Multi-agent SLAM Using Extended Kalman Filter

1st Ryan Wu
Mechanical Engineering
Carnegie Mellon University
weihuanw@andrew.cmu.edu

1st Kevin Tang
Robotics Institute
Carnegie Mellon University
kevintang@cmu.edu

1st Nithya Sampath
Electrical & Computer Engineering
Carnegie Mellon University
nssampat@andrew.cmu.edu

1st Tianqi Yu
Mechanical Engineering
Carnegie Mellon University
tianqiyu@andrew.cmu.edu

***Abstract*—This paper investigates the multi-robot 2D SLAM problem via the Extended Kalman Filter (EKF). Each robot detects the complete set of landmarks and uses these observations to update its state estimate independently. Finally, an information fusion method is applied to compute a global estimate of the landmarks, which is calculated through a weighted averaging method based on the individual covariance matrices of the agents. Simulation using the NVIDIA Isaac Sim tool and several experiments were conducted to show the effectiveness of the proposed method; it was found that the proposed method produced better accuracy when evaluated against single-agent EKF. The GitHub repository for this project is available at: https://github.com/ryanwu0521/SLAM_Project**

***Index Terms*—Extended Kalman filter, multi-agent EKF, NVIDIA Isaac Sim.**

I. Introduction

The Kalman filter has been widely used as the optimal solution to many modern tracking and localizing tasks. The ease of implementation and the accuracy of performance are some of the benefits the Kalman filter has over other methodologies. However, the Kalman filter can only model linear systems. For non-linear systems, the EKF is used to model the non-linear measurements.

The EKF is another classic simultaneous localization and mapping (SLAM) algorithm that leverages motion models to predict the robot's next pose and integrates sensor measurements to correct these predictions. By accounting for uncertainties associated with motion and sensor readings, EKF provides estimates of the robot's trajectory and the surrounding map. This iterative process enables autonomous systems to simultaneously localize themselves within the environment and construct a map of previously unknown areas.

For more complex models with larger state estimations, a decentralized approach can improve the efficiency and performance of the implementation. Our multi-agent EKF aims to study the benefits of multi-agent collaboration. We will be developing and solving the localization problem with our multi-agent EKF solver. Furthermore, both 2D and 3D visualization tools will be provided to better understand our implementation and compare the single-agent and multi-agent estimation results.

II. Motivation

In the realm of SLAM studies, most systems operate in single-agent environments. Data are gathered and processed without any cross-agent collaborations or communication. However, most real-world scenarios call for multi-agent collaboration. For example, in disaster rescue, a single deployment of an unmanned ground vehicle (UGV) will have limited range and view on dangerous terrains. Yet, with the help of an unmanned aerial vehicle (UAV), the rescue task will completed with more success. Multi-agent collaborations also introduce more efficient solutions for exploration tasks and data collection. Each agent will take on a divide-and-conquer approach leading to a faster convergence.

The team is motivated to expand our understanding from single-agent to multi-agent SLAM systems. We hope to investigate the advantages of multi-agent SLAM and implement some state-of-the-art algorithms in our project. In particular, we will be implementing a multi-agent EKF framework in a simulation environment (NVIDIA Isaac Sim). We hope with this project, the team understands the nuances of creating a workable multi-agent framework.

III. Related Work

As [1] mentions, there are two different approaches we can take for implementing an EKF SLAM system with multiple agents, and in particular, for computing the state estimate $\hat{\mathbf{x}}$. The first approach involves collecting all measurements $\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^M$, and $\mathbf{u}$ at each time step, and then computing the Kalman Filter for the entire system. However, this approach is computationally expensive, so we consider a second approach, in which each agent maintains its own measurements and state estimate. That is, each robot runs its own EKF independently, and after each update step, individual estimates of mean and covariance for landmarks are combined to obtain a global estimate for landmark positions and the corresponding covariance.

Both of the papers discussed below employ this second approach to implementing a multi-agent EKF SLAM system. There are several important takeaways from each paper listed below, which we use to inform our own proposed method and implementation.

