Underwater Cave Exploration using Graph SLAM with Invariant EKF Initialization

Onur Bagoren Rastri Dey Zhongru Liang Zhen Zhong Zihan Zhou

***Abstract*—This work illustrates a recent advancement on simultaneous localization and mapping, namely Graph SLAM to model the 3D trajectory of an autonomous underwater vehicle (AUV). The AUV is equipped with a multitude of onboard sensors and has its dynamics represented with a 9 dimensional parameterization of position, velocity, and orientation. The prior research has not focused on the trajectory estimation in reference to a Graph SLAM problem. The designed Graph SLAM methodology constitutes the construction of a factor graph defined with the AUV poses as nodes and the sensory measurements as factors on the nodes. Additionally, we have utilized the graph initialization based on the Invariant EKF output of the robot trajectory to further ensure a robust solution to the trajectory estimation problem. Finally, we have validated our trajectory to demonstrate a comparative analysis of our Graph SLAM implementation using the GTSAM library with respect to the existing approaches of prior work. The open-source code, and an accompanying video presentation of the results can be found in the following link: https://github.com/onurbagoren/Underwater_SLAM**

I. Introduction

An Autonomous Underwater Vehicle (AUV) is an untethered robot under the control of an onboard computer, which can gather data with sufficient resolution in time and space to provide an understanding of natural structure. AUVs are increasingly used in oceanography because of their ability to collect data, while offering a low risk and cost-effective solution, especially compared to manned or tethered vehicles. The accuracy of an AUV's position over the course of a long mission is crucial for an AUV to complete a mission. Variations and inaccuracies in the AUV's motion aggregate over these missions, especially when not processing outputs of noisy sensors correctly. During long missions, these inaccuracies accumulate and can become detrimental to the field work [1]. To achieve the best possible navigation accuracy, methods that incorporate knowledge about these measurement uncertainties must be combined with accurate representations of the state dynamics, necessitating robust state estimation methods.

Apart from currents and other underwater phenomena, the 3D world makes the navigation of AUVs more demanding; more parameters should be taken into account when building models compared to land wheeled robots moving on surfaces. In this way, it is justifiable that the localization and mapping of AUVs in unstructured and unconstrained areas is challenging.

This work focuses on the tasks for localization of a marine robot in an underwater environment, specifically for underwater caves. The proposed method will incorporate a state of the art simultaneous localization and mapping (SLAM) method known as Graph SLAM. Furthermore, an Invariant Extended Kalman Filter (IEKF) will be implemented in order to utilize the error convergence properties for state prediction and use as an informative estimate of the trajectory to initialize the factor graph with. The methodology will utilize sensory information captured from an on-board camera, IMU, pressure sensor, and Doppler Velocity Logger (DVL). [2].

The novelty of our work lies in the Graph SLAM approach and the utilization of informed priors to initialize the factor graph with a filtering method before optimization, compared to related work. The objective of this work will be to produce an estimation of the trajectory followed by the robot.

The bench marking for the method proposed in this body of work will be against a filtering method that was developed for trajectory estimation [3] and a Augmented State EKF SLAM (ASEKF SLAM) method that utilizes sonar scan matching [4].

II. Related Work

Previous work on state estimation of underwater vehicles utilize well established filtering methods, primarily utilizing IEKF to represent the state of the robot as a matrix Lie Group [3] [5]. The work done by [5] derive the IEKF method and its applications for underwater vehicles, and demonstrate its effectiveness in an simulated environment. In [3], the IEKF method is applied to the Underwater Caves and Sonar dataset [2], and demonstrate improvements upon existing SLAM methods that were applied on this dataset for trajectory estimation.

Other works attempt to solve the full SLAM problem by incorporating acoustic and other exteroceptive sensors. In [6], a probabilistic scan matching algorithm is introduced, along with other methods to process planar sonar outputs for the task of SLAM in AUVs. The method is applied to the Underwater Caves and Sonar dataset [2].

III. Methodology

We have formulated a Pose-Graph SLAM approach to solve for an underwater autonomous vehicle navigating system for the trajectory estimation of a robotic vehicle. This formulation utilizes factor graphs, which can be used to efficiently solve the non-linear optimization problem to maximize the posterior probability distribution of the robots state. The formulation is then implemented using the GTSAM library [7], where the back-end optimization can be done on the constructed factor graph. This section describes the formulation and construction of the factor graph, the method for initialization, and the optimization method followed by this body of work.

