On-manifold probabilistic Iterative Closest Point: Application to underwater karst exploration

Yohan Breux, André Mas, Lionel Lapierre

On-manifold Probabilistic ICP : Application to Underwater Karst Exploration.

Yohan Breux[a,*], André Mas[b], Lionel Lapierre[a]

[a]*LIRMM, Univ Montpellier, CNRS, 161 Rue Ada CEDEX 5, 34095 Montpellier, France*
[b]*IMAG, Univ Montpellier, CNRS, Place Eugène Bataillon 34090 Montpellier, France*

Abstract

This paper proposes MpIC, an on-manifold derivation of the probabilistic Iterative Correspondence (pIC) algorithm which is a stochastic version of the original Iterative Closest Point (ICP). It is developed in the context of autonomous underwater karst exploration based on acoustic sonars. First, a derivation of pIC based on the Lie group structure of SE(3) is developed. The closed-form expression of the covariance modeling the estimated rigid transformation is also provided. In a second part, its application to 3D scan matching between acoustic sonar measurements is proposed. It is a prolongation of previous work on elevation angle estimation from wide-beam acoustic sonar. While the pIC approach proposed is intended to be a key component in a Simultaneous Localization and Mapping (SLAM) framework, this paper focuses on assessing its viability on a unitary basis. As ground truth data in karst aquifer are difficult to obtain, quantitative experiments are carried out on a simulated karst environment and show improvement compared to previous state-of-the-art approach. The algorithm is also evaluated on a real underwater cave dataset demonstrating its practical applicability.

*Corresponding author
Email addresses: yohan.breux@gmail.com (Yohan Breux), andre.mas@umontpellier.fr (André Mas), lionel.lapierre@lirmm.fr (Lionel Lapierre)

1. Introduction

In this paper, we are interested in the Iterative Closest Point (ICP) algorithms firstly designed for 2D/3D shape registration and further applied in robotic applications to estimate rigid transformations between point clouds acquired by sensors (e.g. LIDAR, Acoustic sonar). Basically, it is an iterative procedure with two steps. First, the association step searches for corresponding points in both point clouds. It is followed by an optimization step minimizing the associated points distances with respect to the rigid transformation between the two input point clouds. In particular, we are interested in the ICP probabilistic extension first proposed by [1] called *probabilistic Iterative Correspondence* (pIC). It takes into account uncertainties in the positions of the points and in the initial displacement estimate by modeling them as Gaussian random variables. This is particularly important when using strongly noisy sensors such as underwater acoustic sonar. Furthermore, unlike LIDAR which provides full scans almost instantaneously, mechanically scanning sonar systems (MSIS) can take several seconds for a full 360° scan due to higher acoustic time-of-flight underwater. Besides, as the robot is moving during the acquisition, its pose uncertainty is also propagated to the sonar measurements.

In the context of underwater karst exploration, we have to rely on acoustic sonars as water turbidity affects the efficiency of vision-based and LIDAR approaches. pIC is then the natural candidate in a SLAM framework to estimate the displacement between two overlapping scans. To guarantee overlapping between two successive scans, we have to rely on wide-beam sonar. The drawback of such sonar is that the elevation angle in each measurement beam is unknown, leading to high uncertainty in the positions of the measured 3D points. Previous systems based on MSIS [2][3] deal with this problem by considering a 2D SLAM approach. In our previous work [4], we propose a method to estimate the probability distribution of elevation angles. We first compute a probabilistic surface model from the measurements of a vertically mounted narrow-beam sonar which is then leveraged to estimate the elevation angles from an horizon-

tally mounted wide-beam sonar. We thus obtain 3D measurements following non-Gaussian distributions.

SLAM algorithms' localization accuracy is directly related to the ICP/pIC performance. Our first contribution in this paper is to propose the derivation of on-manifold pIC (noted MpIC) as well as the derivation of the estimated transformation covariance. We show that it leads to superior performance compared to estimation based on Euler angles or quaternion representations. To the best of our knowledge, the literature [1][2][5][6] exploiting the pIC algorithms implicitly supposes an error covariance independent of the rigid transformation estimation to obtain their closed-form solution in the optimization step. In this work, we take into account the dependence of the error covariance on the transformation to estimate.

We then apply it in the context of underwater karst exploration based on an extension of our previous work [4]. In particular, we propose a Gaussian approximation to our estimated sonar measurements. Furthermore, in order to use a point-to-plane association scheme, we need to compute normals at each point. As we can expect sparse point clouds, we propose to compute the normals and their covariances based on the probabilistic surface model as obtained in [4]. To quantitatively assess our algorithm, we need to have access to ground truth displacements which are difficult to obtain in natural karst environment. We thus evaluate our algorithm on a simulated karst environment and compare it to the state-of-the-art 2D approach successfully used in real underwater cave [3][7].

The structure of the paper is as follows. Section 2 gives an overview on ICP approaches with a focus on underwater robotic applications. Section 3 briefly introduces generalities on Lie groups and particularly on the special Euclidean Group SE(3) which represents the rigid transformations of space. The description of pIC and our on-manifold derivation MpIC is given in Section 4. Section 5 succinctly introduces our previous elevation angle estimation method and its extensions with notably normal computations and Gaussian approximation of the estimated 3D points distribution. Finally, Section 6 evaluates our approach

through experiments on a simulated karst environment. Conclusion and future working directions are left to Section 7.

2. Related Works

The original ICP algorithm for 3D shapes registration comes from the seminal works by [8] and [9] for point-to-plane associations. A lot of improvements have been proposed by exploiting additional features such as surface orientations [10][11] or color [12][13], improving the association step [1][14] and/or the objective function [1][15][16]. The range of ICP applications is large and we are interested here in its usage in mobile robotics, especially with acoustic sonar measurements. A comprehensive overview can be found in [17].

In order to take into account sensor uncertainties, Montesano et al. [1] propose the *probabilist Iterative Correspondence* (pIC), a probabilistic approach of the original ICP algorithm. They consider both points and the relative transformation between the point clouds as Gaussian random variables. The Euclidean distance used in the ICP association step and in the objective function is replaced by the Mahalanobis distance. Doing so, each association error is weighted by its covariance so that uncertain data contribute less to the objective function.

Segal et al. [11] also introduce a probabilistic approach with their *generalized ICP* (G-ICP). The association step is done as in the original ICP using Euclidean distances. However, they consider points as Gaussian random variables with fixed uncertainties in the plane orthogonal to the surface at the considered point. While this does not directly model uncertainties related to the measurement, it takes into account the sampling error of the surface, e.g. the fact that corresponding points are not exactly the same point on the surface. Note that unlike pIC, the transformation is not considered as a random variable.

Agamennoni et al. [14] recently propose another probabilistic approach without explicit association phase. More precisely, they search to maximize the likelihood of points from the target cloud conditioned on the associated point in the reference cloud. Each possible association is weighted by a latent variable.

They use an Expectation-Maximization (EM) procedure to iteratively estimate the transformation and the associations weights.

In the field of underwater exploration, localization with SLAM approaches are often based on acoustic sonar scan matching. The level of noise of such sensors is higher than scan laser used in terrestrial robotics. Furthermore, due to the propagation time of acoustic wave in water and the inner mechanics to rotate the acoustic emitter (for Mechanical pencil beam sonars (MPBS). Electronically pencil beam sonar (EPBS) are still expensive and cumbersome), a full scan can take several seconds to complete while the robot is moving. We thus have to consider uncertainties in the robot position during the scan acquisition on top of measurement errors.

Palomer et al [6] develop an EKF SLAM framework for seafloor mapping. They use a multi-beam echosounder and the resulting scans are matched using pIC in 3D. However, the closed-form expression for the estimated transformation in the pIC error minimization step is obtained by implicitly considering an error covariance independent of the transformation. In fact, this covariance is dependent on the transformation through the jacobians involved in its definition. Furthermore, optimization is made using Euler angle representation which can be prone to gimbal lock singularity [18].

In this paper, we are interested in methods exploiting mechanically scanned imaging sonar (MSIS) as our practical application is based on those sensors. Mallios et al. [2][7] use the pIC algorithm [1] in an Extended Kalman Filter (EKF) SLAM framework for underwater cave exploration in a 2D setting. A similar approach is proposed by Burguera et al. [5][19]. Recently, McConnell and all [20] propose a method for mapping shallow water in a human-made littoral setting based on a pair of imaging sonars with orthogonal axes. They infer elevation angles from the overlapping of the sonars [21] and extends the mapping coverage with a Bayesian inference framework. As this method strongly depends on the structure of the environment, it is not appropriate for unstructured karst aquifer exploration. Furthermore, it does not provide the incertitude (or probability distribution) on the estimated 3D points so that it can't be used

in a pIC framework.

This work is the continuity of our previous work [4] which proposes a method for estimating the distribution of elevation angles from wide-beam sonar measurements by leveraging measurements from a secondary narrow-beam sonar. We first derive MpIC, a generic on-manifold pIC algorithm, with the closed-form expressions of the estimated transformation covariance. We then propose a practical application in the context of underwater karst exploration using MpIC to match 3D scan obtained by MSIS acoustic sonars where elevation angles are estimated using the method in [4].

3. Generalities on Lie groups

In this section, we introduce the notations and some differential geometry concepts relevant to the derivations exposed in the following sections. In particular, we focus on Lie Groups and more specifically on the 3D rigid transformation group SE(3). A good introduction to Lie groups for application in Robotics can be found in [22]. A more detailed coverage of differential geometry and Lie groups can be found in [23].

A N-dimensional manifold $\mathcal{M}$ is a topological space where every neighborhood V_p of a point $p \in \mathcal{M}$ is homeomorphic to $\mathbb{R}^N$.

A N-dimensional manifold embedded in $\mathbb{R}^D$, $N < D$, is said to be smooth (or differentiable) if every point $p \in \mathcal{M}$ can be locally parametrized by a C^∞-diffeomorphism $\phi : \Omega \to U$ with $p \in U$, $\mathbf{O}_N \in \Omega$ and (Ω, U) being respectively open subsets of $\mathbb{R}^N$ and $\mathcal{M}$. The tangent plane at a point p on $\mathcal{M}$ is designed by $T_p\mathcal{M}$.

A Lie group $\mathcal{G}$ is defined as a smooth manifold with a group structure such that its group product and its inverse are differentiable. We can then define its Lie algebra $\mathfrak{g} = T_e\mathcal{G}$ (a vector space equipped with a bilinear product called here the Lie bracket) which corresponds to the tangent space of $\mathcal{G}$ at its identity element e. From a practical point of view, computations can be easier on the Lie Algebra. In particular, it is useful for on-manifold non-linear optimization :

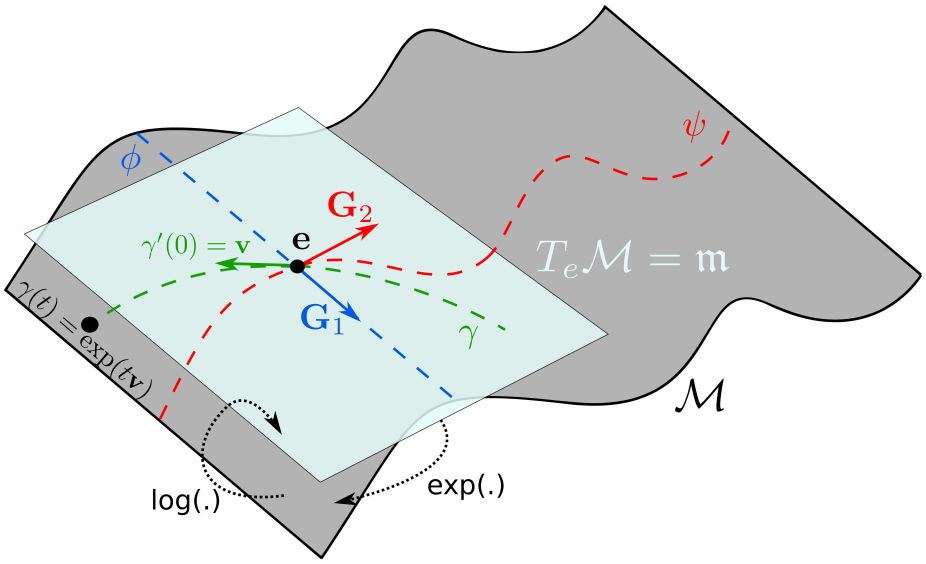


Figure 1: Representation of a 2-manifold $\mathcal{M}$ in $\mathbb{R}^3$ with its tangent space $T_e\mathcal{M}$ at a point $\mathbf{e} \in \mathcal{M}$. ϕ and ψ are two curves of $\mathcal{M}$ passing through e such that their tangent vectors G_1, G_2 at $\mathbf{e}$ generate a basis of $T_e\mathcal{M}$. If $\mathcal{M}$ is a Lie group and $\mathbf{e}$ its neutral element, then $T_e\mathcal{M} = \mathfrak{m}$ is its associated Lie algebra. The transition from one to another is done through the applications $exp : \mathfrak{m} \mapsto \mathcal{M}$ and $log : \mathcal{M} \mapsto \mathfrak{m}$. γ is an example of path with velocity $\mathbf{v}$.

methods such as Gauss-Newton or Levenberg-Marquardt are designed to work on vector spaces and not on general manifolds.

In the following, we only consider matrix Lie groups. By noting $GL(n)$ the general linear group in dimension n (set of inversible real $n \times n$ inversible matrices), the Von Neumann-Cartan theorem states that every closed subgroup of $GL(n)$ is a Lie group (Theorem 4.8, [23]).

The exponential map $exp : \mathfrak{g} \to \mathcal{G}$ maps elements of the algebra to the group. It is defined, in the case of matrix Lie group, as the exponential of a matrix defined by the following power series :

$$e^A = \sum_{k=0}^{+\infty} \frac{1}{k!} A^k$$

We note its inverse the logarithm map $log : \mathcal{G} \to \mathfrak{g}$.

We define the isomorphism $^\wedge$ which maps elements of $\mathbb{R}^N$ (local parametriza-

tion) to elements of the Lie algebra :

$$^{\wedge} : \begin{cases} \mathbb{R}^N & \to \mathfrak{g} \\ \mathbf{g} & \mapsto (\mathbf{g})^{\wedge} \equiv \widehat{\mathbf{g}} = \sum_{i=1}^{N} g_i G_i \end{cases} \tag{1}$$

where $\{G_i\}$ is a basis (called generators) of the Lie algebra. We denote its inverse mapping by the operator $^{\vee}$.

Smooth manifolds, and thus Lie groups, are also Riemannian manifolds (Theorem 14.2, [23]). Hence we can define a smooth inner product $< \cdot, \cdot >_p$ on the tangent space $T_p\mathcal{M}$ at each point $p \in \mathcal{M}$. The metric induced by this inner product is used to measure the length of piecewise-C^1 curves $\gamma : [a, b] \subset \mathbb{R} \to \mathcal{M}$ such that

$$L(\gamma) = \int_a^b \langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}^{\frac{1}{2}} dt \tag{2}$$

with the velocity $\gamma'(t) \in T_{\gamma(t)}\mathcal{M}, \forall t \in [a, b]$.

Geodesics are an important class of curves. They are the curves which locally minimize the length between two points and are equivalent to the straight line in Euclidean geometry (null curvature). In fact, by definition, geodesics are curves with constant velocity. This means that if γ is a geodesic of $\mathcal{M}$ then $\forall t, \gamma'(0) = \gamma'(t) = v \in T_{\gamma(0)}\mathcal{M}$. We can then define for a given velocity v a geodesic $\gamma(t) = e^{tv}$ going through a point $p \in \mathcal{M}$. It is illustrated in Figure 1.

In this paper, we are interested in two particular matrix Lie groups : the Special Orthogonal group $SO(3)$ representing the rotations and the Special Euclidean group $SE(3)$ representing the rigid transformations in $\mathbb{R}^3$.

3.1. The sets SO(3) and SE(3)

$SO(3)$ is a closed subgroup of $GL(3)$ defined as

$$SO(3) = \left\{ R \in GL(3) |\ RR^T = I_3 \text{ and } \det(R) = 1 \right\}$$

The corresponding Lie algebra $\mathfrak{so}(3)$ is defined by the space of skew-symmetric matrices

$$\forall W \in \mathfrak{so}(3), W = \widehat{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_{\times}\ ,\ \boldsymbol{\omega} \in \mathbb{R}^3$$

where the operator $[\ ]_\times$ is defined in Appendix A. The generators $G_i^{\mathfrak{so}(3)}$ of $\mathfrak{so}(3)$ are given by

$$G_i^{\mathfrak{so}(3)} = \widehat{\mathbf{e}}_i \tag{3}$$

where $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ is the canonical basis of $\mathbb{R}^3$.

$SE(3)$ is defined as the semidirect product $SO(3) \ltimes \mathbb{R}^3$ and is isomorphic to the affine group $GA(3)$ (a subgroup of $GL(4)$) through the application

$$\begin{cases} SO(3) \times \mathbb{R}^3 & \to GA(3) \\ R, \mathbf{t} & \mapsto T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \end{cases} \tag{4}$$

Its Lie algebra $\mathfrak{se}(3)$ is given by

$$\forall \Xi \in \mathfrak{se}(3),\ \Xi = \widehat{\boldsymbol{\xi}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{\tau} \\ \mathbf{0}^T & 0 \end{bmatrix}$$
$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\omega}^T & \boldsymbol{\tau}^T \end{bmatrix}^T, \boldsymbol{\omega} \in \mathbb{R}^3,\ \boldsymbol{\tau} \in \mathbb{R}^3$$

The generators $G_i^{\mathfrak{se}(3)}$ of $\mathfrak{se}(3)$ are given by

$$G_i^{\mathfrak{se}(3)} = \begin{bmatrix} G_i^{\mathfrak{so}(3)} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix},\ i \in \{1,2,3\}$$
$$G_i^{\mathfrak{se}(3)} = \begin{bmatrix} 0 & \mathbf{e}_i \\ \mathbf{0}^T & 0 \end{bmatrix},\ i \in \{4,5,6\} \tag{5}$$

Expressions for the exponential / log maps of $SO(3)$ and $SE(3)$ can be found in [22].

Throughout this paper, we use the left-invariant Riemannian metric induced by the point-independent Riemannian metric on $GA(3)$ (Section 3.2, [24])

$$\begin{aligned} \langle u, v \rangle_A &= Tr\left(\mathbf{u}^T \mathbf{v}\right) \\ &= \langle A^{-1}\mathbf{u}, A^{-1}\mathbf{v} \rangle_e \\ &= \langle \mathbf{s}_u, \mathbf{s}_v \rangle_e \\ &= \mathbf{s}_u^T G \mathbf{s}_v \end{aligned} \tag{6}$$

$$G = \begin{bmatrix} I_3 & 0 \\ 0 & 2I_3 \end{bmatrix}, A \in SE(3),\ \mathbf{u}, \mathbf{v} \in T_A SE(3)$$

where e is the identity element of $SE(3)$, $T_A SE(3)$ the tangent plane at A and $\mathbf{s}_u, \mathbf{s}_v \in \mathfrak{se}(3)$.

Pose-pose and pose-point compositions (resp. inverse compositions) are represented by the operator $\oplus$ (resp. $\ominus$). More precisely, for a 3D pose $q \in SE(3)$ with $SE(3)$ seen as a matrix group, the pose-pose composition corresponds to the group multiplication and the pose-point composition to the group action on $\mathbb{R}^3$.

We also define the $\boxplus$ operator used to compose a transformation with an increment expressed in the tangent vector space (local parametrization). It is given by

$$\boxplus : \begin{cases} \mathcal{G} \times \mathbb{R}^n & \to \mathcal{G} \\ G,\ \boldsymbol{\xi} & \mapsto G \boxplus \boldsymbol{\xi} = G e^{\widehat{\boldsymbol{\xi}}} \end{cases} \tag{7}$$

3.2. Jacobian on a Riemannian Manifold

Both ICP optimization step and uncertainty propagation involve computation of Jacobians and Hessians. As we work directly on SE(3), we recall in the following sections how to compute them on a Riemannian manifold.

