}}, \boldsymbol{P}_{\boldsymbol{x}})$ is the position of the frame $\mathbb{I}$ that will be used to anchor the submap in the state vector of the pose-based SLAM framework described in Section 4; $\boldsymbol{O}$ is the same set of transformations as in $\boldsymbol{S}_{temp}$; $\boldsymbol{W} = \{\boldsymbol{p}_1, ..., \boldsymbol{p}_n | \ \boldsymbol{p}_i = N(\hat{\boldsymbol{p}}_i, \boldsymbol{P}_{\boldsymbol{p}_i})\}$ is the set of points referenced to $\mathbb{I}$ that have been calculated and, finally, $\boldsymbol{B}$ is a volume containing all the points that pertain to the patch. On the horizontal plane, $\boldsymbol{B}$ is the polygon containing all the $\hat{\boldsymbol{p}}_i$ points, while, on the z direction, the boundary is defined by the minimum and maximum depth of all the points $\hat{\boldsymbol{p}}_i$ (see Figure 2).



(**a**) Dead Reckoning

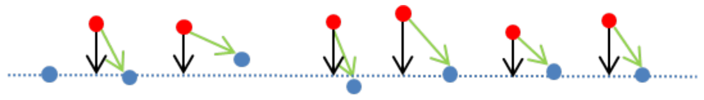


(**b**) Scan Forming

Figure 2. Example of the dead reckoning (**a**) and the scan forming (**b**).

3. Registration Algorithm

This section explains the procedure to register two submaps using probabilistic ICP. The inputs of the algorithm are a reference submap $\boldsymbol{S}_{ref}$, which has been already stored in the SLAM framework; a newly generated submap $\boldsymbol{S}_{new}$, and an initial guess of their relative displacement $\boldsymbol{q}_0 = N(\hat{\boldsymbol{q}}_0, \boldsymbol{P}_{q_0})$ obtained from the navigation. The algorithm uses a two-stage correction procedure. First, a point-to-point correction is performed to roughly align the two submaps (until their relative displacement in two consecutive iterations falls below a threshold), and then, a point-to-plane correction is executed to refine the result. Point-to-point association tends to produce undesired effects in the presence of small misalignments (see for instance the lateral displacement depicted in Figure 3a). This is because associated points do not necessarily correspond to the exact same spot in the original surface and therefore their arbitrary occurrence may prevail over the general shape of the surface. However, point-to-point association is powerful when large displacements are present (see Figure 3b). On the other hand, point-to-plane associations tend to be driven by the shape of the surface and hence, perform better in the presence of small misalignments (see Figure 3a), but may fail when dealing with large displacements (see Figure 3b). To complement their strengths and weaknesses, we combine both methods by using an error threshold which determines when to switch from one strategy to the other.


(**a**) Small displacement

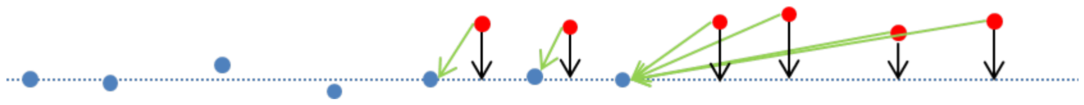

(**b**) Large displacement

Figure 3. Point-to-point and point-to-plane comparison in the presence of small displacement (**a**) and large displacement (**b**). In blue, the points of the reference scan with the plane, in red, the points of the new scan. Green arrows correspond to point-to-point association while black ones represent point-to-plane.

3.1. Point-to-Point Association

Given a certain point $\boldsymbol{n}_i = N(\hat{\boldsymbol{n}}_i, \boldsymbol{P}_{n_i})$ from the new patch $\boldsymbol{S}_{new}$, and a matching candidate $\boldsymbol{r}_j = N(\hat{\boldsymbol{r}}_j, \boldsymbol{P}_{r_j})$ from the reference surface $\boldsymbol{S}_{ref}$, both represented in Cartesian coordinates and referenced to their respective frames ($\mathbb{I}_{new}$ and $\mathbb{I}_{ref}$), the association error $\boldsymbol{e}_{ij} = N(\hat{\boldsymbol{e}}_{ij}, \boldsymbol{P}_{e_{ij}})$ can be defined as:

$$\hat{\boldsymbol{e}}_{ij} = \hat{\boldsymbol{r}}_j - \hat{\boldsymbol{q}}_0 \oplus \hat{\boldsymbol{n}}_i \tag{20}$$

$$\boldsymbol{P}_{ij} = \boldsymbol{P}_{r_j} + \boldsymbol{J}_{1\oplus} \boldsymbol{P}_{q_0} \boldsymbol{J}_{1\oplus}^T + \boldsymbol{J}_{2\oplus} \boldsymbol{P}_{n_i} \boldsymbol{J}_{2\oplus}^T \tag{21}$$

so the point-to-point association may be solved through a simple individual compatibility test over the corresponding Mahalanobis distance:

$$d^2 = \hat{\boldsymbol{e}}_{ij}^T \boldsymbol{P}_{ij}^{-1} \hat{\boldsymbol{e}}_{ij} < \chi^2_{d,\alpha} \tag{22}$$

All the points individually compatible with $\boldsymbol{n}_i$ form the set $\boldsymbol{A}_i$. From this set, the one with smaller Mahalanobis distance is chosen to be associated with $\boldsymbol{n}_i$.

3.2. Point-to-Plane Association

At the second stage, the metric changes and the point-to-plane distance is used instead. Now, the set of compatible points $\boldsymbol{A}_i$ is used to estimate a local plane $\boldsymbol{\Pi}(\boldsymbol{\nu}_i, d_i)$ whose equations are given by $\boldsymbol{\nu}_i^T \boldsymbol{x} - d_i = 0$, being d_i the plane distance to the origin and $\boldsymbol{\nu}_i$ its normal vector. Because of the probabilistic nature of our algorithm, we are interested not only in the plane parameters but also in their uncertainty. An iterative method is reported in [31] for this purpose, being too computationally expensive for our case. In [32], the authors use a two-step minimization method for estimating: (1) the plane using region growing algorithms and (2) its uncertainty. Finally, in [33], the error of a set of samples is minimized using the uncertainty related to the range of the sensor by means of a weighted Principal Component Analysis (PCA). This last method is the one which best fits our requirements because of its reduced computational complexity, and also because of its nature, since it does not search for the points forming the plane, but fits the plane among the given points.

Given a plane $\boldsymbol{\Pi}(\boldsymbol{\nu}, d)$, whose equation is $\boldsymbol{\nu}^T \boldsymbol{x} = d$, the likelihood of observing a plane point $\boldsymbol{r}_j \in \boldsymbol{A}_i$ is given by:

$$p(\boldsymbol{r}_j | \boldsymbol{\nu}, d) = \frac{1}{\sqrt{2\pi} |\boldsymbol{P}_{r_j}|} \exp\left((\boldsymbol{\nu}^T \boldsymbol{r}_j - d) * \boldsymbol{\nu})^T \boldsymbol{P}_{r_j}^{-1} ((\boldsymbol{\nu} \boldsymbol{r}_j - d) * \boldsymbol{\nu} \right) \tag{23}$$

The objective here is to maximize the sum of the log-likelihood of the previous equation. The problem cannot be solved in a simple way since the error of the uncertainty depends twice on the normal $\boldsymbol{\nu}$. To solve the problem in an efficient way, it was necessary to approximate the uncertainty by the trace of $\boldsymbol{P}_{\boldsymbol{r}_j}$: $\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)$. In this way, the error ellipsoid is approximated to a sphere, and it is possible to solve the equation analytically and as efficiently as in [33] (please refer to this work for a more extended derivation).