A. Multi-Robot SLAM with Information Fusion Extended Kalman Filter

[1] explores localization and mapping using the Extended Kalman Filter algorithm and multiple agents; the information from the agents is combined using an information fusion method which is discussed below.

As previously stated, [1] uses the Extended Kalman Filter for localization and mapping (and in fact, also proposes a

simpler multi-agent SLAM implementation using the Kalman Filter). First, there is a prediction step based on a nonlinear motion model, and then there is an update step based on a measurement model. As part of the state vector, the pose of one of the robots is represented by a vector (x, y, θ), where x, y are position coordinates and θ is the heading angle of the robot. For the state estimation, each robot uses EKF to independently estimate its position, the position of other robots, and the positions of landmarks based on its sensor measurements. Importantly, the measurement model incorporates distances/directions to other agents.

After each robot computes its state estimate using EKF, these estimates are sent to a "base station". At the base station, an optimal information fusion algorithm is applied. This algorithm minimizes estimation error covariance through weighted averaging of the individual estimates from the robots. Importantly, the optimal weight matrices for averaging are determined by the covariance matrices of the individual estimates, and in particular, computing and stacking "cross-covariance" matrices. This method of computing weights will be discussed in more detail in Section IV, and it ensures that more accurate estimates have a greater influence on the final fused estimate.

In terms of experimental validation, the authors compare their proposed method with standard EKF using a single agent. These results show improved estimation accuracy for both robot positions and landmark locations compared to standard EKF approaches. The experiments underscore the practical viability of the approach, highlighting its potential in real-world multi-robot navigation and mapping tasks. However, it is important to note that this approach makes use of distributed computational resources, and this is something to keep in mind when we try our implementation of multi-agent SLAM with EKF.

In summary, our major takeaways from this paper when implementing our method for multi-agent EKF are as follows:

- Each agent should run EKF independently, and after each iteration, a "base station" should combine the state estimates.
- Weights should be computed based on covariances, such that agents with generally lower covariances (and thus, more precise estimates) should be given greater weights when computing the global estimate.
- If implemented correctly, we can expect improved estimation accuracy for landmarks with the multi-agent system compared to single-agent EKF.

B. Extending SLAM to Multiple Robots

Once again, the authors of [2] aim to enable robots to share SLAM data and collaborate on tasks, and the authors specifically mention time-sensitive applications like search and rescue. Each robot creates a map relative to an internal basis and updates its state based on its movements and measurements. When incorporating multiple agents, a robot must integrate the map data from another robot by transforming and adding it to its map, considering that each robot has limited information about the environment. Unlike [1], there is direct communication from one robot to another, instead of having a "base station" where information from all robots is collected.

Similar to the previous paper, the state vector includes the robot position, angle, and positions of all objects. Again, the key idea in this paper is that robots communicate their SLAM data directly to others: this includes sharing state estimates and covariance matrices. The receiving robot treats this communicated data as measurements to update its state estimates.

Additionally, the primary objective of the system proposed in [2] seems to be to seek out unmapped regions and using all of the agents, to develop a comprehensive map of the area in real time.

Like [1], there is an information fusion technique. Unlike [1], the main focus of the information fusion seems to be coverage of different portions of the map, specifically in real-time; robots seek out unmapped regions and the system uses all of the agents to develop a comprehensive map. Specifically, the shared data is transformed into the receiving robot's coordinate system, which is then used to update the robot's state estimates and covariance matrices, similar to processing regular measurements. A novel aspect discussed is the determination of each robot's origin in the coordinate system of others, which is essential for correctly interpreting shared data. Robots may learn each other's origins through direct observation, communication with other robots, or by recognizing common landmarks.

In summary, our major takeaways from this paper when implementing our method for multi-agent EKF are as follows:

- Rather than having a base station, we can have direct communication between agents in our implementation so that robots can obtain each others' state estimates and covariance matrices.
- Not all landmarks necessarily have to be observed by every robot at each iteration, although assuming that this is the case may make it easier to run EKF.
- Most importantly, we should consider using a synthetic dataset for our experiments, in which we can add desired Gaussian noise to the laser measurements based on specific covariance matrices.