For any smooth function $f : \mathcal{M} \to \mathbb{R}^m$ with $\mathcal{M}$ being a Riemannian manifold, the differential of the function f at $p \in \mathcal{M}$ is defined by

$$\forall u \in T_p\mathcal{M}, Df_p(u) = \frac{d}{dt} f\left(\gamma(t)\right)\Big|_{t=0} \tag{8}$$

where γ is a geodesic of $\mathcal{M}$ going through p ($\gamma(0) = p$) and with velocity u ($\gamma'(0) = u$). Note that this is equivalent to the definition given in [23](Definition 14.3). If $\mathcal{M}$ is a Lie group and $\mathfrak{m}$ its Lie algebra, it can be shown (Proposition 21.20, [23]) that the one-parameter subgroups $exp(tX_0),\ X_0 \in \mathfrak{m}$ are its geodesics.

In the case of a matrix Lie group $\mathcal{G}$ with its Lie algebra $\mathfrak{g}$, the geodesics are thus

$$\gamma(t) = T e^{t\widehat{g}},\ T \in \mathcal{G},\ \widehat{g} \in \mathfrak{g} \tag{9}$$

Put simply, $Df_p(u)$ is the directional derivative along the direction vector u. In the case of $SO(3)$ and $SE(3)$ manifolds, we can thus compute the Jacobian matrix of f at T, $J^f_{|T}$, by computing its derivatives along the generators G_i of $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$. Formally

$$J^f_{|T} = \left[Df_T \left(TG_j \right) \right] \tag{10}$$

Note that we have the following relation

$$J^f_{|T}\mathbf{g} = Df_T \left(T\widehat{g} \right), \ \mathbf{g} \in \mathbb{R}^n \tag{11}$$

The advantage of using this jacobian definition is to only operate univariate derivatives which keeps computation simple.

Jacobian computation on Lie groups could also be done directly using matrix expression leading to complex tensorial calculus involving matrix vectorization and kronecker products. While this approach gives exact matricial equalities, it is unnecessarily complex for implementation purpose. The definition proposed here gives simple expression for each element of the Jacobian which are easier to implement and computationally efficient.

By noting $q \equiv T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$, $\mathbf{a} \in \mathbb{R}^3$ and $\boldsymbol{\xi} \in \mathbb{R}^6$, the differentials of $SE(3)$ group action and its inverse are given by

$$D \oplus_{|\mathbf{q},\mathbf{a}} \left(T\widehat{\boldsymbol{\xi}} \right) = T\widehat{\boldsymbol{\xi}}\mathbf{a} \tag{12}$$

$$D \ominus_{|\mathbf{q},\mathbf{a}} \left(T\widehat{\boldsymbol{\xi}} \right) = -\widehat{\boldsymbol{\xi}}T^{-1}\mathbf{a} \tag{13}$$

Using (8) with (12)(13), jacobians of $SE(3)$ group action and its inverse can be computed and are given by

$$J^{\oplus}_{\mathbf{q}_{|\mathbf{q},\mathbf{a}}} \equiv \frac{\partial \mathbf{q} \oplus \mathbf{a}}{\partial \mathbf{q}} = R \begin{bmatrix} -[\mathbf{a}]_\times & I_3 \end{bmatrix} = RU_{\mathbf{a}}, \ U_{\mathbf{a}} = \begin{bmatrix} -[\mathbf{a}]_\times & I_3 \end{bmatrix} \tag{14}$$

$$J^{\oplus}_{\mathbf{a}_{|\mathbf{q},\mathbf{a}}} \equiv \frac{\partial \mathbf{q} \oplus \mathbf{a}}{\partial \mathbf{a}} = R \tag{15}$$

$$J^{\ominus}_{\mathbf{q}_{|\mathbf{q},\mathbf{a}}} \equiv \frac{\partial \mathbf{a} \ominus \mathbf{q}}{\partial \mathbf{q}} = \begin{bmatrix} \left[R^T\mathbf{a} - R^T t \right]_\times & -I_3 \end{bmatrix} = -R^T U_{\mathbf{a}-\mathbf{t}} \left(\mathbf{1}_2 \otimes R \right) \tag{16}$$

$$J^{\ominus}_{\mathbf{a}_{|\mathbf{q},\mathbf{a}}} \equiv \frac{\partial \mathbf{a} \ominus \mathbf{q}}{\partial \mathbf{a}} = R^T \tag{17}$$

with $\mathbf{1}_2 = [1 \ 1]^T$ and $\otimes$ the Kronecker product defined in Appendix A.

3.3. Hessian on a Riemannian Manifold

The hessian at p of a smooth function $f : \mathbb{R}^n \to \mathbb{R}$ is defined by

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n,\ Hess(f)_p(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t H_p \mathbf{y} \tag{18}$$

where $H_p = \left[\frac{\partial^2 f_i}{\partial x_i \partial x_j}\right]$ is the usual hessian matrix. Similarly, for a smooth function f on a C^{k+1} Riemannian manifold $\mathcal{M}$, its hessian at $p \in \mathcal{M}$ is denoted $Hess(f)_p(X, Y)$ where X, Y are two C^k- vector fields on $\mathcal{M}$(A C^k-vector field X is a mapping $p \in \mathcal{M} \to X(p) \in T_p\mathcal{M}$ such that $X(p)$ is C^k with respect to p. See Section 10.2, [23]). In particular, for $u \in T_p\mathcal{M}$, we have (Proposition 16.22, [23])

$$Hess(f)_p(u, u) = \frac{d^2}{dt^2} f\left(\gamma(t)\right)\Big|_{t=0} \tag{19}$$

where γ is a geodesic of $\mathcal{M}$ going through p ($\gamma(0) = p$) and with velocity u ($\gamma'(0) = u$).

Similarly to the differential, by noting that $T_T\mathcal{G} = T\mathfrak{g}$, the hessian matrix is deduced from the hessian operator and the vector fields $X_i : T \to TG_i \in T\mathfrak{g}$ such that

$$H_T^f = \Big[Hess(f)_T\left(TG_i, TG_j\right)\Big] \tag{20}$$

where G_i are the generators of $\mathfrak{g}$ as defined respectively in (3) and (5) for $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$.

Hence the diagonal terms of the hessian matrix are computed using (19). Other terms are obtained by exploiting the fact that the hessian is a symmetric bilinear form :

$$\begin{aligned} &Hess(f)_T\left(TG_i, TG_j\right) = \\ &\frac{1}{2}\Big(Hess(f)_T\left(TG_i + TG_j, TG_i + TG_j\right) \\ &- Hess(f)_T\left(TG_i, TG_i\right) - Hess(f)_T\left(TG_j, TG_j\right)\Big) \end{aligned} \tag{21}$$

4. Probabilistic Iterative Correspondence (pIC)

The Iterative Closest Point (ICP) algorithm is an iterative non-linear 2-step process for registering unstructured 2D or 3D point clouds. In other words, it

aims at estimating the displacement (rigid transformation) between two point clouds. Given an initial displacement, each iteration consists in associating points in the first cloud to points in the displaced second cloud.

The algorithm then computes, by means of non-linear optimization, the displacement minimizing the sum of errors between each pair of associated points. Originally, the error is defined by the Euclidean distance between associated points.

A probabilistic variant taking into account point uncertainties is proposed by [1] and further exploited in a similar context to our work in 2D [2] and 3D [6].

In those works, the optimization is based on Euler angles representation and the error covariance is implicitly assumed to be independent of the estimated transformation. We improve upon previous iterations with two main contributions : we directly optimize on the SE(3) manifold using derivative on its Lie algebra and we also take into account the error covariance dependence on the estimated displacement.

The inputs of the algorithm are the reference 3D point cloud $S_{ref} = \{\mathbf{r}_i\}_{i=1\ldots n}$, the newly acquired 3D point cloud $S_{new} = \{\mathbf{c}_i\}_{i=1\ldots m}$ and an initial transformation $\mathbf{q}^{(0)}$ such that

$$\mathbf{r}_i \sim \mathcal{N}\left(\bar{\boldsymbol{r}}_i, \Sigma_{r_i}\right),\ \mathbf{c}_i \sim \mathcal{N}\left(\bar{\boldsymbol{c}}_i, \Sigma_{c_i}\right) \tag{22}$$

$$\mathbf{q}^{(0)} \sim \mathcal{N}\left(\bar{\boldsymbol{q}}^{(0)} \in SE(3), \Sigma_{\mathbf{q}^{(0)}}\right) \tag{23}$$

where the notation $\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{x}}, \Sigma_x)$ represents a nD random variable following a Gaussian distribution with mean $\bar{\mathbf{x}} \in \mathbb{R}^n$ and covariance matrix $\Sigma_x \in \mathbb{S}^n_{++}$ where $\mathbb{S}^n_{++}$ is the space of $n \times n$ definite positive matrix. Note that we do not need to specify how we represent the transformation $\mathbf{q}$. Generally, it is represented as a 7D vector made up of a quaternion for the rotation part and a 3D vector for the translation part. It can also be represented as a 6D vector using Euler angles for the rotation part.

We now consider the k-th iteration. The current displaced point cloud is

$S_{new}^{(k)} = \left\{ \mathbf{n}_i^{(k)} \right\}_{i=1...m}$ defined by

$$\mathbf{n}_i^{(k)} = \mathbf{q}^{(k)} \oplus \mathbf{c}_i \sim \mathcal{N}\left(\bar{\boldsymbol{n}}_i^{(k)},\ \Sigma_{\mathbf{n}_i^{(k)}} \right) \tag{24}$$

The mean and covariance of $\mathbf{n}_i^{(k)}$ are given by

$$\bar{\boldsymbol{n}}_i^{(k)} = \bar{\boldsymbol{q}}^{(k-1)} \oplus \bar{\boldsymbol{c}}_i \tag{25}$$

$$\Sigma_{\mathbf{n}_i^{(k)}} = J^{\oplus}_{\mathbf{q}_{|\bar{\boldsymbol{q}}^{(k-1)}, \bar{\boldsymbol{n}}_i}} \Sigma_{q^{(0)}} J^{\oplus^T}_{\mathbf{q}_{|\bar{\boldsymbol{q}}^{(k-1)}, \bar{\boldsymbol{n}}_i}} + J^{\oplus}_{\mathbf{a}_{|\bar{\boldsymbol{q}}^{(k-1)}, \bar{\boldsymbol{n}}_i}} \Sigma_{\mathbf{c}_i} J^{\oplus^T}_{\mathbf{a}_{|\bar{\boldsymbol{q}}^{(k-1)}, \bar{\boldsymbol{n}}_i}} \tag{26}$$

Note that the covariance is obtained with a first-order approximation. The involved jacobians $J^{\oplus}_{\mathbf{q}}$ and $J^{\oplus}_{\mathbf{a}}$ are respectively defined by (14) and (15).

4.1. Associations

In the association step, we search the set of correspondence candidates $A_i^{(k+1)} \subset S_{ref}$ for each point $\mathbf{n}_i^{(k+1)} \in S_{new}^{(k+1)}$. In the following, the errors $\mathbf{e}_{ij} \sim \mathcal{N}\left(\bar{\boldsymbol{e}}_{ij}, \Sigma_{e_{ij}}\right)$ between two points $\mathbf{n}_i^{(k+1)}$ and $\mathbf{r}_j$ are defined by

$$\bar{\boldsymbol{e}}_{ij} = \bar{\boldsymbol{n}}_i^{(k+1)} - \bar{\boldsymbol{r}}_j = \left(\bar{\boldsymbol{q}}^{(k)} \oplus \bar{\boldsymbol{c}}_j \right) - \bar{\boldsymbol{r}}_j \tag{27}$$

$$\Sigma_{e_{ij}} = \Sigma_{\mathbf{n}_i^{(k+1)}} + \Sigma_{\mathbf{r}_j} \tag{28}$$

The Mahalanobis distance $D^2(\mathbf{n}_i, \mathbf{r}_j)$ between a pair of points is given by

$$D^2(\mathbf{n}_i, \mathbf{r}_j) = \bar{\boldsymbol{e}}_{ij}^T \Sigma_{e_{ij}}^{-1} \bar{\boldsymbol{e}}_{ij} \tag{29}$$

As the previous Mahalanobis distance follows a χ^2 distribution with 3 degrees of freedom (DoF), we consider a valid candidate for association if the distance is inferior to the critical value for a confidence level $\alpha \in [0, 1]$:

$$D^2(\mathbf{n}_i, \mathbf{r}_j) < \chi^2_{3,\alpha} \tag{30}$$

In the case of *point-to-point* association, the final association $\mathbf{a}_i \in A_i^{(k+1)}$ for $\mathbf{n}_i^{(k+1)}$ is simply the element in $A_i^{(k+1)}$ which minimizes $D^2(\mathbf{n}_i, \cdot)$:

$$\mathbf{a}_i = \arg \min_{\mathbf{r} \in A_i^{(k+1)}} D^2(\mathbf{n}_i, \mathbf{r}) \tag{31}$$

The drawback of the *point-to-point* association is to suppose that the exact same points belong to both point clouds. In practice, the two inputs point cloud result from two different samplings of the environment leading to misalignment.

Complementary to the *point-to-point* association, the *point-to-plane* association uses local information on the surface shape and performs better against misalignment. The general approach is then to first use *point-to-point* association for the first iterations and to switch later on to *point-to-plane* association.

First, we need to estimate the tangent planes

$$\mathbf{p}_i = \begin{bmatrix} \mathbf{v}_i^T & d_i \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \bar{\boldsymbol{v}}_i^T & \bar{d}_i \end{bmatrix}, \Sigma_{\mathbf{p}_i} \right)$$

defined by $\mathbf{v}_i^T \mathbf{x} = d_i$ at the surface from which each points has been measured. If the point clouds are dense enough, the classic approach is to fit a plane on the candidate sets $A_i^{(k+1)}$. In [6] an efficient method to find the normals and their associated covariance is proposed. It is summarized in Algorithm 1. In our work, we consider rather sparse clouds so that this method is not practicable. However, in Section 5.3, we propose a novel method to estimate the normals based on a Gaussian process regression of the environment surface.

Once the normals are computed, the final association $\mathbf{a}_i = \mathbf{a}_i^\perp \sim \mathcal{N}(\bar{\boldsymbol{a}}_i^\perp, \Sigma_{\mathbf{a}_i^\perp})$ for $\mathbf{n}_i$ is the orthogonal projection of $\mathbf{n}_i$ on the plane $\mathbf{p}_i$ and is given by

$$\bar{\boldsymbol{a}}_i^\perp = \bar{\boldsymbol{n}}_i - \left(\bar{\boldsymbol{n}}_i^T \bar{\boldsymbol{v}}_i - \bar{d}_i \right) \bar{\boldsymbol{v}}_i \tag{32}$$

$$\Sigma_{\mathbf{a}_i^\perp} = J_{\mathbf{n}_i}^{\mathbf{a}_i^\perp} \Sigma_{\mathbf{n}_i} J_{\mathbf{n}_i}^{\mathbf{a}_i^{\perp T}} + J_{\mathbf{p}_i}^{\mathbf{a}_i^\perp} \Sigma_{\mathbf{p}_i} J_{\mathbf{p}_i}^{\mathbf{a}_i^{\perp T}} \tag{33}$$

In both associations schemes, the final errors $\mathbf{e}_i$ are defined by

$$\bar{\boldsymbol{e}}_i = \bar{\boldsymbol{n}}_i - \bar{\boldsymbol{a}}_i \tag{34}$$

$$\Sigma_{\mathbf{e}_i} = \Sigma_{\mathbf{n}_i} + \Sigma_{\mathbf{a}_i} \tag{35}$$

Developing (35) with (14)(15)(26), we can write the covariance as follows

$$\Sigma_{\mathbf{e}_i} = \Sigma_{\mathbf{a}_i} + R \Omega_{\bar{\boldsymbol{c}}_i} R^T \tag{36}$$

$$\Omega_{\bar{\boldsymbol{c}}_i} = \Sigma_{\mathbf{c}_i} + U_{\bar{\boldsymbol{c}}_i} \Sigma_{\mathbf{q}} U_{\bar{\boldsymbol{c}}_i}^T \tag{37}$$

where R is the rotation matrix related to the current mean transformation $\bar{\mathbf{q}}$ and $U_{\bar{\boldsymbol{c}}_i}$ defined in (14).

Algorithm 1 PICP Association

Input: Association scheme T_{assoc}, Transformation $\mathbf{q}$, Gaussian point $\mathbf{n} \in \mathbf{q} \oplus S_{new}$, Candidate set $A \subset S_{ref}$
Output: Gaussian point $\mathbf{a} \in S_{ref}$ associated with $\mathbf{n}$

if (T_{assoc} = point-to-point) **then**
  $\mathbf{a} = \arg\min_{\mathbf{r} \in A} D^2(\mathbf{n}, r)$
else if (T_{assoc} = point-to-plane) **then**
  $\alpha = \sum_{\bar{\boldsymbol{r}} \in A} \mathrm{Tr}(\Sigma_{\mathbf{r}})^{-2}$
  $\mathbf{p}_\mu = \left(\sum_{\mathbf{r} \in A} \mathrm{Tr}(\Sigma_{\mathbf{r}})^{-2} \bar{\boldsymbol{r}}\right) \alpha^{-1}$
  $C = \sum_{\mathbf{r} \in A} \frac{(\bar{\boldsymbol{r}} - \mathbf{p}_\mu)(\bar{\boldsymbol{r}} - \mathbf{p}_\mu)^T}{Tr(\Sigma_{\mathbf{r}}^2)}$
  $K = \sum_{\mathbf{r} \in A} \frac{\bar{\boldsymbol{r}}(\bar{\boldsymbol{r}} - \mathbf{p}_\mu)^T}{Tr(\Sigma_{\mathbf{r}}^2)}$
  $H = \begin{bmatrix} K - \bar{\boldsymbol{v}}^T K^T \bar{\boldsymbol{v}} & \mathbf{p}_\mu \\ \mathbf{p}_\mu^T & \alpha \end{bmatrix}$
  $\bar{\boldsymbol{v}}$ = Eigen vector of C associated with minimal eigen value
  $\bar{d} = \bar{\boldsymbol{v}}^T \mathbf{p}_\mu$
  $\mathbf{p} \sim \mathcal{N}\left([\bar{\boldsymbol{v}}\ \bar{d}]^T, H^+\right)$
  $\bar{\boldsymbol{a}} = \bar{\boldsymbol{q}} \oplus \bar{\boldsymbol{n}} - \left((\bar{\boldsymbol{q}} \oplus \bar{\boldsymbol{n}})^T \bar{\boldsymbol{v}} - \bar{\boldsymbol{d}}\right) \bar{\boldsymbol{v}}$
  $\Sigma_a = J^{\mathbf{a}}_{\mathbf{q}} \Sigma_{\mathbf{q}} J^{\mathbf{a}^T}_{\mathbf{q}} + J^{\mathbf{a}}_{\mathbf{n}} \Sigma_{\mathbf{n}} J^{\mathbf{a}^T}_{\mathbf{n}} + J^{\mathbf{a}}_{\mathbf{p}} \Sigma_{\mathbf{p}} J^{\mathbf{a}^T}_{\mathbf{p}}$
  $\mathbf{a} \sim \mathcal{N}(\bar{\boldsymbol{a}}, \Sigma_a)$
end if

4.2. Optimization

After the association step, we have two sets of N associated points $\{a_i\}$ and $\{n_i\}$ with corresponding errors $\mathbf{e}_i = \mathbf{n}_i - \mathbf{a}_i$. The optimization step consists in searching for the transformation $\mathbf{q}^{(k+1)}$ minimizing the squared Mahalanobis norm of errors

$$\mathbf{q}^{(k+1)} = \arg \min_{\mathbf{q} \in SE(3)} F = f^2 = \bar{\boldsymbol{e}}^T \Sigma_{\mathbf{e}}^{-1} \bar{\boldsymbol{e}} \tag{38}$$

where the errors are concatenated into a single vector $\bar{\boldsymbol{e}} = \left[\bar{\boldsymbol{e}}_i^T\right]^T$. The covariance matrix $\Sigma_{\mathbf{e}}^{-1}$ is a block-diagonal matrix with its i-th block being $\Sigma_{\mathbf{e}_i}$. Equation (38) is generally solved iteratively using non-linear least square method such as the Levenberg-Marquardt (LM) algorithm.