A. Pose Graph SLAM and Factor Graphs

Pose Graph SLAM is a method of representing the states and observations of a robot using a Factor Graph [8]. A factor graph is a probabilistic graphical model that allows to represent the joint density of a robots' state over a trajectory. The notation in this section will treat x_i as the random variable representing the robots state at the i^{th} time and ${}_{F}X = \{x_0, \ldots, x_t\}$ as the collection of random variables, represented in the frame F.

The task of a SLAM algorithm at a high level is to perform inference, where a set of measurements Z are used to infer the observed states ${}_{F}X$, and maximize the posterior $\mathbb{P}({}_{F}X|Z)$. This process is referred to as the maximum a posterior (MAP). For systems where the posterior probability distribution is very high dimensional, such as a long trajectory with many observations, it is good to factor the distribution into local representations, utilizing properties of the model that represents the joint distribution (i.e. leveraging Markov assumptions). With Gaussian assumptions, the MAP method will be able to find a maximum, but for many robotic state estimation problems, the measurement models of sensors rarely produce Gaussian distributions, and make it difficult to use MAP to find the optimal solution [8].

Factor graphs then build upon this shortcoming of necessary Gaussian assumptions by representing the posterior without the measurements that were made, avoiding non-Gaussian distributions in the optimization problem. The formulation of the factor graph can then be represented as a bipartite graph $F = (\mathcal{U}, \mathcal{V}, \mathcal{E})$. The graph consists of two types of nodes: $\phi_i \in \mathcal{U}$ represent the factors and $x_j \in \mathcal{V}$ represent the variables. The edges, denoted $e_{ij} \in \mathcal{E}$, connect the factors and variables in the factor graph. The factors $\mathcal{U}$ then represent any sort of measurement or information that can be used for inferring information about the variables $\mathcal{V}$. In addition, a factor may have edges to multiple nodes, necessitating the representation of the neighboring variables to a factor as $\mathcal{N}(\phi)$. A factor consisting of neighbors that are non-sequential then lead for the utilization of smoothing for the SLAM application.

For this work, the factor graph is modeled as such:

1) Variables ${}_{w}\boldsymbol{x}_j \in \mathcal{V}$ represent the pose of the robot in the world frame, using the $SE_2(3)$ matrix group, as shown in eq. (1).
2) Factors $\phi_i \in \mathcal{U}$ represent measurements taken by the three sensors: IMU, Doppler Velocity Logger (DVL), and depth sensor.

$$ {}_{w}X_t = \begin{bmatrix} R_t & v_t & p_t \\ 0_{3\times1} & 1 & 0 \\ 0_{3\times1} & 0 & 1 \end{bmatrix}_{9\times9} \in SE_2(3) \tag{1} $$

For the implementation of the factor graph, the GTSAM library was used, where each sensor measurement that was used in the factor graph was implemented as either a custom defined factor, or the existing factor implementations provided by GTSAM.

TABLE I
INFORMATION REGARDING THE SENSORS THAT WERE USED IN THE FACTOR GRAPH.

	Sampling Frequency (Hz)	Measurement Returns	Measurement Frame
IMU	10	$T \in SE_2(3)$	Body
DVL	2	$v \in \mathbb{R}^3$	Body
Depth Sensor	10	$z \in \mathbb{R}^1$	World

B. Graph Construction

1) Keyframe Selection: For constructing the graph, there were multiple methods that could be selected for defining keyframes. Keyframes [9] represent selected times or events during the robots operation that can be used to best represent key moments in the robots operation. The purpose of keyframes is to have a graphical model that can describe the state of the robot over time in a concise and sparse manner, as without the keyframes, the graph would be dense and require more computation power or time to solve, without a large marginal benefit.

As the sensors that were being used to construct the graph did not include any exteroceptive sensors, defining a keyframe by a new sensor registration would not be possible. In addition, due to the fact that each sensor captured data at a different rates (see table I for the corresponding sampling rates of each sensor) and different times, defining a keyframe based on one of the sensors would be challenging.

The solution to the keyframe selection was then defined as time based, where after a certain amount of time of the robots operation, a new node to the factor graph would be added. In order to then incorporate sensor measurements to the factor graph, the timestamp at which the node was added was compared to the closest time in each sensors measurement. If the time of the measurement was within a certain amount of time to the time at which the node was added, it would be added as a factor to the newly added node, implying that the sensor measurement is informative of the robots state at that time.