IV. Methodology

In this section, we give a detailed overview of our methodology, including the complete pipeline, our proposed method, dataset generation, and the multi-agent EKF algorithm which we attempted to implement from [1].

A. Pipeline

- Generate synthetic datasets of the robot's measurements and control data with noise.
- Solve the multi-agent problems using our multi-agent EKF solver.
- Compare the landmarks position estimation results between single-agent EKF and multi-agent EKF frameworks.

- Visualize the landmark's positions and robot motion in 3D using NVIDIA Isaac Sim.

B. Dataset Generation

Since there are no available datasets of the system in question, we used simulation to generate our own dataset. The generation is done in a simulation environment featuring predefined landmarks with known 2D coordinates represented by a matrix:

$$L = \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ \vdots & \vdots \\ x_m & y_m \end{bmatrix},$$

where m is the number of landmarks. At each timestep k, robot control inputs are simulated as follows: the distance traveled d, and the change in angle θ_t. The robot's pose is updated at the timestep by

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta s_k \cos(\theta_{k-1} + \Delta\theta_k) \\ \Delta s_k \sin(\theta_{k-1} + \Delta\theta_k) \\ \Delta\theta_k \end{bmatrix}.$$

Subsequently, measurements $z_{k,i}$ to each landmark i are simulated based on the robot's pose using

$$Z_k = \begin{bmatrix} d_0 & \alpha_0 \\ d_1 & \alpha_1 \\ \vdots & \vdots \\ d_m & \alpha_m \end{bmatrix}$$

where

$$d_i = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}$$

$$\alpha_i = \text{atan2}((y_i - y_k), (x_i - x_k))$$

To make the dataset more realistic and to test the performance of multi-agent EKF, noises are added to the data generation. Noise $n_s \sim \mathcal{N}(0, \sigma_s^2)$ and $n_\theta \sim \mathcal{N}(0, \sigma_\theta^2)$ mimic the error of controls and locomotion of the robot, while $n_d \sim \mathcal{N}(0, \sigma_d^2)$ and $n_\alpha \sim \mathcal{N}(0, \sigma_\alpha^2)$ mimic the distance and angular error of measurement.
With noise included, the state update formula is now:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} (\Delta s_k + n_s)\cos(\theta_{k-1}) \\ (\Delta s_k + n_s)\sin(\theta_{k-1}) \\ \Delta\theta_k + n_\theta \end{bmatrix}.$$

and the measurements become:

$$Z_k = \begin{bmatrix} d_0 + n_d & \alpha_0 + n_\alpha \\ d_1 + n_d & \alpha_1 + n_\alpha \\ \vdots & \vdots \\ d_m + n_d & \alpha_m + n_\alpha \end{bmatrix}$$

This simulation tool allows us to generate different trajectories with different noise parameters, which helps to verify and evaluate our multi-agent SLAM algorithm.

C. EKF Algorithm

The standard single-agent EKF serves as the foundation for the algorithm. The EKF SLAM approach combines an estimation of the robot's pose and landmark positions into a vector $\mathbf{x}_t$ at each timestep t. The robot's motion model is represented by

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t)$$

where $\mathbf{u}_t$ denotes the controls (linear motion and angular motion) at time step t. Measurements of landmarks are modeled as

$$\mathbf{z}_t = h(\mathbf{x}_t)$$

which includes the distance and angle to each landmark. At each step, the EKF updates the robot's pose and the map based on the predicted motion and observed measurements, iteratively refining the state estimates. Figure 1 below is the standard algorithm for EKF.

Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**
$\bar{\mu}_t = g(u_t, \mu_{t-1})$
$\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$
$K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$
$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
$\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$
return μ_t, Σ_t

Fig. 1. The standard EKF algorithm.

D. Multi-Agent EKF Algorithm

Here, we discuss both the method from the reference paper and our proposed method in detail.

