A real-time unscented Kalman filter on manifolds for challenging AUV navigation

Théophile Cantelobre, Clément Chahbazian, Arnaud Croux, Silvère Bonnabel

A real-time unscented Kalman filter on manifolds for challenging AUV navigation

Théophile Cantelobre[1], Clément Chahbazian[2], Arnaud Croux[2], Silvère Bonnabel[3]

***Abstract*— We consider the problem of localization and navigation of Autonomous Underwater Vehicles (AUV) in the context of high performance subsea asset inspection missions in deep water. We propose a solution based on the recently introduced Unscented Kalman Filter on Manifolds (UKF-M) for onboard navigation to estimate the robot's location, attitude and velocity, using a precise round and rotating Earth navigation model. Our algorithm has the merit of seamlessly handling nonlinearity of attitude, and is far simpler to implement than the extended Kalman filter (EKF), which is widely used in the navigation industry. The unscented transform notably spares the user the computation of Jacobians and lends itself well to fast prototyping in the context of multi-sensor data fusion. Besides, we provide the community with feedback about implementation, and execution time is shown to be compatible with real-time. Realistic extensive Monte-Carlo simulations prove uncertainty is estimated with accuracy by the filter, and illustrate its convergence ability. Real experiments in the context of a 900m deep dive near Marseille (France) illustrate the relevance of the method.**

I. Introduction

Various industrial fields use Remotely Operated underwater Vehicles (ROV) to inspect critical subsea assets, at great cost. Thanks to recent progresses in telemetry and autonomous systems, Autonomous Underwater Vehicles (AUV) are currently being developed and deployed as a more cost-efficient and capable solution. One such example is Schlumberger's AUV project, displayed in Figure 1 and described in Section V-A. However, many challenges remain before ubiquitous adoption of AUVs in industrial use cases. One of the most significant challenges is guidance and navigation, as the capacity for the robot to localize itself and correctly sense its environment is pivotal for autonomy.

Given an Autonomous Underwater Vehicle, the goal of the present work is to estimate its position, velocity and attitude (kinematic state) as well as the associated confidence envelopes, in real-time and in challenging industrial conditions where great accuracy is required. Although most high grade inertial measurements units (IMU) come with built-in (proprietary) navigation softwares, it proves important for the roboticist to develop their own simple yet efficient navigation code to keep grip on the onboard navigation system, whether it be for fusion with other sensors, or in relation to other modules such as motion planning, or LiDAR-based mapping.

This work was supported by Schlumberger-Doll Research, Cambridge, MA (USA).

[1]Théophile Cantelobre is with INRIA and Mines ParisTech, PSL Research University, 60 bd Saint-Michel, 75006 Paris, France. Work undertaken during an internship at Schlumberger-Doll Research. theophile.cantelobre@mines-paristech.fr

[2]Clément Chahbazian and Arnaud Croux are with Schlumberger-Doll Research, 1 Hampshire St, Cambridge, MA, 02139. ACroux@slb.com

[3]Silvère Bonnabel is with Mines ParisTech, PSL Research University, Center for Robotics, 60 bd Saint-Michel, 75006 Paris, France; and University of New Caledonia, Institut ISEA, Noumea, New-Caledonia. silvere.bonnabel@mines-paristech.fr

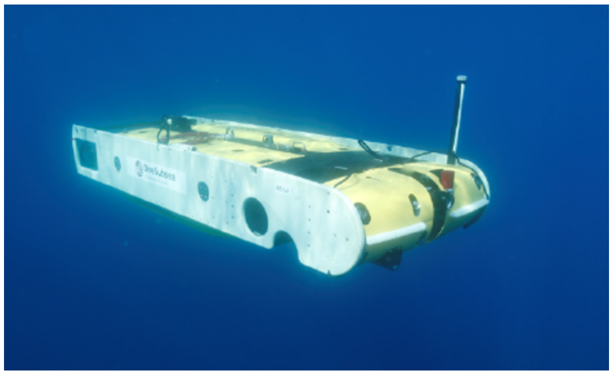
Fig. 1. Schlumberger's untethered AUV platform used in the experiments.

A. Motivations

In spite of half a century of experience with the Extended Kalman Filter (EKF), its implementation and tuning in the context of high precision inertial navigation still requires expertise, that can essentially be found inside companies that manufacture high end IMUs. Following the work of Julier and Ulhmann, [1], we believe an implementation based on the unscented Kalman filter (UKF) is much more desirable for the roboticist having limited experience in the field of inertial navigation. Indeed, resorting to the UKF spares the designer the computation of Jacobians, as required by the EKF methodology (and the more recent smoothing based methods [2]) which can be tedious, error-prone, and lack versatility (changing the model slightly or the sensors might require intensive re-computation).

Despite the EKF being well-established, it was shown to suffer from inconsistent uncertainty estimates, see for example [3], especially for fusion of IMU and vision. This is an important consideration for practicing roboticists, because a credible uncertainty estimate is crucial for filter performance, see e.g., Section IV-B.2 and Reference [4].

B. Key contributions

In the present work, to accommodate for the nonlinearity of the rotation matrices encoding the robot's orientation, we advocate the recently introduced UKF-M methodology, see [5]. The complete solution we present here handles both nonlinear state spaces and nonlinear dynamics, while remaining simple to implement and to grasp. Moreover it is shown to be compatible with the requirements of high performance real-time subsea inspection in deep water. Our contributions include:

1) Developing a complete Inertial Navigation System (INS) based on the UKF-M achieving high accuracy, consistency, and fast runtime performance required for industrial deep water AUV use cases;
2) Showing the relevance of combining a precise round-Earth kinematic model with the UKF-M in order to guarantee robustness to dead reckoning, which is essential during dive phases;
3) Developing an end-to-end navigation simulation framework using `ROS 2` to validate the performance of the filter in simulation, and with experimental data collected using the AUV platform shown in Figure 1;
4) The `ROS 2` implementation provides modularity: the same UKF-M code is seamlessly used for pure simulation, in "hardware in the loop" mode, and onboard as well. This approach allows speeding up the development phase and improves testing methods.