2) IMU Integration: The IMU measurements return information regarding the robots angular velocity and linear acceleration in the body frame. The integration of these measurements over the time at which they were measured can be used to compute the rotation, translation and velocity associated with the robots transformation.

For the factor graph that was constructed in this project, the IMU measurements were used as factors representing the transformation between two subsequent nodes. In order to add these factors to the graph, GTSAMs [7] preintegrated IMU instance is used [10]. This method utilizes the preintegration theory for IMU measurements that properly represents the manifold structure of the matrix group that can represent the robots motion in exponential coordinates. This method allows compounding consecutive measurements of IMU measurements into a single measurement. This single measurement can then be used to describe the transformation between consecutive keyframes.

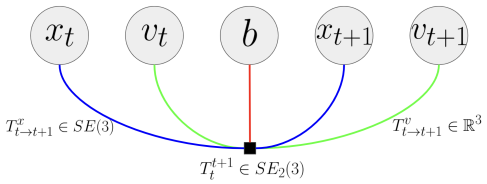

Fig. 1. Representation of the IMU Factor in the factor graph. The factor itself is shown as the connector between the five nodes, indicating that the constraint is set on all of these nodes. The blue line represents the constraint on the poses, and the green line represents the constraint on the velocity. These constraints are represented by the single IMU factor, which both are connected to.

For IMU measurements, the robot state can be composed of four different variables, $X_t = [R_t, p_t, v_t, b_t]$, such that the pose $[R_t, p_t] \in SE(3)$, velocity $v_t \in \mathbb{R}^3$, and bias $b_t = [b^a_t, b^\omega_t] \in \mathbb{R}^6$, where b^a_t is the bias associated with the accelerometer and b^ω_t is the bias associated with the gyroscope. The measurements from the IMU represent the true state with additive Gaussian white noise and compensation for the bias of each sensor. In addition, it is important to note that the gyroscope measurements return the rate of rotation of the body relative to the world frame W, expressed in the body frame B, and the accelerometer measurements are the acceleration of the sensor, in the body frame B. These measurements are concisely shown in eq. (2) - (3), where $\boldsymbol{\eta}^\omega, \boldsymbol{\eta}^a$ are the noise associated with the sensor measurements, and $\boldsymbol{b}^\omega$, $\boldsymbol{b}^a$ are the bias terms.

$$ {}_B\tilde{\boldsymbol{\omega}}_{WB}(t) = {}_B\,\boldsymbol{\omega}_{WB}(t) + \boldsymbol{b}^\omega(t) + \boldsymbol{\eta}^\omega(t) \tag{2} $$

$$ {}_B\tilde{\boldsymbol{a}}(t) = R^T_{WB}(t)\left({}_W\boldsymbol{a}(t) - {}_W\, g\right) + \boldsymbol{b}^a(t) + \boldsymbol{\eta}^a(t) \tag{3} $$

With these measurements, and the robots kinematic model. the integration of the measurements can be computed using Euler's method of integration, to compute the state after Δt.

This definition of the integration, although able to be used, requires continuously adding new measurements as factors to the graph, as integrating over multiple measurements between keyframes will cause drift and inaccurate integration. Work from [10] utilizes integration on the manifold of the Lie group $SE_2(3)$, and is able to mitigate this issue by accurately propogating the error associated with each transformation. Equations (4, 5, 6) represent this integration on the manifold of $SE_2(3)$ Lie Group for $(\boldsymbol{R}_j, \boldsymbol{v}_j, \boldsymbol{x}_j)$ after accounting for measurements $t_i \to t_j$.

$$ \boldsymbol{R}_j = \boldsymbol{R}_i \prod_{k=i}^{j-1} \exp\left(\left(\tilde{\omega}_k - \boldsymbol{b}^\omega_k - \boldsymbol{\eta}^\omega_k\right)\Delta t\right) \tag{4} $$

$$ \boldsymbol{v}_j = \boldsymbol{v}_i + \boldsymbol{g}\Delta t_{ij} + \sum_{k=1}^{j-1} \boldsymbol{R}_k\left(\tilde{\boldsymbol{a}}_k - \boldsymbol{b}^a_k - \boldsymbol{\eta}^a_k\right)\Delta t \tag{5} $$

$$ \boldsymbol{p}_j = \boldsymbol{p}_i + \sum_{k=i}^{j-1} \boldsymbol{v}_k \Delta t + \frac{1}{2}\boldsymbol{g}\Delta t^2_{ij} + \frac{1}{2}\sum_{k=i}^{j-1} \boldsymbol{R}_k\left(\tilde{\boldsymbol{a}}_k - \boldsymbol{b}^a_k - \boldsymbol{\eta}^a_k\right)\Delta t^2 \tag{6} $$