Paper Reference Method: Now, we discuss the method implemented in [1] in detail. As a reminder, the authors of this paper implement multi-agent EKF such that each agent independently runs EKF, and the state estimates are combined into a global estimate. In particular, we describe the system process below; recall that the goal of this process is to compute weight matrices $\bar{A}_1, \bar{A}_2, ..., \bar{A}_m$ that will be used to calculate a weighted average of the state estimates $\hat{\mathbf{x}}^1_{t|t}, \hat{\mathbf{x}}^2_{t|t}, ..., \hat{\mathbf{x}}^m_{t|t}$.

1) Each agent i computes state estimates $\hat{x}^i_{t|t}$ and covariance matrix $P^i_{t|t}$.
2) Then, after each robot has performed its update step, we collect all of the state estimates to compute "cross-covariance matrices" $P^{ij}_{t|t}$. These are computed given the following equation:

$$P^{ij}_{t|t} = (I_n - K^i_{t+1}H^i_{t+1})(F_t P^{ij}_{t|t} F_t^T + G_t Q G_t^T)(I_n - K^i_{t+1}H^i_{t+1})^T + K^i_{t+1} S_{ij} (K^j_{t+1})^T$$

Note the similarities between this cross-covariance matrix calculation and the covariance matrix computation in the observation update step.
3) Once we have computed these cross-covariance matrices for every combination of agents (i, j), we stack them into a large matrix Σ, which will eventually be used in

the computation of the weight matrices for each agent. The diagonal of this matrix Σ contains elements $P^i_{t|t}$, which is simply the covariance matrix for a given agent i. Here is the definition of Σ:

$$\Sigma = \begin{bmatrix} P^1 & P^{12} & \dots & P^{1m} \\ P^{21} & P^2 & \dots & P^{2m} \\ \vdots & \vdots & \ddots & \vdots \\ P^{m1} & P^{m2} & \dots & P^{mm} \end{bmatrix}$$

4) Finally, we can compute the optimal weight matrices $\bar{A}_1, \bar{A}_2, ..., \bar{A}_m$. We compute them altogether by calculating a single matrix $\bar{A} = \begin{bmatrix} \bar{A}_1 & \bar{A}_2 & \cdots & \bar{A}_m \end{bmatrix}$ as follows:

$$\bar{A} = (e^T \Sigma^{-1} e)^{-1} e^T \Sigma^{-1}$$

Here, note that $e = \begin{bmatrix} I_n & I_n & \cdots & I_n \end{bmatrix}^T \in \mathbb{R}^{nm \times n}$, where n is the number of landmarks and m is the number of agents. This aligns with the dimensions of Σ, which is $nm \times nm$.

5) From above, we find that each individual weight matrix ends up having size $n \times n$, and this is multiplied with the corresponding state vector of size $n \times 1$ and summed up to get a final global state estimate of size $n \times 1$. Here is the final weighted average computation using the weight matrices computed in the previous step:

$$\hat{x}^0_{t|t} = \bar{A}_1 \hat{x}^1_{t|t} + \bar{A}_2 \hat{x}^2_{t|t} + \cdots + \bar{A}_m \hat{x}^m_{t|t} = \sum_{i=1}^{m} \bar{A}_i \hat{x}^i_{t|t}$$

Here, $\hat{x}^0_{t|t}$ denotes the global

Our Proposed Method: For our proposed method, we combined key features from [1] and [2], aiming for a simplified version of the method used in [1]. In our approach, each agent runs the EKF independently, and we combine estimates into a global estimate after each iteration. We define an `Agent` class for each robot to track its own state estimates and covariance matrices. Additionally, we define a `multi_main` function to loop through and perform the predict and update steps for each agent before computing the global estimates for landmarks using a weighted average.

Similar to [1], we compute weights based on covariance matrices for each agent. However, a major difference is that to reduce computation time, we compute a single scalar weight directly from the covariance matrices, whereas [1] computes weight matrices. Another difference is that a given agent in our method does not attempt to track the positions of other agents, only landmarks and its own pose. Figure 2 below is an overview of our proposed multi-agent EKF algorithm.