The log-likelihood that we want to maximize, ignoring constants, is the approximate least squares problem:

$$\ell = \underset{\boldsymbol{\nu},d}{\mathrm{argmax}} \quad -\frac{1}{2}\sum_{i=1}^{N}\frac{(\boldsymbol{\nu}^T\boldsymbol{r}_j - d)^2}{\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)^2} \tag{24}$$

with Lagrangian

$$\mathcal{L} = -\frac{1}{2}\sum_{i=1}^{N}\frac{(\boldsymbol{\nu}^T\boldsymbol{r}_j - d)^2}{\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)^2} - \lambda(\boldsymbol{\nu}^T\boldsymbol{\nu} - 1) \tag{25}$$

Setting $\frac{\partial\mathcal{L}}{\partial d} = 0$, we find the solution

$$d_\star = \boldsymbol{\nu}_\star^T\boldsymbol{p}_\mu, \qquad \text{with} \qquad \boldsymbol{p}_\mu = \left(\sum_{i=1}^{N}\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)^{-2}\boldsymbol{r}_j\right)\left(\sum_{i=1}^{N}\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)^{-2}\right)^{-1} \tag{26}$$

$\boldsymbol{p}_\mu$ being the weighted center of the set of points $\boldsymbol{A}_i$. Finally,

$$\boldsymbol{\nu}_\star = \underset{\boldsymbol{\nu}}{\mathrm{argmin}} \quad \boldsymbol{\nu}^T\left(\sum_{i=1}^{N}\frac{(\boldsymbol{r}_j - \boldsymbol{p}_\mu)(\boldsymbol{r}_j - \boldsymbol{p}_\mu)^T}{\mathrm{Tr}\left(\boldsymbol{P}_{\boldsymbol{r}_j}\right)^2}\right)\boldsymbol{\nu} \tag{27}$$

The minimizing normal $\boldsymbol{\nu}_\star$ is defined by the eigenvalues of the covariance matrix of the points as in the common weighted PCA method. The uncertainty of the estimator is found as:

$$\boldsymbol{P}_f = -\boldsymbol{H}^+ = \begin{pmatrix} \boldsymbol{P}_\nu & \boldsymbol{P}_\nu\boldsymbol{P}_d \\ \boldsymbol{P}_d\boldsymbol{P}_\nu & \boldsymbol{P}_d, \end{pmatrix} \tag{28}$$

where $\boldsymbol{H}$ is the Hessian of the Lagrangian in the optimal plane.

Given the point $\boldsymbol{n}_i$ and the plane $\boldsymbol{\Pi}_i(\boldsymbol{\nu}_i, d_i)$ estimated from all the compatible points in $\boldsymbol{A}_i$, the point $\boldsymbol{a}_i$ is defined as the orthogonal projection of $\boldsymbol{n}_i$ over the plane $\boldsymbol{\Pi}_i(\boldsymbol{\nu}_i, d_i)$:

$$\hat{\boldsymbol{a}}_i = \hat{\boldsymbol{q}}_0 \oplus \hat{\boldsymbol{n}}_i - ((\hat{\boldsymbol{q}}_0 \oplus \hat{\boldsymbol{n}}_i)^T\hat{\boldsymbol{\nu}}_i - \hat{d}_i)\hat{\boldsymbol{\nu}}_i \tag{29}$$

$$P_{\boldsymbol{a}_i} = \frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{q}_0}\boldsymbol{P}_{\boldsymbol{q}_0}\frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{q}_0}^T + \frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{n}_i}\boldsymbol{P}_{\boldsymbol{n}_i}\frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{n}_i}^T + \frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{\nu}_i}\boldsymbol{P}_{\boldsymbol{\nu}_i}\frac{\partial\boldsymbol{a}_i}{\partial\boldsymbol{\nu}_i}^T + \frac{\partial\boldsymbol{a}_i}{\partial d_i}\boldsymbol{P}_{d_i}\frac{\partial\boldsymbol{a}_i}{\partial d_i}^T \tag{30}$$

This new virtual point $\boldsymbol{a}_i$ is actually the point that will be associated with $\boldsymbol{n}_i$ to execute the new registration phase using the same point-to-point equations we already presented in Equations (20) and (21), but using $\boldsymbol{a}_i$ instead of $\boldsymbol{r}_j$.

3.3. Minimization

At the end of each association stage, a minimization process is executed to estimate the robot displacement $\boldsymbol{q}_{\min}$ that minimizes the addition of the Mahalanobis distance of the association error:

$$\boldsymbol{q}_{\min} = \underset{\boldsymbol{q}}{\mathrm{argmin}}\sum\{\boldsymbol{\xi}\boldsymbol{P}_{\boldsymbol{\xi}}^{-1}\boldsymbol{\xi}\} \tag{31}$$

$\boldsymbol{\xi}$ being a vector composed of all the $\hat{\boldsymbol{e}}_{ij}$ error vectors (see Equation (20)) after associating all the points (either virtual or real) and $\boldsymbol{P}_{\xi}$ the block diagonal matrix with their corresponding covariances $\boldsymbol{P}_{\boldsymbol{e}_{ij}}$ (Equation (21)). This minimization is done using weighted least squares:

$$\boldsymbol{q}_{\min} = [\boldsymbol{J}^T \boldsymbol{P}_{\xi}^{-1} \boldsymbol{J}]^{-1} \boldsymbol{J}^T \boldsymbol{P}_{\xi}^{-1} \boldsymbol{\xi} \tag{32}$$

$\boldsymbol{J}$ being the Jacobian matrix of the error function at the previous estimation evaluated in all the points.

3.4. Submap Simplification

Traditional ICP-based methods may encounter some problems in a scenario like the one depicted in Figure 4a, where two almost flat surfaces share a poorly visible feature. For instance, ICP tends to associate each point with its closest neighbour according to a particular metric. Because of that, it may be difficult to correctly associate the feature areas when the displacement is large (*i.e.*, they are far from each other, and the proximity of flat areas may lead to a local minimum). This particular issue will benefit from the proposed probabilistic ICP approach, since the uncertainty of the points should constrain the possible matching candidates to those compatible with the real accumulated error. For instance, uncertainties may be large in the horizontal plane, making it possible to match two distant features, but small in the z direction, so points in the flat areas will not be compatible with those in the features.

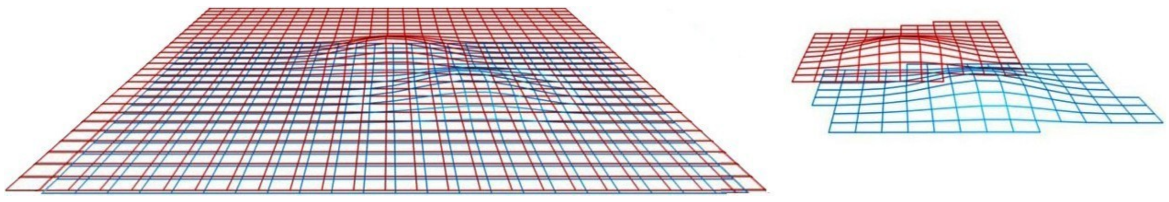
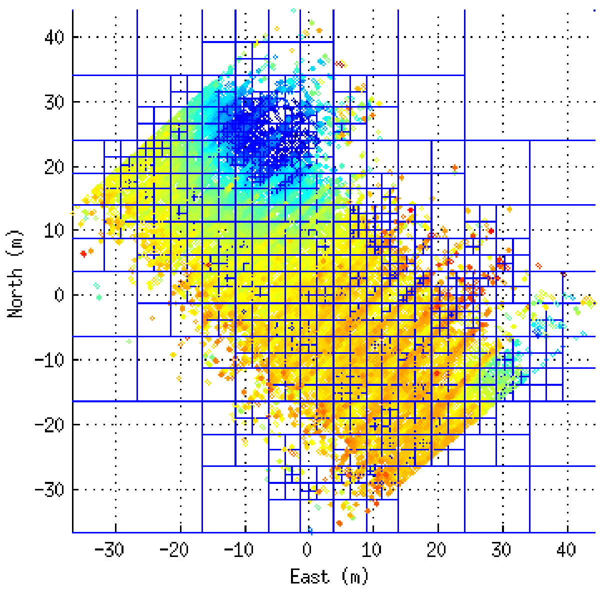

(**a**) (**b**)

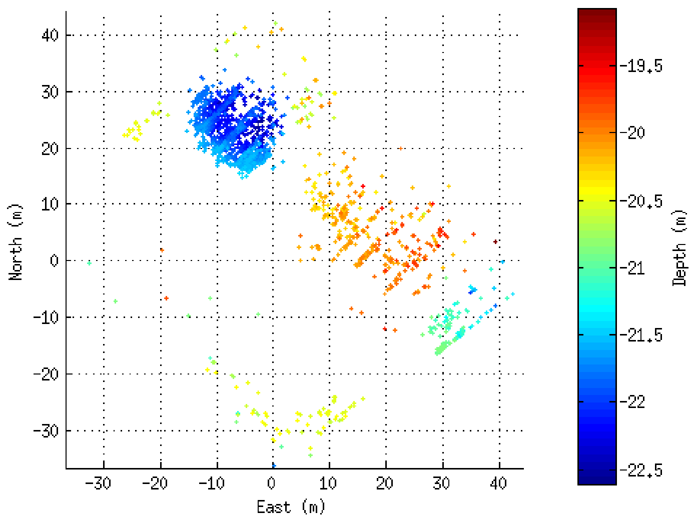
Figure 4. A 3D visual concept of the idea behind octree use: two surfaces to be matched (**a**); the same surfaces resampled (**b**).