Using this definition, adjusting for repeated integrations by removing pose and velocity at t_i, and applying first order approximations to the compounding pose estimations by assuming that rotation noise is small, the end preintegrated IMU model can be expressed as shown in eq. (7, 8, 9). In these expressions $\left(\delta\phi_{ij}, \delta\boldsymbol{v}_{ij}, \delta\boldsymbol{p}_{ij}\right)$ represent the zero-mean Gaussian noise associated with the transformations.

$$ \Delta\tilde{\boldsymbol{R}}_{ij} = \boldsymbol{R}^T_i \boldsymbol{R}_j \exp\left(\delta\phi_{ij}\right) \tag{7} $$

$$ \Delta\tilde{\boldsymbol{v}}_{ij} = \boldsymbol{R}^T_i\left(\boldsymbol{v}_j - \boldsymbol{v}_i - \boldsymbol{g}\Delta t_{i\to j}\right) + \delta\boldsymbol{v}_{ij} \tag{8} $$

$$ \Delta\tilde{\boldsymbol{p}}_{ij} = \boldsymbol{R}^T_i\left(\boldsymbol{p}_j - \boldsymbol{p}_i - \boldsymbol{v}_i\Delta t_{i\to j} - \frac{1}{2}\boldsymbol{g}\Delta t^2_{ij}\right) + \delta\boldsymbol{p}_{ij} \tag{9} $$

The IMU factors then are defined by the corresponding error for each transformation and Jacobian matrix, which is defined in equations $(A.21) - (A.33)$ of [11]. Now, with the definitions of the residuals, Jacobians and the preintegrated measurements that yield the transformations, the constraints in the factor graph can be added, as shown in Fig. 1. Specifically in the GTSAM framework, these transformations are represented as the $SE_2(3)$ matrix group, as shown in eq. (1) and is referred to as the NavState.

Additionally, as a new keyframe is defined and a new node is added, the IMU preintegrator is reset to stop accumulating measurements from previous nodes. This also acts as a measure to maintain the the Markov identity of the factor graph.

3) Sensor Fusion: As discussed in Section III-A, the sensors that were used for the trajectory estimation of the AUV were the IMU, depth sensor and Doppler velocity logger (DVL). To use each sensor in the factor graph, two properties must be defined for the measurements that they collect: the residual from the measurements made by the sensor, and the Jacobian of the sensor models.

These are important for the optimization step, where the objective is to minimize the residual by following in the direction of steepest descent of the gradient, which requires the Jacobian.

a) Depth Sensor: The depth sensor returns depend on the on-board pressure sensor, which measures the robots depth relative to atmospheric pressure. The sensor model is provided in eq. (10), where the state is represented as the pose of the robot as a column vector. ${}_W X = [\phi\ \theta\ \psi\ x\ y\ z]^T \in \mathbb{R}^6$. This measurement can be treated as measuring the robots state in the world frame, shown as ${}_w\hat{\boldsymbol{z}}_{depth}$. As there is a direct correspondence with the state of the robot, which is also being tracked in the world frame, the definition of the residual is simply the difference of the measurement and the estimated robot state, as shown in eq. (11).

$$ h_{depth}({}_W X_i) = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}_W X^x_i \tag{10} $$

$$ r_{depth} = {}_w\,\hat{\boldsymbol{z}}_{depth} - {}_w\,\boldsymbol{X}^z_t \in \mathbb{R}^1 \tag{11} $$

The Jacobian matrix is then formulated as a 6 dimensional row vector, as shown in eq (12), where h_{depth} is shortened to h_d for readability.