In this work, we optimize directly on the $SE(3)$ manifold. If we use the superscript k for the current ICP iteration as a subscript, the l-th iteration of

LM solves

$$\begin{aligned}\boldsymbol{\xi}^{(l)} &= \arg\min_{\boldsymbol{\xi}\in\mathbb{R}^6} f^2\left(\bar{\boldsymbol{q}}_k^{(l-1)} \boxplus \boldsymbol{\xi}\right) \\ &= -\left(J_{\mathbf{q}}^{f^{(l)}\,T} J_{\mathbf{q}}^{f^{(l)}} + \lambda^{(l)} I\right)^{-1} J_{\mathbf{q}}^{f^{(l)}} f\left(\bar{\boldsymbol{q}}_k^{(l-1)}\right) \qquad (39)\\ \bar{\boldsymbol{q}}_k^{(l)} &= \bar{\boldsymbol{q}}_k^{(l-1)} \boxplus \boldsymbol{\xi}^{(l+1)} \qquad (40)\end{aligned}$$

with here $\bar{\boldsymbol{q}}_k^{(0)} = \bar{\boldsymbol{q}}^{(k)}$ and λ the dumping parameter of LM. Note here the difference with equation (32) from [6]. In our case, we do not make the assumption that $\Sigma_{\mathbf{e}}$ is constant relative to the displacement $\mathbf{q}$. To alleviate the notation, we omit the LM iteration superscript in the following.

Proposition 1. *The differential $DF_{\mathbf{q}}$ of $F = f^2$ at $\mathbf{q} \in SE(3)$ can be expressed as*

$$DF_{\mathbf{q}}(T\widehat{\boldsymbol{\xi}}) = \sum_{i=1}^{N} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R\Big(2U_{\bar{\boldsymbol{c}}_i}\boldsymbol{\xi} + \left(\Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} - \widehat{\boldsymbol{\omega}}\Omega_{\bar{\boldsymbol{c}}_i}\right) R^T \Sigma_{\mathbf{e}_i}^{-1} \bar{\boldsymbol{e}}_i\Big) \qquad (41)$$

with $\widehat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ and $U_{\bar{\boldsymbol{c}}_i}, \Omega_{\bar{\boldsymbol{c}}_i}$ defined by (14)(37). The jacobian $J_{|\mathbf{q}}^F$ is obtained using (10) and $J_{|\mathbf{q}}^f$ follows by

$$J_{|\mathbf{q}}^f = \frac{1}{2f(\mathbf{q})} J_{|\mathbf{q}}^F \qquad (42)$$

In particular, note that

$$\forall i \in \{4,5,6\},\ \widehat{\boldsymbol{\xi}} = G_i^{SE(3)} \implies \widehat{\boldsymbol{\omega}} = 0 \qquad (43)$$

This leads to a simple expression for the last 3 columns of $J_{|\mathbf{q}}^F$

$$J_{|\mathbf{q}}^F[4:6] = 2\sum_{i=1}^{N} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R = \bar{\boldsymbol{e}}^T \Sigma_{\mathbf{e}}^{-1} \left(\mathbf{1}_N \otimes R\right) \qquad (44)$$

The proof is given in Appendix B.1.

4.3. Covariance estimation

In this section, we are interested in estimating the covariance $\Sigma_{\mathbf{q}}$ of the previously optimized transformation $\mathbf{q}$. A closed-form expression is given by [25] as follows

$$\begin{aligned}\Sigma_{\mathbf{q}} &= \left(\frac{\partial^2 F}{\partial \mathbf{q}^2}\right)^{-1} \frac{\partial^2 F}{\partial \mathbf{q}\partial \mathbf{z}} \Sigma_z \frac{\partial^2 F}{\partial \mathbf{q}\partial \mathbf{z}}^T \left(\frac{\partial^2 F}{\partial \mathbf{q}^2}\right)^{-1} \\ &= \left(H_{\mathbf{q}}^F\right)^{-1} H_{\mathbf{q},\mathbf{z}}^F \Sigma_z H_{\mathbf{q},\mathbf{z}}^{F^T} \left(H_{\mathbf{q}}^F\right)^{-1} \qquad (45)\end{aligned}$$

with $F = f^2$ and the hessian of F :

$$H^F = \begin{bmatrix} H^F_{\mathbf{q}} & H^F_{\mathbf{q},\mathbf{z}} \\ H^F_{\mathbf{q},\mathbf{z}} & H^F_{\mathbf{z}} \end{bmatrix} \tag{46}$$

The Gaussian vector $\mathbf{z} = [\mathbf{z}_1^T \dots \mathbf{z}_p^T]^T$ with $\mathbf{z}_i^T = [\mathbf{a}_i^T \ \mathbf{c}_i^T]$ is a vector of dimension $6p$ concatenating the points from both clouds. Its covariance matrix is defined by the block-diagonal matrix of dimension $6p \times 6p$

$$\Sigma_{\mathbf{z}} = \begin{bmatrix} \Sigma_{a_1} & & \dots & \dots & & 0 \\ & \Sigma_{c_1} & & & & \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & \vdots \\ & & & & \Sigma_{a_p} & \\ 0 & & \dots & \dots & & \Sigma_{c_p} \end{bmatrix} \tag{47}$$

Proposition 2. *The hessian $Hess(F)_{\mathbf{q}}$ for all $\widehat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$ is given by*

$$\begin{aligned} Hess(F)_{\mathbf{q}}\left(T\widehat{\boldsymbol{\xi}}, T\widehat{\boldsymbol{\xi}}\right) = \sum_{i=0}^{N} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R\Big(2\widehat{\boldsymbol{\omega}}\left(\widehat{\boldsymbol{\omega}}\bar{\boldsymbol{c}}_i + \tau\right) + D_{\omega_i} R^T \Sigma_{\mathbf{e}_i}^{-1} \bar{\boldsymbol{e}}_i \\ + K_{\omega_i} R\left(2K_{\omega_i}\bar{\boldsymbol{e}}_i + 3U_{\bar{\boldsymbol{c}}_i}\boldsymbol{\xi}\right)\Big) + \boldsymbol{\xi}^T U_{\bar{\boldsymbol{c}}_i}^T R^T \Sigma_{\mathbf{e}_i}^{-1} R\Big(2U_{\bar{\boldsymbol{c}}_i}\boldsymbol{\xi} + K_{\omega_i}\bar{\boldsymbol{e}}_i\Big) \end{aligned} \tag{48}$$

with

$$\Lambda_{\omega_i} = \Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} \tag{49}$$

$$D_{\omega_i} = 2\widehat{\boldsymbol{\omega}}\Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} - \Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}}^2 - \widehat{\boldsymbol{\omega}}^2\Omega_{\bar{\boldsymbol{c}}_i} \tag{50}$$

$$K_{\omega_i} = (\Lambda_{\omega_i} + \Lambda_{\omega_i}^T) R^T \Sigma_{\mathbf{e}_i}^{-1} \tag{51}$$

The corresponding hessian matrix $H^F_{\mathbf{q}}$ is obtained using (20) and (21).

The derivation is given in Appendix B.2. For $H^F_{\mathbf{q},\mathbf{z}}$, the expressions are more involved and are provided in Section Appendix B.2.2 .

All the jacobian and hessian expressions are implemented and tested as a part of our MpIC algorithm. The source code used in this paper is available at `https://gite.lirmm.fr/breux/mpic`.

Algorithm 2 On-manifold Probabilistic Iterative Closest Point

Input: Initial gaussian transformation $\mathbf{q}$, point clouds $S_{ref} = \{\mathbf{r}_i\}$, $S_{new} = \{\mathbf{c}_i\}$, Maximum iteration k_{max}, Association scheme T_{assoc}
Output: Relative gaussian transformation $\mathbf{q}^*$ between S_{new} and S_{ref}

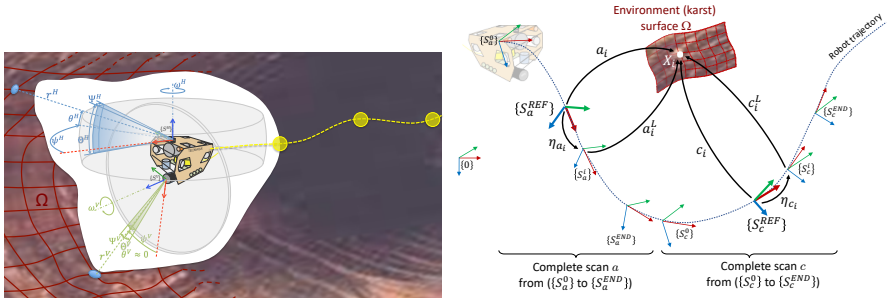
$k = 0,\ \mathbf{q}^* = \mathbf{q}$
do
  for $(i = 1; i \leq m; i++)$ **do**
    $\bar{\boldsymbol{n}}_i = \bar{\boldsymbol{q}}^{(k)} \oplus \bar{\boldsymbol{c}}_i$
    $\Sigma_{\mathbf{n}_i} = J^{\oplus}_{\mathbf{q}} \Sigma_{\mathbf{q}} J^{\oplus^T}_{\mathbf{q}} + J^{\oplus}_{\mathbf{a}} \Sigma_{\mathbf{c}_i} J^{\oplus^T}_{\mathbf{a}}$
    $A_i = \left\{ \mathbf{r} \in S_{ref} | D^2(\mathbf{n}_i, \mathbf{r}) < \chi^2_{3,\alpha} \right\}$
    $\mathbf{a}_i = \textbf{PICP\_Association}(T_{assoc}, A_i, \mathbf{n}_i, S_{ref})$
    $\bar{\boldsymbol{e}}_i = \mathbf{n}_i - \mathbf{a}_i$
    $\Sigma_{\mathbf{e}_i} = \Sigma_{\mathbf{n}_i} + \Sigma_{\mathbf{a}_i}$
  end for
  $F = \frac{1}{2} \sum_{i=1}^{m} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i} \bar{\boldsymbol{e}}_i$
  for $(l = 0; l < l_{max}; l++)$ **do**
    $\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} F(\bar{\boldsymbol{q}}^{(k)} \boxplus \boldsymbol{\xi})$
    $\bar{\boldsymbol{q}}^{(k+1)} = \bar{\boldsymbol{q}}^{(k)} \boxplus \boldsymbol{\xi}^*$
  end for
  $k++$
while (!hasConverged() and $k < k_{max}$)
$\bar{\boldsymbol{q}}^* = \bar{\boldsymbol{q}}^{(k)}$
$\Sigma_{\mathbf{q}^*} = \left(H^F_{\mathbf{q}}\right)^{-1} H^F_{\mathbf{q},\mathbf{z}} H^{F\ T}_{\mathbf{q},\mathbf{z}} \left(H^F_{\mathbf{q}}\right)^{-1}$

5. Application to Sonar Data with unknown elevation angle

In this section, we propose an application of our MpIC algorithm to acoustic sonar data in the context of underwater karst exploration. Acoustic sonar range measurements are obtained with beams which could have large aperture. Thus, the elevation angle θ of each measurement is generally unknown and the 3D information is lost. In our previous work [4], we propose a method which leverages the data acquired with a narrow-beam acoustic sonar to estimate the elevation angles from a wide-beam acoustic sonar. In the following, we briefly introduce the original approach and its extensions used in this paper.

5.1. Notations

In the following, we reuse the notation introduced in Section 4 with some additions. We denote by b the sonar beam-width such that each elevation angle θ is in the range $\left[-\frac{b}{2},\ \frac{b}{2}\right]$. We also define $\boldsymbol{\eta}_{\mathbf{p}_i} \in SE(3)$ as the poses relative to



(a) Illustration of the robot mounted with two perpendicular sonars

(b) Pair of scans

Figure 2: Notations related to the robot sensing and local scans. Note that in practice, associated points $\mathbf{a}_i, \mathbf{c}_i$ does not correspond to the same 3D point X_i on the surface.

the current scan reference frame at which the data point $\mathbf{p}_i$ has been measured. The corresponding coordinates in the sonar local frame are denoted $\mathbf{p}_i^L$. The function g maps local spherical coordinates to local Cartesian coordinates and is given by

$$g : \begin{cases} \mathbb{R}^+ \times \left[-\frac{b}{2}, \frac{b}{2}\right] \times [0, 2\pi] \to \mathbb{R}^3 \\ \rho, \theta, \phi \mapsto \mathbf{p}^L = \rho \begin{bmatrix} \cos(\theta)\cos(\phi) \\ \cos(\theta)\sin(\phi) \\ \sin(\theta) \end{bmatrix} \end{cases} \tag{52}$$

Similarly, we define the function h which maps the local spherical coordinates to the global reference frame in Cartesian coordinates

$$h : \begin{cases} \mathbb{R}^+ \times [-\frac{b}{2}, \frac{b}{2}] \times [0, 2\pi] \times SE(3) \to \mathbb{R}^3 \\ \rho, \theta, \phi, \boldsymbol{\eta} \mapsto \boldsymbol{\eta} \oplus g(\rho, \theta, \phi) \end{cases} \tag{53}$$

Hence for each data point $\mathbf{p}$ from any of the two point clouds

$$\exists \rho, \theta, \phi, \text{ s.t. } \mathbf{p} = h\left(\rho, \theta, \phi, \boldsymbol{\eta}_{\mathbf{p}}\right) \tag{54}$$

Figures 2a and 2b illustrate the notation and the problem configuration.

In the sonar local frame, the measurements in spherical coordinates $[\rho\ \theta\ \phi]^T$

can be modelled as the following independent random variables

$$\rho_i \sim Gamma\left(\alpha_{\rho_i}, \beta_{\rho_i}\right) \tag{55}$$

$$\theta_i \sim SBeta\left(\alpha_i, \beta_i, -\frac{b}{2}, \frac{b}{2}\right) \tag{56}$$

$$\phi_i \sim \mathcal{N}\left(\bar{\phi}_i, \sigma^2_{\phi_i}\right) \tag{57}$$

where $SBeta(\alpha, \beta, min, max)$ is a shifted and scaled Beta distribution. Its probability density function is defined by the following function

$$f(\theta; \alpha, \beta, b) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\left(\theta + \frac{b}{2}\right)^{\alpha-1} \left(\frac{b}{2} - \theta\right)^{\beta-1}}{b^{\alpha+\beta-1}} \tag{58}$$

with Γ the Gamma function.

In practice, the mean and variance of ρ are such that it can be approximated by a normal distribution. In the following we use this approximation by noting

$$\rho_i \sim \mathcal{N}\left(\bar{\rho}_i, \sigma^2_{\rho_i}\right) \tag{59}$$

Note that θ follows a $Beta$ distribution even when no prior information is available for the elevation angle θ. Indeed, in this case, θ follows a uniform distribution $\theta \sim \mathcal{U}\left(\frac{b}{2}, \frac{b}{2}\right)$ which is also a Beta distribution as we have

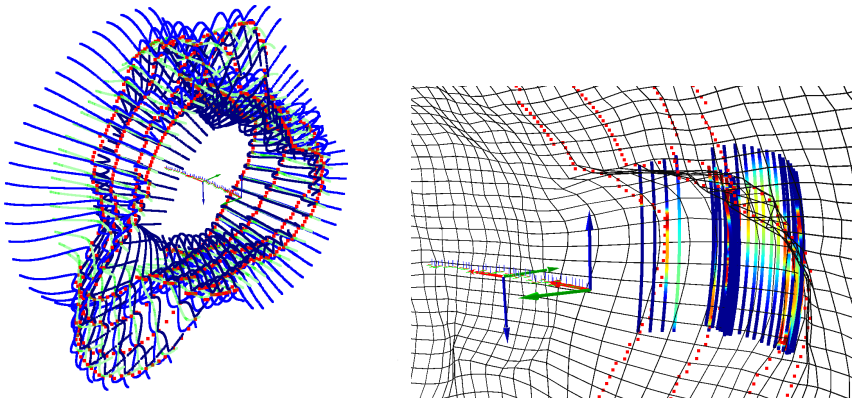
$$\forall a, b \in \mathbb{R},\ \mathcal{U}(a, b) = SBeta(1, 1, a, b) \tag{60}$$

In order to estimate the displacement between two points cloud following the presented distribution, we first need to use a Gaussian approximation for the points in global Cartesian coordinates. We can then apply the on-manifold pIC as shown in Section 4.

5.2. Elevation angles estimation

In this section, we briefly cover our previous work [4] which focuses on estimating the elevation angles from a wide-beam sonar. We also cover some improvements made in the current implementation used in this paper.

We consider a robot mounted with two perpendicular sonars : one placed vertically with narrow beams and the other placed horizontally with wide beams.



(a) Surface estimation

(b) Scaled-beta distribution of estimated elevation angles for horizontal sonar measurements from the first scan

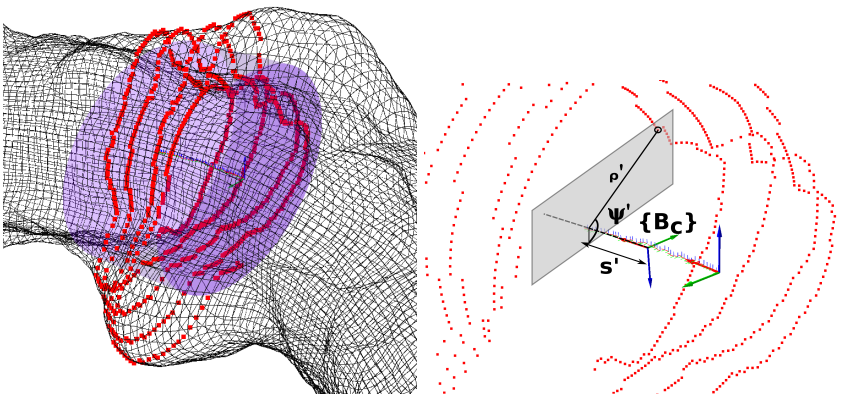
Figure 3: Illustration of the elevation angles estimation method in [4] on two successive scans. Red points corresponds to the vertical sonar measurements, the green (resp. blue) lines corresponds to samples from the estimated mean (resp. $\pm 3\sigma$ bounds) surface. Scaled-beta pdf color map for elevation angles goes from dark blue (low) to red (high). Turquoise corresponds to the density for a uniform distribution $\left(\frac{1}{b}\right)$. Best viewed in color.

This is illustrated in Figure 2a. This work aims at leveraging the information on the surface provided by the vertical sonar to estimate the elevation angles from the horizontal sonar measurements. This is done by estimating a stochastic model of the environment surface with a Gaussian process [26] trained on the vertical sonar measurements. Formally, the Gaussian process $\mathcal{GP}$ is defined on the vertical sonar output (range) ρ^v as a function of the sonar rotation angle ψ^v and the pose $\boldsymbol{\eta}$ (in practice, the curvilinear abscissa s) along the robot trajectory such that

$$\rho^v = f(s, \psi^v) + \epsilon, \;\; f \sim \mathcal{GP} \tag{61}$$

with a centered noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

Once the Gaussian process trained on the vertical sonar measurements, we sample each horizontal measurement beam into N angles. We estimate the likelihood of each angle as the corresponding 3D point likelihood to belong to the estimated surface. We also compute the associated uncertainty based on Fisher information. Thus, we obtain a scaled-beta distribution for the elevation angles



(a) Prior Cylinder

(b) Sonar data expressed with the prior cylinder axis (red axis of B_c) as the reference curve

Figure 4: Modification of original method in [4] using the prior cylinder axis as a reference curve instead of the robot trajectory

for every horizontal measurement beams. Figure 3a represents the estimated environment surface by sampling (for display purpose) the mean surface (green) and its lower/upper bounds (blue). Figure 3b represents the θ distributions of several horizontal measurements along their beam arc.

Two main modifications to [4] were made in this work and are detailed in the following sections.