Now, we discuss our multi-agent EKF algorithm in more detail. Here are the major steps:

1) Perform the predict and update steps on each `Agent`, running the EKF algorithm independently.
2) Initialize a weights matrix to zeros, then iterate through all landmarks and all agents.
3) For each landmark-agent combination, compute the weight by calculating the norm of the column in the agent's covariance matrix corresponding to the landmark. Set the corresponding entry in the weights matrix, `W[ℓ, i]`, to the norm.
4) After iterating through all agents with a fixed landmark, normalize the row in the weights matrix such that the weights for a given landmark sum to 1 across all agents.
5) Once the full weights matrix is obtained, compute a weighted average (a global estimate) of the landmarks. For each landmark, iterate through each agent and obtain the corresponding weight. Then, for agents $a_1, a_2, ..., a_m$ and landmarks $l_1, l_2, ..., l_n$, compute the final estimate of landmark l_i in terms of the weights and individual state estimates $\hat{x}^1_i, \hat{x}^2_i, ..., \hat{x}^m_i$ for landmark l_i as follows:

$$\hat{x}_i = \texttt{W[i, 0]}\hat{x}^1_i + \texttt{W[i, 1]}\hat{x}^2_i + ... + \texttt{W[i, m-1]}\hat{x}^m_i$$

Here, $\hat{x}_i$ denotes the global estimate of landmark l_i for this iteration.

6) Compute a global covariance matrix $\hat{P}$ by naively computing an inverse sum of the inverses of the covariance matrices for each individual agent. For agents with indices 1 through m with covariance matrices $P_1, P_2, ..., P_m$, compute the global covariance $\hat{P}$ as follows:

$$\hat{P} = (P_1^{-1} + P_2^{-1} + ... + P_m^{-1})^{-1} = \left(\sum_{i=1}^{m} P_i^{-1} \right)^{-1}$$

The intuition behind this calculation is that the global covariance will remain small or large if the individual agents' covariances are all small or large. If one agent has a large covariance matrix and the other agent has a small covariance matrix, the global covariance will be even smaller than the second agent's covariance due to the weighting in the weighted average computation of the mean landmark positions.

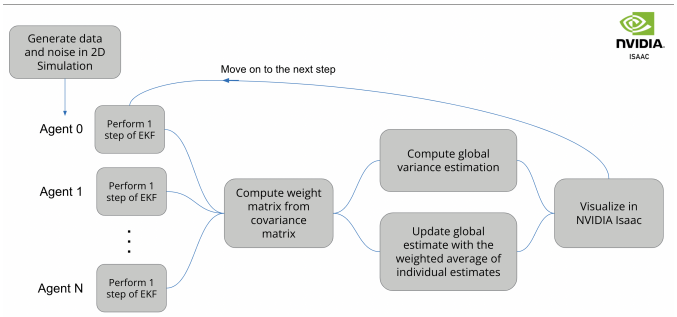

Fig. 2. The multi-agent EKF algorithm.

E. NVIDIA Isaac Sim

To provide a better visualization for our implementation, the team chose NVIDIA Isaac Sim as our 3D visualization tool. NVIDIA Isaac Sim is an extensible robotics simulation platform capable of delivering scalable, photorealistic, and physically accurate high-fidelity simulations. The simulation environment relies solely on Python language interpreter for

scripting but some C++ functions are wrapped into the interpreter for computational efficiency improvement.

In the simulation environment, we initialized 6 ground truth landmarks with their corresponding x and y positions. We introduced two high-fidelity models to simulate multi-agent robot movement: Clearpath Robotics's Theia UGV and a forklift model. Both agents take different control inputs with Gaussian noise introduced and execute the movement command simultaneously. The team believes this 3D visualization provided a better understanding of our multi-EKF implementation. Figures 3 and 4 below are visuals attained in NVIDIA Isaac Sim. We encourage readers to explore the simulation animation on our GitHub repository.

