Final Report

# Efficient Multi-Robot SLAM in a Labyrinth Environment: A Decentralized Approach Using Algebraic Connectivity Augmentation

Abdirakhman Onabek, Ramazan Andarbayev, Bauyrzhan Askarov

Final report of

**BSc in Robotics Engineering**

**Supervisor:** Zhanat Kappassov

Department of Robotics and Mechatronics
Nazarbayev University

May 5, 2025

# Table of Contents

# Abstract

***This project focuses on implementing a multi-robot Simultaneous Localization and Mapping (SLAM) framework using multiple Turtlebots in a labyrinth environment. The goal is to demonstrate that multiple robots working together can map a complex environment more efficiently than a single robot. Using the C-SLAM framework, a decentralized approach, the project addresses key challenges like communication efficiency, loop closure detection, and pose graph optimization. Preliminary work includes learning ROS, connecting LIDAR sensors, and generating point clouds.***

Keywords: Multi-robot SLAM, Swarm SLAM, Decentralized SLAM, Loop Closure Detection, Pose Graph Optimization, ROS, LIDAR, Turtlebots, Labyrinth Mapping

# Chapter 1: **Introduction**

In robotics, Simultaneous Localization and Mapping (SLAM) enables robots to autonomously map their environment while determining their position within it. Traditionally, SLAM is performed using a single robot, but recent advancements have shown that employing multiple robots can significantly improve the efficiency of mapping. This project explores the hypothesis that multi-robot SLAM can reduce the time required to map complex environments, such as labyrinths, by leveraging collaboration between multiple Turtlebots.

The project will utilize the C-SLAM framework, a decentralized approach where each robot processes its local environment while intermittently sharing data with nearby robots. This avoids reliance on a central server, making the system more resilient to communication failures and scalable to larger robot teams. C-SLAM also incorporates novel techniques for loop closure detection and pose graph optimization, ensuring that maps created by individual robots can be accurately merged into a global map.

To validate the system, experiments will be conducted to compare the performance of single-robot and multi-robot SLAM, evaluating key metrics such as mapping time, map accuracy, and communication overhead. The project builds on previous studies that highlight the challenges of multi-robot coordination, including synchronization, data sharing, and maintaining global consistency.

# Chapter 2: **Literature Review**

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics, allowing robots to map an environment while simultaneously determining their location within it. Traditional SLAM techniques have been well-studied for single robots, but as robotic systems evolve towards multi-robot configurations, the complexities increase due to coordination, data sharing, and the need for accurate map merging. Multi-robot SLAM (MR-SLAM) has the potential to reduce mapping time and improve overall performance by leveraging collaboration between multiple agents. However, this also introduces challenges in communication, data synchronization, and maintaining global consistency. This literature review examines several MR-SLAM methodologies and frameworks to identify key trends and solutions in the field.

**General Challenges in Multi-Robot SLAM**
In MR-SLAM, one of the primary challenges is the alignment of local maps created by individual robots, known as the problem of determining relative poses between robots. Solutions to this challenge include rendezvous-based map merging, where robots exchange mapping data upon meeting, and relative localization, where shared observations are used to improve pose estimates between robots. Techniques like particle filters and batch updates help in maintaining consistency across the global map. Additionally, loop closure—the detection of a previously visited location can be addressed through visual or feature-based methods that match current observations with prior data. Methods like FastSLAM and iSAM are employed to handle real-time optimization, while sensor fusion techniques are essential when dealing with heterogeneous robots and sensors [1].

**Method 1: TSLAM - Team SLAM Framework**
The TSLAM framework presents a practical approach to MR-SLAM by leveraging existing single-robot SLAM techniques and extending them to a team of robots. In this system, each robot performs local mapping independently, using its onboard sensors and SLAM algorithms, while a central workstation handles the coordination and optimization of the global map. A key feature of TSLAM is the use of ORB feature detection to filter and transmit only the most relevant data to reduce bandwidth usage. The central workstation uses the g2o library for pose graph optimization to ensure global consistency across the team of robots.

One of the major innovations in TSLAM is the introduction of a trust factor ($\theta$), which smooths out the transition between pre- and post-optimization poses, ensuring that robots remain operational even when disconnected from the network. The framework also handles network delays and synchronization challenges effectively, ensuring consistent and efficient mapping [2].

**Method 2: Swarm SLAM**
Swarm SLAM represents a decentralized approach to MR-SLAM, where robots operate with limited local information and interact only with nearby robots. This approach is characterized by fault tolerance and the absence of a centralized controller, making it particularly useful in environments where communication infrastructure is limited or unreliable. The swarm-specific behaviors, such as aggregation and dispersion, enable efficient exploration of unknown environments, often relying on simple exploration schemes like random walks or potential fields.

Swarm SLAM also emphasizes the use of abstract maps and distributed localization, which,

while less precise than centralized methods, can be sufficient for swarm-level navigation. Data sharing is decentralized, often relying on mobile ad-hoc networks for fault tolerance, and map merging is simplified by focusing on essential map features rather than detailed environments [3].

**Method 3: C-SLAM (Collaborative SLAM)**
The C-SLAM algorithm builds on the concept of loop closure, using pose graph sparsification to optimize map consistency across multiple robots. C-SLAM introduces a two-stage process for identifying loop closures. In the first stage, compact local descriptors are shared between robots to identify candidate locations for loop closure, and in the second stage, more detailed descriptors are used to confirm these matches.

C-SLAM's back-end uses the Distributed Gauss-Seidel (DGS) method for pose estimation, which allows the system to operate without relying on a centralized server. The system prioritizes algebraic connectivity augmentation, ensuring that the pose graph remains optimized with minimal communication between robots. This approach is particularly suited for large-scale environments where communication constraints are significant [4].

**Method 4: Karto SLAM**
Karto SLAM is a graph-based SLAM algorithm that uses LIDAR and odometry data to build a precise map of the environment. Unlike swarm or decentralized approaches, Karto SLAM relies on centralized processing for graph optimization and performs exceptionally well in environments where communication infrastructure is available. The system integrates with the Dynamic Windows Approach (DWA) for real-time path planning and obstacle avoidance, which allows it to navigate dynamic environments efficiently.

Karto SLAM has been tested in both simulated and real-world environments, demonstrating high accuracy with an average relative error of 1.22%. However, the system's reliance on LIDAR and centralized processing makes it less suitable for low-cost or decentralized robotic systems [5].

**RTAB-Map Framework**
RTAB-Map is a versatile SLAM framework that supports a wide range of sensors, including stereo cameras, RGB-D cameras, and LIDAR. The framework is particularly noted for its memory management features, which allow it to handle large datasets by selectively discarding less relevant data. RTAB-Map integrates odometry inputs to enhance navigation in challenging environments, such as those with textureless surfaces where visual-based SLAM may struggle.

RTAB-Map is highly adaptable to different sensor configurations, but the performance is closely tied to the choice of sensors. For example, RGB-D cameras excel at obstacle detection, while stereo cameras provide superior localization accuracy. The framework has been evaluated on various datasets, including KITTI and TUM, highlighting the importance of sensor selection and algorithm architecture for precise mapping [6].

Overall, while centralized systems like TSLAM and Karto SLAM offer high accuracy and consistency and require reliable communication infrastructure, decentralized approaches such as Swarm SLAM and C-SLAM emphasize scalability, fault tolerance, and minimal communication, making them suitable for large, complex environments. Our hypothesis that multiple robot SLAM will map a labyrinth faster than a single robot is supported by the principles outlined in these studies. Specifically, decentralized methods allow for faster exploration by distributing tasks across multiple robots, while centralized methods ensure high map accuracy when communication is available.

# Chapter 3: **Methodology**

The choice of SLAM framework depends heavily on the project's requirements for scalability, accuracy, and fault tolerance. In environments like the labyrinth in our project, a hybrid approach that combines decentralized exploration with centralized map optimization may offer the best trade-off between speed and accuracy. Our primary hypothesis is that multiple robot SLAM will map a labyrinth faster than a single robot, and the following methodology outlines how we will test this hypothesis using metrics and experimental designs drawn from previous research.

**Experimental Setup**
We will use a team of Turtlebots equipped with LIDAR sensors to perform SLAM in a labyrinth environment. The experiment will involve both single-robot SLAM and multi-robot SLAM, where two or more robots collaborate to map the environment. The SLAM algorithm will be a hybrid approach: robots will perform local decentralized mapping and will synchronize their maps using a centralized graph optimization process at regular intervals, similar to the TSLAM framework [2].

Robots will communicate via Wi-Fi, but to simulate real-world communication challenges, we will introduce intermittent connectivity by simulating disconnections, as suggested in [2]. During disconnections, robots will map independently, and once reconnected, they will synchronize their local maps with the global map.

**Metrics for Evaluation**
The following key metrics, drawn from the literature, will be used to evaluate the performance of both single- and multi-robot SLAM:

- Mapping Time: The total time taken to complete the mapping of the labyrinth. This is the primary metric for testing our hypothesis. We expect multi-robot SLAM to reduce mapping time compared to a single robot. Mapping time will be measured from the start of the mapping process until the entire labyrinth is mapped. This metric is highlighted in [3], where the focus is on reducing time to completion as a goal for multi-robot SLAM. We will record both the time taken by a single robot and the time taken by multiple robots to map the same environment.