5.2.1. Reference Curve

The Gaussian process in the original paper is trained on data given by the vertical sonar. The horizontal sonar data are thus expressed as if they were measured by the vertical sonar to allow Gaussian process regression. Doing so, we have to consider the robot poses along its trajectory. This can lead to non-smooth surface estimation as the angular position of the robot are not explicitly taken into account in the Gaussian process kernels. While this does not impact much the elevation angles estimation, it leads to wrong normals estimation required for the *point-to-plane* association scheme in our MpIC.

Instead of using the vertical sonar as a reference for the Gaussian process and for horizontal sonar estimation, we can use an independent curve contained

inside the environment for the scan length. Previously, sonar data were expressed by a robot pose $\boldsymbol{\eta}$, a sonar angle ψ and a range ρ. This is replaced by the point $\mathbf{p}$ on the curve such that the orthogonal plane to the curve at $\mathbf{p}$ contains the sonar measurement. ρ, ψ are then the polar coordinates of the sonar measurement in this orthogonal plane. As a result, both the vertical and horizontal measurements are expressed relatively to the reference curve.

In this work, we use the prior cylinder (see [4], section 4.2.1) axis as our reference curve. An example is provided in Figure 4a. Let note respectively $B_c = [\mathbf{x}_c\ \mathbf{y}_c\ \mathbf{z}_c] \in \mathcal{M}_{3,3}$, $\mathbf{o}_c \in \mathbb{R}^3$ the cylinder orthonormal basis and center. Here the first basis vector $\mathbf{x}_c$ corresponds to the cylinder axis.

If $\mathbf{p} \in \mathbb{R}^3$ is a point measured by the vertical or horizontal sonar, we then have the new abscissa $s' = \mathbf{x_c}^T(\mathbf{p} - \mathbf{o}_c)$ with range in the cylinder orthogonal plane $\rho' = ||\mathbf{y_c}^T(\mathbf{p}-\mathbf{o}_c)\mathbf{y}_c + \mathbf{z_c}^T(\mathbf{p}-\mathbf{o}_c)\mathbf{z}_c||$ and angle $\psi' = \arctan\left(\frac{\mathbf{z_c}^T(\mathbf{p}-\mathbf{o}_c)}{\mathbf{y_c}^T(\mathbf{p}-\mathbf{o}_c)}\right)$. This is illustrated in Figure 4b.

In the case where $\mathbf{p}$ was measured by the vertical sonar, (s', ρ', ψ') replace respectively (s, ρ^v, ψ^v) in (61). We now modify the derivation in [4] section 4.2.3 by replacing $\hat{s}, \hat{\rho}^v, \hat{\psi}^v$ with s', ρ', ψ' which, as a side effect, simplify the computation.

5.2.2. Trajectory interpolation

While in [4] we assumed that robot odometry and sonar measurements were synchronous, this is generally not the case in practice. This requires to interpolate the robot poses at each sonar measurement. We use a simple on-manifold linear interpolation scheme with constant velocity between two robot poses measurement. Let consider $\boldsymbol{\eta}_1 \sim \mathcal{N}(\bar{\boldsymbol{\eta}}_1, \Sigma_{\eta_1}), \boldsymbol{\eta}_2 \sim \mathcal{N}(\bar{\boldsymbol{\eta}}_2, \Sigma_{\eta_2})$ two successive robot poses and a geodesic γ such that $\gamma(0) = \boldsymbol{\eta}_1$ and $\gamma'(0) = \log(\boldsymbol{\eta}_2 \ominus \boldsymbol{\eta}_1)$. Thus $\gamma(t) = e^{t\log(\boldsymbol{\eta}_2 \ominus \boldsymbol{\eta}_1)}$ and, for $t \in [0, 1]$, the interpolated mean pose $\bar{\boldsymbol{\eta}}(t)$ between $\bar{\boldsymbol{\eta}}_1$ and $\bar{\boldsymbol{\eta}}_2$ is given by

$$\bar{\boldsymbol{\eta}}(t) = \bar{\boldsymbol{\eta}}_1 \oplus e^{t\log(\boldsymbol{\eta}_2 \ominus \boldsymbol{\eta}_1)} \tag{62}$$

We also need to interpolate the uncertainty between the poses. Recall that covariance matrices are definite positives. The space of definite positive matrices

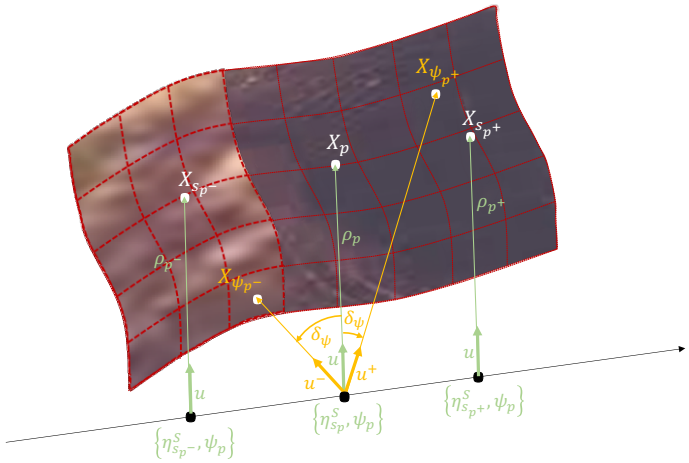


Figure 5: Notations related to normal computation. Note the abuse of notation here as $\mathbf{X}_p$, $\mathbf{X}_{s_p^\pm}$ and $\mathbf{X}_{\psi_p^\pm}$ are not "fixed" 3D points but denote 3D random variables.

$\mathbb{S}^n_{++}$ is a smooth Riemannian manifold. In particular, we are interested here in the geodesics based on the Fisher-Rao metric. As $\Sigma_{\eta_1}, \Sigma_{\eta_2}$ are two definite positives matrices, the geodesic $\Sigma_\eta(t)$ connecting Σ_{η_1} and Σ_{η_2} is given by [27]

$$\Sigma_\eta(t) = \Sigma_{\eta_1}^{\frac{1}{2}} \left(\Sigma_{\eta_1}^{-\frac{1}{2}} \Sigma_{\eta_2} \Sigma_{\eta_1}^{-\frac{1}{2}} \right)^t \Sigma_{\eta_1}^{\frac{1}{2}}, \ t \in [0,1] \tag{63}$$

Note that the non-integer power of a matrix can be computed based on the Schur-Padé algorithm [28].

5.3. Normal estimation

In this section, we derive a method for computing normals distribution associated to each data point based on the Gaussian process regression of the environment surface. The normals are required for the *point-to-plane* association as described in Section 4.1. We first propose the derivation for the approach in [4]. The derivation is similar when using a reference curve and in particular the prior cylinder axis as explained in Section 5.2.1.

As we do not have a closed-form expression of our surface in Cartesian coordinates, we approximate the normal at a point by sampling points on the surface around it and taking their cross product. The notations defined in the

following are illustrated in Figure 5. Let $\mathbf{X}_p$ be a point on the surface defined by

$$\mathbf{X}_p = h\left(\rho_p,\ 0,\ \psi_p,\ \boldsymbol{\eta}^S(s_p)\right) \tag{64}$$

$$\rho_p = f(s_p, \psi_p) \tag{65}$$

where $\boldsymbol{\eta}^S_{s_p} = \boldsymbol{\eta}^S(s_p)$ is the vertical sonar pose at the curvilinear abscissa s_p along the robot trajectory, h defined by (53) and f the Gaussian process in (61). We also define the following random variables

$$\begin{aligned}
\rho_{s_p^\pm} &= f\left(s_p \pm \delta s,\ \psi_p\right) + \epsilon \\
\rho_{\psi_p^\pm} &= f\left(s_p,\ \psi_p \pm \delta\psi\right) + \epsilon \\
\boldsymbol{\eta}^S_{s_p^\pm} &= \boldsymbol{\eta}^S\left(s_p \pm \delta s\right) \\
\mathbf{u}(\psi) &= \begin{bmatrix}\cos(\psi_p) & \sin(\psi_p) & 0\end{bmatrix}^T \\
\mathbf{u}_\pm &= \mathbf{u}(\psi_p \pm \delta\psi)
\end{aligned}$$

where $\delta s, \delta\psi$ are small increments. We note $R^S_\pm,\ \mathbf{t}^S_\pm$ the rotation matrices and translation vectors corresponding to the poses $\boldsymbol{\eta}^S_{s_p^\pm}$. Thus we have the sampled points around $\mathbf{X}_p$

$$\mathbf{X}_{s_p^\pm} = h\left(\rho_{s^\pm},\ 0,\ \psi_p,\ \boldsymbol{\eta}^S_{s_p^\pm}\right) \tag{66}$$

$$\mathbf{X}_{\psi_p^\pm} = h\left(\rho_{\psi^\pm},\ 0,\ \psi_p^\pm, \boldsymbol{\eta}^S_{s_p}\right) \tag{67}$$

The cross-product between the vectors $\mathbf{X}_{s_p^+} - \mathbf{X}_{s_p^-}$ and $\mathbf{X}_{\psi_p^+} - \mathbf{X}_{\psi_p^-}$ gives the unnormalized normal $\mathbf{v}(\mathbf{X}_p)$ at $\mathbf{X}_p$

$$\begin{aligned}
\mathbf{v}(\mathbf{X}_p) &= \left(\mathbf{X}_{s_p^+} - \mathbf{X}_{s_p^-}\right) \times \left(\mathbf{X}_{\psi_p^+} - \mathbf{X}_{\psi_p^-}\right) \\
&= \rho_{s_p^+}\rho_{\psi_p^+}a_0 + \rho_{s_p^-}\rho_{\psi_p^-}a_1 + \rho_{s_p^-}\rho_{\psi_p^+}a_2 \\
&\quad + \rho_{s_p^+}\rho_{\psi_p^-}a_3 + \rho_{\psi_p^+}b_0 - \rho_{\psi_p^-}b_1 \\
&= \mathbf{v}\left(\rho_{s_p^+}, \rho_{\psi_p^+}, \rho_{s_p^-}, \rho_{\psi_p^-}\right)
\end{aligned} \tag{68}$$

with

$$a_0 = R_+^S \mathbf{u} \times R^S \mathbf{u}_+ \qquad a_1 = R_-^S \mathbf{u} \times R^S \mathbf{u}_-$$
$$a_2 = -R_-^S \mathbf{u} \times R^S \mathbf{u}_+ \qquad a_3 = -R_+^S \mathbf{u} \times R^S \mathbf{u}_-$$
$$b_0 = \left(\mathbf{t}_+^S - \mathbf{t}_-^S\right) \times R\mathbf{u}_+ \qquad b_1 = \left(\mathbf{t}_+^S - \mathbf{t}_-^S\right) \times R\mathbf{u}_-$$

In order to compute the mean and covariance of $\mathbf{v}$, we need the covariance matrix Σ of the joint gaussian vector $\begin{bmatrix} \rho_{s_p^+} & \rho_{\psi_p^+} & \rho_{s_p^-} & \rho_{\psi_p^-} \end{bmatrix}^T$. It can be computed using the formula (2.24) provided in [26], p.16. In the following, for any two random variables X,Y, we denote their covariance $\sigma_{X,Y} = Cov(X,Y)$.

For any order 2 term in (68), we have $E\left(\rho_{s_p^\pm}\rho_{\psi_p^\pm}\right) = \sigma_{\rho_{s_p^\pm}\rho_{\psi_p^\pm}} + \bar{\rho}_{s_p^\pm}\bar{\rho}_{\psi_p^\pm}$ and thus

$$\begin{aligned}\bar{\mathbf{v}}\left(\mathbf{X_p}\right) = \mathbf{v}\left(\bar{\rho}_{s_p^+}, \bar{\rho}_{\psi_p^+}\bar{\rho}_{s_p^-}, \bar{\rho}_{\psi_p^-}\right) + a_0\sigma_{\rho_{s_p^+}\rho_{\psi_p^+}} \\ + a_1\sigma_{\rho_{s_p^-}\rho_{\psi_p^-}} + a_2\sigma_{\rho_{s_p^-}\rho_{\psi_p^+}} + a_3\sigma_{\rho_{s_p^+}\rho_{\psi_p^-}}\end{aligned} \tag{69}$$

The covariance Σ_v of $\mathbf{v}$ is obtained with the classic relation $\Sigma_v = E\left(\mathbf{v}\mathbf{v}^T\right) - \bar{\mathbf{v}}\bar{\mathbf{v}}^T$. We therefore need to compute the 4-order multivariate polynomial $\mathbf{v}\mathbf{v}^T$. It has the general expression

$$\mathbf{v}\mathbf{v}^T = \sum_{\substack{i,j,k,l=0 \\ i+j+k+l\leq 4}}^{4} \rho_{s_p^+}^i \rho_{\psi_p^+}^j \rho_{s_p^-}^k \rho_{\psi_p^-}^l A_{i,j,k,l} \tag{70}$$

where $A_{i,j,k,l}$ are 3×3 symmetric matrices defined by outer product between the a_i and b_j defined in (69). Their expressions are given in Appendix C. This yields to

$$\Sigma_v = \sum_{\substack{i,j,k,l=0 \\ i+j+k+l\leq 4}}^{4} E\left(\rho_{s_p^+}^i \rho_{\psi_p^+}^j \rho_{s_p^-}^k \rho_{\psi_p^-}^l\right) A_{i,j,k,l} - \bar{\mathbf{v}}\bar{\mathbf{v}}^T \tag{71}$$

The covariance terms of order 3 and 4 are computed thanks to the Isserlis theorem which, in case of a centered joint Gaussian RV Z_i, gives $E(Z_1Z_2Z_3) = 0$ and $E(Z_1Z_2Z_3Z_4) = \sigma_{12}\sigma_{34} + \sigma_{13}\sigma_{24} + \sigma_{14}\sigma_{23}$. We deduce then, for non-centered

variables Y_i

$$E(Y_1Y_2Y_3) = \sigma_{23}\bar{Y}_1 + \sigma_{13}\bar{Y}_2 + \sigma_{12}\bar{Y}_3 + \bar{Y}_1\bar{Y}_2\bar{Y}_3 \tag{72}$$

$$\begin{aligned} E(Y_1Y_2Y_3Y_4) = {} & \sigma_{12}\sigma_{34} + \sigma_{13}\sigma_{24} + \sigma_{14}\sigma_{23} + \sigma_{12}\bar{Y}_3\bar{Y}_4 \\ & + \sigma_{13}\bar{2}\bar{4} + \sigma_{14}\bar{Y}_2\bar{Y}_3 + \sigma_{23}\bar{Y}_1\bar{Y}_4 \\ & + \sigma_{24}\bar{Y}_1\bar{Y}_3 + \sigma_{34}\bar{Y}_1\bar{Y}_2 + \bar{Y}_1\bar{Y}_2\bar{Y}_3\bar{Y}_4 \end{aligned} \tag{73}$$

Finally, we obtain the Gaussian approximation pdf of the normalized normal by noting f_n the normalization operator such that

$$f_n(\mathbf{v}) = \frac{\mathbf{v}}{||\mathbf{v}||} \sim \mathcal{N}\left(f_n(\bar{\mathbf{v}}), J^{f_n}_{|\bar{\mathbf{v}}} \Sigma_v J^{f_n^T}_{|\bar{\mathbf{v}}}\right) \tag{74}$$

$$J^{f_n}(\mathbf{v}) = \frac{1}{||\mathbf{v}||^3} \begin{bmatrix} ||\mathbf{v}||^2 - v_x^2 & -v_x v_y & -v_x v_z \\ -v_x v_y & ||\mathbf{v}||^2 - v_y^2 & -v_y v_z \\ -v_x v_z & -v_y v_z & ||\mathbf{v}||^2 - v_z^2 \end{bmatrix} \tag{75}$$

Figure 6 shows an example of normals and their associated incertitude cone estimated as explained in this section.

5.3.1. Application to point-to-plane association

We explain here how *point-to-plane* association, explained in Section 4.1, is performed in our practical application. For the sake of clarity, we omit the k iteration superscript. Consider a point $\mathbf{n}_i$ in the current new point cloud S_{new} and its corresponding set of candidates A_i. As in the *point-to-point* association scheme, we consider the best candidate $\mathbf{a}_i$ as in (31). The tangent plane $\mathbf{p}_i$ to the environment surface at $\mathbf{a}_i$ with normal $\mathbf{v}_i \equiv \mathbf{v}_i(\mathbf{a}_i)$ is obtained with the computations in Section 5.3.

Note that in our case, the general expression of $\mathbf{a}_i^\perp$ (orthogonal projection of $\mathbf{n}_i$ on $\mathbf{p}_i$) can be rewritten from (32) as follows

$$\bar{\boldsymbol{a}}_i^\perp = \bar{\boldsymbol{n}}_i - \bar{\boldsymbol{v}}_i^T \left(\bar{\boldsymbol{n}}_i - \bar{\boldsymbol{a}}_i\right) \bar{\boldsymbol{v}}_i \tag{76}$$

by using the fact that $\mathbf{a}_i \in \mathbf{p}_i \iff d_i = \mathbf{v}_i^T \mathbf{a}_i$. By consequence, the computation of $\Sigma_{a_i^\perp}$ in our case is slightly different from (33) :

$$\Sigma_{\mathbf{a}_i^\perp} = J^{\mathbf{a}_i^\perp}_{\mathbf{n}_i} \Sigma_{\mathbf{n}_i} J^{\mathbf{a}_i^{\perp T}}_{\mathbf{n}_i} + J^{\mathbf{a}_i^\perp}_{\mathbf{v}_i} \Sigma_{\mathbf{v}_i} J^{\mathbf{a}_i^{\perp T}}_{\mathbf{v}_i} \tag{77}$$

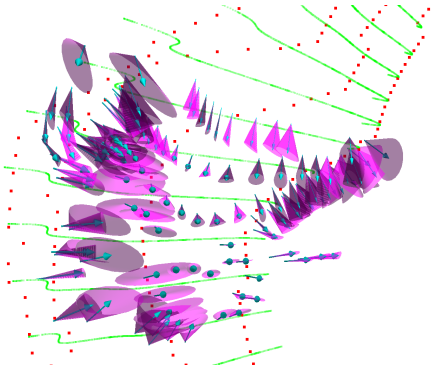


Figure 6: Estimated mean normals (turquoise arrow) and their 1-σ uncertainty cone (purple)

The jacobians in (77) are given by the following expressions :

$$J_{\mathbf{n}_i}^{\mathbf{a}_i^\perp} = I_3 - \bar{\boldsymbol{v}}_i \bar{\boldsymbol{v}}_i^T \tag{78}$$

$$J_{\mathbf{v}_i}^{\mathbf{a}_i^\perp} = \left(\bar{\boldsymbol{v}}_i \left(\bar{\boldsymbol{n}}_i - \bar{\boldsymbol{a}}_i \right)^T + \bar{\boldsymbol{v}}_i^T \left(\bar{\boldsymbol{n}}_i - \bar{\boldsymbol{a}}_i \right) I_3 \right) J^{f_n}(\bar{\boldsymbol{v}}_i) \tag{79}$$

5.4. Gaussian approximation

In order to apply a pIC algorithm to the point clouds obtained from the horizontal sonar, we must have data points following Gaussian distributions as seen in Section 4.

Let $\mathbf{p}^L \in \mathbb{R}^3$ a measured point expressed in the horizontal sonar local frame as defined by (52)(56)(57)(59). Clearly, $\mathbf{p}^L$ is not following a normal distribution.