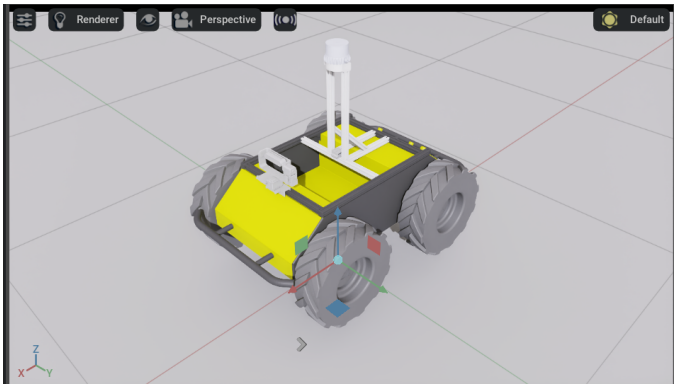

Fig. 3. The Clearpath Robotics's Theia UGV model in NVIDIA Isaac Sim.

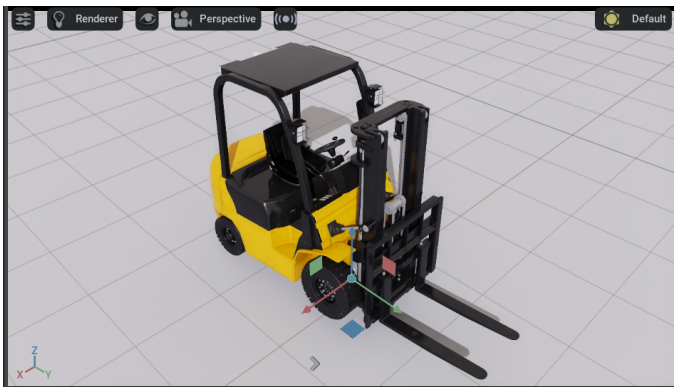

Fig. 4. The forklift model in NVIDIA Isaac Sim.

V. Experiments

Below are 2D plots showing both single-agent and multi-agent EKF trajectories, landmarks, and uncertainties. Compared to the previously mentioned NVIDIA Isaac Sim visualization, it is an alternative way for us to visualize our implementation. The magenta and blue ellipses represent the predicted and updated uncertainties of the robot's position at each time respectively. The red and green ellipses represent the initial and all the updated uncertainties of the landmarks. Different trajectories are applied to different agents to simulate the multi-agent behaviors. Both smaller robot position errors and landmark errors were observed in the multi-agent experiment results.

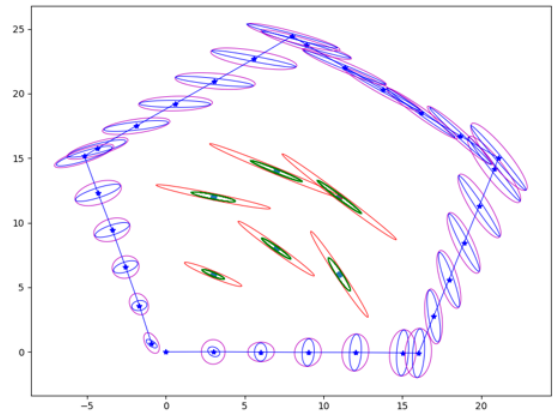

Fig. 5. The single-agent robot trajectory and landmark estimation in 2D plot.

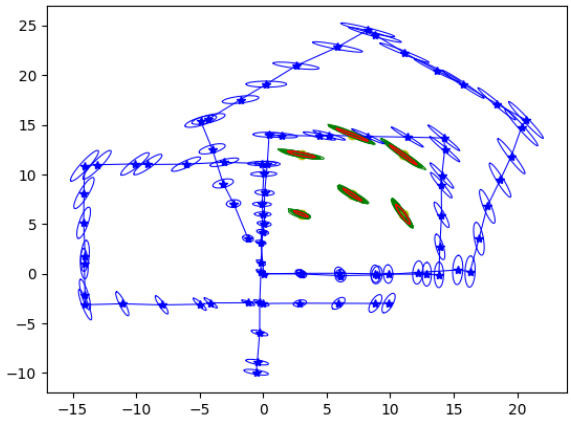

Fig. 6. The multi-agent robot trajectory and landmark estimation in 2D plot.