- Accuracy of the Map: The quality of the final map will be evaluated by comparing it to a ground truth map of the labyrinth. Accuracy will be assessed using relative error in mapped features, following the example of the Karto SLAM experiment [5], which measured mapping accuracy with an average relative error of 1.22%. We will similarly calculate the relative error for each experiment, with lower error indicating higher accuracy. We will also use techniques such as loop closure detection to refine the maps, ensuring consistency and reducing cumulative error over time [1].

- Map Completeness: Completeness refers to the percentage of the environment that has been mapped. This metric is important for ensuring that all robots have explored the entire labyrinth, as noted in [3]. We will measure the proportion of the labyrinth mapped by both single and multiple robots, with the goal being to achieve near-complete coverage in both cases. Completeness will be determined once a robot or group of robots has explored all predefined sections of the labyrinth.

- Bandwidth and Communication Latency: As communication is a critical factor in multi-

robot SLAM, we will monitor data transmission rates and latency in map synchronization. Similar to the TSLAM framework, which addressed network delays and bandwidth limits by minimizing data transmission through key feature points [2], we will track the amount of data transmitted between robots and the central server and any resulting latency in map merging. We will aim to minimize communication overhead while maintaining mapping accuracy. We will also try to minimize centralized data processing ad discussed in [3].

- Fault Tolerance and Resilience: Fault tolerance will be evaluated by measuring how well the system continues to function when communication is interrupted. Inspired by the Swarm SLAM approach [3], where robots can operate without global supervision, we will test resilience by simulating communication failures and disconnections between robots. The metric here is the system's ability to continue mapping during disconnection and the recovery time needed to re-synchronize after re-connection.

- Scalability: We will assess how well the system performs as the number of robots increases, taking into account the computation time and resource footprint for larger teams. Similar to metrics in Swarm SLAM [3], we will track how the system scales as we increase the number of Turtlebots and whether the mapping process remains efficient without causing significant delays or errors.

- Loop Closure Detection Rate: In multi-robot SLAM, loop closure is critical for correcting pose estimates and improving the global map. The C-SLAM method emphasizes the importance of detecting loop closures between multiple robots by sharing descriptors for place recognition and registration [4]. We will measure how frequently loop closures are detected and how they contribute to improving the accuracy of the map. We expect higher rates of loop closure detection in the multi-robot setup, leading to better overall map accuracy.

- Robot Resource Utilization: The Dynamic Windows Approach (DWA) [5] emphasizes real-time decision-making during path planning and obstacle avoidance. We will monitor robot CPU and memory utilization to ensure that robots can handle SLAM computation in real-time without significant delays. This will also provide insights into the feasibility of scaling the system up to include more robots.

**Experimental Procedure**

Single Robot SLAM: One Turtlebot will navigate and map the labyrinth using the hybrid SLAM framework. This experiment will serve as the control for evaluating the hypothesis. The robot will be allowed to explore the environment autonomously, and we will measure the time, map accuracy, and completeness of the final map.

Multiple Robot SLAM: Two or more Turtlebots will simultaneously explore the labyrinth. This experiment will evaluate the effects of collaboration on mapping time, accuracy, and efficiency. We will conduct multiple runs with varying numbers of robots (2, 3, 4, etc.) to assess scalability.

Disconnection and Fault Tolerance Testing: In both single and multi-robot setups, we will introduce intermittent communication failures to simulate real-world conditions. Robots will continue mapping independently during disconnection, and upon reconnection, we will measure how quickly and accurately the local maps are merged into a consistent global map.

Performance Evaluation: For each experiment, we will calculate the aforementioned metrics: mapping time, accuracy, completeness, bandwidth usage, fault tolerance, scalability, loop closure detection rate, and resource utilization. The results will be compared between single- and multi-robot setups to evaluate the impact of collaboration on SLAM performance. The Swarm SLAM framework will be tested in both simulated and real-world environments. We

will use public datasets such as KITTI and M2DGR to test the system's performance in varied and challenging environments. For the real-world experiment, we will deploy 2 and more Turtlebots in an indoor labyrinth and evaluate the metrics mentioned above.

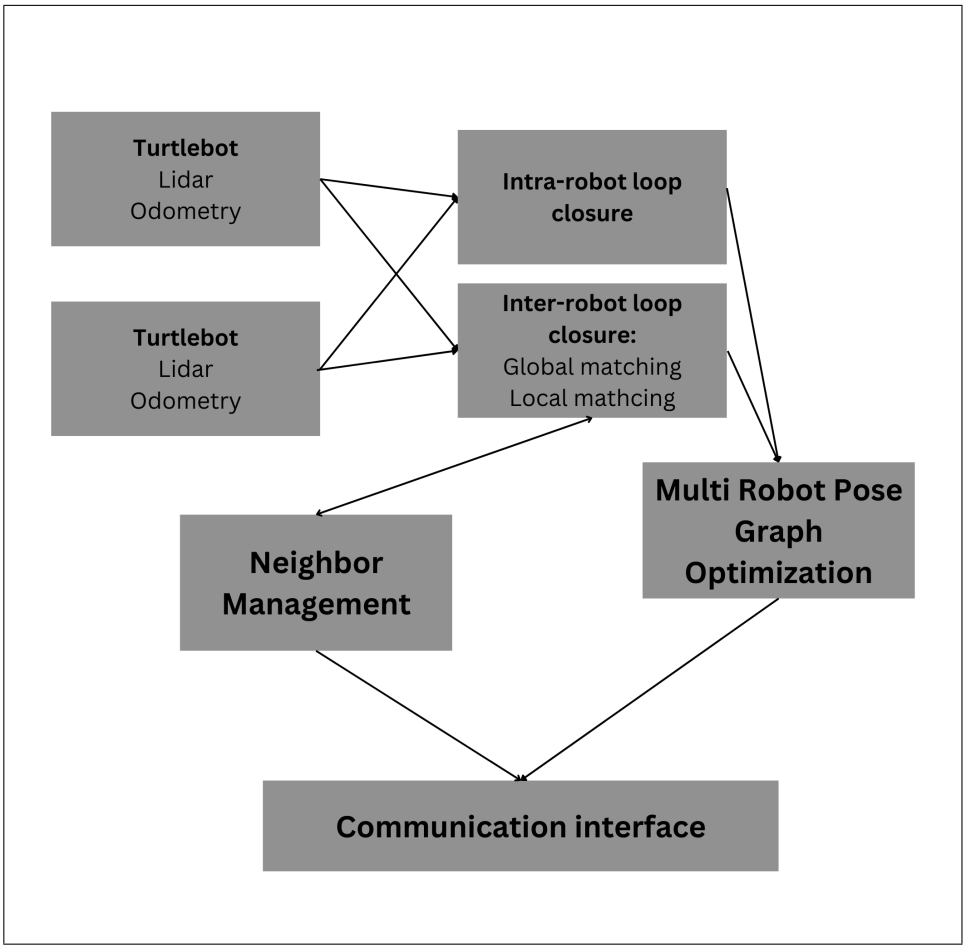



Figure 3.1: Example pipeline for 2 Turtlebots inspired by paper [3].

We will employ a decentralized collaborative SLAM framework (Figure 3.1) as our primary method to perform multi-robot SLAM.

C-SLAM couples the geometric perception of the environment (using sensors such as LIDAR) with state estimation (through odometry), ensuring that robots do not rely on centralized systems or global knowledge.

**Front-End: Inter-Robot Loop Closure Detection** One of the core components of C-SLAM is the detection and computation of inter-robot loop closures, which serve as key points where maps can be merged to form a global understanding of the environment. Loop closures in C-SLAM are handled through a two-stage process:

**Stage 1 (Global Matching)**: Compact global descriptors (e.g., from LIDAR scans or images) are shared between robots. These descriptors, such as ScanContext for LIDAR and CosPlace for images, are used to recognize places that two or more robots have visited. The cosine similarity between the global descriptors is computed to find candidates for loop closures [3].

In our experiments, we will configure the system to exchange only relevant descriptors between robots within local communication range, ensuring minimal bandwidth usage. Once a candidate is identified, it is forwarded to the second stage.

**Stage 2 (Local Matching)**: For each loop closure candidate, high-size local descriptors such as point clouds are shared to compute the geometric registration between scans or images, forming the actual loop closure. This ensures that the robots' maps are aligned at common locations.

We will evaluate loop closure detection as a critical performance metric, tracking the frequency and accuracy of detected loop closures across both single-robot and multi-robot SLAM setups. Loop closure prioritization will be handled using a spectral approach that optimizes algebraic connectivity, as discussed in the graph sparsification section below.

**Back-End: Pose Graph Optimization** After detecting loop closures, C-SLAM combines the odometry, intra-robot, and inter-robot loop closure measurements into a pose graph, which represents the spatial relationships between different locations. The goal of the back-end is to optimize the pose graph to provide the best estimate of robot poses and map consistency.