C. Relation to prior literature

The body of work devoted to ROV and AUV navigation is too broad to be covered here, see the surveys [6], [7]. However, solutions based on the UKF are scarce [6]. In [8], a modified dual UKF is used for AUV localization. However the application contrasts with our work owing to the use of a low-cost IMU: the localization accuracy is hence low, the navigation model is simplistic and does not account for round and rotating Earth effects, and experiments are conducted in shallow water (a test tank), where Euler angles are used, which results in singularites [9]. The recent work [10] presents an AUV based navigation system that utilizes the UFK, and proves that the UKF has superior performance to the EKF. However, the method and the context drastically differ from ours. First, the dynamics rely on a simplified physical model of the robot based on the rotational speed of the motors related to the delivered thrusts which by nature is too approximate to compete with high grade IMUs. This model is completed with Xsens MTI IMU which is of much lower grade than ours, and the orientation estimated by the Xsens proprietary algorithm is used in the UKF, which is undesirable at two levels: 1) we want to keep control over the entire estimation pipeline, and 2) the proprietary algorithm delivers highly suboptimal angles estimation as it does not correlate the IMU with other available sensors such as GPS, DVL, USBL, and depth, inevitably leading to degraded performance. Finally, the tests are performed at a few meters depth.

The rest of this paper is organized as follows. In Section II, we present the round-Earth system model we use in our filter. In Section III, we present our proposed UKF-M algorithm and discuss its advantages over the industry standard, the EKF. In Section IV, we present the simulation pipeline and then demonstrate our algorithm's performance using Monte Carlo simulations using accuracy and consistency metrics. In Section V, we show that the developed filter can run in real-time using experimental data from the AUV platform.

II. System model and sensor noise models

In this section, we first present a precise, round-Earth navigation model and then show how we can discretize it accurately, along with models of the vehicle's sensors.

A. Accurate Navigation Model

The retained state for the AUV consists of latitude, longitude and ellipsoidal altitude (L, λ, h) to describe position, and navigation frame-relative velocity v_{en}^n and attitude C_b^n. Our model follows methods from the field of inertial navigation and does not rely on any particular modelling of the vehicle and its dynamics, i.e., only relies on the IMU signals. Moreover, given the depth at which we aim to dive and the overall length of the mission, we anticipate that the filter accumulates a large amount of drift during the dive phase, see Figure 3. As a result, we combine a high performance inertial measurement unit (IMU) and a navigation model that accounts for Earth rotation, curvature, and vertical gradient and change in direction of the gravitational field.

1) Continuous-Time Navigation Equations: Following the derivation in [11], state evolution equations can be written as:

$$\dot{L} = \frac{v_{en\,N}^n}{R_N(L) + h}, \tag{1}$$

$$\dot{\lambda} = \frac{v_{en\,E}^n}{(R_E(L) + h)\cos(L)}, \tag{2}$$

$$\dot{h} = -v_{en\,D}^n, \tag{3}$$

$$\begin{aligned}\dot{v}_{en}^n &= C_b^n f_{ib}^b - g_b^n(L, h) \\ &\quad - \big(\Omega_{en}^n(L, v_{en}^n) + 2\Omega_{ie}^n(L)\big) v_{en}^n,\end{aligned} \tag{4}$$

$$\dot{C}_b^n = C_b^n \Omega_{ib}^b - \big(\Omega_{ie}^n(L) + \Omega_{en}^n(L, v_{en}^n)\big) C_b^n, \tag{5}$$

where Ω_{ib}^b and f_{ib}^b are the rotation rate and acceleration from the inertial frame to the body frame. The latter quantities are measured (up to bias and noise) by the gyroscopes and the accelerometers of the IMU, respectively. Let us detail our notation which emphasizes the dependency of Earth model quantities upon state variables.

$g_b^n(L, h)$: model of gravity vector at the vehicle's position. The dependency in L and h is expected and due to a model of the Earth's shape, and of Earth's gravitational force decreasing with altitude, respectively.

$\Omega_{ie}^n(L)$: model of the Earth's rotation vector at the vehicle's position. The magnitude of the vector is constant, but its direction depends on latitude L.

$\Omega_{en}^n(L, h, v_{en}^n)$: model of the vehicle's transport rate, i.e. the rotation caused by the movement of the tangent navigation frame with respect to the Earth's surface, because of the vehicle's movement. The transport rate thus depends on the curvature at its position and thus on L and h, its latitude and height.

Thus, our model aims to capture the full nonlinearities of the system, without any first-order approximations.

2) Model Discretization: We now derive the discrete time equations from the continuous-time model above in view of our filter's implementation. The inputs of this discrete propagation step are thus: the previous state, the body frame specific force f_{ib}^{b} returned by accelerometers, the rotation rate ω_{ib}^{b} returned by gyroscopes (both up to bias and noise), and the time step Δt. From our experience, the key insight brought by [11] is to integrate the equations in an order that minimizes computations as variables computed can be reused in the next equation. Experimentally, we have found the following order to be the best.