$$H_{depth} = \begin{bmatrix} \frac{\partial h_d}{\partial \phi} & \frac{\partial h_d}{\partial \theta} & \frac{\partial h_d}{\partial \psi} & \frac{\partial h_d}{\partial x} & \frac{\partial h_d}{\partial y} & \frac{\partial h_d}{\partial z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{1\times 6} \tag{12}$$

b) Doppler Velocity Logger: The Doppler Velocity Logger (DVL) is an acoustic based sensor which measures the velocity of a body, relative to another frame. For this application, the velocity relative to the earth was used. As this measurements is in the robots body frame relative to the world, the measurements must be multiplied by the estimated rotation in order to place it in the worlds frame. The sensor model for the DVL is shown in eq. (13), where ${}_F X_i^v = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \in \mathbb{R}^3$ is the velocity of the state at time i in frame F. The computed residual is then given in eq. (14), where $\tilde{(\cdot)}$ represents the measured values and $\hat{(\cdot)}$ represent the state estimate, in the world frame. The corresponding Jacobian is then the rotation matrix, which is multiplied with the measurement as shown in eq. (15).

$$h_{DVL}\left({}_{WB}X_i^v\right) = {}_W\hat{\boldsymbol{R}}_i \; {}_{WB}X_i^v \tag{13}$$

$$r_{DVL} = {}_W\hat{\boldsymbol{R}}_i \begin{bmatrix} {}_{WB}\tilde{v}_x \\ {}_{WB}\tilde{v}_y \\ {}_{WB}\tilde{v}_z \end{bmatrix} - \begin{bmatrix} {}_W\hat{v}_x \\ {}_W\hat{v}_y \\ {}_W\hat{v}_z \end{bmatrix} \in \mathbb{R}^3 \tag{14}$$

$$H_{DVL} = {}_W\hat{\boldsymbol{R}}_i \in SO(3) \tag{15}$$

C. Graph Initialization

The graph initialization is a process where after the graph is constructed, the nodes that represent the poses are initialized to be used as the initial condition for the optimizer. This section describes the method that was used to initialize the graph, using a Right IEKF.

1) Right IEKF: The input for initial values at the nodes of the factor graph are the state estimate outputs of an IEKF from [3]. The right IEKF represents the state in an augmented special euclidean matrix group $SE_2(3)$, as shown in eq. 1. This representation of the state corresponds to the modeling of the matrix group that exists on the manifold of $SE_2(3)$, tracked in the world frame W. The Lie Algebra of the Lie Group can then be formulated as shown in eq. 16, which represents the tangent space to the manifold, at the identity.

$$\begin{bmatrix} 0 & -\boldsymbol{R}_z & \boldsymbol{R}_y & \boldsymbol{v}_x & \boldsymbol{p}_x \\ -\boldsymbol{R}_z & 0 & -\boldsymbol{R}_x & \boldsymbol{v}_y & \boldsymbol{p}_y \\ -\boldsymbol{R}_y & \boldsymbol{R}_x & 0 & \boldsymbol{v}_x & \boldsymbol{p}_z \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}_2(3) \tag{16}$$

Under this framework, the AUV motion model is then represented with a constant velocity model, expressed in the Lie Group. The motion model is shown in eq. (17-19), where ω_k represents the angular velocity of the robot and a_t, g are the acceleration in body frame and the constant gravity vector respectively.

$$\bar{R}_t = R_{t-1}\exp\left(\omega_k^{\wedge}\Delta t\right) \tag{17}$$

$$\bar{v}_t = v_{t-1} + R_{t-1}a_t\Delta t + g\Delta t \tag{18}$$

$$\bar{p}_t = p_{t-1} + v_{t-1}\Delta t + \frac{1}{2}R_{t-1}a_t\Delta t^2 + \frac{1}{2}g\Delta t^2 \tag{19}$$

a) Right IEKF Propagation: For the propagation, the measurements from the IMU are used, which outputs measurements regarding the robot's angular velocity, and linear acceleration in the robot frame, relative to the world frame as shown in eq. 20.

$$\tilde{u} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & a_x & a_y & a_z & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

The propagation then computes the estimation of the robot's state along with its covariance in the Lie Group space, as eq. (21-26).

$$A = \begin{bmatrix} 0 & 0 & 0 \\ g^{\wedge} & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \tag{21}$$

$$\mathrm{Ad}_x = \begin{bmatrix} R & 0 & 0 \\ v^{\wedge}R & R & 0 \\ p^{\wedge}R & 0 & R \end{bmatrix} \tag{22}$$

$$\Phi = \exp\left(A\Delta t\right) \tag{23}$$

$$Q_d = \Phi Q \Delta t \Phi^T \tag{24}$$

$$\bar{X}_{t+1} = f\left(X_t, \tilde{u}_t, \Delta t\right) \tag{25}$$

$$\bar{P}_{t+1} = \Phi P_t \Phi^T + \mathrm{Ad}_{X_t} Q_d \mathrm{Ad}_{X_t}^T \tag{26}$$

where the covariance Q is represented as a 9×9 block diagonal matrix incorporating the individual covariances related to angular velocity ω, acceleration a and position p respectively.