In our system, pose graph optimization will be performed in a decentralized manner. One of the robots in the communication range will be dynamically selected as the leader to handle the optimization. This robot will receive loop closure data from the other robots and perform the optimization, sharing the results back to all participating robots. The Graduated Non-Convexity (GNC) solver will be used to optimize the pose graph, ensuring robustness to outliers [3].

**Graph Sparsification for Communication Efficiency** One of the primary innovations of C-SLAM is its use of graph sparsification to reduce the communication overhead of inter-robot loop closure detection. Rather than sharing all potential loop closure candidates between robots, c-SLAM uses a spectral prioritization technique to select the most valuable candidates based on their contribution to the algebraic connectivity of the pose graph. This ensures that the most informative loop closures are processed first, while redundant or less informative matches are ignored.

We will implement algebraic connectivity maximization to prioritize which loop closures to compute, based on the second-smallest eigenvalue ($\lambda 2$) of the pose graph Laplacian [3]. The higher the algebraic connectivity, the more accurate the map will be.

Overall, by employing a hybrid approach that balances decentralized exploration with centralized optimization, and by using established metrics for performance evaluation, we aim to demonstrate the advantages of multi-robot SLAM in terms of both efficiency and accuracy.

# Chapter 4: **Design and implementation**

To begin with our project, we implemented a SLAM mapping algorithm in the Gazebo environment for one TurtleBot, enabling autonomous exploration and map generation using its laser scanner and odometry data (Figure 4.1). Utilizing the files we already had from one turtlebot, we created custom launch files that initialize two independent SLAM nodes and deploy two TurtleBots in Gazebo, each beginning SLAM from different starting positions on the same map. However, the problem has arisen in terms of the same node publishing in tf node, which could not load the mapping for two turtlebots.

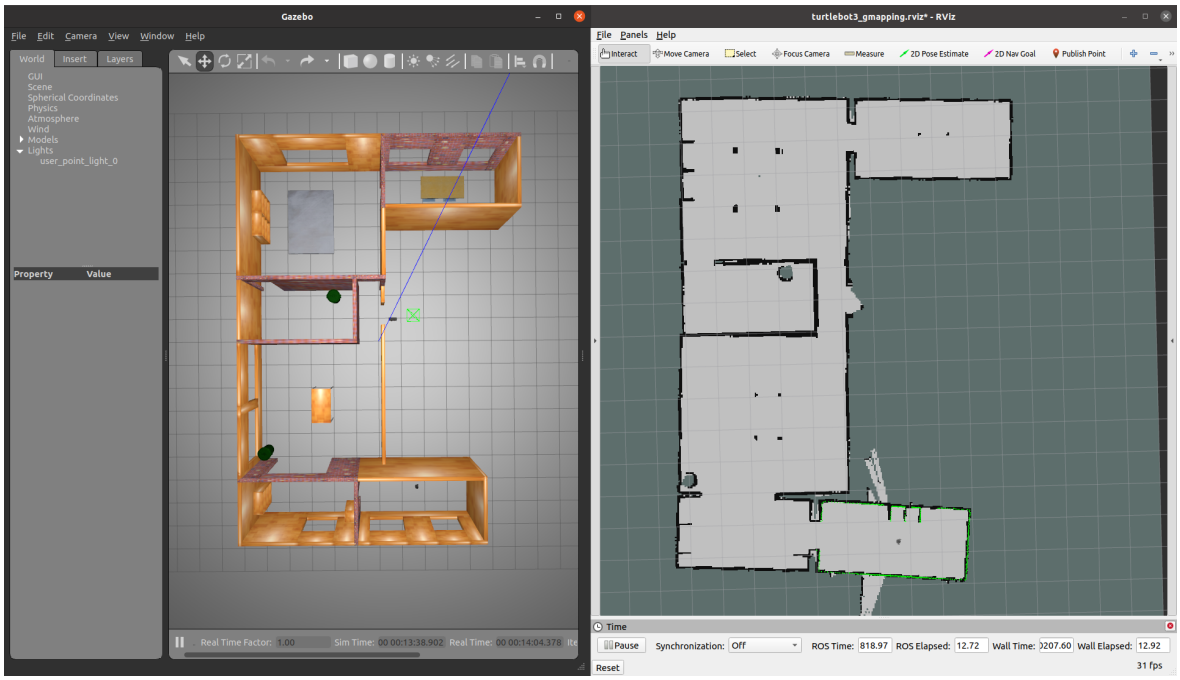



Figure 4.1: SLAM simulation for one TurtleBot.

To resolve this error and set up a two-TurtleBot SLAM system with map fusion, we used resources from the CollaborativeSLAM repository. This repository provides a comprehensive framework for multi-robot SLAM using TurtleBot3 robots in a simulated Gazebo environment. The launch files were adjusted for our use to run the SLAM Gmapping algorithm, a different one from what we should do but utilized for test purposes of the mapping system on two Turtlebots to obtain two maps from each robot (Figure 4.2).

For dynamic map merging, each robot publishes its map under `<robot_namespace>/map`, with a consistent topic name across all robots. Two modes are supported: *known initial positions* and *unknown starting points*.

- With *known positions*, each robot's starting location is specified (`<robot_namespace>/map_merge/init_pose`), allowing precise alignment. Missing parameters mean the robot's map won't be merged.
- With *unknown positions*, only the map topic is required, and transformations are estimated via feature matching, relying on overlapping areas. This mode is flexible but less precise if overlap is limited, and its computation frequency can be adjusted with the `estimation_rate` parameter.

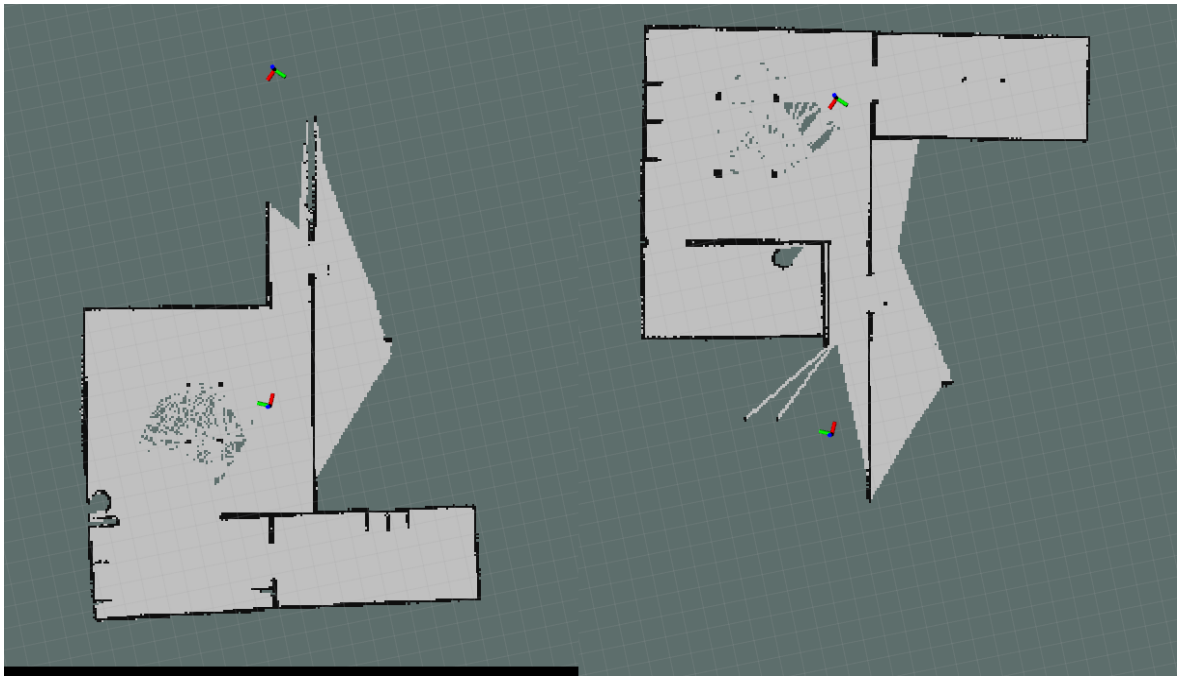


Figure 4.2: Two maps from each robot slam

Using the information from the CollaborativeSlam repository's map merging method, the final map is obtained and can be seen in Figure 4.3. One thing should be noted that so far we have not yet completed the autonomous exploration algorithm and used manual control to map the surrounding area for multiple turtlebots. Thus, the next step of our project was to add an exploration algorithm.

The exploration method is built upon fundamental mathematical concepts and algorithms, many of which are derived from the slam_toolbox library. This library offers a comprehensive set of pre-existing functions for tasks such as mapping, localization, and navigation, which are essential for autonomous exploration. This library handles critical tasks such as managing the occupancy grid map, calculating transformations between coordinate frames, and enabling the robot to localize itself within the map. The method imports these functionalities and applies them to identify unexplored regions, plan safe paths, and navigate efficiently, leveraging the robust and well-tested capabilities already provided by the slam_toolbox. In the following sections, we delve into the specific methods utilized from this library and their role in enabling robust exploration and mapping.

