Politecnico di Torino

Politecnico di Torino

Computer Engineering - Artificial Intelligence and Data Analytics

A.a. 2024/2025

Graduation Session October 2025

Multi-robot Collaborative Simultaneous Localization and Mapping

Relatori:

Marcello Chiaberge

Giorgio Audrito

Mauro Martini

Stefano Primatesta

Candidato:

Simone Borella

Abstract

Advancements in multi-robot systems represent a significant frontier in robotics, with growing applications in domains such as agriculture, search and rescue, industry, domestic environments, and transportation. A key challenge in these settings is enabling multiple robots to operate collaboratively, safely, and efficiently while navigating complex, dynamic environments. This thesis focuses on the development of a multi-robot collaborative SLAM (Simultaneous Localization and Mapping) system designed to provide robust and accurate localization while enabling the coordinated and efficient exploration and mapping of unknown environments. By employing a centralized framework, the system integrates sensor data and robot dynamics to maintain global consistency across the multi-robot network. Factor graph formulations are employed as a powerful tool to jointly solve SLAM, decision-making, and trajectory planning for the multi-robot system, implemented using the GTSAM library. Within the SLAM component, the framework optimizes the entire system's state by integrating motion constraints and sensor measurements while simultaneously building the map and exploring frontiers to discover unknown areas. For decision-making, discrete factor graphs are used to assign frontiers to individual robots, ensuring an efficient allocation of exploration tasks. In local trajectory planning, the factor graph solution naturally extends to jointly optimize robot paths, promoting efficient, safe navigation while enforcing collision avoidance and coordinated behaviors across the team. Experimental evaluations are conducted in simulation and indoor environments using a team of TurtleBot3 robots, with a Vicon motion capture system providing ground truth references. These experiments assess mapping accuracy, localization reliability, and efficiency in multi-robot exploration. The results show the effectiveness of factor graph-based frameworks enhancing multi-robot collaboration in complex, unstructured environments, while also pointing out some limitations and directions for further improvement.

Acknowledgements

This thesis marks the end of an incredible journey at Politecnico di Torino, a journey filled with challenges, discoveries, and unforgettable experiences that have shaped both my academic path and personal growth. Along the way, I've been lucky to have the support of many wonderful people, and this section is dedicated to all of you with my deepest gratitude.

To all the people I've met in Turin - thank you for the moments we shared, and the friendships that made this city feel like home. Each of you, in your own way, has left a mark on this journey.

To my housemates - thank you for sharing daily life with me, for all the Clash Royale and Burraco sessions, and for the daily support. You turned our home into a place that I can really call home.

To the friend of "Scugna" - thank you for being there through every high and low, always ready to share laughter, encouragement, and, of course, something to drink together. You made these years brighter and unforgettable.

To Giorgio - one of the most important people in my life. Your friendship has been a constant source of fun, strength, and inspiration. Thank you for being there, always.

To Stefania - thank you for the wonderful moments we've shared, for always cheering for me, and for your constant support.

To my whole family - who have continuously pushed me to grow, to do better, and to become a better person every day. Your support has been the foundation of everything I've achieved up until now.

To Luca - my brother and great adventure companion. Thank you for walking this path with me. I can't wait to dive into all the amazing adventures that still await us.

To my mom and dad - my life teachers, my life supporters, my life. Thank you for your love, for your wisdom, for your patience, and for celebrating every small victory with me. None of this would have been possible without you.

I am deeply grateful to all of you and for making this journey truly unforgettable.

Oh and thank you, dear reader, hope you find my thesis interesting!

And last but not least...

Table of Contents

List of Tables

List of Figures

Glossary

PIC4SeR

PoliTo Interdipartimental Center for Service Robotics

SLAM

Simultaneous Localization And Mapping

A-SLAM

Active Simultaneous Localization And Mapping

AC-SLAM

Active Collaborative Simultaneous Localization And Mapping

V-SLAM

Visual Simultaneous Localization And Mapping

VI-SLAM

Visual Inertial Simultaneous Localization And Mapping

GTSAM

Georgia Tech Smoothing and Mapping Library

MAP

Maximum a Posteriori

iSAM2

Incremental Smoothing and Mapping 2

GPS

Global Positioning System

IMU

Inertial Measurement Unit

LiDAR

Laser Imaging Detection and Ranging

EKF

Extended Kalman Filter

PGO

Pose Graph Optimization

SfM

Structure from Motion

WFD

Wavefront Frontier Detection

FFD

Fast Frontier Detection

OBB-FD

Oriented Bounding Box Frontier Detection

NaiveAA

Naive Active Area

EWFD

Expanded Wavefront Frontier Detection

FTFD

Frontier Tracing Frontier Detection

GBP

Gaussian Belief Propagation

ORB

Oriented Fast and Rotated Brief

SE

Special Euclidean

JCBB

Joint Compatibility Branch and Bound

PDA

Probabilistic Data Association

DBSCAN

Density-Based Spatial Clustering of Applications with Noise

BFS

Breadth First Search

DWA

Dynamic Window Approach

FGP

Factor Graph-based Planner

ROS2

Robot Operating System 2

LAN

Local Area Network

RTAB-Map

Real Time Appearance Based Mapping

Chapter 1

Introduction

As automation continues to advance, multiple robotic systems will increasingly need to operate side by side — whether ground rovers in farms, service robots in homes, industrial machines in factories, or autonomous vehicles on roads. Ensuring both safety and efficiency in such settings, especially when operating at higher speeds or within limited spaces, requires careful coordination of motion and task planning. Robots first need to ensure collision avoidance by being aware of each other's trajectories. Beyond this basic requirement, tighter coordination enables significant gains in efficiency, where large teams can move in harmony and collectively accomplish complex tasks.

At the most basic level, multi-robot SLAM enables each robot to estimate its trajectory and build a local map of the environment. When combined with centralized coordination, these local maps and individual pose estimates can be fused into a consistent global representation, reducing uncertainty in both localization and mapping while enabling the team to plan efficient, non-conflicting paths. Centralized approaches can use a single computational system to manage the positions, localization estimates, decision-making goals, and trajectories of all agents together. By optimizing these elements at the system-wide level, the approach can coordinate localization and motion planning, prevent collisions, and improve exploration efficiency.

Multi-robot SLAM and path planning pose two key challenges in complex environments: (i) maintaining a consistent and accurate global map as multiple rovers navigate and sense the environment, and (ii) generating coordinated trajectories that maximize coverage while avoiding collisions and respecting dynamic constraints. Factor graph formulations are particularly well-suited for this task, as they can naturally encode robot dynamics, sensor measurements, and obstacle avoidance constraints within a unified probabilistic framework. Centralized inference over the factor graph ensures that global consistency is maintained, and optimal trajectories can be computed for the entire team.

In agricultural settings, rovers must efficiently cover large fields while navigating around obstacles such as crops, vineyards, orchards, trees, uneven terrain, and farm equipment. In search and rescue, robots navigate cluttered, unpredictable environments with limited visibility, where efficient exploration and mapping are critical for mission success. Centralized, factor-graph-based SLAM and path planning allow teams of ground rovers to operate safely and effectively in both scenarios, providing accurate global maps and smooth, coordinated trajectories.

In summary, the thesis contributions are:

- A centralized factor-graph-based multi-robot SLAM framework enabling consistent global localization of the robot system and global mapping in previously unknown environments.
- Integration of centralized trajectory optimization and path planning into the SLAM framework, ensuring collision-free, efficient, and smooth exploration.
- A broad evaluation in agricultural and search and rescue scenarios, highlighting the key advantages of the proposed approach and outlining potential directions for future improvements.

1.1 Goal

The main goal of this thesis is to design, implement, and evaluate a centralized multi-robot framework that integrates SLAM with trajectory optimization and path planning. The framework leverages a factor-graph formulation to ensure global consistency in both localization and mapping, while simultaneously enabling efficient and safe coordination of multiple ground rovers in complex and dynamic environments.

The proposed approach is analyzed in scenarios relevant to agriculture and search and rescue, where efficient exploration and mapping are critical.

Furthermore, the work investigates the potential of factor graphs as a unified tool for addressing the coupled challenges of localization, mapping, and planning in multi-robot systems.

1.2 Thesis structure

The thesis is structured as follows:

- **Chapter 2: Background** introduces the theoretical foundations of SLAM, factor graphs, and optimization techniques that form the basis of the proposed framework.

- **Chapter 3: State of the Art** reviews the existing literature on SLAM, and other algorithms employed, applied to multi-robot systems.
- **Chapter 4: Methodology** presents the design and development of the centralized multi-robot SLAM and path planning system, describing the architecture, algorithms, and integration details.
- **Chapter 5: Experiments and Results** outlines the experimental setup, evaluation metrics, and test scenarios in both simulation and real-world experiments. Reports and analyzes the outcomes of the experiments, evaluating the performance of the proposed framework in terms of localization accuracy, mapping quality, trajectory efficiency, and robustness.
- **Chapter 6: Conclusions and Future Works** summarizes the main findings of the thesis, discusses the advantages and limitations of the current approach, and suggests directions for future research and improvements.

Chapter 2

Background

2.1 Factor Graphs and Probabilistic Inference

A factor graph is a bipartite probabilistic graphical model that represents how a joint probability distribution can be decomposed into a product of local factors. This structure provides an intuitive and efficient way to reason about high-dimensional probabilistic models, making it especially useful in robotics, signal processing, and computer vision. Factor graphs form the foundation of many inference algorithms by explicitly encoding the relationships, defined as factors, between variables.

2.1.1 Definition of Factor Graphs

Formally, let $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of variables. Suppose we want to represent a global function $F(\mathbf{X})$ that factorizes into a product of local functions:

$$F(\mathbf{X}) = \prod_i f_i(\mathbf{X}_i), \tag{2.1}$$

where each factor f_i depends only on a subset of the variables $\mathbf{X}_i \subseteq \mathbf{X}$.

A factor graph is a bipartite graph $G = (V, F, E)$ consisting of:

- **Variable nodes** V: each corresponding to a variable $x_k \in \mathbf{X}$,
- **Factor nodes** F: each corresponding to a local function f_i,
- **Edges** E: connecting a factor node f_i to all variables $x_k \in \mathbf{X}_i$ on which it depends.

This bipartite structure makes explicit which subsets of variables interact with each other through local factors, enabling modular and sparse representations of large systems.

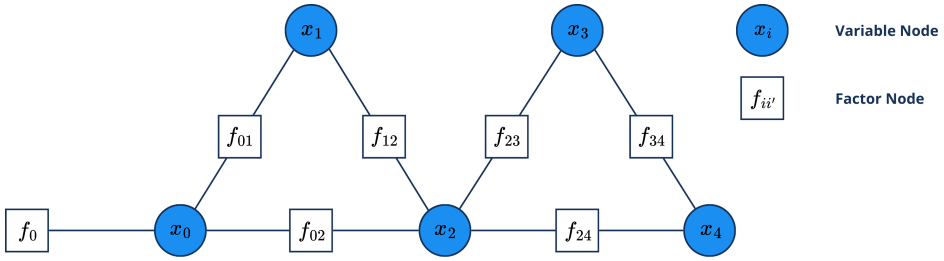


Figure 2.1: Factor graph variable and factor representation

2.1.2 Probabilistic Inference

In probabilistic models, the global function $F(\mathbf{X})$ often represents a joint probability distribution. For example, if $\mathbf{X}$ denotes random variables, we may write:

$$p(\mathbf{X}) \propto \prod_i f_i(\mathbf{X}_i), \tag{2.2}$$

where each factor f_i is typically associated with a probability distribution, likelihood term, or prior. This factorization expresses the conditional independencies in the model and is the starting point for efficient inference.

2.1.3 Maximum a Posteriori Inference

A central task in probabilistic inference is to compute the most probable value of the unknown variables given measurements and prior knowledge. This task, known as Maximum a Posteriori (MAP) inference, can be formulated as:

$$\mathbf{X}_{\text{MAP}} = \arg\max_{\mathbf{X}} \prod_i f_i(\mathbf{X}_i), \tag{2.3}$$

where $f_i(\mathbf{X}_i)$ are the factors encoding measurements or prior information.

In robotics applications such as SLAM, factors are often modeled using measurement functions $h_i(\mathbf{X}_i)$ with associated observations z_i corrupted by zero-mean Gaussian noise with covariance Σ_i. In this case, each factor takes the form:

$$f_i(\mathbf{X}_i) \propto \exp\left(-\frac{1}{2}\|h_i(\mathbf{X}_i) - z_i\|^2_{\Sigma_i}\right). \tag{2.4}$$

where $\|\mathbf{e}_i\|^2_{\Sigma_i} = \mathbf{e}_i^\top \Sigma_i^{-1} \mathbf{e}_i$ is the Mahalanobis distance and $\mathbf{e}_i = h_i(\mathbf{X}_i) - z_i$ is the residual error of the measurement.

Taking the negative logarithm of the MAP objective yields the equivalent nonlinear least squares problem:

$$\mathbf{X}_{\text{MAP}} = \arg\min_{\mathbf{X}} \sum_i \|h_i(\mathbf{X}_i) - z_i\|^2_{\Sigma_i}. \tag{2.5}$$

Thus, MAP inference reduces to minimizing the sum of squared residual errors between predicted and observed measurements, weighted by their uncertainties. This process naturally performs sensor fusion by combining multiple measurement likelihoods and priors to produce a consistent estimate of the unknowns. In practice, algorithms such as the elimination algorithm or its nonlinear extensions form the computational core of MAP inference in robotics.

2.1.4 Nonlinear Optimization

Because the measurement functions $h_i(\cdot)$ are nonlinear, the optimization problem cannot be solved in closed form. Instead, iterative algorithms such as Gauss–Newton or Levenberg–Marquardt are applied. These methods rely on successive linearizations of the measurement functions around a current estimate $\mathbf{X}_0$:

$$h_i(\mathbf{X}_i) \approx h_i(\mathbf{X}_i^0) + H_i\,\delta_i, \tag{2.6}$$

where H_i is the measurement Jacobian evaluated at $\mathbf{X}_i^0$, and δ_i is a perturbation update.

By stacking all Jacobians into a global sparse matrix A and residuals into a vector b, the update step is expressed as a standard least squares problem:

$$\delta^* = \arg\min_{\delta} \|A\delta - b\|^2. \tag{2.7}$$

The sparsity and block structure of A mirror the underlying factor graph, and efficient solvers exploit this structure through sparse elimination algorithms. The updated estimate $\mathbf{X} \leftarrow \mathbf{X}_0 + \delta^*$ is then re-linearized, and the process repeats until convergence.

2.1.5 Incremental Inference

While batch optimization solves the full problem at once, robotics applications such as SLAM require continuous updates as new sensor data arrive. Recomputing the entire solution from scratch is computationally prohibitive, motivating incremental inference techniques.

A key innovation in this context is the use of the Bayes tree, a data structure that compactly represents the factorization of the posterior distribution after variable elimination. The Bayes tree organizes the problem into a hierarchy of conditional probability densities, where each clique corresponds to a subset of variables conditioned on their ancestors in the tree. This representation not only exposes the conditional independence structure of the problem but also enables localized updates: when new factors are added, only the affected part of the

tree needs to be relinearized and updated, while the rest of the solution remains unchanged.

The iSAM2 algorithm, explained in [1] and [2], leverages Bayes trees to perform efficient incremental inference. It supports:

- **Selective relinearization:** only variables whose linearization points are significantly outdated are re-expanded.
- **Incremental variable reordering:** the variable elimination order can be updated locally to maintain sparsity and efficiency.
- **Localized updates:** thanks to the tree structure, updates propagate only through the relevant branches, avoiding redundant computations.

By exploiting Bayes trees, iSAM2 achieves real-time performance while maintaining the accuracy of batch optimization. The use of Bayes trees allow also an improvement in multi-robot SLAM application as outlined in [3]. This makes it particularly suitable for large-scale, online SLAM and other robotic estimation problems where the factor graph grows continually with new measurements.

2.2 Simultaneous Localization and Mapping

In robotics, the problem of enabling a robot to navigate an initially unknown environment requires solving two tasks simultaneously: constructing a map of the surrounding environment while concurrently estimating its own pose with respect to the map. This coupled problem, known as Simultaneous Localization and Mapping (SLAM), is challenging because accurate localization relies on a reliable map, while map construction itself depends on precise localization.

The importance of SLAM in robotics research comes from the fact that robot poses are rarely available in practice. External systems such as differential GPS or motion capture can provide accurate localization, but they are costly and limited to constrained environments, making them unsuitable for large-scale or real-world deployments. To address this, SLAM algorithms aim to estimate both the environment and the robot trajectory directly from sensor measurements, combining information from proprioceptive sensors, such as wheel odometry and IMUs, with exteroceptive sensors, such as cameras, LiDAR, or radar.

Formally, SLAM can be viewed as an inverse problem: given a sequence of measurements, the task is to determine the map and the corresponding robot poses that best explain them, enabling autonomous navigation without relying on external localization infrastructure.

2.2.1 Problem Formulation

Formally, the full SLAM problem can be expressed as the estimation of the joint posterior:

$$p(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathcal{Z}_{1:T}, \mathcal{U}_{1:T}), \tag{2.8}$$

where:

- $\mathbf{x}_{1:T}$ are the robot poses over time,
- $\mathbf{m}$ represents the map,
- $\mathcal{U}_{1:T}$ are control inputs (odometry measurements),
- $\mathcal{Z}_{1:T}$ are sensor measurements.

This posterior expresses the probability of the robot poses and map given all sensor and odometry measurements. Solving this problem can be approached in two main ways. From a probabilistic perspective, Bayesian filtering techniques can be used, such as the Extended Kalman Filter (EKF) or particle filters (FastSLAM). Alternatively, the problem can be formulated as a batch optimization task, where the goal is to find the Maximum a Posteriori (MAP) estimate of the poses and map that best satisfies all measurement constraints. Graph-based SLAM and factor-graph formulations fall into this category, exploiting the sparse structure of the problem to efficiently compute optimal solutions for large-scale environments.

2.2.2 Graph-Based SLAM

Graph-based SLAM reformulates the problem as a nonlinear optimization over a factor graph structure (2.1), where nodes represent robot poses and landmarks, and edges encode motion or measurement constraints. The goal is to find the configuration of poses and landmarks that best satisfies all constraints, typically using nonlinear least-squares optimization. This approach scales well and has become the dominant paradigm in modern SLAM systems.

Factor graphs nodes, in the SLAM context, are defined as:

- **Variable nodes** representing the unknown quantities to estimate, which are robot poses $\mathbf{x_t}$ at time $\mathbf{t}$ and landmark positions $\mathbf{l_i}$.
- **Factor nodes** encoding probabilistic constraints between variables. Typical factors in SLAM include:
  - **Odometry factors** which relate consecutive robot poses through control inputs or odometry measurements.

– **Measurement factors** which relate robot poses to observed landmarks or environmental features measured through sensors.

– **Prior factors** which encode known information about initial poses or fixed landmarks.

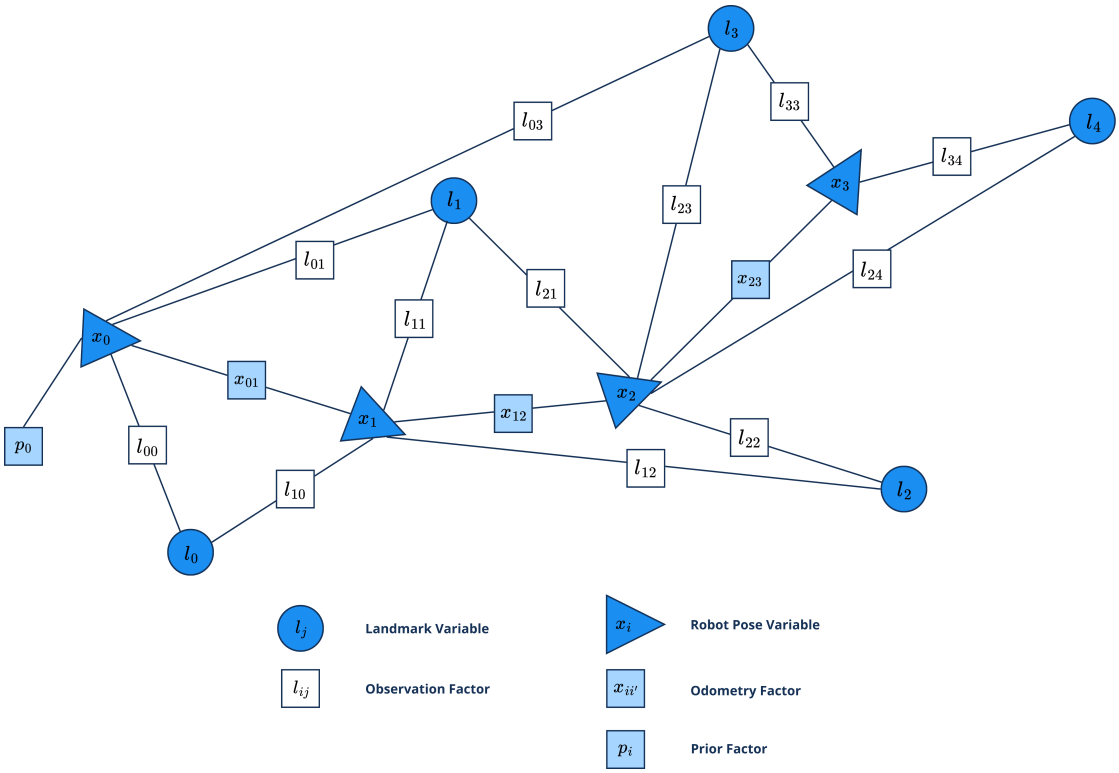


Figure 2.2: Graph-Based SLAM factor graph problem formulation

Using this formulation, the joint posterior over all variables can be factorized as:

$$p(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathcal{Z}_{1:T}, \mathcal{U}_{1:T}) \propto \prod_k f_k(\mathbf{x}_k, \mathbf{l}_k), \tag{2.9}$$

where each factor f_k represents the likelihood of a measurement or the constraint imposed by the motion model.

