AUV Localization with Right InEKF with AHRS

Brandon Chao, Arnav Mummineni, and Uygur Tepe
University of Michigan Ann Arbor, MI
{bchao, arnavmu, tepe}@umich.edu

***Abstract*— Advancements in the utilization of Lie Groups for robotic localizations have led to an increase in the accuracy of estimation. One of these techniques, Invariant Extended Kalman Filter (InEKF), extends the Extended Kalman Filter (EKF) by leveraging that some error dynamics defined by matrix Lie Groups satisfy a log-linear differential equation. In this paper, we show that utilizing the primary sensors for underwater vehicles, inertial measurement units (IMUs), Doppler velocity logs (DVLs), and depth singleton measurements are able to perform well with an InEKF but have greater potential with the addition of an attitude and heading reference system (AHRS). The open-souce code and video presentation of the results can be found at the following link: `https://github.com/EECS-568-Final-Project/AUV-InEKF`**

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) have many uses and applications. In this paper, we apply them to a yearly collegiate AUV competition called RoboSub. The competition includes several tasks that vary from year to year. Most of these tasks require precise navigation, which can benefit from localization techniques. Notably, underwater localization is a challenging problem because traditional land-based sensors such as GPS, LiDAR, etc. are unavailable. As such, it restricts the different options available for localization.

In this paper, we demonstrate how the Invariant Extended Kalman Filter (InEKF), with the addition of an AHRS sensor, can be used to improve upon current methods for underwater localization.

II. PRIOR WORK

Localization in AUV presents unique challenges due to the unavailability of GPS and the limitations of visual sensors in underwater environments. Classic EKF approaches typically rely on sensor fusion to combine data from multiple sources, such as IMUs, DVLs, and depth sensors. While these filters are effective for short-term state estimation, they are susceptible to drift, especially in scenarios with high uncertainty or poor initial conditions.

Several improvements have been proposed to enhance the consistency and accuracy of these filters. Quaternion-based EKFs (QEKFs) attempt to address rotational nonlinearities by using quaternion representation of orientation, which helps mitigate singularities associated with Euler angles. However, they still suffer from linearization errors that depend on the current state estimate. More recently, the Right InEKF has emerged as a promising alternative. The Right InEKF leverages Lie group structures to model the system dynamics, leading to error dynamics that satisfy a log-linear differential equation. This enables state-independent linearization and eliminates first-order approximation errors, improving convergence and robustness.

Potokar et al. [1] developed an InEKF implementation for AUVs that integrates IMU, DVL, and depth measurements. Their approach demonstrates significantly faster convergence and better long-term accuracy compared to QEKF methods, particularly under conditions of high initial uncertainty. Notably, they also introduced a principled way to incorporate singleton measurements, such as depth, into the Right InEKF framework, using pseudo-measurements that analytically derive infinite covariance. This allows for more accurate state estimation in scenarios where traditional EKFs may struggle.

However, prior work has largely focused on leveraging primary inertial and velocity sensors. AHRS, which combine accelerometers, gyroscopes, and magnetometers to provide drift-corrected orientation estimates, have been proven effective in aerial and land-based robotics applications. Despite their potential to reduce drift and improve state observability, the integration of AHRS into underwater navigation pipelines – especially in the context of invariant filtering – remains underexplored. This paper aims to bridge that gap by evaluating the benefits of incorporating AHRS data within the Right InEKF framework for AUV localization.

III. METHODOLOGY

In the given time constraints we had, we knew we wouldn't be able to collect AUV data until near the end of the semester. As such we had to create a simulation environment allowing us to perform so rudimentary checks on our algorithm. This would also suffice as a debugging tool for when issues are encountered using actual hardware data.