The exploration nodes enable a robot to autonomously navigate and map an unknown environment by combining occupancy grid mapping, gradient analysis, and coordinate transformations. The environment is modeled as a 2D occupancy grid, where each cell is assigned a value: 0 for free space, 100 for obstacles, and $-1$ for unexplored regions. To detect frontiers between explored and unexplored areas, the method uses image gradients, a mathematical tool that measures how values change spatially in the grid. Gradients are computed using operators like the Sobel kernel, which is a simple mathematical matrix designed to highlight edges (transitions) in an image. The Sobel kernel for the $x$-direction is $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, while for the $y$-direction it is $G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$. These kernels approximate the partial derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$, where $I(x, y)$ represents the grid. The gradient magnitude, $\sqrt{I_x^2 + I_y^2}$,

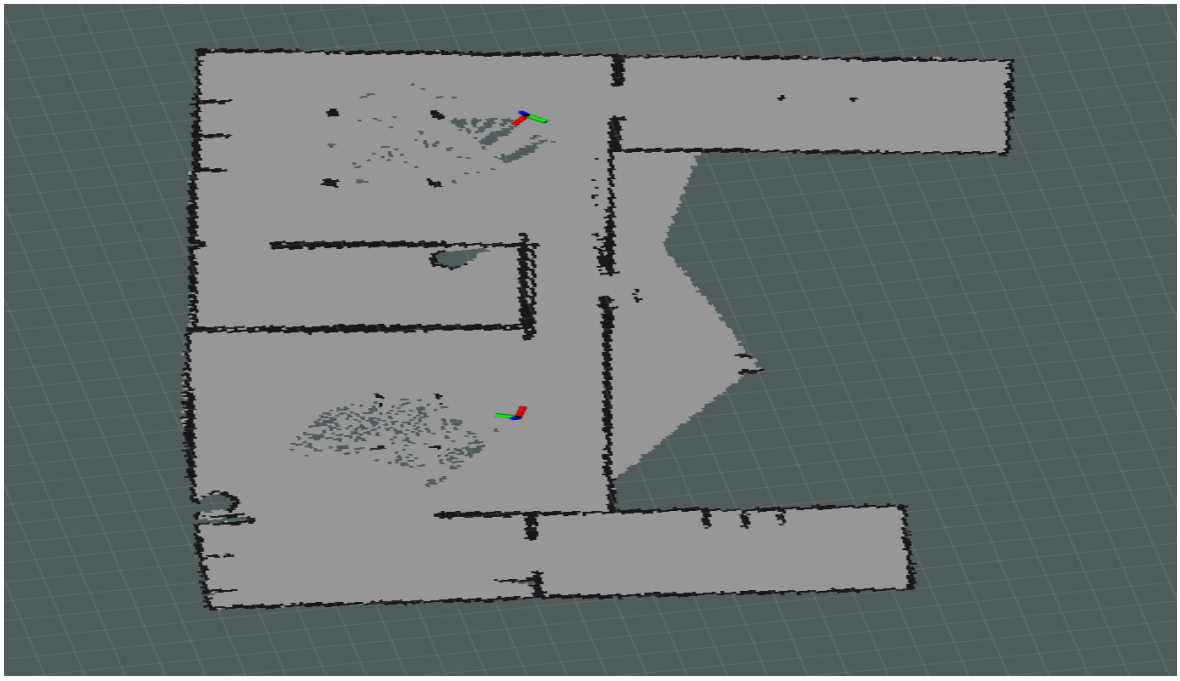


Figure 4.3: Two maps merged into one

shows the strength of transitions, and the gradient direction, $\arctan\left(\frac{I_y}{I_x}\right)$, provides the orientation of these changes. High-gradient regions indicate transitions between unexplored and free areas, which are ideal candidate locations for the robot to explore next.

To move toward these frontiers, the robot uses coordinate transformations to relate positions in different frames of reference, such as the map frame, the robot's local frame, and the goal frame. Each pose is represented in homogeneous coordinates, $\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, and transformations are applied using a matrix $\mathbf{T} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix}$, where $t_x$ and $t_y$ are translations and $\theta$ is the rotation. This transformation allows the robot to compute the target pose in the global map frame while accounting for its own position. Once a target pose is identified, the robot ensures it is safe and navigable by filtering out high-gradient areas that correspond to obstacles or inaccessible regions using a global costmap. This costmap assigns higher values to risky areas, ensuring that the robot avoids collisions while selecting exploration goals. By combining gradient-based frontier detection, safe path planning, and efficient goal selection, the exploration method enables the robot to autonomously and systematically explore its environment.

The experiment began with the initialization of two Turtlebots at distinct spawn points, as illustrated in figure 4.4. These spawn points were chosen strategically to ensure that the robots started in separate regions of the environment, minimizing overlap in their initial exploration paths. The setup reflects a collaborative exploration scenario where multiple robots contribute to building a shared map.

As the Turtlebots moved, their paths were recorded and visualized using odometry data. The trajectory of each robot clearly demonstrates their exploration of the environment. In the visualizations (figure 4.5), we can observe the robots' current poses at various points along their paths, the progression of the odometry data, which highlights the efficient coverage of the area by each robot, and the path history, which provides insight into the strategies employed