The SLAM estimation problem then reduces to finding the set of variable values that maximizes the posterior, equivalently minimizing the sum of squared residuals associated with each factor:

$$\mathbf{x}^*, \mathbf{l}^* = \arg\min_{\mathbf{x},\mathbf{l}} \sum_k \|r_k(\mathbf{x}_k, \mathbf{l}_k)\|^2_{\Sigma_k^{-1}}, \tag{2.10}$$

where r_k are the residuals corresponding to each factor, and Σ_k are the associated covariance matrices representing measurement uncertainty.

Factor graphs exploit the sparsity inherent in SLAM, since each measurement typically connects only a small subset of variables. This enables efficient optimization using techniques such as Gauss-Newton, Levenberg-Marquardt, or incremental solvers like iSAM2, supporting real-time performance even in large-scale environments.

By separating variables and factors explicitly, factor graphs also provide a natural foundation for extending SLAM to centralized or distributed multi-robot systems.

2.2.3 Active SLAM

Active SLAM (A-SLAM) extends the traditional SLAM problem by not only estimating the robot trajectory and the map, but also enabling the robot to actively and autonomously explore unknown environments. The system plans its own movements to gather the most informative measurements, planning actions that actively reduce uncertainty, and build a complete map, all while ensuring safe navigation.

From an architectural perspective, Active SLAM can be described using the following structure:

- **Front-End:** The front-end is responsible for processing raw sensor data to generate observations and constraints. This typically includes:
  - Feature extraction to extract meaningful observations from raw sensor data.
  - Data association, matching observed features to previously seen ones, stored as landmarks.
  - Odometry estimation or motion prediction.
  - Detection of loop closures when revisiting known areas.

  The front-end essentially produces the measurements that will be incorporated in the SLAM solution employed in the back-end.

- **Back-End:** The back-end performs state estimation and optimization. Given the constraints produced by the front-end, the back-end updates the robot trajectory and map using techniques such as:
  - Nonlinear least-squares optimization over a factor graph.

  - Probabilistic filtering methods (EKF, particle filter) for incremental updates.

  The back-end maintains a consistent global estimate and propagates uncertainty, which is essential for informed decision-making in active exploration.

- **Decision-making and Planning Layers:** In Active SLAM, decision-making and planning modules use the current map and uncertainty estimates from the back-end to choose control actions that maximize information gain. Common strategies include:

  - Selecting paths that observe unexplored regions or reduce map uncertainty.
  - Balancing exploration and exploitation to efficiently cover unknown areas.
  - Considering risk and collision constraints to ensure safe navigation.

  These layers close the loop by generating control inputs that feed back into the front-end and back-end, forming a continuous perception-action cycle.

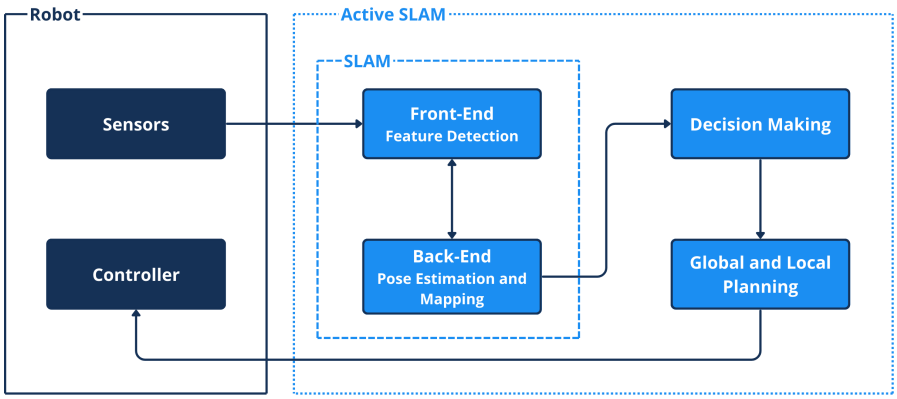


Figure 2.3: Active SLAM perception-action cycle

2.2.4 Active Collaborative SLAM

Collaborative SLAM (C-SLAM) is a method where multiple robots work together by sharing information, allowing them to improve both their own position estimates and the quality of the map they build. This cooperation is especially valuable for tasks that require large-scale exploration. Despite its benefits, C-SLAM also introduces new difficulties. Robots usually have limited computing capabilities,

and they can only exchange a restricted amount of data due to communication bandwidth, while handling possible network interruptions or failures.

Active Collaborative SLAM (AC-SLAM) builds on the idea of Active SLAM but adds specific requirements for how robots communicate and share parameters with each other. The shared information can include different types of data such as uncertainty, or entropy, measures, differences between probability distributions, localization details, visual features captured from sensors, and frontier points that mark unexplored regions.

2.2.5 Mapping

In the context of SLAM, mapping is the process of constructing a representation of the environment while simultaneously estimating the robot's trajectory. Different mapping strategies are designed to balance trade-offs between memory usage, computational efficiency, scalability, and the level of detail captured in the map.

For small and structured environments, fine-grained approaches such as occupancy grids or direct dense mapping are effective, since the computational and memory demands remain manageable while producing highly detailed reconstructions. In contrast, large-scale or unstructured environments benefit from more lightweight and scalable methods, such as topological or feature-based mapping, which reduce memory requirements and allow long-term operation over extended trajectories. Hybrid techniques combine the advantages of both, employing detailed local maps for accuracy while keeping global relationships between local maps for efficiency and scalability.

Common approaches include:

- **Feature-based Mapping** Represents the environment as a sparse set of landmarks used for localization. Highly efficient in terms of memory and computation.

- **Occupancy Grid Mapping** Divides the environment into grids of fixed resolution, each storing an occupancy probability. Memory-intensive, but widely used with 2D LiDAR due to its simplicity and robustness.

- **Topological Mapping** Models the environment as a graph of nodes, representing places and edges, representing connections. Lightweight and scalable, making it suitable for very large spaces.

- **Hybrid Mapping** Combines occupancy grids with topological representations, maintaining a local occupancy grid for each topological place.

- **Direct Mapping** Builds maps directly from raw sensor data rather than extracted features, enabling highly detailed reconstructions at the cost of higher computation.

2.3 Path Planning

Path planning is the task of computing a feasible and efficient trajectory that allows a robot to move from its current position to a desired goal while avoiding obstacles. Formally, the path planning problem can be expressed as the search for a sequence of robot states $\mathbf{x}_{1:T}$ that satisfies kinematic and dynamic constraints while minimizing a cost function, such as path length, energy consumption, or risk of collision.

In practice, path planning is typically separated into Global Planning and Local Planning, each serving a distinct but complementary role.

2.3.1 Global Planning

Global planning computes a path from the current position to a distant goal using a global map of the environment. Because it reasons over the entire map, it ensures completeness and optimality with respect to the chosen cost function. Global planning is especially effective in structured and static environments where the map is known and obstacles remain largely unchanged. Since the evaluation of the path is made globally on the entire map it requires high computational effort, making global planning slower and less suitable for frequent updates in dynamic settings. Global planning is typically executed at a lower frequency to provide a long-term reference path, which is later refined and adapted by local planning to account for short-term changes and unforeseen obstacles.

2.3.2 Local Planning

Local planning, often referred to as reactive planning, is responsible for generating short-term trajectories based on local sensor information. Its main objective is to guarantee immediate collision avoidance and smooth motion while following the trajectory suggested by the global plan. Local planning is reactive and allows the robot to respond dynamically to unexpected changes in the environment, such as moving obstacles or uncertainties in the map, maintaining a smooth local trajectory and providing control actions.

Chapter 3

State of the Art

3.1 Factor Graphs applications in Robotics

Factor graphs, as explained in details in [4] naturally capture the structure of a wide range of estimation and decision-making problems in robotics.

3.1.1 Tracking

In tracking, the goal is to estimate a trajectory over time given noisy measurements and motion models. This problem can be viewed as trajectory optimization or smoothing, with tracking as its incremental version. Under linear-Gaussian assumptions, it reduces to classical Kalman filtering and smoothing.

3.1.2 Switching Systems

Switching systems extend the tracking formulation by introducing discrete variables that select between different motion models. These models allow robots to adapt to changing dynamics but significantly increase computational complexity, as many possible mode sequences must be considered. A well-known example under linear-Gaussian assumptions is the switching linear dynamical system.

3.1.3 Optimal Control

Factor graphs can also encode decision-making problems such as planning and control. Here, factors represent system dynamics and cost functions on states and control inputs. Optimizing the resulting graph yields optimal control policies, with the classical linear quadratic regulator arising in the linear case. The same structure generalizes to nonlinear dynamics and objectives.

3.1.4 Pose Graph Optimization (PGO)

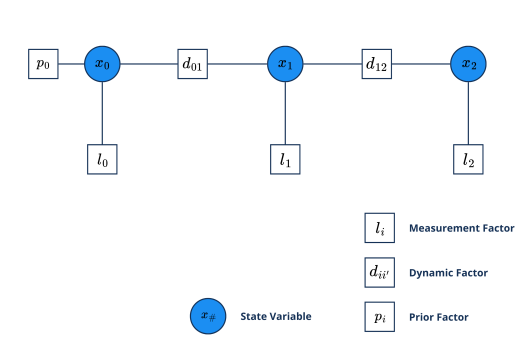
Pose graph optimization (PGO) estimates a robot's poses over time from relative pose measurements, often obtained from lidar, cameras, or inertial sensors. The graph is composed of binary factors between successive poses, with additional loop closure factors when the robot revisits a known location. PGO is a key building block for SLAM and structure-from-motion pipelines.

3.1.5 Simultaneous Localization and Mapping (SLAM)

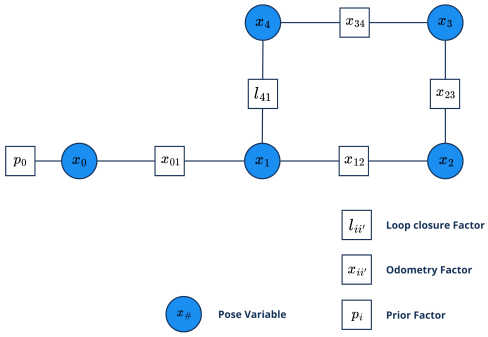
SLAM extends PGO by jointly estimating the robot's trajectory and the positions of landmarks in the environment. Landmark variables are connected to robot poses through measurement factors, such as bearing–range or camera observations. Visual SLAM is a common special case where a moving camera observes 2D projections of 3D landmarks.

3.1.6 Structure from Motion (SfM)

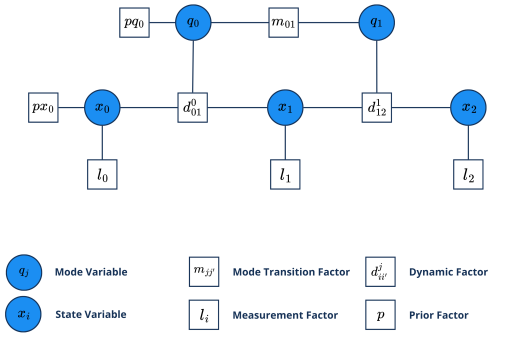
SfM, originating in computer vision, reconstructs 3D structures from collections of images taken from different viewpoints. Its factor graph resembles that of SLAM, but with additional variables for camera calibration parameters and typically without relative pose constraints. SfM has become a standard tool for 3D reconstruction from large, unstructured photo collections.



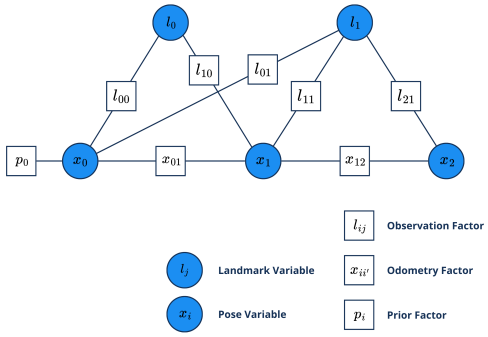
(a) Tracking



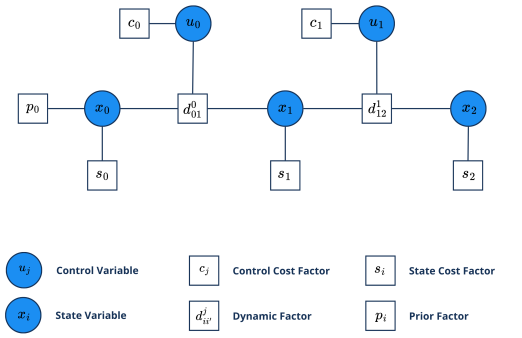
(b) Pose Graph Optimization



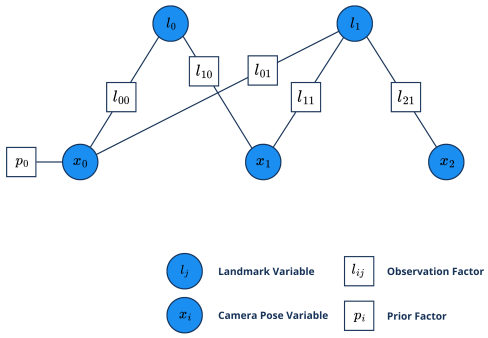
(c) Switching system



(d) Simultaneous Localization and Mapping



(e) Optimal control



(f) Structure from Motion

Figure 3.1: Factor graph applications

3.2 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics and autonomous systems, concerning the ability of a robot or vehicle to construct a representation or a map of an unknown environment while simultaneously determining its own position within that map. The concept was first introduced in the mid-1980s, emerging as one of the core challenges in autonomous navigation and perception. SLAM provides the foundation for enabling truly autonomous operation in environments where prior maps or external localization systems are unavailable or unreliable. The survey at [5] review the actual state of the art and new frontiers of SLAM.

The SLAM problem can be formally defined as a state estimation task: given a sequence of sensor measurements, the goal is to estimate both the trajectory of the robot and the spatial configuration of environmental features. This dual estimation introduces a strong coupling between localization, estimating the robot's pose, and mapping, estimating the environment, making SLAM a highly interdependent and nonlinear problem. In practical terms, accurate localization depends on the quality of the map, while building an accurate map requires precise localization.

Over the years, Simultaneous Localization and Mapping (SLAM) has evolved through several methodological paradigms.

3.2.1 Evolution of SLAM techniques

- **Filter-based Methods:** These include the Extended Kalman Filter (EKF-SLAM) and Particle Filter-based approaches like FastSLAM, which were dominant in the early 2000s. Such methods maintain a probabilistic belief over both the robot and map states, recursively updating them as new sensor measurements become available. However, they often suffer from scalability issues, as computational complexity increases quadratically with the number of landmarks, limiting their applicability to large-scale environments.

- **Optimization-based or Graph-based SLAM:** These approaches reformulate the estimation problem as a nonlinear least-squares optimization, where robot poses and landmarks are represented as variable nodes in a graph, and constraints, like odometry or sensor observations, are represented as factor nodes. Solving SLAM then reduces to minimizing the global error defined over this graph structure. This paradigm, thanks to the advances in sparse matrix optimization and numerical solvers, has established graph-based SLAM as the standard framework for modern large-scale mapping and localization systems.

- **Visual SLAM (VSLAM):** Visual SLAM extends these principles using visual data from cameras, providing dense and rich spatial information at a lower cost and power consumption compared to LiDAR. Prominent examples include MonoSLAM, ORB-SLAM, and Direct Sparse Odometry (DSO), which achieve accurate motion estimation and map reconstruction from image sequences. A complete review of visual SLAM methods can be found in [6]. Visual methods are particularly attractive for applications where lightweight, low-cost sensors are required, such as mobile robotics and augmented reality.

- **Visual-Inertial SLAM (VI-SLAM):** To further improve robustness against motion blur, feature loss, and scale ambiguity, Visual-Inertial SLAM (VI-SLAM) integrates visual information from cameras with inertial measurements from IMUs. This multimodal fusion enhances motion estimation accuracy and enables reliable operation in dynamic or GPS-denied environments, such as indoor navigation or aerial robotics.

3.2.2 Modern Trends and Challenges

Recent advancements in SLAM research increasingly leverage machine learning and deep neural networks to enhance key components such as feature extraction, loop closure detection, and semantic scene understanding. Deep learning-based feature descriptors and place recognition systems have improved the robustness of SLAM in challenging conditions, including dynamic or poorly illuminated environments.

Furthermore, semantic SLAM integrates high-level contextual understanding, such as object recognition, surface segmentation, and scene semantics, into the mapping process, enriching geometric maps with meaningful information for higher-level robotic reasoning and decision-making.

Despite significant progress, several open challenges remain. These include:

- Achieving real-time performance on computationally constrained platforms.
- Ensuring scalability and consistency in large-scale and dynamic environments.
- Managing uncertainty and drift over extended trajectories.
- Integrating semantic and geometric information without compromising accuracy or efficiency.

Additionally, the deployment of SLAM in real-world applications, such as autonomous driving, aerial robotics, and augmented or virtual reality, demands systems that are not only accurate but also robust, adaptable, and computationally efficient under diverse environmental conditions.

3.3 Active SLAM

Active Simultaneous Localization and Mapping extends traditional SLAM by integrating localization, mapping, and motion control into a unified decision-making framework. As explained in [7], while classical SLAM methods passively estimate the robot's trajectory and environment based on available sensor data, Active SLAM enables a robot to autonomously decide its future actions to optimize information gain and map quality. The objective is to build the most accurate and complete model of an unknown environment while ensuring efficient exploration and safe navigation.

Active SLAM can be interpreted as a decision-making process in which the robot must balance two competing objectives: exploration of new areas to expand the map and exploitation of known regions to improve map accuracy and localization confidence. This trade-off, known as the exploration–exploitation dilemma, lies at the core of Active SLAM research.

Historically, the concept of active perception dates back in the 1980s and 1990s, who emphasized the importance of actively acquiring information to achieve specific goals. In the context of robotics, related problems were initially studied independently as active mapping and active localization. Active mapping focuses on determining the next best view to improve environmental reconstruction, whereas active localization seeks the optimal robot motion that minimizes future pose uncertainty.

Recent advances in Active SLAM are being driven by developments in spatial perception, machine learning and deep learning. Neural network-based models have enabled predictive reasoning, improved uncertainty modeling, and enhanced decision-making in dynamic and unstructured environments. Furthermore, Active SLAM research increasingly incorporates ideas from planning under uncertainty, information theory, Reinforcement Learning and Graph Neural Networks, offering powerful frameworks for autonomous exploration.

3.4 Frontier detection

The concept of frontiers was first introduced in 1997 as a fundamental strategy for autonomous robot exploration. Frontiers are defined as the boundary between known free space and unexplored regions in an occupancy grid map. By navigating toward these frontier cells, a robot can systematically explore an unknown environment. Since its introduction, frontier-based exploration has been widely adopted in both single-robot and multi-robot systems.

In occupancy grid mapping, the environment is discretized into cells, each representing a physical location that can be in one of three states: unknown, free

space, or occupied. Frontier cells are those marked as free space and adjacent to at least one unknown cell. Detecting these cells efficiently is critical, as frontier detection often represents the computational bottleneck in real-time exploration. Faster frontier detection directly contributes to improved exploration performance, enabling the robot to make quicker and more informed decisions based on up-to-date map data.

Early approaches, referred to as Naïve Frontier Detection, examined every cell in the occupancy grid to determine whether it met the frontier condition. Although conceptually simple, this approach scales poorly with map size and is impractical for large environments due to its high computational cost.

To overcome these limitations, several optimized algorithms, described in detail in [8], have been proposed:

- **Wavefront Frontier Detection (WFD)**: Performs a Breadth-First Search (BFS) from the robot's current position through free-space cells until frontiers are found, significantly reducing the search space compared to the naive method.
- **Incremental WFD (WFD-INC)**: Restricts the BFS to the active area modified by the most recent sensor scans, making computation proportional to the updated region rather than the entire map.
- **Incremental-Parallel WFD (WFD-IP)**: Extends WFD-INC with parallel processing to further improve runtime efficiency.
- **Fast Frontier Detection (FFD)**: Evaluates only the edges of the latest scan, where new frontiers are likely to appear. Although efficient, its accuracy decreases for frontiers at the sensor's maximum range.
- **Oriented Bounding Box Frontier Detector (OBB-FD)**: Tracks updated cells and their prior states to maintain frontier consistency with minimal redundant computation.
- **Tree-based Frontier Detection**: Employs a random tree expansion from the robot's current position to locate frontiers efficiently, dynamically pruning explored branches to conserve memory.
- **Multi-Robot Frontier Detection**: Each robot detects frontiers locally, and these local maps are later merged into a global map for cooperative exploration.

