Received: 14 November 2022 | Revised: 14 July 2023 | Accepted: 31 October 2023

DOI: 10.1002/rob.22272

RESEARCH ARTICLE

WILEY

Underwater Pose SLAM using GMM scan matching for a mechanical profiling sonar

Pau Vial[1] | Narcís Palomeras[1] | Joan Solà[2] | Marc Carreras[1]

[1]Institut VICOROB, Universitat de Girona, Girona, Catalunya, Spain

[2]Institut de Robòtica i Informàtica Industrial, Universitat Politècnica de Catalunya, Barcelona, Catalunya, Spain

Correspondence
Pau Vial, Institut VICOROB, Universitat de Girona, Girona 17003, Catalunya, Spain.
Email: pau.vial@udg.edu

Funding information
CRUE-CSIC; PLOME, Grant/Award Number: PLEC2021-007525; Biter-AUV, Grant/Award Number: PID2020-114732RB-C33; Spanish Government, Grant/Award Number: FPU19/03638

Abstract

The underwater domain is a challenging environment for robotics because widely used electromagnetic devices must be substituted by acoustic equivalents, much slower and noisier. In this paper a two-dimensional pose simultaneous localization and mapping (SLAM) system for an Autonomous Underwater Vehicle based on inertial sensors and a mechanical profiling sonar is presented. Two main systems are specially designed. On the one hand, a dead reckoning system based on Lie Theory is presented to track integrated pose uncertainty. On the other hand, a rigid scan matching technique specialized for acoustic data is proposed, which allows one to estimate the uncertainty of the matching result. Moreover, Bayesian–Gaussian mixtures models are introduced to the scan matching problem and the registration problem is solved by an optimization in Lie groups. The SLAM system is tested on real data and executed in real time with the robotic application. Using this system, section maps at constant depth can be obtained from a three-dimensional underwater domain. The presented SLAM system constitutes the first achievement towards an underwater Active SLAM application.

KEYWORDS
acoustic scan registration, autonomous underwater vehicles, field robotics, Gaussian mixtures model, lie theory, optimization, Pose SLAM, profiling sonars

1 | INTRODUCTION

Localization is a fundamental problem in achieving true autonomy for robotic applications. In the underwater domain this problem is even more challenging as global localization systems, like, Global Positioning System (GPS), or widely used communication systems, like, WiFi, are not available due to water attenuation of electromagnetic waves. Moreover, Visual Odometry is not viable due to the poor visibility conditions in the water. Therefore, Autonomous Underwater Vehicles (AUVs) often have to only relay on dead reckoning navigation that drifts over time. Drifting is problematic when mapping using scanners because when an explored region is revisited it appears in a different position generating inconsistencies in the map. To prevent maps from inconsistencies a simultaneous localization and mapping (SLAM) problem, reviewed in Durrant-Whyte and Bailey (2006) and Bailey and Durrant-Whyte (2000), must be solved fusing exteroceptive data from the surroundings of the robot with proprioceptive data from the robot. In this way, previously observed regions can be related maintaining the map consistency.

When solving an SLAM problem using a range sensor, overlapping scans are registered to get its correct alignment and find a relative pose constraint between both viewpoints. Landmarks are missing in this SLAM problem formulation, leading to a Pose SLAM problem, presented in Lu and Milios (1997), which only estimates the

robot trajectory. Therefore, the map is implicit in the trajectory and is formed by the composition of all the scans, each one related to a pose of the robot trajectory.

Many techniques to solve a Pose SLAM problem are available nowadays. On the one hand, filtering techniques only estimate the current robot pose, marginalizing all the old observations in a prior on the current state. Many filters have been proposed to solve the SLAM problem, such as the Extended Kalman Filter (EKF) applied in Solà (2014) or the Rao-Blackwellized Particle Filter, implemented in the well-known `GMapping` library presented in Grisetti et al. (2007). On the other hand, smoothing techniques estimate the whole robot trajectory and all observations are taken into account in a least-squares problem. As explained by Dellaert and Kaess (2017) smoothing techniques express the SLAM problem by a factor graph, a bipartite graph $F = (\mathcal{X}, \mathcal{Z}, \mathcal{E})$ with two types of nodes $\mathcal{X}$ and $\mathcal{Z}$ and edges $\mathcal{E}$. Variable nodes $x_i \in \mathcal{X}$ represent estimated variables while factor nodes $\phi_j \in \mathcal{Z}$ represent observations on the variables. Edges $e_{ij} \in \mathcal{E}$ can only connect variable nodes with factor nodes. This graphical representation of the SLAM problem allows one to divide it into two layers: the SLAM Front-End and the SLAM Back-End. The SLAM Front-End is in charge of building the factor graph, where each sensor measure is modeled by a factor ϕ_j and each robot pose is modeled as a variable x_j. This is a practical way to solve sensor fusion, as every sensor is modeled by a particular type of factor and every sensor measurement is an instance of its corresponding factor. The SLAM Back-End is in charge of solving the defined problem, searching the Maximum a Posteriori (MAP) estimate of the trajectory X given all the measurements Z. Using the factor graph and its factorization properties, the MAP inference problem can be defined as

$$\mathcal{X}^{\text{MAP}} = \arg\max_{X} \prod_{j} \phi_j,$$

which is equivalent to solve a nonlinear least-squares problem. Therefore, general-purpose tools for least-squares optimization can be used as an SLAM Back-End, like, the `GTSAM` library available in Dellaert (n.d.), the `SLAM++` solver from Ila et al. (2017), or the `Ceres` solver available in Agarwal and Mierle (n.d.). As discussed by Dellaert (2021), these solvers can be applied to other big problems in robotics, such as Structure from Motion, Bundle Adjustment, Optimal Control, Calibration, Inertial Measurement Unit (IMU) Preintegration, or Path Planning.

In the underwater domain, the capabilities of perception sensors that use electromagnetism as their working principle—such as optical cameras or light detection and ranging (LIDAR) sensors—are hugely reduced due to the water attenuation of electromagnetic waves. In its place, acoustic devices are the alternative. For example, in point cloud perception, the LIDAR sensor, widely used in ground or aerial robotic applications, must be substituted by a Forward-Looking Sonar or a Profiling Sonar. However, due to the high difference between the light velocity and the sound velocity, acoustic sensors are slower and less accurate in comparison with their electromagnetic counterpart. Hence, they produce sparser point clouds with a large quantity of outliers that make scan matching a more complicated problem.

Few SLAM problems using range sensors have been attempted in the underwater domain. Mallios et al. (2009) proposed to register scans obtained from a Mechanical Scanning Imaging Sonar by applying the probabilistic Iterative Correspondence method and to fuse the result with dead reckoning using an EKF. Johannsson et al. (2010) solved an online Pose SLAM problem using the Normal Distributions Transforms (NDTs) technique to register point clouds extracted from acoustic images obtained using an Imaging Sonar. Following this localization framework, Hover et al. (2012) proposed an exploration method for ship hull inspection using an AUV. Keeping in ship hull inspection, VanMiddlesworth et al. (2015) and Teixeira et al. (2016) used a Multibeam Profiling Sonar to construct and register overlapping submaps using the Iterative Closest Point (ICP) method to solve a Pose SLAM problem. Finally, Vallicrosa and Ridao (2018) used a mechanical profiling sonar to learn a Hilbert Map that models occupancy and a Rao-Blackwellized Particle Filter was used to solve the Pose SLAM problem. However, none of these algorithms combine a full robot trajectory estimation using a smoothing technique properly defined using Lie groups to represent pose (see Solà et al., 2020 for an introduction to this topic) with a scan matching technique specialized for the key features of acoustic point clouds and able to provide the uncertainty of the registration according to the structure implicit in the scan.

In this paper we solve an online Pose SLAM problem using an AUV equipped with a mechanical profiling sonar applying smoothing techniques. The use of smoothing techniques allows one to close loops in the factor graph when loop closure events are detected, which means that we are able to detect when the AUV revisits some previously mapped region and register the current views against the previous ones to avoid map inconsistencies. As a mechanical profiling sonar provides plain scans, only layer maps of the environment can be generated and, in consequence, an SLAM problem in the Special Euclidean SE(2) group is solved. Two main innovations are introduced in the SLAM algorithm. First, we propose a new dead reckoning system formulated using Lie Theory able to measure integrated pose uncertainty in the Special Euclidean SE(n) group. This system is specially designed for the underwater environment as it is based on angular velocities, measured by gyroscopes, and linear velocities, measured by a Doppler Velocity Log (DVL); in contrast to usual IMU used in ground and aerial applications based on gyroscopes and accelerometers. Second, we present a new scan matching technique to register sonar scans, specially designed taking into account the nature of acoustic data. Apart from providing a registration result, this technique also returns an estimation of the uncertainty of the scan matching result based on the structure implicit in the scan; a feature not common in most registration techniques and necessary when applying scan matching in an SLAM problem. Moreover, considering the sparsity and the noisy nature of acoustic point clouds, the proposed registration technique applies Gaussian mixtures models (GMMs) to model and register scans. In this line, for the first time we propose to use the Bayesian-GMM algorithm, presented in Attias (2000), to automatically learn the optimal number of clusters needed to model a scan using a GMM, achieving a flexible tool to

autonomously adapt to different environments with different structures. Finally, the registration problem is also properly formulated using Lie groups to represent pose. The remainder of this paper is organized as follows. Section 2 presents a survey of the available point cloud registration techniques and SLAM Back-End solvers. Section 3 presents the dead reckoning and the scan matching systems used by the SLAM Front-End, described in Section 4. Section 5 presents the data sets used to test the SLAM system, while the results of these experiments are shown in Section 6. Finally, Section 7 presents the conclusions and suggests future work.

2 | RELATED WORK

In this section, the state-of-the-art techniques related to the proposed SLAM system are presented. First, point cloud registration algorithms used to align overlapping scans obtained by a range sensor are summarized. These techniques are the key components for the proposed SLAM Front-End. Second, the available open-source solvers that can be used for the SLAM Back-End are presented.

2.1 | Point cloud perception

Two broad categories of point cloud registration algorithms exist: those that do direct point-to-point matching, and those based on probabilistic fields. The former category includes the widely used ICP algorithm, presented in Besl and McKay (1992), and all its variants, being Generalized ICP from Segal et al. (2009) the most popular. The latter category refers to registration algorithms where point clouds are represented by GMMs, a probabilistic model that gives a continuous representation of a scan. Fitting point clouds into GMMs has many advantages, such as data compression or sensor noise modeling by the covariance matrix of each Gaussian component. Mathematically, the field representation means that the scan is modeled by a function $p(X) : \mathbb{R}^D \to \mathbb{R} \in C^\infty$—where D is the point cloud dimension—which enables the use of gradient-based methods to solve the registration problem. Moreover, the analytic derivatives of the model allow one to explicitly track the uncertainty of the problem during the registration process.

The most popular field-based scan matching technique is the NDTs. The key characteristic of all NDT-based algorithms is that the GMM is just fitted by sampling the point cloud into a Cartesian grid. NDT was first proposed by Biber and Strasser (2003) to align two-dimensional (2D) point clouds. Magnusson (2009) extended it to three-dimensional (3D) registration. This method, called Point to Distribution (P2D), fits the first scan into a GMM and finds the robot motion which provides the maximum likelihood (ML) solution for the points of the second scan falling in the probabilistic field that models the first scan. P2D was applied in many problems involving LIDAR sensor. For example, Li et al. (2010) solved a localization problem using an EKF inside a mine, Saarinen, Andreasson, and Stoyanov (2013) solved a localization problem using a Particle Filter in an industrial environment or Stoyanov et al. (2010) solved a path planning problem based on an NDT map of a mine.

Stoyanov et al. (2012) proposed an improved method called Distribution to Distribution (D2D). This method fits each scan into a GMM and the $\mathcal{L}_2$ distance between both models is minimized to solve the registration problem. This is a similar approach to ICP because the cost function for each component is evaluated only with the closest component of the other scan. However, D2D uses a probabilistic cost function, instead of the Euclidean distance of the ICP algorithm, and a reduced number of associations must be done due to the compression of the point clouds given by the GMM representation.

To use the NDT framework in mapping and path planning applications, Saarinen, Andreasson, Stoyanov, Ala-Luhtala et al. (2013) proposed the Normal Distributions Transform Occupancy Maps (NDT-OM) to keep explicitly the information about the free space, as NDT only models the shape of the contour of the free space and does not label explicitly the free space. The NDT-OM method fuses the NDT representation into the OctoMap framework, presented by Hornung et al. (2013). To do so, a global probabilistic Octree is generated to store occupancy information but, instead of only saving the probability of occupation for each voxel, occupied voxels also store a component of a global GMM. This fusion allows one to use a coarser grid than an OctoMap, because each Gaussian component models the shape of the contour of the free space, and also allows one to work in dynamic environments, thanks to the probabilistic occupancy information of an OctoMap. NDT-OM has been used, for example, by Stoyanov et al. (2013) to solve a mapping problem in an industrial environment with the presence of dynamic objects, where, first, the scan is registered against the global map using D2D algorithm and, then, the scan is added to the global NDT-OM. Moreover, Einhorn and Gross (2013) applied NDT-OM to solve a Pose SLAM problem in a domestic environment, where the loop closure registration between submaps is attempted applying the D2D algorithm solved using heuristic optimization methods to handle bad initialization and local minima. At this point, it has to be noted that all presented NDT methods are solved using the Newton Method, which is not able to guarantee global minima and, consequently, find the true alignment between two misaligned scans. To handle this, Bouraine et al. (2021) proposed a Particle Swarm Optimization solver for the D2D method (NDT–P2D–PSO), which is able to numerically guarantee global minima and can be run in real time to solve an SLAM problem.

All NDT-based methods described up to here have been derived for and applied on LIDAR sensors and depth cameras. However, other kinds of range sensors exists which return point cloud data. The radar sensor is a well-known example. It has a similar behavior to the aforementioned sensors, however, it returns point cloud data with additional noise but it allows one to work in low vision conditions, such as dusty, smoky, or foggy environments. In Mielle et al. (2019) D2D has been applied to radar sensor data and it was concluded that it works well without any change to the registration algorithm. Another kind of point cloud data sensor is the sonar sensor. Sonar sensors, instead of relying on electromagnetic waves, they use

mechanical waves as their working principle, which changes a lot of their capabilities. First, its sampling rate is on the order of seconds, as opposed to the order of hundreds of milliseconds of the aforementioned sensors, which means that two successive scans are further in space and its overlap is lower. Also, scans are wrapped by robot motion, which has to be compensated in some way. Second, mechanical waves generate more noise and a bigger quantity of outliers on the output point clouds. To handle this situation Burguera et al. (2009) defined the filtered sNDT–P2D method, where a filter to detect spurious points and a methodology to check the continuity between GMM components are defined. The former is based on a random sample consensus algorithm that checks whether the points fit a Gaussian distribution and the latter is based on the analysis of the eigenvalues and the eigenvectors of the covariance matrix of each component.

As appointed in the beginning of this section, the key characteristic of all NDT-based algorithms is that GMM components are derived just by sampling the point cloud into a Cartesian grid. However, a rigid sampling of the point cloud into a grid is not the best way to capture the implicit structure of a point cloud. To fix that, unstructured methods could be used. Jiang et al. (2019) proposed to use K-means algorithm to fit a point cloud into a GMM. They use a fixed number of components K—depending on the number of points in the scan—and, instead of minimizing the $\mathcal{L}_2$ distance like in the D2D algorithm, they minimize the Kullback–Leibler (KL) divergence between the GMMs that represent the two registering scans. This technique is known as the Symmetrical Kullback–Leibler Divergence Distribution to Distribution (SKLD-D2D). However, K-means algorithm is a heuristic method that does not solve the clustering problem based on probability theory. To improve that, one can use a ML estimator to fit a GMM of fixed K components into a point cloud applying the Expectation-Maximization (EM) algorithm, proposed by Dempster et al. (1977). Tabib et al. (2018) proposed to use the EM algorithm to fit a GMM to each scan. Also, a registration method which improves the D2D algorithm is proposed: converting anisotropic covariance matrices into isoplanar matrices—to improve the convexity of the cost function—and using a more complex objective function—avoiding simplifications on the $\mathcal{L}_2$ distance for the Gaussian. To determine the number of components K for the model that best captures the intrinsic structure of the scan, Eckart et al. (2016) proposed a hierarchical EM algorithm, called Expectation Sparsification, which starts with a low number of K_0 components and, then, the data are soft partitioned into more components following an Octree structure until the likelihood of the model does not improve or a maximum number of levels is reached. Using this framework, in Eckart (2017) REM-Seg, MLMD and MDME methods are proposed to solve the registration problem using the EM algorithm to assign points or components of one scan to points or components of another scan, depending on the specific method.

Our registration proposal follows this line of development using the EM algorithm to fit a GMM into a point cloud, but giving a Bayesian treatment to the model parameters reaching a MAP estimator, as it was proposed in Attias (2000). The most interesting implication of this improvement is that now the fitting process allows one to also learn the optimal number of components K for the GMM that best models the structure of the scan and its measurement noise. In this way, the number of K components will be adjusted optimally for each point cloud. To do so, Variational Inference is needed to be able to use the EM algorithm to fit a GMM with priors on its parameters. Then, a gradient-based method is used to solve the registration problem using cost functions from NDT-based algorithms reviewed for general GMM.

The proposed method shares structure with the ICP algorithm, as both are based on the EM algorithm. As it is observed in Jiang and Vemuri (2011), in the E-step of the ICP algorithm, correspondences point-to-point are established and, in the M-step, the Euclidean distance is minimized—a collapse of the KL divergence for Gaussian mixtures where both mixtures shares the same spherical covariances in all components. In our approach, the EM algorithm is also used but, instead of using it to solve the registration, it is used to cluster the point cloud and fit a GMM into the scan. Then, the registration is solved by means of gradient-based methods thanks to the continuous representation given by the fitted model.

2.2 | SLAM back-end solvers

An SLAM problem modeled by a nonlinear factor graph can be numerically solved if it is, first, linearized and, then, solved applying least-squares optimization by means of the Gauss–Newton Method (see Grisetti et al., 2020 for a further explanation). Many general-purpose solvers implemented in open-source libraries have been used in the graph SLAM community, such as `g2o` from Kümmerle et al. (2011), `Ceres` available in Agarwal and Mierle (n.d.), $\sqrt{\texttt{SAM}}$ from Dellaert and Kaess (2006), `iSAM2` from Kaess et al. (2012) and `SLAM++` from Ila et al. (2017). However, some differences exist between them, which allows us to classify them into two broad categories: batch solvers and incremental solvers. Batch solvers solve the problem from scratch every time the solver is called, making them useful in offline applications. In contrast, incremental solvers reuse the solution from the last solver call and only recalculate parts of the solution affected by the new factors added to the factor graph. This preservation of information allows one to improve the computational efficiency of the algorithms, making this type of solvers suitable for online applications where measurements arrive as a temporal sequence and intermediate solutions are needed.