In order to realize a solution to adding AHRS data, we divide the project into 3 main sections:

- Data Simulation
- Algorithm Modification
- Collection and Integration with AUV

A. Data Simulation

We planned to use Python and Matplotlib to create potential paths that an AUV could take. We would then transform these into sensory data. Specifically, it would be contained in IMU, depth, DVL, and AHRS data. The task is split up into 2 components. The derivation of simulated paths and conversion to SE (3), and the conversion from pose information to sensor data at given time intervals. We tested a multitude

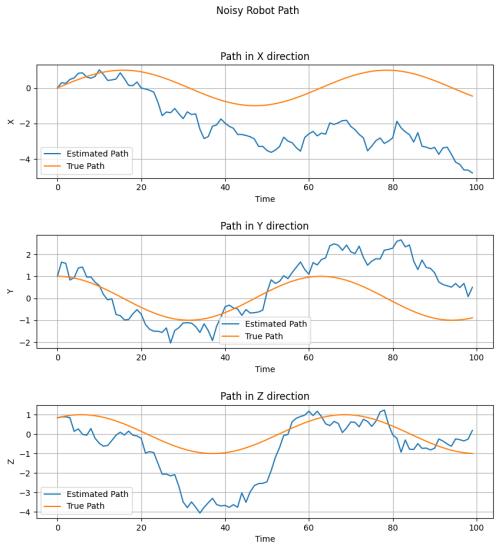

Fig. 1. Example of sine wave trajectory across XYZ. The blue line indicates the path that was simulated and would be used for deriving sensor data from.

of patterns, such as sine waves, straight lines, and random trajectories. To introduce noise, we use an additive, zero-mean noise variable as well as an accumulative bias to vary the trajectory. Additionally, some of these generated paths included faulty data to further provide testing grounds to test the capability of the InEKF. One of the main challenges we faced at this step was determining how to verify that the simulated paths were properly transforming data into sensor information correctly. To do so, we created simple plotting methods to iterate over IMU data and update the pose with respect to time to roughly visualize the performance.

In addition to these simulation scripts, we implemented different scripts to convert other sources of data into the expected CSV formats that the InEKF algorithm was expecting them. Examples of data sources that were accepted included ROS bags and custom CSVs, which represent robot sensory data in a single file. In both of these cases, there were different methodologies applied to determine how to split up the data into the 4 necessary data sources for the InEKF.

B. Algorithm Modification

With this experimental data, we were able to try making the baseline InEKF algorithm as outlined in [1]. Their code contained multiple flavors of localization algorithms. As such, trying to decipher exactly what their code did was somewhat challenging. Because of this, we relied more on our intuition and knowledge from this class to analyze their paper and reference math to design our version. This turned out to be somewhat successful.

Algorithm 1 shows an outline of the Right InEKF algorithm that we have implemented. Note that the depth measurement is not in the body frame, but can accommodate using a rigid body transformation. We assume that the DVL's data is in the same frame as the IMU. In addition to what was found in the paper, Algorithm 1 includes the update step that will be used with the newly incorporated AHRS data.