Recent research has introduced hybrid and incremental frontier detection algorithms that improve both speed and accuracy. Examples include:

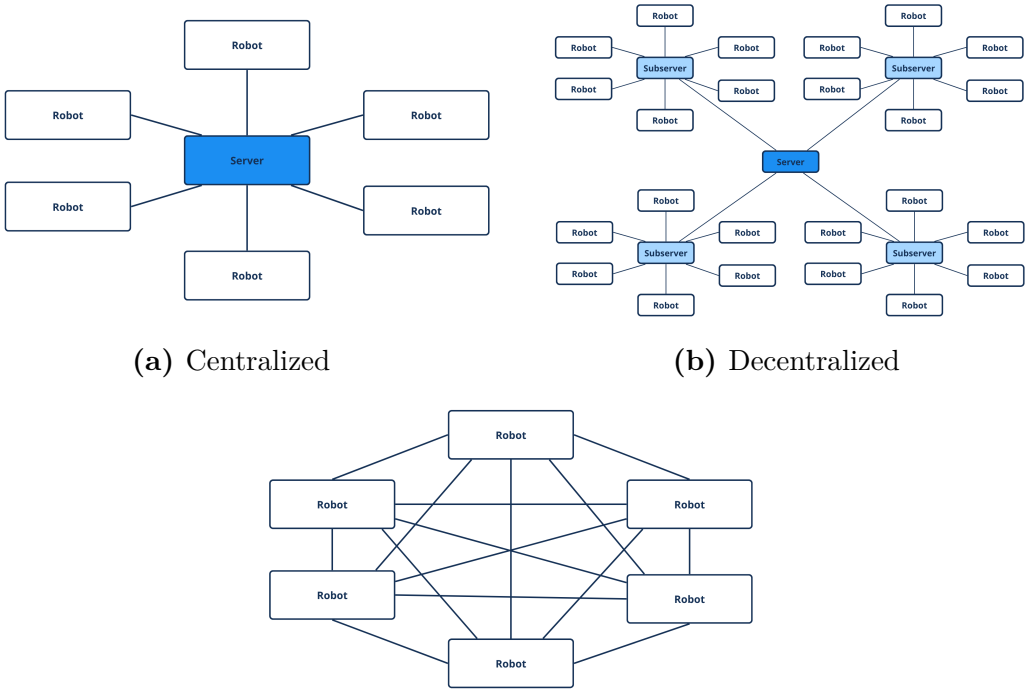
- **Naïve Active Area (NaïveAA)**: The Naïve Active Area algorithm focuses on evaluating only the active area of the map, which include the region modified by the most recent sensor scans, rather than the entire occupancy grid. During each update cycle, all cells within this active area are checked to determine whether they have become new frontier cells.
- **Expanding Wavefront Frontier Detection (EWFD)**: The EWFD algorithm extends the wavefront-based frontier detection paradigm. Initially, when no frontiers exist, the method performs a Breadth-First Search (BFS) from the robot's position to locate frontier cells adjacent to unknown space. Once an initial set of frontiers is established, subsequent iterations start BFS only from previously detected frontiers that fall within the new active area, marking them as unvisited and re-evaluating their status. This incremental process avoids re-examining the entire map and ensures that only updated regions are processed.
- **Frontier-Tracing Frontier Detection (FTFD)**: The Frontier-Tracing Frontier Detection (FTFD) algorithm leverages spatial relationships between previously known frontiers and the perimeter of the robot's latest sensor observation. The approach begins by identifying previous frontier cells within the active scan area and combining them with the endpoints of new sensor rays as the starting points for a BFS. During traversal, new frontiers are added, obsolete ones are removed, and obstacle-adjacent freespace cells are queued for further evaluation.

3.5 Active Collaborative SLAM

3.5.1 Network Topology

In AC-SLAM, the network topology defines how robots communicate and exchange data. The main approaches discussed in the literature are:

- **Centralized:** All data is transmitted through a single central server. This structure is simple but highly sensitive to server failures.
- **Decentralized:** Communication is organized through intermediate subservers or clusters, reducing load on the main server but still maintaining a degree of central dependence.
- **Distributed:** Robots exchange information directly with each other, removing the need for a central server. This approach is more robust but requires careful management of bandwidth, task allocation, and data sharing.



(a) Centralized

(b) Decentralized

(c) Distributed

Figure 3.2: AC-SLAM network topologies

3.5.2 Distributed AC-SLAM

Distributed Active Collaborative SLAM (AC-SLAM) focuses on enabling multi-robot teams to perform localization, mapping, and exploration without reliance on a central server. In such systems, each robot performs local computation and communicates directly with nearby peers through peer-to-peer links. This design enhances scalability, robustness, and fault tolerance, making it suitable for large-scale or communication-constrained environments.

However, achieving full autonomy in distributed teams remains challenging. A multi-robot system must simultaneously address several interdependent competencies such as localization, path planning, mapping, and coordinated information gathering. Each of these tasks is complex in isolation, and coupling them in a distributed fashion requires maintaining consistency and cooperation among all robots. For instance, accurate localization is essential for effective planning, while planning directly affects the success of coordinated exploration and data acquisition.

Recent research has proposed representing these interdependent competencies as layers within a unified factor graph framework. Each layer captures a distinct aspect of the robot's operation, and connections between layers encode how these aspects influence one another. This representation allows the system to jointly optimize multiple objectives while maintaining modularity and scalability.

The key innovation in distributed AC-SLAM lies in enabling robots to optimize their local copies of a shared global state asynchronously. Using Gaussian Belief Propagation (GBP), as employed in [9], robots exchange only partial or summarized information with their neighbors, iteratively refining their estimates until reaching a global consensus. This allows each robot to contribute to the joint estimation process based on its own observations while remaining robust to communication delays or losses.

Chapter 4

Methodology

This chapter presents the methodology adopted to address the active collaborative multi-robot SLAM problem, along with the corresponding implementation details.

As a preliminary step, the A-SLAM problem was first addressed in the context of a single-robot system. The solution was then extended to the multi-robot scenario by incorporating collaborative strategies and interactions among the robots.

4.1 Sensor Setup

This section describes the sensor setup employed on each robot, which provides the necessary perception capabilities for localization, mapping, and navigation. Each robot is equipped with:

- **LiDAR**: provides accurate 2D range measurements by emitting laser beams and measuring the time of flight of the reflected signals. This enables the estimation of distances to surrounding objects and the generation of geometric representations of the environment.

- **RGB-D Camera**: provides both color and depth information, allowing feature extraction and the generation of 3D landmarks for visual SLAM.

- **IMU**: delivers inertial measurements that support odometry estimation and improve robustness in feature-poor environments.

- **Wheel Encoders**: provides velocity feedback for motion estimation.

The data produced by these sensors form the input to the system architectures described in the following section.

4.2 Architectures

This section examines the overall system architectures in order to provide a comprehensive overview of the framework and its main components.

4.2.1 Single-robot architecture

Figure 4.1 shows the block diagram of the A-SLAM framework for a single-robot system. In this scenario, a single computational unit handles all processing tasks.

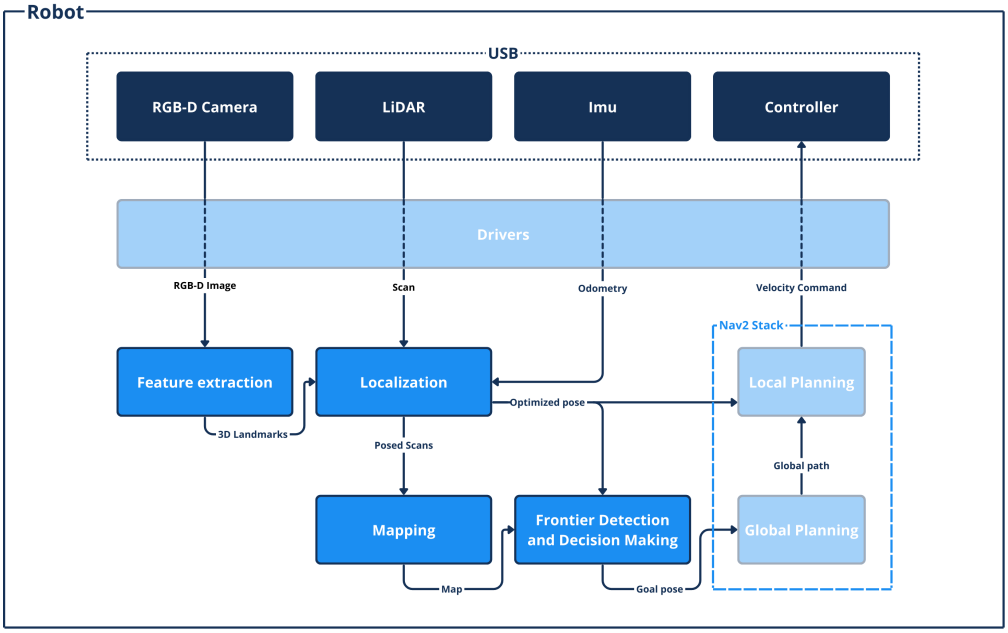


Figure 4.1: Single-robot architecture

Here is a detailed description of the role of each component:

- **Feature extraction:** 3D landmarks extraction from RGB-D images to allow accurate localization.
- **Localization:** integrates odometry, LiDAR scan, and visual landmarks to estimate the pose, aligning scans for subsequent processing.
- **Mapping:** builds and maintains a map for environment representation.
- **Frontier Detection and Decision Making:** identifies unexplored regions, known as frontiers, within the map, serving as candidate goals for exploration and assigns a frontier to the robot.

- **NAV2 Stack:** perform navigation given a frontier goal position executing Global Planning and Local Planning internal modules and sending the relative velocity command.

4.2.2 Multi-robot architecture

Figure 4.2 illustrates the block diagram of the proposed AC-SLAM framework in the multi-robot scenario. A centralized network topology is employed, introducing a centralized server that aggregates data from multiple robots and coordinates their exploration strategies and navigation directives.

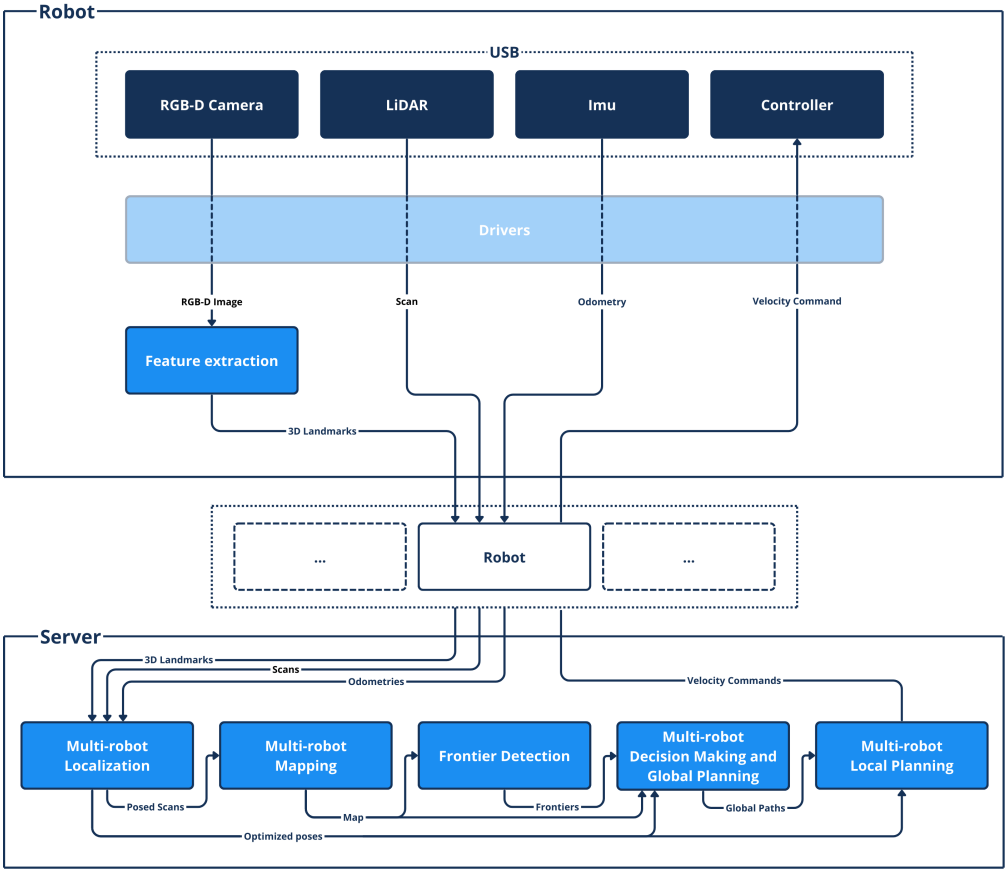


Figure 4.2: Multi-robot architecture

The scheme represents the framework distinguishing the components executed on the robots and those executed by the server, providing a view of the framework's division of responsibilities.

Each robot communicates with the server, providing the current 3D landmarks extracted from the RGB-D camera, LiDAR scans, and odometry measurements

while listening to velocity commands.

Here is a detailed description of the role of robot components:

- **Feature extraction:** 3D landmarks extraction from RGB-D images to allow accurate localization.

Here is a detailed description of the role of server components:

- **Multi-robot Localization:** integrates odometry, LiDAR scans, and visual landmarks from all robots to estimate their poses in a common reference frame. This module provides optimized poses and aligned scans for subsequent processing.
- **Multi-robot Mapping:** builds and maintains a shared global map by merging the contributions of individual robots. This allows consistent environment representation and supports collaborative exploration.
- **Frontier Detection:** identifies unexplored regions, known as frontiers, within the shared map, serving as candidate goals for exploration.
- **Multi-robot Decision Making and Global Planning:** assigns frontiers to different robots and computes global paths, ensuring efficient distribution of exploration tasks and avoiding redundancy.
- **Multi-robot Local Planning:** generates feasible and optimized local trajectories for each robot based on its assigned goal and the surrounding environment, while taking into account to each other positions, obstacles and kinematic constraints.

4.3 Feature extraction

The feature extraction process relies on RGB-D images, where each pixel provides both color and depth information. From the RGB image, a set of salient visual features (keypoints) is extracted using the ORB (Oriented FAST and Rotated BRIEF) detector from the OpenCV library, ensuring that the most distinctive image regions are selected. The corresponding depth map is then used to assign a metric 3D position to each keypoint, thus creating a set of landmarks in space.

Given a keypoint located at pixel coordinates (x, y) in the image plane and its associated depth value d, the 3D position (X, Y, Z) of the landmark in the camera reference frame is computed using the pinhole camera projection model:

$$Z = d, \quad X = \frac{(x - c_x) \cdot Z}{f_x}, \quad Y = \frac{(y - c_y) \cdot Z}{f_y} \tag{4.1}$$

where (f_x, f_y) are the focal lengths of the camera, (c_x, c_y) denotes the principal point (optical center), and d is the measured depth.

To ensure robustness, only landmarks within a valid depth range are considered:

$$Z_{\min} \leq Z \leq Z_{\max} \tag{4.2}$$

where $Z_{\min}$ and $Z_{\max}$ define the minimum and maximum sensing ranges of the depth sensor. Furthermore, landmarks near the image borders are discarded to avoid unreliable measurements. Finally, the resulting set of landmarks $\mathcal{L} = \{(X, Y, Z)\}$ provides a sparse representation of the surrounding environment. These landmarks serve as stable reference points that can be exploited by the localization module, enabling the robots to estimate their pose accurately within the space.

Figure 4.3: RGB-D image keypoint and feature extraction

4.4 SLAM

Simultaneous Localization and Mapping (SLAM) is the fundamental process that enables a robot system to build a consistent representation of the environment while simultaneously estimating robots trajectories within that environment. In the considered setup, localization and mapping tasks are formulated within a probabilistic framework, where environmental feature measurements extracted from the RGB-D camera and odometry information are fused to incrementally refine estimates of the robot poses and the positions of landmarks. Here are the implementation details explained about the localization, data association, and mapping tasks.

4.4.1 Localization

Localization refers to the problem of estimating the robot's trajectory given a sequence of observations from the environment together with motion or odometry constraints. The framework employed is GraphSLAM, which formulates this problem as an optimization over a factor graph.

In this formulation, robot poses and landmarks are represented as nodes in the graph, while constraints arising from prior knowledge, odometry measurements, and landmark observations are represented as factors. The solution to the localization problem is then obtained by optimizing the factor graph to produce the most likely trajectory consistent with all available information.

Before describing the variables and factors, here is a brief introduction to the mathematical spaces in which variables are defined and the relative operations.

The Special Euclidean Group $SE(3)$ is a space in which elements encode a 3D rotation and translation:

$$\mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \mathbf{R} \in SO(3),\ \mathbf{t} \in \mathbb{R}^3 \tag{4.3}$$

where $SO(3)$ is the space of 3D rotations and $\mathbf{t}$ is a translation vector in $\mathbb{R}^3$.

The difference operator $\ominus$ between elements $\mathbf{x}_a, \mathbf{x}_b \in SE(3)$ can be interpreted as the relative transformation between the two elements and is described as:

$$\mathbf{x}_a \ominus \mathbf{x}_b = \begin{bmatrix} \mathrm{Log}(\mathbf{R}_b^\top \mathbf{R}_a) \\ \mathbf{R}_b^\top (\mathbf{t}_a - \mathbf{t}_b) \end{bmatrix} \in \mathbb{R}^6 \tag{4.4}$$

where $\mathrm{Log}(\cdot)$ maps a rotation matrix to its 3D rotation vector.

The composition operator $\cdot$ between two elements $\mathbf{x}_a, \mathbf{x}_b \in SE(3)$ represents the composition of rigid body transformations:

$$\mathbf{x}_a \cdot \mathbf{x}_b = \begin{bmatrix} \mathbf{R}_a\mathbf{R}_b & \mathbf{R}_a\mathbf{t}_b + \mathbf{t}_a \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3) \tag{4.5}$$

Intuitively, making the composition of $\mathbf{x}_b$ by $\mathbf{x}_a$ applies the transformation of $\mathbf{x}_b$ first, followed by the transformation of $\mathbf{x}_a$.

The relative transformation from $\mathbf{x}_a$ to $\mathbf{x}_b$ is obtained using the inverse of $\mathbf{x}_a$:

$$\mathbf{x}_a^{-1} \cdot \mathbf{x}_b = \begin{bmatrix} \mathbf{R}_a^\top\mathbf{R}_b & \mathbf{R}_a^\top(\mathbf{t}_b - \mathbf{t}_a) \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3) \tag{4.6}$$

This operation gives the transformation of $\mathbf{x}_b$ expressed in the coordinate frame of $\mathbf{x}_a$.

The following paragraphs provide a detailed description of the variables and the different types of factors used in the graph-based formulation.

Pose variables The robot pose variables are elements of the Special Euclidean group $SE(3)$, which jointly represent a 3D rotation and translation. A pose $\mathbf{x}_i \in SE(3)$ at time i can be expressed as a homogeneous transformation matrix:

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{4.7}$$

where $\mathbf{R}_i \in SO(3)$ is the rotation matrix and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation vector. Thus, each pose variable encodes both the robot's orientation and position in the world frame.

Landmark variables Landmark variables represent fixed points in the environment observed by the robot. Each landmark $\mathbf{l}_j \in \mathbb{R}^3$ encodes its position in 3D space as a simple vector:

$$\mathbf{l}_j = \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix} \tag{4.8}$$

where X_j, Y_j, Z_j denote the coordinates of the landmark in the world frame.

Prior factors Prior factors encode the initial knowledge about the robot's pose at the start of the trajectory. Let the first pose be denoted by $\mathbf{x}_0 \in SE(3)$. A prior measurement is modeled as a Gaussian distribution with mean $\boldsymbol{\mu}_0 \in SE(3)$, denoting the initial guess, and covariance $\Sigma_0 \in \mathbb{R}^{6\times6}$, encoding the uncertainty in translation and rotation. The prior factor can be written as:

$$f_{\text{prior}}(\mathbf{x}_0) = \mathcal{N}\Big(\mathbf{x}_0 \;\Big|\; \boldsymbol{\mu}_0, \Sigma_0\Big) \propto e^{-\frac{1}{2}\|\mathbf{x}_0 \ominus \boldsymbol{\mu}_0\|^2_{\Sigma_0}} \tag{4.9}$$

where $\ominus$ denotes the pose difference operator in $SE(3)$.

A small covariance Σ_0 is assigned to reflect high confidence in the prior knowledge.

Odometry factors Odometry factors introduce constraints between consecutive poses based on relative motion estimates. Let $\mathbf{x}_i, \mathbf{x}_{i+1} \in SE(3)$ be consecutive poses and $\mathbf{u}_{i,i+1}$ the relative motion measurement obtained from the state estimation using wheel encoders and IMU sensor fusion. The measurement model is:

$$\mathbf{e}_{i,i+1} = \mathbf{u}_{i,i+1} \ominus (\mathbf{x}_i^{-1} \cdot \mathbf{x}_{i+1}) \tag{4.10}$$

with Gaussian noise

$$\mathbf{e}_{i,i+1} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\text{odom}}) \tag{4.11}$$

The corresponding odometry factor is:

$$f_{\text{odom}}(\mathbf{x}_i, \mathbf{x}_{i+1}) = \mathcal{N}\Big(\mathbf{e}_{i,i+1} \;\Big|\; \mathbf{0}, \Sigma_{\text{odom}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{e}_{i,i+1}\|^2_{\Sigma_{\text{odom}}}} \tag{4.12}$$

where $\Sigma_{\text{odom}} \in \mathbb{R}^{6\times6}$ encodes translational and rotational noise from wheel slip, IMU drift, and encoder errors.

Observation factors Observation factors connect robot poses to the positions of observed landmarks in the environment. Let $\mathbf{x}_i \in SE(3)$ be the robot pose at time i, and let $\mathbf{l}_j \in \mathbb{R}^3$ be the position of landmark j in the world frame. The measurement model is expressed as the difference between the landmark position and the robot pose:

$$\mathbf{e}_{ij} = \mathbf{l}_j - \mathbf{t}_i \tag{4.13}$$

where $\mathbf{t}_i \in \mathbb{R}^3$ is the translation component of the robot pose $\mathbf{x}_i \in SE(3)$.

Since observations can include outliers due to incorrect data association between new features and existing landmarks, the measurement noise is modeled with a Huber robust kernel to reduce the effects of outliers:

$$\mathbf{e}_{ij} \sim \mathcal{N}(\mathbf{0}, \Sigma_{\text{lm}}) \tag{4.14}$$

with Huber weighting applied to $\|\mathbf{e}_{ij}\|^2_{\Sigma_{\text{lm}}}$.