Many batch solvers are available. $\sqrt{\texttt{SAM}}$ solver, presented in Dellaert and Kaess (2006) as part of the `GTSAM` library available in Dellaert (n.d.), applies the Levenberg–Marquard (LM) method and uses QR decomposition to perform matrix inversion. To deal with the problem sparsity, it uses the `COLAMD` algorithm to get a favorable factorizing order to favor sparsity in the factorized matrix R. On the contrary, `g2o` solver, presented in Kümmerle et al. (2011), also applies the LM method but uses Cholesky factorization to perform matrix inversion. `g2o` solver was the first solver to consider Lie Theory (see Solà et al., 2020) to treat rotation correctly from an

analytical point of view in an on-manifold optimization framework, since orientation belongs to a non-Euclidean space. Finally, `Ceres` solver, available in Agarwal and Mierle (n.d), is a general framework to perform nonlinear batch optimization. It takes advantages of vector operations in Central Processing Units (CPUs) and its most relevant feature is the efficient use of Automatic Differentiation, which consists of the algorithmic computation of derivatives.

The most popular incremental solver is `iSAM2`, presented by Kaess et al. (2012) and included in the `GTSAM` library available in Dellaert (n.d.). This solver is based on an incremental QR decomposition and uses a Bayes Tree to encode the structure of the matrix R to detect the affectation of the newly added factors into the last factorization. Thus, only the affected parts of R are recovered. In `iSAM2` the factorization algorithm is done by graph manipulation on the graphical model of the problem and the relinearization and the variable ordering are performed incrementally as needed. To choose a favorable variable ordering for the incremental factorization, `Constrained COLAMD` algorithm is used to maintain at the end of the list the newest nodes as they are the most sensitive to be affected by the new factors. To maintain a good linearization of the problem, the nodes for which its estimate diverges from the current linearization are incrementally relinearized. `SLAM++`, presented by Ila et al. (2017), is another incremental solver. Contrary to `iSAM2`, `SLAM++` uses an incremental Cholesky factorization and it avoids matrix-graph conversions to perform the factorization. In `SLAM++` also the variable ordering of the affected parts of R by an update is computed incrementally to favor factorization. A key feature of this solver is its module for efficient marginalization of the solution uncertainty for an isolated set of nodes. However, new versions of `iSAM2` also incorporate this capability.

In this paper, a Pose SLAM problem is solved online and, consequently, an incremental solver is needed. We chose `iSAM2` as the SLAM Back-End because the explicit graphical representation of the problem seems interesting in the future when solving an active SLAM problem and updates on an occupancy map must be done due to SLAM updates. In addition, `GTSAM` library has a larger active community which simplifies the implementation.

3 | AUV NAVIGATION AND PERCEPTION SYSTEMS

We concretize the proposed SLAM system for use on the Sparus II AUV, presented by Carreras et al. (2018), a torpedo-like shape vehicle. It is a nonholonomic vehicle with three degrees of freedom: surge, heave, and yaw. When it is equipped with a mechanical profiling sonar, the depth is fixed and the AUV follows a plain motion with a minimum change in depth. To navigate the robot uses an IMU—from Analog Devices—and a DVL—from Teledyne RDI—pointed at the seabed. The robot has a PC104 embedded computer, which runs an Ubuntu Linux distribution with the Robotic Operation System (ROS) open framework, that manages all systems. A partially open-source control architecture called `COLA2`, presented by Palomeras et al. (2012) and currently distributed by IQUA Robotics S.L. (see Iqua Robotics SL, n.d.), manages all equipment and systems of the vehicle and allows the integration of new software components.

In the following subsections we describe the software components that we implemented to convert sensor measurements to inputs to the Pose SLAM system. Figure 1 shows how these systems are connected and which data are exchanged.

3.1 | Probabilistic dead reckoning system

To integrate the inertial data of the AUV proprioceptive sensors, a probabilistic dead reckoning system is built to get the robot pose expectation and its related uncertainty. As the underwater domain is a GPS-denied environment and Ultra-Short Baseline (USBL) systems are not considered, no absolute sensors are available and no updates to a pose filter can be performed. Therefore, only covariance propagation on sensed velocities can be computed, which implies that robot pose uncertainty can only grow in time. This is equivalent to only performing prediction steps in an EKF. To build this system, some software components available on `COLA2` architecture are used. On the one hand, `COLA2` incorporates a filter which compensates IMU's gyroscopes biases using roll and pitch measures

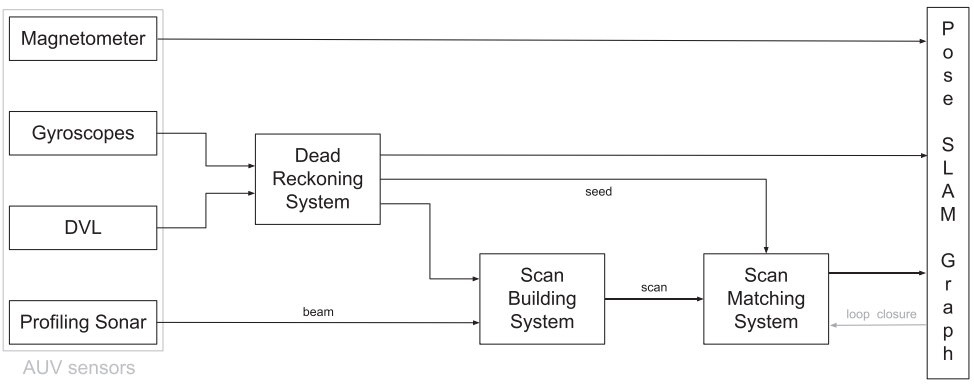

FIGURE 1 Software components that build the Pose SLAM system Front-End. On the left AUV sensors are listed and on the right the factor graph encoding the SLAM problem is shown. AUV, Autonomous Underwater Vehicle; DVL, Doppler Velocity Log; SLAM, simultaneous localization and mapping.

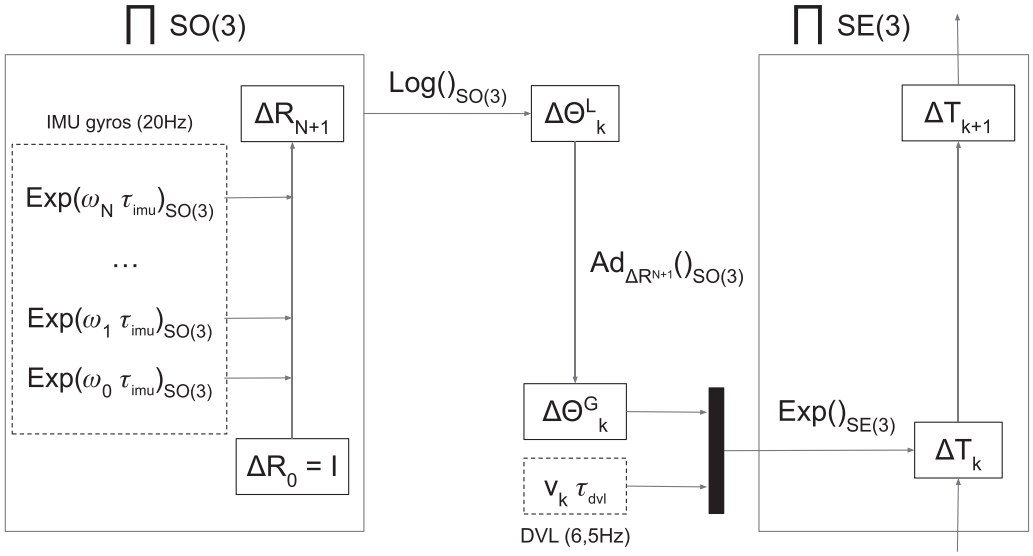

FIGURE 2 Probabilistic dead reckoning system combining inertial measurements from IMUs gyroscopes and DVL working at different rates. Left part is in charge of integrating angular velocities ω giving orientation increments $\Delta\Theta$ represented at the tangent space of the SO(3) group. Right part combines orientation increments with linear velocities v returning AUV pose increments ΔT represented on the SE(3) group.
AUV, Autonomous Underwater Vehicle; DVL, Doppler Velocity Log; IMU, Special Orthogonal; SE, Special Euclidean; SO, Special Orthogonal.

obtained from IMU's accelerometers and yaw measures obtained from IMU's magnetometer, previously corrected from soft and hard iron. On the other hand, `COLA2` implements an EKF which filters DVL measures and returns filtered linear velocities in the robot frame. However, as the IMU works at a higher frequency than the DVL, the dead reckoning system is divided into two components to avoid losing information. As shown in Figure 2, first, a faster system—shown in the left block of the figure—is in charge of performing the integration of the angular velocities ω measured by IMU's gyroscopes to get robot orientation increment ΔR between DVL readings. Then, a slower component—shown in the right block of the figure—fuses the integrated orientation increment $\Delta\theta$ between two DVL readings with the linear velocities v measured by the DVL and integrates them jointly to get the AUV pose T given an initial condition T_0. As it can be seen, linear accelerations from IMU's accelerometers are not integrated twice to get the robot position due to the noisy nature of this sensor.

To build the orientation integrator, unbiased angular velocities $\omega = (\omega_x, \omega_y, \omega_z)^T$ provided by `COLA2` architecture are integrated according to the IMU sampling time τ_{imu}. Following Lie Theory (see Solà et al., 2020), orientation belongs to the Special Orthogonal SO(3) group. Therefore, covariance propagation in this group follows

$$\begin{aligned} R_{i+1} &= R_i \oplus \omega_i \tau_{\text{imu}} \triangleq R_i \operatorname{Exp}(\omega_i \tau_{\text{imu}}), \\ P_{\theta_{i+1}} &= F_{x_i} P_{\theta_i} F_{x_i}^T + F_{u_i} P_{\text{imu}} F_{u_i}^T, \end{aligned} \tag{1}$$

where

$$\begin{aligned} F_{x_i} &= \frac{\partial R_{i+1}}{\partial R_i} = \operatorname{Ad}^{-1}_{\operatorname{Exp}(\omega_i \tau_{imu})} \triangleq \operatorname{Exp}(\omega_i \tau_{\text{imu}})^T, \\ F_{u_i} &= \frac{\partial R_{i+1}}{\partial \omega_i} = J_r(\omega_i \tau_{\text{imu}}). \end{aligned}$$

R_i is a 3D rotation matrix which represents robot orientation, P_{θ_i} is a covariance matrix expressing the uncertainty of the Rodrigues parameters θ_i related to the rotation matrix R_i and P_{imu} is the angular velocity covariance matrix. $\theta = (\theta_1, \theta_2, \theta_3)^T$ parameterizes the tangent space of the SO(3) group and $\operatorname{Exp}(\theta)$ and $\operatorname{Ad}_{\operatorname{Exp}(\theta)}$ are, respectively, the exponential map and the adjoint matrix of the group defined as

$$\operatorname{Exp}(\theta) = \operatorname{Ad}_{\operatorname{Exp}(\theta)} = I_{3\times 3} + \frac{\sin(\|\theta\|)}{\|\theta\|}[\theta]_\times + \frac{1 - \cos(\|\theta\|)}{\|\theta\|^2}[\theta]_\times^2, \tag{2}$$

where

$$\|\theta\|^2 = \theta_1^2 + \theta_2^2 + \theta_3^2,$$
$$[\theta]_\times = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}.$$

$J_r(\theta)$ is the right Jacobian of the SO(3) group defined in Appendix B of Solà et al. (2020).

When the pose integrator receives a DVL measurement, the orientation integrator is reset to the identity element of the SO(3) group $\Delta R_0 = I_{3\times 3}$. Then, the received angular velocities ω_i—with $i = 0, \ldots, N$—are integrated till a new DVL reading is available. At this point, the orientation state ΔR_{N+1}—which quantifies the orientation increment between two DVL readings—is projected to the tangent space to get the rotation increment using Rodrigues parameters

$$\Delta\theta^L = \operatorname{Log}(\Delta R_{N+1}). \tag{3}$$

$\operatorname{Log}(R)$ is the logarithm map of the SO(3) group

$$\text{Log}(R) = \frac{\|\theta\|}{2\sin\|\theta\|}(R - R^T)^V \tag{4}$$

with

$$\|\theta\| = \cos^{-1}\left(\frac{\text{trace}(R) - 1}{2}\right)$$

and the vee operator $()^V$ is the inverse of the hat operator $[]_\times$. The rotation increment $\Delta\theta^L$ and its corresponding uncertainty $P_{\theta_{N+1}}$ are defined in the local frame where the last integration was performed. However, the DVL reading v_k and its related uncertainty are defined in the frame where the orientation integration started (or the frame of x_k) which corresponds with the Lie algebra, as orientation integration started at the identity of the group. To map the orientation increment and its corresponding uncertainty between these two frames, the adjoint matrix $\text{Ad}_{\text{Exp}(\theta)}$ of the SO(3) group is used (see Solà et al., 2020):

$$\begin{aligned}\Delta\theta^G &= \text{Ad}_{\text{Exp}(\Delta\theta^L)}\Delta\theta^L,\\ P_{\Delta\theta^G} &= \text{Ad}_{\text{Exp}(\Delta\theta^L)} P_{\theta_{N+1}} \text{Ad}^T_{\text{Exp}(\Delta\theta^L)},\end{aligned} \tag{5}$$

where $\text{Ad}_{\text{Exp}(\theta)}$ is the adjoint matrix defined in Equation (2). However, assuming some properties of the SO(3) group

$$\begin{aligned}\text{Exp}(\text{Ad}_{\text{Exp}(\Delta\theta^L)}\Delta\theta^L) &= \text{Exp}(R\Delta\theta^L) = R\,\text{Exp}(\Delta\theta^L)R^T = RRR^T = R\\ &= \text{Exp}(\Delta\theta^L),\end{aligned}$$

which implies that for this Lie Group $\Delta\theta^G = \Delta\theta^L$ and no map is needed.

Once we are sure that DVL measurements and gyroscopes integrations are in the same frame, pose can be integrated. To do it, filtered linear velocities $v = (u, v, w)^T$ provided by `COLA2` architecture and orientation increments $\Delta\theta^G = (\Delta\theta_1, \Delta\theta_2, \Delta\theta_3)^T$ are fused by covariance propagation in the Special Euclidean SE(3) according to the DVL sampling time τ_{dvl}:

$$\begin{aligned}T_{k+1} &= T_k \oplus u_k \triangleq T_k\,\text{Exp}(u_k),\\ P_{\eta_{k+1}} &= F_{x_k} P_{\eta_k} F^T_{x_k} + F_{u_k} P_{u_k} F^T_{u_k},\end{aligned} \tag{6}$$

where

$$\begin{aligned}u_k &= \left(v_k\tau_{\text{dvl}}, \Delta\theta^G_k\right)^T,\\ P_{u_k} &= \begin{bmatrix}P_{\text{dvl}} & 0_{3\times3}\\ 0_{3\times3} & P_{\Delta\theta^G}\end{bmatrix},\\ F_{x_k} &= \frac{\partial T_{k+1}}{\partial T_k} = \text{Ad}^{-1}_{\text{Exp}(u_k)},\\ F_{u_k} &= \frac{\partial T_{k+1}}{\partial u_k} = J_r(u_k).\end{aligned}$$

T_k is a four-dimensional homogeneous transformation matrix that represents robot pose, P_{η_k} is a covariance matrix expressing the uncertainty of the pose parameters η_i related to the transformation matrix T_k, P_{dvl} is the linear velocity covariance matrix obtained from `COLA2` velocity EKF and $P_{\Delta\theta^G}$ is the orientation covariance matrix defined in Equation (5). $u_k \triangleq \eta = (\rho_1, \rho_2, \rho_3, \theta_1, \theta_2, \theta_3)$ parameterizes the tangent space of the SE(3) group and $\text{Exp}(\eta)$ and $\text{Ad}_{\text{Exp}(\eta)}$ are, respectively, the exponential map and the adjoint matrix of the group defined as

$$\begin{aligned}\text{Exp}(\eta) &= \begin{bmatrix}\text{Exp}(\theta) & J_r(\theta)^T\rho\\ 0 & 1\end{bmatrix},\\ \text{Ad}_{\text{Exp}(\eta)} &= \begin{bmatrix}\text{Exp}(\theta) & [J_r(\theta)^T\rho]_\times\text{Exp}(\theta)\\ 0 & \text{Exp}(\theta)\end{bmatrix},\end{aligned} \tag{7}$$

where $\text{Exp}(\theta)$ and $J_r(\theta)$ are, respectively, the exponential map and the right Jacobian of the SO(3) group defined in Equation (2). $J_r(\eta)$ is the right Jacobian of the SE(3) group defined in Appendix D of Solà et al. (2020). As it is seen, the complete SE(3) group is considered. However, as pose increments u_k are small, the approximation "rotation then translation" of the group could be considered to save computational resources.

The implementation of this dead reckoning system is done in an ROS node. To evaluate the exponential maps, logarithm maps, adjoint matrices, and Jacobians of each group we use `Manif` library available in Deray and Solà (n.d.).

3.2 | Sonar scan building system

The AUV is equipped with a Super SeaKing Profiler (see "Tritech International Ltd., Super SeaKing Profiler", n.d.)—a mechanical profiling sonar from Tritech, Westhill, Scotland—to perform the robot surroundings perception task. It is assembled on the AUV with its Field of View (FoV) parallel to the robot plane of motion and pointing in the longitudinal direction. A profiler sonar is formed by a single beam with a narrow vertical aperture angle (around 1º), which contrasts with the bigger vertical aperture of the Forward-Looking Sonar. The beam is mechanically actuated and rotates on a plane. It is composed of an array of bins that encode the distance from the transceiver and the received sound intensity. Taking the bin with the higher intensity per beam in a complete turn, an acoustic point cloud can be built.

As mechanical waves are taken as the working principle of sonar sensors, they are very slow to take a complete scan in comparison to LIDAR sensors. Sound speed in the water is around 1.52×10^3 m/s, compared with the 2.25×10^8 m/s of light. This implies that the Profiling Sonar takes some seconds to get a complete scan. As the AUV is in continuous motion, the AUV navigation data obtained from the dead reckoning system must be fused with each sonar beam to prevent the scan from warping. However, a dead reckoning system always drifts over time and sonar perception is proposed to mitigate it. To avoid this contradiction the following hypothesis is taken: the AUV drift is imperceptible in the scan scale, however, drift appears in the robot trajectory scale. Moreover, acoustic devices, by nature, are more noisy compared with the electromagnetic ones. This implies that sonar sensors return a larger quantity of noise and outlier detections that must be properly treated.

To fuse sonar detections with robot motion, the robot pose ${}^{W}T_{k}$ given by the dead reckoning system is stored for each beam. When the scan is finished, all points are projected to the robot pose ${}^{W}T_{k_{\text{ref}}}$ where the last beam is received applying

$$ {}^{k_{\text{ref}}}p_i = {}^{W}T_{k_{\text{ref}}}^{-1}\,{}^{W}T_k\,{}^{R}T_S \begin{bmatrix} {}^{S}d_i \cos {}^{S}\alpha_i \\ {}^{S}d_i \sin {}^{S}\alpha_i \end{bmatrix}, \tag{8} $$

where $\left({}^{S}d_i, {}^{S}\alpha_i\right)$ is a polar detection measured in the sonar frame, ${}^{k_{\text{ref}}}p_i$ is the Cartesian position of the detection in the built scan and ${}^{R}T_S$ is the pose of the Profiling Sonar frame in the robot frame. Following this formulation, every scan is identified at the robot pose where it ends ${}^{W}T_{k_{\text{ref}}}$.