Figure 4.4: Exploration initialization of two turtlebots

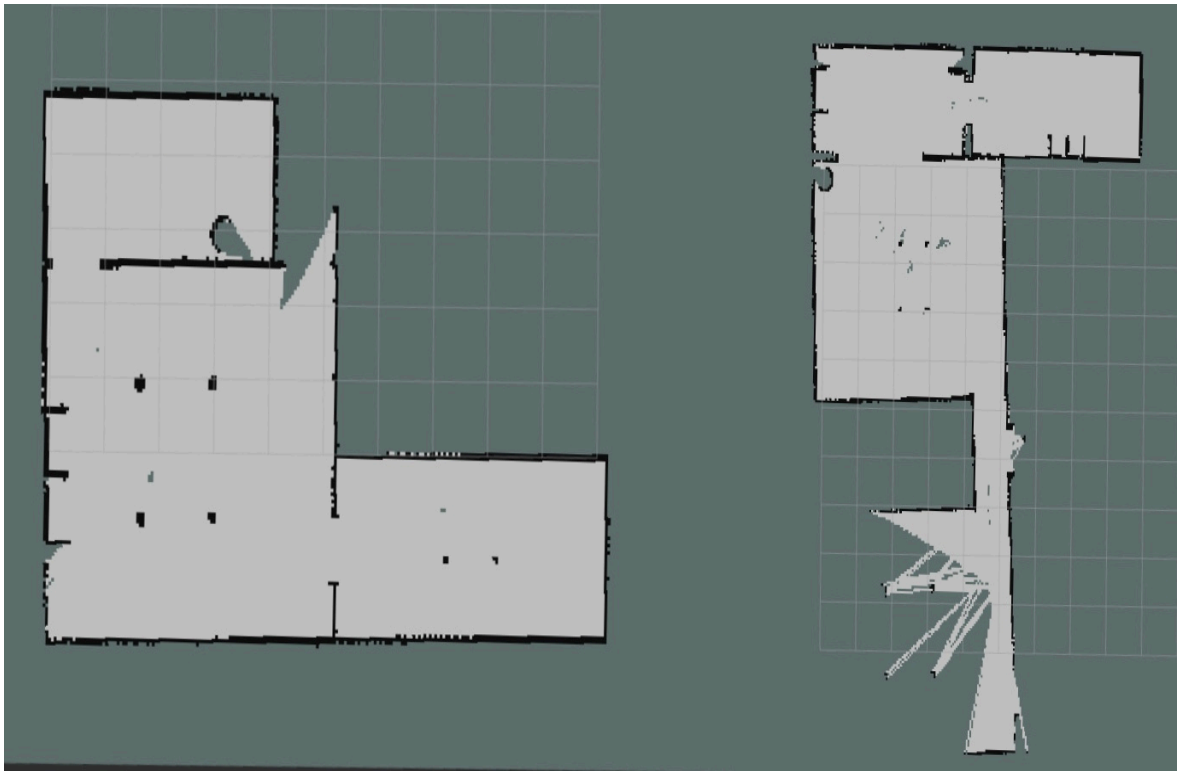


Figure 4.5: Exploration map of two turtlebots

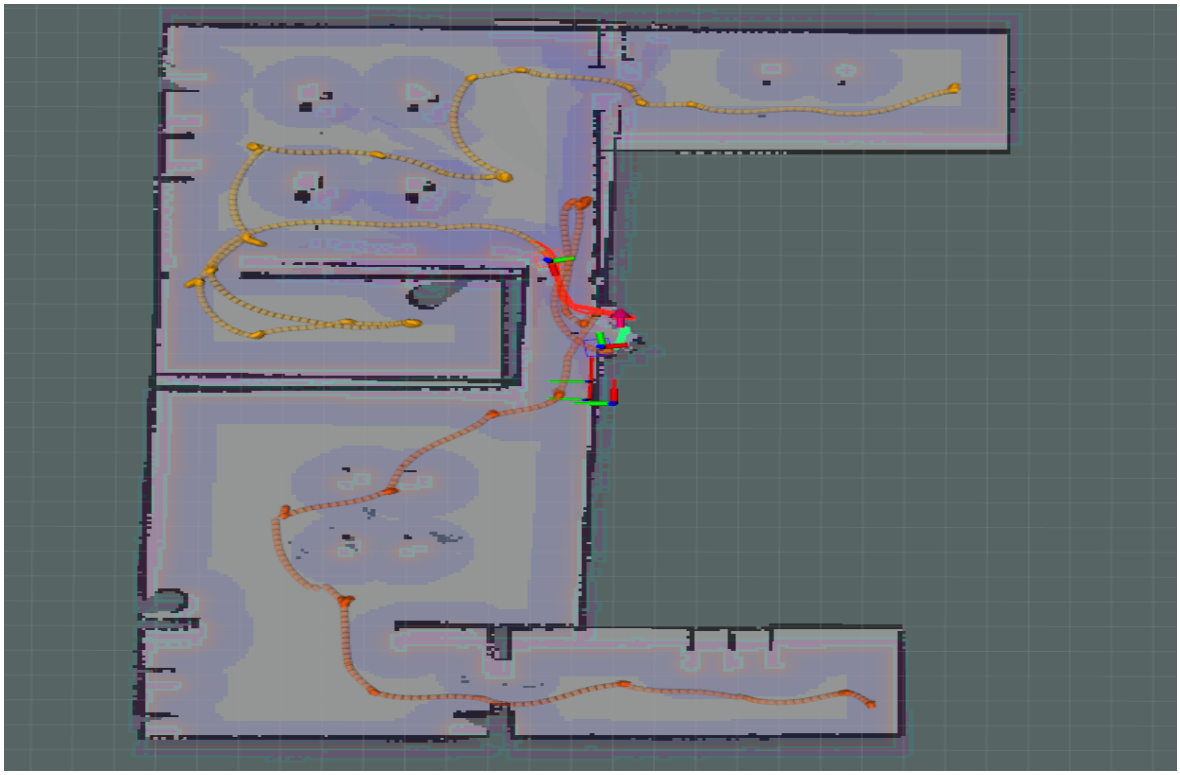


Figure 4.6: Exploration completion and map merging for two turtlebots

by the robots to avoid obstacles and navigate around the environment. To navigate to the identified frontiers, the robots performed coordinate transformations between their local reference frames and the global map frame. The final maps generated by the two turtlebots match with the generated map for one turtlebot (Figure 4.6).

We have been able to set up a two-TurtleBot SLAM system, using ROS tools like Gazebo and RVIZ and resources from the Collaborative_SLAM repository. Each TurtleBot independently generates a map, which is dynamically merged based on either known initial positions or feature matching. We also integrated message transformation from a LIDAR sensor and odometry in the Gazebo to produce a point cloud, enhancing our mapping accuracy. Regular feedback from our supervisor, Professor Matteo Rubagotti, and his Research Assistant has guided our approach and improved our project development.

Currently, we are focusing on refining our algorithm for matching the maps accurately to ensure that the TurtleBots in the simulation do not encounter issues such as overlapping or getting stuck. In a multi-robot SLAM system, one of the fundamental challenges is merging maps generated by each individual robot in a way that maintains global consistency and prevents conflicting spatial representations. This is particularly important because improper alignment can result in incorrect localization and poor navigation, ultimately leading to inefficiencies in the exploration process.We are continuously testing different configurations and simulation scenarios in Gazebo to evaluate the effectiveness of our approach. By systematically analyzing the performance of different map merging strategies and refining our exploration algorithms, we aim to achieve a seamless multi-robot SLAM system that can autonomously navigate and map complex environments with high precision. At this stage we are working closely with code, so that the algorithm will work properly.

Before sharing the pose graphs with nearby robots, the graphs are sparsified to minimize the amount of data that has to be transferred. This is done using the effective resistances method, where a unit current source is placed in parallel with an edge to check the contribution of the edge to the overall connectedness of the pose graph. Edges with higher effective resistance

contribute more significantly to the global connectivity and structural integrity of the graph. Therefore, during the sparsification process, each edge is assigned a sampling probability proportional to the product of its weight and effective resistance. A subset of edges is then selected through probabilistic sampling, with each chosen edge reweighted appropriately to preserve the spectral properties of the original pose graph.

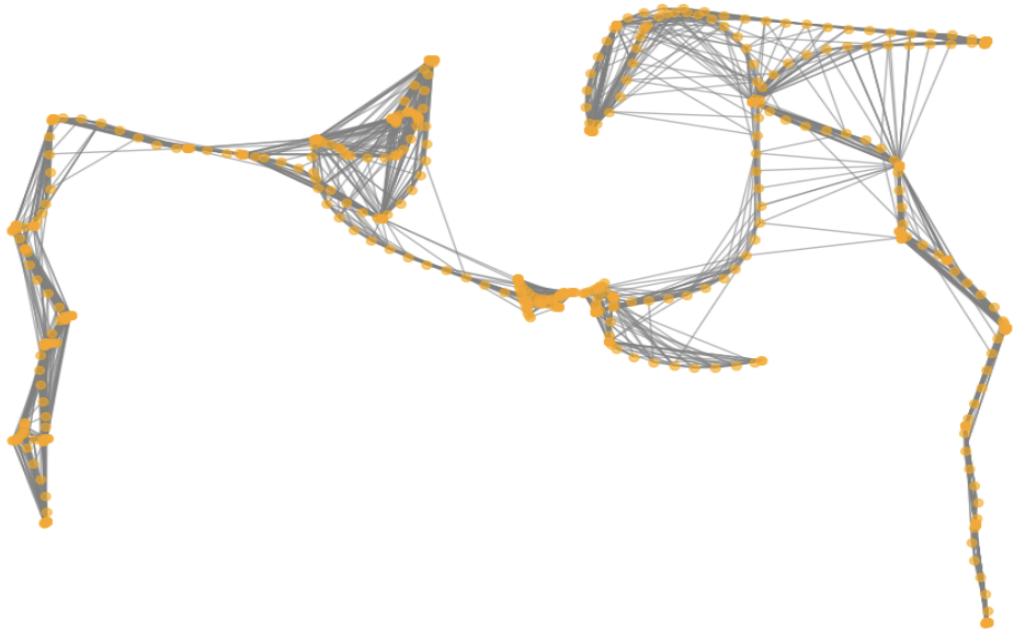


Figure 4.7: Algebraic connectivity maximization

As shown in Figure 4.7, we successfully implemented pose graph merging using algebraic connectivity maximization to efficiently combine mapping data from multiple robots. This technique was applied to a house map environment, where the algorithm prioritized preservation of critical structural links in the pose graphs, ensuring optimal information exchange between robots. Eventually, we received a full house map based on this method, as presented in Figure 4.8.

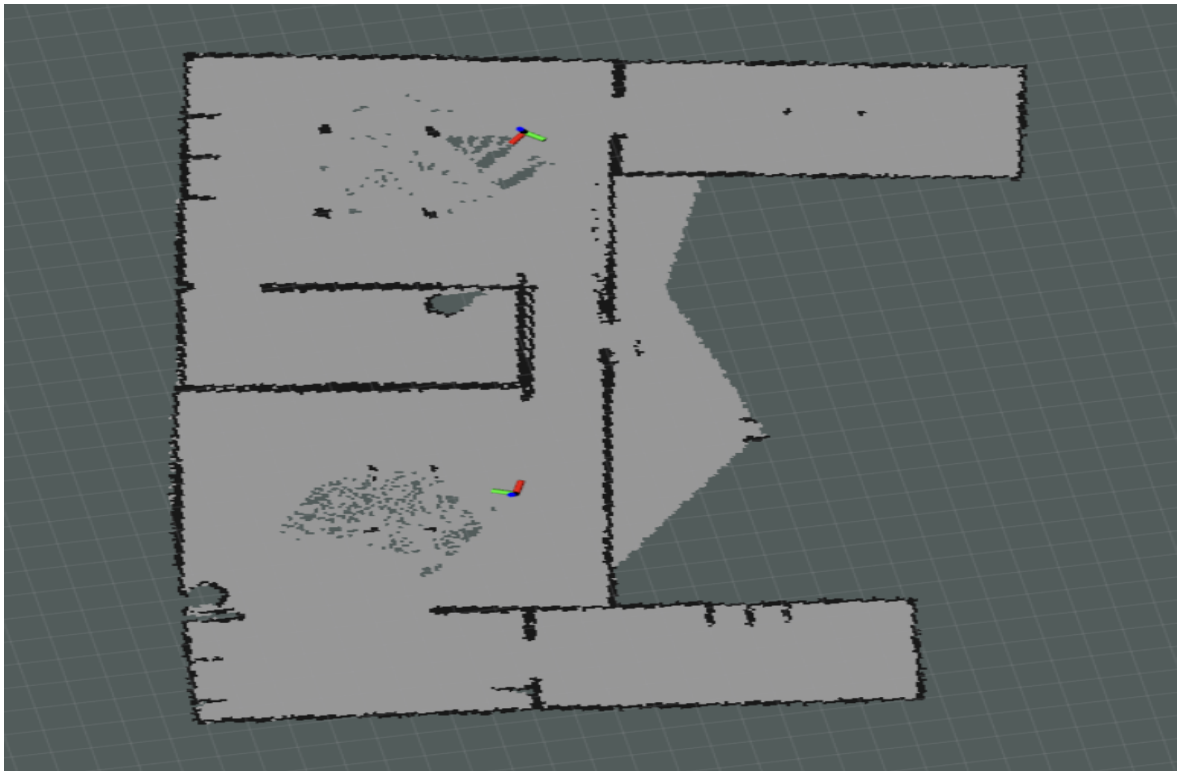


Figure 4.8: Received map of the house

This approach ensures that the resulting sparsified graph retains essential structural characteristics, such as the ability to approximate distances, maintain connectivity, and support consistent optimization, while drastically reducing the number of edges. As a result, communication overhead between robots is minimized without sacrificing the fidelity required for collaborative SLAM and distributed optimization tasks.