The corresponding observation factor is expressed in the factor graph as:

$$f_{\text{obs}}(\mathbf{x}_i, \mathbf{l}_j) = \mathcal{N}\Big(\mathbf{e}_{ij} \;\Big|\; \mathbf{0}, \Sigma_{\text{lm}}\Big) \propto e^{-\frac{1}{2}\rho\left(\|\mathbf{e}_{ij}\|^2_{\Sigma_{\text{lm}}}\right)} \tag{4.15}$$

where $\rho(\cdot)$ is the Huber loss function.

For a scalar residual e and a threshold $\delta > 0$, the Huber loss is defined as:

$$\rho(e) = \begin{cases} \frac{1}{2}e^2, & \text{if } |e| \leq \delta \\ \delta|e| - \frac{1}{2}\delta^2, & \text{if } |e| > \delta \end{cases}. \tag{4.16}$$

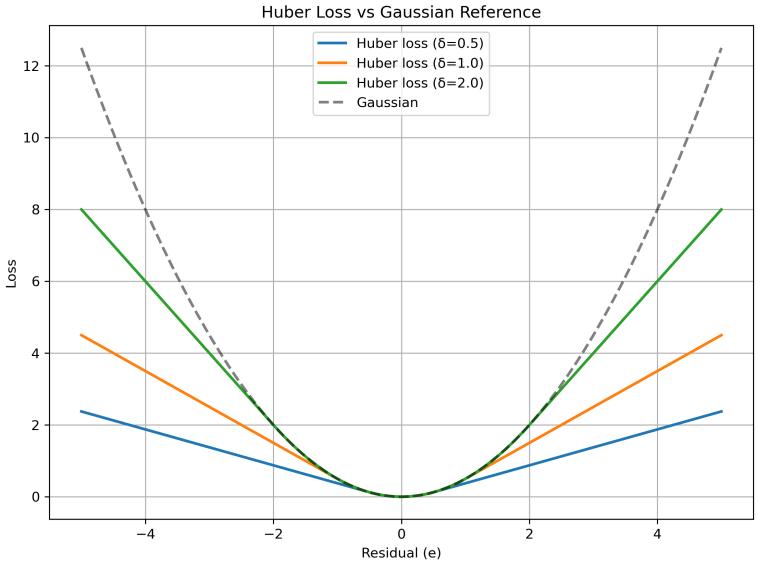


Figure 4.4: Huber loss function

Factor graph optimization The goal of GraphSLAM is to estimate the set of robot poses $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and landmark positions $\mathbf{L} = \{\mathbf{l}_0, \ldots, \mathbf{l}_M\}$ that maximize the posterior probability given all measurements.

Using the factorization provided by the factor graph, the joint posterior can be written as:

$$p(\mathbf{X}, \mathbf{L} \mid \text{measurements}) \propto \underbrace{f_{\text{prior}}(\mathbf{x}_0)}_{\text{prior factor}} \prod_{i=0}^{N-1} \underbrace{f_{\text{odom}}(\mathbf{x}_i, \mathbf{x}_{i+1})}_{\text{odometry factors}} \prod_{i,j} \underbrace{f_{\text{obs}}(\mathbf{x}_i, \mathbf{l}_j)}_{\text{observation factors}} \tag{4.17}$$

Equivalently, in the nonlinear least-squares form, the maximum a posteriori (MAP) estimate is obtained by minimizing the sum of squared residuals:

$$
\begin{aligned}
(\mathbf{X}^*, \mathbf{L}^*) = \arg\min_{\mathbf{X},\mathbf{L}} \Bigg\{ & \underbrace{\|\mathbf{x}_0 \ominus \boldsymbol{\mu}_0\|^2_{\Sigma_0}}_{\text{prior factor}} \\
& + \sum_{i=0}^{N-1} \underbrace{\|\mathbf{u}_{i,i+1} \ominus (\mathbf{x}_i^{-1} \cdot \mathbf{x}_{i+1})\|^2_{\Sigma_{\text{odom}}}}_{\text{odometry factors}} \\
& + \sum_{i,j} \underbrace{\rho\Big(\|\mathbf{l}_j - \mathbf{t}_i\|^2_{\Sigma_{\text{lm}}}\Big)}_{\text{observation factors}} \Bigg\}
\end{aligned}
\tag{4.18}
$$

To solve the optimization problem efficiently, the iSAM2 algorithm provided by the GTSAM library is employed. This incremental smoothing method updates the solution whenever new odometry or landmark factors are introduced, without requiring full re-optimization of the graph. As a result, the system achieves real-time localization performance while maintaining global consistency of both robot poses and landmark estimates.

Multi-robot factor graph optimization For multi-robot systems, the localization problem is formulated over a single, large factor graph that includes the poses and landmarks of all robots in the system. Let $\mathbf{X}^{(r)} = \{\mathbf{x}_0^{(r)}, \mathbf{x}_1^{(r)}, \ldots, \mathbf{x}_{N_r}^{(r)}\}$ denote the trajectory of robot r, and let $\mathbf{X} = \bigcup_r \mathbf{X}^{(r)}$ be the set of all robot poses. Similarly, let $\mathbf{L}^{(r)} = \{\mathbf{l}_1^{(r)}, \ldots, \mathbf{l}_{M_r}^{(r)}\}$ be the set of landmarks observed by robot r, and let $\mathbf{L} = \bigcup_r \mathbf{L}^{(r)}$ denote the set of all landmarks in the system. The joint posterior for all robots can then be factorized as:

$$
p(\mathbf{X}, \mathbf{L} \mid \text{measurements}) \propto \prod_r \left[\underbrace{f_{\text{prior}}(\mathbf{x}_0^{(r)})}_{\text{prior factor}} \prod_{i=0}^{N_r-1} \underbrace{f_{\text{odom}}(\mathbf{x}_i^{(r)}, \mathbf{x}_{i+1}^{(r)})}_{\text{odometry factors}} \prod_{i,j} \underbrace{f_{\text{obs}}(\mathbf{x}_i^{(r)}, \mathbf{l}_j)}_{\text{observation factors}} \right]
\tag{4.19}
$$

Equivalently, in the nonlinear least-squares form, the MAP estimate is obtained

by minimizing the sum of residuals for all robots:

$$
\begin{aligned}
(\mathbf{X}^*, \mathbf{L}^*) = \arg\min_{\mathbf{X},\mathbf{L}} \Bigg\{ \sum_r \Bigg[& \underbrace{\| \mathbf{x}_0^{(r)} \ominus \boldsymbol{\mu}_0^{(r)} \|^2_{\Sigma_0}}_{\text{prior factor}} \\
& + \sum_{i=0}^{N_r - 1} \underbrace{\| \mathbf{u}_{i,i+1}^{(r)} \ominus (\mathbf{x}_i^{(r)} \cdot \mathbf{x}_{i+1}^{(r)}) \|^2_{\Sigma_{\text{odom}}}}_{\text{odometry factors}} \\
& + \sum_{i,j} \underbrace{\rho\Big(\| \mathbf{l}_j - \mathbf{t}_i^{(r)} \|^2_{\Sigma_{\text{lm}}} \Big)}_{\text{observation factors}} \Bigg] \Bigg\}
\end{aligned}
\tag{4.20}
$$

This formulation naturally incorporates all robots' trajectories and relative landmarks into a single optimization problem.

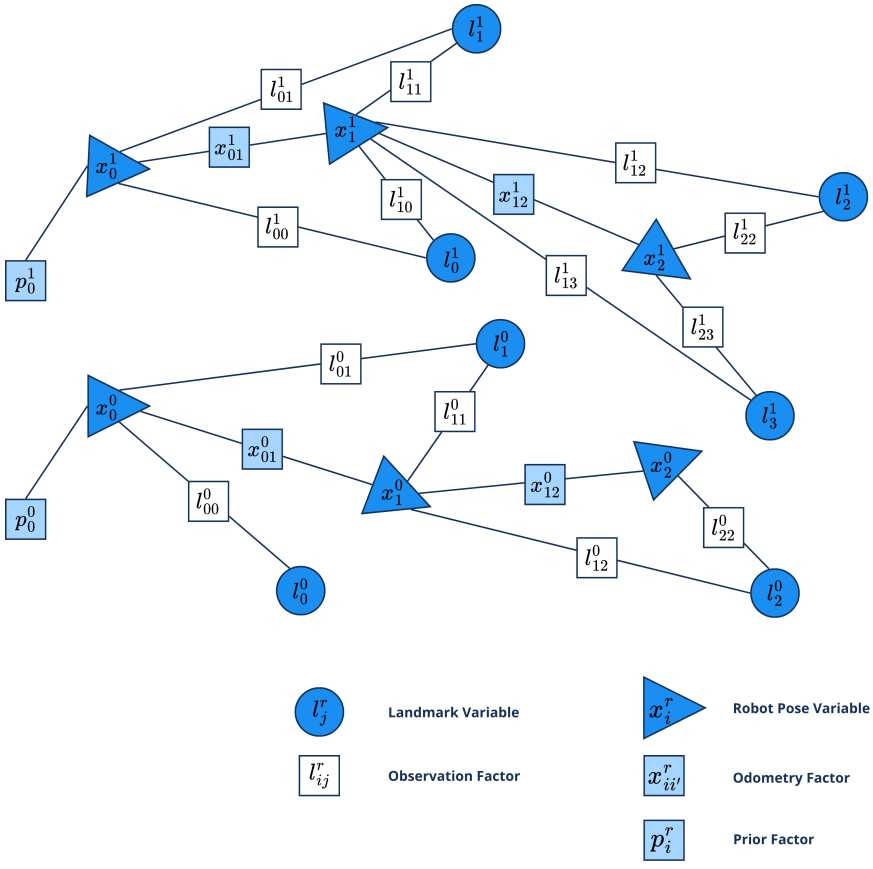


Figure 4.5: GraphSLAM factor graph formulation

LiDAR posed scans 2D LiDAR measurements provide geometric information about the environment. Each scan is associated with the estimated robot pose at the time of acquisition, resulting in a posed scan. Storing scans together with their reference pose enables the system to preserve spatial consistency between successive LiDAR acquisitions. These posed scans are employed concurrently to construct an accurate mapping of the environment.

4.4.2 Data association

A critical step in SLAM is to determine whether a newly observed feature corresponds to an existing landmark in the graph or should instead be inserted as a new node. This task, known as data association, is typically performed by comparing the newly observed features with the set of already estimated landmarks.

Non-probabilistic strategies, such as nearest-neighbor search with a k-d tree or Joint Compatibility Branch and Bound (JCBB), assign each measurement to the most likely landmark deterministically. While these approaches are robust in environments with sparse landmarks, they may fail in dense scenes where landmarks are close to each other, leading to ambiguous associations.

To address this, Probabilistic Data Association (PDA) is employed, which assigns to each candidate landmark a probability of being the correct correspondence, rather than committing to a single hard assignment. This formulation naturally accounts for measurement noise and landmark uncertainty.

Probabilistic Data Association Let $\mathbf{z}$ denote an observed feature in the robot's local frame, and let $\mathbf{l}_j$ be the position estimate of landmark j in the map with covariance Σ_j.

The innovation is defined as the Euclidean distance between the landmarks and the observation:

$$\mathbf{d}_{ij} = \mathbf{z} - \mathbf{l}_j \tag{4.21}$$

Landmarks with innovation over a certain threshold are discarded from the set of candidate associations.

The Mahalanobis distance between the observation and landmark is then

$$D_{ij} = \sqrt{\mathbf{d}_{ij}^{\top} \Sigma_j^{-1} \mathbf{d}_{ij}} \tag{4.22}$$

which measures how consistent the observation is with landmark j, relative to its covariance.

The likelihood of landmark j generating observation $\mathbf{z}$ is computed as:

$$\mathcal{L}_{ij} = \frac{1}{\sqrt{(2\pi)^d \, |\Sigma_j|}} \exp\!\left(-\tfrac{1}{2} D_{ij}^2\right) \tag{4.23}$$

where $d = 3$ is the dimension of the innovation $\mathbf{d}_{ij}$. The likelihood is shaped as a multivariate Gaussian where the covariance determinant $|\Sigma_j|$ ensures proper normalization, and the exponential term penalizes landmarks that are far in Mahalanobis distance.

Finally, after filtering landmarks with likelihood below a threshold, probabilities are obtained by normalizing the likelihoods across all candidate landmarks:

$$P_{ij} = \frac{\mathcal{L}_{ij}}{\sum_k \mathcal{L}_{ik}}, \tag{4.24}$$

so that $\sum_j P_{ij} = 1$. This yields a soft assignment, allowing each measurement to be distributed among multiple plausible landmarks.

In practice, a threshold is applied on the Euclidean distance $\|\mathbf{d}_{ij}\|$ between the new features and the landmarks to extract the set of candidate associations. Then the Mahalanobis distance D_{ij} and relative likelihood are evaluated for each landmark and low likelihood values are discarded. After a normalization of likelihood values across all candidates, probabilities are then used to make weighted soft association to landmarks in the factor graph.

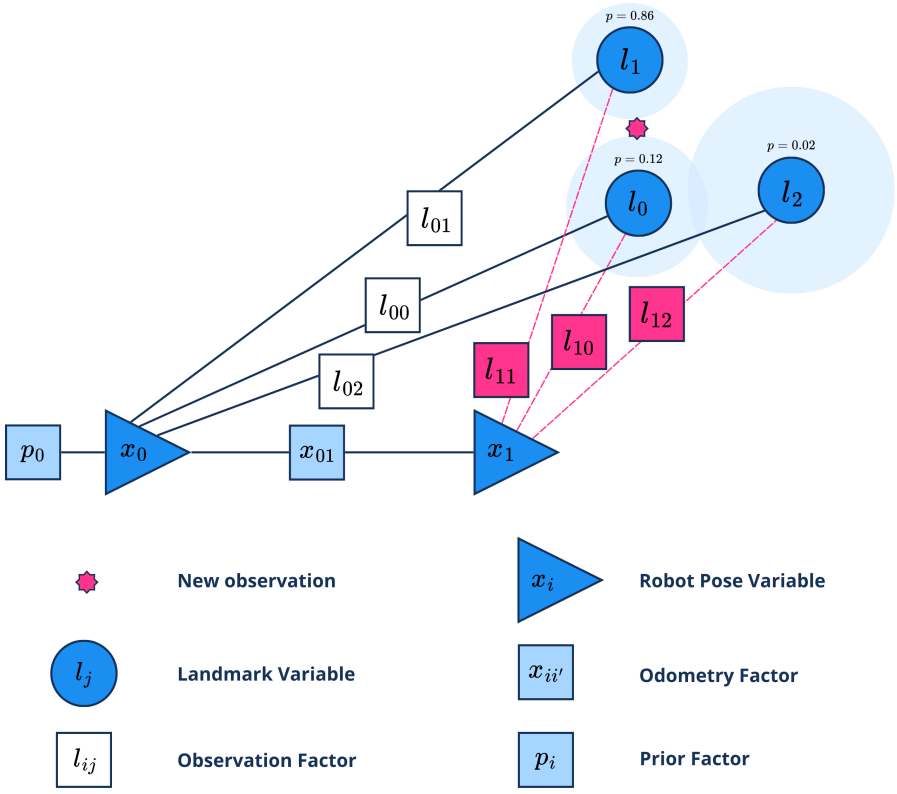


Figure 4.6: Probabilistic Data Association

4.4.3 Mapping

The mapping module constructs a consistent spatial representation of the environment from the sensor data collected by the robot system. The chosen representation is a probabilistic occupancy grid, where the environment is discretized into a two-dimensional array of cells, and each cell is assigned a probability of being occupied. This formulation naturally accounts for measurement uncertainty and has become a standard approach in mobile robotics. A global occupancy grid map is maintained in the world frame, integrating information from all robots. This unified map provides the necessary environment knowledge for subsequent tasks such as frontier detection, path planning, and obstacle avoidance.

Probabilistic Occupancy Grid Representation Each cell in the occupancy grid is modeled as a binary random variable, representing whether the cell is occupied or free. To allow recursive updates, the log-odds formulation is employed:

$$L_t = L_{t-1} + \log\left(\frac{p(z_t|m)}{1 - p(z_t|m)}\right) \tag{4.25}$$

where L_t is the log-odds of occupancy at time t, m is the map state, and z_t is the current sensor measurement. This formulation ensures that new measurements can be integrated additively and avoids numerical instabilities. The occupancy probability can be recovered from the log-odds using the logistic function:

$$p(m \mid z_{1:t}) = \frac{1}{1 + e^{-L_t}}. \tag{4.26}$$

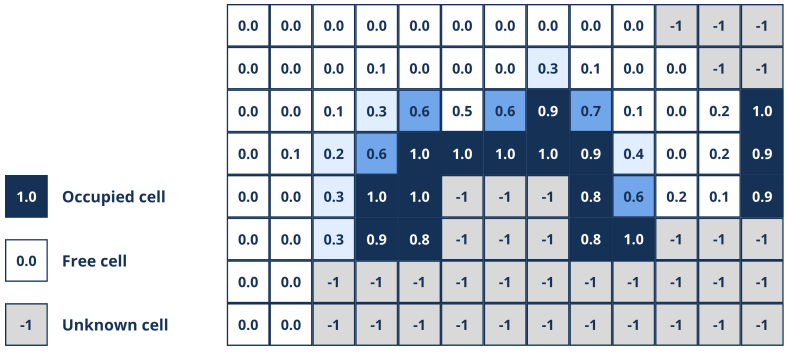


Figure 4.7: Probabilistic Occupancy Grid Map

Posed Scans integration LiDAR measurements are integrated into the probabilistic occupancy grid through the posed scans update concurrently by the localization process. A posed scan consists of the raw laser range data together with the corresponding robot pose estimate. By associating each scan with its pose, the mapping remains consistent with pose corrections from localization and enables incremental map construction.

To integrate LiDAR beams into the occupancy grid, the path of each beam is discretized into grid cells using the Bresenham line algorithm. Given a beam from the robot origin (x_r, y_r) to an endpoint (x_b, y_b) in grid coordinates, the goal of the algorithm is to determine the sequence of grid cells that approximate the continuous straight line segment.

The line can be described by the implicit equation

$$F(x, y) = (y_r - y_b)x + (x_b - x_r)y + (x_r y_b - x_b y_r) = 0 \tag{4.27}$$

where $\text{sign}(F(x, y))$ indicates on which side of the line a point (x, y) lies.

Bresenham's algorithm in an iterative algorithm that constructs the path incrementally along the dominant axis of the line.

Having initially $(x, y) = (x_r, y_r)$ let's define

$$\Delta x = |x_b - x_r|, \quad \Delta y = |y_b - y_r|, \quad s_x = \text{sign}(x_b - x_r), \quad s_y = \text{sign}(y_b - y_r) \tag{4.28}$$

and initialize the error term as

$$e = \Delta x - \Delta y \tag{4.29}$$

At each iteration, the current grid cell (x, y) is part of the ray. The algorithm then updates the error and the coordinates according to:

$$e_2 = 2e \tag{4.30}$$

$$\text{if } e_2 > -\Delta y: \quad e \leftarrow e - \Delta y, \quad x \leftarrow x + s_x \tag{4.31}$$

$$\text{if } e_2 < \Delta x: \quad e \leftarrow e + \Delta x, \quad y \leftarrow y + s_y \tag{4.32}$$

The procedure is repeated until $(x, y) = (x_b, y_b)$.

During this traversal, each visited cell along the ray, except for the final endpoint, is updated as a free cell since the LiDAR beam has passed through it without hitting an obstacle. Let $L(x, y)$ denote the log-odds value of the occupancy probability of cell (x, y). The free-space update follows

$$L(x, y) \leftarrow L(x, y) + \delta_{\text{free}} \cdot \Big(1 - \alpha d(x, y)\Big) \tag{4.33}$$

where $\delta_{\text{free}} < 0$ is the log-odds belief factor for free cells, $d(x, y)$ is the distance of the cell from the sensor along the ray, and α is a distance-dependent scaling factor that reduces the confidence in updates for distant cells.

At the beam endpoint (x_b, y_b), the grid is updated as occupied. To account for sensor uncertainty, the occupied update is applied not only to the endpoint but also to neighboring cells within a radius r using a Gaussian kernel:

$$L(x,y) \leftarrow L(x,y) + \delta_{\text{occ}} \cdot e^{-\frac{\|(x,y)-(x_b,y_b)\|^2}{2\sigma^2}} \cdot \Big(1 - \alpha d(x,y)\Big) \tag{4.34}$$

where $\delta_{\text{occ}} > 0$ is the log-odds increment for occupied cells, and σ controls the spread of the noise model.

This probabilistic update step models distance-dependent confidence and local uncertainty around obstacle boundaries.

Costmap generation In addition to the probabilistic occupancy grid, a costmap is generated to represent the traversability of the environment for safe navigation. Each occupied cell (x_o, y_o) in the occupancy grid acts as a source of potential risk. To model the effect of obstacles on nearby space, a distance-based decay function is applied to the surrounding cells, assigning high costs near obstacles and gradually lower costs with increasing distance

Let $C(x, y)$ denote the cost assigned to cell (x, y), and $d((x, y))$ the Euclidean distance to an obstacle cell (x_o, y_o). The cost contribution of a single obstacle can be expressed as

$$C_o(x,y) = C_{\max}\, e^{-\lambda\, d((x,y),(x_o,y_o))} \tag{4.35}$$

where $C_{\max}$ is the maximum cost assigned to obstacle cells, and λ is a decay parameter controlling how quickly the cost decreases with distance from the obstacle.

The final cost of a cell is obtained by combining the contributions from all nearby obstacles, typically via a max operation to ensure that the cell cost reflects the closest or most significant obstacle:

$$C(x,y) = \max_{(x_o,y_o)} C_o(x,y) \tag{4.36}$$

for each (x_o, y_o) in the set of all occupied cells in the occupancy grid within a specified radius r.