3.3 | Scan matching system

Once scans are formed by the scan building system, the next step is to register overlapping scans—not necessarily consecutive—to get pose constraints through the robot trajectory. In this section, we present a new fields-based technique that takes as reference NDT methods but goes deep into the language of GMMs. Using a field representation of the scan, it is possible to model the sensor noise and reject outlier detections making the registration algorithm robust. This is very important in the underwater domain, as acoustic data are sparse and very noisy. Moreover, a method to evaluate the registration uncertainty is given, which is not common in most scan matching techniques.

In the following subsections the proposed scan matching algorithm is described. For the purpose of clarity, it is split into components detailed in the following subsections.

3.3.1 | Gaussian mixtures Front-End

The objective of the Gaussian mixtures Front-End is to fit a GMM to a given point cloud, converting discrete observations into a continuous representation. A Gaussian mixture distribution can be written as a linear superposition of Gaussians in the form

$$ p(x|\Phi) = \sum_{k}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \;\text{ with }\; \sum_{k}^{K} \pi_k = 1, \tag{9} $$

where the model parameters $\Phi = (\pi_k, \mu_k, \Sigma_k)$ are, respectively, the weight of each component and the mean and covariance of each Gaussian distribution. To fit a GMM into a data set $\{x_1, \ldots, x_n\}$ of N observations, we need to define a latent variable z_i for each observation x_i which assigns each observation x_i to a cluster k. Thus, z_i is a K-dimensional binary random variable having a 1-of-K representation where $z_{i,k} \in \{0, 1\}$, $\sum^{K} z_{i,k} = 1$ and its density is given by

$$ p(z|\Phi) = \prod_{k}^{K} \pi_k^{z_k}. \tag{10} $$

This theoretical framework can be represented by the graphical model of Figure 3a. Following this model, the joint distribution of x and z is given by

$$ \begin{aligned} p(x, z|\Phi) &= p(x|z, \Phi)p(z|\Phi) = \prod_{k}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k} \prod_{k}^{K} \pi_k^{z_k} \\ &= \prod_{n}^{N} \prod_{k}^{K} \pi_k^{z_{nk}} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_{nk}} \end{aligned} \tag{11} $$

and the marginal distribution over the observed data set is obtained by summing the joint distribution over all possible states of z, which gives the distribution defined in Equation (9).

As it was discussed in Section 2.1, NDT methods from Biber and Strasser (2003) and Stoyanov et al. (2012) fit a GMM into a point cloud using a Cartesian grid. For each cell with a minimum number of points assigned, a GMM component is fit using the ML estimator for the Gaussian. This is a simple method to determine the latent variables of the model because the shape of each cluster is defined a priori by the mesh and points are only assigned to the cell where they fall. Therefore, this method does not take into account the implicit structure in the scan coming from the morphology of the scanned surfaces when fitting the model. This is exemplified in Figure 4 left,

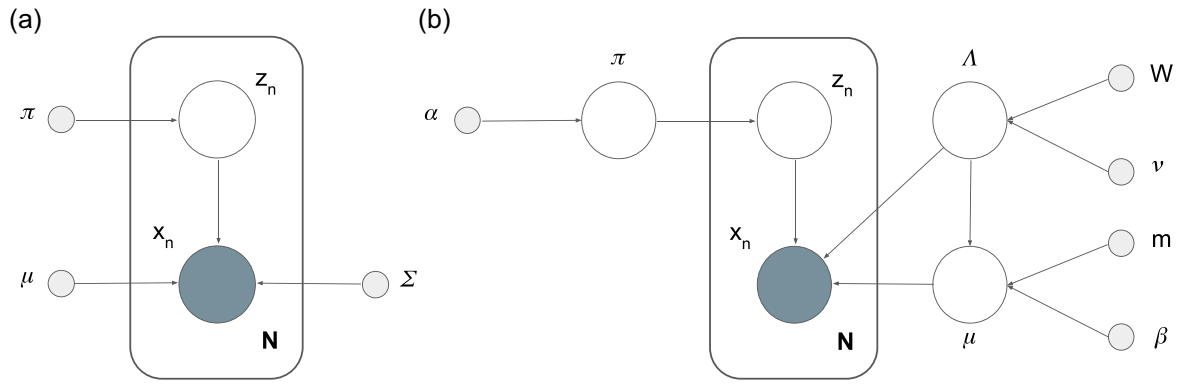

FIGURE 3 Graphical models. (a) GMM for a set of N data points x_n with corresponding latent variables z_n. (b) Bayesian-GMM that sets prior distributions to each model parameter Φ. GMM, Gaussian mixtures model. [Color figure can be viewed at wileyonlinelibrary.com]

where two GMMs are fitted by applying the NDT Front-End on two synthetic scans from a corner and a corridor. As it can be seen, where the mesh does not agree with the walls direction, the GMM does not model well the scan. These phenomena could be amplified in natural environments where rocks form irregular surfaces.

More advanced methods make use of the K-means algorithm (Jiang et al., 2019) or the EM algorithm for Gaussian mixtures (Tabib et al., 2018) to fit a GMM to a given scan taking into account its intrinsic structure. K-means algorithm, presented in Lloyd (1982), is a heuristic solution to the ML estimator for the probabilistic model shown in Figure 3 left; whereas the EM algorithm for a GMM, presented in Dempster et al. (1977), is its optimal solution. However, no analytical solution for this estimator is available due to the summation over the latent variables

$$\begin{aligned}\Phi^* &= \arg\min_{\Phi} \ln \prod^{N} p(x|\Phi) = \arg\min_{\Phi} \ln \prod^{N} \left\{ \sum^{Z} p(x, z|\Phi) \right\} \\ &= \arg\min_{\Phi} \sum^{N} \ln \left\{ \sum^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \right\}\end{aligned} \tag{12}$$

and an EM procedure is needed. Conceptually, these algorithms differ in the assignations done by the latent variables. While K-means algorithm makes hard assignations of each observation to a single cluster, EM algorithm sets a posterior distribution to each latent variable once the observation is done, called responsibility, which allows one to do soft assignations of each observation to all clusters. Nevertheless, both algorithms share that the number of clusters K is a predefined parameter that is not learned during the fitting process. Moreover, both algorithms try to fill all components with data, which can drive the estimation process to a singularity when a component has only one assignation, normally an outlier. In Figure 4 it is shown that using the EM algorithm, GMM components explain better the walls than applying the NDT Front-End. However, using this algorithm models of constant size are fitted, opposed to the NDT Front-End where a fixed cell size is set and the model size depends on the number of filled cells.

To heuristically learn the K parameter to avoid unnecessary components that could drive to a singularity, in Eckart et al. (2016) a hierarchical EM method is proposed to soft partition the initial clusters K_0 until no improvement in the model likelihood is reached. However, the Bayesian-GMM, proposed by Attias (2000), is the optimal solution to this problem and it is proposed as the Gaussian mixtures Front-End in this paper. As appointed before, the EM algorithm for a GMM is based on an ML estimator. In contrast, a Bayesian-GMM sets prior distributions to the GMM parameters Φ according to the graphical model in Figure 3b and, therefore, the fitting process is based on a MAP estimator. To favor analytical treatment of the joint distribution, conjugate priors are taken. As weights π_k are priors for a multinomial distribution, a Dirichlet distribution is taken. As means μ_k and covariances Σ_k are priors for a normal distribution, a Gaussian–Whishart distribution is set. Taking into account this, the joint distribution for the Bayesian-GMM is

$$p(x, z, \pi, \mu, \Lambda) = p(x|z, \mu, \Lambda^{-1}) p(z|\pi) p(\pi) p(\mu, \Lambda), \tag{13}$$

where $\Lambda_k = \Sigma_k^{-1}$ is called the information matrix, $p(x|z, \mu, \Lambda^{-1})$ and $p(z|\pi)$ are given in Equation (11) and

$$\begin{aligned}p(\pi_k) &\sim \mathcal{D}ir(\pi_k|\alpha_{0_k}), \\ p(\mu_k, \Lambda_k) &\sim \mathcal{N}\left(\mu_k|m_{0_k}, (\beta_{0_k}\Lambda_k)^{-1}\right)\mathcal{W}(\Lambda_k|W_{0_k}, \nu_{0_k}).\end{aligned}$$

As it can be seen the parameters for the new probabilistic model are $\Phi = (\alpha_{0_k}, m_{0_k}, \beta_{0_k}, W_{0_k}, \nu_{0_k})$.

As in the previous model, the marginal distribution over the observations is not analytically treatable due to the summation over

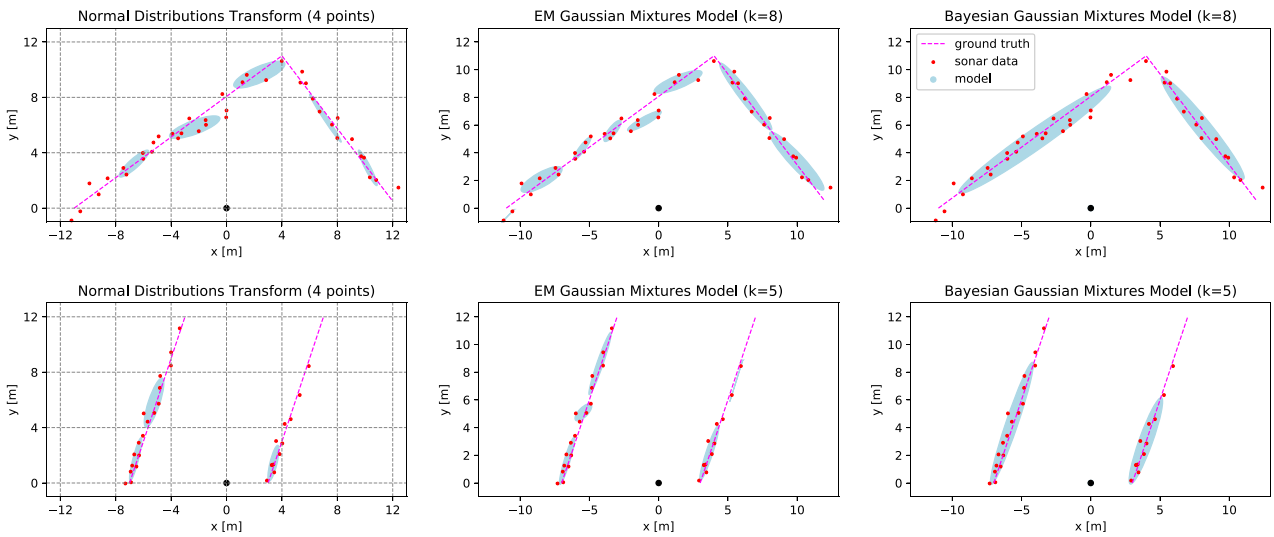

FIGURE 4 Comparison of the three Gaussian mixtures Front-Ends applied on two synthetic scans (up, corner; down, corridor). (Left) NDT using square cells of 4 m and a minimum of four points to set a component. (Center) EM algorithm used to learn a GMM of K components. (Right) Bayesian-GMM learned using K components. EM, Expectation Maximization; GMM, Gaussian mixtures model; NDT, Normal Distributions Transform. [Color figure can be viewed at wileyonlinelibrary.com]

the latent variables. Thus, the MAP estimator must be solved iteratively applying the EM algorithm. However, the joint distribution $p(x, z|\Phi)$ is also not analytically treatable and the following factorization must be imposed:

$$q(z, \pi, \mu, \Lambda) = q(z)q(\pi, \mu, \Lambda) \tag{14}$$

to solve the problem by means of Variational Inference. Following the algorithm proposed in Attias (2000), the model parameters Φ can be found iteratively; first, evaluating the responsibility of each data point into each model component and, then, solving the MAP estimator $\Phi^* = \arg\min_{\Phi} \ln \prod^N p(x|\Phi)$ imposing the factorization from Equation (14).

Thanks to modeling the component weights π_k with a Dirichlet distribution, the components that provide an insufficient contribution to explaining the data set have their weights driven to zero without generating any singularity to the optimization process. The algorithm starts setting the mean of all K_0 components to the point cloud mean and the weights at $\frac{1}{K_0}$. Then, the algorithm starts the clustering process by moving the means away from the point cloud mean and discards useless components by setting its weight to a small number close to zero. Therefore, this approach allows one to infer the optimal number of components K to model the data set, given an upper bound K_0 for this parameter. These properties are shown in Figure 4 right where two Bayesian-GMMs of two components are learned, reflecting the morphology of a corner and a corridor in the simplest way. Both models have been initialized with eight components and six of them have been removed by the fitting process, contrasting to the EM algorithm (Figure 4 center) that fills all components. Using the EM algorithm, although the scan structure is captured, redundant components are set splitting the walls in an excessive number of components and returning a model with unnecessary complexity. To initialize the Bayesian-GMM algorithm, we propose to first run K-means algorithm with the upper bound K_0 and a random initialization. Then, the result is used to run the Variational Inference algorithm for Gaussian mixtures to refine the parameters Φ and get the optimal K.

The use of a Bayesian-GMM Front-End supposes to increase the computational complexity of the fitting process. However, the NDT and EM Front-Ends have been conceived for applications with LIDAR sensors or depth cameras, which produce denser point clouds at a faster rate than acoustic sensors. Taking advantage of the sparsity of an acoustic point cloud and the slower velocity of mechanical waves, experiments show that Bayesian-GMMs can be learned in real time. Also, the advanced theoretical complexity of this approach helps better model the morphology of the robot surroundings using noisier data with less information content.

3.3.2 | Registration policies for the Gaussian mixtures formulation

The registration problem solves the robot displacement between two overlapping scans taken by a range sensor mounted on a robot. The oldest scan, called the fixed scan $\mathcal{F}$, is taken as the reference and the current scan, called the moving scan $\mathcal{M}$, is moved to solve the registration by optimizing some policy. Thanks to the field representation of the scan, the registration problem can be formulated as an optimization problem and solved by applying gradient-based methods. The optimization variable Ω is the pose increment between both viewpoints defined by a rotation matrix R and a translation vector t. The problem restrictions are those related to pose given by the Lie Theory (see Solà et al., 2020). In this section, two cost functions applied to NDT techniques are reviewed for a purely Gaussian mixtures formulation.

Given a fixed scan $\mathcal{F}$ represented by a GMM $p(x|\Phi_{\mathcal{F}})$ and a moving scan $\mathcal{M}$ given in point cloud form $\mathcal{D}_{\mathcal{M}} = \{q_1, \ldots, q_n\}$ whose relative displacement is defined by a pose constraint $\Omega = (t, R)$; the P2D method presented by Biber and Strasser (2003) finds the ML solution of $\mathcal{D}_{\mathcal{M}}$ transformed by Ω into $p(x|\Phi_{\mathcal{F}})$:

$$\begin{aligned} p(\mathcal{D}_{\mathcal{M}}|\Phi_{\mathcal{F}}, \Omega) &= \prod^{\mathcal{D}} p(x = Rq_d + t|\Phi_{\mathcal{F}}) \\ &= \prod^{\mathcal{D}} \sum^{K} \pi_k \mathcal{N}(x = Rq_d + t|\mu_k, \Sigma_k). \end{aligned} \tag{15}$$

To solve the ML problem applying gradient-based methods, the log-likelihood is minimized and, as it was proposed in Magnusson (2009), to favor the analytic treatment of derivatives, the logarithm of the Gaussian mixtures is approximated by another GMM

$$\begin{aligned} \Omega^* &= \arg\min_{\Omega} \ln p(\mathcal{D}|\Phi_{\mathcal{F}}, \Omega) = \arg\min_{\Omega} \sum^{\mathcal{D}} \ln p(x = Rq_d + t|\Phi_{\mathcal{F}}) \\ &\sim \arg\min_{\Omega} - \sum^{\mathcal{D}} \sum^{K} \pi_k \mathcal{N}(x = Rq_d + t|\mu_k, \Sigma_k). \end{aligned} \tag{16}$$

Using the NDT technique, where $p(x|\Phi_{\mathcal{F}})$ is fitted using a Cartesian grid, the weights π_k are modeled as a multinomial distribution where $\pi_{kd} = 1$ only for the component k of the cell of $\mathcal{F}$ where the point q_d of $\mathcal{M}$ falls. This is proposed to save computational resources for dense scans. However, when using acoustic sensors, it is possible to evaluate the whole model, reaching a continuous objective function which makes easier the convergence of the optimization problem. In Appendix A the analytical form of the objective function and its derivatives can be found.

Given two GMMs, one modeling a fixed scan $p(x|\Phi_{\mathcal{F}})$ and the other modeling a moving scan $p(x|\Phi_{\mathcal{M}}, \Omega) = p(x|\pi_{\mathcal{M}}, R\mu_{\mathcal{M}} + t, R\Sigma_{\mathcal{M}}R^T)$ whose relative displacement is defined by a pose constraint $\Omega = (t, R)$; the $\mathcal{L}^2$ distance between both distributions is defined as

$$\begin{aligned} \mathcal{L}^2 = \int (p(x|\Phi_{\mathcal{F}}) - p(x|\Phi_{\mathcal{M}}, \Omega))^2 dx = \int p(x|\Phi_{\mathcal{F}})^2 dx \\ + \int p(x|\Phi_{\mathcal{M}}, \Omega)^2 dx - 2\int p(x|\Phi_{\mathcal{F}}) p(x|\Phi_{\mathcal{M}}, \Omega)\, dx. \end{aligned} \tag{17}$$

Thus, the D2D method proposed by Stoyanov et al. (2012) finds the displacement that maximizes the $\mathcal{L}^2$ distance between both distributions. As it was noted by Jiang and Vemuri (2011), as the first two terms from Equation (17) remain constant during the optimization and the integral of the third term has a closed form for the Gaussian distribution, the cost function has the following analytical form:

$$\Omega^* = \arg\max_{\Omega} \mathcal{L}^2(\Omega) = \arg\max_{\Omega} \sum_{i=1}^{n_{\mathcal{F}}} \sum_{j=1}^{n_{\mathcal{F}}} \pi_{\mathcal{F}i} \pi_{\mathcal{M}j} \mathcal{N}(0 | \mu_{\mathcal{F}i} - R\mu_{\mathcal{M}i} - t, \Sigma_{\mathcal{F}i} + R\Sigma_{\mathcal{M}i} R^T). \quad (18)$$

Contrary to the P2D method, now, the normalization factor for the Gaussian distribution depends on the optimization variables Ω, as the determinant of the covariance matrix is affected by the rotation matrix. As shown in Tabib et al. (2018), maintaining this dependency increases hugely the analytical complexity of the derivatives when solving the optimization by applying gradient descent methods. To avoid this dependency, our results show that the approximation of the normalization factor by a constant d_1—as proposed in Stoyanov et al. (2012)—is enough to get good matching results when using acoustic data.

Using the NDT technique, one-to-one correspondences are established between scan components, what implies a nearest neighbor search in $\mathcal{F}$ for each component in $\mathcal{M}$. However, in Tabib et al. (2018) it is suggested that, as the GMM representation is a way of compressing the point cloud data, it is possible to evaluate the whole correspondences making the optimization problem continuous. In Appendix B the analytical form of the objective function and its derivatives can be found.