Algorithm 1 RInEKF for Underwater Navigation

$H_1 := \begin{bmatrix} 0 & I & 0 & 0 & 0 \end{bmatrix}$;
$H_2 := \begin{bmatrix} 0 & 0 & I & 0 & 0 \end{bmatrix}$;
$H_3 := \begin{bmatrix} I & 0 & 0 & 0 & 0 \end{bmatrix}$;
$\hat{\Sigma} = \Sigma_0$;
$\hat{X} = X_0$;
while receiving data **do**
  if IMU measurement **then**
    $\hat{X}, \hat{b} = f_{u_t}(\hat{X}, \hat{b})$
    $$\Phi = \exp\left(\begin{bmatrix} 0 & 0 & 0 & -\hat{R}_t & 0 \\ (g)_\times & 0 & -(\hat{v}_t)_\times \hat{R}_t & -\hat{R}_t & 0 \\ 0 & I & -(\hat{p}_t)_\times \hat{R}_t & -\hat{R}_t & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Delta t\right)$$
    $\hat{\Sigma} = \Phi \hat{\Sigma} \Phi^\top + \Phi \mathrm{Ad}_{\hat{X},\hat{b}} Q \mathrm{Ad}_{\hat{X},\hat{b}}^\top \Phi^\top \Delta t$
  else if $z =$ DVL Measurement **then**
    $z = R_{BD} z + (B P_{BD})_\times (\omega_t - \hat{b}_t^\omega)$
    $S^{-1} = \left(H_1 \hat{\Sigma} H_1^T + \hat{R} M^v \hat{R}^T\right)^{-1}$
    $K^\xi, K^b = \hat{\Sigma} H_1^T S^{-1}$
    $\hat{X} = \exp(K^\xi \Pi X z) \hat{X}$
    $\hat{b} = \hat{b} + K^b \Pi X z$
    $\hat{\Sigma} = (I - K H_1) \hat{\Sigma}$
  else if $z =$ Depth Measurement **then**
    $\tilde{\Sigma} = \left(H_2 \mathrm{Ad}_{\hat{X},\hat{b}} \hat{\Sigma} \mathrm{Ad}_{\hat{X},\hat{b}}^T H_2^T\right)^{-1}$
    $S^{-1} = \tilde{\Sigma} - \tilde{\Sigma}(\hat{R}^T M^z \hat{R} + \tilde{\Sigma}) \tilde{\Sigma}$
    $K^\xi, K^b = \hat{\Sigma} \mathrm{Ad}_{\hat{X},\hat{b}}^T H_2^T S^{-1}$
    $\hat{X} = \exp(K^\xi \Pi X^{-1} z) \hat{X}$
    $\hat{b} = \hat{b} + K^b \Pi X^{-1} z$
    $\hat{\Sigma} = (I - K H_2 \mathrm{Ad}_{\hat{X},\hat{b}}) \tilde{\Sigma}$
  else if $z =$ AHRS Measurement **then**
    $R_{\text{measured}} = \text{Convert}(z) \in \mathrm{SO}(3)$
    $V = \vee\left(\log\left(\hat{R}_t^\top R_{\text{measured}}\right)\right)$
    $\tilde{\Sigma} = H_3 \,\mathrm{Ad}_{\hat{X}^{-1},\hat{b}} \,\hat{\Sigma}\, \mathrm{Ad}_{\hat{X}^{-1},\hat{b}}^\top H_3^\top$
    $S^{-1} = \left(\tilde{\Sigma} + \hat{R}_t M^{\text{ahrs}} \hat{R}_t^\top\right)^{-1}$
    $K^\xi, K^b = \hat{\Sigma}\, \mathrm{Ad}_{\hat{X}^{-1},\hat{b}}^\top H_3^\top S^{-1}$
    $\hat{X} \leftarrow \exp(K^\xi V)\, \hat{X}$
    $\hat{b} \leftarrow \hat{b} + K^b V$
    $\hat{\Sigma} \leftarrow (I - K H_3\, \mathrm{Ad}_{\hat{X}^{-1},\hat{b}}) \hat{\Sigma}$
  end if
end while

C. Integration with AUV

There were 2 main goals when working with integrating and collecting data from the AUV. We knew that the first objective would be to capture ROS bags and convert them to the necessary data format for offline processing. There were a few reasons for this. We had no sense of how slow the latency would be to run on the actual AUV's compute resources, and we had not had a chance to validate the performance of the

InEKF with real sensor data. Additionally, introducing the InEKF as a real-time component could be prone to error, as a real-time deadline is now being introduced to the system. As such, the process of collecting ROS bags is relatively simple.

Throughout this process, we encountered unexpected issues. We realized that sensor data could be very unreliable. For the many attempts we made to collect data, the fuse on the DVL sensor blew, resulting in less-than-meaningful input. Additionally, finding the true pose information proved to be more difficult than we anticipated. The original metric was to consider how far off the AUV deviated from its true trajectory. A reasonable assumption to make is to have preplanned robot motions, but even then, due to current, drift, etc, verifying paths is not a simple task. Adding markers onto the floor or other landmarks could pose a possible solution to this, but is not a feasible choice to make given the restrictions of the testing environment.

Lastly, attempts were made to run the algorithm in real time. Data handling would occur directly with a HAL interface, streaming data into the algorithm. But due to the latency and speed of data producers, the algorithm could not keep up and resulting in many NaNs being received. These issues can be resolved with algorithm optimizations and lowering the code to a language with less overhead. Additionally, adding more robustness to the system to change the behavior when sensor data seems invalid or missing would be necessary to ensure that, during autonomous control, nothing breaks.