In what follows, for each quantity z we denote $z(-)$ the current value, $z(+)$ the propagated value. We denote $\alpha_{ib}^{b} = \omega_{ib}^{b}\Delta t$ and $\alpha = ||\alpha_{ib}^{b}||$[1].

a) Velocity: First, transform the specific force from the body to the navigation frame $f_{ib}^{n} = \bar{C}_{b}^{n} f_{ib}^{b}$ using an intermediate frame transformation

$$\bar{C}_{b}^{n} \overset{\text{def}}{=} C_{b}^{n} C_{\bar{b}}^{b-}(\alpha_{ib}^{b}) - \frac{1}{2}\left(\Omega_{ie}^{n}(-) + \Omega_{en}^{n}(-)\right) C_{b}^{n}(-)\Delta t$$

Next, calculate $v_{en}^{n}(+)$ in two steps, including the transport and Coriolis terms.

$$\begin{aligned} v_{en}^{n}(+) = v_{en}^{n}(-) &+ \Delta t f_{ib}^{n} + \Delta t g_{b}^{n}(L(-), h(-)) \\ &- \frac{\Delta t}{2}(\Omega_{en}^{n}(-) + 2\Omega_{ie}^{n}(-)) v_{en}^{n}(-) \\ &- \frac{\Delta t}{2}(\Omega_{en}^{n}(v_{en}^{n\prime}) + 2\Omega_{ie}^{n}(-)) v_{en}^{n\prime} \end{aligned} \tag{6}$$

where $v_{en}^{n}{}'$ is a straightforward integration of Equation (4) using f_{ib}^{n}.

b) Longitude, latitude and height: from there, position is computed using a second order integration method.

$$h(+) = h(-) - \frac{\Delta t}{2}\left(v_{en\,D}^{n}(-) + v_{en\,D}^{n}(+)\right) \tag{7}$$

$$\begin{aligned} L(+) = L(-) &+ \frac{\Delta t}{2}\frac{v_{en\,N}^{n}(-)}{R_N(L(-)) + h(-)} \\ &+ \frac{\Delta t}{2}\frac{v_{en\,N}^{n}(+)}{R_N(L(-)) + h(+)} \end{aligned} \tag{8}$$

$$\begin{aligned} \lambda(+) = \lambda(-) &+ \frac{\Delta t}{2}\frac{v_{en\,E}^{n}(-)}{(R_E(L(-)) + h(-))\cos(L(-))} \\ &+ \frac{\Delta t}{2}\frac{v_{en\,E}^{n}(+)}{(R_E(L(+)) + h(+))\cos(L(+))} \end{aligned} \tag{9}$$

c) Attitude: Finally, attitude is propagated using the previously computed quantities.

$$\begin{aligned} C_{b}^{n}(+) = &\left(I_3 - \Delta t\left(\Omega_{ie}^{n}(-) - \frac{\Omega_{en}^{n}(-) + \Omega_{en}^{n}(+)}{2}\right)\right) \\ &\times C_{b}^{n}(-) C_{b+}^{b-}(\alpha_{ib}^{b}) \end{aligned} \tag{10}$$

[1]See [11] for other notations.

Note that this succession of equations defines a map f such that if χ is the kinematic state and u the IMU inputs, $\chi_{n+1} = f(\chi_n, u_n, \Delta t)$.

B. Sensor Errors

Much of the uncertainty in the model above stems from the presence of noise in the IMU's signals. However, IMU sensor noises are very difficult to model. For this reason, only *a posteriori* noise models exist, including [12], [13]. Based on manufacturer calibration information and an Allan variance analysis [14] of our hardware, we concluded that a biased, white noise model with covariance matrix Q_{in} we could compute from the specifications is sufficient, even for hours long missions.

We made similar considerations for the other sensors used in the mission, that is, Global Positioning System (GPS) at the surface, Doppler Velocity Log (DVL), depth sensor, and acoustic Ultra Short Base Line (USBL) positioning system at the bottom.

III. Unscented Kalman Filter on Manifolds for AUV Navigation

Given a kinematic model and noisy sensor outputs, our goal is thus to fuse all the prior information we have about our system (the kinematic and sensor models) with the sensor data the robot measures. In our use case, we are faced with two problems that make it difficult to apply the commonplace Extended Kalman Filter (EFK).

1) Rotation-error handling: our state representation does not lie in a vector space, but on a manifold owing to the presence of rotation matrices (or quaternions) that encode the attitude C_{b}^{n}.
2) Complexity of the model: because we take into consideration an Earth and gravity model, and because of the precise integration scheme, deriving the Jacobian of f as required in the EKF methodology is complex and error-prone. Moreover, the slightest model modification requires to re-calculate Jacobians.

For these two reasons, we resort to the method proposed in [5], the Unscented Kalman Filter on Manifolds, as the back-bone of our data-fusion algorithm. The latter is a recent variant of the UKF as introduced by Julier and Uhlmann in [15], which accommodates for the nonlinear structure of the model and of the state space, owed to the presence of rotation matrices. In this section, we describe the algorithm, the design choices we made and how it handles the two issues described above.

A. Application of UKF-M to the present problem

In order to use the machinery of UKF-M to accommodate rotation matrices, we naturally define the manifold in which the state $\chi = (L, \lambda, h, v_{en}^{n}, C_{b}^{n})$ lives as $\mathcal{M} = \mathbb{R}^6 \times \mathcal{SO}_3(\mathbb{R})$. This choice is also relevant because $\mathcal{M}$ has a trivial Lie group structure, see [16]. The second ingredient of the method is the choice of a retraction $\varphi : \mathcal{M} \times \mathbb{R}^9 \rightarrow \mathcal{M}$ that we define as

$$\varphi(\chi,\xi) = \begin{pmatrix} L+\xi_1 \\ \lambda+\xi_2 \\ h+\xi_3 \\ v_{en}^n+\xi_{4:6} \\ C_b^n\exp(\xi_{7:9}) \end{pmatrix} \tag{11}$$

which leads us to define $\varphi_\chi(\tilde{\chi})^{-1} \overset{\text{def}}{=} \big(\tilde{\chi}_{1:6} - \chi_{1:6}, \log_{SO(3)}(\chi_{7:9}^{-1}\tilde{\chi}_{7:9})\big)$, where we have defined the $\exp_{SO(3)}$ and $\log_{SO(3)}$ maps on $\mathcal{SO}_3(\mathbb{R})$ as in [17]. This allows one to define an UKF despite the fact the state space is a manifold which includes rotation matrices. Indeed, the key observation is that $\varphi_\chi^{-1}(\tilde{\chi}) \in \mathbb{R}^9$ defines a (vector) error between state $\tilde{\chi}$ and state χ. Note that, recent works have advocated the use of a more sophisticated Lie group for navigation (and hence a different map φ), in the context of invariant filtering [18]. The potential benefits of such a choice are left for future work, though.