3.3.3 | Scan matching Back-End

The purpose of the scan matching Back-End is to solve the formulated optimization problems. As both optimization policies are continuous and analytic derivatives can be computed, the problem is solved applying descent methods based on the following iterative procedure to the objective variables

$$\Omega_{k+1} = \Omega_k \oplus \alpha_k d_k \quad (19)$$

where $\alpha_k \in \mathbb{R}$ and $d_k \in \mathbb{R}^n$ are, respectively, called steep size and direction. We propose to use the Newton Method as it is the optimal descent method. This method sets the direction as $d_k = -H_k^{-1} J_k$, where J_k and H_k are, respectively, the Jacobian vector and the Hessian matrix of the cost function evaluated in the current solution Ω_k. Moreover, to deal with the nonlinearities of the problem, a line search algorithm based on the Wolfe conditions—Algorithm 11.5 from Bierlaire (2015)—is added to set a step size that favors minimization for each optimization iteration. Also, to ensure a minimization direction, the Gill, Murray and Wright modified Cholesky factorization algorithm—Algorithm MC form (Gill et al., 1981)—is used to invert the Hessian matrix H_k, perturb it to get a definite positive matrix if it is an indefinite matrix and ensure that the resulting matrix is reasonably well conditioned. Using these tools the convergence of the algorithm is achieved in most practical scenarios.

To deal with the Lie group that defines pose, it is parameterized using the composite manifold $<\mathbb{R}^n, SO(n)>$ formed by the concatenation of an n-dimensional space representing position and the Special Orthogonal $SO(n)$ group representing orientation. Using a composite manifold, instead of working at the $SE(n)$ group, derivatives can be computed by blocks, separating rotation from translation (see Solà et al., 2020). For example, the derivative of the group action function $f(\Omega, q) : <\mathbb{R}^n, SO(n)>, \mathbb{R}^n \rightarrow \mathbb{R}^n = Rq + t$, which is applied in both optimization policies, can be computed as

$$\frac{Df(\Omega, q)}{D\eta} = \begin{bmatrix} \frac{Df(R,t)}{Dp} & \frac{Df(R,t)}{D\theta} \end{bmatrix} = \begin{bmatrix} \frac{Dt}{Dp} & \frac{DR}{D\theta} q \end{bmatrix}, \quad (20)$$

where $\eta = (p, \theta)$ defines a perturbation in a tangent space of the composite manifold. This block structure of the derivatives favors the analytic simplicity of the first and second derivatives of the optimization policies.

To perform the optimization updates, the pose manifold must be considered and the plus operator $\oplus$ for the $<\mathbb{R}^n, SO(n)>$ must be defined. As using this parameterization rotation is independent of translation, the addition of a perturbation $\eta = (p, \theta)$ to a pose $\Omega = (t, R)$ can be computed by applying

$$\Omega_k \oplus \eta = \begin{bmatrix} t + p \\ R Exp(\theta) \end{bmatrix}. \quad (21)$$

One of the benefits of applying a second-order solver is that there are available evaluations of the Hessian matrix of the problem. Bengtsson and Baerveldt (2003) suggested that the Hessian matrix gives information about the shape of the cost function at the point where it is evaluated. This means that computing the Hessian matrix at the optimum, it is a measure of the uncertainty of the registration result. Censi (2007) suggested that the uncertainty of a match is also related to the noise of the sensor used to build the scan. Nevertheless, Stoyanov et al. (2012) empirically proved that both approaches for the D2D method give the same uncertainty shape, only with different sizes. Therefore, we conclude that the covariance matrix of a registration is proportional to the Hessian matrix computed at the optimum. This constitutes an automatic method to recover a particular uncertainty for each match according to the optimization process. This result is not common in many scan matching techniques, as solvers are only based on first derivatives and Hessian evaluations are not available. Moreover, this is very interesting in SLAM problems because it constitutes an automatic tool to separate good matches from those that are not so good and, also, detect for each particular match in which direction the result is better in comparison to the others.

As cost function derivatives are defined in the composite manifold, the covariance matrix related to each match is also defined in the $<\mathbb{R}^n, SO(n)>$. However, the majority of the SLAM frameworks—such as the `GTSAM` library available in Dellaert (n.d.)—are defined in the $SE(n)$ group. Thus, a map between both parameterizations is needed. Moreover, its Jacobian matrix lets relate their uncertainties applying

$$\Sigma_{SE(n)} = J(\Omega)\Sigma_{<\mathbb{R}^n,SO(n)>}J(\Omega)^T, \quad (22)$$

where $\Sigma_{<\mathbb{R}^n,SO(n)>}$ is a covariance matrix defined in the composite manifold, $\Sigma_{SE(n)}$ is a covariance matrix defined in the SE(n) group and $J(\Omega)$ is the Jacobian matrix given by Proposition 1. As it is seen, we suggest to solve the registration problem in the composite manifold to simplify the analytic computation of derivatives. However, an extra steep is needed to map the registration covariances from the composite manifold to the SE (n) group.

Proposition 1. *Given a pose* $\Omega \in <\mathbb{R}^n, SO(n)>$ *and a map function* $h(\Omega) : <\mathbb{R}^n, SO(n)> \rightarrow SE(n)$, *its Jacobian matrix is*

$$J(\Omega) = \frac{\partial h(\Omega)}{\partial \Omega} = \begin{bmatrix} -R^T & O_{n\times n} \\ O_{n\times n} & I_{n\times n} \end{bmatrix}.$$

The proof of this derivative is given in Appendix C.

Finally, the defined optimization problems are convex but suffer from local minima because the Newton Method does not guarantee global minima. Thus, good seeds for the optimization problem are needed to not fall in local optimums. To do so, information from other components of the proposed system is used. To register successive scans, seeds from the dead reckoning system are taken. When attempting loop closure, information comes from the Pose Graph.

3.3.4 | Scan matching algorithm

In this paper a scan registration formed by a double match is proposed combining the strengths of P2D and D2D policies. As stated before, the D2D method minimizes the divergence between two distributions, whereas the P2D method measures the likelihood of a point cloud to a model. This means that D2D has more attraction force when two scans are very misaligned. In contrast to D2D policy, the P2D likelihood function tends to zero very fast when some point gets far from a component and its derivatives, which are the engine to the solver, become negligible. Moreover, the P2D method uses the full data set for the moving scan and a more accurate match can be reached when the seed is near to the optimum, as only the reference scan is compressed by a GMM. However, the price to pay is the computational cost of evaluating derivatives for the whole point cloud. Nevertheless, when using acoustic data, the aim of the GMM representation is not data compression but modeling perception noise. So it is possible to spend computational resources solving a P2D method, since point clouds are sparser. Therefore, following Algorithm 1, first, a D2D match with an external seed is attempted, to bring closer the scans solving huge misalignment. Second, a P2D match with the D2D result as seed is performed to get a more accurate result.

Algorithm 1. Scan matching algorithm

```
Require: reference_scan, current_scan, pose_seed, pose_seed_cov
  reference_gmm = BayesianGmmFrontEnd(reference_scan)
  current_gmm = BayesianGmmFrontEnd(current_scan)
  p2d_method = PointsToDistribution(reference_gmm, current_scan)
  d2d_method = DistributionToDistribution(reference_gmm,
    current_gmm)
  d2d_solver = CholeskyLineSearchNewtonMethod(d2d_method)
  p2d_solver = CholeskyLineSearchNewtonMethod(p2d_method)
  d2d_solver.compute_optimum(pose_seed)
  if d2d_solver.has_converged() then
    p2d_solver.compute_optimum(d2d_solver.get_optimal())
    if p2d_solver.has_converged() then
      return p2d_solver.get_optimal(), p2d_solver.get_optima_cov()
    else
      return d2d_solver.get_optimal(), d2d_solver.get_optima_cov()
    end if
  else
    p2d_solver.compute_optimum(pose_seed)
    if p2d_solver.has_converged() then
      return p2d_solver.get_optimal(), p2d_solver.get_optima_cov()
    else
      return pose_seed, pose_seed_cov
    end if
  end if
```

The implementation of the GMM Front-End, registration policies, and scan matching Back-End are taken from the open-source library `GMM Registration` available in Vial (n.d.). This library was presented by the same authors in the conference paper (Vial et al., 2023), where the speed and convergence of the scan matching algorithm are characterized in comparison to other state-of-the-art registration techniques.

4 | UNDERWATER POSE SLAM SYSTEM

Combining the described dead reckoning, scan building, and scan matching systems a Pose SLAM problem is formulated. This information is used to build a factor graph, constituting the SLAM Front-End. Using the `iSAM2` solver—presented in Kaess et al. (2012) and provided by the `GTSAM` library available in Dellaert (n.d.)—the Pose SLAM problem is solved incrementally while the Front-End builds the graph, reaching an online estimator for the robotic application.

As a mechanical profiling sonar provides 2D information in the form of plain scans, only layer maps can be built corresponding to horizontal sections of a 3D space. Therefore, the AUV equipped with this sensor has to follow a plain motion with minimal changes in depth. Thus, a Pose SLAM problem, formulated in the SE(2) group, is proposed to estimate the whole robot trajectory. Each pose is defined as $\vec{x} = (x, y, \varphi)^T$ being x and y the position of the robot center of gravity in the world frame and φ the z-axis orientation of the robot in the world frame. The other degrees of freedom of the robot are considered constant as a plain motion to the AUV is imposed. The Pose SLAM problem is modeled by a factor graph,

where nodes represent the robot poses where each sonar scan is referenced. The factors linking these nodes and the methodology to add them to the factor graph are detailed below.

4.1 | Pose SLAM Front-End

First, if the dead reckoning system is reset every time a new scan starts, it provides an SE(3) constraint between two successive scans composed of an expectation vector and a covariance matrix. However, to set an inertial factor in the pose graph, the SE(3) constraint must be projected into the SE(2) group as the Pose SLAM problem is formulated in this last group. The projection can be done by marginalizing the x and y components of the translation and taking the yaw rotation of the ZYX Euler angles parameterization, as in this Euler angles combination the z rotation is performed first in the fixed frame. To project the rotation, the SO(3) group is mapped into quaternions and then into ZYX Euler angles following Blanco (2010). Keeping only the z rotation, the projected expectation is

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ a\tan 2\left(\frac{\theta_3 \| \theta \| \sin \| \theta \| + \theta_1\theta_2(1-\cos \| \theta \|)}{\theta_1^2 + \cos \| \theta \| \left(\theta_2^2+\theta_3^2\right)}\right) \end{bmatrix} \tag{23}$$

where θ_i are the Rodrigues parameters for the 3D rotation, t_i are the 3D translation components and pitch $\neq\{\pi, -\pi\}$ is supposed as it is an impossible configuration for the Sparus II AUV. Assuming the plain motion imposed to the AUV—which means that roll, pitch, and depth variations are negligible—the projected rotation can be approximated as $\varphi \sim \theta_3$. Assuming this approximation, the SE(3) covariance $P_{SE(3)}$ can be projected into the SE(2) group applying

$$P_{SE(2)} \sim J_{proj} P_{SE(3)} J_{proj}^T, \quad \text{where } J_{proj} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{24}$$

and its demonstration is given in Appendix D.

Second, the scan matching system provides an SE(2) constraint between two nodes for which the registration problem has been solved successfully. This constraint consists of an expectation vector and a matrix proportional to its uncertainty. Using this information a scan matching factor between registered nodes can be set in the factor graph, given a proportional constant r for the uncertainty. This factor is used to register successive scans, with r_{sm} as the uncertainty constant. It is also used for loop closure, applying r_{lc}.

Finally, absolute rotation measures obtained from the IMU magnetometer—compensated from soft and hard iron by the COLA2 architecture—are also added to the factor graph as a Special Orthogonal SO(2) before each node. As an orientation expectation is just a sensor reading, its uncertainty R_{mag} is constant for all nodes.

4.2 | SLAM methodology

Given the described factors, a factor graph is built using the `GTSAM` library. Every time a new sonar scan is available, the factor graph is extended and solved. Algorithm 2 describes the proposed incremental methodology, suitable for a real time application, and Figure 5 shows the structure of the resulting Pose Graph.

Algorithm 2. Pose SLAM algorithm

```
odometry = ProbabilisticDeadReckoning()
scanner = ScanBuilder()
match = GMMScanMatching()
yaw = Magnetometer()
graph = NonlinearFactorGraph()
graph.set_node(1)
graph.set_prior(1, x_0)
graph.set_seed(1, seed)
loop = LoopClosureManager()
i = 0
while true do
  if scanner.scan_available() then
    scan.append(scanner.build_scan())
    dead_reckoning = odometry.get_pose()
    odometry.reset()
    graph.set_node(i)
    graph.set_dead_reckoning_factor(i-1, i, dead_reckoning)
    graph.set_prior(i, yaw)
    match.solve_register(scan(i-1), scan(i), dead_reckoning)
    if match.converged() then
      graph.set_scan_matching_factor(i-1, i, match.get_result())
    end if
    graph.set_seed(i, seed)
    graph.solve_graph()
    if loop.loop_found() then
      k = loop.get_candidate_node()
      match.solve_register(scan(k), scan(i), graph.get_delta_pose(k, i))
      if match.converged() then
        graph.set_scan_matching_factor(k, i, match.get_result())
      end if
    end if
    i++
  end if
end while
```

When a new sonar scan is built, a new node in the factor graph is created. Then, the dead reckoning system is used to set a navigation factor between this node and the previous one and, immediately, it is reset to start building the next sonar scan. Also, an SO(2) before the new node is set using magnetometer measures. Moreover, a registration problem is attempted between the current scan and the previous one, with which

overlapping is theoretically guaranteed. The scan matching solver is initialized taking the dead reckoning measure as seed, to prevent local minima. If the problem converges, a scan matching factor is set on the factor graph between the registered nodes. Finally, and before updating the solution of the factor graph, a seed to the new node is given to the Back-End. The seed is found combining the available solution for the previous node and the dead reckoning measure perturbed with some random noise. As it can be seen in Figure 5, the majority of nodes in the factor graph are chained by two SE(2) factors, one coming from inertial navigation and the other coming from acoustic perception. However, as uncertainty is estimated for both systems, the two constrains are balanced at each particular link in the chain.

Once the graph is solved and a solution for all nodes is available, loop closure is attempted using the positional information of the graph. When the robot revisits some explored region, the old view and the new view can be registered and a scan matching factor can be set closing a loop into the factor graph. However, as scans provide very local information of the surroundings of the robot, we propose to group scans into hyperscans to have wider views with more features to avoid false positives registrations of scans that in reality do not overlap. Using the SLAM solution, hyperscans of constant length referenced to its central node are built and only loops between hyperscans are searched.

Whenever a new hyperscan is available, loop closure candidates are searched. To do it, no cross covariance between nodes is considered. As suggested by Ila et al. (2007), the search is based on the Bhattacharyya distance. First, the first term of the Bhattacharyya distance for the Gaussian—which corresponds to the Mahalanobis distance—is used to find close candidates. Given the current hyperscan origin i, the square Mahalanobis distance is calculated for all hyperscans k

$$d_M^2 = (\mu_i - \mu_k)^T \left(\frac{\Sigma_k + \Sigma_i}{2}\right)^{-1} (\mu_i - \mu_k), \tag{25}$$

where μ_j and Σ_j are, respectively, the mean and the covariance matrix of hyperscan j origin provided by the Back-End. Second, for all hyperscans whose Mahalanobis distance is below a threshold d_{Mlim}^2, the second term of the Bhattacharyya distance for the Gaussian is calculated

$$d_B = \frac{1}{2} \ln \frac{\left|\frac{\Sigma_k + \Sigma_i}{2}\right|}{\sqrt{|\Sigma_k||\Sigma_i|}}. \tag{26}$$

This term gives the nodes separability due to covariance (see Fukunaga, 2006), which means that it distinguishes uncertain nodes from the more certain ones. Assuming that the current hyperscan is one of the most uncertain due to drift accumulation, maximizing d_B for the selected hyperscans means finding the most certain one. Typically, this hyperscan will be far from the current node and it will set a strength constraint to the problem in terms of information. Therefore, d_M^2 filters nearby nodes in terms of pose and d_B selects the most certain one which normally means the farthest one in terms of time, all based on uncertainty criteria. If this search returns a candidate, then, a registration between the candidate hyperscan k and the current hyperscan i is attempted. To prevent the scan matching problem from local minima, a seed is found using the Pose SLAM solution to evaluate the pose increment between those nodes

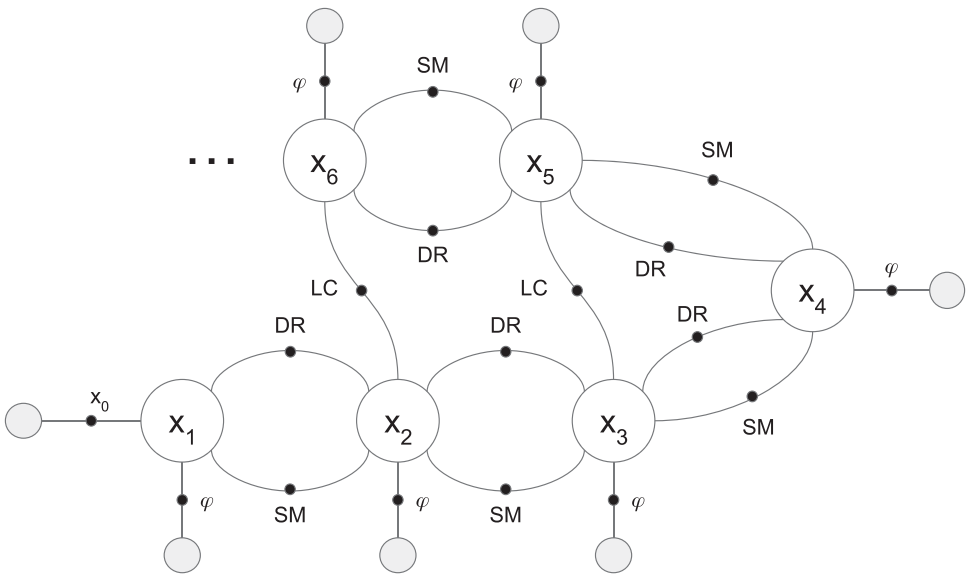

FIGURE 5 Factor graph structure for the defined Pose SLAM problem. SE(2) nodes $\vec{x}$ describing robot trajectory are estimated. A prior pose $\vec{x_0}$ is set for the first node and an SO(2) prior from magnetometer measures φ is defined for each node. All nodes are chained by two factors, one coming from dead reckoning and the other from scan matching. Finally, scan matching loops are closed for revisited places. LC, loop closure; SE, Special Euclidean; SLAM, simultaneous localization and mapping; SM, scan matching; SO, Special Orthogonal. DR, dead reckoning; SM, scan matching.

$$h(\vec{x}_k, \vec{x}_i) = \vec{x}_i \ominus \vec{x}_k \triangleq T_k^{-1} T_i, \quad (27)$$

where T_k and T_i are, respectively, the homogeneous transformation matrices related to nodes $\vec{x}_k$ and $\vec{x}_i$. Finally, if the registration problem converges, an SE(2) factor between both nodes can be set into the factor graph. However, the graph is not optimized until a new scan is received as loop closure is implemented in an asynchronous CPU thread to the main loop.