IV. RESULTS

To evaluate the performance of the modified Right InEKF, we compare estimations both with and without the AHRS correction step across various motion trajectories and sensor modalities. Ground truth is unavailable in real-world trials due to the limitations of underwater localization; thus, for hardware experiments, we evaluate performance relative to raw sensor observations. For simulation data, we compare against the true trajectory.

A video version of our results can be found at this YouTube link: `https://youtu.be/6RysDLbGtNM`

A. Trajectory Estimation

Figure 2 shows a 3D predicted trajectory from a real-world run. The AHRS-enhanced InEKF (oranged dashed line) tracks a smoother and more stable path than the baseline InEKF (red dashed line), which exhibits noticeable drift and instability. Integrating AHRS helps constrain orientation drift, indirectly improving positional accuracy through more stable integration of IMU and DVL data.

B. Depth Consistency

Figure 3 compares the Z-position estimation against the depth sensor. Both filters follow the general trend, but the AHRS-enhanced InEKF achieves a much closer state estimation with lower variance. The no-AHRS variant shows erratic correction early in the trajectory, likely due to the lack of orientation correction, leading to diverging sharply fromm the true depth.

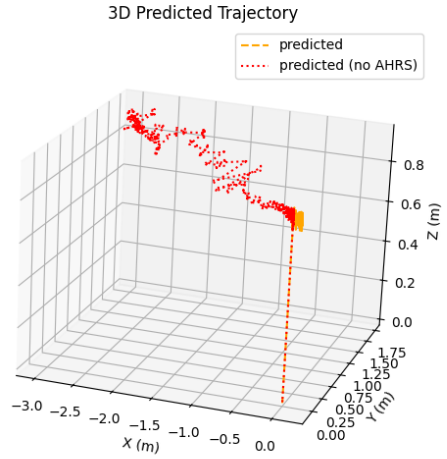

Fig. 2. 3D Plot of the XYZ position of the AUV. Reference the results section for further clarification on what each color represents.

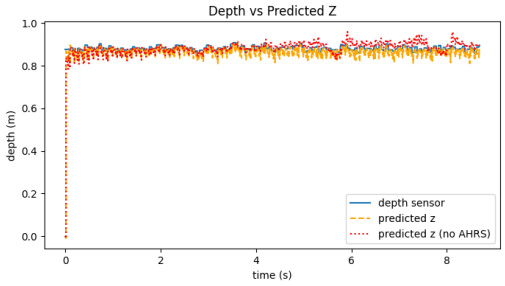

Fig. 3. Depth singleton of the AUV. Both with and without the AHRS step have similar performance.

C. Linear Velocity Estimation

Figure 4 demonstrates the velocity prediction compared to the DVL sensor measurements. The InEKF without AHRS performs poorly across all three axes, with large oscillations and drift, especially in the Z direction. The AHRS-enhanced InEKF, on the other hand, shows smoother, bounded velocity estimates that align well with the DVL measurements, especially in the X and Y directions. However, the Z-axis still shows some oscillations, which may be due to bias accumulation or initial covariance settings, indicating that the bias estimation may need further tuning.

D. Rotation Estimation

Figure 5 highlights the predictions of yaw, pitch, and roll. Without AHRS, orientation estimates diverge severely from sensor readings, leading to large errors in the predicted pose. The AHRS-enhanced InEKF closely follows the sensor data, demonstrating significantly improved rotational estimation – a key benefit of incorporating AHRS data. The AHRS update step effectively constrains the orientation drift, leading to more accurate and stable pose estimates. This is particularly

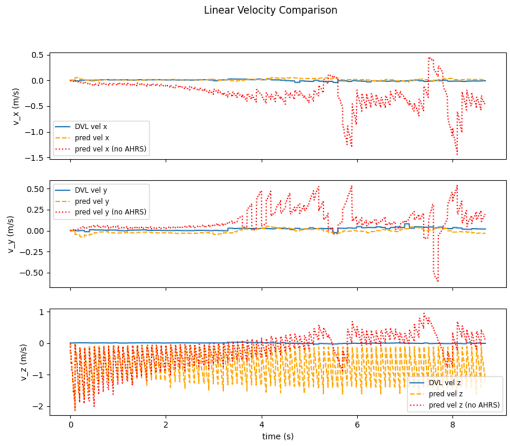

Fig. 4. Linear Velocity information split by the direction. The AHRS InEKF performs better across the X and Y dimensions but has some higher oscillations on the Z axis.

important for underwater navigation, where accurate orientation is crucial for reliable localization and control.