Another inconvenience related to traditional ICP-based methods is the weak contribution of flat areas to the registration (they all look alike and their matching possibilities are high). Moreover, when few features are present in scenarios of large flat areas, the planar areas may prevail and lead to poor matching. ICP algorithms have better results when the associated points are significant (*i.e.*, distinguishable from each other). For that reason, a new sampling procedure to reduce the number of points in the cloud by removing the less informative ones is presented (see Figure 4b). Since the surface distribution is not available, the sampling procedure is performed using the discrete points. This resampling improves the odds of successful matchings, even when large displacements are present, as well as decreasing the computation time in the registration by drastically decreasing the number of points to be associated, thus increasing the performance of the algorithm.

The approach proposed here uses an octree structure to sample the scan in its most significant areas (*i.e.*, areas with rich relief). The subsampling algorithm works as follows: the point cloud is contained in a discretized tridimensional space structured as an octree. Using the points contained in each cell of the octree, a relief-based subsampling criteria is evaluated recursively, and if the condition is fulfilled, the cell is divided into eight subcells. After the subdivision process comes to an end, only one point is taken from each cell of the octree (see Figure 5). This makes areas with bigger (*i.e.*, not significant) cells contribute with fewer points than areas with smaller (*i.e.*, significant) cells. In [34], several different criteria were studied to drive the octree subdivision. Although some criteria were more suitable for specific types of environments, in this work, the *difference from principal plane* method has been selected given its overall performance in both 2.5D and 3D. This criteria basically dictates that a cell should be divided if the average distance between the points in the cell, and the best fitting plane of the cloud is higher than a given threshold. For a more detailed description, please refer to [34].



(**a**)



(**b**)

Figure 5. The contribution of octree in point resampling: octree construction, top view (**a**); points after resampling (**b**).

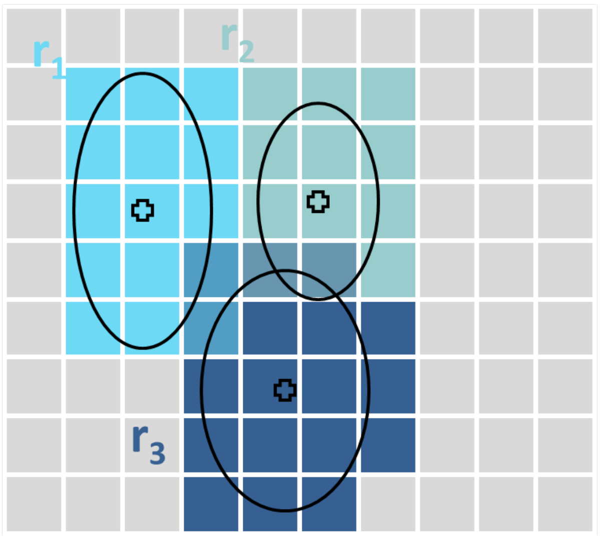
3.5. Association in Linear Time

The association step in ICP based methods has an $O(n^2)$ computational cost because it is necessary to compare each reference point $\boldsymbol{r}_j$ in S_{ref} against all the $\boldsymbol{n}_i$ points in S_{new} to compute their distances. Moreover, the probabilistic implementation of the ICP method requires several matrix operations, including an inversion, to calculate the uncertainty of the association of the points from the two point clouds. Hereafter, a new method for reducing complexity taking advantage of the uncertainty estimates of the points, which are already available, is proposed.

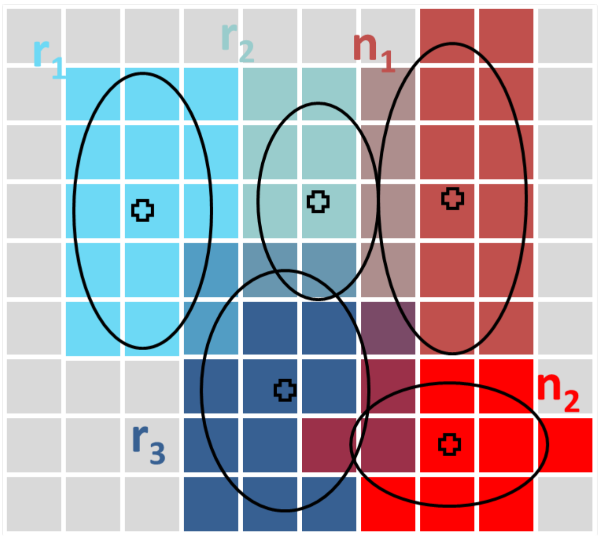
A probabilistic point $\boldsymbol{p}$ with uncertainty $\boldsymbol{P}$ can be represented graphically as an ellipsoid defined by a χ^2 distribution at a certain confidence level α and for d degrees of freedom (DoF):

$$\left\{(\boldsymbol{x}|(\boldsymbol{x}-\boldsymbol{p})^T\boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{p}) = \chi^2_{d,\alpha}\right\} \tag{33}$$

Given that, in our approach, a 3D grid is generated covering the two patches to be matched, and for each point in $\boldsymbol{S}_{ref}$, the cells falling inside its uncertainty ellipse are marked (see Figure 6a). During the association process, the same procedure is followed for each point in $\boldsymbol{S}_{new}$.



(**a**) Before Matching



(**b**) During Association

Figure 6. The figure shows how the support grid is used during the association step. First, the points in the reference scan (**blue**) are inserted into the cells using their uncertainty ellipses (**a**). Then, each point in the new scan (**red**) is also laid inside the grid (**b**). In this case, $\boldsymbol{n}_1$ overlaps with $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ while $\boldsymbol{n}_2$ overlaps only with $\boldsymbol{r}_3$. Moreover, $\boldsymbol{r}_1$ has no potential associations.

At this point, the following heuristic is applied: To check the compatibility of two points $\boldsymbol{n}_i$ and $\boldsymbol{r}_j$, we define their ellipsoids given a certain confidence level α. If the ellipsoids do not intersect with each other, the corresponding points are assumed not to be individually compatible. In other words, if:

$$\left\{(\boldsymbol{x}|(\boldsymbol{x}-\boldsymbol{n}_i)^T\boldsymbol{P}_{\boldsymbol{n}_i}^{-1}(\boldsymbol{x}-\boldsymbol{n}_i)=\chi^2_{d,\alpha}\right\}\cap \left\{(\boldsymbol{x}|(\boldsymbol{x}-\boldsymbol{r}_j)^T\boldsymbol{P}_{\boldsymbol{r}_j}^{-1}(\boldsymbol{x}-\boldsymbol{r}_j)=\chi^2_{d,\alpha}\right\}=\emptyset \tag{34}$$

then,

$$(\boldsymbol{n}_i-\boldsymbol{r}_j)(\boldsymbol{P}_{\boldsymbol{n}_i}+\boldsymbol{P}_{\boldsymbol{r}_j})^{-1}(\boldsymbol{n}_i-\boldsymbol{r}_j)^T>\chi^2_{d,\alpha} \tag{35}$$

Note that evaluating the compatibility in this way is still computationally expensive. However, in our method, the space occupied by the ellipses has been previously registered inside a grid, so it is possible to rapidly find the intersecting ellipsoids using a direct grid look-up (see Figure 6b). In other words, association candidates for a given point in the new scan can be easily identified by searching only among the cells occupied by its own ellipse, for tags denoting occupancy of those same cells by ellipses corresponding to points in the reference scan. In that way, candidates are directly determined, without a need to evaluate all the remaining points in the reference scan, and thus, the complexity is reduced to $O(n)$.

4. SLAM Algorithm

This section describes the EKF implementation of the pose-based SLAM framework in charge of optimizing the surface map.