To compute its Gaussian approximation, we have to estimate its mean $\bar{\boldsymbol{p}}^L$ and covariance matrix Σ_{p^L}. From (52) we have directly

$$\bar{\boldsymbol{p}}^L = \bar{\rho} \begin{bmatrix} E(\cos\phi)E(\cos\theta) \\ E(\sin\phi)E(\cos\theta) \\ E(\sin\theta) \end{bmatrix} \tag{80}$$

The covariance matrix is given by the classic relation

$$\begin{aligned}\Sigma_{p^L} &= E(\mathbf{p}^L \mathbf{p}^{L^T}) - \bar{\boldsymbol{p}}^L \bar{\boldsymbol{p}}^{L^T} \\ &= \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{bmatrix} \end{aligned} \tag{81}$$

with

$$\sigma_x^2 = E(\rho^2)E(\cos^2\phi)E(\cos^2\theta) - \bar{\rho}^2(E\cos\phi)^2(E\cos\theta)^2 \tag{82}$$

$$\sigma_y^2 = E(\rho^2)E(\sin^2\phi)E(\cos^2\theta) - \bar{\rho}^2(E\sin\phi)^2(E\cos\theta)^2 \tag{83}$$

$$\sigma_z^2 = E(\rho^2)E(\sin^2\theta) - \bar{\rho}^2(E\sin\theta)^2 \tag{84}$$

$$\sigma_{xy} = E(\rho^2)E(\cos\phi\sin\phi)E(\cos^2\theta) - \bar{\rho}^2 E(\cos\phi)E(\sin\phi)(E\cos\theta)^2 \tag{85}$$

$$\sigma_{xz} = E(\rho^2)E(\cos\phi)E(\cos\theta\sin\theta) - \bar{\rho}^2 E(\cos\phi)E(\cos\theta)E(\sin\theta) \tag{86}$$

$$\sigma_{yz} = E(\rho^2)E(\sin\phi)E(\cos\theta\sin\theta) - \bar{\rho}^2 E(\sin\phi)E(\cos\theta)E(\sin\theta) \tag{87}$$

$$E(\rho^2) = \bar{\rho}^2 + \sigma_\rho^2 \tag{88}$$

The following proposition gives the expressions of the different involved expectations over trigonometric functions

Proposition 3. *Let θ a random value following a shifted and scale Beta distribution $SBeta(\alpha, \beta, -\frac{b}{2}, \frac{b}{2})$. Then*

$$E(\cos\theta) = \sum_{n=0}^{+\infty} \frac{(-1)^n}{2n!} \left(\frac{b}{2}\right)^{2n} {}_2F_1(-2n, \alpha, \alpha+\beta; 2) \tag{89}$$

$$E(\cos^2\theta) = 1 + \frac{1}{2}\sum_{n=1}^{+\infty} \frac{(-1)^n}{2n!} b^{2n} {}_2F_1(-2n, \alpha, \alpha+\beta; 2) \tag{90}$$

$$E(\sin\theta) = \frac{-b}{2}\sum_{n=0}^{+\infty} \frac{(-1)^n}{2n+1!} \left(\frac{b}{2}\right)^{2n} {}_2F_1(-2n-1, \alpha, \alpha+\beta; 2) \tag{91}$$

$$E(\cos\theta\sin\theta) = \frac{-b}{2}\sum_{n=0}^{+\infty} \frac{(-1)^n}{2n+1!} b^{2n} {}_2F_1(-2n-1, \alpha, \alpha+\beta; 2) \tag{92}$$

where ${}_2F_1$ is the hypergeometric function.

The proof is given in Appendix B.3. Note that $E(\cos\theta\sin\theta), E(\sin^2\theta)$ are deduced from Proposition 3 using the relations $\cos\theta\sin\theta = \frac{\sin 2\theta}{2}$ and $\sin^2\theta = 1 - \cos^2\theta$.

N	$E(\cos\theta)$	$E(\sin\theta)$	$E(\cos^2\theta)$	$E(\sin\theta)$
2	$4.23e^{-2}$	$5.60e^{-3}$	$7.47e^{-1}$	$2.59e^{-1}$
3	$1.18e^{-3}$	$1.11e^{-4}$	$8.33e^{-2}$	$2.07e^{-2}$
4	$1.76e^{-5}$	$1.29e^{-6}$	$4.98e^{-3}$	$9.63e^{-4}$
5	$1.64e^{-7}$	$9.88e^{-9}$	$1.85e^{-4}$	$2.39e^{-5}$

Table 1: Error bounds on expectation in Proposition 3

For practical computation, we have to approximate the previous series by the sum of the first N terms. To estimate a good value of N with low approximation error, we compute bounds on the series reminders which are given in the following proposition.

Proposition 4. *Let $N > 0$ and $\theta \in \left[-\frac{b}{2}, \frac{b}{2}\right]$. The reminders of order N for the series in Proposition 3 are given by*

$$|R_N\left(E(\cos\theta)\right)| \leq \frac{1}{(2N)!}\left(\frac{3b}{2}\right)^{2N}\cosh\left(\frac{3b}{2}\right) \tag{93}$$

$$|R_N\left(E(\sin\theta)\right)| \leq \frac{1}{(2N+1)!}\left(\frac{3b}{2}\right)^{2N+1}\sinh\left(\frac{3b}{2}\right) \tag{94}$$

$$|R_N\left(E(\cos^2\theta)\right)| \leq \frac{1}{2}\frac{1}{(2N)!}\left(3b\right)^{2N}\cosh\left(3b\right) \tag{95}$$

$$|R_N\left(E(\sin\theta)\right)| \leq \frac{1}{2}\frac{1}{(2N+1)!}\left(3b\right)^{2N+1}\sinh\left(3b\right) \tag{96}$$

The proof is given in Appendix B.4. The corresponding bounds for small values of N in our application where $b = 0.61$ $(35°)$ are given in Table 1. In our following experiments, we use $N = 5$.

Proposition 5. *Let $\phi \sim \mathcal{N}\left(\bar{\phi}, \sigma_\phi^2\right)$. Then*

$$E(\cos\phi) = \cos\bar{\phi}e^{-\frac{\sigma_\phi^2}{2}} \tag{97}$$

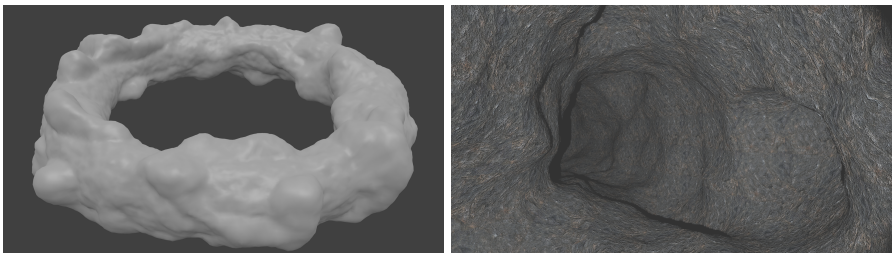
$$E(\sin\phi) = \sin\bar{\phi}e^{-\frac{\sigma_\phi^2}{2}} \tag{98}$$

$$E(\cos^2\phi) = \frac{1}{2}\left(1 + e^{-2\sigma_\phi^2}\cos 2\bar{\phi}\right) \tag{99}$$

$$E(\sin^2\phi) = \frac{1}{2}\left(1 - e^{-2\sigma_\phi^2}\cos 2\bar{\phi}\right) \tag{100}$$

$$E(\cos\phi\sin\phi) = \frac{1}{2}e^{-2\sigma_\phi^2}\sin 2\phi \tag{101}$$

The result is obtained by the exact computation of $e^{i\phi}$ and decomposing ϕ as $\phi = \sigma_\phi Z + \bar{\phi}$ with $Z \sim \mathcal{N}(0,1)$. Finally, the probability density function



(a) External view (b) Inside view

Figure 7: Karst model used for our experiments

(pdf) of the point $\mathbf{p} = \boldsymbol{\eta}_{\mathbf{p}} \oplus \mathbf{p}^L$ expressed in the current scan reference frame is given by

$$\mathbf{p} \sim \mathcal{N}\left(\bar{\boldsymbol{\eta}}_{\mathbf{p}} \oplus \bar{\boldsymbol{p}}^L, J^{\oplus}_{\mathbf{q}_{|\bar{\boldsymbol{\eta}}_{\mathbf{p}}, \bar{\boldsymbol{p}}^L}} \Sigma_{p^L} J^{\oplus^T}_{\mathbf{q}_{|\bar{\boldsymbol{\eta}}_{\mathbf{p}}, \bar{\boldsymbol{p}}^L}}\right) \tag{102}$$

where $\boldsymbol{\eta}_{\mathbf{p}}$ is the pose at which the point has been observed.

6. Experiments

In this section, we assess the performance of the proposed algorithms with several experiments. First, in order to validate our computation, we simulate point clouds following probability distributions (55)(56)(57) with known associations and apply our MpIC algorithm.

Due to the lack of available ground truth with real data, we have to rely on simulation for quantitative analysis. In section 6.2, we generate data point cloud from measurements along a trajectory inside the simulated karst environment shown in Figure 7. The scan matching results are quantitatively compared to ground truth and state-of-the-art pIC approach proposed in [2][7].

We then demonstrate the use of our method on a real data set provided by [7] corresponding to the underwater cave "Coves de Cala Viuda" located in the Escala area of Costa Brava (Spain). As no ground truth is provided, we locally consider the dead-reckoning (DR) trajectory as a pseudo ground truth. We qualitatively assess our approach by comparing DR and our estimated trajectories. The MpIC Python source code used in this paper is available at `https://gite.lirmm.fr/breux/mpic`.

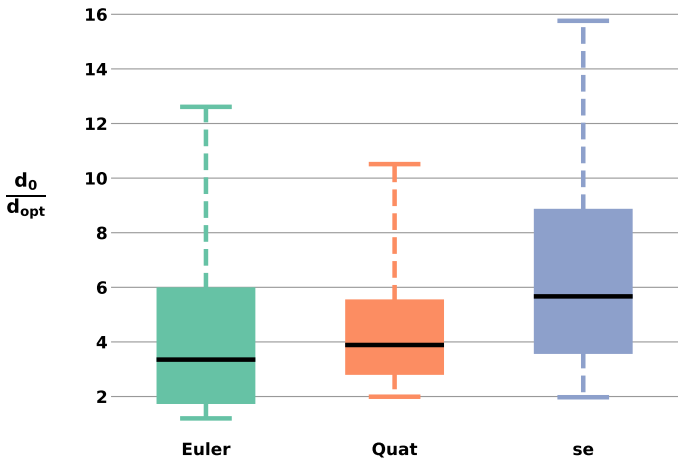


Figure 8: Comparison of distances ratio for 500 trials. Medians are indicated by black lines. The bottom/top whiskers correspond to 5%/95% of the distributions.

6.1. Evaluation of the optimization step in pIC

In this section, we assess the improvement of MpIC compared to previous methods. At each trial, we randomly generate 100 3D points distribution $\mathbf{c}_i$ and a random normal transform $\mathbf{q} \sim \mathcal{N}(\bar{\boldsymbol{q}}, \Sigma_q)$. From each distribution $\mathbf{c}_i$ we sample one point $\hat{\mathbf{c}}_i$ considered as the ground truth point. Similarly, we sample the ground truth transformation $\hat{\mathbf{q}}$ from $\mathbf{q}$. The ground truth points of the second point cloud are computed as $\hat{\mathbf{a}}_i = \hat{\mathbf{q}} \oplus \hat{\mathbf{c}}_i$. Finally, the distributions $\mathbf{a}_i$ are generated with random covariance Σ_{a_i} and the means $\bar{\boldsymbol{a}}_i$ as samples from the distributions $\mathcal{N}(\hat{\mathbf{a}}_i, \Sigma_{a_i})$. The initial transformation is given by $\mathbf{q}^{(0)} \sim \mathcal{N}(\bar{\boldsymbol{q}}^{(0)} = \bar{\boldsymbol{q}}, \Sigma_q)$.

We consider here only the optimization step of pIC with perfectly associated points. We compare the optimized transformation $\mathbf{q}_{opt}$ obtained using Euler angles, quaternion and the proposed on-manifold optimization. The distance of the estimated pose $\mathbf{q}_{opt}$ to the ground truth $\hat{\mathbf{q}}$ is computed using (2) and the metric on $SE(3)$ defined by (6) such that

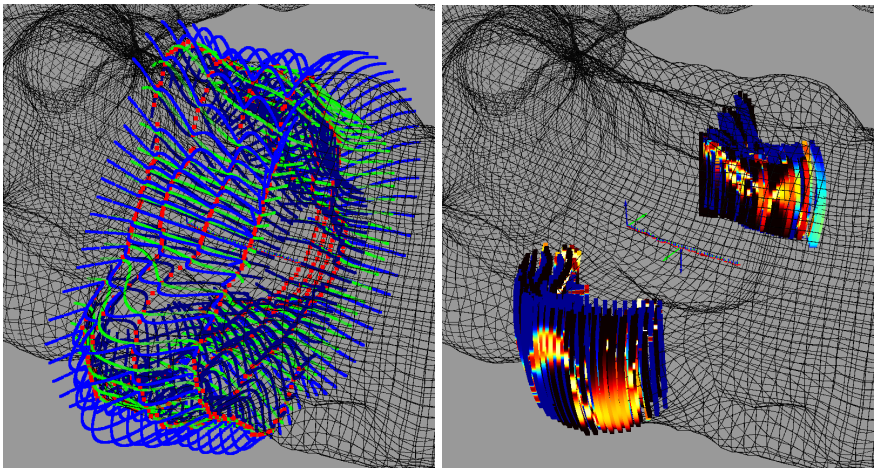
$$\begin{aligned} d_{opt} &= d(\mathbf{q}_{opt}, \hat{\mathbf{q}}) \\ &= \sqrt{log\left(\mathbf{q}_{opt} \ominus \hat{\mathbf{q}}\right)^T G \; log\left(\mathbf{q}_{opt} \ominus \hat{\mathbf{q}}\right)} \end{aligned} \tag{103}$$

Parameter	Value
Robot sensing	
Depth sensor std	0.016 m
IMU orientation std	0.16°
Dead-reckoning noise std on x,y	0.022 m
Dead-reckoning noise std on yaw	0.13°
MSIS	
Vertical sonar range resolution	0.2 m
Horizontal sonar range resolution	0.05 m
Horizontal sonar beam width	35°
Sonar Angular step	1.8°
Simulation	
Steps for a full sonar scan	200
Vertical sonar period step	1
Horizontal sonar period step	2
Dead-reckoning odometry period step	10
Gaussian process	
Length scale l_s	Fixed (1)
Length scale l_ψ	Fixed (0.25)
Signal std σ_f	Fixed (0.64)
Noise std σ_n	Learned

Table 2: Parameters used in the simulation experiments

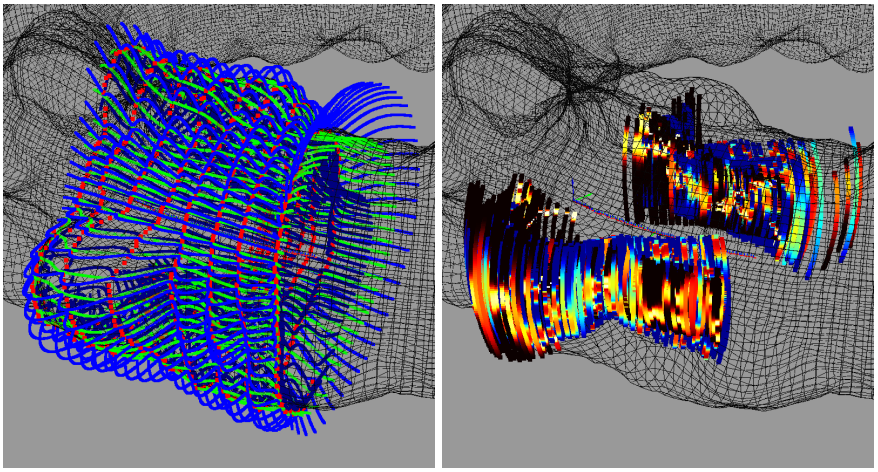
In order to compare the different trials, we consider the ratio d_0/d_{opt} with the initial distance $d_0 = d(\bar{\boldsymbol{q}}^{(0)}, \hat{\mathbf{q}})$. This means that the higher the ratio is, the better $\mathbf{q}_{opt}$ improves over the initial transformation $\mathbf{q}^{(0)}$.

In Figure 8, we represent the distributions of ratio for 500 trials and maximum of iterations set to 100. As expected, the Euler representation is more prone to failure depending on the trial configuration. The quaternion representation is more limited by its convergence rate due to the re-normalization at each iteration step. The on-manifold approach provides the best results expectation-wise.



(a) Estimated surface from a pair of 360° scan

(b) Elevation angles distributions for a 360° scans



(c) Estimated surface from a pair of 720° scan

(d) Elevation angles distributions for a 720° scan

Figure 9: Estimated surface and elevation angles estimation used for the first scan matching in our simulated environment. Red points corresponds to the vertical sonar measurements, the green (resp. blue) lines corresponds to samples from the estimated mean (resp. $\pm 3\sigma$ bounds) surface. Scaled-beta pdf color map for elevation angles goes from dark blue (low) to red (high) for the first scan and from black(low) to Yellow(high) for the second one. Best viewed in color.

6.2. Quantitative evaluation on simulated karst environment

It is difficult to obtain ground truth data in field experiments for underwater karst exploration. At best, real field experiments can be qualitatively analyzed

to assess the pertinence of an approach. In a large part of the literature on underwater robotics, systems are globally evaluated but there are few "test-unit" of the algorithms composing it. For instance, in [2][7][6], the proposed adaption of the pIC algorithm is indirectly evaluated through its integration in a SLAM framework.

[19] first evaluates its pIC approach by matching real scans with them-selves and adding a Gaussian noise to the initial transformation estimate. While based on data from real experiments, it does not take into account for the surface sampling error, e.g. associated points from each scan do not correspond exactly to the same point on the environment surface.

We propose here to evaluate quantitatively MpIC independently from any SLAM framework. We also compare our approach to re-implementation of the 2D approach proposed [2] which has been successfully applied in real underwater caves mapping [7].

We consider for this a simulated karst environment as shown in Figures 7a, 7b. The ground truth trajectory is sampled from a circle contained inside the karst model. At each time step, we generate a noisy odometry by adding Gaussian noise to the odometry obtained from the ground truth trajectory. The different sensors are simulated based on real characteristics provided in [3][7]. The different parameters are resumed in Table 2.

In real experiments, dead-reckoning (Kalman filter fusion of IMU and DVL sensors) and sonars data are received asynchronously with an attached timestamp. Here, timestamps are replaced by the simulation step. We consider the vertical sonar as our reference so that each step corresponds to a vertical sonar measurement. The relative frequency of the other data are based on the real data set provided by [7]. This can be seen by comparing simulated data in Figure 9 to the real data set in Figure 15. We can observe that we have between 2 and 3 full vertical sonar scans for one horizontal scan. Furthermore, odometry data are received every 10 vertical sonar measurements. Thus, we consider an horizontal sonar measurement every 2 steps and one odometry data every 10 steps. Note that we only consider point clouds from the horizontal sonar for the

pIC. As the angular step of both sonar is 1.8°, a full vertical (resp. horizontal) scan takes 200 (resp. 400) simulations steps. Thus scan matching with pIC algorithms is executed every 400 steps excepted for the first time (800 steps to generate the two first scans).

The vertical and horizontal MSIS measurements are simulated by ray-tracing. We then perform quantization on the obtained ranges to reproduce the original sonar resolution based on the maximum range and the number of intensity bins.

To obtain the robot pose at each sonar measurement, we interpolate the poses obtained by dead-reckoning as explained in Section 5.2.2.

The parameters used for the Gaussian process are also shown in Table 2. Following [4], we use a product covariance kernel $K = K_s * K_\psi$ where K_s is a Matern52 kernel on the abscissa s and K_ψ a Matern52 kernel based on angle chordal distance such that

$$K_s(s,s') = \sigma_f \left(1 + \frac{\sqrt{5}}{|s-s'|} l_s + \frac{5|s-s'|^2}{3l_s^2}\right) exp\left(-\frac{\sqrt{5}|s-s'|}{l_s}\right) \tag{104}$$

$$K_\psi(\psi,\psi') = \sigma_f \left(1 + \frac{\sqrt{5}}{2\sin\frac{|\psi-\psi'|}{2}} l_\psi + \frac{20\sin\frac{|\psi-\psi'|}{2}^2}{3l_\psi^2}\right) exp\left(-\frac{2\sqrt{5}\sin\frac{|\psi-\psi'|}{2}}{l_\psi}\right) \tag{105}$$

Recall that σ_n^2 in Table 2 is the variance of the Gaussian additive noise defined in (61). While all the values related to the Gaussian process can be learned, we fix them except σ_n^2 to ensure a smooth surface estimation.