B. Uncertainty Representation

In the UKF-M methodology, the statistical belief about the state is represented as $\varphi(\chi,\xi)$ with χ the mean or average estimate, and $\xi \sim \mathcal{N}(0,P)$ a centered Gaussian with covariance matrix P that reflects statistical dispersion, and where φ is the chosen retraction, in our case (11). This may be related to the notion of concentrated Gaussian, see [19], [20], when the state is a Lie group.

C. Proposed Algorithm

We now present our application of the UKF-M algorithm and how it handles model nonlinearities in a derivative-free manner. The main idea behind the UKF-M is to use sigma-points as in the UKF methodology [15] to capture the nonlinearities in the model. These sigma-points statistically linearize the model instead of using an analytic expression of the Jacobian, as required by the EKF. The UKF-M iterates the two following steps over time: (i) propagation (of the distribution through the dynamics), and (ii) update (re-estimation of the distribution in the light of observations).

1) UKF-M: Propagation step: In the UKF-M framework, the mean χ is propagated directly through the system's equations described in Section II-A.2, as in the standard EKF and in contrast with the canonical UKF methodology. The motivations are as follows: (i) because they do not lie on a vector space, averaging rotation matrices would greatly increase the complexity of the algorithm [21], (ii) with mean propagation separate from covariance propagation in dead reckoning, the filter is more stable. Indeed, in dead reckoning, covariance divergence occurs due to a corollary of the Schuler effect [11], and in turn degrades the mean propagation if the canonical UKF methodology is used.

By contrast, to propagate the uncertainty estimate, sigma points over state and input variables are used. This presents three advantages: (i) it captures second order terms [22], (ii) it avoids error-prone Jacobian calculations [1], and (iii) it eliminates the difficult to compute (or arbitrary [1]) propagation noise matrix, essentially eliminating a matrix hyperparameter. The UKF-M propagation step is summarized in Algorithm 1.

Algorithm 1: UKF-M: propagation step

```
Input      : $\chi_n$, $P_n$, $u_n$, $\Delta t$, $Q_{in}$
Parameters: $\alpha$, $d = \dim(\chi_n)$, $\lambda(d) = d(\alpha^2 - 1)$
// Mean propagation
$\chi_{n+1}^+ = f(\chi_n, u_n, \Delta t)$ with $f$ from Section II-A.2;
// State Sigma-point calculation
$\xi_i = col_i(\sqrt{d+\lambda(d)}\sqrt{P_n})$, $1 \le i \le d$;
$\xi_{d+i} = -col_i(\sqrt{d+\lambda(d)}\sqrt{P_n})$, $1 \le i \le d$;
// State Covariance propagation
$\varepsilon_i = \varphi_{\chi_{n+1}^+}^{-1}\big(f(\varphi(\chi_n,\xi_i))\big)$, $1 \le i \le 2d$;
$P_{n+1}^s = \sum_{i=1}^{2d} w_i\varepsilon_i\varepsilon_i^T$;
// Noise Sigma-point calculation
$\xi_i' = col_i(\sqrt{6+\lambda(6)}\sqrt{Q_{in}})$, $1 \le i \le 6$;
$\xi_{6+i}' = -col_i(\sqrt{6+\lambda(6)}\sqrt{Q_{in}})$, $1 \le i \le 6$;
// Noise Covariance propagation
$\varepsilon_i' = \varphi_{\chi_{n+1}^+}^{-1}\big(f(\varphi(\chi_n,\xi_i))\big)$, $1 \le i \le 12$;
$P_{n+1}^n = \sum_{i=1}^{12} w_i\varepsilon_i'\varepsilon_i'^T$;
Result: $\chi_{n+1}^+$, $P_{n+1}^+ = P_{n+1}^s + P_{n+1}^n$
```

2) UKF-M: update step: With the introduction of sensor noise from the IMU, the uncertainty obtained during propagation (called *dead reckoning*) grows over time. This can be observed, for example, in Figure 3. Algorithm 2 describes how the UKF-M refines the state estimate χ and associated uncertainty P in the light of measurements. In the following, observations are assumed to be of the following form:

$$\forall t \ge 0, \quad y(t) = h\big(\chi(t)\big) + n(t) \tag{12}$$

where h is a known map and n is white noise. This may represent DVL, acoustic positioning, and more generally all readings from sensors of the AUV.

As is usual in data fusion, the state vector can be extended to include parameters such as model biases. This allows easy calibration of IMU biases, for example, see Figure 5.

D. Numerical Aspects

While less accurate and consistent than the UKF [3][22], the EKF still prevails in the navigation industry because the former is considered as more challenging numerically.