b) Right IEKF Correction: The predicted state representation element $\bar{X}_{t+1} \in SE_2(3)$ and the covariance propagation $\bar{P}_{t+1}$ from eq. (25-26) further undergoes the Invariant EKF correction step. For the correction, the measurements from the on-board depth sensor, Doppler velocity logger (DVL) and magnetometer are used. The measurements and their Jacobians are then stacked to use in the correction step.

The filter then computes the corrected robot state and covariance in the Lie Group, following eq. (27-30). This output is then used to initialize the factor graph at each matching timestep.

$$S = \hat{H}\bar{P}_t\hat{H}^T + \hat{N} \tag{27}$$

$$L = \bar{P}_t\hat{H}^T S^{-1} \tag{28}$$

$$P_t = \left(I - LH\right)\bar{P}_t\left(I - LH\right)^T + L\hat{N}L^T \tag{29}$$

$$X_t = \exp\left(\left(L\left(\hat{X}_t Y - \hat{b}\right)\right)^{\wedge}\right)\bar{X}_t \tag{30}$$

D. Nonlinear Graph Optimization

For nonlinear graphs, assuming Gaussian noise models for the measurements, solving for MAP is equivalent to solving a nonlinear optimization problem. This Gaussian assumption for the noise model can be shown as eq. (31), where a

factor in the graph is represented according to a sensor with model $h_i({}_F X_i)$, and a measurement made z_i. Reframing the expression for the factor in the factor graph, it can be posed as a nonlinear optimization problem seen in eq. (32), where the residual r_i is defined as $r_i = h_i({}_F X_i) - z_i$.

$$\phi_i({}_F X_i) \propto \exp\left\{-\frac{1}{2}\|r_i\|^2_{\Sigma_i}\right\} \tag{31}$$

$${}_F X = \arg\min_{{}_F X} \sum_i \|r_i\|^2_{\Sigma_i} \tag{32}$$

To then perform nonlinear optimization, the logic follows of searching over the space of the residual generated from each factor and minimizing it over the entire graph. In order to perform this, linearization of every sensor measurement is necessary with a first order Taylor-series expansion, necessitating the definition of the Jacobian for each sensor model.

1) Batch Optimization: For the optimization of the nonlinear factor graph, batch optimization was performed, using the Levenberg Marquardt optimization method [12]. Batch optimization is a method that optimizes over the entire factor graph after it is constructed, yielding an offline solution after the front end is constructed. This optimization method builds upon the Gauss-Newton optimization method [13], which utilizes an approximation of the Hessian matrix to improve the descent in the steepest direction over the residual space. The improvement of the Levenberg-Marquardt method is that a trust-region is defined in which iterative steps can be taken, in order to ensure quicker convergence. With proper initialization and an objective function that is quadratic, this method can provide quadratic convergence [8].

IV. Results

The associated variance for each factor is then set to $\sigma^2 = 0.1\ au$ for each state dimension that a sensor returns information on. Additionally, the keyframes are defined as 1 second of robot run time, and sensor measurements are added as factors if the measurement time is within 1e-4 seconds of the time at which the node was added. The bias values obtained by dataset [2] were used for initializing the bias of the IMU sensor. The Levenberg-Marquardt optimizer is set to have a maximum number of iterations of 1000.

The results are evaluated based on metrics defined by Mallios, et. al [4]. The metric is defined based on known landmarks in the underwater cave where the data was collected. These landmarks are neon orange cones and are hence referred to as cones. The trajectory of the robot, when the data was captured, was so that the robot traversed over these cones twice.

The first metric is measuring the differences in positions of the robot during the first and second pass over the same cone. The ground truth distances for these are measured by the on board camera, where with known camera calibration values and geometric properties of the cone, the distance can be inferred. Although the difference of position of the robot when traversing over the same cone is not exactly 0 m, it is the objective to minimize this distance. The second metric is the traveled distance between subsequent cones. The ground truth for these distances are determined by measurements taken when the cones were initially placed. The extracted trajectory is visualized in figures 2 and 3. Both metrics are measured in meters.