Every time a submap is finished, the estimate of the robot pose at the reference point of each surface patch $\boldsymbol{x}_S$ is incorporated to the state vector $\boldsymbol{x}$ so it contains all the information regarding the submap distribution:

$$\hat{\boldsymbol{x}}_k=\left[\hat{\boldsymbol{x}}_{S_n}\ \ldots\ \hat{\boldsymbol{x}}_{S_1}\right]_k^T \tag{36}$$

with a pose state $\boldsymbol{x}_S$ being:

$$\boldsymbol{x}_S=[x\ y\ z\ \phi\ \theta\ \psi\ ]^T \tag{37}$$

where (x,y,z) is the position of the robot and (ϕ,θ,ψ) are the roll, pitch and yaw angles. The poses are referred to the same common frame $\mathbb{B}$ that was used during the dead reckoning. The covariance matrix for this state is defined as:

$$\boldsymbol{P}_k=E\left([\boldsymbol{x}_k-\hat{\boldsymbol{x}}_k][\boldsymbol{x}_k-\hat{\boldsymbol{x}}_k]^T\right) \tag{38}$$

4.1. Prediction and State Augmentation

The submap poses stored in the state vector are assumed to be static during the execution of the mission. Therefore, the prediction stage of the EKF just maintains the estimated values for the state vector and its covariance. However, every time a new patch is completed, and its pose is introduced in the state vector. This is done during the prediction stage. To be able to fit the requirements of this algorithm (such as the location of the frame $\mathbb{I}$), a new procedure has been developed for the prediction and state augmentation. The procedure explained hereafter uses the previously computed $\boldsymbol{o}_k$ to find the relationship between the patch $\boldsymbol{S}_n$ and $\boldsymbol{S}_{n+1}$ by adding all the incremental displacements in the xy plane and copying the position of the other 4 DoF at the position chosen as frame $\mathbb{I}$ in patch $\boldsymbol{S}_{n+1}$.

Let $\boldsymbol{S}_{n+1}$ be the new patch to be added to the state vector and $\boldsymbol{S}_n$ the last one already added. Then, we need to estimate the transformation ${}^n\boldsymbol{q}_{n+1}=N({}^n\hat{\boldsymbol{q}}_{n+1},\boldsymbol{P}_{{}^n\boldsymbol{q}_{n+1}})$ relating $\boldsymbol{S}_n$ and $\boldsymbol{S}_{n+1}$. The process begins by defining two functions that will be applied to the set of stored $\boldsymbol{o}_k$ relationships between the two patches:

$$f_1(\hat{\boldsymbol{o}})=f_1\left(\begin{bmatrix}x, & y, & z, & \phi, & \theta, & \psi\end{bmatrix}\right)=\begin{bmatrix}x, & y, & 0, & 0 & 0, & 0\end{bmatrix} \tag{39}$$

$$f_2(\hat{\boldsymbol{o}}) = f_2\left(\begin{bmatrix} x, & y, & z, & \phi, & \theta, & \psi \end{bmatrix}\right) = \begin{bmatrix} 0, & 0, & z, & \phi, & \theta, & \psi \end{bmatrix} \tag{40}$$

with Jacobians:

$$\boldsymbol{F}_1 = \begin{bmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times4} \\ \boldsymbol{0}_{4\times2} & \boldsymbol{0}_{4\times4} \end{bmatrix}, \qquad \boldsymbol{F}_2 = \begin{bmatrix} \boldsymbol{0}_{2\times2} & \boldsymbol{0}_{2\times4} \\ \boldsymbol{0}_{4\times2} & \boldsymbol{I}_{4\times4} \end{bmatrix} \tag{41}$$

Then, taking the stored $\boldsymbol{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_m\} \in \boldsymbol{S}_n$, the parameter $\boldsymbol{q}_1 = N(\hat{\boldsymbol{q}}_1, \boldsymbol{P}_{q_1})$ representing the distance from the central position of the $\boldsymbol{S}_n$ patch (defined with the subindex mp) and its last position can be calculated as:

$$\hat{\boldsymbol{q}}_1 = \sum_{k=mp+1}^{m} f_1(\hat{\boldsymbol{o}}_k), \qquad \boldsymbol{P}_{q_1} = \sum_{k=mp+1}^{m} \boldsymbol{F}_1 \boldsymbol{P}_{\boldsymbol{o}_k} \boldsymbol{F}_1^T \tag{42}$$

Next, using the stored $\boldsymbol{O} \in \boldsymbol{S}_{n+1}$, the parameter $\boldsymbol{q}_2 = N(\hat{\boldsymbol{q}}_2, \boldsymbol{P}_{q_2})$ representing the distance from the beginning of $\boldsymbol{S}_{n+1}$ to its center plus the final orientation of the patch can be obtained as:

$$\hat{\boldsymbol{q}}_2 = f_2(\hat{\boldsymbol{o}}_{mp}) + \sum_{k=1}^{mp} f_1(\hat{\boldsymbol{o}}_k), \qquad \boldsymbol{P}_{q_2} = \boldsymbol{F}_2 \boldsymbol{P}_{\boldsymbol{o}_{mp}} \boldsymbol{F}_2^T + \sum_{k=1}^{mp} \boldsymbol{F}_1 \boldsymbol{P}_{\boldsymbol{o}_k} \boldsymbol{F}_1^T \tag{43}$$

Finally, the complete transformation ${}^n\boldsymbol{q}_{n+1}$ relating the centers of both patches is calculated as:

$${}^n\hat{\boldsymbol{q}}_{n+1} = \hat{\boldsymbol{q}}_1 + \hat{\boldsymbol{q}}_2, \qquad \boldsymbol{P}_{{}^n\hat{\boldsymbol{q}}_{n+1}} = \boldsymbol{P}_{q_1} + \boldsymbol{P}_{q_2} \tag{44}$$

Knowing this ${}^n\boldsymbol{q}_{n+1}$ transformation, the state of the filter can be augmented with the new position of $\boldsymbol{S}_{n+1}$ by doing:

$$\hat{\boldsymbol{x}}_k^+ = \begin{bmatrix} \hat{\boldsymbol{x}}_{S_n} \odot^n \hat{\boldsymbol{q}}_{n+1} & \hat{\boldsymbol{x}}_{S_n} & \hat{\boldsymbol{x}}_{S_{n-1}} & \dots & \hat{\boldsymbol{x}}_{S_1} \end{bmatrix}_k^T \tag{45}$$

$$\boldsymbol{P}_{\hat{\boldsymbol{x}}_k^+} = \boldsymbol{J}_{1\odot} \boldsymbol{P}_{\hat{\boldsymbol{x}}_k} \boldsymbol{J}_{1\odot}^T + \boldsymbol{J}_{2\odot} \boldsymbol{P}_{{}^n\hat{\boldsymbol{q}}_{n+1}} \boldsymbol{J}_{2\odot}^T \tag{46}$$

Note that the $\odot$ operator is introduced here to define the way in which the global coordinates of the $\boldsymbol{S}_n$ patch are combined with the relationship between consecutive patches $\boldsymbol{S}_n$ and $\boldsymbol{S}_{n+1}$ to find the position of the patch $\boldsymbol{S}_{n+1}$ in the world frame. The $\odot$ operator is described as:

$$\hat{\boldsymbol{x}}_{S_n} \odot^n \hat{\boldsymbol{q}}_{n+1} = \begin{bmatrix} x_{x_{S_n}} + x_{{}^n\hat{\boldsymbol{q}}_{n+1}} \\ y_{x_{S_n}} + y_{{}^n\hat{\boldsymbol{q}}_{n+1}} \\ z_{{}^n\hat{\boldsymbol{q}}_{n+1}} \\ \phi_{{}^n\hat{\boldsymbol{q}}_{n+1}} \\ \theta_{{}^n\hat{\boldsymbol{q}}_{n+1}} \\ \psi_{{}^n\hat{\boldsymbol{q}}_{n+1}} \end{bmatrix} \tag{47}$$

with Jacobians:

$$\boldsymbol{J}_{1\odot} = \begin{bmatrix} \boldsymbol{I}_{2\times2} & \boldsymbol{0}_{2\times(4+6(n-1))} \\ \boldsymbol{0}_{4\times2} & \boldsymbol{0}_{4\times(4+6(n-1))} \\ & \boldsymbol{I}_{6n\times6n} \end{bmatrix}, \quad \boldsymbol{J}_{2\odot} = \begin{bmatrix} \boldsymbol{I}_{6\times6} \\ \boldsymbol{0}_{6n\times6} \end{bmatrix} \tag{48}$$

4.2. Matching Strategy

When a new patch is available, potential matches are searched among the previously created patches. This is done by determining the intersection between the volumes $\boldsymbol{B}$ (see Equation (19)) of the two potentially matching patches. In this way, two patches are considered to be intersecting if more than a given % of their volumes is shared. The new patch may potentially intersect with several of the patches that already exist, which may or may not be consecutive in time (see the ones overlapping with patch number 13 in Figure 7). As can also be observed, consecutive patches (such as number 1 and 2, or 8 and 9) may have a small overlap with the new patch. For this reason, a new approach is used that consists in joining consecutive patches to maximize the intersecting area. However, this is not recommended for contiguous non-consecutive patches since the drift between them might be significant (e.g., patches number 1 and 6). The proposed approach involves three steps: (1) search for patches intersecting with the new one; (2) search for consecutive patches among those previously selected; and (3) join the patches that are found to be consecutive. The resulting patches are the result of combining the points of the two surfaces and representing them in the frame $\mathbb{I}$ of the earliest created patch.