1) Runtime performance: A straightforward application of the UKF methology implies running Algorithm 1 at high frequency (typically 100 Hz), with $2d+1$ forward model evaluations and a matrix square root computation. This is often considered prohibitive on standard hardware. However, we show that in the context of industrial-grade navigation systems for AUVs, the UKF-M is a feasible solution. We observed a $40\times$ margin compared to real-time, with full state updates. Our runtime evaluations are summarized in Table I. The experiments were executed on a Intel Core i7-8850H CPU (2.60GHz) akin to embedded Intel NUC boards. We believe it is a contribution of this work to the robotics community to have shown that the UKF-M's simpler

Algorithm 2: UKF-M: update step

```
Input      : $\chi^+_{n+1}$, $P^+_{n+1}$, $u_{n+1}$, $h$
Parameters: $\alpha$, $d = \dim(\chi_n)$, $\lambda(d) = d(\alpha^2 - 1)$
// Sigma-point calculation
$\xi_i = col_i(\sqrt{d+\lambda(d)}\sqrt{P^+_{n+1}})$, $1 \leq i \leq d$;
$\xi_{d+i} = -col_i(\sqrt{d+\lambda(d)}\sqrt{P^+_{n+1}})$, $1 \leq i \leq d$;
// Mean and sigma-point prediction
$y_0 = h(\chi^+_{n+1})$;
$y_i = h(\varphi(\chi^+_{n+1}, \xi_i))$, $1 \leq i \leq 2d$;
// Mean calculation
$\hat{y} = \sum_{i=0}^{2d} w_i^s y_i$;
// Covariance calculation
$P_{yy} = \sum_{i=0}^{2d} w_i^c (y_i - \hat{y})(y_i - \hat{y})^T$;
// Cross-covariance calculation
$P_{\chi y} = \sum_{i=1}^{2d} w_i^c \xi_i (y_i - \hat{y})^T$;
// Covariance update
$K = P_{\chi y} P_{yy}^{-1}$;
$P_{n+1} = P^+_{n+1} - K P_{yy} K^T$;
// Mean update
$\chi_{n+1} = \varphi(\chi^+_{n+1}, K(y_{n+1} - \hat{y}))$;
Result: $\chi_{n+1}$, $P_{n+1}$
```

implementation can actually be leveraged in practice without sacrifying runtime performance.

TABLE I
RUNTIME PERFORMANCE.

Mission type	Mission length (s)	f_s IMU (s)	Runtime (s)
Full navigation	1945	100 Hz	45.6
Full navigation	1945	10 Hz	4.6
Propagation only	1945	100 Hz	32.4
Propagation only	1945	10 Hz	3.3

2) Rotation matrix stability: In the navigation and aeronautics literature and industries, it is customary to use quaternions in place of rotation matrices[2]. However, the use of rotation matrices is common among roboticists. We found that by applying Gram-Schmidt normalization to C_b^n when necessary, our performance was indistinguishable from that of a quaternion-based system.

IV. Realistic Numerical Experiments

In AUV applications, especially subsea asset inspection in the Oil & Gas industry where robots may operate in deep water, ground truth is very costly - if not impossible - to obtain, especially during the design stage. We thus made extensive use of a simulation framework built on the `ROS 2` [24] middleware to develop and test the algorithm. This allows us to assess accuracy, consistency, and runtime performance of the filter.

[2]For an UKF implementation of attitude-only estimation using quaternions, see the USQUE algorithm in [23].

A. Simulation pipeline

As part of the design and the implementation of the navigation algorithm we presented above, we developed an end-to-end simulation pipeline. From a concise description of the trajectory, it simulates the vehicle's sensors and, using `ROS 2` as a middleware, runs the navigation algorithm against this data. Our simulation framework is composed of `ROS 2` nodes, and can be presented in four layers:

1) Ground truth trajectory generation, based on the open-source `GNSS INS Sim` package[3];
2) Sensor simulation nodes, which use the ground truth trajectories to generate sensor output, taking into consideration sensor visibility aspects;
3) UKF-M node, which estimates the vehicle's state and associated uncertainty from simulated sensor output;
4) Finally, real-time visualization and diagnostics relying on a `Postgres` database interfaced with a `Grafana` dashboard.

This approach has several advantages including: (i) it is modular and interactive: it can easily be extended to include new sensors, physical model or vehicle behaviors or decision making capabilities; (ii) it simulates many of the challenges of real-time navigation, such as delays and message loss; (iii) it is compatible with Hardware-in-the-loop simulation in a *Digital avatar* in a lab or testing onboard a vehicle, such as onboard Schlumberger's Sabertooth platform described in Section V-A.

B. Monte Carlo Experiments and Evaluation Metrics

In order to rigorously evaluate our model and our algorithm, we simulated the navigation system presented above on a realistic inspection-type mission, using Monte Carlo simulations. We draw N different noise realizations over the input data (IMU, depth sensor, GPS, DVL and USBL) and run the algorithm against these N realizations. The output of the different runs $\chi^i(t)$ allow us to evaluate the algorithm's performance according to two criteria: its accuracy and its consistency. These simulations were run on a variant of the above pipeline compatible with parallel simulation, using GNU Parallel [25].

The metrics we use to assess filter's performance are:

1) Accuracy through Standard Deviation: We measure the dispersion of the different runs, i.e. the risk we take by accepting to use one of the trajectories as our mean trajectory. For this, we denote $\sigma_N(t)$ the standard deviation of the different runs' estimates at time t.

2) Consistency through Normalized Estimation Error Squared (NEES): One of our algorithm's desirable properties is the quality of its uncertainty estimate. This is paramount as it impacts the quality of the estimation later during the mission for the two following reasons. First, as presented by Y. Bar-Shalom, in [4], "wrong covariances yield wrong gains", which in turn may degrade performance. This is especially true in navigation, where it is customary to propagate at high frequency (typically at $f_s^{IMU} = 100$ Hz) and update

[3]`https://github.com/Aceinna/gnss-ins-sim`

less frequently (typically at $f_s^{GPS} = 1$ Hz). Second, the uncertainty estimate P is a quantity that can be used as input to other algorithms, especially for motion planning and resulting decision making on board the vehicle (for example, to safely approach subsea infrastructure during an inspection mission). Finally, uncertainty may also be used when correlating feeds from imaging sensors [26], and our AUV is also equipped with LiDAR for survey purposes.