5 | EXPERIMENTAL DATA

The proposed Pose SLAM framework was tested on several underwater data sets, for which a set of experiments was carried on January 2022 by the technical staff of VICOROB laboratory using Universitat de Girona boat, Sextant. Two different experimental areas were considered. On the one hand, a couple of data sets were gathered on a harbor where straight vertical walls were scanned, referred as structured data sets. On the other hand, another data set was gathered in a natural environment where irregular rocks were perceived, referred to as an unstructured data set. All data sets were obtained by the Sparus II AUV (Carreras et al., 2018), presented in Section 3. The robot was teleoperated through a surface buoy and equipped with a Super SeaKing Profiler from Tritech International Ltd. (Tritech International Ltd., Super SeaKing Profiler, n.d.) for range measurements, shown in Figure 6a. This sensor was mounted at the payload space of the AUV and provided range measures in the front of the vehicle at 1.8° angular increments. Each sonar beam provided ranges from 0 to 15 m at 0.037 m resolution with their corresponding intensity values. This information is represented in Figure 6b, where beams are lines rotating around the robot in motion. Red intensities represent no detection and green to blue intensities show a harbor corridor. The Sparus II AUV also provided depth information from a pressure sensor; angular velocities, linear accelerations, and yaw rotation from an IMU; and linear velocities and altitude from a DVL. No GPS or USBL was available to measure robot absolute position.

The structured data sets were gathered at a man-made breakwater structure outside of the Sant Feliu de Guixols harbor. The breakwater is formed by a line of 20 squared reinforced concrete blocks of 14 m side with a spacing of 5 m between blocks. See Figure 7 for major clarity. The verticality of the walls allows one to consider the 3D component negligible in this environment. The profiling sonar was tuned with an FoV of 270° pointing to the front of the vehicle as the AUV could perceive walls from both sides. As it is shown in Figure 8 top, in this kind of environment the sonar scans are formed by parallel or perpendicular straight lines. Due to the steel of the blocks structure, the magnetometer measurements are corrupted, especially when the AUV passes through a corridor between two blocks. In these data sets the AUV followed different loop trajectories between the blocks favouring different loop closure events for the Pose SLAM problem. These data sets are formed by 63, 93, and 137 scans over 10.9, 16.5, and 23.9 min missions at 2.5 m constant depth.

The unstructured data set was gathered making a turn and a half around a natural rock called La Galera located at Cap de Creus Natural Park (El Port de la Selva). La Galera is approximately a 120-m length and 50-m width rock with mostly vertical walls, as shown in Figure 9. As it is shown in Figure 8 down, the rock morphology causes irregular scans with more features than in the structured data set, improving the registration if the vehicle motion is stable and the same layer is scanned. In this data set the AUV did a turn and a half around the rock in the counterclockwise direction. During the last half turn, loops in the Pose SLAM problem can be closed with the first half turn. Following this trajectory, the AUV could only perceive the rock on its left side. For this reason, a nonsymmetric FoV for the profiling sonar was

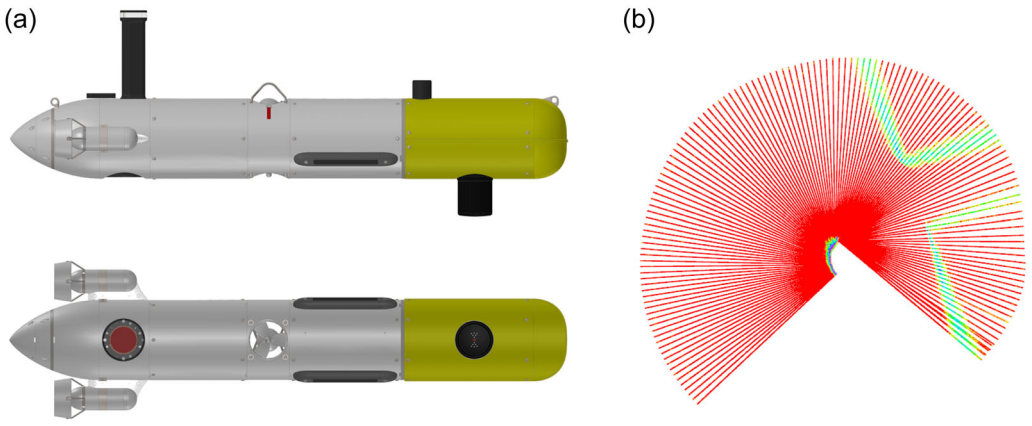

FIGURE 6 (a) Sparus II AUV equipped with a Super SeaKing Profiler from Tritech International Ltd. (b) Raw data from an acoustic scan while pointing at a harbor corridor. Red intensities mean no detection, while green to blue intensities show the perceived corridor. [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 7 Experimental area for the structured data sets. (Left) Satellite view of the experimental location with the working area marked with a red rectangle (*source*: Google Maps). (Right) Isometric view of a block. [Color figure can be viewed at wileyonlinelibrary.com]

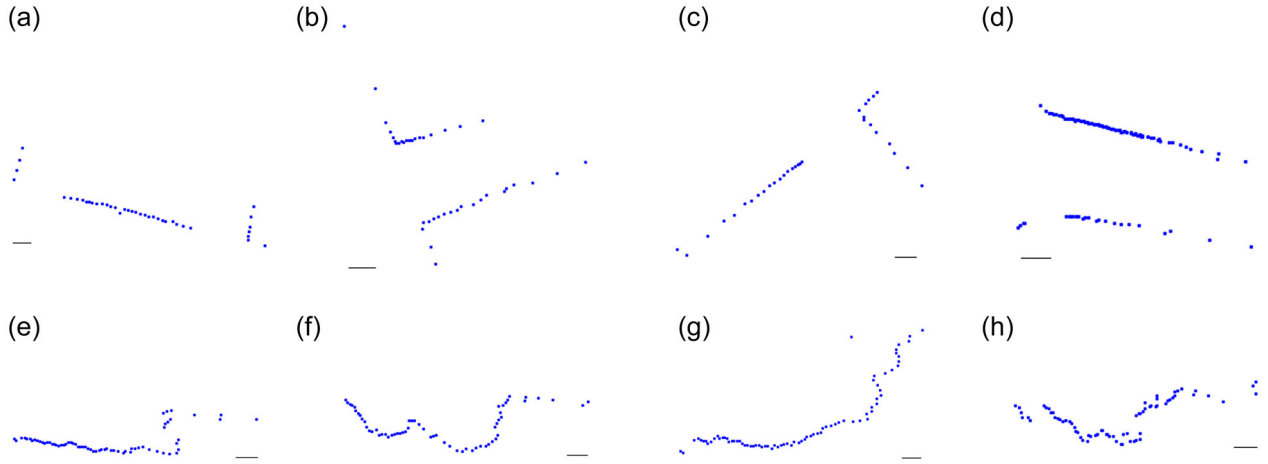

FIGURE 8 Sonar scans. (a–d) Scans from the structured data sets. (e–h) Scans from the unstructured data set. Black line is a 2-m scale for all scans. [Color figure can be viewed at wileyonlinelibrary.com]

set to favor the information content of the scans. Therefore, an FoV from 135° to −20° with respect to the forward direction of the AUV was set, reaching an FoV of 155°. The data set is formed by 260 scans over 23.1 min mission at 2.5 m constant depth.

An important requirement for all these experimental zones was to ensure that only vertical straight walls were scanned. The depth of the Sparus II AUV is easily controlled using the pressure sensor in combination with the vertical thruster. However, perturbations on roll and pitch motions are not controllable for the Sparus II AUV, which means that there is not any possibility for the vehicle to minimize these motions in an active way. Although the depth is maintained constant, these uncontrolled rotations change the orientation of the scanning plane of the profiling sonar projecting to different level curves on the walls are scanned. Nevertheless, as the dead reckoning system is built in the SE(3) group, we are able to measure these perturbations although we cannot control them. Moreover, the scan builder uses the 3D estimation from the dead reckoning navigation to build the scan. It is not until the scan is completely built that it is projected into 2D dimensions. Therefore, if scanned walls are completely vertical, these uncontrolled perturbations have no effect on the scans, as walls appear in the same expected position even if the AUV is pitching or rolling. For this reason we decided to work at the harbor blocks, as walls can be considered perfectly vertical. Furthermore, La Galera rock was carefully selected to guarantee as much as possible the vertically of the walls, taking into account the real possibilities in a natural environment. The protected side of the rock from the open sea is mostly vertical. However, on the other side of the rock, verticality is less guaranteed.

6 | RESULTS AND DISCUSSION

In this section tests on the proposed Pose SLAM framework using the aforementioned data sets are presented. The whole SLAM pipeline is executed in real time in all data sets and applies similar parameters to ease the comparison of the results. Gyroscopes run at 20 Hz, DVL runs at 6. 7 Hz and the SLAM system sets a key frame every time a

FIGURE 9 Experimental area for the unstructured data set. (Left) Satellite view of the location with the working area marked with a red rectangle (*source*: Google Maps). (Right) Isometric view of La Galera rock. [Color figure can be viewed at wileyonlinelibrary.com]

sonar scan is received. For the structured data sets configuration, a scan is built every 10.40 s and for the unstructured data set, as the FoV is reduced, every 6.10 s.

The parametrization used by the dead reckoning and SLAM system is shown in Table 1. Gyroscopes, DVL, and magnetometer variance values are taken from the sensors data sheet. However, the variance value of the DVL is not directly used by our dead reckoning system as a filter from `COLA2` is in between. Moreover, for the structured data set we had to increment the magnetometer uncertainty due to the perturbations caused by the steel inside the reinforced concrete of the harbor blocks. Otherwise, corrupted yaw measurements generated huge inconsistencies on the optimized factor graph. Covariance proportionality constants for scan matching, hyperscans length, and $dm^2_{\max}$ were learned through experimentation. In the unstructured data sets, as loops are favored, $dm^2_{\max}$ was more restrictive to avoid confusing loops.

Table 2 shows the parametrization used by the scan matching system. Beam intensities and solver algorithms parameters were tuned through experimentation. Maximum Bayesian GMM components were fixed to a value never exceeded by the Front-End in this experimentation.

TABLE 1 Pose SLAM system parameters.

	Structured	Unstructured
Angular velocity variance P_{imu}(rad/s)	0.0001	0.0001
Linear velocity variance P_{dvl} (m/s)	0.008	0.008
Magnetometer variance R_{mag}(rad)	0.5	0.01
Scan matching covariance proportionality constant r_{sm}	100.0	10.0
Loop closure covariance proportionality constant r_{lc}	10.0	10.0
Hyperscans length	6	5
Squared Mahalanobis distance threshold $dm^2_{\max}$	150.0	400.0

Abbreviations: IMU, Inertial Measurement Unit; LC, loop closure; SLAM, simultaneous localization and mapping; SM, scan matching.

TABLE 2 Scan matching system parameters.

	Distribution to Distribution	Point to Distribution
Minimum bin intensity	60.0	60.0
Maximum scan Bayesian GMM components K	10	10
Maximum hyperscan Bayesian GMM components K	30	30
Newton method maximum iterations	20	15
First Wolfe condition β_1	1e−4	1e−4
Second Wolfe condition β_2	0.8	0.9
Line Search maximum iterations	20	25
Gill, Murray, and Wright ρ	1e−6	1e−6

Abbreviation: GMM, Gaussian mixtures model.

6.1 | Structured data set

In the structured data sets the AUV was teleoperated making loops around the harbor blocks, retracing many sections of the previous trajectory. Therefore, these data sets are used to show the loop closure capabilities of the Pose SLAM system. Moreover, as it is shown in Figure 8 top, the scans in these data sets are formed by parallel or perpendicular straight lines. Therefore, scanning noise is easy to identify and scan matching is not ambiguous.

6.1.1 | Scan matching system

Figure 10 shows the performance of the scan matching system in two particular examples. However, it has to be noted that all registrations apply a sequential match of a D2D method followed by a P2D method. In the left column of the figure the match of two overlapping scans is provided. Figure 10a shows the scan association given by the dead reckoning system, where the fixed scan is plotted in blue and the moving scan in red. As it is seen, this association drifts and does

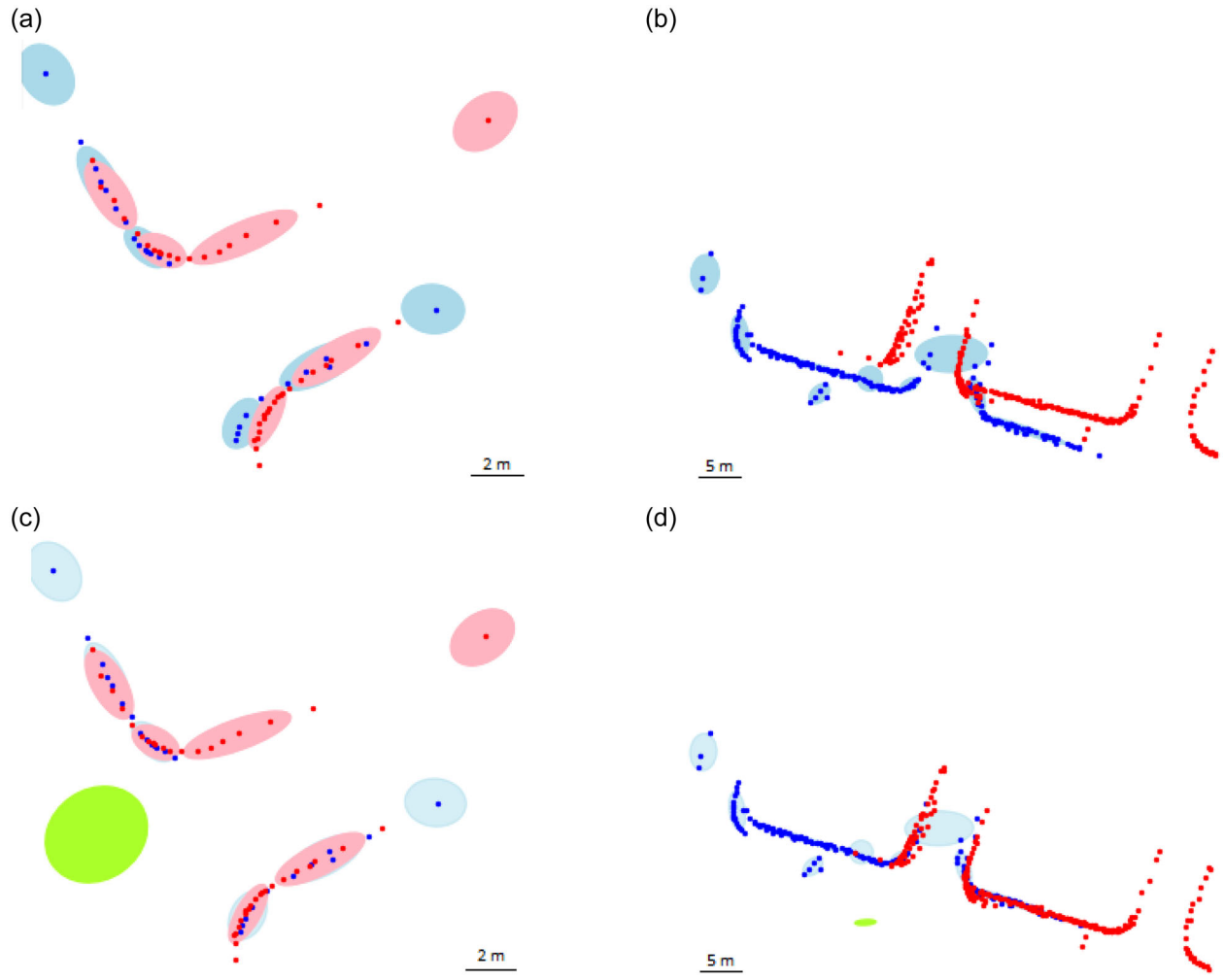

FIGURE 10 Scan matching results from the structured data set. (Blue) reference scan, (red) matching scan, and (green ellipsoid) positional uncertainty of the match. (a, c) D2D match on overlapping scans and (b, d) P2D match on hyperscans used to close loops in the Pose Graph. D2D, Distribution to Distribution; P2D, Point to Distribution. [Color figure can be viewed at wileyonlinelibrary.com]

not match. However, as it is close to the solution, it is used as a seed for the scan matching system. In this case, a D2D match is solved. Therefore, both scans have to be modeled by a GMM. As it can be seen in the figure, the Bayesian-GMM Front-End needs six components out of 10 to model both scans. Moreover, it is shown how the Bayesian-GMM uses three components to distinguish far detections, setting a component to model a single data observation. These components have an insignificant weight and, consequently, are ignored by the scan matching optimization. In Figure 10c the scan matching solution is given. Apart from showing that the two scans match, the green ellipsoid represents the shape of the positional uncertainty of the match. As both scans perceive a block corner that provides information in the two directions of the plain, the scan matching process is able to return information in both directions and, consequently, the uncertainty has a round shape.

Furthermore, the right column of Figure 10 shows the match of two hyperscans—formed by six scans—used for the SLAM system to close a loop in the Pose Graph. First of all, it can be seen how hyperscans provide a farther view of the environment with more structural features than in a single scan. In this case, three blocks and two corridors can be identified. Figure 10b shows the Pose SLAM solution before closing a loop. This scan association is drifted generating inconsistencies in the global map. However, it is used as the seed for the scan matching process. In this case, a P2D match is solved, so the fixed scan must be modeled by a GMM. As it is seen, the Bayesian-GMM needs only 9 components out of 30 to model the hyperscan, where two components in the left part of the hyperscan are used to isolate noisy points. Figure 10d shows the scan matching solution and its uncertainty. Now, as the hyperscan has more information in the horizontal direction than in the vertical direction, the positional uncertainty plotted as a green ellipsoid is degenerated. As it is expected, the narrow direction of the green ellipsoid, which corresponds to the most certain direction of the registration, matches with the most informative direction of the hyperscan.

6.1.2 | Pose SLAM system

Figure 11 shows the Pose SLAM results for different experiments. In the first experiment (first row of Figure 11) the AUV followed an eight-shape loop trajectory between the blocks. Through this trajectory, 3 loop closure events were created: one in the central corridor and two in the left side of the scanned blocks. As it is shown in Figure 11a the drift of the dead reckoning system is not huge and the structure of the scanned area is maintained. However, inconsistencies appear in the parallelism of the central corridor and between the two passes through the left side of the blocks. Figure 11b shows that thanks to the SLAM system, the trajectory is well reconstructed. Matching successive scans, a thin and coherent profile of the blocks is reached; while matching hyperscans, loops in the Pose Graph are closed removing the inconsistencies of the dead reckoning solution. Finally, any error with the estimated trajectory can be computed to quantitatively validate the estimation, as no ground truth is available in our experiments. In sea tests, in comparison to a laboratory water tank, it is very difficult to generate a structure to acquire AUV ground truth. Taking into account our experimental resources, mounting a set of optical cameras or artificial markers to constantly track robot position, was an impossible task regarding the size of the experimental area. Moreover, range measurements from acoustic modems are not accurate enough underwater. Therefore, the only way left to evaluate the performance of the proposed SLAM system is qualitatively, checking if the reconstructed map fits satellite views. Figure 11c shows the superposition of the SLAM solution with a satellite view of the area, showing that the SLAM solution shape and scale match reality.