VI. Results

We compared the performance of our multi-agent SLAM frameworks against single-agent EKF, primarily focusing on the accuracy of predicted agents and landmarks locations. We utilized metrics such as covariance analysis, Euclidean, and Mahalanobis distance to quantify the discrepancy between predicted and ground truth positions. By analyzing these metrics, we aim to assess the effectiveness of EKF in accurately estimating agent positions. For the results, our multi-EKF solver works even better in the varying measurement noise case.

A. Multi-agent with Constant Measurement Noise

1) Covariance matrix: In comparing the covariance matrices of multi-agent EKF results, we employed the Fisher

information matrix to calculate the global covariance matrix. By utilizing the trace and determinant of the matrix, we evaluated and compared the covariance across the agents. This approach allowed us to assess the overall uncertainty and variability within the multi-agent EKF framework, providing insights into the efficacy and reliability of the estimation process. From the tables below, we can see that the global covariance matrix has a smaller trace than any single agent. We also observed with more agents initialized, the covariance matrix estimates improve.

TABLE I
TRACE AND DETERMINANT OF COVARIANCE MATRIX FOR 2 AGENTS

Agent	Variance (Trace)	Variance (Determinant)
0	0.8437586991701194	$5.956457947843978 \times 10^{-46}$
1	0.7538095066088809	$2.776141962880351 \times 10^{-47}$
Global	0.37933236097997064	$3.418390817798469 \times 10^{-51}$

TABLE II
TRACE AND DETERMINANT OF COVARIANCE MATRIX FOR 3 AGENTS

Agent	Variance (Trace)	Variance (Determinant)
0	0.8437586991701194	$5.956457947843978 \times 10^{-46}$
1	0.7538095066088809	$2.776141962880351 \times 10^{-47}$
2	0.4489979232015054	$4.30099432473347 \times 10^{-49}$
Global	0.18491298280224647	$8.532633996729576 \times 10^{-55}$

2) Euclidean error and Mahalanobis error: Besides the trace and determinant of the covariance matrix, we also used the Euclidean error and Mahalanobis error to represent the Multi-Agent EKF result. As the plots below show, our Multi-Agent EKF shows overall better accuracy than any of the Single-Agent results.

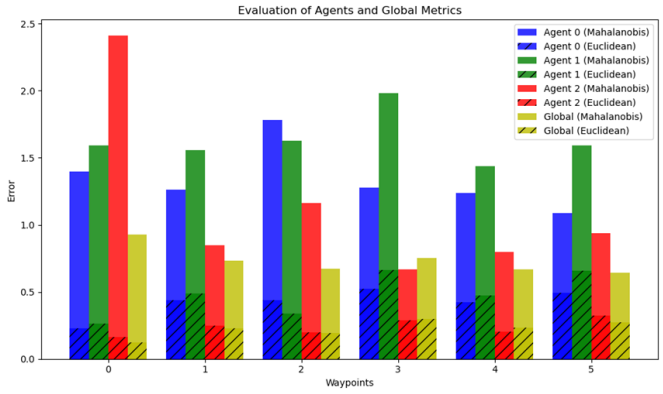

Fig. 7. The agents' Euclidean error and Mahalanobis error.

B. Multi-agent with Varying Measurement Noise

In the last section, we applied the same amount of noise to all the agents. In real-world situations, the noises from sensors will vary with distance, sensor conditions, temperatures, and many other parameters. We want to test the accuracy and robustness of Multi-Agent in different noise environments. We applied three agents with different noise levels, agent 0 with the lowest noise and agent 2 with the highest noise.

1) Covariance matrix: Similar to the last section, we compared the trace and determinant of the covariance matrix. From the tables below, we can see that with more agents initialized the covariance matrix estimates improve. Also, we can observe the global variance of varying noise smaller than the constant noise, which somehow proved that the Multi-Agent EKF might deal with noise conditions closer to real-world situations

TABLE III
TRACE AND DETERMINANT OF COVARIANCE MATRIX FOR 2 AGENTS

Agent	Variance (Trace)	Variance (Determinant)
0	0.7528768622565092	$5.7010584630081 77 \times 10^{-46}$
1	0.952835086056161	$6.86727828673139 \times 10^{-48}$
Global	0.08939531558406963	$5.537786301195614 \times 10^{-53}$