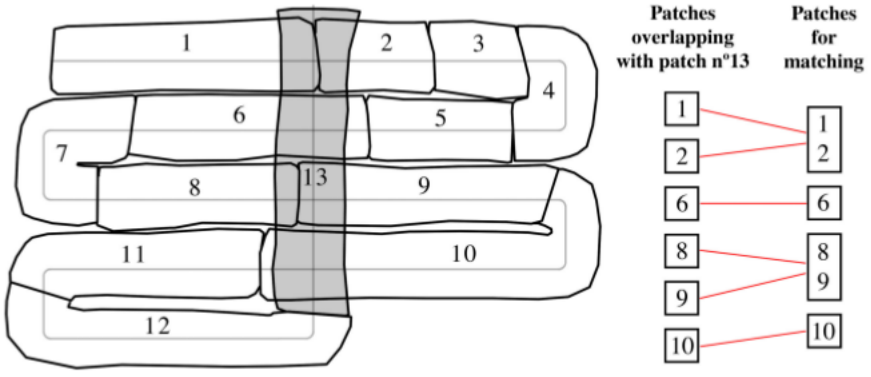


Figure 7. Patch number 13 overlaps with patches 1, 2, 6, 8, 9 and 10. For improving the matching process, patches that are consecutive (1 and 2 as well as 8 and 9) are merged. This results in four patches taking part in the matching process.

4.3. Scan Matching

In order to execute the probabilistic registration algorithm, given two overlapping scans $\boldsymbol{S}_i$ and $\boldsymbol{S}_n$ with related poses $\hat{\boldsymbol{x}}_{S_i}$ and $\hat{\boldsymbol{x}}_{S_n}$, an initial guess $\boldsymbol{q}_0 = N(\hat{\boldsymbol{q}}_0, \boldsymbol{P}_{q_0})$ of their relative displacement is necessary. This can be easily extracted from the state vector using the tail-to-tail transformation:

$$\hat{\boldsymbol{q}}_0 = \ominus \hat{\boldsymbol{x}}_{S_i} \oplus \hat{\boldsymbol{x}}_{S_n} \tag{49}$$

where $\oplus$ and $\ominus$ are the compounding and inverse compounding operators as defined in [30]. Since the tail-to-tail transformation is actually a non-linear function of the state vector $\hat{\boldsymbol{x}}_k$, the uncertainty of the initial guess can be computed using:

$$\boldsymbol{P}_{q_0} = \boldsymbol{H}\boldsymbol{P}_{x_k}\boldsymbol{H}^T \tag{50}$$

where $\boldsymbol{H}$ is the Jacobian computed as:

$$\boldsymbol{H} = [\boldsymbol{J}_{2\oplus_{6\times 6}} \ \ \boldsymbol{0}_{6\times 6(n-i-1)} \ \ \boldsymbol{J}_{1\oplus}\boldsymbol{J}_{\ominus_{6\times 6}} \ \ \boldsymbol{0}_{6\times 6(i-1)}] \tag{51}$$

where $\boldsymbol{J}_{1\oplus}$, $\boldsymbol{J}_{2\oplus}$ and $J_{\ominus}$ are the Jacobians of the compounding and inverse compounding functions as defined in [30]. Finally, $\boldsymbol{0}_{6\times 6(n-i-1)}$ and $\boldsymbol{0}_{6\times 6(i-1)}$ are zero matrices whose dimensions are determined according to the position in the state vector of the surfaces to be registered.

Once the initial displacement guess is available, the registration algorithm presented in Section 3 can be used to produce an updated measurement of this displacement.

4.4. State Update

The initial guess in Equation (49) defines the relationship between two patch poses in the state vector. This can be expressed by means of the following measurement model:

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{v}_k) = \ominus \boldsymbol{x}_{S_i} \oplus \boldsymbol{x}_{S_n} + \boldsymbol{v}_k \tag{52}$$

$\boldsymbol{z}_k$ being the estimated displacement $\boldsymbol{q}_{\min}$ and $\boldsymbol{v}_k$ a zero-mean white Gaussian noise with covariance $\boldsymbol{P}_{\boldsymbol{q}_{\min}}$ accounting for the errors in the registration process. Given that, the standard EKF equations can be used to update the state vector.

5. Experiments and Results

The algorithm has been used to produce the maps for two different underwater datasets. The first one is a bathymetric (2.5D) survey carried out by the *Sirius* AUV [35] on a site of geological interest off the coast of Tasmania (Australia) which has been previously used for bathymetric SLAM [22], while the second one is a full 3D dataset gathered in the Sant Feliu de Guíxols Harbor (Spain) using the *Girona 500* AUV [36] with the multibeam mounted on a pan and tilt unit. The parameters and thresholds that were set for the execution of the algorithm in these experiments can be found in Table 1. Unfortunately, none of the datasets used during the experimental testing have ground truth of the terrain. Therefore, the only option to assess the performance of the algorithm is evaluating the consistency of the resulting map.

Table 1. Thresholds used for the experiments.

	Experiment	
	2.5D	**3D**
Minimum patch size (Section 2.2)	30 m	-
Maximum patch size (Section 2.2)	80 m	-
Normal occupancy (Section 2.2)	23%	-
Patch overlapping (Section 4.2)	30%	30%
Point cloud subsampling (Section 3.4)	0.5 m	1.5 m
Relative displacement to switch from point-to-point to point-to-plane association (Section 3)	1 cm	1 cm

5.1. Bathymetric Survey

This dataset includes depth from a pressure sensor, bottom lock velocities from a DVL, attitude measurements from an AHRS and bathymetric data from a multibeam echosounder installed in the conventional down-looking configuration. The mission surveyed a rectangular area of geological interest several times to generate multiple loop closures. The explored area, mainly flat, has a number of pockmarks with depths of approximately three meters.

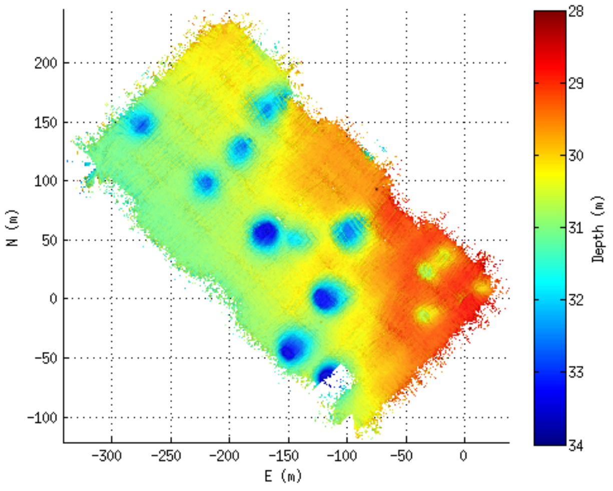
Figure 8 shows the elevation maps built using the dead reckoning navigation (Figure 8a) as well as the one obtained with the proposed technique (Figure 8b). In these two maps, it is possible to observe several differences in the pockmarks. While in the corrected solution (Figure 8b), the pockmarks appear clearly and without much bathymetry-related artefacts on the dead reckoning map, and they are blurred and with some artefacts.

To better assess the correction, the consistency-based error [37] is computed for each cell of the bathymetric map. In Figure 9, it can be seen how the areas of high discrepancy (yellow to dark red) on the dead reckoning error map (Figure 9a) are drastically reduced when the proposed technique is applied (Figure 9b). Table 2 contains the numerical evaluation of the results over the bathymetric data. There, it is possible to see that using the 2.5D statistics (Sum and Mean for the consistency-based error) the improvement is around 19%. Moreover, an additional 3D statistic we have named #Cells has been computed. This statistic consists in counting the number of cells that each map occupies within the same 3D grid. If a map occupies less cells, it is probably because their point clouds are more densely packed due to a better registration. Using this statistic, the improvement of the proposed approach compared to the dead reckoning navigation is 2.17%.