Figure 12 shows the evolution of the AUV position uncertainty during the estimation of the same experiment. In these plots we show the uncertainty of each key frame the first time that it is solved, which means just after setting it. We do not plot the uncertainty of each key frame obtained from the last solver call. This way it can be seen how the SLAM system bounds uncertainty growth, in comparison to the dead reckoning system where no updates to a filter are performed. Figure 12c,d shows the uncertainty evolution during estimation, where loop closure events are marked by green dots. As it can be seen, the most important uncertainty reductions happen when a loop is closed, especially if the time elapsed with the last loop closure event is big. Small uncertainty reductions out of loop closure events are caused by successive scan registrations and orientation priors set using the magnetometer, showing how these systems also help bounding uncertainty grow. Finally, Figure 12a,b allows us to analyze the shape of the uncertainty matrix. During the first vertical transect of the AUV, the uncertainty ellipsoid is degenerated through the vertical direction, which is consistent as the AUV moves forward. This deformation in the forward direction is maintained when the AUV starts to rotate around the blocks. However, when the AUV completes a full turn around itself (this is when the AUV goes out of the top corridor and starts descending) the uncertainty shape becomes circular. This means that the robot has accumulated error in all possible directions. This behavior is maintained during the turn around the bottom block. Finally, during the last vertical transect, the uncertainty is again deformed in the forward direction as turning directions become decompensated. The same behavior is shared with the scan matching system, as registrations can only provide information on the normal direction to the walls, which agrees with being uncertain in the forward direction of the AUV in this particular trajectory.

In the second experiment (central row of Figure 11) the AUV followed a longer path. The robot, first, navigated by the side of three blocks. Then, it crossed to the other side of the blocks by a corridor between the blocks and it navigated by the other side of the last two blocks. After, it navigated through a corridor and it closed a loop navigating by the side of the first block. Then, it crossed again to the other side closing another loop. Finally, it navigated through a corridor to close the last loop. As it is shown in Figure 11d, in this experiment the drift is important and lots of inconsistencies appear in the dead reckoning solution. Figure 11e shows how the scan matching system allows one to close loops in the Pose Graph and correct all inconsistencies. Only an imprecision is maintained in the bottom corridor. However, it is insignificant compared with what happens at the same point in the dead reckoning solution. Finally, Figure 11f shows how the SLAM system maintains the shape and the scale, matching the SLAM solution with reality.

In the third experiment (last row of Figure 11) the AUV followed an even longer path. Through this trajectory two types of loops were established. In the beginning, the AUV navigated up and down by the side of 4 blocks. Then, it crossed to the other side of the blocks through a corridor and again it navigated up and down by the side of the same blocks. Therefore, loops can be closed between the up and down passes, but both sides could not still be related. After, when the robot returned to the starting point and started a zig-zag trajectory crossing corridors from one side to the other, both sides could be related. As it is shown in Figure 11g, again the drift of the dead reckoning system is considerable forming a very inconsistent map. However, Figure 11h shows how the SLAM system can build a more coherent map. As it can be seen in the figure, the side loops are easy to close, however, the loops in the corridors that relate both sides are more difficult to close and some inconsistencies appear at the end of the AUV trajectory. Finally, Figure 11i shows how the SLAM map is close to the reality in the bottom part. However, as loops between both sides did not succeed at all, some inconsistencies appear at the upper part.

To sum up, these experiments show how equipping an AUV with a range sensor and matching the resulting scans allows one to build a navigation system that maintains the structure of the environment. When the AUV trajectory is not so long, the dead reckoning system is able to maintain the consistency of the map. However, as the AUV trajectory gets longer, more and more inconsistencies appear in dead reckoning maps evidencing the need of the SLAM system. The most essential capability of this system is the ability of closing loops in the Pose Graph, which removes the major inconsistencies of the dead reckoning solution. Furthermore, matching overlapping scans allows one to maintain a thin profile of the scanned surfaces and IMU's magnetometer priors help maintain robot orientation, even the distortion provided by the reinforced concrete of the surroundings. Moreover, the automatic method to recover uncertainty from the

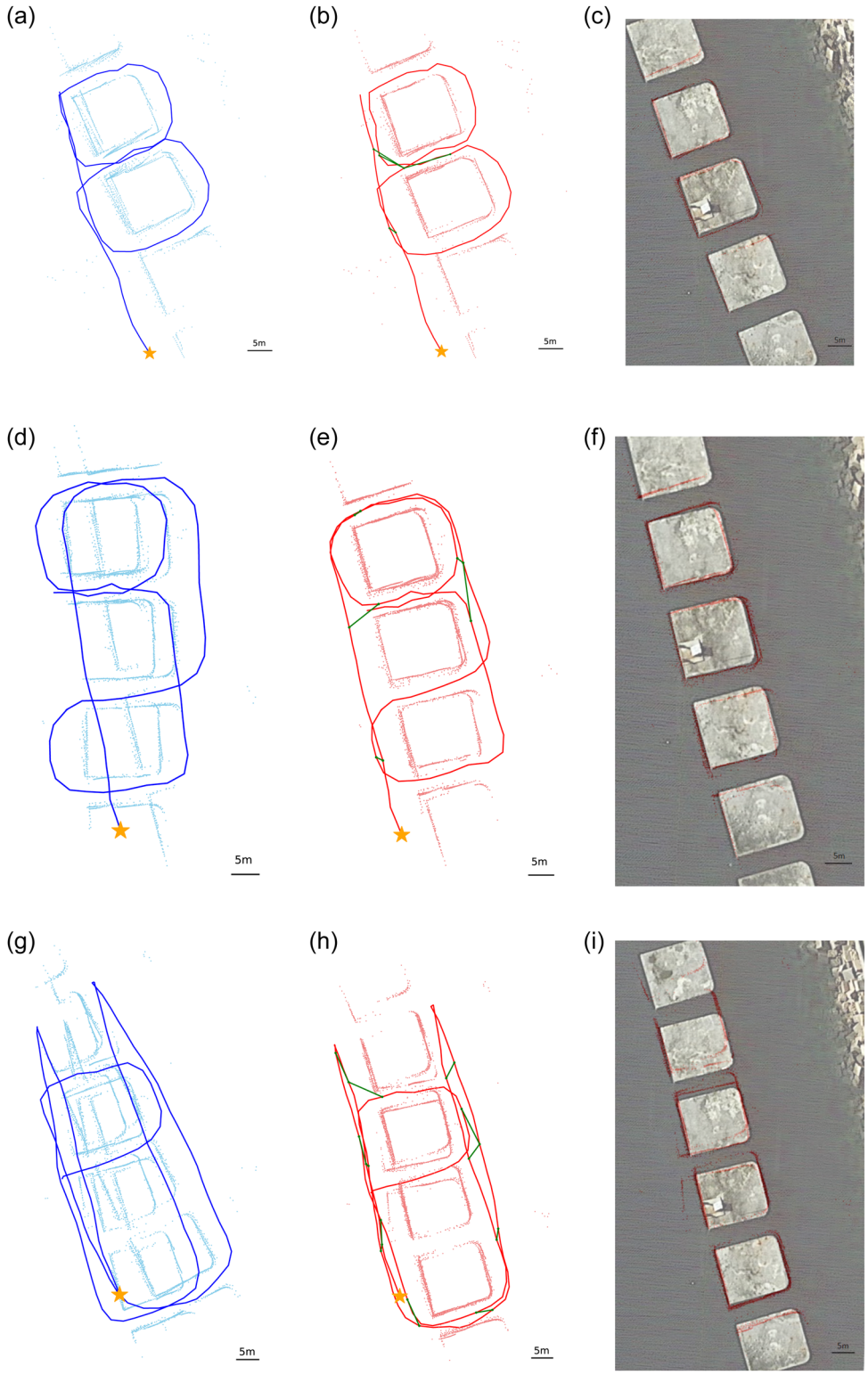

FIGURE 11 Structured data sets experimentation. (a, d, g) dead reckoning trajectory where the AUV starting position is marked by a star. (b, e, h) SLAM trajectory where closed loops are represented by green lines. (c, f, i) Superposition of the SLAM blocks profile at 2.5 m depth to the satellite view (*source*: Google Maps). AUV, Autonomous Underwater Vehicle; SLAM, simultaneous localization and mapping. [Color figure can be viewed at wileyonlinelibrary.com]

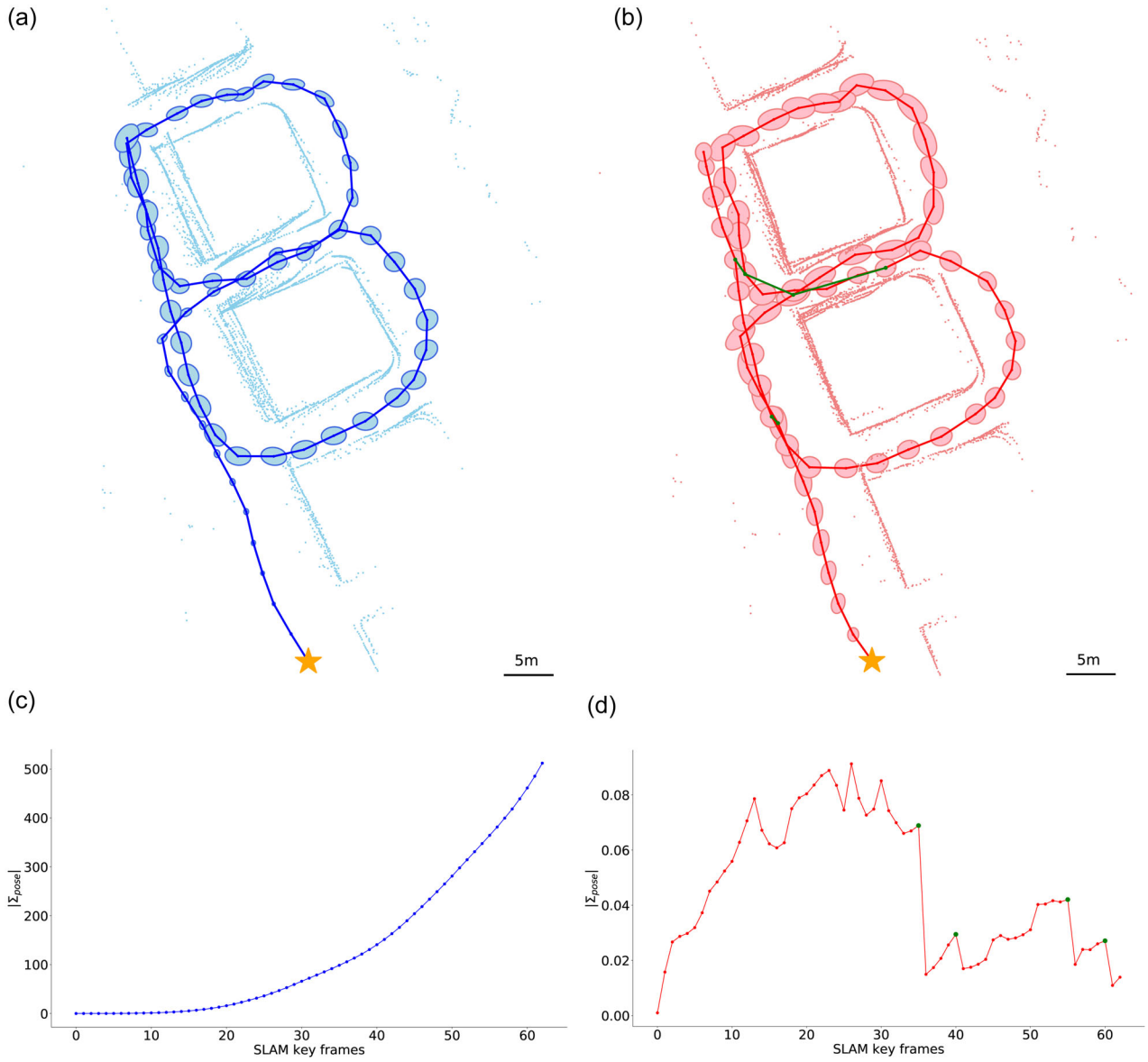

FIGURE 12 Estimated uncertainty on Experiment 1 from the structured data sets. (a, b) AUV position uncertainty at the current key frame during optimization. The uncertainty is scaled at each plot for visualization reasons. (c, d) AUV pose uncertainty determinant evolution of the current key frame during optimization. Green dots indicate loop closure events. AUV, Autonomous Underwater Vehicle. [Color figure can be viewed at wileyonlinelibrary.com]

scan matching problem provides a very promising performance. As scans are composed of straight lines, in many cases only information in one direction can be extracted from the scan matching process. The scan matching uncertainty transmits this feature to the SLAM Back-End and, as it can be seen in the results, allows the algorithm to find good solutions.

6.2 | Unstructured data set

The unstructured data set is used to test the proposed Pose SLAM system in a real case application where loops are present but not favored. In a natural environment, there are no perturbations in the magnetometer measurements caused by reinforced concrete. Therefore, a less noisy problem is solved. However, as it is shown in Figure 8 down, scans have irregular shapes not obvious to register, shadows, and lots of imprecision.

In Figure 13 the experimental results are provided. Figure 13a shows the dead reckoning trajectory, whereas in Figure 13b the Pose SLAM trajectory is given. As it can be seen, the dead reckoning system drifts over time and the half overlapping turn does not match the first turn. Moreover, inconsistencies at the tip of the rock appear in both turns, detailed in Figure 14a. In this location the rock has a very steep wall entering to the sea. When the robot passed over it, the DVL sensor loosed the seafloor reference and its linear velocity

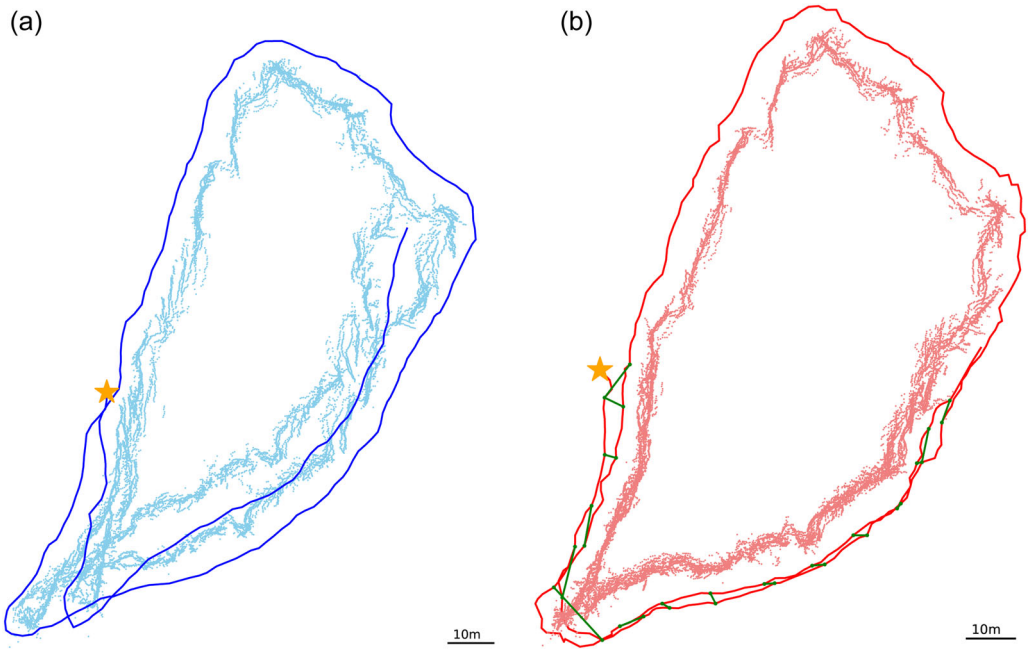

FIGURE 13 Unstructured data set experimentation. (a) Dead reckoning trajectory where the AUV starting position is marked by a star. (b) SLAM trajectory where closed loops between the first and second turns are represented by green lines. An animation on these results can be found at https://www.youtube.com/watch?v=4nYtELVcGMA. AUV, Autonomous Underwater Vehicle; SLAM, simultaneous localization and mapping. [Color figure can be viewed at wileyonlinelibrary.com]

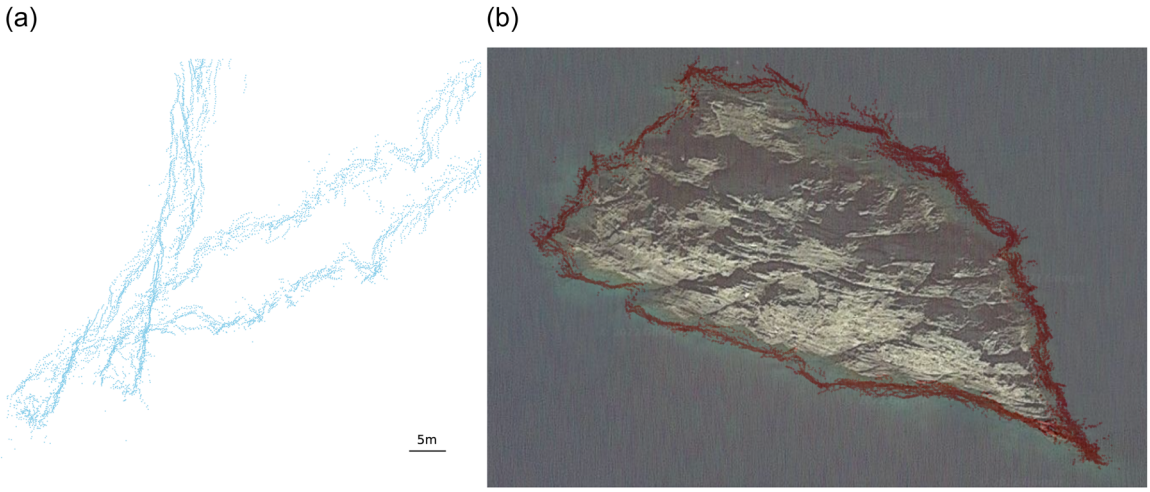

FIGURE 14 (a) Detail of the dead reckoning map at the tip of the rock where lots of inconsistencies appear. (b) Superposition of the SLAM rock profile at 2.5 m depth to the satellite view (*source*: Google Maps). SLAM, simultaneous localization and mapping. [Color figure can be viewed at wileyonlinelibrary.com]

measurements relative to the bottom were corrupted. As the dead reckoning system is based on these measurements, the large quantity of outliers generates a navigation drift and, consequently, a map inconsistency at this location. In addition to this and considering the rock location at the map shown in Figure 9 left, the AUV started its trajectory at a location protected from water currents by the rock. When the AUV reached the tip of the rock, it went out to the open sea and it received the sea current acting on it for the first time. In a normal situation, perturbations on the AUV motion caused by sea currents are measured by the inertial sensors of the vehicle, being observable perturbations. However, in the peculiar situation of the experiment, linear velocity measurements were corrupted at this particular location. Thus, the perturbation caused by the sea current could not be perceived and the dead reckoning suddenly jumped, mapping the rock wall in a place previously mapped as free space.