We define the Normalized Estimation Error Squared (NEES) for each trajectory $\chi^i(t)$ and uncertainty estimate $P^i(t)$, comparing the estimated uncertainty to the estimation error [4][27]. Formally, if $\xi^G(t)$ is the ground truth trajectory and $\xi^i(t)$ is an estimated trajectory with associated uncertainty $P^i(t)$, then $\varepsilon^i(t) = \varphi^{-1}_{\chi^G(t)}\big(\chi^i(t)\big)$ is the estimation error. The NEES is defined by:

$$\nu^i(t) \overset{\text{def}}{=} \frac{1}{d}\varepsilon^i(t)^T P^i(t)^{-1}\varepsilon^i(t) \tag{13}$$

We can then define the Average NEES (ANEES) over the different trajectories, $\nu(t) \overset{\text{def}}{=} \frac{1}{N}\sum_{k=1}^{N}\nu^i(t)$.

To ensure we have a reliable uncertainty estimate, $\nu(t)$ should be around 1: if $\nu > 1$, the uncertainty is overestimated; otherwise, it is underestimated.

TABLE II
SENSOR VISIBILITY ON INSPECTION MISSION

Phase	IMU	GPS	Depth	DVL	USBL
Calibrate	✓	✓	✗	✗	✗
Dive	✓	✗	✓	✗	✗
Survey	✓	✗	✓	✓	✓

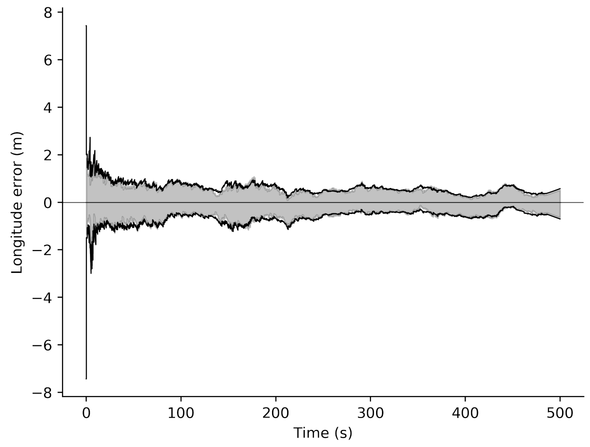

Fig. 2. Longitude error (m) during the calibration phase. Despite the calibration process, the estimates are stable and the uncertainty well-estimated.

C. Simulation Mission Design

We evaluated our model and algorithm against a realistic inspection mission, a dive and survey mission (DAS). The

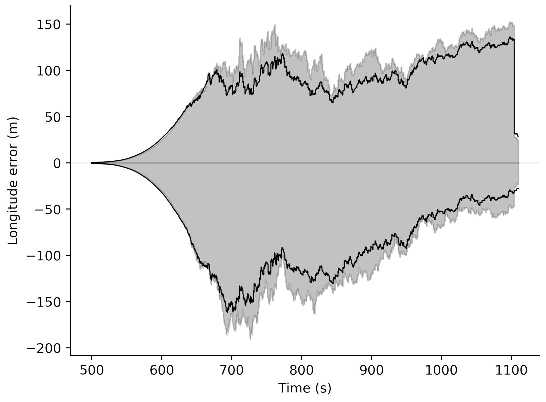

Fig. 3. Longitude error (m) during the dive phase. Note that the uncertainty in the estimate is well predicted by $\sqrt{P(t)}$.

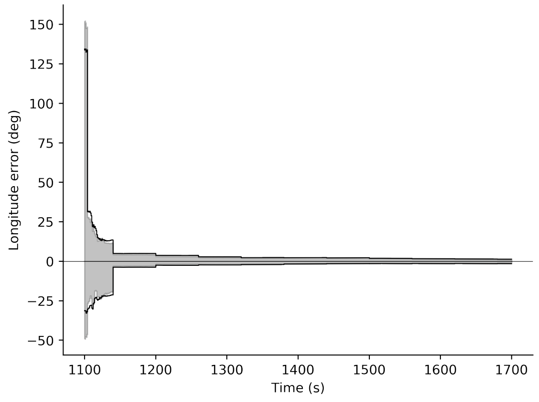

Fig. 4. Longitude error (m) during the survey phase after large drift has been accumulated in the dead reckoning dive phase. This illustrates the filter's convergence properties owed to the consistent uncertainty estimation during dead reckoning.

mission can be divided into three successive parts described below. Conservative hypotheses are assumed with respect to sensor visibility and performance, summarized in Tables II and III respectively. These characteristics reflect those of the robot displayed in Figure 1.

1) Surface calibration: At the surface, the vehicle calibrates its IMU biases by executing figure-eights, with only GPS visible. Note that no special behavior is imparted to the filter during this calibration phase, it runs identically to the other phases.

2) Dive: Dive from the surface to 1200 meters depth at a 80 degree incline. Only the depth sensor is available.

3) Survey: At 1200 meters depth, level and execute a high-altitude survey over equipment in a lawnmower pattern. Depth, DVL and USBL sensors are then all available.

D. Results

To analyze the performance of our framework, we use the two metrics defined in Section IV-B:

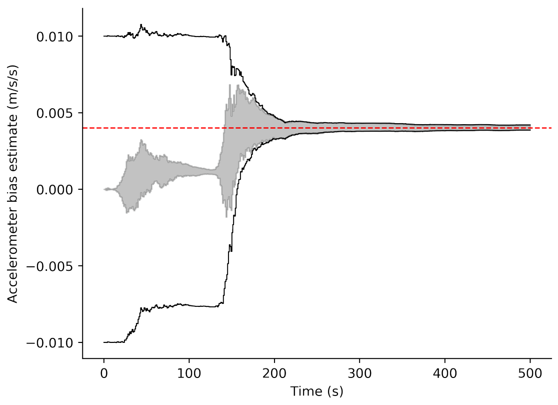

Fig. 5. Accelerometer bias estimate (m/s^2) during the calibration phase. The horizontal red dashed line is the true bias parameter $0.004 m.s^{-2}$, and we see it is fully recovered. Note that the convergence $\sqrt{P(t)}$ to 0 can be used to mark the end of the calibration phase.