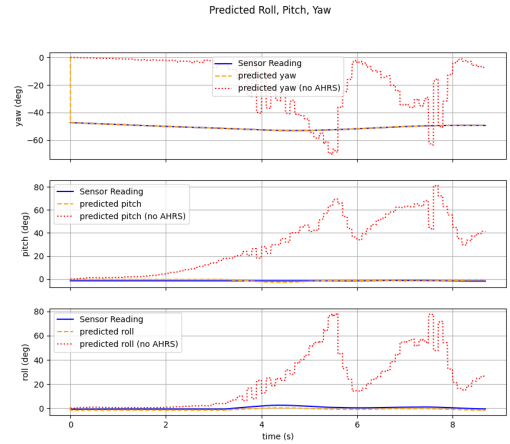

Fig. 5. Rotation information of the AUV split by Euler Angles. The AHRS performs dramatically better than the prior work.

E. Simulation Accuracy

To further validate the estimator under controlled conditions, we simulated noisy AUV trajectories and compared the predicted vs ground-truth paths (Fig. 1). Although deviations are present due to noise, the estimator captures the general motion pattern. These plots were instrumental in debugging the algorithm before hardware deployment.

F. Performance and Latency

While real-time integration was not feasible due to latency issues, offline evaluation on different platforms showed InEKF run times between 2 and 10 seconds for complete trajectory processing. Future work will explore lowering this overhead by transitioning to C++ and optimizing matrix operations.

V. CONCLUSION AND FUTURE WORK

Using this updated InEKF algorithm for AUVs has significantly improved the rotation localization, as well as the linear velocity for the X and Y dimensions. Using just IMU, depth, DVL, and AHRS data results in relatively accurate predicted poses, but can be further improved by using more information, such as cameras. To fully utilize this system, it should be lowered to a language with less overhead and closer to hardware to reduce the latency involved. More error checking should be incorporated to handle unusual or missing sensor information. We have compared the results of the InEKF with and without the AHRS update step and have noticed a significant improvement and conclude that it can be a viable solution if the latency issues can be resolved. It is noted that our AHRS update step might not be considered invariant, as it does not follow the same structure as the other update steps. This could be a potential area of improvement for future work. However, the results show that the AHRS update step is still beneficial. Another future works may also want to investigate the effect of bias tracking on DVL data and possible improvements.

VI. APENDIX

The code we created and dependency information can be found at this open source repository link: `https://github.com/EECS-568-Final-Project/AUV-InEKF` It contains a README to set up the environment and some sample files to test it out with. Included is a sub-module to generate simulation data and handle the conversion of ROS bag data.

ACKNOWLEDGMENT

This paper and work would not have been possible without the support of the instruction team of EECS 568 at the University of Michigan, Ann Arbor. Additionally, we would like to acknowledge the University of Michigan's Robosub team for providing a platform to test these algorithms on. Lastly, the work of [1] provided a foundation for us to explore the integration of AHRS data.

REFERENCES

[1] E. R. Potokar, K. Norman and J. G. Mangelson, "Invariant Extended Kalman Filtering for Underwater Navigation," in IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5792-5799, July 2021
[2] Robosub 2025 Primer and Task Ideas
[3] S. Bonnabel, "Left-invariant extended Kalman filter and attitude estimation," in Proc. IEEE Conf. Decision Control, New Orleans, LA, USA, Dec 2007, pp. 1027–1032
[4] A. Barrau, "Non-linear state error based extended Kalman filters with applications to navigation," Ph.D. dissertation, Mines Paristech, 2015.
[5] A. Barrau and S. Bonnabel, "Invariant Kalman Filtering," Annual Review of Control, Robotics, and Autonomous Systems, vol. 1, no. 1, pp. 237–257, 2018.
[6] A. Barrau and S. Bonnabel, "The invariant extended Kalman filter as a stable observer," IEEE Trans. on Automatic Control, vol. 62, no. 4, pp. 1797–1812, 2017.