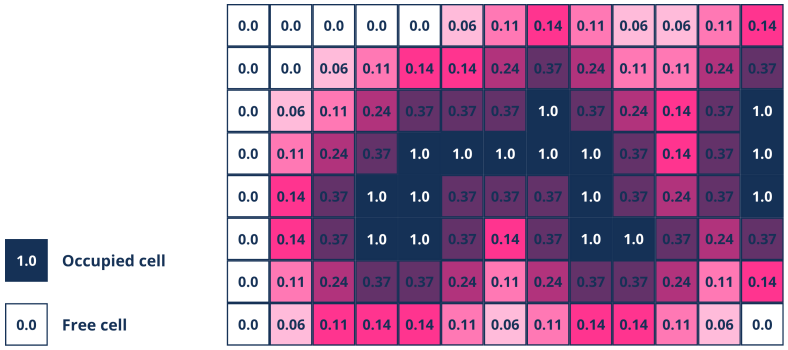


Figure 4.8: Costmap

4.5 Frontier detection

Frontiers are defined as regions at the boundary between known free space and unexplored areas. Frontier detection is performed by analyzing the occupancy grid and identifying cells that are adjacent to unknown regions.

A first strategy to extract frontier cells is the Expanding Wavefront Frontier Detection (EWFD) algorithm. Subsequently, a density-based clustering method (DBSCAN) is applied to group nearby frontier cells into coherent clusters, each representing a candidate exploration frontier.

Expanded Wavefront Frontier Detection The Expanded Wavefront Frontier Detection (EWFD) algorithm is an efficient strategy for detecting frontier cells by expanding wavefronts outward from the robot's current position. Unlike standard breadth-first search methods, EWFD reduces redundant computations by avoiding revisits to already explored regions and expanding only into newly discovered free cells. During this expansion, a frontier cell is identified whenever a free cell has at least one neighboring cell classified as unknown. The wavefront initially consists of the robot initial position cell and in the first iteration a Breadth First Search (BFS) is performed to find new frontier cells, keeping memory of already visited cells. By the second iteration the starting point of the BFS is not the robot position but the set of frontier cells, and is performed passing only through new explored cells, within an active exploration area, which radius is defined as the maximum sensor range. By incrementally propagating the wavefront only through new free explored space, the algorithm efficiently constructs a frontier map without exhaustively scanning the entire occupancy grid. This efficiency allow online exploration also in large-scale environments.

In order to adapt this strategy to a multi-robot system, each robot independently executes the EWFD algorithm within its own local occupancy grid, centered around its current pose, while having knowledge of globally visited cells. An active

exploration area is defined by a radius proportional to the maximum sensor range, ensuring that each robot only considers frontiers that are within its effective sensing capability. The results of local EWFD are then merged into a shared frontier grid, which maintains a global memory of visited cells. For each robot, the frontier cells detected locally are projected into the global frontier map representation, resolving conflicts between different robot detections by prioritizing newly confirmed frontiers when cells were previously globally marked as unknown, and identifying as free those cells visited by at least one robot.

This fusion process ensures consistency across the multi-robot system and prevents redundant exploration of already visited cells during the BFS step, increasing efficiency. Furthermore, by distributing the computation of EWFD and fusing the results into a unified frontier grid, the system leverages the parallel sensing capabilities of multiple robots, improving scalability.

Frontier Clustering Once frontier cells are identified, they are grouped into exploration frontiers using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. This clustering algorithm is well-suited for this task as it does not require the number of clusters to be specified in advance and is able to filter out isolated noise points, avoiding to identify frontiers with a poor granularity. The algorithm groups cells into clusters based on two parameters: the neighborhood radius ε and the minimum number of points n_p. Formally, a point p is a core point if its ε-neighborhood contains at least n_p points. Points within the ε-neighborhood of a core point are considered part of the same cluster, while points not reachable from any core point are treated as noise. DBSCAN clusters adjacent frontier cells into contiguous groups, each corresponding to an actual exploration frontier.

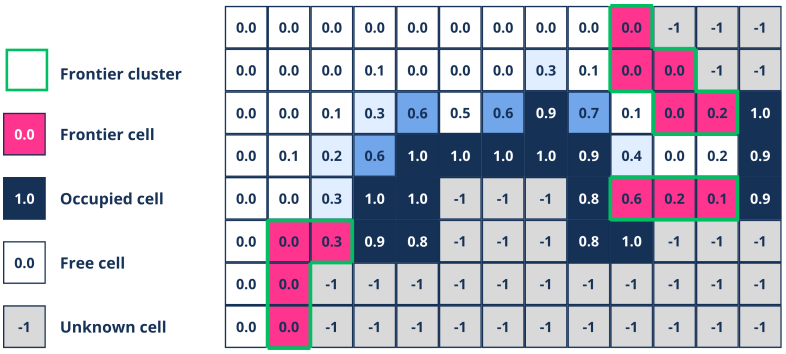


Figure 4.9: Frontier detection

4.6 Decision Making

In a multi-robot exploration system, decision making is responsible for assigning exploration tasks to each robot in a coordinated and efficient way. Each robot must autonomously select which frontier region to explore next, while ensuring that the overall system minimizes redundant coverage and maximizes the exploration rate.

Several factors influence this decision process, including the distance between each robot and the available frontiers, the size or information gain associated with each frontier, and the conflict avoidance between robots that may attempt to reach the same target region. Since these aspects are inherently coupled across multiple robots and frontiers, a global optimization framework is required to balance these objectives.

Discrete Factor Graph Formulation The assignment problem is formulated as a discrete optimization task represented through a factor graph, where variables correspond to robot assignments, and factors encode the cost or reward associated with each possible allocation. Let $\mathcal{R} = \{r_1, \ldots, r_N\}$ denote the set of robots and $\mathcal{F} = \{f_1, \ldots, f_M\}$ the set of frontier centroids. The goal is to find the optimal assignment $\mathbf{x}^* = \{x_1, \ldots, x_N\}$, where each $x_i \in \mathcal{F}$ corresponds to the frontier assigned to robot r_i.

The global cost function is expressed as:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \prod_{i=1}^{N} \phi_i(x_i) \prod_{i<j} \psi_{ij}(x_i, x_j) \tag{4.37}$$

where $\phi_i(x_i)$ represents the individual cost for robot r_i to reach frontier x_i, and $\psi_{ij}(x_i, x_j)$ models the pairwise compatibility between robots r_i and r_j.

Taking the negative log-likelihood formulation this additive cost minimization problem is obtained:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} -\log\left(\prod_{i} \phi_i(x_i) \prod_{i<j} \psi_{ij}(x_i, x_j) \right) \tag{4.38}$$

$$= \arg\min_{\mathbf{x}} \sum_{i=1}^{N} -\log \phi_i(x_i) + \sum_{i<j} -\log \psi_{ij}(x_i, x_j) \tag{4.39}$$

The unary cost captures both the distance to the frontier and its expected utility:

$$\phi_i(x_i) = e^{\underbrace{-\alpha\, d(r_i, x_i)}_{\text{distance cost}} + \underbrace{\beta\, s(x_i)}_{\text{size cost}}} \tag{4.40}$$

where $d(r_i, x_i)$ is the Euclidean distance between robot r_i and the potentially assigned frontier x_i, $s(x_i)$ denotes the size or information gain of the frontier, and

α, β are scaling parameters that control the influence of distance and frontier size, respectively.

To avoid the robots to have redundant assignments, pairwise factors introduce a penalty when two robots select the same frontier:

$$\psi_{ij}(x_i, x_j) = \begin{cases} e^{\lambda}, & \text{if } x_i = x_j \\ e, & \text{otherwise} \end{cases} \tag{4.41}$$

where $\lambda << 1$ is a conflict penalty factor that discourages multiple robots from targeting the same region.

Finally the optimization function is formulated as follows:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} \underbrace{\alpha\, d(r_i, x_i)}_{\text{distance cost}} - \underbrace{\beta\, s(x_i)}_{\text{size cost}} - \sum_{i<j} \underbrace{\psi_{ij}(x_i, x_j)}_{\text{uniqueness factor}} \tag{4.42}$$

This discrete factor graph formulation allows the optimization problem to be efficiently solved using probabilistic inference techniques, providing a globally consistent assignment of frontiers to robots. Compared to heuristic or greedy methods, the factor graph approach ensures scalability, robustness to uncertainty, and the ability to incorporate easily additional decision criteria such as energy consumption or communication constraints in a modular way.

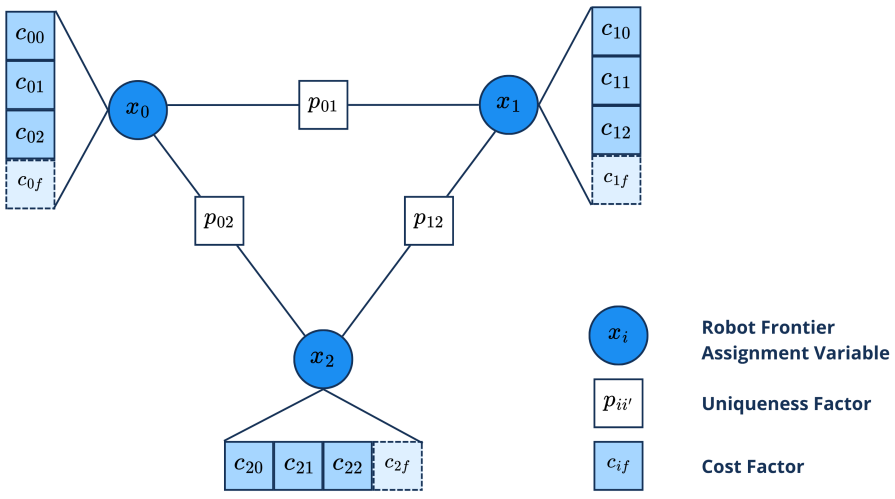


Figure 4.10: Decision-making factor graph

4.7 Global Planning

Once the objective frontiers are assigned to each robot, the global planning provides the high-level navigation strategy that determines the optimal path for each robot to reach its assigned frontier while avoiding static obstacles and minimizing traversal costs. In this framework, the global path planning problem is formulated as a discrete graph search on the costmap representation of the environment, where each cell corresponds to a node in the graph, and edges connect adjacent traversable cells. The A* algorithm is adopted due to its optimality, completeness, and computational efficiency when guided by an admissible heuristic.

A* Search Algorithm The A* algorithm finds the path that minimizes the total traversal cost between a start node n_s and a goal node n_g. At each step, A* evaluates nodes according to a cost function:

$$f(n) = g(n) + h(n) \tag{4.43}$$

where $g(n)$ represents the accumulated cost from the start node to n, and $h(n)$ is a heuristic estimate of the cost to reach the goal from n. The algorithm expands nodes in order of increasing $f(n)$, ensuring that the first time the goal is reached, the corresponding path is optimal if $h(n)$ never overestimates the true cost-to-go.

As the search progresses, the cost-to-come $g(n)$ is incrementally updated when moving from node n to a neighboring node n' as:

$$g(n') = g(n) + c(n, n') \tag{4.44}$$

where $c(n, n')$ is the traversal cost associated with the edge between n and n'. This cost reflects the risk of crossing that region of the map, which in this case is cost defined by the costmap in the moving cell, describing the proximity to obstacles.

The resulting optimal path $\mathcal{P}^* = \{n_s, \ldots, n_g\}$ is obtained by recursively selecting the sequence of nodes that minimize $f(n)$ until the goal is reached:

$$\mathcal{P}^* = \arg\min_{\mathcal{P}} \sum_{n \in \mathcal{P}} c(n) \tag{4.45}$$

subject to connectivity constraints between consecutive nodes. The A* algorithm guarantees that $\mathcal{P}^*$ is the least-cost path provided that the heuristic $h(n)$ is admissible and consistent.

Heuristic Function The heuristic $h(n)$ provides an estimate of the remaining distance to the goal, guiding the search efficiently toward the target. Typical

admissible heuristics include:

$$h_{\text{Manhattan}}(n) = |x_n - x_g| + |y_n - y_g| \tag{4.46}$$

$$h_{\text{Euclidean}}(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \tag{4.47}$$

where (x_n, y_n) are the coordinates of node n and (x_g, y_g) are the coordinates of the goal. The heuristic function ensures that the search is both directed and efficient, with the choice depending on whether the robot moves along four or eight directions. To make the robots move on eight directions in the costmap the Euclidean heuristic results to be more effective.

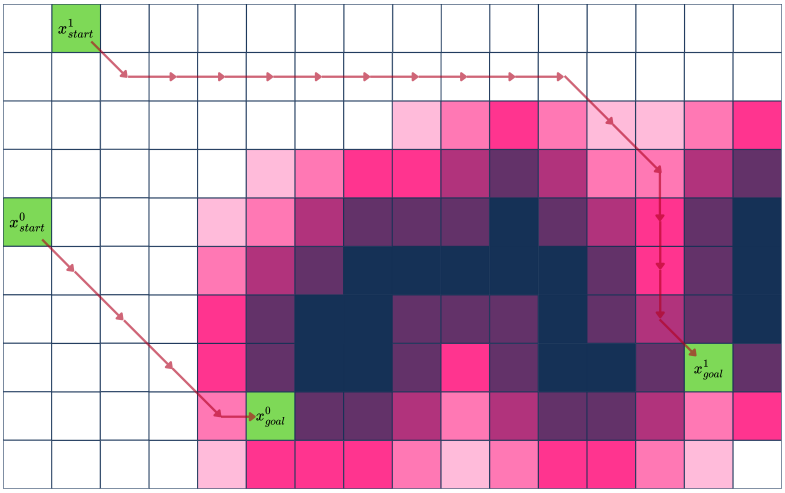


Figure 4.11: A* Search

This global planning approach ensures that each robot efficiently navigates toward its assigned frontier while avoiding collision with static obstacles detected during mapping task. The global planning will be further integrated with a local motion controller for real-time navigation and collision avoidance with each other robot.

4.8 Local planning

The local planning module bridges the gap between the global planner and the robot's low-level motion control. While global planning provides the overall path toward the assigned frontier for each robot, local planning is responsible for generating dynamically feasible and collision-free velocity commands that allow the robot to safely follow the path in real time. Its objectives include ensuring smooth trajectory tracking, maintaining obstacle avoidance within the dynamic environment, and preserving inter-robot safety in shared exploration spaces. To satisfy these requirements, a first approach adopts the Dynamic Window Approach (DWA). A second approach employs a factor graph optimization framework as a factor graph-based local planner solution.

Dynamic Window Approach The Dynamic Window Approach (DWA) is a well-established reactive control technique that selects the optimal linear and angular velocities (v, ω) according to the robot's dynamic constraints, ensuring real-time feasibility and safety.

At every control cycle, DWA evaluates a discrete set of velocity commands within the dynamically admissible velocity space:

$$\mathcal{V}_{\text{adm}} = \Big\{ (v_p, \omega_p) \;\Big|\; v_{\min} \le v_p \le v_{\max}, \quad \omega_{\min} \le \omega_p \le \omega_{\max}, \\ |v_p - v| \le a_v\, T_p, \quad |\omega_p - \omega| \le a_\omega\, T_p \Big\} \tag{4.48}$$

where (v, ω) are the current velocities, a_v and a_ω represent the maximum linear and angular accelerations, and T_p is the time prediction horizon. For each admissible velocity pair, the corresponding trajectory is forward simulated using the differential-drive kinematics:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v_p \cos\theta \\ v_p \sin\theta \\ \omega_p \end{bmatrix} \tag{4.49}$$

Denoting as $\mathbf{x} = [x, y, \theta]$ the current state of the robot, the predicted state $\mathbf{x_p} = [x_p, y_p, \theta_p]$ over a short time horizon T_p for a candidate trajectory is defined as:

$$\mathbf{x_p} = \mathbf{x} + \dot{\mathbf{x}} \cdot T_p = \begin{bmatrix} x + T_p v_p \cos\theta \\ y + T_p v_p \sin\theta \\ \theta + T_p \omega_p \end{bmatrix} \tag{4.50}$$

Each trajectory is evaluated through a multi-objective scoring function that balances progress toward the goal, obstacle avoidance, and coordination with other

robots:

$$\begin{aligned} J(v_p, \omega_p) = & \, w_p \, f_{\text{progress}}(v_p, \omega_p) \\ & + w_h \, f_{\text{heading}}(v_p, \omega_p) \\ & + w_c \, f_{\text{costmap}}(v_p, \omega_p) \\ & + w_r \, f_{\text{robot}}(v_p, \omega_p) \end{aligned} \tag{4.51}$$

The contributions of each term in the objective function are described as follows:

- f_{progress} measures advancement along the global path, favoring trajectories that move the robot closer to its next waypoint.

$$f_{\text{progress}}(v, \omega) = 1 - \frac{d_p^{\text{goal}}}{d^{\text{goal}}} = \frac{\sqrt{(x_p - x_{\text{goal}})^2 + (y_p - y_{\text{goal}})^2}}{\sqrt{(x - x_{\text{goal}})^2 + (y - y_{\text{goal}})^2}} \tag{4.52}$$

- f_{heading} evaluates the angular alignment between the robot's orientation and the desired heading toward the goal.

$$f_{\text{heading}}(v, \omega) = 1 - \frac{|\Delta\theta|}{\pi}, \quad \Delta\theta = (\theta_{\text{goal}} - \theta) \in [-\pi, \pi] \tag{4.53}$$

- f_{costmap} penalizes proximity to obstacles using the costmap representation, thus ensuring safety margins.

$$f_{\text{costmap}}(v, \omega) = -\left(1 - \frac{d_p^{\text{obs}}}{d_{\text{max}}^{\text{obs}}}\right) \tag{4.54}$$

  where d_p^{obs} is the distance between the predicted robot state and the closest obstacle costmap cell and $d_{\text{max}}^{\text{obs}}$ is the maximum range in which the obstacle factor is evaluated.

- f_{robot} introduces inter-robot repulsion, preventing collisions and maintaining safe distances in multi-robot contexts.

$$f_{\text{robot}}(v, \omega) = -(1 - \frac{d_p^{\text{robot}}}{d_{\text{max}}^{\text{robot}}}) \tag{4.55}$$

  where d_p^{robot} is the distance between the predicted robot state and the closest robot in its current position and $d_{\text{max}}^{\text{robot}}$ is the maximum range in which the inter-robot repulsion factor is evaluated.

The optimal velocity command (v^*, ω^*) is obtained as:

$$(v^*, \omega^*) = \arg \max_{(v,\omega) \in \mathcal{V}_{\text{adm}}} J(v, \omega), \tag{4.56}$$

subject to kinematic and dynamic feasibility constraints. This ensures that the chosen control inputs lead to smooth, safe, and dynamically achievable motion.

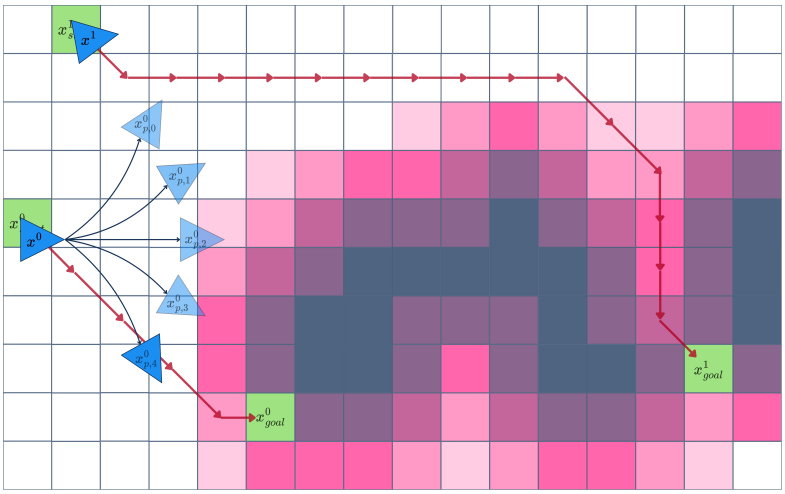


Figure 4.12: Dynamic Window Approach

A significant limitation of the Dynamic Window Approach (DWA) is the possibility of some robots becoming stuck without having a good prediction in certain conditions. This occurs because the predicted trajectory over the short time horizon T_p is generated assuming a constant velocity command (v, ω), without reaching a good long-term prediction. As a result, the robot may fail to explore alternative motions that could escape narrow passages or situations with multiple nearby robots, potentially causing it to oscillate or stop indefinitely.

Factor Graph-based Local Planner Inspired by [10], the Factor Graph-based Planner (FGP) formulates the local planning problem as a nonlinear least-squares optimization problem, where the robot trajectory over a finite horizon is represented as a sequence of discrete states connected through factors that encode dynamic feasibility, goal attraction, obstacle avoidance, and inter-robot interactions.