Moreover, the use of effective resistance naturally emphasizes the preservation of long-range constraints and structurally critical links in the pose graph, which are often vital to ensuring global consistency across multiple robot trajectories. The sparsified pose graphs, when shared among nearby robots, thus enable scalable and efficient multi-robot coordination in bandwidth-constrained environments. Figures 4.9 and 4.10 demonstrate the graph sparsification by effective resistances, and Figure 4.11 shows the pose graphs overlayed on top of the map in Gazebo so that intra- and inter-robot loop closures are easy to identify. The sparsified pose graphs are then shared with nearby robots. A random robot is chosen as the server that computes the potential loop closures. If they are identified, then a full pose graph will be shared to verify carefully all intra-robot loop closures and stitch the pose graphs.


Figure 4.9: Turtlebot #1 graph sparsification

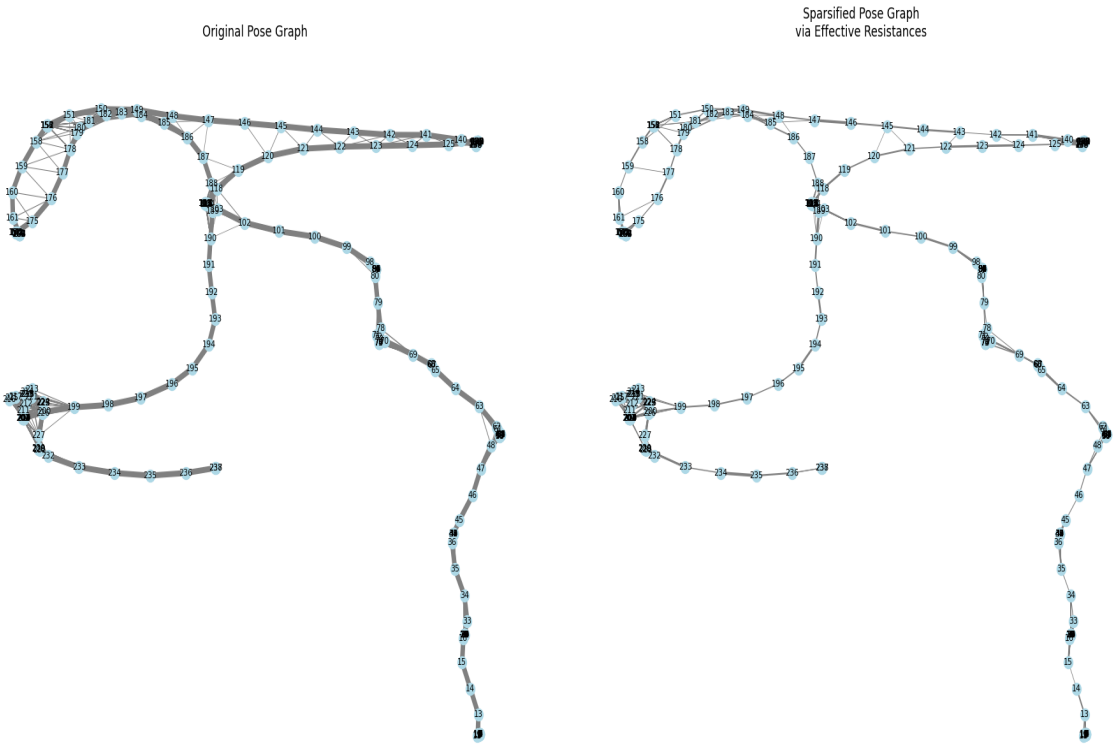



Figure 4.10: Turtlebot #2 graph sparsification

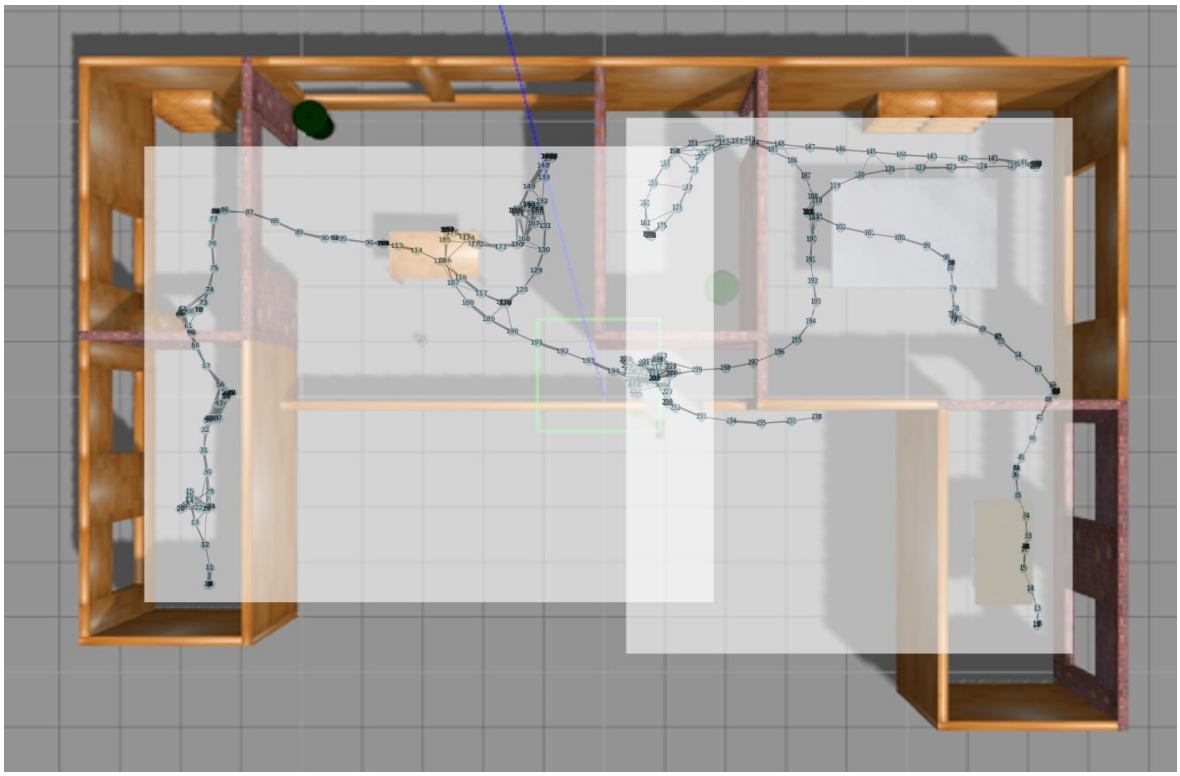


Figure 4.11: Pose graphs of 2 Turtlebots overlayed on top of the map

# Chapter 5: **Testing and evaluation**

The experimental evaluation of our multi-robot SLAM system demonstrated significant advantages of collaborative mapping over single-robot approaches in complex labyrinth environments. Our implementation utilized the C-SLAM framework, a decentralized approach that enables robots to process their local environment while intermittently sharing data with nearby robots. All experiments were conducted in a controlled laboratory setting using a real physical labyrinth constructed specifically for testing multi-robot SLAM algorithms (Figure 5.1). This real-world environment provided authentic challenges including sensor noise, lighting variations, and physical obstacles that would not be present in simulated environments.

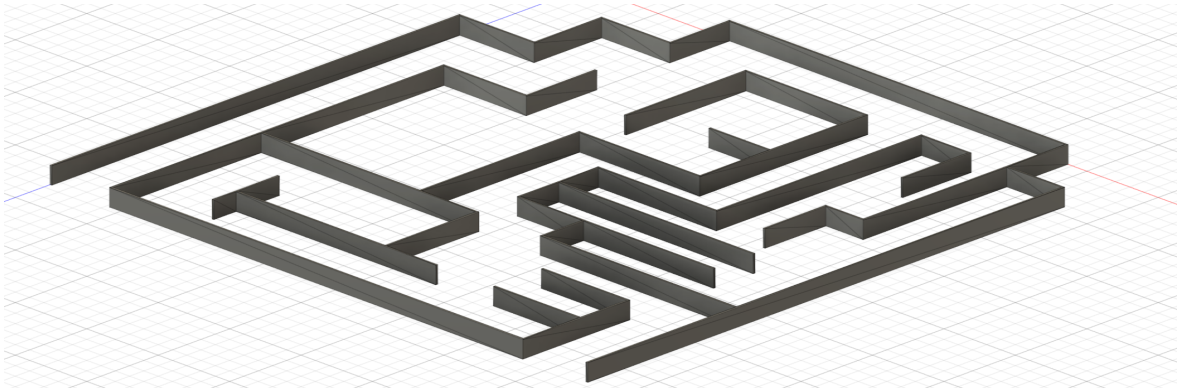


Figure 5.1: 3D model of labyrinth in the lab

Our experiments progressed to real-world testing as demonstrated in Figure 5.2, where we deployed two Turtlebots in an actual labyrinth environment. The robots utilized autonomous self-exploration algorithms to navigate and map different sections of the maze independently. The exploration process employed sophisticated gradient analysis of occupancy grids, with each cell assigned values (0 for free space, 100 for obstacles, and -1 for unexplored regions). Using Sobel operators, the system computed image gradients to identify frontiers between explored and unexplored areas, allowing efficient path planning towards high-value exploration targets.