As in Section 6.1, we consider the ratio $\frac{d_0}{d_{opt}}$ and compare the distributions obtained with different approaches :

- **2DpIC** : re-implementation of [2][3]. Note that the original papers only propose a *point-to-point* association scheme.
- **MpIC** : the proposed approach with *point-to-point* association.
- **MpIC$_\mathbf{a}$** : MpIC including measurements with a uniform distribution for the elevation angle. Uniform distributions arise mainly from measures outside the estimated surface covered by the vertical sonar measurements.

- **MpIC$_\mathbf{p}$** : MpIC with *point-to-plane* association as explained in Section 5.3. Note that the initial transformation here is the optimized transformation obtained by the MpIC with *point-to-point* association.

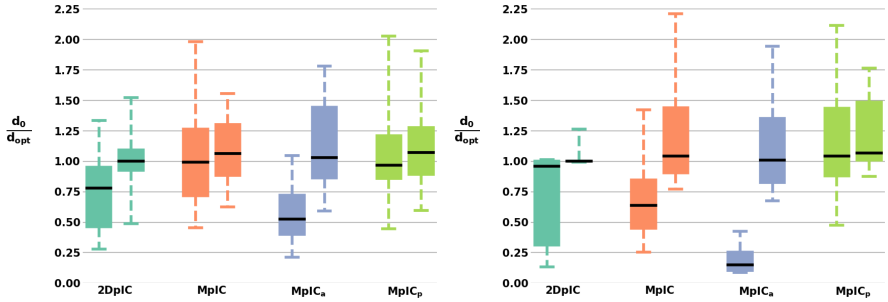
While this normalized ratio allows a global evaluation of the results, it has some drawbacks. Obviously a ratio of 1 has not the same meaning when d_0 is small or large : it is expected in the former and can be considered as a failure in the latter. Thus a more fine-grained representation in the form of scatter plot is also proposed.

In the original paper of 2DpIC [2], each point cloud is built from a full 360° horizontal sonar scan. For MpIC and MpICp, we are limited to horizontal sonar points measured from the surface delimited by the vertical sonar measurements. This leads to sparser 3D point clouds. MpICa, which also considers points outside the estimated surface, is an attempt to solve this potential drawback. Another solution is to consider larger scans. Here, we also propose to build point cloud based on 720° scan (two full horizontal sonar scans). We are interested to see if improving the point cloud density this way overcomes the increasing robot pose uncertainty accumulated by the dead-reckoning odometry.

Note that for 2DpIC, as in the original papers, we obtain the final 3D transformation from the 2D transformation (x, y, yaw) using directly the absolute values given by the depth sensor (z) and the IMU (pitch and roll). As can be seen in Table 2, those values are relatively accurate. For fair comparison, we similarly fix the z, pitch and roll values of the 3D transformation obtained with MpIC, MpIC$_\mathrm{a}$ and MpIC$_\mathrm{p}$.

In order to assess the effect of the association threshold defined in (30), we also compare the results obtained with two confidence levels $\alpha = 0.95$ and $\alpha = 0.5$. The higher the confidence level, the higher the threshold $\chi^2_{2,\alpha}$ or $\chi^2_{3,\alpha}$. In Figures 10 and 12, left (resp. right) boxes corresponds to $\alpha = 0.95$ (resp. $\alpha = 0.5$).

We consider two classes of scan matching : scan matching of successive scans and loop closure. For the first one, we estimate the distance d_{opt} for each



(a) Matching between 360° scans

(b) Matching between 720° scans

Figure 10: Diagrams of relative errors for successive scans. **2DpIC** : pIC proposed by [2], **MpIC**: MpIC (point-to-point), **MpIC$_a$** : MpIC (point-to-point) including uniform arcs, **MpIC$_p$** : MpIC (point-to-plane). Left (resp. right) boxes in each group have been computed with association confidence level $\alpha = 0.95$ (resp. $\alpha = 0.5$)

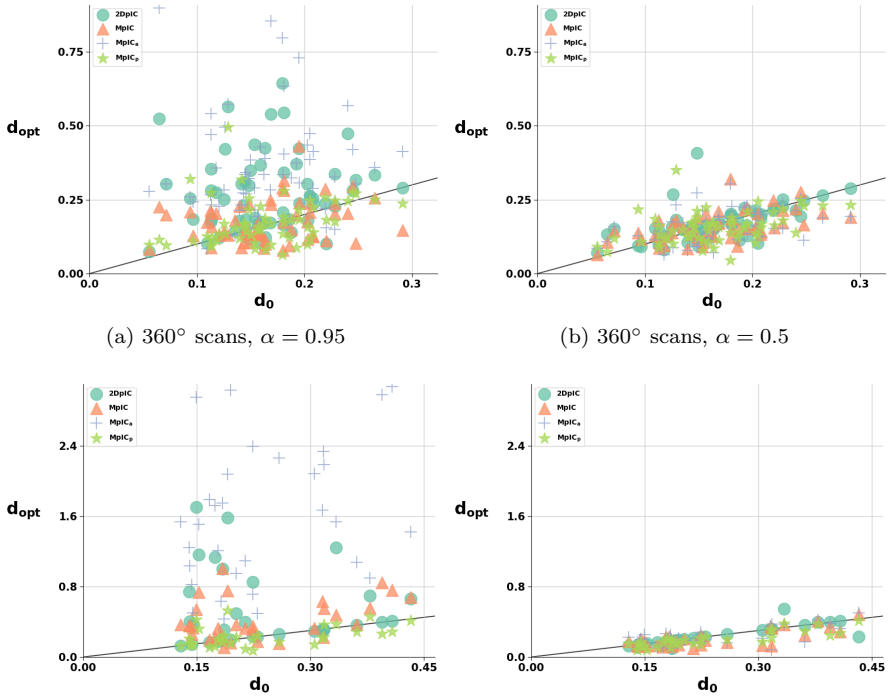
successive scan matching along the trajectory inside the karst model (61 scan matching). For the loop closure case, we consider a pair of scans and set higher diagonal values to the initial transformation covariance Σ_{q_0}. We sample the initial transformation $\mathbf{q}_0 \sim \mathcal{N}(\bar{\mathbf{q}}_0, \Sigma_{q_0})$ and apply the different pIC algorithms. Thus we obtain higher initial errors in the order of magnitude encountered in case of loop closure scan matching. We repeat this $n = 50$ times. In the following, we analyze and discuss the results obtained.

6.2.1. Successive scans

Figures 10 and 11 sum up the results obtained for the scan matching of successive scans.

The resulting distributions show that MpIC with *point-to-point* association improves over the state-of-the-art 2DpIC approach. While the median is slightly higher, it exhibits lower variance and higher minimum ratios. Note that MpIC only uses points on the surface estimated with the vertical sonar. Points seen behind and in front of the robot are thus ignored. This means that the matched point cloud are expected to be smaller and less discriminative. Nevertheless, the recovery of the 3D information overcomes this drawback.

In order to also take into account measurements outside the estimated surface, we include such horizontal sonar measurements by considering a uniform el-



(a) 360° scans, $\alpha = 0.95$

(b) 360° scans, $\alpha = 0.5$

(c) 720° scans, $\alpha = 0.95$

(d) 720° scans, $\alpha = 0.5$

Figure 11: Scatter plots for successive scans with different association confidence level. The black line corresponds to $x = y$ (unitary ratio) so that data points under this line corresponds to improved estimated pose over dead-reckoning.

evation angle distribution along the beam width. This approach, called MpIC_a, gives results which are a mixed of the 2DpIC and MpIC results. It yields an accuracy improvement from the 3D information but at the expense of higher variance as in the 2D case.

While *point-to-plane* ICP/pIC provides generally better performance than its *point-to-point* version, MpIC_p seems to be only slightly better. For successive scans, the 3D transformation uncertainty is relatively low. Indeed, the transformation uncertainty is the accumulation of low odometry uncertainties acquired along the robot trajectory during the scan acquisition. Informally, this means that the initial transformation uncertainty (represented by Σ_q) is, to a large extend, included in the interval of confidence obtainable with the pIC al-

gorithm. In other words, pIC would provide only little improvements over the initial transformation hence the low ratios in Figure 10. This is not the case when scan matching is done in a Loop closure as can be seen in Section 6.2.2. The initial transformation uncertainty has been accumulated along the trajectory covering several scans and is relatively larger than the local scan odometry uncertainties.

In regards to association confidence level, it is clear that $\alpha = 0.95$ gives relatively bad results regardless of the method used. It is intuitive as this confidence level corresponds to a higher association threshold and thus can potentially match points with lower likelihood. In the case of successive scans, we can expect the initial transformation to be near the local optimum so that matching points are initially near from each other. Low likelihood points should then be discarded to avoid spurious associations.

Finally, we now compare the results obtained using 360° and 720° scans. The difference observed for the two values of association confidence level are accentuated with larger scans, excepted for MpICp. In the case of $\alpha = 0.5$, we observe an higher lower bound of distance ratio regardless of the method used. Note that for 2DpIC, as can clearly be seen in Figure 11d, the algorithm is almost always stuck to a local minimum at the initial transformation. Globally both scan sizes give similar results. The optimal size for a given area depends obviously on the environment structure or lack thereof. Typically, corridor-like environments may require larger scan size to capture more structure.

6.2.2. Loop Closure

Similarly to the previous section, Figures 12 and 11 sum up the results with large initial transformation error. Regardless of the method, improvements over the initial transformation obtained with the dead-reckoning odometries are more obvious than in the previous case of small transformations. This confirms our previous hypothesis on the fact that for successive scans, the initial error uncertainty is almost contained in the algorithm output uncertainty. In case of larger initial error, the *point-to-plane* MpIC (MpIC_p) gives more accurate

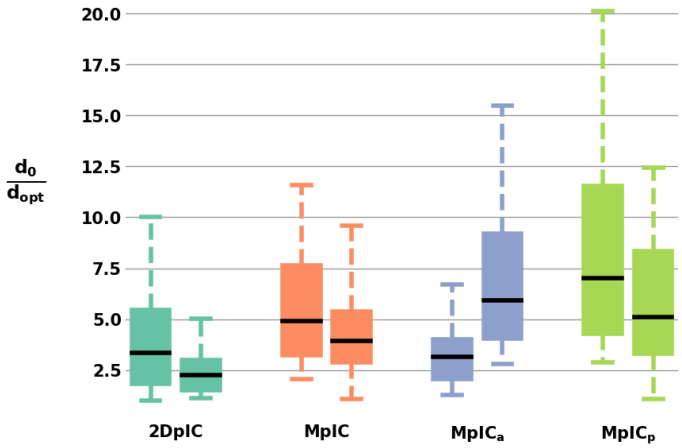
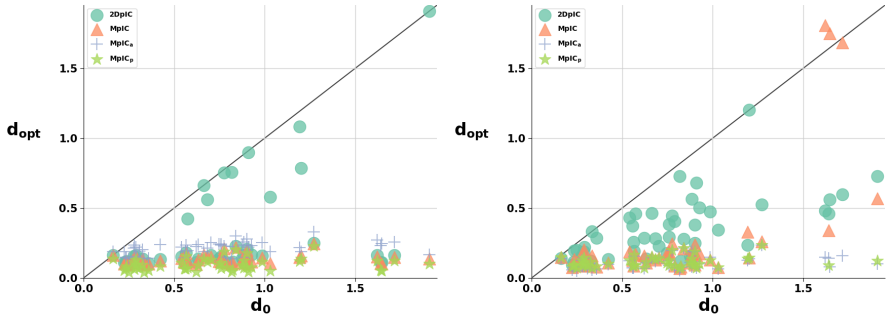


Figure 12: Diagrams of relative errors for loop closure with 360° scans. **2DpIC** : pIC proposed by [2], **MpIC**: MpIC (point-To-point), **MpIC$_a$** : MpIC(point-to-point) including uniform arcs, **MpIC$_p$** : MpIC (point-to-plane). Left (resp. right) boxes in each group have been computed with association confidence level $\alpha = 0.95$ (resp. $\alpha = 0.5$)



(a) $\alpha = 0.95$ (b) $\alpha = 0.5$

Figure 13: Scatter plots for loop closure with different association confidence level (360° scans). The black line corresponds to $x = y$ (unitary ratio) so that data points under this line corresponds to improved estimated pose over dead-reckoning. Note the natural difference of scales compared to the successive scans case (Figure 11).

results compared to the *point-to-point* MpIC.

Unlike the previous case, lower confidence level for association provides better associations for large initial transformation. This is intuitive as in the first iterations, corresponding points are rather far between leading to higher Mahalanobis distances (lower likelihood). We then need a lower association thresholds to obtain a sufficient number of correspondences.

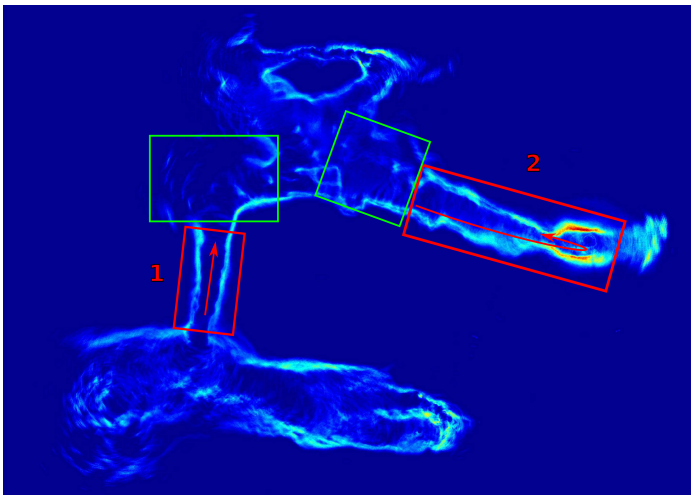


Figure 14: Segments (red rectangles) used to test our approach on the real experimental data provided by [7]. The segments start respectively at timestamp 1372687311775722265 and 1372687577375451326. Green rectangles show areas where horizontal sonar elevation angles is difficult due to the fact that only one side of the karst is visible by the vertical sonar. Extracted and modified from [3].

6.3. Experiment on Real Terrain data

In order to validate our approach on real data, we exploit the data set provided by the University of Girona [3] [7]. It was acquired with an early version of the Sparus AUV [29] and contains measurements from a DVL, two Inertial Measurement Unit (IMU) and two MSIS. The dead-reckoning trajectory obtained by an extended Kalman Filter (EKF) fusing DVL and IMU measurements is also provided. The MSIS are configured similarly to Figure 2a. The horizontal (resp. vertical) sonar has a beamwidth of 35° vertical by 3° horizontal (resp. 2° by 1°) and configured to scan the 360° sector at 20 m range (resp. 10 m) with a 0.05 m (resp. 0.2 m) resolution at 1.8° angular step.

The data set also includes six traffic cones as ground truth points which are detected with an analog video camera. They were placed in locations where the vehicle passed twice (loop closure) in order to evaluate the performance of a SLAM algorithm as done in [3]. In our case, we are interested in evaluating our scan matching which will require a lot more ground truth points.

To qualitatively assess our approach, we relatively compare the dead-reckoning

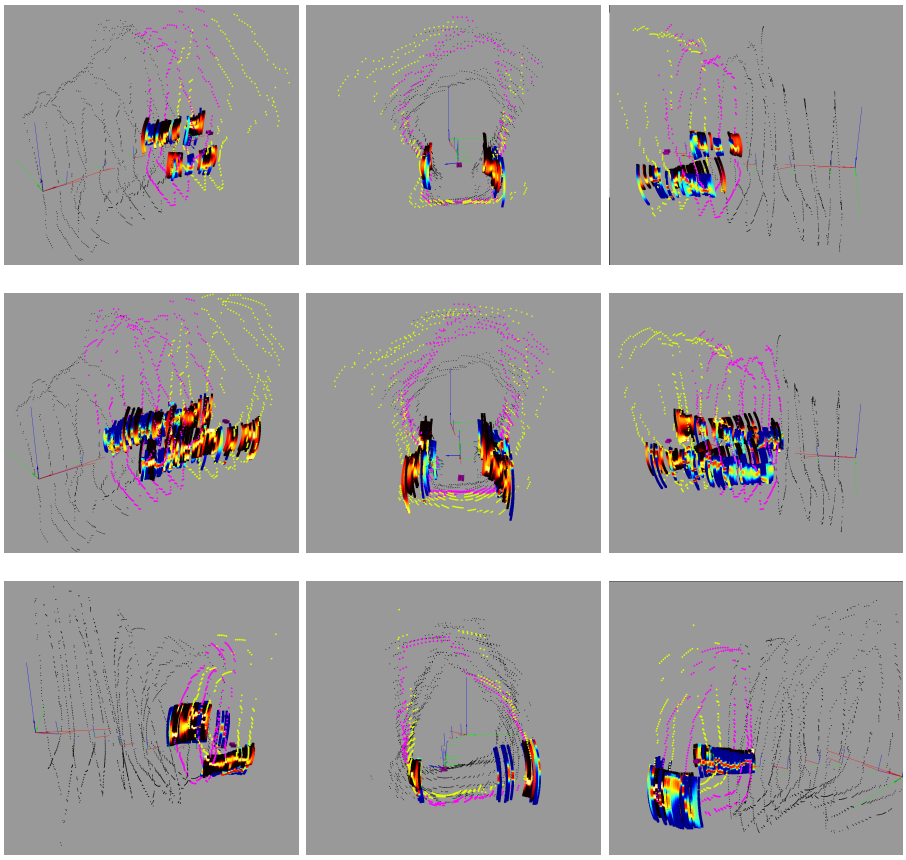
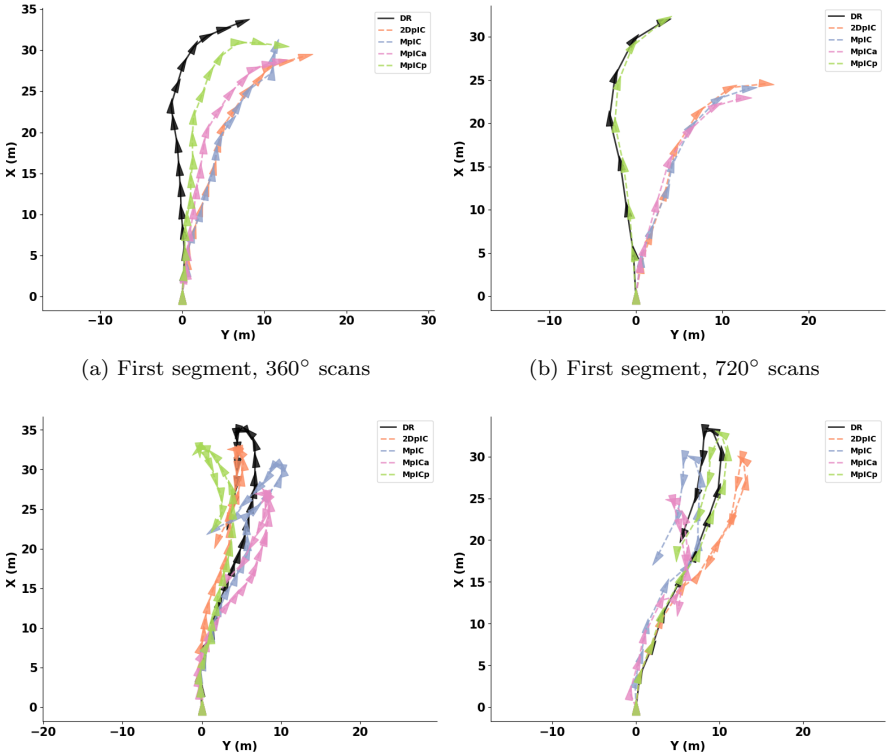


Figure 15: Examples of elevation angles estimations on the dataset [7]. First row : first segment with 360° scans. Second row : first segment with 720° scans. Third row : second segment with 360° scans. Yellow (resp. purple) points corresponds to the current (resp. previous) scan's vertical points. Blue-to-Red (resp. Black-to-Yellow) arcs represents the first (resp. second) horizontal scan elevation angles pdf. Best viewed in color.