Each robot trajectory is discretized into N time steps over a prediction horizon $T_p = N\,\Delta t$, where Δt denotes the discretization time step. Robot states in the factor graph are represented as $N + 1$ variable nodes, including the start and goal

nodes. The state of robot r at time step i is defined as:

$$\mathbf{x}_i^r = \begin{bmatrix} x_i^r \\ y_i^r \\ v_{x,i}^r \\ v_{y,i}^r \end{bmatrix}, \quad i = 0, \ldots, N \tag{4.57}$$

Prior factors, binary factors between consecutive state variables of the same robot and binary factors between state variables of different robots in the same timestep impose constraints respectively related to obstacle avoidance, motion dynamics, and inter-robot collision avoidance. Accordingly, the local planning problem can be formulated as a probabilistic inference task in which the optimal joint trajectory $\mathbf{X}^*$ maximizes the overall likelihood of all factors in the graph:

$$\begin{aligned} \mathbf{X}^* \;=\; \arg\max_{\mathbf{X}} \prod_{r\in\mathcal{R}} \Bigg[& \underbrace{f_{\text{start}}^r\Big(\mathbf{x}_0^r\Big)}_{\text{start prior}} \cdot \underbrace{f_{\text{goal}}^r\Big(\mathbf{x}_N^r\Big)}_{\text{goal prior}} \\ & \cdot \prod_{i=0}^{N-1} \underbrace{f_{\text{dyn},i}^r\Big(\mathbf{x}_i^r, \mathbf{x}_{i+1}^r\Big)}_{\text{dynamics factors}} \cdot \prod_{i=0}^{N} \underbrace{f_{\text{obs},i}^r\Big(\mathbf{x}_i^r; \mathcal{M}\Big)}_{\text{obstacle factors}} \\ & \cdot \prod_{i=1}^{N} \prod_{s\neq r} \underbrace{f_{\text{int},i}^{r,s}\Big(\mathbf{x}_i^r, \mathbf{x}_i^s\Big)}_{\text{inter-robot factors}} \Bigg] \end{aligned} \tag{4.58}$$

Below is a detailed description of the formulation of each factor and their role in the optimization framework.

Start prior factors The start prior anchors the initial state of each robot trajectory to the robot's current measured state. This factor acts as a probabilistic prior:

$$f_{\text{start}}^r(\mathbf{x}_0^r) = \mathcal{N}\Big(\mathbf{x}_0^r \;\Big|\; \bar{\mathbf{x}}_0^r, \Sigma_{\text{start}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{x}_0^r - \bar{\mathbf{x}}_0^r\|^2_{\Sigma_{\text{start}}}} \tag{4.59}$$

where $\bar{\mathbf{x}}_0^r$ is the estimated current state and Σ_{start} encodes the confidence in the state estimate. Low covariance values in Σ_{start} express high certainty about the initial robot state.

Dynamics factors The dynamics factor enforces smooth and dynamically consistent motion between consecutive states. Assuming a constant-velocity kinematic model, the expected next state is predicted as:

$$\Phi(\mathbf{x}_i^r, \Delta t) = \begin{bmatrix} x_i^r + v_{x,i}^r \Delta t \\ y_i^r + v_{y,i}^r \Delta t \\ v_{x,i}^r \\ v_{y,i}^r \end{bmatrix}, \tag{4.60}$$

and the corresponding dynamics error is defined as:

$$\mathbf{e}_{\text{dyn},i}^r = \mathbf{x}_{i+1}^r - \Phi(\mathbf{x}_i^r, \Delta t). \tag{4.61}$$

The resulting factor is modeled as a Gaussian model that penalizes deviations from the expected motion model:

$$f_{\text{dyn},i}^r(\mathbf{x}_i^r, \mathbf{x}_{i+1}^r) = \mathcal{N}\Big(\mathbf{x}_{i+1}^r \;\Big|\; \Phi(\mathbf{x}_i^r, \Delta t), \Sigma_{\text{dyn}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{e}_{\text{dyn},i}^r\|_{\Sigma_{\text{dyn}}}^2} \tag{4.62}$$

This factor penalizes large accelerations or deviations from the motion model, promoting temporal smoothness and ensuring dynamically consistent trajectories that remain feasible for the robot's kinematic limits.

Obstacle factors The obstacle factor ensures collision avoidance by penalizing proximity to occupied regions in the local costmap. For each predicted state, the error is defined based on the distance to the nearest obstacle within a safety radius:

$$\mathbf{e}_{\text{obs},i}^r = \begin{cases} 1 - \dfrac{d^{\text{obs}}(\mathbf{x}_i^r)}{d_{\text{max}}^{\text{obs}}}, & \text{if } d^{\text{obs}}(\mathbf{x}_i^r) \le d_{\text{max}}^{\text{obs}} \\ 0, & \text{if } d^{\text{obs}}(\mathbf{x}_i^r) > d_{\text{max}}^{\text{obs}} \end{cases} \tag{4.63}$$

where $d^{\text{obs}}(\mathbf{x}_i^r)$ is the Euclidean distance from the robot's position to the nearest obstacle within the safety radius $d_{\text{max}}^{\text{obs}}$.

The corresponding factor is modeled as a Gaussian potential:

$$f_{\text{obs},i}^r(\mathbf{x}_i^r; \mathcal{M}) = \mathcal{N}\Big(\mathbf{0} \;\Big|\; \mathbf{e}_{\text{obs},i}^r, \Sigma_{\text{obs}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{e}_{\text{obs},i}^r\|_{\Sigma_{\text{obs}}}^2} \tag{4.64}$$

penalizing states that violate the safe distance and guiding the robot through collision-free regions.

If no obstacle is detected within the safety radius at time step i, the corresponding obstacle factor is omitted from the graph.

Inter-robot factors The inter-robot factor promotes cooperative and collision-free navigation among multiple robots. For each pair of robots, the error is defined based on their relative distance and a safety threshold:

$$\mathbf{e}_{\text{int},i}^{r,s} = \begin{cases} 1 - \dfrac{\|\mathbf{p}_i^r - \mathbf{p}_i^s\|}{d_{\max}^{\text{robot}}}, & \text{if } \|\mathbf{p}_i^r - \mathbf{p}_i^s\| \leq d_{\max}^{\text{robot}} \\ 0, & \text{if } \|\mathbf{p}_i^r - \mathbf{p}_i^s\| > d_{\max}^{\text{robot}} \end{cases} \tag{4.65}$$

where $\mathbf{p}_i^r$ and $\mathbf{p}_i^s$ denote the 2D positions of robots r and s at time step i, and $d_{\max}^{\text{robot}}$ is the minimum allowed distance for the error to be evaluated.

The corresponding factor is modeled as a Gaussian potential:

$$f_{\text{int},i}^{r,s}(\mathbf{x}_i^r, \mathbf{x}_i^s) = \mathcal{N}\Big(\mathbf{0} \;\Big|\; \mathbf{e}_{\text{int},i}^{r,s}, \Sigma_{\text{int}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{e}_{\text{int},i}^{r,s}\|_{\Sigma_{\text{int}}}^2} \tag{4.66}$$

penalizing states that violate the safety margin with collision risk with other robots.

If no other robot lies within the safety radius at time step i, no inter-robot factor is included.

Goal prior factors The goal prior attracts the terminal state of the trajectory toward a short-term target, previously computed as a lookahead point along the global path. The lookahead point is found as the point along the global path with a distance from the current robot position equal to:

$$d_{\text{lookahead}} = T_p \cdot v_{\max} \tag{4.67}$$

It provides directional guidance within the local planning horizon while maintaining flexibility, allowed by moderate values of covariances inside Σ_{goal}, to adapt to dynamic obstacles or other robots:

$$f_{\text{goal}}^r(\mathbf{x}_N^r) = \mathcal{N}\Big(\mathbf{x}_N^r \;\Big|\; \mathbf{x}_{\text{goal}}^r, \Sigma_{\text{goal}}\Big) \propto e^{-\frac{1}{2}\|\mathbf{x}_N^r - \mathbf{x}_{\text{goal}}^r\|_{\Sigma_{\text{goal}}}^2} \tag{4.68}$$

The tuning of Σ_{goal} values is really important to balance smoothness with performance in terms of velocity, ensuring also a good flexibility for obstacle and other robots collision avoidance.

The maximization of the joint likelihood can be equivalently expressed as a

nonlinear least-squares problem:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \Bigg\{ \sum_{r\in\mathcal{R}} \Bigg[\underbrace{\|\mathbf{x}_0^r - \bar{\mathbf{x}}_0^r\|^2_{\Sigma_{\text{start}}}}_{\text{start prior}} + \underbrace{\|\mathbf{x}_N^r - \mathbf{x}_{\text{goal}}^r\|^2_{\Sigma_{\text{goal}}}}_{\text{goal prior}} + \sum_{i=0}^{N-1} \underbrace{\|\mathbf{x}_{i+1}^r - \Phi(\mathbf{x}_i^r, \Delta t)\|^2_{\Sigma_{\text{dyn}}}}_{\text{dynamics factors}} + \sum_{i=0}^{N} \underbrace{\|f_{\text{obs},i}^r(\mathbf{x}_i^r; \mathcal{M})\|^2_{\Sigma_{\text{obs}}}}_{\text{obstacle factors}} + \sum_{i=1}^{N} \sum_{s\neq r} \underbrace{\|f_{\text{int},i}^{r,s}(\mathbf{x}_i^r, \mathbf{x}_i^s)\|^2_{\Sigma_{\text{int}}}}_{\text{inter-robot factors}} \Bigg] \Bigg\} \tag{4.69}$$

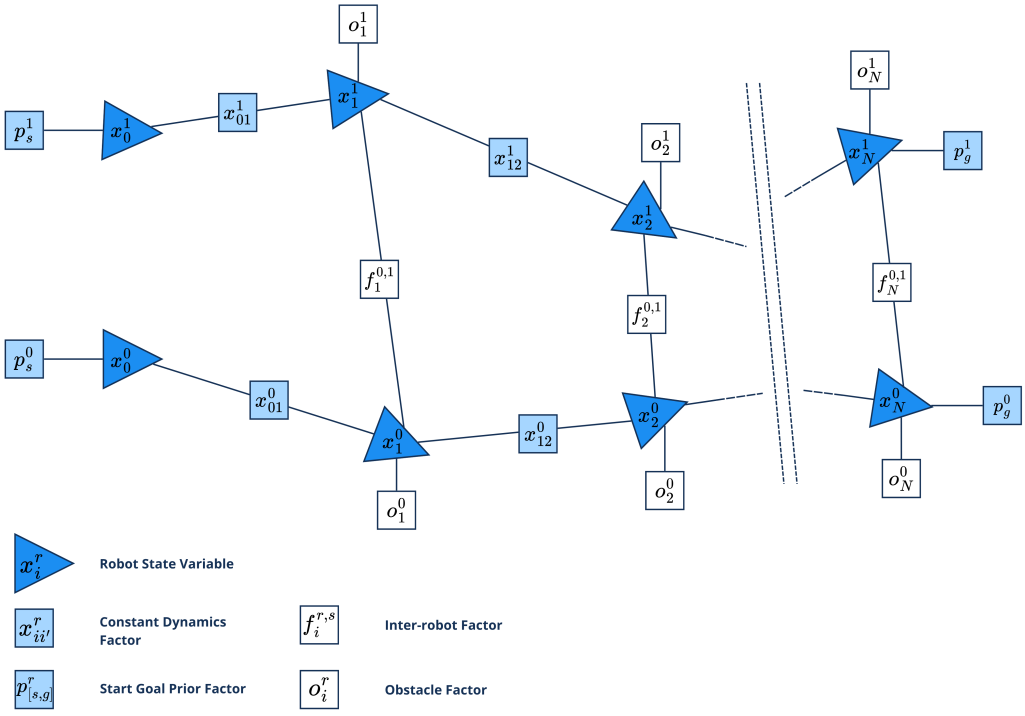


Figure 4.13: Factor graph-based local planner formulation

After solving the optimization problem, the first control command for each robot is derived from the displacement between the first two optimized states. The linear velocity is computed as the Euclidean displacement over the discretization interval:

$$v^r = \frac{\sqrt{(x_1^r - x_0^r)^2 + (y_1^r - y_0^r)^2}}{\Delta t} \tag{4.70}$$

The angular velocity is obtained from the difference between the heading implied by the velocity components at the current state and the direction of motion toward the next state:

$$\omega^r = \frac{\text{atan2}(y_1^r - y_0^r,\, x_1^r - x_0^r) - \text{atan2}(v_{y,0}^r,\, v_{x,0}^r)}{\Delta t} \tag{4.71}$$

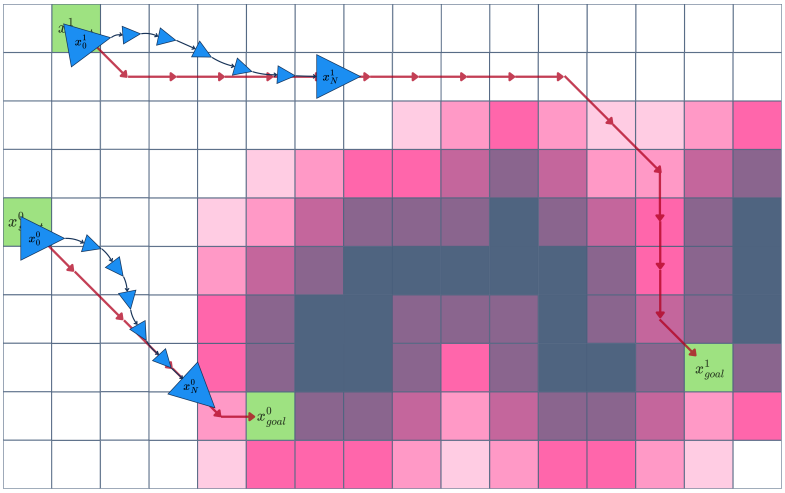


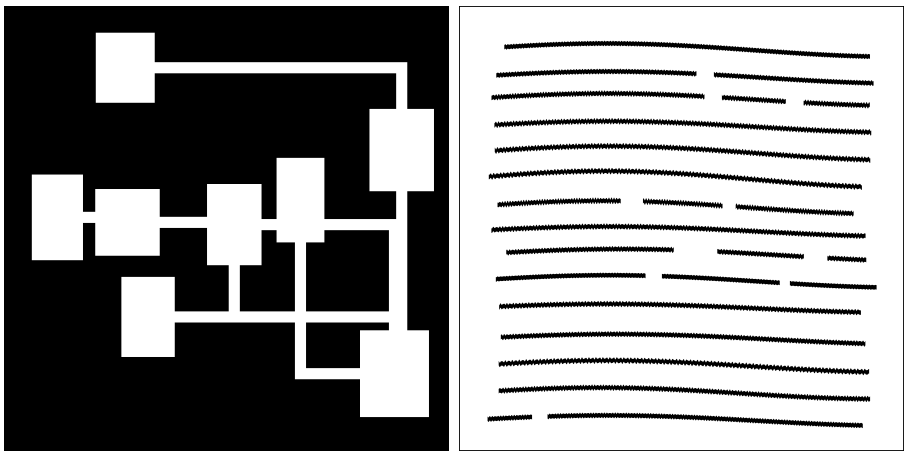
Figure 4.14: Factor graph-based local planner

This receding-horizon control strategy updates the command at each iteration using the most recent optimized trajectory, enabling responsive and coordinated multi-robot motion while ensuring dynamically feasible and collision-free trajectories. The factor graph framework further allows the seamless integration of additional constraints such as velocity bounds and safety margins in a unified probabilistic formulation.

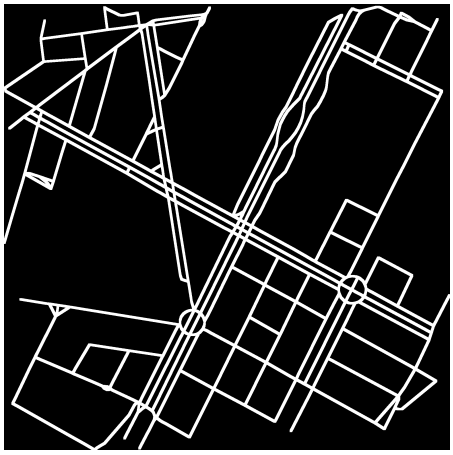
4.9 Simulation

A simulation environment was developed to evaluate the algorithms before the deployment on a real multi-robot system.

The environment is represented as an occupancy grid map, where each cell corresponds to a discrete portion of the world with a defined resolution. To test the algorithms in different conditions, random map generators are employed to create environments such as vineyard-like maps and indoor dungeon-style maps. Additionally, maps are derived from real-world road networks using OpenStreetMap data. These generators are fully parametric, enabling the design of multiple testing scenarios with varying levels of complexity.



(a) Indoor map **(b)** Vineyard-like map



(c) Road network map

Figure 4.15: Random generated simulation environments

The simulator adopts a multi-threaded architecture in which each robot runs

several concurrent threads to emulate both its internal state evolution and its sensors. One thread is responsible for updating the robot's ground-truth state and IMU-based estimate at a fixed frequency, while additional threads emulate the LiDAR and camera sensors at realistic update rates. Each of these threads also accounts for sensor noise, biases, and limitations, ensuring that the simulated data closely resembles real-world measurements. The following describes the function of each thread executed by every robot:

- **Robot state and IMU thread:** This thread is responsible for updating both the ground-truth state of the robot and the IMU-based estimate. At each iteration, the ground-truth pose is advanced using the commanded velocities according to:

$$x_{t+1} = x_t + v\cos(\theta_t)\Delta t, \tag{4.72a}$$
$$y_{t+1} = y_t + v\sin(\theta_t)\Delta t, \tag{4.72b}$$
$$\theta_{t+1} = \theta_t + \omega\Delta t. \tag{4.72c}$$

  where v and ω are the linear and angular velocity commands, respectively, and Δt is the integration step.

  The IMU-based pose is then updated using a noisy motion model that introduces both systematic and stochastic errors. Sensor biases evolve according to a first-order Gauss–Markov process:

$$b_t = \alpha\, b_{t-1} + \mathcal{N}(0, \sigma_b^2), \tag{4.73}$$

  where b_t is the bias at time t, $\alpha \in [0,1]$ is the correlation coefficient, and $\mathcal{N}(0, \sigma_b^2)$ represents Gaussian noise. The measured linear and angular velocities are then corrupted by scale errors and biases:

$$v_m = (1 + \epsilon_v)v + b_v, \tag{4.74}$$
$$\omega_m = (1 + \epsilon_\omega)\omega + b_\omega, \tag{4.75}$$

  and integrated using the same kinematic equations employed for the robot state update (4.72) to obtain the IMU trajectory. This process produces realistic drift and noise, essential for testing localization algorithms.

- **LiDAR thread:** This thread simulates a 2D LiDAR by casting laser beams into the occupancy grid and performing ray tracing. At each update, the positions and shapes of other robots are temporarily added to the map so they can be perceived as obstacles. For every beam, the global LiDAR position is computed from the robot's pose, and a ray is traced using the Bresenham algorithm to check for collisions. If a hit is detected, the intersection point

is transformed into the LiDAR frame and stored; otherwise, the beam is marked as infinite. Random noise is added to the range and angle of each beam before the final scan is published.

- **Camera thread:** The camera is modeled for landmark detection. At initialization, fixed landmarks are randomly generated in the free space of the map. At each update, the relative positions of landmarks within the sensor's range and field of view are reported, with Gaussian noise added to simulate real measurement uncertainty.

Figure 4.16 show the simulator visualization with Rviz tool where LiDAR scans are represented as points in red, orange and yellow respectively for *robot_0*, *robot_1* and *robot_2* and landmarks are marked as blue points.

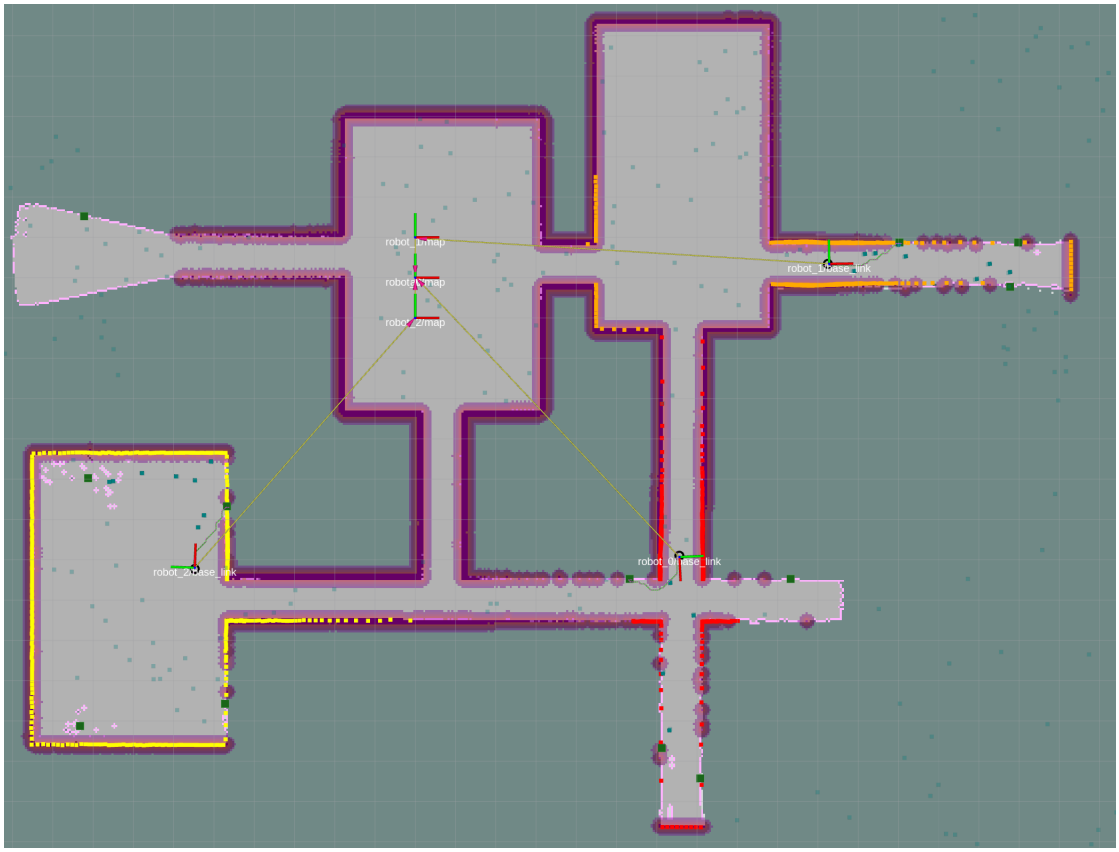


Figure 4.16: Simulator Rviz visualization

4.10 Implementation

The entire system has been developed using the ROS2 Humble framework, which offers a modular, distributed, and real-time architecture well-suited for robotic applications. Its support for publish-subscribe communication, real-time execution, and cross-platform compatibility makes it ideal for multi-robot SLAM systems like the one proposed here.