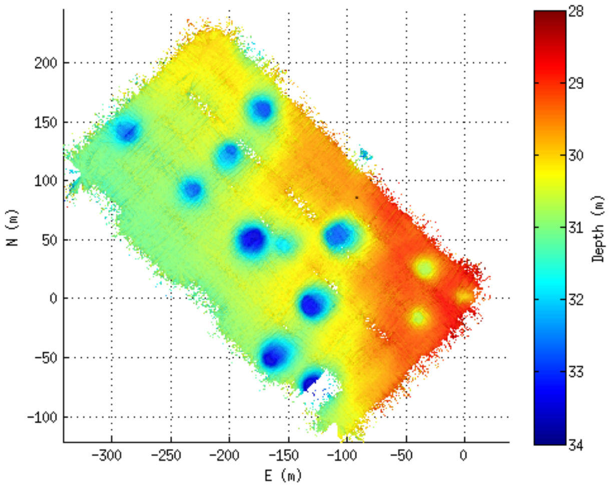
Table 2. Numerical results of the algorithm applied to the pockmarks dataset. The first column (Sum) contains the sum of the error in all the cells, the second one (Mean) contains the mean of the error while the 3rd one (#Cells) contains the number of cells occupied on a 3D grid of 0.5 m resolution.

	Sum	**Mean**	**#Cells**
Dead reckoning	70,986.2	0.3988	37,3121
SLAM	57,521.8	0.3223	36,5014
Improvement *	18.97%	19.2%	2.17%

* The improvement is computed as $\frac{dr-slam}{dr}$ where *dr* stands for dead reckoning.

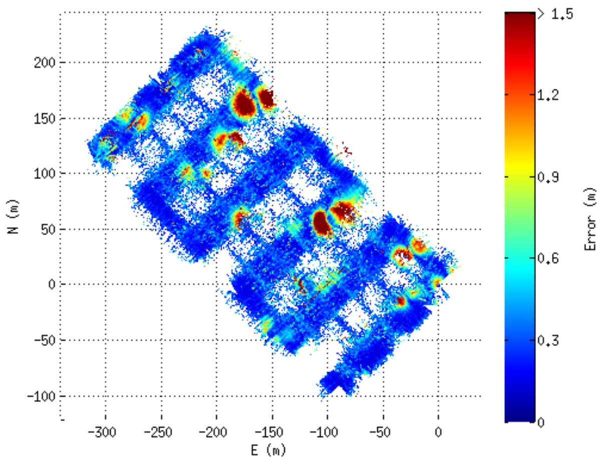


(**a**) Dead Reckoning

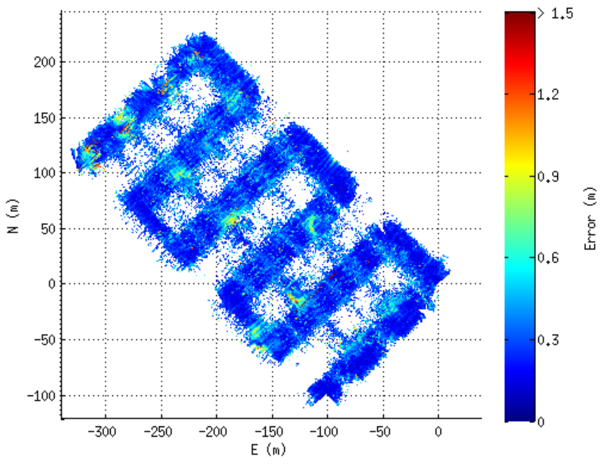


(**b**) SLAM

Figure 8. Bathymetric maps of the area. The color goes from deep (**dark blue**) to shallow (**dark red**). The bathymetry is gridded at 0.5 m. (**a**) Dead Reckoning; (**b**) SLAM.



(**a**) Dead Reckoning



(**b**) SLAM

Figure 9. Consistency-based error maps. The error is color plotted from low (**dark blue**) to high (**dark red**) with 0.5 m grid resolution. (**a**) Dead Reckoning; (**b**) SLAM.

5.2. 3D Experiments

The data was gathered during some field trials for the MORPH European project in 2015. The experiment involved a formation of four marine vehicles (an Autonomous Surface Craft (ASC) and 3 AUVs) exploring a submerged area of the St. Feliu harbor. The *Girona 500* was leading the formation while exploring with the multibeam sonar mounted on a pan and tilt unit (Figure 10) both the seabed and the pier walls, so the formation could be adapted to the presence of obstacles. The mission performed one and a half loops following a zero-shaped trajectory at one corner of the harbor (Figure 11).

During the experiment, the *Girona 500* (Figure 10) was equipped with a DVL, an AHRS, a pressure sensor and a multibeam echosounder. An acoustic modem on *Girona 500* was also used to gather position measurements from a USBL mounted on the ASC navigating on the surface with help of a GPS receiver. The multibeam mounted on the pan and tilt unit allowed us to get full 3D scans by vertically steering the multibeam in front of the robot. Note that, in this experiment, the closure of the surface patches is determined by the completion of a sweep of the pan and tilt, and not by the size or richness of the covered area (see Table 1).

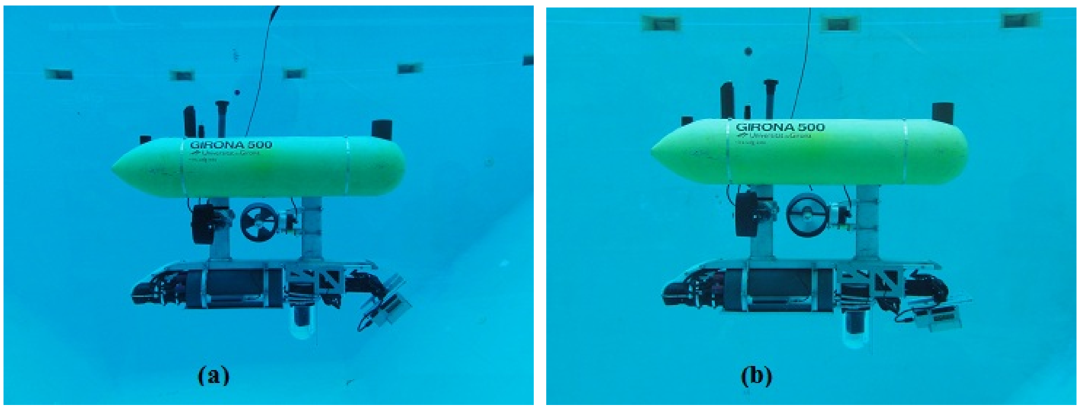


Figure 10. The *Girona 500* AUV in the water tank with the configuration used for the experiments. The multibeam sonar and the pan and tilt unit can be seen at the lower-right side of the vehicle facing in two different directions. In (**a**) the multibeam is tilted at a pitch of around 45° while in (**b**) it is in a downward-looking position.

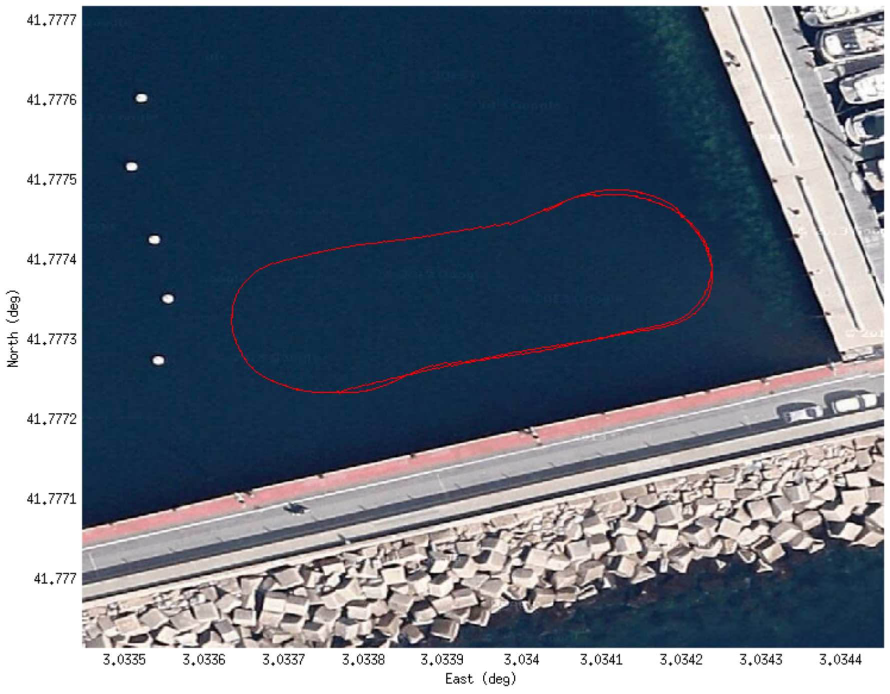


Figure 11. Trajectory of the experiment over the Google Maps image of the St. Feliu de Guíxols harbor.