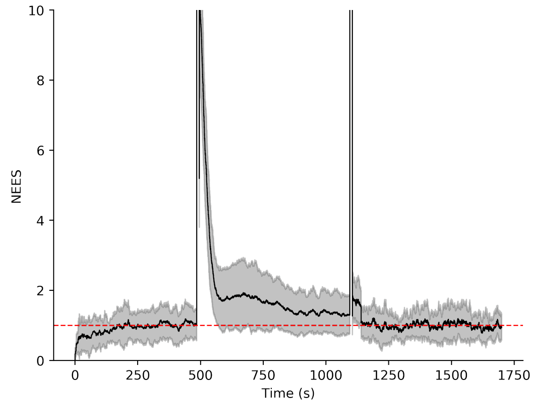

Fig. 6. NEES over the mission. At the beginning of the mission, uncertainty is overestimated but the NEES rapidly converge towards 1. At each change in navigation regime, the NEES becomes very large for a few seconds, the time for the filter to converge. It quickly converges back to 1 (red dashed line). The black curve is the ANEES and the grey envelope the dispersion of the NEES trajectories as defined in Section IV-B.2, over $N = 48$ runs.

- to evaluate consistency: in Figure 6, we compare $\nu(t)$ to 1 and verify that the spread of the $\nu^i(t)$ are reasonable;
- to evaluate accuracy: in Figures 2-4, we show the $1{-}\sigma_N(t)$ envelope for the different estimated trajectories, and compare them to our estimated uncertainty $\sqrt{P^i(t)}$[4]. In these figures, the grey envelope is between $\pm\sigma_N(t)$ and the black lines are the estimated covariance $\sqrt{P(t)}$, for $N = 48$ runs.

The results illustrate several desirable properties of the algorithm:

1) Overall performances: Figures 2-6 show that our filter achieves metric accuracy and is consistent, on a realistic deep water inspection-type mission, including an (auto) calibration phase where IMU biases are recovered.

2) Robustness to dead-reckoning: Figure 3 shows that the accumulated dispersion is only 200 m after 15 minutes of dead reckoning of diving.

3) Convergence property: Figure 4 illustrates filter convergence: the filter recovers the correct position thanks to DVL and USBL updates, despite dispersion incurred during dead reckoning dive. This also emphasizes the importance of a consistent uncertainty estimate which yields correct Kalman gains. Note that these results are obtained using nominal noise models, without so-called "robust learning" or "stabilizing noise" [1] which helps the filter to converge and combat the nonlinearities but ultimately degrades performance, since it is based on erroneous noise statistics.

4) Sensor bias integration: Figure 5 shows how sensor biases can be observed with great confidence executing a simple maneuver during a short period of time. More importantly, one can use the convergence of the bias uncertainty (the $\sqrt{P(t)}$ envelope) to detect the bias estimate's convergence.

5) Consistency: Finally, Figure 6 shows that the ANEES is reliably close to 1 during the mission, except when there are transitions in sensor visibility. The ability of the filter to very accurately convey its uncertainty is key notably for motion planning during inspection and maintenance.

TABLE III
SENSOR CHARACTERISTICS

Sensor	f_s (Hz)	Noise model	Bias model
Gyroscope	100 Hz	$0.012\ deg/\sqrt{hr}$	2 deg/hr
Accelerometer	100 Hz	$0.0141\ m.s^{-1}.hr^{-1/2}$	$4.10^{-3} m/s^2$
Depth	1 Hz	$1.0\ m$	-
GPS	1 Hz	$2\ m$	-
DVL	1 Hz	$5\ mm/s$	-
USBL	1/60 Hz	$5\ m$, $5\ m$, $0.1\ m$	-

V. Experimental Validation

In order to show that the INS presented in this work can be used onboard an AUV, we evaluated it against data collected in the field by Schlumberger with the AUV displayed in Figure 1 and described below. Our experiments show that without algorithm adaptation, we obtain a credible estimate of the AUV's position.

A. Experimental Platform & Data Collection

The platform used to collect the data is an untethered AUV, developed by Schlumberger within a wider multi-agent autonomy system for subsea asset inspection. Based on Saab's Sabertooth platform, the AUV features a standard underwater navigation sensor suite including: a high-grade IMU, Doppler Velocity Log (DVL), depth sensor, GPS, and Ultra-Short Baseline (USBL) acoustic postioning system. Sensors' characteristics are summarized in Table III.

The AUV was used to capture IMU, GPS, DVL, USBL and depth data from the sensor suite in order to evaluate the framework presented above. The mission was executed off

[4]Note that although the $P^i(t)$ are date-dependent, in practice their dispersion is negligible, so any $P^i(t)$ can be used.

the coast of La Ciotat (France) and is similar to the one used for simulation. In particular, it includes a dive as well as a survey at around 900 meters depth.

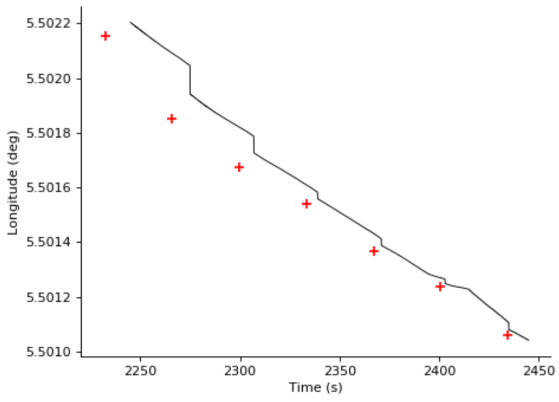

Fig. 7. Estimated trajectory on experimental data captured using Schlumberger's AUV. USBL used in the update step are materialized (red crosses).