The core functionalities are implemented in C++, ensuring efficiency and real-time performance in computationally demanding tasks.

For visualization and debugging, Rviz is employed, allowing an intuitive representation of the robot states, sensor data, and mapping results during both simulation and real-world experiments.

In order to guaranty portability, reproducibility, and efficient resource management across different robotic platforms, the entire software stack is fully containerized using Docker. This virtualization approach simplifies deployment, facilitates system integration, and ensures consistent execution in heterogeneous environments.

Chapter 5

Experiments and Results

In this chapter, the experimental validation of the proposed multi-robot exploration framework is presented. The objective of these experiments is to evaluate the performance, and scalability of the system under different environmental conditions and operational scenarios.

The evaluation encompasses all the main components described in the previous chapter, including localization, mapping, frontier detection, decision making, and both global and local planning, analyzing how they interact to achieve coordinated and efficient exploration.

The experiments are designed to demonstrate the multi-robot system's ability to:

- construct consistent and accurate maps through cooperation
- detect and assign frontiers efficiently among the robots
- plan globally optimal and locally safe trajectories to reach the frontiers
- achieve robust and fast navigation with real-time coordination and cooperation

Different real-world scenarios are used to assess the framework's behavior in semi-structured environments, varying in complexity, obstacle density, and map topology. Quantitative metrics such as exploration time, localization accuracy, exploration rate and map accuracy are measured to provide an objective comparison between the different multi-robot setups explained below. Qualitative results are also reported through trajectory visualizations and occupancy maps, showing exploration for each robot employed in the experiments and illustrating the overall system performance.

This chapter is organized as follows:

- **Multi-robot system:** description of the multi-robot system with hardware and software configurations.
- **Experimental environments:** specification of the testing environments and evaluation objectives.
- **Ground truth:** definition of how ground truth data is obtained for performance evaluation.
- **Testing modality:** explanation of the procedure employed to perform the experiments.
- **Results and discussion:** presentation and analysis of experimental results, highlighting key findings and insights.

5.1 Multi-robot system

The experimental platform consists of a team of three TurtleBot3 robots, denoted as *robot_12*, *robot_13*, and *robot_14*, configured to perform fully autonomous and cooperative exploration tasks within indoor environments. Each robot is equipped with on-board control unit, perception, and actuation modules enabling distributed sensing and navigation while maintaining wireless communication with the server.

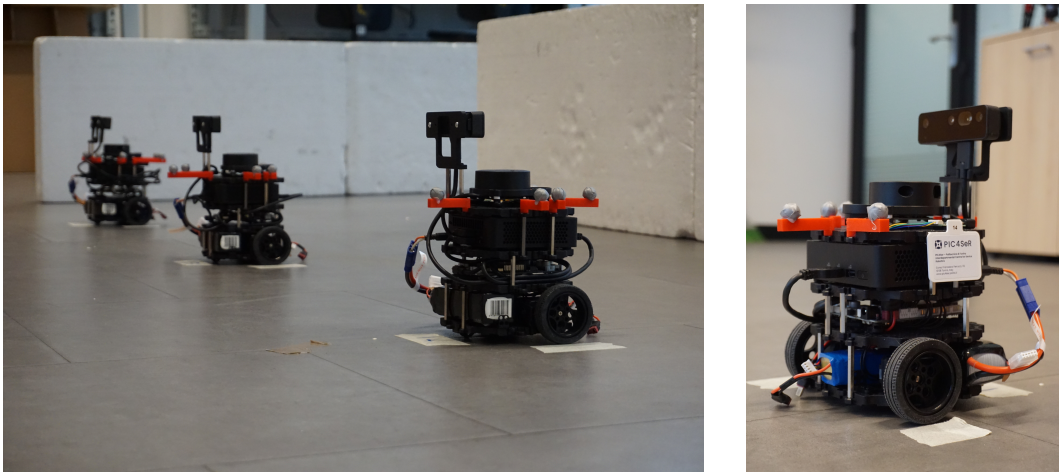


Figure 5.1: Multi-robot TurtleBot3 system with *robot_12*, *robot_13*, and *robot_14*

Hardware configuration Each TurtleBot3 unit is equipped with a set of hardware components designed to support autonomous perception, computation, and actuation. The main elements of the system are summarized as follows:

- **LiDAR sensor:** an LDS-02 2D LiDAR scanner provides 360° horizontal coverage with a maximum range of 8 m, an angular resolution of approximately 1°, and distance accuracy of ±10 mm within 0.3 m, ±3% between 0.3 m and 6 m, and ±5% between 6 m and 8 m. Operating at 10 Hz, it delivers dense range measurements used for obstacle detection and occupancy grid mapping.
- **RGB-D camera:** an OAK-D Pro stereo depth camera integrates an RGB sensor with an active IR projector, enabling accurate depth estimation and robust visual perception. This sensor is used for landmark detection and 3D mapping tasks.
- **Differential-drive configuration:** each robot employs a two-wheel differential drive system powered by independently actuated DC motors with wheel encoders.
- **Control unit:** a compact Intel NUC computer equipped with an Intel Core i3 processor, running Ubuntu 22.04 and the ROS 2 framework. This onboard unit handles communication with the server, sensor handling and feature extraction and motion control.
- **Low-level control board:** motor control and encoder feedback are managed by an OpenCR 1.0 microcontroller board, which directly interfaces with the motor drivers and wheel encoders to execute velocity commands and provide real-time odometric data.
- **Power system:** two independent batteries supply power separately to the OpenCR board and the control unit.

The system follows a centralized multi-robot architecture, illustrated in Figure 4.2. A central server, equipped with an Intel Core i7 processor, manages the global coordination, localization, mapping, frontier detection, decision making, global and local planning, while each robot executes local computation for perception and motion control.

Communication and coordination Communication between the robots and the central server is managed through the ROS 2 communication framework, operating over a wireless LAN. This setup enables reliable message exchange with low latency, ensuring real-time coordination, data sharing, and synchronization among all robots and the central control unit.

5.2 Experimental environments

The experimental evaluation is conducted within indoor environments constructed specifically in the laboratory of the PoliTo Interdepartmental Center for Service Robotics (PIC4SeR). Each environment is designed to progressively increase in spatial complexity, number of frontiers, and possible exploration paths, allowing for a systematic assessment of the efficiency and scalability of the proposed AC-SLAM multi-robot exploration framework. The environments are thus configured to test how the system adapts its frontier detection, assignment, and navigation behavior under varying levels of environmental complexity.

The figures below show the four test environments employed during the experiments, illustrating the expected distribution of frontiers and the corresponding qualitatively feasible paths for the robots. These visualizations provide an overview of the exploration scenarios and highlight the increasing difficulty and coverage requirements of each test.

Figure 5.2 show the first experiment environment in which there is a single initial frontier and the at most two frontiers to discover during the exploration process.

Figure 5.3 show the second experiment environment with three starting exploration frontiers, one for each robot when tested with all three robots.

Figure 5.4 show the third experiment environment with a slightly more complex structure and four initial frontiers.

Figure 5.5 shows the fourth experiment environment with a more labyrinth-style structure and five initial frontiers to explore.

A fifth experiment is conducted on a big area of the PIC4SeR floor including the laboratory and other rooms.

The experiments are designed to evaluate the performance of the multi-robot system when operating with different numbers of agents (one, two, or three robots), analyzing how the number of frontiers and their spatial distribution affect the overall exploration efficiency, coverage time, and coordination dynamics with respect to the number of agents employed.

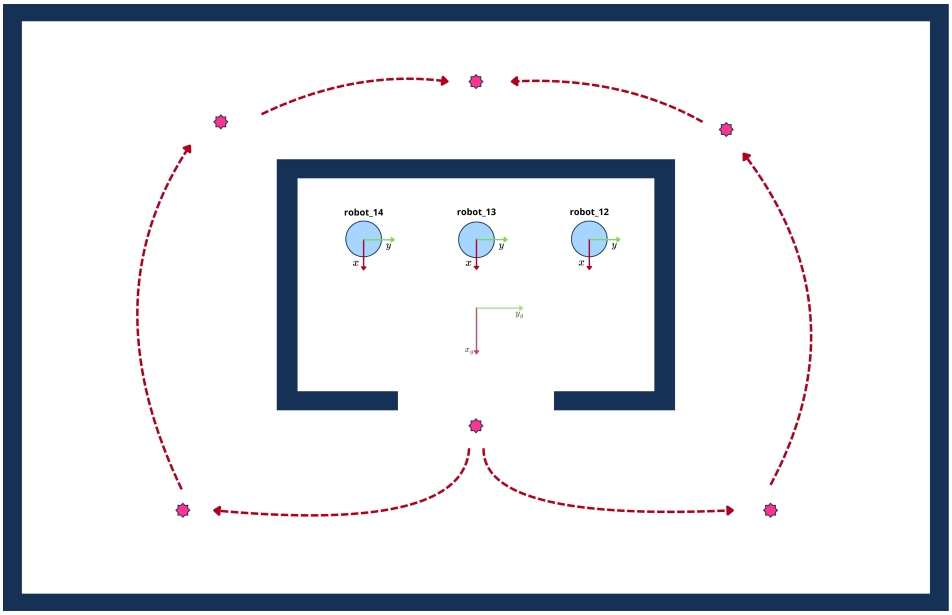


Figure 5.2: Experiment 1 environment

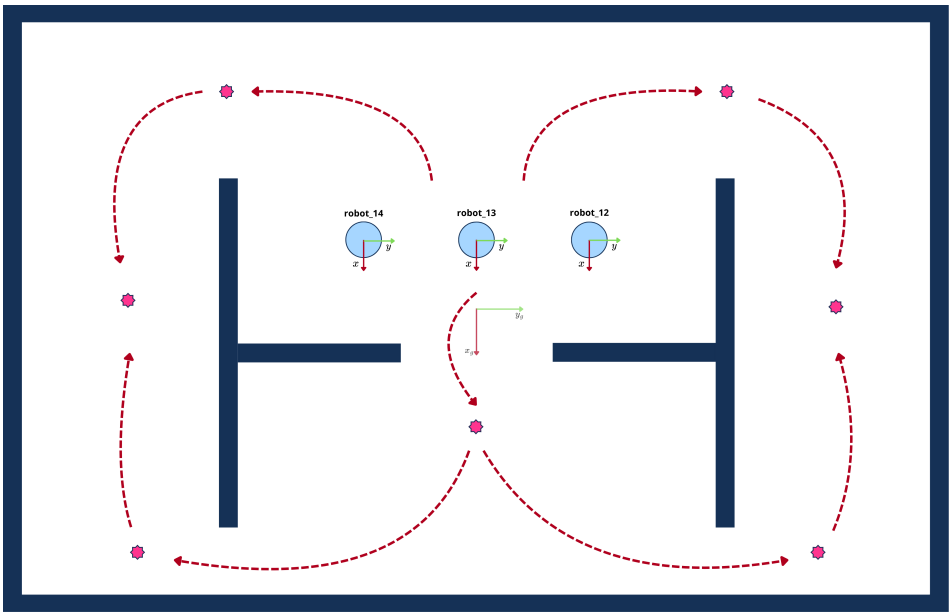


Figure 5.3: Experiment 2 environment

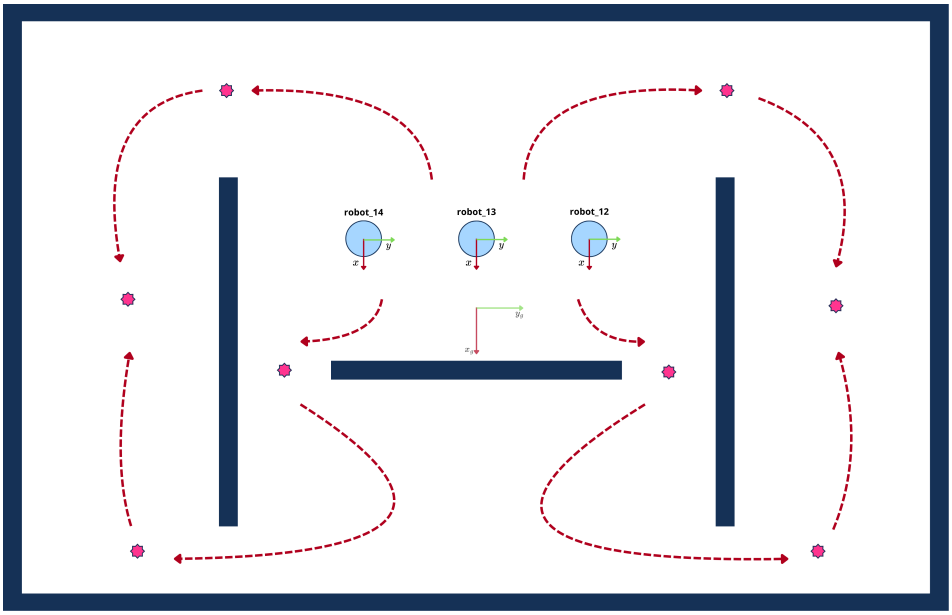


Figure 5.4: Experiment 3 environment

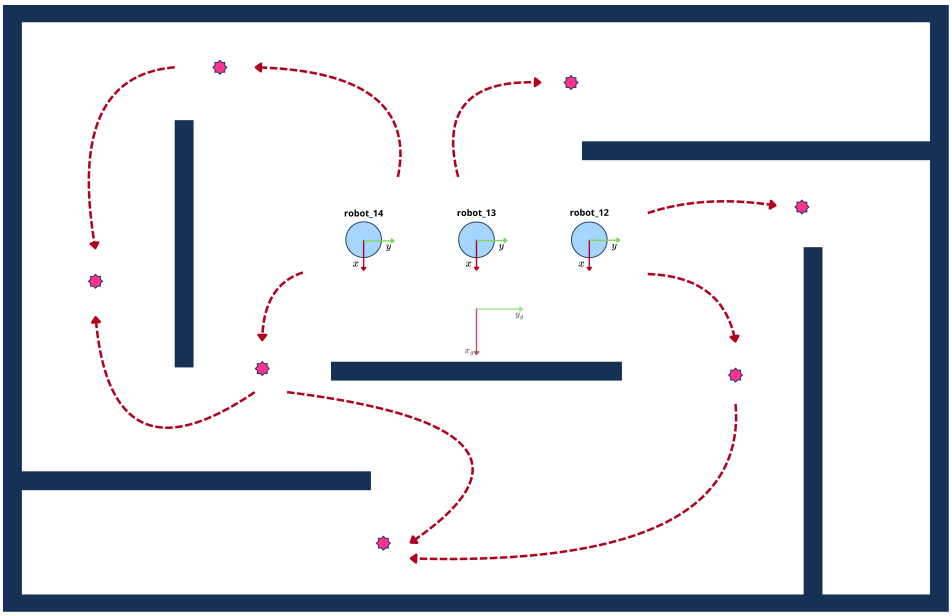


Figure 5.5: Experiment 4 environment

5.3 Ground Truth

To assess the performance of the proposed multi-robot exploration framework, reference ground truth data are employed for both localization and mapping evaluation, providing quantitative measurement of accuracy.

Localization ground truth For localization assessment, a Vicon motion capture system is used to provide high-precision ground truth trajectories. The system is composed of eleven infrared cameras strategically placed around the experimental area, capable of capturing the three-dimensional position and orientation of each robot with millimetric accuracy. Each TurtleBot3 is equipped with reflective markers that allow the Vicon system to track its pose (x, y, θ) in real time. This setup enables precise comparison between the estimated robot trajectories, obtained from the AC-SLAM system, and the ground truth trajectories measured by Vicon.

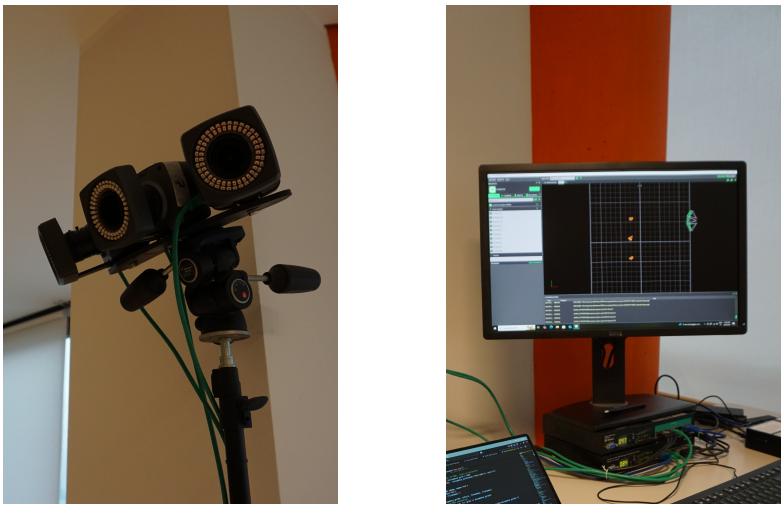


Figure 5.6: Vicon motion capture system

Mapping ground truth Since a precise ground truth map of the experimental environments is not available, the mapping accuracy is evaluated through a comparative approach. The occupancy grid maps produced by the proposed AC-SLAM framework are compared with reference maps generated using the well-established RTAB-Map (Real-Time Appearance-Based Mapping) system. RTAB-Map is a widely recognized SLAM method based on graph optimization and visual loop closure detection, providing high-quality maps suitable as a comparison reference. This approach allows assessing the fidelity of the reconstructed maps and the consistency of the merged multi-robot maps relative to a proven SLAM baseline.

5.4 Testing Modality

For each of the four experimental environments presented above, three sets of experiments are conducted to evaluate the performance and scalability of the proposed AC-SLAM framework when operating with different numbers of robots, specifically, one, two, and three robots. This configuration allows assessing how cooperative exploration improves mapping efficiency, coverage rate, and overall system robustness as the team size increases.

All experiments are performed under controlled indoor conditions within the PIC4SeR laboratory, ensuring repeatability and comparability across tests. Each experimental trial begins with all robots positioned at predefined starting locations, chosen to ensure similar initial conditions for every setup.

During each trial, the Vicon motion capture system continuously records the ground truth position and orientation of each robot at a high sampling rate, providing precise reference trajectories for localization accuracy assessment. Simultaneously, elaborated data are logged to enable offline analysis of performance metrics.

At the end of each exploration session, the occupancy grid maps generated by the system are compared against reference maps produced using the RTAB-Map SLAM framework in the same controlled environment.

5.5 Results and discussion

5.5.1 Experiment 1

In the following description of results, the sub-experiments and their respective robot configurations are indicated as follows:

- **Experiment 1.1:** (*robot_13*)
- **Experiment 1.2:** (*robot_12*, *robot_13*)
- **Experiment 1.3:** (*robot_12*, *robot_13*, *robot_14*)

Timing evaluation

Robot	**Exp 1.1**	**Exp 1.2**	**Exp 1.3**
robot_12	–	129.724 s	158.238 s
robot_13	165.404 s	133.751 s	105.732 s
robot_14	–	–	135.568 s
Total time	**165.404 s**	**133.751 s**	**158.238 s**
Gap time	–	4.027 s	52.505 s

Table 5.1: Experiment 1 - Timing evaluation

As shown in Figure 5.2, the robots starting positions are located within a closed area containing a single opening, thus initially yielding only one frontier. After the discovery of this first frontier, the system identified a maximum of two frontiers during exploration process.

Table 5.1 indicates that the best performance, in terms of exploration time, was achieved with the (*robot_12*, *robot_13*) configuration, with a remarkable low gap time between the end of the exploration of the first and the last robot. This setup proved optimal for efficiently covering both sides of the map once the robots exited the initial area. In contrast, the single-robot configuration (*robot_13*) was significantly slower, while the three-robot configuration (*robot_12*, *robot_13*, *robot_14*) was less efficient due to the redundancy of one robot initially following the same trajectory as another, causing interference and reducing overall flexibility in path planning.