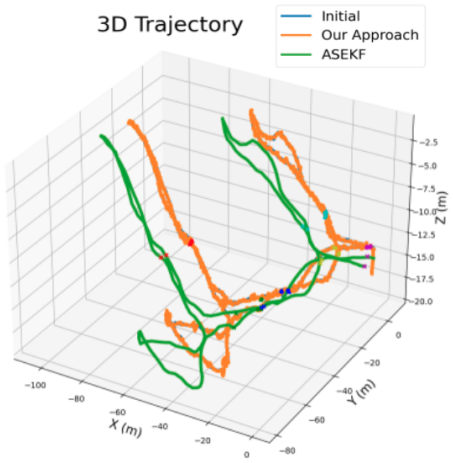

Fig. 2. The predicted robot trajectory in the Underwater Caves Dataset. Results shown in orange, with the known poses of the cones marked as different colors along the trajectory to indicate distance of the robot to the cone.

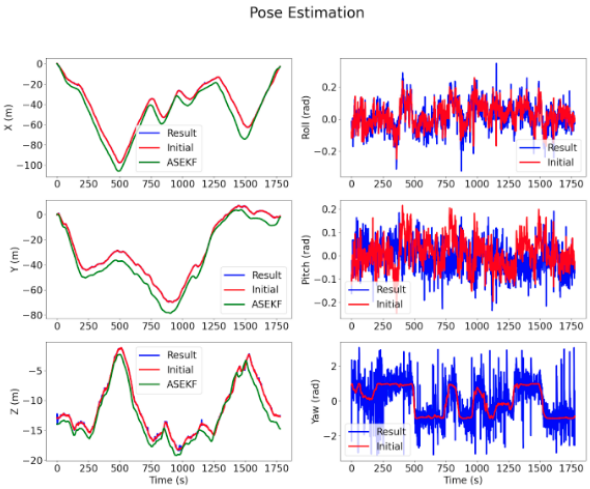

Fig. 3. The individual dimensions of the robots state, compared to the benchmarked methods. The first column shows the position in x, y, z; second column shows the robots orientation in roll, pitch yaw. Results from this work are shown in blue.

A. Distance from Known Landmarks Between Multiple Observations

This is the first metric defined above, which aims to minimize the distance the robot has to the same cone between the two times it passes over it. The distance to the cones are determined by the position of the robot at the timestep that is closest to the ground truth cone position, and subtracting the

distances. Table II displays the results that were obtained for this metric.

TABLE II
DISTANCE BETWEEN PREDICTED POSITIONS IN THE 1ST AND 2ND PASS (M)

Cone #	Our Approach	InEKF	ASEKF SLAM
1	2.51	2.30	**1.96**
2	2.49	2.36	**1.77**
3	2.47	3.03	**1.64**
4	3.86	3.66	**1.79**
5	**0.63**	0.7	2.10
6	2.83	**2.02**	2.31

B. Distance Between Known Landmarks

The results for this metric aim to compute whether the robots trajectory within two cones were consistent. The objective for this metric is to produce a distance that is close to the ground truth distance.

TABLE III
DISTANCE BETWEEN SUCCESSIVE CONES (M)

Cone #	Ground Truth	Our Approach	InEKF	ASEKF SLAM
1 →2	19	15.4	**17.8**	14.6
2→3	32	29,7	**30.8**	46.7
3→4	16	13.1	**14.2**	20.0
4→5	16	12.8	**17.7**	19.5
5→6	32	28.4	**32.0**	38.6
6→1	30	22.6	25.2	**30.8**

V. DISCUSSION AND CONCLUSION

The proposed method outperforms the other methods on the 5^{th} cone for the metric which computes the distance between the two passes over the same cones. For the other cones of the same metric, our method performed comparably to the other methods.

For the second metric, the proposed method consistently outperformed one of the existing methods. This metric aims to capture whether the predicted trajectory is representative of the ground truth trajectory when unable to measure the robots ground truth pose.

Visually, the comparison of the trajectories can be seen from Fig. 2 and 3. Fig. 2 shows the poses tracked in the 3D coordinate frame, while Fig. 3 displays the values for each dimension of the pose individually. The output from the proposed method tracks the output of the IEKF method, and has a shift compared to the ASEKF SLAM method. This shift can be explained by the corrections the ASEKF SLAM method accounts for with the loop closures it registers from the sonar scans.