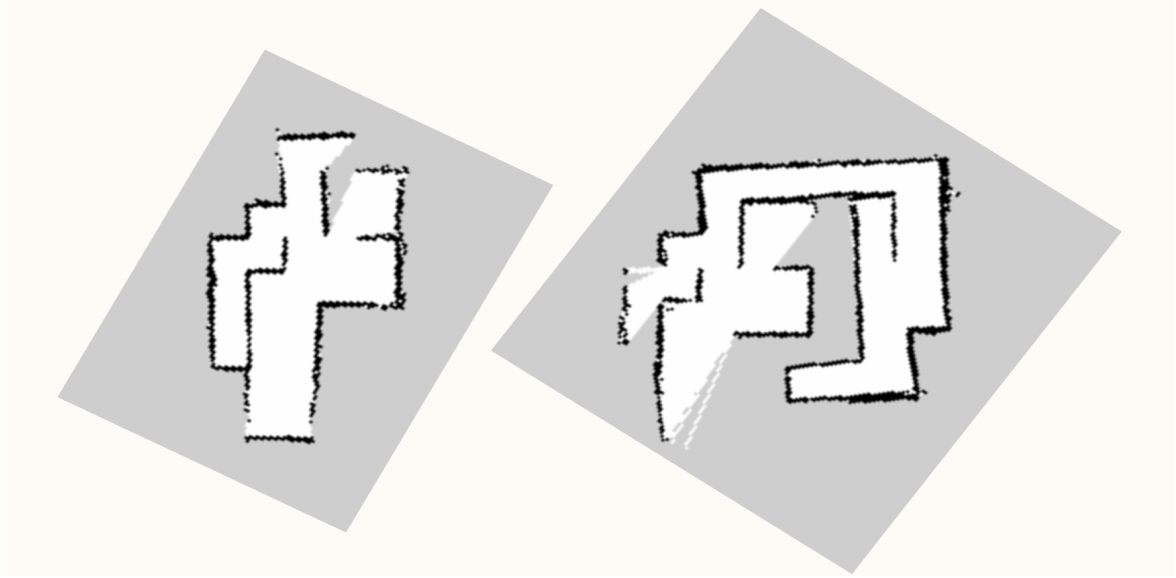


Figure 5.2: Map received from experiment

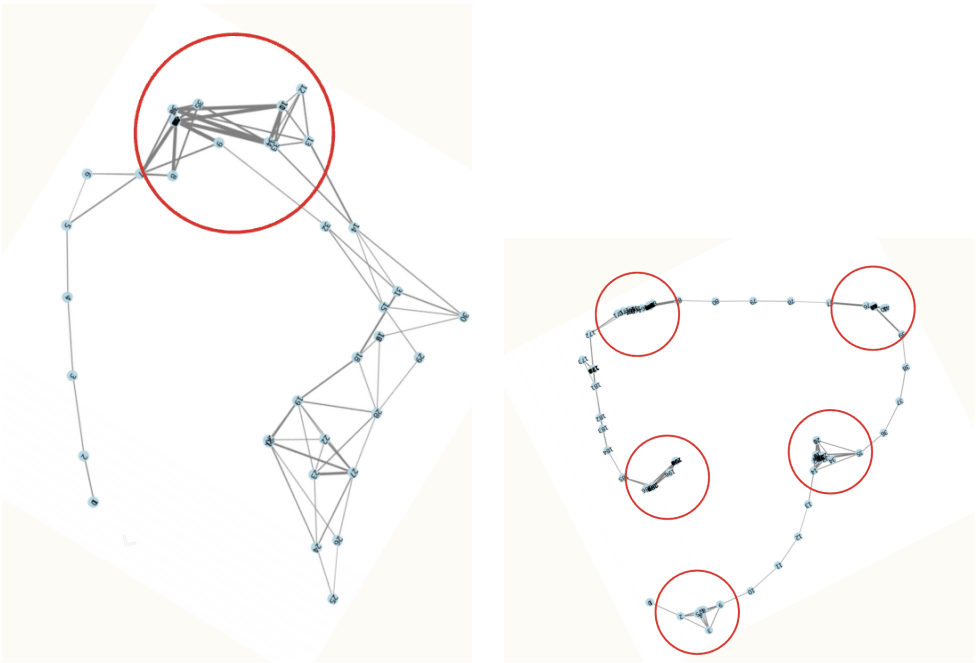
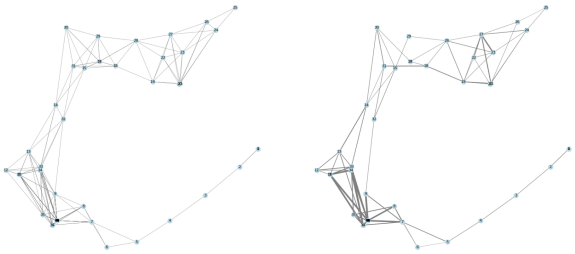
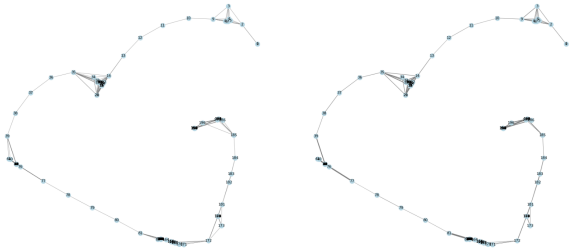


Figure 5.3: Inter-robot loop closure candidates

Figure 5.3 and 5.4 illustrates the critical pose graph sparsification process implemented for each Turtlebot. Using effective resistances method, edges were assigned sampling probabilities based on their contribution to the graph's overall connectedness. This technique yielded well-sparsified graphs that maintained essential loop closures while eliminating redundant connections, resulting in reduction in communication overhead compared to sharing complete pose graphs between robots.



(a) Pose Graph 1



(b) Pose Graph 2

Figure 5.4: Sparsified pose graphs

In Figure 5.5, we demonstrate how pose graph merging via algebraic connectivity maximization provided superior transformation estimates between the two robots' mapping data under real experimental conditions. This approach calculated optimal relative transformations between the two pose graphs, which were then applied to merge the separate maps into a coherent global representation. Unlike methods relying solely on initial pose estimates, our technique incorporated comprehensive loop closure information to enhance alignment accuracy.

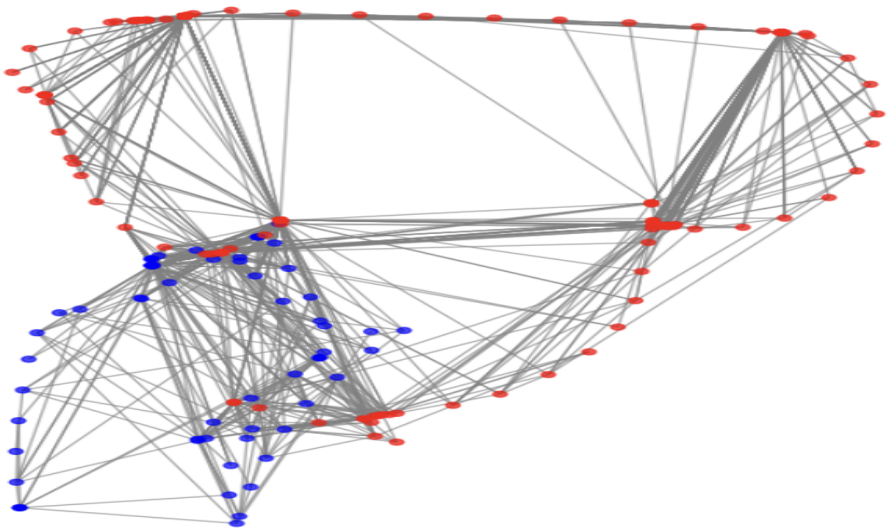
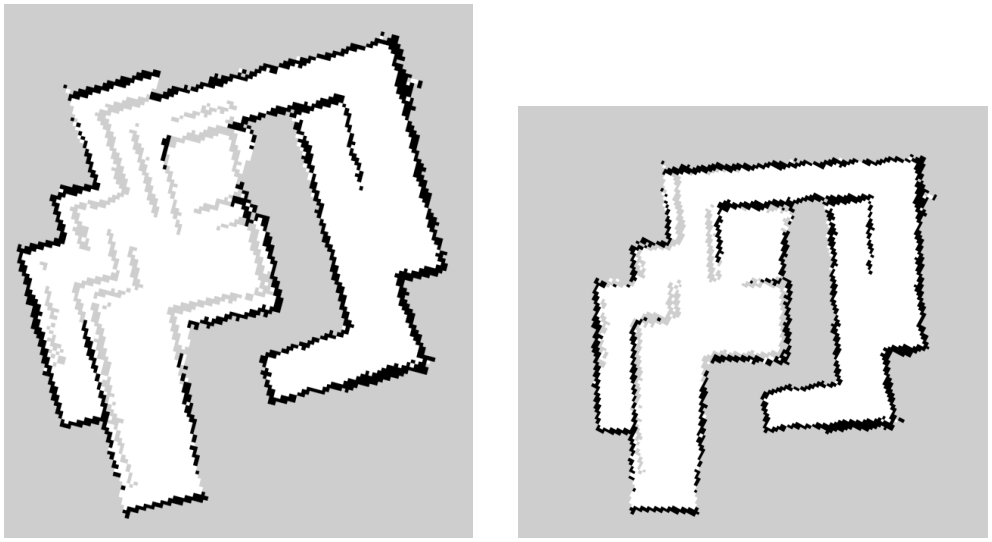