B. *Results & Discussion*

Runnning against this data, we showed that our INS system was capable of reproducing the performance achieved in simulation, in real-time, and with comparable performance to the simulation results.

Figure 7 shows the longitude estimate over an extract of the estimated trajectory, during the survey phase at the around 900 meters depth. During that segment, the IMU, DVL, and USBL sensors are available. In Figure 7, USBL datapoints are represented by red crosses. They show that despite the filter being initialized erroneously, the filter converges through USBL and DVL updates.

VI. Conclusion

Our work shows that our UKF-M-based navigation solution can be used in real-time, while attaining high accuracy and consistency on challenging, sensor-denied missions. Furthermore, we showed that it is robust to dead-reckoning and to sensor biases. These results have been shown to hold in simulation and on experimental data from various sensors (high-grade IMU, DVL, and USBL) in the context of a real, challenging 900 m deep mission.

In future work we plan to (1) integrate perception-based updates to the navigation system, and (2) develop manoeuvres to re-calibrate underwater.

References

[1] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proc. IEEE*, vol. 92, pp. 401–422, Mar. 2004.

[2] M. F. Fallon, M. Kaess, H. Johannsson, and J. J. Leonard, "Efficient AUV navigation fusing acoustic ranging and side-scan sonar," in *2011 IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 2398–2405, IEEE, May 2011.

[3] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of extended Kalman filter based SLAM," in *2008 IEEE International Conference on Robotics and Automation*, (Pasadena, CA, USA), pp. 473–479, IEEE, May 2008.

[4] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, USA: John Wiley & Sons, Inc., 2001.

[5] M. Brossard, A. Barrau, and S. Bonnabel, "A Code for Unscented Kalman Filtering on Manifolds (UKF-M)," in *2020 International Conference on Robotics and Automation (ICRA)*, IEEE, 2020.

[6] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A SURVEY OF UNDERWATER VEHICLE NAVIGATION: RECENT ADVANCES AND NEW CHALLENGES," p. 12.

[7] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV Navigation and Localization: A Review," *IEEE J. Oceanic Eng.*, vol. 39, pp. 131–149, Jan. 2014.

[8] G. C. Karras, S. G. Loizou, and K. J. Kyriakopoulos, "On-line state and parameter estimation of an under-actuated underwater vehicle using a modified Dual Unscented Kalman Filter," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Taipei), pp. 4868–4873, IEEE, Oct. 2010.

[9] P. Allgeuer and S. Behnke, "Fused angles: A representation of body orientation for balance," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Hamburg, Germany), pp. 366–373, IEEE, Sept. 2015.

[10] B. Allotta, A. Caiti, R. Costanzi, F. Fanelli, D. Fenucci, E. Meli, and A. Ridolfi, "A new AUV navigation system exploiting unscented Kalman filter," *Ocean Engineering*, vol. 113, pp. 121–132, Feb. 2016.

[11] P. D. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. GNSS technology and application series, Boston: Artech House, 2nd ed ed., 2013. OCLC: ocn820530994.

[12] R. J. Vaccaro and A. S. Zaki, "Statistical Modeling of Rate Gyros," *IEEE Trans. Instrum. Meas.*, vol. 61, pp. 673–684, Mar. 2012.

[13] "IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros," tech. rep., IEEE, 1997.

[14] D. Allan, "Time and Frequency (Time-Domain) Characterization, Estimation, and Prediction of Precision Clocks and Oscillators," *IEEE Trans. Ultrason., Ferroelect., Freq. Contr.*, vol. 34, pp. 647–654, Nov. 1987.

[15] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI* (I. Kadar, ed.), vol. 3068, pp. 182 – 193, International Society for Optics and Photonics, SPIE, 1997.

[16] M. Brossard, S. Bonnabel, and J.-P. Condomines, "Unscented Kalman filtering on Lie groups," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Vancouver, BC), pp. 2485–2491, IEEE, Sept. 2017.

[17] T. D. Barfoot, *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017.

[18] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Trans. Automat. Contr.*, vol. 62, pp. 1797–1812, Apr. 2017.

[19] Yunfeng Wang and G. Chirikjian, "Error propagation on the Euclidean group with applications to manipulator kinematics," *IEEE Trans. Robot.*, vol. 22, pp. 591–602, Aug. 2006.

[20] G. Bourmaud, R. Mgret, M. Arnaudon, and A. Giremus, "Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions," *J Math Imaging Vis*, vol. 51, pp. 209–228, Jan. 2015.

[21] M. Moakher, "Means and Averaging in the Group of Rotations," *SIAM J. Matrix Anal. & Appl.*, vol. 24, pp. 1–16, Jan. 2002.

[22] F. Gustafsson and G. Hendeby, "Some Relations Between Extended and Unscented Kalman Filters," *IEEE Trans. Signal Process.*, vol. 60, pp. 545–555, Feb. 2012.

[23] J. L. Crassidis and F. L. Markley, "Unscented Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 26, pp. 536–542, July 2003.

[24] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," p. 6.

[25] P. D. Hanlon and P. S. Maybeck, "GNU Parallel - The Command-Line Power Tool," *login: The USENIX Magazine*, no. 2, pp. 42–47, 2011.

[26] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU Dead-Reckoning," *arXiv:1904.06064 [cs, stat]. Accepted in IEEE Transactions on Intelligent Vehicles.*, Apr. 2020. arXiv: 1904.06064.

[27] X. R. Li, Z. Zhao, and V. P. Jilkov, "Practical Measures and Test for Credibility of an Estimator,"