It is observed that the method works especially similar to the Invariant EKF method, reporting similar results in the benchmark metrics. This can be linked to the fact that the initialization of the poses in the factor graph used the poses of the Invariant EKF output. As the Invariant EKF used the same sensors for the task of filtering, it is expected that the added constraints would not cause a large deviation from the initialization. In addition, both methods model the IMU and state of the robot in the same way, utilizing Lie group theory in order to integrate the measurements made by the sensor and propagate error on the Lie Algebra before lifting back to the Lie Group of $SE_2(3)$. So, it can be expected that the methods process the sensory inputs similarly in how it is integrated into the problem of trajectory estimation.

The methods differ in the manner that the trajectory itself is estimated. The filtering algorithm follows the process of marginalizing all previous poses in order to estimate the pose at the latest time. This is significantly different than the method proposed by this work, which aims to minimize the error on the graph in batch, aiming to optimize over the constraint set on each of the factors. Although the initialization of the graph utilizes a very informed trajectory from the Invariant EKF, if the cost function of the optimization formulation is not quadratic, it is possible for the optimizer to fall in a local minimum [7], and the results will not be good representations of the actual robot trajectory.

In addition, the proposed method requires the utilization of keyframes, which is a way to reduce the dimension of the optimization problem. This use of keyframes results in a lower-resolution representation of the full state of the robot, while the Invariant EKF, due to marginalizing out the previous poses every step, does not necessitate reducing the resolution of the full state over time. This reduced resolution can lead to inaccurate trajectories between the nodes representing poses.

In conclusion, it can be seen that the proposed method, and the novelty provided by it, enable accurate estimation of the robot trajectory, improving upon existing methods in some categories of the benchmarking methods. With future work including the fusion of more sensors, the existing framework can easily be altered and tested with the appropriate metrics.

VI. FUTURE WORK

In future work, we want to extend our batch optimization of the Graph SLAM problem, with an improved incremental solver, such as iSAM2 [14] to enhance the trajectory optimization. Moreover, we want to utilize the raw sonar scans from the given dataset to obtain loop closures in localization and mapping to construct a more accurate graph with minimal error deviation from the groundtruth landmarks. This method of using planar sonar scans for scan matching and loop closures is described in the work produced in [4], but requires significant alterations for the method of using factor graphs, which is a direction that this work can proceed in.

VII. ACKNOWLEDGEMENTS

This paper and the research behind it would not have been possible without the exceptional support of Professor Maani Ghaffari and the instructional team of Mobile Robotics course at University of Michigan, Ann Arbor. Additionally, we would like to acknowledge the foundational work that is open sourced by [3] that enabled us to inquire about this topic.

REFERENCES

[1] L. Stutters, H. Liu, C. Tiltman, and D. J. Brown, "Navigation technologies for autonomous underwater vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 4, pp. 581–589, 2008.

[2] A. Mallios, E. Vidal, R. Campos, and M. Carreras, "Underwater caves sonar data set," *The International Journal of Robotics Research*, vol. 36, no. 12, p. 1247–1251, 2017.

[3] Sansaldo, "Sansaldo/iekf_auv_cave_navigation." [Online]. Available: https://github.com/sansaldo/IEKF_AUV_Cave_Navigation

[4] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 33, no. 7, pp. 994–1012, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21640

[5] E. Potokar, K. Norman, and J. Mangelson, "Invariant extended kalman filtering for underwater navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, p. 5792–5799, 2021.

[6] A. Mallios, P. Ridao, D. Ribas, and E. Hernández, "Scan matching slam in underwater environments," *Autonomous Robots*, vol. 36, no. 3, p. 181–198, 2013.

[7] M. Kaess, "Gtsam library," July 2015.

[8] F. Dellaert and M. Kaess, "Factor graphs for robot perception," 2017.

[9] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular slam: Why filter?" in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2657–2664.

[10] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," 07 2015.

[11] ——, "Supplementary material to : Imu preintegration on manifold for visual-inertial maximum-a-posteriori estimation technical report gt-irim-cp & r-2015-001," 2015.

[12] J. J. Moré, "The levenberg-marquardt algorithm: Implementation and theory," *Lecture Notes in Mathematics*, p. 105–116, 1978.

[13] S. Gratton, A. Lawless, and N. Nichols, "Approximate gauss–newton methods for nonlinear least squares problems," *SIAM Journal on Optimization*, vol. 18, pp. 106–132, 01 2007.

[14] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.