As far as we know, there is no general method to evaluate the consistency of a 3D map. However, it is possible to use the 3D statistic #Cells presented in the previous section. As previously commented, if a map occupies fewer cells, it is probably because their point clouds are better registered. Nonetheless, this has to be supervised since it is also possible to find other positions of the point clouds that can minimize the number of occupied cells. For this reason, the consistency of the 3D experiments will also be evaluated subjectively (visually assessing the consistency) as well as numerically (counting the number of occupied cells).

The top view of the 3D maps produced after the experiments are presented in Figure 12. There, three surfaces are created for different navigation methods: dead reckoning (Figure 12a), USBL-aided (Figure 12b) and the currently proposed algorithm (Figure 12c). Regarding the number of occupied cells, the proposed method occupies 32570 cells, 5.76% less than the dead reckoning model (34559) while the one aided by the USBL occupies 7.24% less cells (32057). Moreover, the black squares represented in each one of the views highlight the places where it is easier to observe the consistency of the map near to the harbor wall. This area is analyzed in detail in Figure 13. In the left column, the one corresponding to the dead reckoning navigation (views Figure 13a,d,g), clearly shows two parallel lines on the point clouds which correspond to the wall being observed during the first and second laps of the mission. In the other two columns, the one corresponding to the USBL navigation (views Figure 13b,e,h), and the one of the proposed SLAM algorithm (views Figure 13c,f,i) show a single wall, and, thus, a better agreement between the different scans. However, if the point cloud from the USBL navigation is analyzed in detail in the bottom left corner (see Figure 13h), there are still some residues of the two observations of the wall that do not appear in the one from the proposed approach.

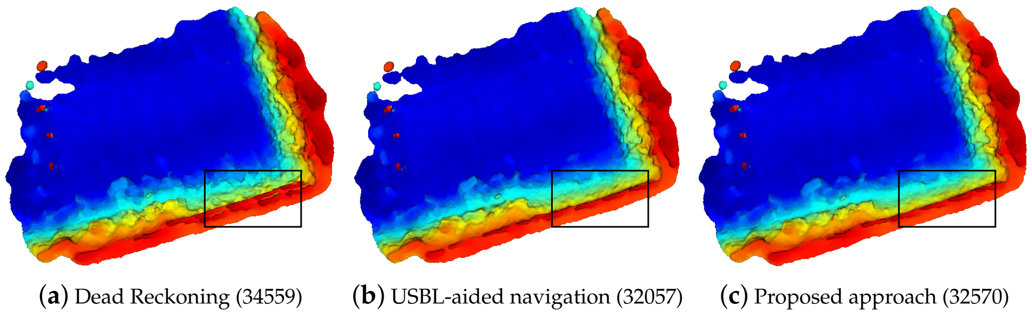

(**a**) Dead Reckoning (34559)
(**b**) USBL-aided navigation (32057)
(**c**) Proposed approach (32570)

Figure 12. Top view of the 3D reconstruction of St. Feliu Harbor using dead reckoning navigation (**a**), USBL-aided navigation (**b**) and the proposed SLAM algorithm (**c**).The bottom part of the model is the vertical wall of the pier. Under each view, written inside parentheses, the number of cells occupied by each model's point clouds can be observed. The meshes are reconstructed using [38] and colored according to the depth (deeper parts are in blue, shallower ones in red).

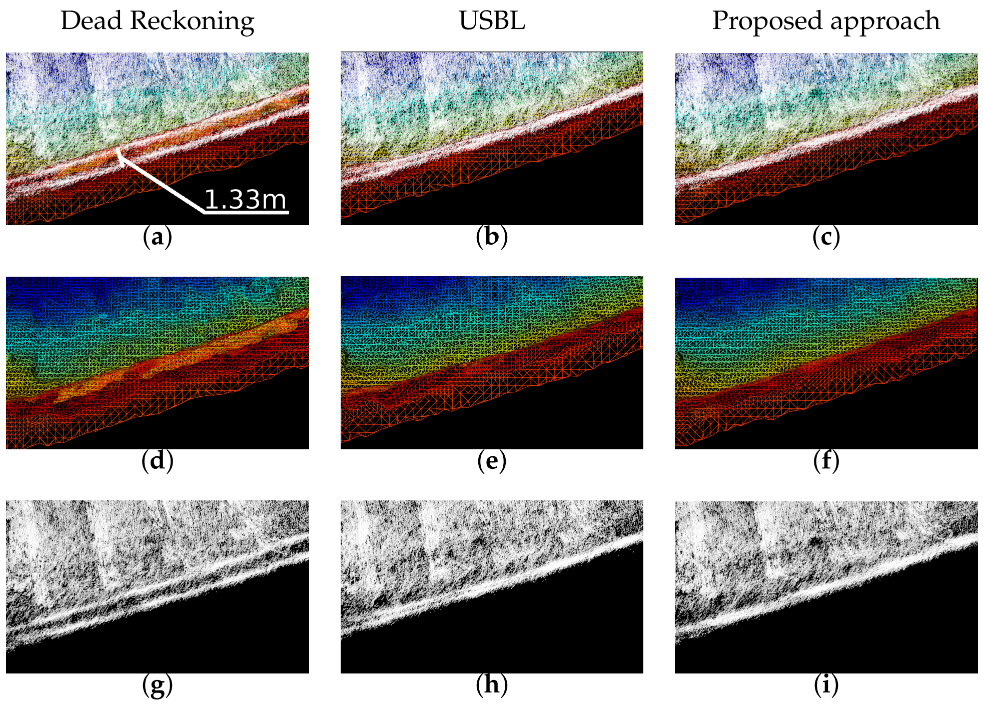


Figure 13. Zoom in the highlighted area of Figure 12. First row (**a**-**c**) shows the point clouds and the reconstructed meshes. Second (**d**–**f**) and third (**g**–**i**) rows show the mesh and point clouds respectively. The columns, from left to right are related to the results obtained with: (1) dead reckoning (**a**, **d** and **g**); (2) USBL-aided (**b**, **e** and **h**) and (3) proposed approach (**c**, **f** and **i**).

6. Conclusions

This paper has presented a probabilistic underwater 3D SLAM for multibeam sonar data that deals with the subdivision of the surface into patches, taking into account the motion uncertainty during their formation. An adaptive sampling procedure for the sensor data has been introduced to deal with areas of the patches that are not relevant (*i.e.*, without relief) to avoid the pICP converging to local minima as well as reducing the computational time. Furthermore, an heuristic has been used to decrease the complexity of the association step of the pICP from $O(n^2)$ to $O(n)$ taking advantage of the probabilistic ellipsoid of each point and using a support grid.

The algorithm has been tested using two real world datasets. In both of them, it is possible to observe how the consistency of the model obtained using the proposed algorithm is higher than that obtained with dead reckoning and is even comparable to the one obtained using USBL navigation in the case of the 3D dataset.

Future work will have to focus on correcting the internal patch error. In the method presented here, only the relative positions of the patches are corrected, but the patch itself is not modified once closed. Although the proposed method has been proved to be useful for obtaining consistent maps, it is not possible to use it online due to its computational cost if the point sampling is not tuned properly. Therefore, further investigation could be done in this field to allow the algorithm to work online. Finally, in the future, we plan to test the algorithm using a camera-laser system, which produces data of similar characteristics to that of a multibeam sonar (2D swath profiles) but with a much different uncertainty level in the measurements.

Acknowledgments: This work was supported by the Spanish project DPI2014-57746-C3-3-R (ARCHROV) and two European Commission's Seventh Framework Program projects: FP7-ICT-2011-7-288704 (MORPH) and FP7-INFRASTRUCTURES-2012-312762 (EUROFLEETS2). We would also like to thank Oscar Pizarro and Stephen Barkby for sharing the bathymetric dataset and information. In addition, we want to thank the staff who permitted the survey mission for the dataset with the *Sirius* AUV, including Ian Mahon, Matthew Johnson-Roberson, Duncan Mercer, George Powell, Ritesh Lal, Paul Rigby, Jeremy Randle and Bruce Crundwell.