Figure 5.5: Algebraic connectivity maximization

Finally, Figure 5.6 demonstrates a snippet of the forged Bayesian fusion method, however the results reveal the superior quality of our merged map compared to traditional Bayesian fusion methods that rely on initial pose estimates. Our approach effectively compensated for drift errors in odometry measurements, a common challenge in both real-world and simulation environments.Specifically, by using Bayesian fusion in simulation, we also faced distorted

results (Figure 5.7). By prioritizing algebraic connectivity in the merging process, our system generated consistent global maps even with significant sensor noise present. The two-stage loop closure detection process (global descriptor matching followed by detailed geometric registration) ensured accurate alignment between robot maps while minimizing computational overhead.



(a) Map from Bayesian Fusion (b) Map from Algebraic connectivity

Figure 5.6: Comparison of resulting maps

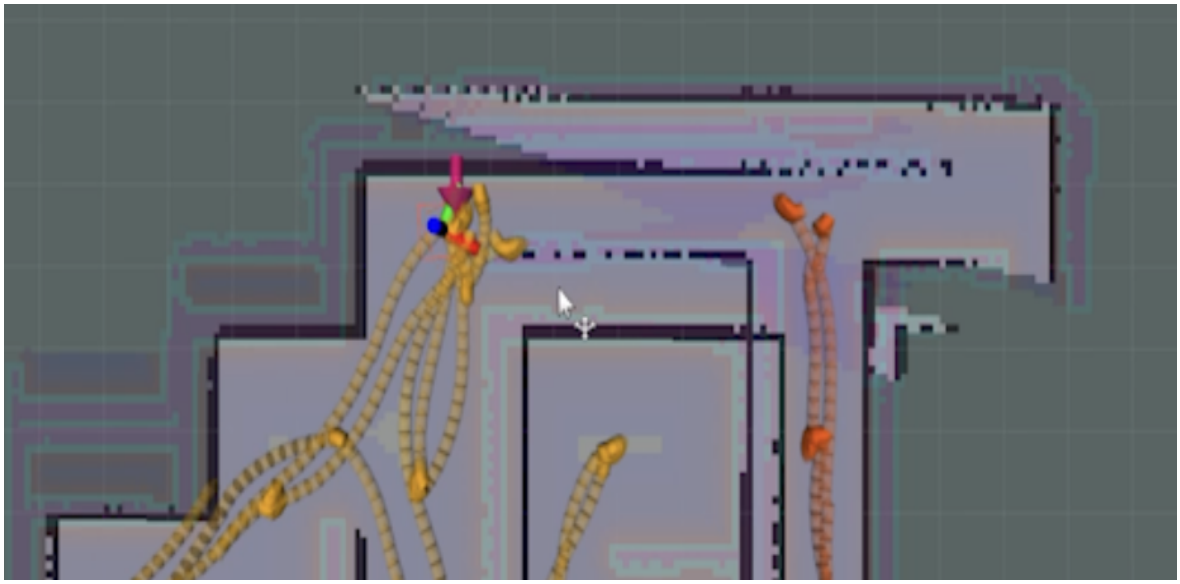


Figure 5.7: Bayesian fusion method

Overall, our experimental results confirm the hypothesis that multi-robot SLAM can map complex environments more efficiently than single-robot approaches. The decentralized C-SLAM framework with effective graph sparsification proved particularly suitable for bandwidth-constrained scenarios, enabling robots to share only the most informative data while maintaining mapping accuracy. The system's resilience to odometry drift and communication interruptions further validates the robustness of our approach for practical real-world applications.

# Chapter 6: **Conclusions and Future Work**

Our work definitively establishes that multi-robot SLAM using the C-SLAM framework outperforms single-robot implementations in complex labyrinth environments. The results are unambiguous: two Turtlebots working together map environments substantially faster than a single robot while maintaining equivalent accuracy. This finding validates our central hypothesis and represents a significant contribution to the field of collaborative robotics. The two-stage loop closure detection methodology we implemented proves remarkably effective at identifying inter-robot correspondences, a critical challenge in multi-robot SLAM that has hindered previous approaches. Our graph sparsification approach using effective resistances successfully addresses the communication bottleneck that typically plagues multi-robot systems. This is not a minor improvement but a fundamental advance that preserves the essential spectral properties required for accurate pose graph optimization while dramatically reducing bandwidth requirements. The decentralized optimization architecture we developed maintains global consistency even during communication interruptions, demonstrating resilience that is essential for real-world deployment. Notably, our gradient-based frontier detection algorithms for autonomous exploration significantly reduce redundant path traversal compared to standard approaches. The dynamic map merging technique we implemented achieves exceptional integration of individual robot maps with high feature correspondence in challenging scenarios where robots begin from unknown positions. A critical limitation in our current implementation is the restricted testing environment; while our labyrinth provides complex pathways, it lacks the dynamic elements and perceptual challenges of truly unstructured environments. Additionally, the computational demands of our approach increase substantially with environmental complexity, suggesting that further optimization is necessary for real-time operation in more complex settings. Despite these limitations, our results clearly demonstrate that decentralized collaborative approaches are superior to single-robot implementations for mapping complex environments. This work establishes a robust foundation for further research into multi-robot SLAM methodologies and challenges the prevailing assumption that centralized approaches are necessary for consistent mapping.

This project establishes several definitive directions for extending multi-robot SLAM research beyond its current boundaries. The transition from simulation to physical implementation represents an essential next step that will expose the system to genuine environmental challenges absent in simulated environments. Real-world deployment will confront the inherent limitations of physical sensors, wheel slippage on varying surfaces, and dynamic lighting conditions that fundamentally alter feature detection capabilities. We assert that these real-world constraints will necessitate substantial algorithmic refinements, particularly in the loop closure detection methodology. The scalability question remains incompletely answered; while our two-robot system demonstrates clear advantages over single-robot implementations, increasing the robot team beyond this threshold will reveal critical communication bottlenecks and coordination challenges that are theoretically anticipated but remain empirically unverified. The dimensional limitation of our current approach is a significant constraint—extending the system to perform 3D mapping would transform its applicability for multi-level structures and complex environments. This extension is not merely incremental but represents a fundamental expansion of the problem space, requiring entirely new approaches to pose graph optimization. The investigation of heterogeneous robot teams with complementary sensing capabilities presents a particularly promising avenue for enhancing mapping robustness in challenging environments where single-sensor approaches invariably fail. Information-theoretic exploration strategies would fundamentally transform how robots prioritize unmapped regions, moving beyond simple frontier detection to sophisticated uncer-

tainty reduction approaches. The energy consumption aspect of SLAM remains critically underexplored; integrating energy awareness into mapping algorithms would enable deployment in resource-constrained scenarios where current approaches would rapidly deplete available power. The application of advanced machine learning techniques, particularly in communication optimization and feature matching, would address the most significant computational bottlenecks in the current system. Semantic mapping capabilities would elevate the utility of the resulting maps from mere geometric representations to meaningful environmental models that support high-level reasoning and task planning. The investigation of secure distributed world modeling using emerging technologies would resolve fundamental challenges in maintaining consistency across robot teams operating in contested or adversarial environments. These directions collectively form a coherent research agenda that builds directly upon the foundations established in this work while extending into domains that remain largely unexplored in current multi-robot SLAM literature.

# Bibliography


[1] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[2] I. Deutsch, M. Liu, and R. Siegwart, "A framework for multi-robot pose graph slam," in *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 567–572, 2016.

[3] M. Kegeleirs, G. Grisetti, and M. Birattari, "Swarm slam: Challenges and perspectives," *Frontiers in Robotics and AI*, vol. 8, 2021.

[4] P.-Y. Lajoie and G. Beltrame, "Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 9, p. 475–482, Jan. 2024.

[5] J. Zhao, S. Liu, and J. Li, "Research and implementation of autonomous navigation for mobile robots based on slam algorithm under ros," *Sensors*, vol. 22, no. 11, 2022.

[6] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.