(DR) trajectories with trajectories obtained by pIC methods. DR estimation drifts over time, but locally the estimated trajectories from pIC approaches should be similar.

Our tests are done on two small segments as shown in Figure 14 which represents the acoustic map obtained with the SLAM approach in [3]. The first and second segment start respectively at timestamp ts_1 and ts_2 given by

$$ts_1 = 1372687311775722265,\ ts_2 = 1372687577375451326$$



(a) First segment, 360° scans

(b) First segment, 720° scans

(c) Second segment, 360° scans

(d) Second segment, 720° scans

Figure 16: 2D trajectories obtained by composing estimated relative poses of the robot. The segments corresponds to the red areas in Figure 14. Best viewed in color.

Similarly to the previous section, we compare here 2DpIC, MpIC, MpICa and MpICp methods. Following the observation in the simulation case, we fix the association confidence level to $\alpha = 0.5$. We also compare the results obtained by considering 360° and 720° scans.

Figure 15 shows the estimated horizontal sonar elevation angles for one pair of scans. While our previous work [4] was evaluated on simulated karst environment, this illustrates its applicability to real data. It can deal with discontinuities or missing data such as holes in the surface.

In areas where large parts of the environment are unseen by the vertical sonar (Green rectangles in Figure 14), the implementation used in this paper can not estimate elevation angles of horizontal measurements related to the unseen part

of the surface. This problem can be partially solved by considering for the local range ρ^v a uniform distribution on $]r_{max}, M]$ (for some large enough M) when no data is returned by the vertical sonar. This information can then be leveraged for the elevation angle estimations of the horizontal sonar measurements.

Figure 16 represents the local trajectories obtained by composing pose estimated by DR or pIC methods. Note that as the trajectories are obtained by composing estimated relative poses, a single error can lead to a large divergence from the ground truth. Compared to our simulated karst, the environment in both segments is similar to a corridor. This means that the relative displacements along the local x-axis (forward direction) are harder to estimate. This can be seen by comparing Figures 16a, 16c with Figures 16b, 16d. MpIC also has an improved trajectory shape with longer scan while the results for 2DpIC are rather similar for both 360° and 720° scans. This is expected as 2DpIC exploits all the horizontal sonar measurements while MpIC and MpICp are restrained to the surface covered by the vertical sonar. MpICa, which is a compromise between 2DpIC and MpIC by considering horizontal sonar measurements with no elevation angle estimation (uniform distribution), does not improve upon 2DpIC or MpIC/MpICp. Finally, in both segments, MpICp provides the trajectories closest to the DR when using 720° scans. Compared to the simulation, we have sparser 3D point clouds with larger misalignement. While this directly affects the performance of MpIC (*point-to-point* association), MpICp (*point-to-plane* association) can deal with such misaligment by leveraging the estimated surface as explained in section 5.3.

7. Conclusion

In this paper we propose MpIC, an on-manifold derivation for the 3D probabilistic Iterative Correspondence (pIC) algorithm based on basic differential geometry related to the SE(3) Lie group. We then propose an application to underwater sonar scan matching in the context of underwater karst exploration. We extended our previous work [4] on elevation angle estimation from wide-beam sonar to generate 3D point Gaussian distribution from the raw sonar

measurements. The approach is evaluated on a simulated karst environment to allow quantitative analysis. It is also compared to the state-of-the-art approach proposed in [7].

In the robotic literature, ICP algorithms are generally evaluated indirectly in a SLAM framework and/or qualitatively on real data experiments [2][3][6]. Quantitative results, when available, are obtained by self-matching scans with added noise [1][19]. We propose here a compromise by proposing a quantitative evaluation of pIC with a more complex setting similar to a real experiment.

We show that, as one could expect, MpIC outperforms original pIC based on Euler angles and Quaternion representations. We then analyzed and compared different pIC approaches when applied to acoustic sonar data in several configurations on a simulated karst environment and on real data. Compared to previous state-of-the-art 2D method, our MpIC approach converges to more accurate transformations with higher lower bounds. The *point-to-plane* association scheme is also more accurate than *point-to-point* one, showing the pertinence of our original normal estimation method. Our experiments also show more general results on pIC. For successive scan matching with normally distributed dead-reckoning noise, the different pIC methods provides only slight improvement on the estimated transformations. However, in real settings, one can expect dead-reckoning noise outliers which do not follow the modeled Gaussian distribution. For large initial transformations such as encountered in loop closure scan matching, MpIC also provides better results over the 2D approach.

The association confidence level is an important parameter of the pIC algorithms. We show through our experiments that it should be adapted to the ratio between the initial transformation error and the scan local odometric errors. This is rather intuitive : for small displacements, corresponding points are near from each other so that the association threshold should be low to avoid spurious associations. For larger displacements, corresponding points are far from each other requiring thus a larger association threshold.

While the results are promising, there is still room for improvements. We approximated our points distribution by Gaussian distributions. For points coming

from sonar measurements with a high elevation angle, a large part of the Gaussian distribution covers an area outside of the original beam measurement. We consider relaxing the Gaussian approximation and use a more generic distance function to replace the Mahalanobis distance. This approach would certainly be costly computationally-wise but can be interesting to further increase localization accuracy offline. We are currently working towards the integration of this algorithm in a SLAM framework for underwater karst exploration. As the robot explores unknown environments, our approach will have to adapt dynamically (eg. Gaussian kernel parameters, scans length).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is a LabEx NUMEV and MUSE institution project publicly funded through ANR (French National Research Agency) under the "Investissements d'avenir" program (PIA) (references ANR-16-IDEX-0006,ANR-10-LABX-20). It is supported by the Region OCCITANIE and European FEDER Funds.

References

[1] Montesano L, Minguez J, Montano L. Probabilistic scan matching for motion estimation in unstructured environments. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE; 2005, p. 3499–504.

[2] Mallios A, Ridao P, Ribas D, Hernández E. Scan matching slam in underwater environments. Autonomous Robots 2014;36(3):181–98.

[3] Mallios A, Ridao P, Ribas D, Carreras M, Camilli R. Toward autonomous exploration in confined underwater environments. Journal of Field Robotics 2016;33(7):994–1012.

[4] Breux Y, Lapierre L. Elevation angle estimations of wide-beam acoustic sonar measurements for autonomous underwater karst exploration. Sensors 2020;20(14):4028.

[5] Burguera A, González Y, Oliver G. The uspic: Performing scan matching localization using an imaging sonar. Sensors 2012;12(6):7855–85.

[6] Palomer A, Ridao P, Ribas D. Multibeam 3d underwater slam with probabilistic registration. Sensors 2016;16(4):560.

[7] Mallios A, Vidal E, Campos R, Carreras M. Underwater caves sonar data set. The International Journal of Robotics Research 2017;36(12):1247–51.

[8] Besl PJ, McKay ND. Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures; vol. 1611. International Society for Optics and Photonics; 1992, p. 586–606.

[9] Chen Y, Medioni G. Object modelling by registration of multiple range images. Image and vision computing 1992;10(3):145–55.

[10] Censi A. An icp variant using a point-to-line metric. In: 2008 IEEE International Conference on Robotics and Automation. Ieee; 2008, p. 19–25.

[11] Segal A, Haehnel D, Thrun S. Generalized-icp. In: Robotics: science and systems; vol. 2. Seattle, WA; 2009, p. 435.

[12] Druon S, Aldon MJ, Crosnier A. Color constrained icp for registration of large unstructured 3d color data sets. In: 2006 IEEE International Conference on Information Acquisition. IEEE; 2006, p. 249–55.

[13] Servos J, Waslander SL. Multi-channel generalized-icp: A robust framework for multi-channel scan registration. Robotics and Autonomous systems 2017;87:247–57.

[14] Agamennoni G, Fontana S, Siegwart RY, Sorrenti DG. Point clouds registration with probabilistic data association. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2016, p. 4092–8.

[15] Rusinkiewicz S. A symmetric objective function for icp. ACM Transactions on Graphics (TOG) 2019;38(4):1–7.

[16] Du S, Xu G, Zhang S, Zhang X, Gao Y, Chen B. Robust rigid registration algorithm based on pointwise correspondence and correntropy. Pattern Recognition Letters 2020;132:91–8.

[17] Pomerleau F, Colas F, Siegwart R. A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics 2015;4(1):1–104.

[18] Blanco JL. A tutorial on se (3) transformation parameterizations and on-manifold optimization. University of Malaga, Tech Rep 2010;3.

[19] Burguera A. A novel approach to register sonar data for underwater robot localization. In: 2017 Intelligent Systems Conference (IntelliSys). IEEE; 2017, p. 1034–43.

[20] McConnell J, Englot B. Predictive 3d sonar mapping of underwater environments via object-specific bayesian inference. arXiv preprint arXiv:210403203 2021;.

[21] McConnell J, Martin JD, Englot B. Fusing concurrent orthogonal wide-aperture sonar images for dense underwater 3d reconstruction. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020, p. 1653–60. doi:10.1109/IROS45743.2020.9340995.

[22] Sola J, Deray J, Atchuthan D. A micro lie theory for state estimation in robotics. arXiv preprint arXiv:181201537 2018;.

[23] Gallier J, Quaintance J. Differential Geometry and Lie Groups: A Computational Perspective; vol. 12. Springer Nature; 2020.

[24] Belta C, Kumar V. Euclidean metrics for motion generation on se (3). Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 2002;216(1):47–60.

[25] Censi A. An accurate closed-form estimate of icp's covariance. In: Proceedings 2007 IEEE international conference on robotics and automation. IEEE; 2007, p. 3167–72.

[26] Williams CK, Rasmussen CE. Gaussian processes for machine learning; vol. 2. MIT press Cambridge, MA; 2006.

[27] Said S, Bombrun L, Berthoumieu Y, Manton JH. Riemannian gaussian distributions on the space of symmetric positive definite matrices. IEEE Transactions on Information Theory 2017;63(4):2153–70.

[28] Higham NJ, Lin L. A schur–padé algorithm for fractional powers of a matrix. SIAM Journal on Matrix Analysis and Applications 2011;32(3):1056–78.

[29] Carreras M, Candela C, Ribas D, Palomeras N, Magí L, Mallios A, et al. Testing sparus ii auv, an open platform for industrial, scientific and academic applications. Instrumentation viewpoint 2015;.

Appendix A. Definitions and properties of matrix product

We note $M_{m,n}$ the vector space of real $m \times n$ matrices. To simplify notations, we note $[a_{ij}] \in M_{m,n}$ a matrix with generic term a_{ij}. Similarly, we define block matrices as $[B_{ij}]$. The $vec\,() : M_{m,n} \to \mathbb{R}^{mn}$ operator stacks the columns of a matrix into a vector.

Definition 1. *Let* $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T \in \mathbb{R}^3$. *The operator* $[\,]_\times$ *maps 3D vector to the space of skew-symmetric matrix and is defined as*

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Using this operator, the cross product operation can be expressed as a matrix vector operation :

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^3,\ \mathbf{x} \times \mathbf{y} = [\mathbf{x}]_\times \mathbf{y}$$

Definition 2. *The Kronecker product, noted* $\otimes$, *between two matrices* $A = [a_{ij}] \in M_{m,n},\ B \in M_{p,q}$ *is defined by the partitioned matrix*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} = [a_{ij}B] \in M_{mp,nq} \tag{A.1}$$

The Kronecker power notation is defined by

$$A^{\otimes n} = A \otimes A \dots \otimes A$$

Proposition 6. *Let* $A \in M_{m,n},\ X \in M_{n,p}$ *and* $B \in M_{p,q}$. *Then*

$$vec\,(AXB) = \left(B^T \otimes A\right) vec\,(X)$$

Proposition 7. *Let* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, $A \in M_{p,m}$ *such that each element* a_{ij} *of* A *is a function from* $\mathbb{R}^n$ *to* $\mathbb{R}$. *Suppose that both* f *and* A *are function of the same variable* $\mathbf{x} \in \mathbb{R}^n$. *Then*

$$\frac{dAf}{d\mathbf{x}} = A\frac{df}{d\mathbf{x}} + \left(I_p \otimes \mathbf{f}^T\right) \frac{dvec\left(A^T\right)}{d\mathbf{x}}$$

Proof. Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{a}_i^T$ the i-th row of A. We then have

$$\begin{aligned} \frac{dAf}{dx} &= \left[\frac{\partial \mathbf{a}_i^T \mathbf{f}}{\partial x_j}\right] \\ &= \left[\mathbf{a}_i^T \frac{\partial \mathbf{f}}{\partial x_j}\right] + \left[\mathbf{f}^T \frac{\partial \mathbf{a}_i}{\partial x_j}\right] \end{aligned}$$

The first term is simply

$$\left[\mathbf{a}_i^T \frac{\partial \mathbf{f}}{\partial x_j}\right] = A\frac{d\mathbf{f}}{d\mathbf{x}}$$

The second term is given by

$$\begin{aligned} \left[\mathbf{f}^T \frac{\partial \mathbf{a}_i}{\partial x_j}\right] &= \begin{bmatrix} \mathbf{f}^T & & \\ & \ddots & \\ & & \mathbf{f}^T \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{a}_1}{\partial x_1} & \dots & \frac{\partial \mathbf{a}_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial \mathbf{a}_p}{\partial x_1} & \dots & \frac{\partial \mathbf{a}_p}{\partial x_n} \end{bmatrix} \\ &= \left(I_p \otimes \mathbf{f}^T\right) \left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right] \end{aligned}$$

where $\left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right]$ is a $p \times 1$ block-row matrix. By considering the transpose of A, we have the corresponding expression

$$\left[\frac{\partial \mathbf{a}_i}{\partial x_j}\right] = \frac{dVec\left(A^T\right)}{d\mathbf{x}}$$

□

Appendix B. Derivations and proofs

We recall here some notations used in the following derivations. Let define $\mathbf{q} \in SE(3)$ and $\widehat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$. Let $F_i = \bar{\boldsymbol{e}}_i^T \Sigma_{e_i} \bar{\boldsymbol{e}}_i$ with $\bar{\boldsymbol{e}}_i = \bar{\boldsymbol{n}}_i - \bar{\boldsymbol{a}}_i = \mathbf{q} \oplus \bar{\boldsymbol{c}}_i - \bar{\boldsymbol{a}}_i$ from (25)(34). Thus the cost function F in (38) can be written as $F = \sum_{i=1}^{N} F_i$. We denote γ the geodesic of $SE(3)$ going through $\mathbf{q}$ ($\gamma(0) = \mathbf{q}$) and with velocity $\widehat{\boldsymbol{\xi}}$ ($\gamma'(0) = \widehat{\boldsymbol{\xi}}$). Following (9), it is given by $\gamma(t) = Te^{t\widehat{\boldsymbol{\xi}}}$ where $\mathbf{q} \equiv T \in GA(3)$ as in (4).

Appendix B.1. Proof of Proposition 1

Based on the differential definition in (8), we have

$$\frac{d}{dt} F_i(\gamma(t)) = \bar{\boldsymbol{e}}_i^T \left(2\Sigma_{\mathbf{e}_i}^{-1} \frac{d\bar{\boldsymbol{e}}_i}{dt} + \frac{d\Sigma_{\mathbf{e}_i}^{-1}}{dt} \bar{\boldsymbol{e}}_i \right) \tag{B.1}$$

We have from (14)

$$\frac{d\bar{\boldsymbol{e}}_i}{dt}\Big|_{t=0} = RU_{\bar{\boldsymbol{c}}_i} \boldsymbol{\xi} \tag{B.2}$$

Similarly,

$$\frac{dR}{dt}\Big|_{t=0} = R\widehat{\boldsymbol{\omega}} \tag{B.3}$$

From (36) and (B.3), we obtain

$$\frac{d\Sigma_{\mathbf{e}_i}^{-1}}{dt}\Big|_{t=0} = -\Sigma_{\mathbf{e}_i}^{-1} \frac{d\Sigma_{\mathbf{e}_i}}{dt}\Big|_{t=0} \Sigma_{\mathbf{e}_i}^{-1} \tag{B.4}$$

$$\begin{aligned} \frac{d\Sigma_{\mathbf{e}_i}}{dt}\Big|_{t=0} &= \frac{d\left(R\Omega_{\bar{\boldsymbol{c}}_i} R^T\right)}{dt}\Big|_{t=0} \\ &= \frac{dR}{dt}\Big|_{t=0} \Omega_{\bar{\boldsymbol{c}}_i} R^T + R\Omega_{\bar{\boldsymbol{c}}_i} \frac{dR^T}{dt}\Big|_{t=0} \\ &= R\left(\widehat{\boldsymbol{\omega}}\Omega_{\bar{\boldsymbol{c}}_i} - \Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}}\right) R^T \end{aligned} \tag{B.5}$$

Replacing (B.2)(B.4)(B.5) in (B.1) gives the expected results provided by (41)

$$\begin{aligned} DF_{\mathbf{q}}(T\widehat{\boldsymbol{\xi}}) &= \sum_{i=1}^{N} \frac{d}{dt} F_i(\gamma(t))\Big|_{t=0} \\ &= \sum_{i=1}^{N} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R\Big(2U_{\bar{\boldsymbol{c}}_i}\boldsymbol{\xi} + (\Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} - \widehat{\boldsymbol{\omega}}\Omega_{\bar{\boldsymbol{c}}_i})\, R^T \Sigma_{\mathbf{e}_i}^{-1} \bar{\boldsymbol{e}}_i\Big) \end{aligned} \tag{B.6}$$

Appendix B.2. Proof of Proposition 2

Appendix B.2.1. Derivation of H_q^F

By derivating (B.1) we have

$$\begin{aligned} \frac{d^2}{dt^2} F_i(\gamma(t)) = \frac{d\bar{\boldsymbol{e}}_i}{dt}^T &\left(2\Sigma_{\mathbf{e}_i}^{-1} \frac{d\bar{\boldsymbol{e}}_i}{dt} + \frac{d\Sigma_{\mathbf{e}_i}^{-1}}{dt}\bar{\boldsymbol{e}}_i\right) \\ &+ \bar{\boldsymbol{e}}_i^T \left(3\frac{d\Sigma_{\mathbf{e}_i}^{-1}}{dt}\frac{d\bar{\boldsymbol{e}}_i}{dt} + 2\Sigma_{\mathbf{e}_i}^{-1}\frac{d^2\bar{\boldsymbol{e}}_i}{dt^2} + \frac{d^2\Sigma_{\mathbf{e}_i}^{-1}}{dt^2}\bar{\boldsymbol{e}}_i\right) \end{aligned} \tag{B.7}$$

In homogeneous coordinates, the second derivative of $\bar{\boldsymbol{e}}_i$ is given by

$$\frac{d^2\bar{\boldsymbol{e}}_i}{dt^2} = T\widehat{\boldsymbol{\xi}}^2 e^{t\widehat{\boldsymbol{\xi}}}\bar{\boldsymbol{c}}_i \tag{B.8}$$

from which we deduce in heterogeneous coordinates

$$\frac{d^2\bar{\boldsymbol{e}}_i}{dt^2}\Big|_{t=0} = R\widehat{\boldsymbol{\omega}}\left(\widehat{\boldsymbol{\omega}}\bar{\boldsymbol{c}}_i + \tau\right) \tag{B.9}$$

We recall (B.4)

$$\frac{d\Sigma_{\mathbf{e}_i}^{-1}}{dt} = \Sigma_{\mathbf{e}_i}^{-1} R\left(e^{t\widehat{\boldsymbol{\omega}}}\Omega_{\bar{\boldsymbol{c}}_i}e^{-t\widehat{\boldsymbol{\omega}}}\widehat{\boldsymbol{\omega}} - \widehat{\boldsymbol{\omega}}e^{t\widehat{\boldsymbol{\omega}}}\Omega_{\bar{\boldsymbol{c}}_i}e^{-t\widehat{\boldsymbol{\omega}}}\right) R^T\Sigma_{\mathbf{e}_i}^{-1} \tag{B.10}$$