TABLE IV
TRACE AND DETERMINANT OF COVARIANCE MATRIX FOR 3 AGENTS

Agent	Variance (Trace)	Variance (Determinant)
0	0.7528768622564067	$5.701058462930158 \times 10^{-46}$
1	0.9528350860561783	$6.867278286755397 \times 10^{-48}$
2	1.1908377138346975	$7.239007224284874 \times 10^{-48}$
Global	0.048934468335938806	$3.1516364283615658 \times 10^{-56}$

2) Euclidean error and Mahalanobis error: Similar to the last section, we applied the Euclidean error and Mahalanobis error to both Multi-Agent and Single-Agent. The plots below show that in different noise conditions, Multi-Agent EKF still holds a better overall accuracy than Single-Agent one.

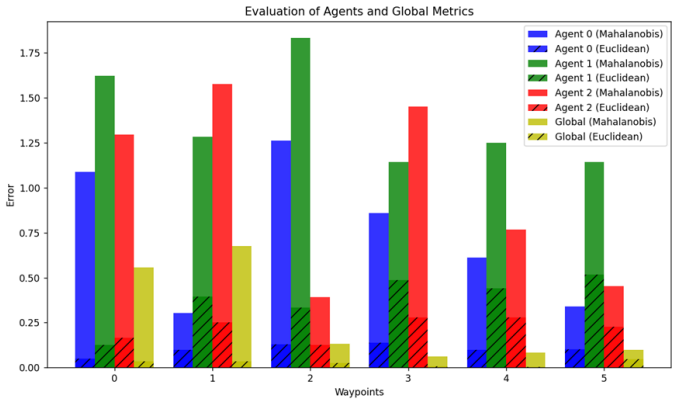

Fig. 8. The agents' Euclidean error and Mahalanobis error.

VII. FUTURE WORK

There is a lot of extension work we can do based on this project. One direction we would want to explore further is the implementation of the method described in [1], which we also describe in detail in Section IV. We would compare these results to both the single-agent estimation error as well as the error from our proposed multi-agent EKF method. We

could also investigate the scalability of the proposed multi-agent EKF approach as the number of agents and landmarks increases. This could involve analyzing the computational complexity and memory requirements, as well as proposing strategies to improve scalability, such as decentralized or hierarchical architectures.

Another possible approach is to extend the multi-agent EKF approach to handle 3D environments, which would involve estimating the 3D positions of landmarks and the 6D poses (position and orientation) of the agents. This extension could be particularly relevant for applications involving aerial robots or robots operating in complex 3D environments. What's more, we could compare the performance of the multi-agent EKF approach with other state-of-the-art SLAM techniques, such as graph-based optimization methods or deep learning-based approaches. This could provide insights into the strengths and weaknesses of different methods and potentially lead to hybrid or improved algorithms.

VIII. Conclusions

In this paper, we proposed a multi-agent EKF solution that is capable of providing more accurate landmark estimations. We discuss some of the state-of-the-art related research done in this domain. A detailed explanation was given to discuss the development of our multi-EKF solver. Our multi-EKF algorithm framework was tested with various datasets and visualized in 2D plots and 3D simulation environments. The results from our experiment showed promising outcomes and benefits in multi-agent collaborations. Lastly, this paper serves as an inroad to more robust multi-agent SLAM solvers in future research.

Acknowledgment

The team would like to acknowledge Dr. Michael Kaess for the insightful lectures throughout the semester, as well as teaching assistants Easton Potokar and Jinyun Xu for debugging homework problems and addressing our project's logistical questions. Lastly, we would also like to acknowledge the Computational Engineering and Robotics Lab (CERLAB) PhD student Grant Metts for the algorithm framework assistance.

References

[1] T. Sasaoka, I. Kimoto, Y. Kishimoto, K. Takaba, and H. Nakashima, "Multi-robot SLAM via information fusion extended Kalman filters," IFAC-PapersOnLine, vol. 49, no. 22, pp. 303–308, 2016.

[2] E. Howe and J. Novosad, "Extending SLAM to multiple robots," March 2005.