Author Contributions: The work presented in this paper has been done as a collaboration by all the authors. Pere Ridao and David Ribas were the project leaders and in charge of the direction and supervision, while Albert Palomer implemented and tested the algorithms with the datasets. All the authors discussed the results obtained and reviewed the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Panish, R.; Taylor, M. Achieving high navigation accuracy using inertial navigation systems in autonomous underwater vehicles. In Proceedings of the MTS/IEEE Oceans, Santander, Spain, 6–9 June 2011; pp. 1–7.
2. Kinsey, J.C.; Whitcomb, L.L. Preliminary field experience with the DVLNAV integrated navigation system for oceanographic submersibles. *Control Eng. Pract.* **2004**, *12*, 1541–1549.
3. Batista, P.; Silvestre, C.; Oliveira, P. Single Beacon Navigation: Observability Analysis and Filter Design. In Proceedings of the American Control Conference (ACC), Baltimore, MD, USA, 30 June–2 Juny 2010; pp. 6191–6196.
4. Vallicrosa, G.; Ridao, P.; Ribas, D. AUV Single Beacon Range-Only SLAM with a SOG Filter. *IFAC Workshop Navig. Guid. Control Underw. Veh.* **2015**, *48*, 26–31.
5. Thomas, H.G. GIB Buoys: An Interface Between Space and Depths of the Oceans. In Proceedings of the Workshop on Autonomous Underwater Vehicles, Cambridge, MA, USA, 20–21 August 1998; pp. 181–184.
6. Mandt, M.; Gade, K.; Jalving, B. Integrateing DGPS-USBL position measurements with inertial navigation in the HUGIN 3000 AUV. In Proceedings of the 8th Saint Petersburg International Conference on Integrated Navigation Systems, Saint Petersburg, Russia, 28–30 May 2001.
7. Carreño, S.; Wilson, P.; Ridao, P.; Petillot, Y.; Carreno, S.; Wilson, P.; Ridao, P.; Petillot, Y. A survey on Terrain Based Navigation for AUVs. In Proceedings of the MTS/IEEE Oceans, Seattle, WA, USA, 20–23 September 2010; pp. 1–7.

8. Bailey, T.; Durrant-Whyte, H. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robot. Autom. Mag.* **2006**, *13*, 108–117.
9. Eustice, R.; Pizarro, O.; Singh, H. Visually Augmented Navigation in an Unstructured Environment Using a Delayed State History. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA, 26 April–1 May 2004; Volume 1, pp. 25–32.
10. Williams, S.; Mahon, I. Simultaneous Localisation and Mapping on the Great Barrier Reef. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA, 26 April–1 May 2004; Volume 2, pp. 1771–1776.
11. Eustice, R.; Singh, H.; Leonard, J.; Walter, M.; Ballard, R. Visually Navigating the RMS Titanic with SLAM Information Filters. In Proceedings of the Robotics Science and Systems, Cambridge, MA, USA, 8–11 June 2005.
12. Johnson-Roberson, M.; Pizarro, O.; Williams, S.B.; Mahon, I. Generation and Visualization of Large-Scale Three-Dimensional Reconstructions from Underwater Robotic Surveys. *J. Field Robot.* **2010**, *27*, 21–51.
13. Aykin, M.D.; Negahdaripour, S. On Feature Matching and Image Registration for Two-dimensional Forward-scan Sonar Imaging. *J. Field Robot.* **2013**, *30*, 602–623.
14. Hurtós, N.; Ribas, D.; Cufí, X.; Petillot, Y.; Salvi, J. Fourier-based Registration for Robust Forward-looking Sonar Mosaicing in Low-visibility Underwater Environments. *J. Field Robot.* **2015**, *32*, 123–151.
15. Leonard, J.J.; Carpenter, R.N.; Feder, H.J.S. Stochastic Mapping Using Forward Look Sonar. *Robotica* **2001**, *19*, 467–480.
16. Carpenter, R.N. Concurrent Mapping and Localization with FLS. In Proceedings of the Workshop on Autonomous Underwater Vehicles, Cambridge, MA, USA, 20–21 August 1998; pp. 133–148.
17. Williams, S.; Newman, P.; Rosenbaltt, J.; Dissanayake, G.; Durrant-whyte, H. Autonomous underwater navigation and control. *Robotica* **2001**, *19*, 481–496.
18. Newman, P.; Leonard, J. Pure Range-Only Sub-Sea SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, 14–19 September 2003; pp. 1921–1926.
19. Ribas, D.; Ridao, P.; Domingo, J.D.; Neira, J. Underwater SLAM in Man-Made Structured Environments. *J. Field Robot.* **2008**, *25*, 898–921.
20. Fairfield, N.; Jonak, D.; Kantor, G.A.; Wettergreen, D. Field Results of the Control, Navigation, and Mapping Systems of a Hovering AUV. In Proceedings of the 15th International Symposium on Unmanned Untethered Submersible Technology, Durham, NH, USA, 19–20 August 2007.
21. Roman, C.; Singh, H. A Self-Consistent Bathymetric Mapping Algorithm. *J. Field Robot.* **2007**, *24*, 23–50.
22. Barkby, S.; Williams, S.B.; Pizarro, O.; Jakuba, M. A Featureless Approach to Efficient Bathymetric SLAM Using Distributed Particle Mapping. *J. Field Robot.* **2011**, *28*, 19–39.
23. Eliazar, A.; Parr, R. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping without Predetermined Landmarks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, 14–19 September 2003; Volume 18, pp. 1135–1142.
24. Zandara, S.; Ridao, P.; Ribas, D.; Mallios, A.; Palomer, A. Probabilistic Surface Matching for Bathymetry Based SLAM. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013.
25. Hernández, E.; Ridao, P.; Ribas, D.; Mallios, A. Probabilistic sonar scan matching for an AUV. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, USA, 10–15 October 2009; pp. 255–260.
26. Mallios, A.; Ridao, P.; Ribas, D.; Maurelli, F.; Petillot, Y. EKF-SLAM for AUV navigation under probabilistic sonar scan-matching. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 4404–4411.
27. Burguera, A.; Oliver, G.; González, Y. Scan-Based SLAM with Trajectory Correction in Underwater Environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 2546–2551.
28. Mallios, A.; Ridao, P.; Ribas, D.; Hernández, E. Scan Matching SLAM in Underwater Environments. *Auton. Robots* **2014**, *36*, 181–198.
29. Burguera, A.; González, Y.; Oliver, G. A Probabilistic Framework for Sonar Scan Matching Localization. *Adv. Robot.* **2008**, *22*, 1223–1241.

30. Smith, R.; Self, M.; Cheeseman, P. Chapter 3: Estimating Uncertain Spatial Relationships in Robotics. In *Autonomous Robot Vehicles*; Cox, I.J., Wilfong, G.T., Eds.; Springer: New York, NY, USA, 1990; Volume 1, pp. 167–193.
31. Kanazawa, Y.; Kanatani, K. Reliability of fitting a plane to range data. *IEICE Trans. Inf. Syst.* **1995**, *E78D*, 1630–1635.
32. Weingarten, J. Feature-based 3D SLAM. Ph.D. Thesis, École Polytechinque Fédérale de Lausanne, Lausanne, Switzerland, 6 September 2006.
33. Pathak, K.; Vaskevicius, N.; Birk, A. Revisiting uncertainty analysis for optimum planes extracted from 3D range sensor point-clouds. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 12–17 May 2009; pp. 1631–1636.
34. Palomer, A.; Ridao, P.; Ribas, D.; Mallios, A.; Vallicrosa, G. Octree-Based Subsampling Criteria for Bathymetric SLAM. In Proceedings of the XXXV Jornadas de Automática, Valencia, Spain, 3–5 September 2014; pp. 1–6.
35. Williams, S.; Pizarro, O.; Mahon, I.; Johnson-roberson, M. Chapter 9: Simultaneous Localisation and Mapping and Dense Stereoscopic Seafloor Reconstruction Using an AUV. In *Experimental Robotics: The Eleventh International Symposium*; Springer: Berlin, Geramny, 2009; pp. 407–416.
36. Ribas, D.; Palomeras, N.; Ridao, P.; Carreras, M.; Mallios, A. Girona 500 AUV: From Survey to Intervention. *IEEE ASME Trans. Mechatron.* **2012**, *17*, 46–53.
37. Roman, C.; Singh, H. Consistency based error evaluation for deep sea bathymetric mapping with robotic vehicles. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL, USA, 15–19 May 2006; pp. 3568–3574.
38. Kazhdan, M.; Hoppe, H. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* **2013**, *32*, 1–13.