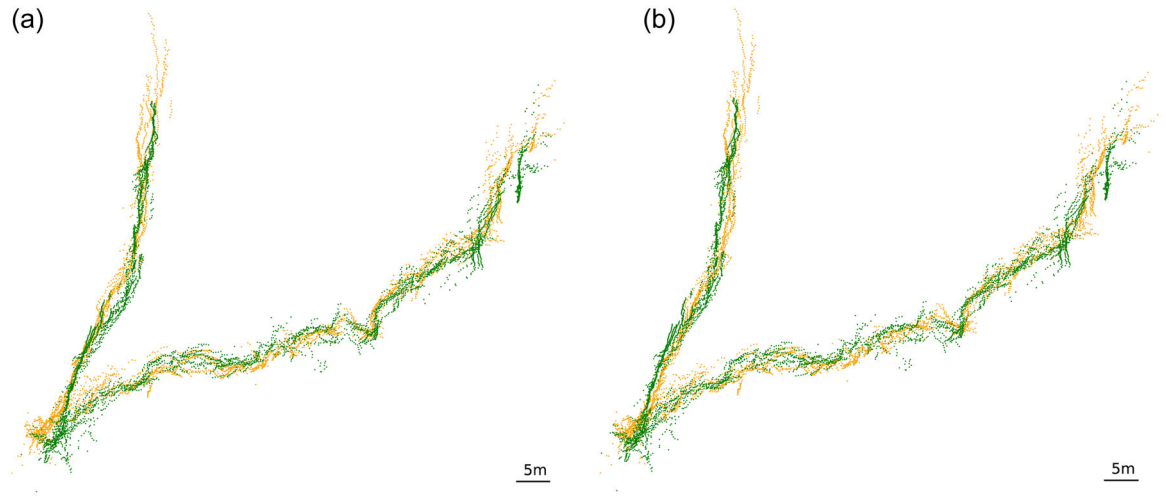

FIGURE 15 Map of the overlapping AUV paths for the unstructured data set. The first turn is plotted in orange and the second turn in green. (a) SLAM solution and (b) SLAM solution registered using the Bayesian-GMM + P2D algorithm. AUV, Autonomous Underwater Vehicle; GMM, Gaussian mixtures model; P2D, Point to Distribution; SLAM, simultaneous localization and mapping. [Color figure can be viewed at wileyonlinelibrary.com]

Figure 13b shows how all this casuistry can be corrected by the proposed Pose SLAM system and a coherent map of the rock is reached. As shown in green, thanks to the scan matching system, many loops in the Pose Graph are closed between the first and the second turn, removing inconsistencies from the dead reckoning map. Moreover, the scan matching system fused with magnetometer measurements can fix all the reported problems that appear when the AUV surrounds the tip of the rock. Nevertheless, as shown in the figure, the obtained map has a thick countour as the SLAM system is not able to remove the scan noise. Although the scan matching algorithm characterizes the noise and manages it, the algorithm is unable to remove it from the point cloud. Thus, as the SLAM system only concatenates raw scans to build the map, this wide dispersion appears in the contour. This scan noise is mainly caused by the uncontroled disturbances on the roll and pitch orientations of the Sparus II AUV that point the profiling sonar to different rock heights. In comparison to the structured data sets, as the robot is scanning an irregular surface, the amount of noise is higher and the contour is wider.

Figure 14b shows that the map obtained by the SLAM system matches the satellite view of the rock. This view qualitatively proves the capabilities of the proposed SLAM system as the shape and the scale of the rock are correctly mapped. Looking in detail at the superposition, the obtained map is slightly bigger than the contour at the surface as the robot is scanning a layer of the rock at 2.5 m depth. To somehow quantify the quality of the obtained map, we analyze the two overlapping trajectories when scanning the lower part of the rock. Figure 15a shows the map obtained by the SLAM system, where the first turn is plotted in orange and the second in green. To quantify the alignment we took the map of each turn as a single scan and we registered them using a P2D policy applying the Bayesian-GMM Front-End. Given an upper bound of K_0 = 20 components, the Front-End fitted a GMM of eight components and, given a zero transform as seed, the scan matching algorithm returned a translation of $(0.196, 0.278)$ meters and a yaw rotation of $-0.060°$. The registered views are shown in Figure 15b. Taking into account the dimensions of the scanned area, the misalignment provided by the scan matching algorithm is insignificant, as shown in the figure, proving the capabilities of the SLAM system in closing loops and reconstructing coherent views.

7 | CONCLUSIONS AND FUTURE WORK

In this paper we present a novel underwater Pose SLAM framework based on inertial sensors and acoustic point cloud registration solved by applying smoothing techniques. Special attention is put on modeling factor uncertainties when building the factor graph. First of all, thanks to the computation of second derivatives on the policy of the scan matching technique, we are able to recover the uncertainty of the scan matching result based on the spatial information implicit in the matched scans. Moreover, a result is provided in Proposition 1 to map uncertainties from the composite manifold where the scan matching solver works to the SE(n) group where the SLAM Back-End works. By this way, the uncertainty pipeline is fully coherent. Second, a dead reckoning system specialized for the underwater domain, based on gyroscopes and DVL, was designed following Lie Theory. Using this system the dead reckoning uncertainty can also be tracked in the SE(3) group. As it can be seen, attention is focused on the mathematical tidiness and on the particular characteristics of the underwater domain.

In addition to the SLAM system, a rigid scan matching technique is also specially designed for the underwater domain. This technique is based on GMMs to model sparse and noisy scans, the key

characteristics of sonar scans. Fit a GMM to an acoustic scan lets automatically model sensor noise in the covariance matrices of the model. Moreover, the introduction of the Bayesian-GMM algorithm to fit a GMM lets set unweighted components to single outlier detections without generating convergence problems to the fitting algorithm, contrary to the performance of the K-means algorithm or the EM algorithm for GMM which suffers convergence problems when only one point is assigned to a component. This way, outliers are automatically removed from the scan and do not participate in the scan matching process. Finally, the application of the Bayesian-GMM algorithm also allows one to learn the optimal number of components needed to model a scan, reaching a flexible tool that automatically adapts to different environments with different structures.

Real experiments were carried out to test our SLAM system on real data. Two different environments have been considered. Experiments in harbor blocks allowed us to test the capabilities of our system in detecting loop closure events and closing loops in the pose graph, whereas experiments in a natural environment let us test our system in a more real situation closer to the application. These experiments show that our Pose SLAM system allows one to simultaneously recover the AUV trajectory and build a coherent layer map at constant depth in a 3D environment; performing both tasks online with the robotic application. Real time is a very interesting feature if this robotic application is extended with path planning capabilities to reach an Active SLAM system. The consistency of the proposed SLAM system has been proved, reaching coherent maps as presented.

Future work should be extending the current Pose SLAM problem to the 3D case, to build 3D maps without imposing movement constraints to the AUV. To do it, the dead reckoning system is already valid as it is integrating robot pose in the SE(3) group. However, the mechanical profiling sonar must be replaced by a Multibeam Sonar mounted on a Pan & Tilt platform to build 3D point clouds. Registering these point clouds following the proposed approach, factors in the SE(3) group can be set and 3D maps can be obtained.

Finally, nonrigid scan matching techniques can be explored to remove the distortion added to the scan by the dead reckoning system while building it. However, to perform this evolution the GMM essence of the technique should be maintained to be able to model scan noise and discard outliers. Using the P2D method this is possible as only the undistorted scan, or reference scan, is modeled by a GMM; whereas, in the P2D paradigm more effort is needed as both scans are embedded in a GMM. Nevertheless, nonrigid techniques are much more computing demanding than rigid techniques, as a coordinate frame must be learned for each data point. Therefore, reaching sufficient computing performance for an online application is difficult, even with the slow nature of sonar scanners.

ACKNOWLEDGMENTS

This study has been supported by the Spanish Government through the PhD Grant/Award Number FPU19/03638 to Pau Vial and by the Biter-AUV and PLOME projects, under the grant agreements PID2020-114732RB-C33 and PLEC2021-007525. Open Access funding was provided thanks to the CRUE-CSIC agreement with Wiley. Finally, the authors are also grateful to Lluís Magí for helping with the Sparus II AUV sea operations, Roger Pi for assisting with the software development and Esther Barrabés from the Applied Mathematics Department of Universitat de Girona for helping in the derivatives development of Annexes A and B.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Pau Vial http://orcid.org/0000-0003-4935-0899

REFERENCES

Agarwal, S. & Mierle, K. (n.d.) *Ceres solver*. Available from: http://ceres-solver.org

Attias, H. (2000) A variational Bayesian framework for graphical models. *Advances in Neural Information Processing Systems*, 12.

Bailey, T. & Durrant-Whyte, H. (2006) Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics & Automation Magazine*, 13, 108–117.

Bengtsson, O. & Baerveldt, A.J. (2003) Robot localization based on scan-matching—estimating the covariance matrix for the IDC algorithm. *Robotics and Autonomous Systems*, 44, 29–40.

Besl, P.J. & McKay, H.D. (1992) A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14, 239–256.

Biber, P. & Strasser, W. (2003) The normal distributions transform: a new approach to laser scan matching. In: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Vol. 3, pp. 2743–2748.

Bierlaire, M. (2015) *Optimization: principles and algorithms*. Lausanne Switzerland: EPFL Press.

Blanco, J. (2010) *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Univ. Málaga Tech. Rep., 012010.

Bouraine, S., Bougouffa, A. & Azouaoui, O. (2021) Particle swarm optimization for solving a scan matching problem based on the normal distributions transform. *Evolutionary Intelligence*, 15(8), 1–12.

Burguera, A., González, A. & Oliver, G. (2009) On the use of likelihood fields to perform sonar scan matching localization. *Autonomous Robots*, 26, 203–222.

Carreras, M., Hernández, J., Vidal, E., Palomeras, N., Ribas, D. & Ridao, P. (2018) Sparus II AUV—a hovering vehicle for seabed inspection. *IEEE Journal of Oceanic Engineering*, 43, 344–355.

Censi, A. (2007) An accurate closed-form estimate of ICP's covariance. In: IEEE International Conference on Robotics and Automation, pp. 3167–3172.

Dellaert, F. (n.d.) *Gtsam 4.0: Factor graphs for sensor fusion in robotics*. Available from: https://gtsam.org/

Dellaert, F. (2021) Factor graphs: exploiting structure in robotics. *Annual Review of Control, Robotics and Autonomous Systems*, 4, 141–166.

Dellaert, F. & Kaess, M. (2006) Square root sam: simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research (IJRR)*, 25, 1181–1203.

Dellaert, F. & Kaess, M. (2017) *Factor graphs for robot perception*. Atlanta: Now Publishers Inc.

Dempster, A.P., Laaird, N.M. & Rubin, D.B. (1977) Maximum likelihood for incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39, 1–38.

Deray, J. & Solà, J. (n.d.) *Manif: A small header-only library for lie theory*. Available from: https://artivis.github.io/manif/index.html

Durrant-Whyte, H. & Bailey, T. (2006) Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13, 99–110.

Eckart, B. (2017) *Compact generative models of point cloud data for 3D perception* (Doctoral Dissertation). Carnegie Mellon University.

Eckart, B., Kim, K., Troccoli, A., Kelly, A. & Kautz, J. (2016) Accelerated generative models for 3D point cloud data. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 5497–5505.

Einhorn, E. & Gross, H. (2013) Generic 2D/3D slam with NDT maps for lifelong application. In: European Conference on Mobile Robots, pp. 240–247.

Fukunaga, K. (2006) *Introduction to statistical pattern recognition*. San Diego: Academic Press.

Gill, P., Murray, W. & Wright, M. (1981) *Practical optimization*. London: Academic Press.

Grisetti, G., Guadagnino, T., Aloise, I., Colosi, M., Della Corte, B. & Schlegel, D. (2020) Least squares optimization: from theory to practice. *Robotics*, 9, 51.

Grisetti, G., Stachniss, C. & Burgard, W. (2007) Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23, 34–46.

Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C. & Burgard, W. (2013) OctoMap: an efficient probabilistic 3D mapping framework based on Octrees. *Autonomous Robots*, 34, 189–206.

Hover, F.S., Eustice, R.M., Kim, A., Englot, B., Johannsson, H., Kaess, M. et al. (2012) Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31, 1445–1464.

Ila, V., Andrade-Cetto, J., Valencia, R. & Sanfeliu, A. (2007) Vision-based loop closing for delayed state robot mapping. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3892–3897.

Ila, V., Polok, L., Solony, M. & Svobodo, P. (2017) SLAM^{++}: a highly efficient and temporally scalable incremental slam framework. *The International Journal of Robotics Research*, 36, 210–230.

Iqua Robotics SL. (n.d.). *COLA2 software architecture*. Available from: https://iquarobotics.com/cola2

Jiang, M., Song, S., Tang, F., Li, Y., Liu, J. & Feng, X. (2019) Scan registration for underwater mechanical scanning imaging sonar using symmetrical Kullback–Leibler divergence. *Journal of Electronic Imaging*, 28, 013026. https://doi.org/10.1117/1.JEI.28.1.013026

Jiang, B. & Vemuri, B.C. (2011) Robust point set registration using Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, 1633–1645.

Johannsson, H., Kaess, B., Englot, B., Hover, F. & Leonard, J. (2010) Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In: International Conference on Intelligent Robots and Systems, pp. 4396–4403.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. & Dellaert, F. (2012) iSAM2: incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31, 216–235.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K. & Burgard, W. (2011) G2o: a general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3607–3613.

Li, J., Bao, J. & Yu, Y. (2010) Study on localization for rescue robots based on NDT scan matching. In: The 2010 IEEE International Conference on Information and Automation, pp. 1908–1912.

Lloyd, S. (1982) Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28, 129–137.

Lu, F. & Milios, E. (1997) Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4, 333–349.

Magnusson, M. (2009) *The three-dimensional normal-distributions transform, an efficient representation for registration, surface analysis and loop detection* (Doctoral Dissertation). Örebro University.

Mallios, A., Ridao, P., Hernandez, E., Ribas, D., Maurelli, F. & Petillot, Y. (2009) Pose-based SLAM with probabilistic scan matching algorithm using a mechanical scanned imaging sonar. In: OCEANS 2009-EUROPE, pp. 1–6.

Mielle, M., Magnusson, M. & Lilienthal, A.J. (2019) A comparative analysis of radar and lidar sensing for localization and mapping. In: European Conference on Mobile Robots, pp. 1–6.

Palomeras, N., El-Fakdi, A., Carreras, M. & Ridao, P. (2012) COLA2: a control architecture for AUVs. *IEEE Journal of Oceanic Engineering*, 37, 695–716.

Saarinen, J., Andreasson, H. & Stoyanov, T. (2013) Normal distributions transform Monte-Carlo localization (NDT-MCL). In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 382–389.

Saarinen, J., Andreasson, H., Stoyanov, T., Ala-Luhtala, J. & Lilienthal, A.J. (2013) Normal distributions transform occupancy maps: application to large-scale online 3D mapping. In: IEEE International Conference on Robotics and Automation, pp. 2233–2238.

Segal, A., Haehnel, D. & Thrun, S. (2009) Generalized ICP. *Robotics: Science and Systems*, 2, 435.

Solà, J. (2014) *Simultaneous localization and mapping with the extended Kalman filter*. Barcelona: UPC.

Solà, J., Deray, J. & Atchuthan, D. (2020) A micro lie theory for state estimation in robotics. ArXiv, abs/1812.01537. https://doi.org/10.48550/arXiv.1812.01537

Stoyanov, T., Magnusson, M., Andreasson, H. & Lilienthal, A.J. (2010) Path planning in 3D environments using the normal distribution transform. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3263–3268.

Stoyanov, T., Magnusson, M., Andreasson, H. & Lilienthal, A.J. (2012) Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The International Journal of Robotics Research*, 31, 1377–1393.

Stoyanov, T., Saarinen, J., Andreasson, H. & Lilienthal, A.J. (2013) Normal distributions transform occupancy map fusion: simultaneous mapping and tracking in large scale dynamic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4702–4708.

Tabib, W., O'Meadhra, C. & Michael, N. (2018) On-manifold GMM registration. *IEEE Robotics and Automation Letters*, 3, 3805–3812.

Teixeira, P.V., Kaess, M., Hover, F.S. & Leonard, J.J. (2016) Underwater inspection using sonar-based volumetric submaps. In: International Conference on Intelligent Robots and Systems, pp. 4288–4295.

Tritech International Ltd., Super SeaKing Profiler. (n.d.). Available from: https://www.tritech.co.uk/media/products/dual-frequency-profiler-super-seaking-dfp.pdf?id=9e4bad8a

Vallicrosa, G. & Ridao, P. (2018) H-slam: Rao-Blackwellized particle filter SLAM using Hilbert maps. *Sensors*, 18, 1386.

VanMiddlesworth, M.A., Kaess, M., Hover, F. & Leonard, J.J. (2015) Mapping 3D underwater environments with smoothed submaps. In: *Field and service robotics. Springer tracts in advanced robotics*, Vol. 105.

Vial, P. (n.d.) *GMM registration repository*. Available from: https://bitbucket.org/gmmregistration/gmm_registration/

Vial, P., Malagón, M., Segura, R., Palomeras, N. & Carreras, M. (2023) GMM registration: a probabilistic scan matching approach for sonar-based AUV navigation. In: 2023 IEEE International Conference on Robotics and Automation.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Vial, P., Palomeras, N., Solà, J. & Carreras, M. (2023) Underwater Pose SLAM using GMM scan matching for a mechanical profiling sonar. *Journal of Field Robotics*, 1–28. https://doi.org/10.1002/rob.22272

APPENDIX A: POINT TO DISTRIBUTION

Given a GMM $p(x|\pi, \mu, \Sigma)$ modeling a 2D point cloud and an overlapping 2D point cloud $\mathcal{D} = \{q_1, ..., q_n\}$, the P2D problem is

$$\Omega^* = \arg\min_{\Omega} \ln p(\mathcal{D}|\pi, \mu, \Sigma, \Omega) = \arg\min_{\Omega} \sum^{\mathcal{D}} \ln p(x = Rq_d + t|\pi, \mu, \Sigma, \Omega) \sim \arg\min_{\Omega} - \sum^{\mathcal{D}} \sum^{K} \pi_k \mathcal{N}(x = Rq_d + t|\mu_k, \Sigma_k, \Omega),$$

where the optimization variable is the relative pose between both point clouds defined by $\Omega = (t_x, t_y, \theta)$. Therefore, the problem cost function is

$$F = -\sum_{i=1}^{N_{\mathcal{D}}} \sum_{k=1}^{K} f_{ik},$$

where

$$f_{ik} = \frac{\pi_k}{(2\pi)^{\frac{D}{2}}\sqrt{|\Sigma_k|}} \exp\left(-\frac{1}{2}(\mu_k - Rq_i - t)^T \Sigma_k^{-1} (\mu_k - Rq_i - t)\right)$$

and

$$t = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

define, respectively, a translation vector and a rotation matrix.