Derivating it once, we obtain after simplification

$$\frac{d^2\Sigma_{\mathbf{e}_i}^{-1}}{dt^2}\Big|_{t=0} = \Sigma_{\mathbf{e}_i}^{-1} R\left(2\Lambda_{\omega_i} R^T\Sigma_{\mathbf{e}_i}^{-1} R\Lambda_{\omega_i} + D_{\omega_i}\right) R^T\Sigma_{\mathbf{e}_i}^{-1} \tag{B.11}$$

with

$$\Lambda_{\omega_i} = \Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} \tag{B.12}$$

$$D_{\omega_i} = 2\widehat{\boldsymbol{\omega}}\Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}} - \Omega_{\bar{\boldsymbol{c}}_i}\widehat{\boldsymbol{\omega}}^2 - \widehat{\boldsymbol{\omega}}^2\Omega_{\bar{\boldsymbol{c}}_i} \tag{B.13}$$

The final result is obtained by replacing the intermediate computations in (B.7)

$$\begin{aligned} Hess(F)_{\mathbf{q}}\left(T\widehat{\boldsymbol{\xi}}, T\widehat{\boldsymbol{\xi}}\right) &= \frac{d^2}{dt^2} F_i(\gamma(t))\Big|_{t=0} \\ &= \sum_{i=0}^{N} \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R\Big(2\widehat{\boldsymbol{\omega}}\left(\widehat{\boldsymbol{\omega}}\bar{\boldsymbol{c}}_i + \tau\right) \\ &\quad + D_{\omega_i} R^T \Sigma_{\mathbf{e}_i}^{-1} \bar{\boldsymbol{e}}_i \\ &\quad + K_{\omega_i} R\left(2K_{\omega_i}\bar{\boldsymbol{e}}_i + 3U_{\bar{\boldsymbol{c}}_i}\boldsymbol{\xi}\right)\Big) \end{aligned} \tag{B.14}$$

$$K_{\omega_i} = (\Lambda_{\omega_i} + \Lambda_{\omega_i}^T) R^T \Sigma_{\mathbf{e}_i}^{-1} \tag{B.15}$$

Appendix B.2.2. Derivation of $H^F_{\mathbf{q},\mathbf{z}}$

The hessian matrix $H^{F_i}_{\mathbf{q},\mathbf{z}_i}$ is defined by

$$H^{F_i}_{\mathbf{q},\mathbf{z}_i} = \frac{\partial J_{\mathbf{q}}^{F_i^T}}{\partial \mathbf{z}_i} \tag{B.16}$$

$$= \left[\frac{\partial DF_{i_{\mathbf{q}}}(TG_k^{SE(3)})}{\partial \mathbf{z}_i}\right] \tag{B.17}$$

$$= \left[\frac{\partial DF_{i_{\mathbf{q}}}(TG_k^{SE(3)})}{\partial \mathbf{a}_i} \;\; \frac{\partial DF_{i_{\mathbf{q}}}(TG_k^{SE(3)})}{\partial \mathbf{c}_i}\right] \tag{B.18}$$

where we recall that $G_k^{SE(3)} = \widehat{\boldsymbol{\xi}}_k$, $\boldsymbol{\xi}_k$ being the k-th unit vector of the canonical base of $\mathbb{R}^6$(Generally the canonical base vector are denoted $\mathbf{e}_i$, as in Section 3.1. We change the notation here to avoid ambiguity with the random variables representing the association errors). First we have

$$\frac{\partial \bar{\boldsymbol{e}}}{\partial \mathbf{z}} = \left[\frac{\partial \bar{\boldsymbol{e}}}{\partial \mathbf{a}} \quad \frac{\partial \bar{\boldsymbol{e}}}{\partial \mathbf{c}}\right] = \left[-I_3 \quad R\right]$$

It follows, with K_{ω_i} defined by (B.15)

$$\frac{\partial DF_{i_{\mathbf{q}}}(T\widehat{\boldsymbol{\xi}})}{\partial \mathbf{a}_i} = -2\left(\boldsymbol{\xi}^T U_{\bar{\boldsymbol{c}}_i}^T + \bar{\boldsymbol{e}}_i^T K_{\omega_i}^T\right) R^T \Sigma_{\mathbf{e}_i}^{-1} \tag{B.19}$$

from which we deduce

$$
\begin{aligned}
H^{F_i}_{\mathbf{q},\mathbf{a}_i} &= \left[\frac{\partial DF_{i_\mathbf{q}}(T\widehat{\boldsymbol{\xi}}_k)}{\partial \mathbf{a}_i}\right] \\
&= -2\Big(U^T_{\bar{\boldsymbol{c}}_i} R^T \Sigma^{-1}_{\mathbf{e}_i} + \left[\bar{\boldsymbol{e}}_i^T K^T_{\omega_i}(\widehat{\boldsymbol{\xi}}_k) R^T \Sigma^{-1}_{\mathbf{e}_i}\right]\Big) \\
&= -2\Big(U^T_{\bar{\boldsymbol{c}}_i} R^T \Sigma^{-1}_{\mathbf{e}_i} \\
&\quad + \left(I_3 \otimes \bar{\boldsymbol{e}}_i^T R^T \Sigma^{-1}_{\mathbf{e}_i}\right)\left[(\Lambda_{\omega_i} + \Lambda^T_{\omega_i})(\widehat{\boldsymbol{\xi}}_k)\right] R^T \Sigma^{-1}_{\mathbf{e}_i}\Big) \\
&= -2\Big(U^T_{\bar{\boldsymbol{c}}_i} + \left(I_3 \otimes \bar{\boldsymbol{e}}_i^T R^T \Sigma^{-1}_{\mathbf{e}_i}\right)\left(K + K^T\right)\Big) R^T \Sigma^{-1}_{\mathbf{e}_i} \qquad \text{(B.20)}
\end{aligned}
$$

with $K = (I_6 \otimes \Omega_{\bar{\boldsymbol{c}}_i}) \begin{bmatrix} G_1^{SO(3)} \\ G_2^{SO(3)} \\ G_3^{SO(3)} \\ 0_{9,3} \end{bmatrix}$.

For the derivation w.r.t $\bar{\boldsymbol{c}}_i$, we have

$$
\begin{aligned}
\frac{\partial DF_{i_\mathbf{q}}(T\widehat{\boldsymbol{\xi}})}{\partial \mathbf{c}_i} &= 2\left(A_1 + A_2\right) + A_3 + A_4 \qquad \text{(B.21)} \\
A_1 &= \bar{\boldsymbol{e}}_i^T \frac{\partial \Sigma^{-1}_{\mathbf{e}_i} R U_{\bar{\boldsymbol{c}}_i} \boldsymbol{\xi}}{\partial \bar{\boldsymbol{c}}_i} \qquad \text{(B.22)} \\
A_2 &= \boldsymbol{\xi}^T U^T_{\bar{\boldsymbol{c}}_i} R^T \Sigma^{-1}_{\mathbf{e}_i} \frac{\partial \bar{\boldsymbol{e}}_i}{\partial \bar{\boldsymbol{c}}_i} \\
&= \boldsymbol{\xi}^T U^T_{\bar{\boldsymbol{c}}_i} R^T \Sigma^{-1}_{\mathbf{e}_i} R \qquad \text{(B.23)} \\
A_3 &= \bar{\boldsymbol{e}}_i^T \left(\Sigma^{-1}_{\mathbf{e}_i} R K_{\omega_i} + \left(\Sigma^{-1}_{\mathbf{e}_i} R K_{\omega_i}\right)^T\right) \frac{\partial \bar{\boldsymbol{e}}_i}{\partial \bar{\boldsymbol{c}}_i} \\
&= 2\bar{\boldsymbol{e}}_i^T \Sigma^{-1}_{\mathbf{e}_i} R \left(\Lambda_{\omega_i} + \Lambda^T_{\omega_i}\right) R^T \Sigma^{-1}_{\mathbf{e}_i} R \qquad \text{(B.24)} \\
A_4 &= \bar{\boldsymbol{e}}_i^T \left[\frac{\partial (\Sigma^{-1}_{\mathbf{e}_i} R K_{\omega_i})^T}{\partial \bar{\boldsymbol{c}}_{i,j}}\right] (I_3 \otimes \bar{\boldsymbol{e}}_i) \qquad \text{(B.25)}
\end{aligned}
$$

We then have $H^{F_i}_{\mathbf{q},\mathbf{c}_i} = 2\left(\left[A_1(\widehat{\boldsymbol{\xi}}_k)\right] + \left[A_2(\widehat{\boldsymbol{\xi}}_k)\right]\right) + \left[A_3(\widehat{\boldsymbol{\xi}}_k)\right] + \left[A_4(\widehat{\boldsymbol{\xi}}_k)\right]$. The first term is given by

$$
\begin{aligned}
\left[A_1(\widehat{\boldsymbol{\xi}}_k)\right] &= \left[\bar{\boldsymbol{e}}_i^T \frac{\partial \Sigma^{-1}_{\mathbf{e}_i} R U_{\bar{\boldsymbol{c}}_i} \boldsymbol{\xi}_k}{\partial c_{i,j}}\right] \qquad \text{(B.26)} \\
&= \left(I_6 \otimes \bar{\boldsymbol{e}}_i^T\right)\left[vec\left(\frac{\partial \Sigma^{-1}_{\mathbf{e}_i} R U_{\bar{\boldsymbol{c}}_i}}{\partial c_{i,j}}\right)\right] \qquad \text{(B.27)}
\end{aligned}
$$

Using the following results for $j \in [\![1,3]\!]$

$$\frac{\partial \Sigma_{\mathbf{e}_i}}{\partial c_{i,j}} = R \frac{\partial \Omega_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} R^T \tag{B.28}$$

$$\frac{\partial \Omega_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} = \left(\frac{\partial U_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} \Sigma_q U_{\bar{\mathbf{c}}_i}^T + U_{\bar{\mathbf{c}}_i} \Sigma_q \frac{\partial U_{\bar{\mathbf{c}}_i}^T}{\partial c_{i,j}} \right) \tag{B.29}$$

$$\frac{\partial U_{\bar{\mathbf{c}}_i}^T}{\partial c_{i,j}} = \begin{bmatrix} -G_j^{SO(3)} & 0_{3,3} \end{bmatrix} \tag{B.30}$$

We obtain

$$\begin{aligned} \frac{\partial \Sigma_{\mathbf{e}_i}^{-1} R U_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} = \Sigma_{\mathbf{e}_i}^{-1} R \Bigg(& \frac{\partial U_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} \\ & - \left(\frac{\partial U_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} \Sigma_q U_{\bar{\mathbf{c}}_i}^T + U_{\bar{\mathbf{c}}_i} \Sigma_q \frac{\partial U_{\bar{\mathbf{c}}_i}^T}{\partial c_{i,j}} \right) R^T \Sigma_{\mathbf{e}_i}^{-1} R U_{\bar{\mathbf{c}}_i} \Bigg) \end{aligned} \tag{B.31}$$

The other terms are given by

$$\left[A_2(\widehat{\boldsymbol{\xi}}_k) \right] = U_{\bar{\mathbf{c}}_i}^T R^T \Sigma_{\mathbf{e}_i}^{-1} R \tag{B.32}$$

$$\left[A_3(\widehat{\boldsymbol{\xi}}_k) \right] = 2 \left(I_6 \otimes \bar{\boldsymbol{e}}_i^T \Sigma_{\mathbf{e}_i}^{-1} R \right) \left(K + K^T \right) R^T \Sigma_{\mathbf{e}_i}^{-1} R$$

$$\left[A_4(\widehat{\boldsymbol{\xi}}_k) \right] = (I_6 \otimes \bar{\boldsymbol{e}}_i) \begin{bmatrix} \frac{\partial (\Sigma_{\mathbf{e}_i}^{-1} R K_{\omega_i} (G_k^{SO(3)}))^T}{\partial \bar{\mathbf{c}}_{i,j}} \\ 0_{9,3} \end{bmatrix} (I_3 \otimes \bar{\boldsymbol{e}}_i) \tag{B.33}$$

with

$$\frac{\partial (\Sigma_{\mathbf{e}_i}^{-1} R K_{\omega_i} (G_k^{SO(3)}))^T}{\partial c_{i,j}} = \Sigma_{\mathbf{e}_i}^{-1} \left(\left(R(K_j' + K_j'^T) - \frac{\partial \Sigma_{\mathbf{e}_i}}{\partial c_{i,j}} K_{\omega_i}^T \right) R^T \right. \tag{B.34}$$

$$\left. - R K_{\omega_i} \frac{\partial \Sigma_{\mathbf{e}_i}}{\partial c_{i,j}} \right) \Sigma_{\mathbf{e}_i}^{-1} \tag{B.35}$$

$$K_j' = \left(I_6 \otimes \frac{\partial \Omega_{\bar{\mathbf{c}}_i}}{\partial c_{i,j}} \right) \begin{bmatrix} G_1^{SO(3)} \\ G_2^{SO(3)} \\ G_3^{SO(3)} \\ 0_{9,3} \end{bmatrix} \tag{B.36}$$

Appendix B.3. Proof of Proposition 3

First, recall the SBeta pdf f followed by θ :

$$\begin{aligned} f(\theta) &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\left(\theta+\frac{b}{2}\right)^{\alpha-1}\left(\frac{b}{2}-\theta\right)^{\beta-1}}{b^{\alpha+\beta-1}} \\ &= K(\alpha,\beta,b)\left(\theta+\frac{b}{2}\right)^{\alpha-1}\left(\frac{b}{2}-\theta\right)^{\beta-1} \end{aligned} \tag{B.37}$$

where Γ is the Gamma function. Using the power series of cosinus, we have

$$\begin{aligned} E(\cos\theta) &= \int_{-\frac{b}{2}}^{\frac{b}{2}} cos(\theta) f(\theta) d\theta \\ &= K(\alpha,\beta,b)\int_{-\frac{b}{2}}^{\frac{b}{2}} \sum_{n=0}^{+\infty} \frac{(-1)^n}{2n!}\theta^{2n}\left(\theta+\frac{b}{2}\right)^{\alpha-1}\left(\frac{b}{2}-\theta\right)^{\beta-1} d\theta \end{aligned} \tag{B.38}$$

One can easily show that the serie-integral inversion holds and gives

$$E(\cos\theta) = K(\alpha,\beta,b)\sum_{n=0}^{+\infty}\frac{(-1)^n}{2n!}\int_{-\frac{b}{2}}^{\frac{b}{2}}\theta^{2n}\left(\theta+\frac{b}{2}\right)^{\alpha-1}\left(\frac{b}{2}-\theta\right)^{\beta-1} d\theta \tag{B.39}$$

With a simple change of variable, the integral can be expressed as

$$\begin{aligned} \int_{-\frac{b}{2}}^{\frac{b}{2}}\theta^{2n}\left(\theta+\frac{b}{2}\right)^{\alpha-1}\left(\frac{b}{2}-\theta\right)^{\beta-1} d\theta &= \frac{b^{\alpha+\beta+2n-1}}{4^n}\int_0^1 \theta^{\alpha-1}(1-2\theta)^{2n}(1-\theta)^{\beta-1} d\theta \\ &= \frac{b^{\alpha+\beta+2n-1}}{4^n}\frac{{}_2F_1(-2n,\alpha,\alpha+\beta;2)}{K(\alpha,\beta,b)b^{\alpha+\beta-1}} \\ &= \left(\frac{b}{2}\right)^{2n}{}_2F_1(-2n,\alpha,\alpha+\beta;2) \end{aligned} \tag{B.40}$$

where we used the integral form of the Hypergeometric function ${}_2F_1$.

The result for $E(\sin\theta)$ and $E(\cos^2\theta)$ are obtained similarly using the power series

$$\begin{aligned} \sin\theta &= \sum_{n=0}^{+\infty}\frac{(-1)^{2n+1}}{2n+1!}\theta^{2n+1} \\ \cos^2\theta &= 1+\sum_{n=1}^{+\infty}\frac{(-1)^n}{2n!}2^{2n-1}\theta^{2n} \end{aligned}$$

Appendix B.4. Proof of Proposition 4

First define the Pochhammer symbol as a generalization of the usual factorial. Let $a \in \mathbb{R}$ and $n \in \mathbb{N}^*$:

$$(a)_n = a(a+1)\dots(a+n-1) \tag{B.41}$$

and set $(a)_0$ otherwise. The hypergeometric function ${}_2F_1$ depends on 3 parameters and is defined for $m \in \mathbb{N}$ and $b, c \in \mathbb{R}^+$ as :

$$ {}_2F_1(-m, b, c; z) = \sum_{n=0}^{m} (-1)^n \binom{m}{n} \frac{(b)_n}{(c)_n} z^n \tag{B.42}$$

We aim at controlling the reminder for the series :

$$ R_N\left(E(\cos\theta)\right) = \sum_{n=N}^{+\infty} \frac{(-1)^n}{2n!} \left(\frac{b}{2}\right)^{2n} {}_2F_1(-2n, \alpha, \alpha+\beta; 2) \tag{B.43}$$

Notice that

$$ {}_2F_1(-2n, \alpha, \alpha+\beta; 2) = \sum_{k=0}^{2n} (-1)^k \binom{2n}{k} \frac{(\alpha)_k}{(\alpha+\beta)_k} 2^k \tag{B.44}$$

Since $\frac{(\alpha)_k}{(\alpha+\beta)_k} \leq 1$, we get $|{}_2F_1(-2n, \alpha, \alpha+\beta; 2)| \leq \sum_{k=0}^{2n} \binom{2n}{k} 2^k = 3^{2n}$ so that

$$ \left|R_N\left(E(\cos\theta)\right)\right| \leq \sum_{n=N}^{+\infty} \frac{1}{(2n)!} \left(\frac{3b}{2}\right)^{2n} = \sum_{k=0}^{+\infty} \frac{1}{(2(k+N))!} \left(\frac{3b}{2}\right)^{2(k+N)} \tag{B.45}$$

$$ \leq \frac{1}{(2N)!} \left(\frac{3b}{2}\right)^{2N} \sum_{k=0}^{+\infty} \frac{1}{(2n)!} \left(\frac{3b}{2}\right)^{2n} \tag{B.46}$$

$$ \leq \frac{1}{(2N)!} \left(\frac{3b}{2}\right)^{2N} \cosh\left(\frac{3b}{2}\right) \tag{B.47}$$

The proof for the other bounds is obtained following similar computation.

Appendix C. Expressions of $A_{i,j,k,l}$ in Section 5.3

The non-null matrices A_{ijkl} defined in (70) are given by

$$
\begin{aligned}
A_{2,2,0,0} &= a_0 a_0^T & A_{0,0,2,2} &= a_1 a_1^T \\
A_{0,2,2,0} &= a_2 a_2^T & A_{2,0,0,2} &= a_3 a_3^T \\
A_{2,1,0,1} &= a_0 a_3^T + a_3 a_0^T & A_{1,2,1,0} &= a_0 a_2^T + a_2 a_0^T \\
A_{0,1,2,1} &= a_1 a_2^T + a_2 a_1^T & A_{1,0,1,2} &= a_1 a_3^T + a_3 a_1^T \\
A_{1,1,1,1} &= a_0 a_1^T + a_1 a_0^T + a_2 a_3^T + a_3 a_2^T & A_{1,2,0,0} &= a_0 b_0^T + b_0 a_0^T \\
A_{0,2,1,0} &= a_2 b_0^T + b_0 a_2^T & A_{0,0,1,2} &= a_1 b_1^T + b_1 a_1^T \\
A_{1,0,0,2} &= a_3 b_1^T + b_1 a_3^T & A_{0,1,1,1} &= a_1 b_0^T + b_0 a_1^T + a_2 b_1^T + b_1 a_2^T \\
A_{1,1,0,1} &= a_0 b_1^T + b_1 a_0^T + a_3 b_0^T + b_0 a_3^T & A_{0,2,0,0} &= b_0 b_0^T \\
A_{0,0,0,2} &= b_1 b_1^T & A_{0,1,0,1} &= b_0 b_1^T + b_1 b_0^T
\end{aligned}
$$

with $a_0, a_1, a_2, a_3, b_0, b_1$ defined in (68)