Taking the derivatives of the cost function in the composite manifold $<\mathbb{R}^2, SO(2)>$ the Jacobian vector is

$$J = \begin{bmatrix} \frac{\partial F}{\partial t_x} \\ \frac{\partial F}{\partial t_y} \\ \frac{\partial F}{\partial \theta} \end{bmatrix} = -\sum_{i=1}^{N_{\mathcal{D}}} \sum_{k=1}^{K} f_{ik} (\mu_k - Rq_i - t)^T \Sigma_k^{-1} \begin{bmatrix} \frac{\partial t}{\partial t_x} & \frac{\partial t}{\partial t_y} & \frac{\partial R}{\partial \theta} q_i \end{bmatrix},$$

where

$$\frac{\partial t}{\partial t_x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \frac{\partial t}{\partial t_y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \frac{\partial R}{\partial \theta} = \begin{bmatrix} -\sin(\theta) & -\cos(\theta) \\ \cos(\theta) & -\sin(\theta) \end{bmatrix}.$$

The Hessian matrix is

$$H = -\sum_{i=1}^{K_{\mathcal{F}}} \sum_{k=1}^{K_{\mathcal{M}}} \begin{bmatrix} \frac{\partial^2 f_{ik}}{\partial t_x^2} & \frac{\partial^2 f_{ik}}{\partial t_x \partial t_y} & \frac{\partial^2 f_{ik}}{\partial t_x \partial \theta} \\ \frac{\partial^2 f_{ik}}{\partial t_y \partial t_x} & \frac{\partial^2 f_{ik}}{\partial t_y^2} & \frac{\partial^2 f_{ik}}{\partial t_y \partial \theta} \\ \frac{\partial^2 f_{ik}}{\partial \theta \partial t_x} & \frac{\partial^2 f_{ik}}{\partial \theta \partial t_y} & \frac{\partial^2 f_{ik}}{\partial \theta^2} \end{bmatrix},$$

where

$$\frac{\partial^2 f_{ik}}{\partial t_a \partial t_b} = \frac{\partial^2 f_{ik}}{\partial t_b \partial t_a} = f_{ik} \left\{ (\mu_k - Rq_i - t)^T \Sigma_k^{-1} \frac{\partial t}{\partial t_b} (\mu_k - Rq_i - t)^T \Sigma_k^{-1} \frac{\partial t}{\partial t_a} - \frac{\partial t}{\partial t_a}^T \Sigma_k \frac{\partial t}{\partial t_b} \right\},$$

$$\frac{\partial^2 f_{ik}}{\partial t_a \partial \theta} = \frac{\partial^2 f_{ik}}{\partial \theta \partial t_a} = f_{ik} \left\{ (\mu_k - Rq_i - t)^T \Sigma_k^{-1} \frac{\partial R}{\partial \theta} q_i (\mu_k - Rq_i - t)^T \Sigma_k^{-1} \frac{\partial t}{\partial t_a} - \frac{\partial t}{\partial t_a}^T \Sigma_k^{-1} \frac{\partial R}{\partial \theta} q_i \right\},$$

$$\frac{\partial^2 g_{ik}}{\partial \theta^2} = f_{ik} \left\{ \left((\mu_k - Rq_i - t)^T \Sigma_k^{-1} \frac{\partial R}{\partial \theta} q_i \right)^2 - q_i^T \frac{\partial R}{\partial \theta}^T \Sigma_k^{-1} \frac{\partial R}{\partial \theta} q_i + (\mu_k - Rq_i - t)^T \Sigma_k^T \frac{\partial^2 R}{\partial \theta^2} q_i \right\},$$

and

$$\frac{\partial^2 R}{\partial \theta^2} = \begin{bmatrix} -\cos(\theta) & \sin(\theta) \\ -\sin(\theta) & -\cos(\theta) \end{bmatrix}.$$

APPENDIX B: DISTRIBUTION TO DISTRIBUTION

Given two GMM $p_{\mathcal{F}}(x|\pi, \mu, \Sigma)$ and $p_{\mathcal{M}}(x|\tau, \nu, \Gamma)$ modeling two overlapping 2D point clouds, the D2D problem is

$$\Omega^* = \arg\max_{\Omega} \mathcal{L}^2(\Omega) = \arg\min_{\Omega} - \sum_{i=1}^{K_{\mathcal{F}}} \sum_{k=1}^{K_{\mathcal{M}}} \pi_i \tau_k \mathcal{N}(0|\mu_i - R\nu_k - t, \Sigma_i + R\Gamma_k R^T, \Omega),$$

where the optimization variable is the relative pose between both point clouds defined by $\Omega = (t_x, t_y, \theta)$. Therefore, the problem cost function is

$$G = -\sum_{i=1}^{K_{\mathcal{F}}} \sum_{k=1}^{K_{\mathcal{M}}} g_{ik},$$

where

$$g_{ik} = \pi_i \tau_k \exp\left(-\frac{1}{2} x_{ik}^T \eta_{ik} x_{ik}\right)$$

with

$$x_{ik} = \mu_i - R\nu_k - t, \quad \eta_{ik} = (\Sigma_i + R\Gamma_k R^T)^{-1}$$

and

$$t = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

define, respectively, a translation vector and a rotation matrix.

Taking the derivatives of the cost function in the composite manifold $<\mathbb{R}^2, SO(2)>$ the Jacobian vector is

$$J = \begin{bmatrix} \frac{\partial G}{\partial t_x} \\ \frac{\partial G}{\partial t_y} \\ \frac{\partial G}{\partial \theta} \end{bmatrix} = -\sum_{i=1}^{K_F}\sum_{k=1}^{K_M} g_{ik}\left\{ x_{ik}^T \eta_{ik}\begin{bmatrix} \frac{\partial t}{\partial t_x} & \frac{\partial t}{\partial t_y} & \frac{\partial R}{\partial \theta}v_k \end{bmatrix} + \begin{bmatrix} 0 & 0 & \frac{1}{2}x_{ik}^T \eta_{ik}\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\eta_{ik}x_{ik} \end{bmatrix}\right\},$$

where

$$\frac{\partial t}{\partial t_x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$
$$\frac{\partial t}{\partial t_y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$
$$\frac{\partial R}{\partial \theta} = \begin{bmatrix} -\sin(\theta) & -\cos(\theta) \\ \cos(\theta) & -\sin(\theta) \end{bmatrix},$$
$$\frac{\partial (R\Gamma_k R^T)}{\partial \theta} = \frac{\partial R}{\partial \theta}\Gamma_k R^T + \left(\frac{\partial R}{\partial \theta}\Gamma_k R^T\right)^T.$$

The Hessian matrix is

$$H = -\sum_{i=1}^{K_F}\sum_{k=1}^{K_M}\begin{bmatrix} \frac{\partial^2 g_{ik}}{\partial t_x^2} & \frac{\partial^2 g_{ik}}{\partial t_x \partial t_y} & \frac{\partial^2 g_{ik}}{\partial t_x \partial \theta} \\ \frac{\partial^2 g_{ik}}{\partial t_y \partial t_x} & \frac{\partial^2 g_{ik}}{\partial t_y^2} & \frac{\partial^2 g_{ik}}{\partial t_y \partial \theta} \\ \frac{\partial^2 g_{ik}}{\partial \theta \partial t_x} & \frac{\partial^2 g_{ik}}{\partial \theta \partial t_y} & \frac{\partial^2 g_{ik}}{\partial \theta^2} \end{bmatrix},$$

where

$$\frac{\partial^2 g_{ik}}{\partial t_a \partial t_b} = \frac{\partial^2 g_{ik}}{\partial t_b \partial t_a} = g_{ik}\left\{ x_{ik}^T \eta_{ik}\frac{\partial t}{\partial t_b}x_{ik}^T \eta_{ik}\frac{\partial t}{\partial t_a} - \frac{\partial t}{\partial t_a}^T \eta_{ik}\frac{\partial t}{\partial t_b}\right\},$$
$$\frac{\partial^2 g_{ik}}{\partial t_a \partial \theta} = \frac{\partial^2 g_{ik}}{\partial \theta \partial t_a} = g_{ik}\left\{ x_{ik}^T \eta_{ik}\left(\frac{\partial R}{\partial \theta}v_k x_{ik}^T + \frac{1}{2}\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\eta_{ik}x_{ik}x_{ik}^T - \frac{\partial (R\Gamma_k R^T)}{\partial \theta}\right)\eta_{ik}\frac{\partial t}{\partial t_a} - \frac{\partial t}{\partial t_a}^T \eta_{ik}\frac{\partial R}{\partial \theta}v_k\right\},$$
$$\frac{\partial^2 g_{ik}}{\partial \theta^2} = g_{ik}\left\{\left(x_{ik}^T \eta_{ik}\left(\frac{\partial R}{\partial \theta}v_k + \frac{1}{2}\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\eta_{ik}x_{ik}\right)\right)^2 - \left(\frac{\partial R}{\partial \theta}v_k\right)^T \eta_{ik}\frac{\partial R}{\partial \theta}v_k + x_{ik}^T \eta_{ik}\left(\frac{\partial^2 R}{\partial \theta^2} - 2\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\eta_{ik}\frac{\partial R}{\partial \theta}\right)v_k + \frac{1}{2}x_{ik}^T \eta_{ik}\left(\frac{\partial^2 (R\Gamma_k R^T)}{\partial \theta^2} - 2\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\eta_{ik}\frac{\partial (R\Gamma_k R^T)}{\partial \theta}\right)\eta_{ik}x_{ik}\right\}$$

and

$$\frac{\partial^2 R}{\partial \theta^2} = \begin{bmatrix} -\cos(\theta) & \sin(\theta) \\ -\sin(\theta) & -\cos(\theta) \end{bmatrix},$$
$$\frac{\partial^2 (R\Gamma_k R^T)}{\partial \theta^2} = \frac{\partial^2 R}{\partial \theta^2}\Gamma_k R^T + 2\frac{\partial R}{\partial \theta}\Gamma_k \frac{\partial R}{\partial \theta}^T + \left(\frac{\partial^2 R}{\partial \theta^2}\Gamma_k R^T\right)^T.$$

APPENDIX C: POSE MAP

Given a pose $\Omega = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$, we want to get the first derivative of the function $h(\Omega) : <\mathbb{R}^n, SO(n)> \rightarrow SE(n)$. To get this Jacobian, we start applying the definition of the derivative

$$J = \frac{\partial h(\Omega)}{\partial \Omega} = \lim_{\eta \to 0}\frac{h(\Omega \oplus \eta) \ominus h(\Omega)}{\eta} \tag{C1}$$

where the perturbation $\eta = (p, \theta)$ is defined in the tangent space of the composite manifold $<\mathbb{R}^n, SO(n)>$. Due to function $h(\Omega)$, the perturbation is added at the composite manifold using its sum operator $\oplus$ and the subtraction is done in the $SE(n)$ group applying $\ominus$. The mathematical development of these operators is

$$\begin{aligned} \Omega \oplus \eta &= \begin{bmatrix} RExp(\theta) & t + p \\ 0 & 1 \end{bmatrix}, \\ (\Omega \oplus \eta) \ominus \Omega &= \Omega^{-}1(\Omega \oplus \eta) = \begin{bmatrix} Exp(\theta) & R^T p \\ 0 & 1 \end{bmatrix}, \end{aligned} \tag{C2}$$

where $\mathrm{Exp}(\theta)$ is the $SO(n)$ exponential map and $\Omega^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix}$. Substituting Equations (29) into Equation (28) and applying the $SE(n)$ logarithm map we get

$$J = \lim_{\eta \to 0}\frac{\mathrm{Log}\begin{bmatrix} \mathrm{Exp}(\theta) & R^T p \\ 0 & 1 \end{bmatrix}}{\eta} = \lim_{\eta \to 0}\frac{\begin{bmatrix} V(\theta)^{-1}R^T p \\ \theta \end{bmatrix}}{\begin{bmatrix} p \\ \theta \end{bmatrix}},$$

which corresponds to the Euclidean definition of derivative. Therefore, taking the partial derivatives of the numerator through the perturbation η and applying the limit

$$J = \lim_{\eta \to 0}\begin{bmatrix} \frac{\partial (V(\theta)^{-1}R^T p)}{\partial p} & \frac{\partial (V(\theta)^{-1}R^T p)}{\partial \theta} \\ \frac{\partial \theta}{\partial p} & \frac{\partial \theta}{\partial \theta} \end{bmatrix} = \lim_{\eta \to 0}\begin{bmatrix} \left(I_{n\times n} - \frac{1}{2}[\theta]_\times\right)R^T & \frac{1}{2}[R^T p]_\times \\ 0_{n\times n} & I_{n\times n} \end{bmatrix} = \begin{bmatrix} -R^T & 0_{n\times n} \\ 0_{n\times n} & I_{n\times n} \end{bmatrix},$$

where

$$\frac{\partial(V(\theta)^{-1}R^T p)}{\partial\theta} = \frac{\partial\left(\left(I_{n\times n} - \frac{1}{2}[\theta]_\times\right)R^T p\right)}{\partial\theta} = \frac{\partial\left(R^T p + \frac{1}{2}[R^T p]_\times\theta\right)}{\partial\theta} = \frac{1}{2}[R^T p]_\times$$

taking into account that $V(\theta)^{-1} \simeq I_{n\times n} - \frac{1}{2}[\theta]_\times$ and $[a]_\times b = -[b]_\times a$ if $a, b \in \mathbb{R}^3$.

APPENDIX D: POSE PROJECTION

Given an SE(3) pose $\Omega = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$, we want to get the first derivative of the function $g(\Omega) : \mathrm{SE}(3) \to \mathrm{SE}(2)$. Assuming the plain motion imposed to the AUV—which means that the roll, pitch and depth variation is negligible—this function can be approximated by

$$\begin{bmatrix} t_x \\ t_y \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix},$$

where t_i are the components of t and θ_i are the exponential coordinates of R. To get the Jacobian of $g(\Omega)$, we start applying the definition of the derivative

$$J = \frac{\partial g(\Omega)}{\partial\Omega} = \lim_{\tau\to 0} \frac{g(\Omega \oplus \tau) \ominus g(\Omega)}{\tau}, \tag{D1}$$

where the perturbation $\tau = (\tau_\rho, \tau_\theta)$ is defined in the tangent space of SE(3). Due to the function $g(\Omega)$, the perturbation is added at the SE(3) and the subtraction is done in the SE(2) group by applying its respective plus and minus operators. Assuming the plain motion, the projection of the sum can be approximated by

$$g(\Omega \oplus \tau) = g(\Omega \mathrm{Exp}_{\mathrm{SE}(3)}(\tau)) \sim g(\Omega) g(\mathrm{Exp}_{\mathrm{SE}(3)}(\tau)),$$

where $\mathrm{Exp}_{\mathrm{SE}(3)}(\tau)$ is the SE(3) exponential map. Therefore, subtraction is approximated as

$$\begin{aligned} g(\Omega \oplus \tau) \ominus g(\Omega) &\sim \mathrm{Log}_{\mathrm{SE}(2)}\{g(\Omega)^{-1} g(\Omega) g(\mathrm{Exp}_{\mathrm{SE}(3)}(\tau))\} \\ &= \mathrm{Log}_{\mathrm{SE}(2)}\{g(\mathrm{Exp}_{\mathrm{SE}(3)}(\tau))\}, \end{aligned} \tag{D2}$$

where $\mathrm{Log}_{\mathrm{SE}(2)}(\chi)$ is the SE(2) logarithm map. Defining $A = [0 \quad 0 \quad 1]$ and $B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, the above expressions are developed as

$$\begin{aligned} g(\mathrm{Exp}_{\mathrm{SE}(3)}(\tau)) &= \begin{bmatrix} \mathrm{Exp}_{\mathrm{SO}(2)}(A\mathrm{Log}_{\mathrm{SO}(3)}(\mathrm{Exp}_{\mathrm{SO}(3)}(\tau_\theta))) & BV_3(\tau_\theta)\tau_\rho \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{Exp}_{\mathrm{SO}(2)}(A\tau_\theta) & BV_3(\tau_\theta)\tau_\rho \\ 0 & 1 \end{bmatrix}, \\ \mathrm{Log}_{\mathrm{SE}(2)}\{g(\mathrm{Exp}_{\mathrm{SE}(3)}(\tau))\} &= [V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho \\ &\quad \mathrm{Log}_{\mathrm{SO}(2)}(\mathrm{Exp}_{\mathrm{SO}(2)}(A\tau_\theta))]^T \\ &= [V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho \; A\tau_\theta]^T, \end{aligned} \tag{D3}$$

where $V_2()$ and $V_3()$ are functions that, respectively, form part of the SE(2) and the SE(3) logarithm map. Substituting Equations (D3) and (D2) into the derivative definition of Equation (D1) we get

$$J = \frac{\partial g(\Omega)}{\partial\Omega} = \lim_{\tau\to 0} \frac{\begin{bmatrix} V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho \\ A\tau_\theta \end{bmatrix}}{\begin{bmatrix} \tau_\rho \\ \tau_\theta \end{bmatrix}},$$

which corresponds to the Euclidean definition of derivative. Therefore, taking the partial derivatives of the numerator through the perturbation τ

$$\begin{aligned} \frac{\partial(V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho)}{\partial\tau_\rho} &= V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta) = \left(I_2 - \frac{1}{2}[A\tau_\theta]_\times\right) B\left(I_3 + \frac{1}{2}[\tau_\theta]_\times\right), \\ \frac{\partial(V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho)}{\partial\tau_\theta} &= \frac{\partial\left(\left(I_2 - \frac{1}{2}[A\tau_\theta]_\times\right) B\left(I_3 + \frac{1}{2}[\tau_\theta]_\times\right)\right)}{\partial\tau_\theta} \\ &= -\frac{1}{2}B[\tau_\rho]_\times + \frac{1}{2}[B\tau_\rho]_\times A + \frac{1}{4}[A\tau_\theta]_\times B[\tau_\rho]_\times \\ &\quad + \frac{1}{4}[B[\tau_\theta]_\times \tau_\rho]_\times A, \\ \frac{\partial(A\tau_\theta)}{\partial\tau_\rho} &= 0_{1\times 3}, \\ \frac{\partial(A\tau_\theta)}{\partial\tau_\theta} &= A, \end{aligned}$$

—assuming that $V(\theta) \simeq I_{n\times n} + \frac{1}{2}[\theta]_\times$, $V(\theta)^{-1} \simeq I_{n\times n} - \frac{1}{2}[\theta]_\times$ and $[a]_\times b = -[b]_\times a$ if $a, b \in \mathbb{R}^n$—and applying the limit; the Jacobian is

$$J = \lim_{\eta\to 0} \begin{bmatrix} \frac{\partial(V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho)}{\partial\tau_\rho} & \frac{\partial(V_2(A\tau_\theta)^{-1} BV_3(\tau_\theta)\tau_\rho)}{\partial\tau_\theta} \\ \frac{\partial(A\tau_\theta)}{\partial\tau_\rho} & \frac{\partial(A\tau_\theta)}{\partial\tau_\theta} \end{bmatrix} = \begin{bmatrix} B & 0_{2\times 3} \\ 0_{1\times 3} & A \end{bmatrix}.$$