Localization Evaluation

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_13	0.14	0.12	0.14	0.07
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_13	0.02	0.04	0.05	0.02

(a) Experiment 1.1

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.13	0.13	0.15	0.07
robot_13	0.04	0.11	0.12	0.06
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.03	0.02	0.02	0.01
robot_13	0.01	0.03	0.03	0.01

(b) Experiment 1.2

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.36	0.14	0.18	0.10
robot_13	0.03	0.08	0.10	0.06
robot_14	0.06	0.05	0.05	0.02
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.14	0.07	0.09	0.05
robot_13	0.08	0.05	0.05	0.02
robot_14	0.02	0.01	0.01	0.01

(c) Experiment 1.3

Table 5.2: Experiment 1 - Position and orientation accuracy metrics

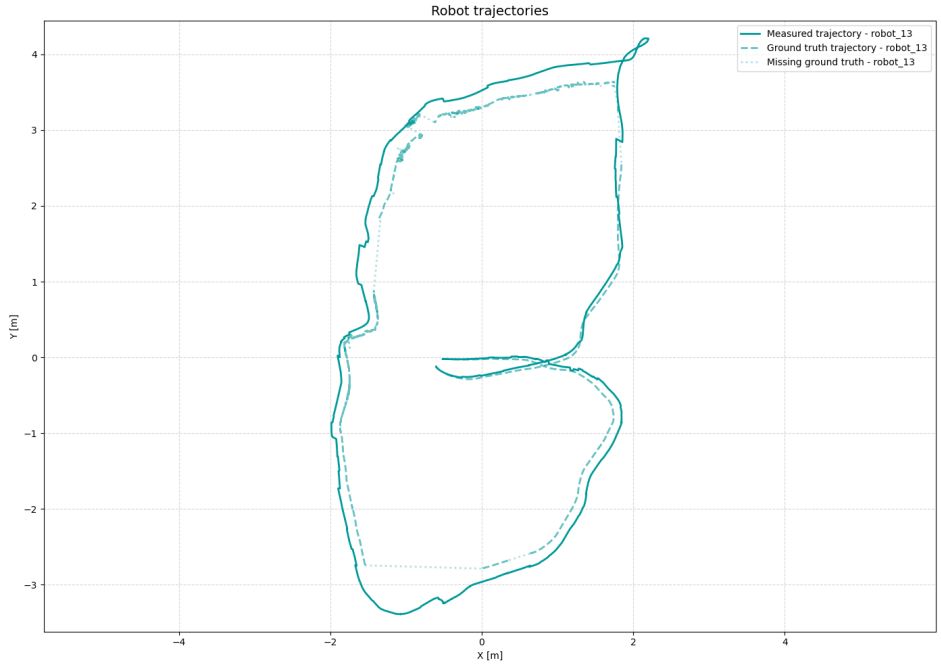


Figure 5.7: Experiment 1.1 - Localization trajectory with Vicon ground truth

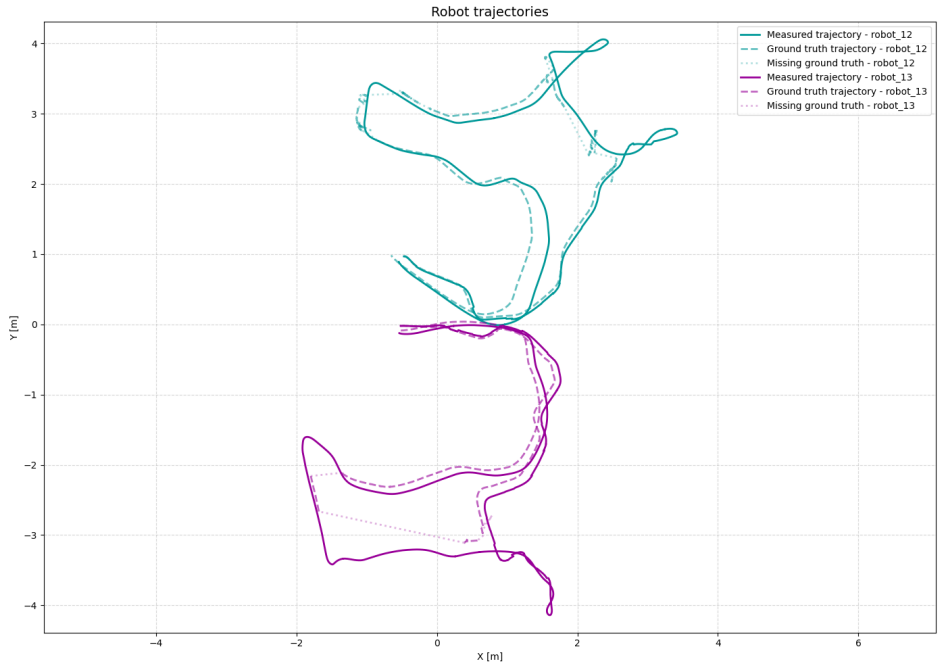


Figure 5.8: Experiment 1.2 - Localization trajectories with Vicon ground truth

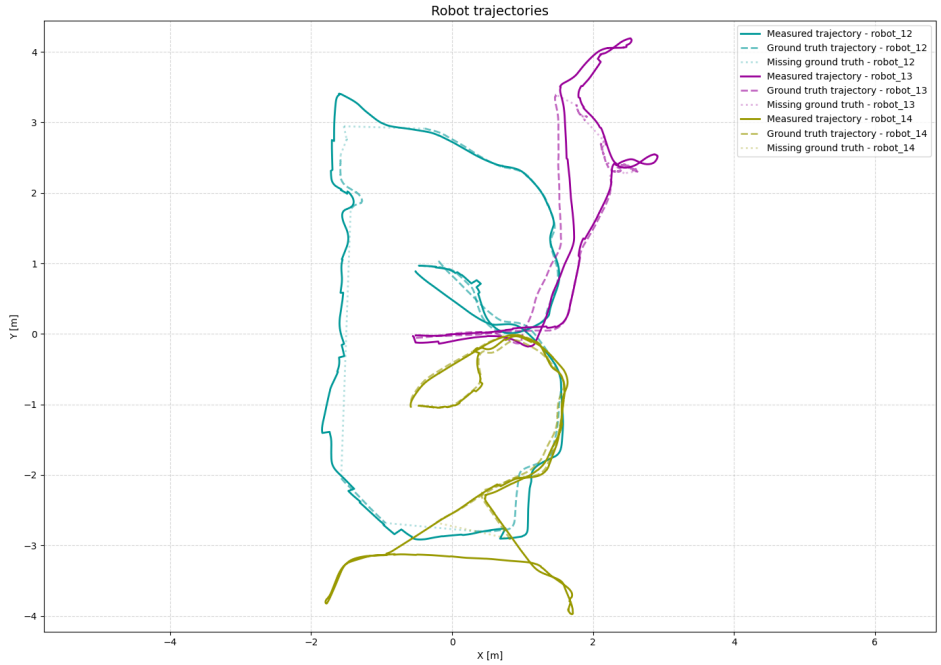


Figure 5.9: Experiment 1.3 - Localization trajectories with Vicon ground truth

Mapping evaluation

The mapping spatial resolution of the grid maps is **0.05 m/pixel**, spanning an area of **40 m** × **40 m** (**800** × **800** cells).

Robots	**Area [m²]**	**Rate [%]**
robot_13	64.79	100.00%

(a) Experiment 1.1

Robots	**Area [m²]**	**Rate [%]**
robot_12	59.29	80.75%
robot_13	52.00	80.30%
robot_12, robot_13	64.75	100.00%

(b) Experiment 1.2

Robots	**Area [m²]**	**Rate [%]**
robot_12	58.74	90.27%
robot_13	41.38	63.60%
robot_14	43.76	67.25%
robot_12, robot_13	62.79	96.50%
robot_12, robot_14	62.26	95.69%
robot_13, robot_14	57.63	88.57%
robot_12, robot_13, robot_14	65.07	100.00%

(c) Experiment 1.3

Table 5.3: Experiment 1 - Exploration coverage for each robots combination

Metric	**Exp 1.1**	**Exp 1.2**	**Exp 1.3**
Accuracy	93.7%	91.9%	93.5%
True Positive Rate (TPR)	42.7%	35.0%	39.4%
True Negative Rate (TNR)	97.4%	95.7%	97.5%
False Positive Rate (FPR)	2.6%	4.3%	2.6%
False Negative Rate (FNR)	57.3%	65.0%	60.6%

Table 5.4: Experiment 1 - Mapping accuracy confusion-based rate metrics

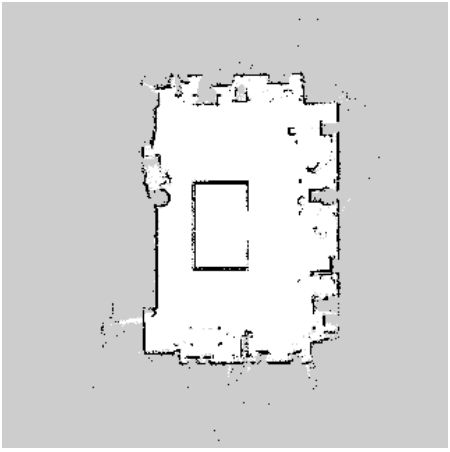
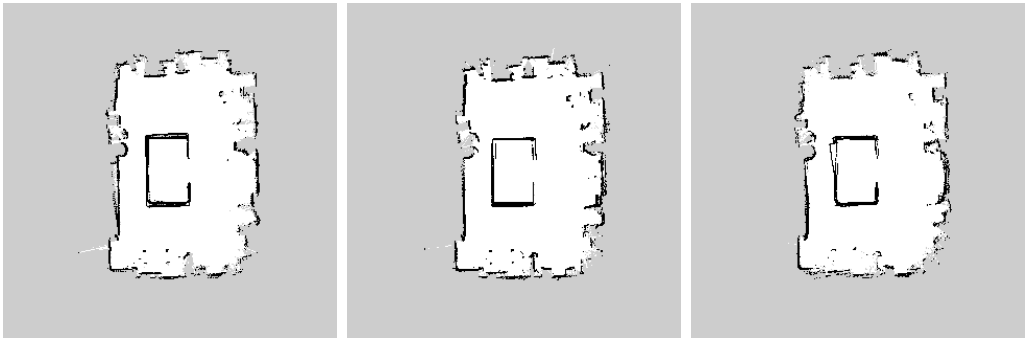
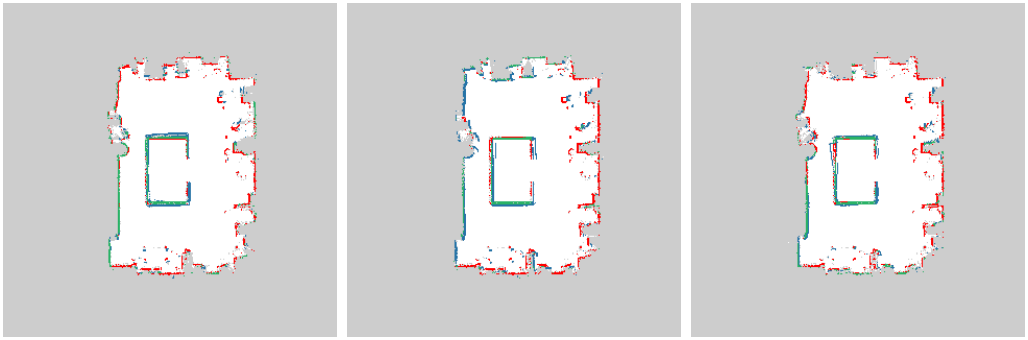


Figure 5.10: Experiment 1 - RTAB-Map ground truth



(a) Experiment 1.1 **(b)** Experiment 1.2 **(c)** Experiment 1.3

Figure 5.11: Experiment 1 - Mapping results



(a) Experiment 1.1 **(b)** Experiment 1.2 **(c)** Experiment 1.3

Figure 5.12: Experiment 1 - Confusion error metric (Green – TP, White – TN, Blue – FP, Red – FN)

Table 5.3 once again shows that the most efficient configuration is (*robot_12*, *robot_13*), achieving a balanced distribution of the explored area between the two robots. From the coverage values, it can also be observed that in the (*robot_12*, *robot_13*, *robot_14*) configuration, the pairs (*robot_12*, *robot_13*) and (*robot_12*, *robot_14*) together explored approximately 96% of the entire area during the task. This indicates that the addition of a supplementary robot is unnecessary, as two robots are sufficient to ensure efficient coverage.

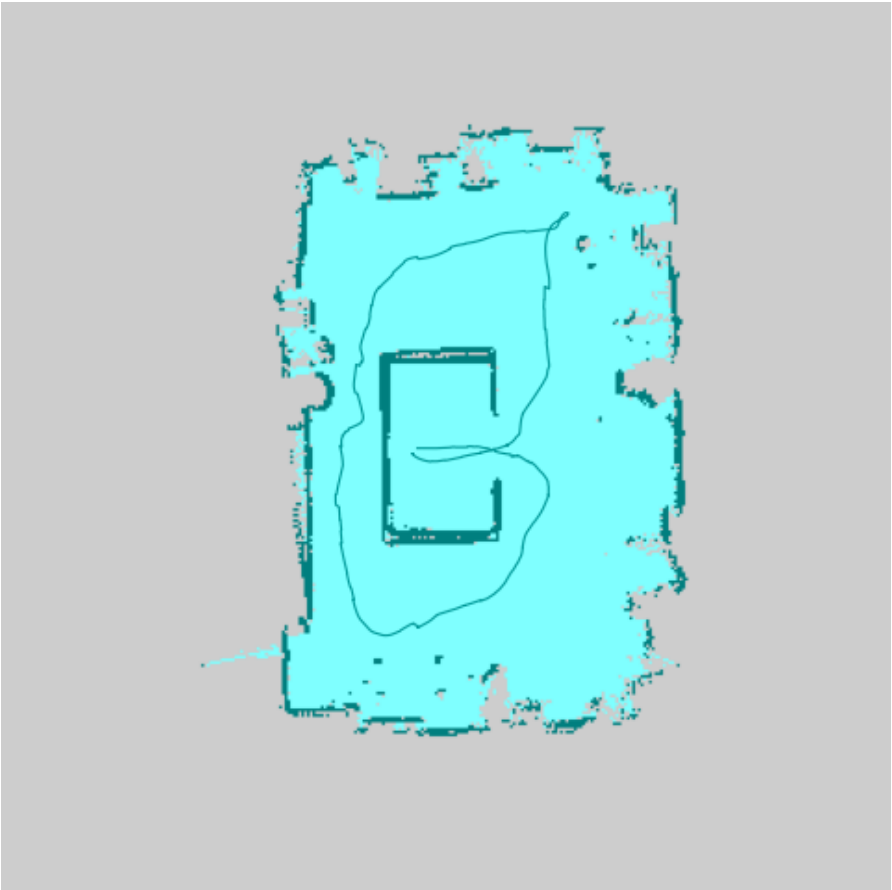
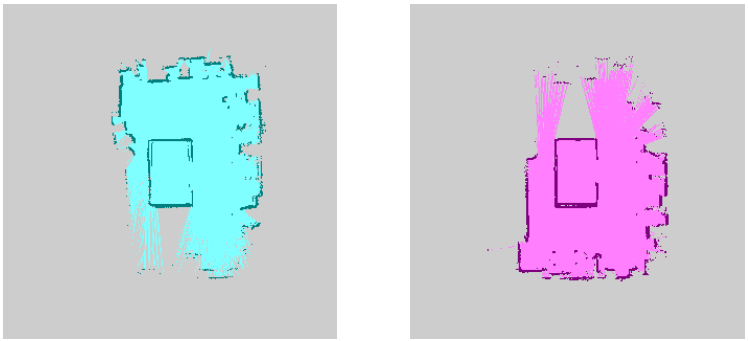


Figure 5.13: Experiment 1.1 - Robot exploration and trajectory



(a) *robot_12* map **(b)** *robot_13* map

Figure 5.14: Experiment 1.2 - Single robots exploration maps

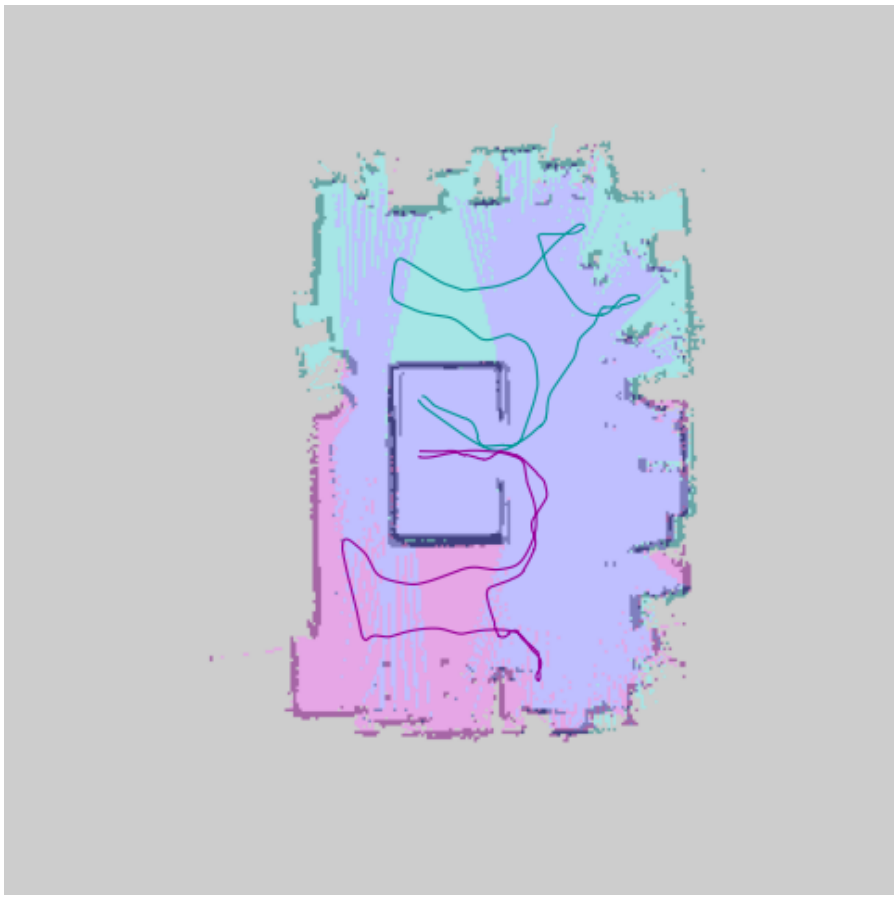
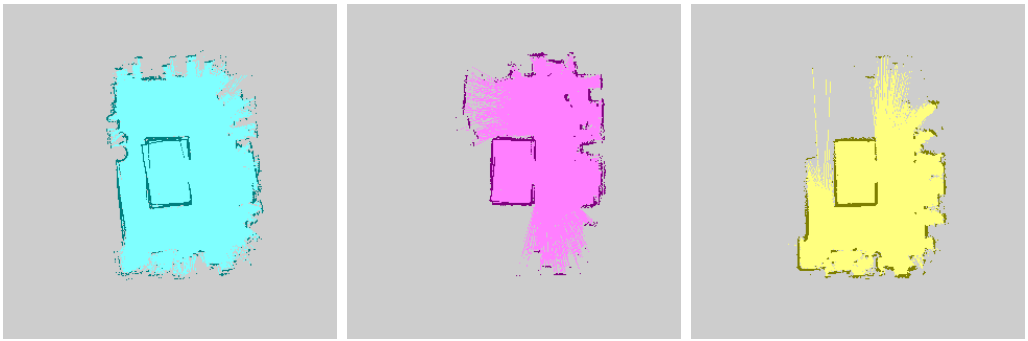


Figure 5.15: Experiment 1.2 - Robots exploration and trajectories



(a) *robot_12* map **(b)** *robot_13* map **(c)** *robot_14* map

Figure 5.16: Experiment 1.3 - Single robots exploration maps

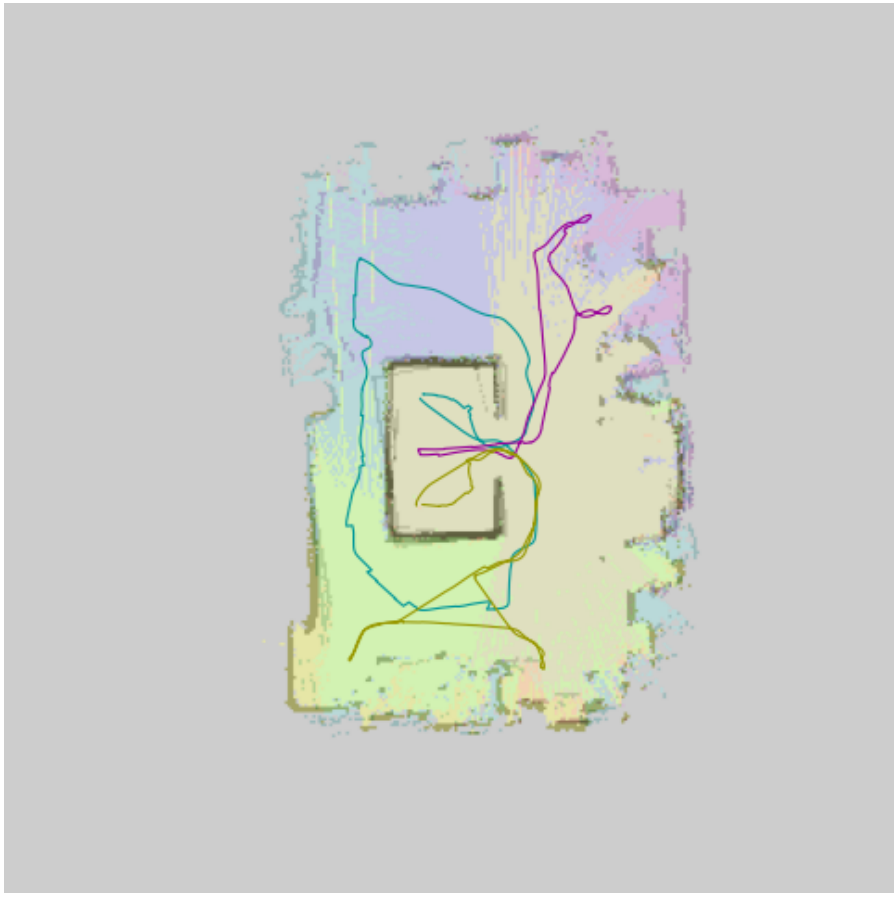


Figure 5.17: Experiment 1.3 - Robots exploration and trajectories

5.5.2 Experiment 2

In the following description of results, the sub-experiments and their respective robot configurations are indicated as follows:

- **Experiment 2.1:** (*robot_13*)
- **Experiment 2.2:** (*robot_12*, *robot_13*)
- **Experiment 2.3:** (*robot_12*, *robot_13*, *robot_14*)

Timing evaluation

Robot	**Exp 2.1**	**Exp 2.2**	**Exp 2.3**
robot_12	–	94.105 s	86.612 s
robot_13	141.704 s	94.709 s	58.303 s
robot_14	–	–	65.513 s
Total time	**141.704 s**	**94.709 s**	**86.612 s**
Gap time	–	0.603 s	28.308 s

Table 5.5: Experiment 2 - Timing evaluation

In this experiment, the environment shown in Figure 5.3 contains three initial frontiers to be discovered, making it particularly suitable for the (*robot_12*, *robot_13*, *robot_14*) configuration, where each robot targets the closest initial frontier. Thanks to the environment's symmetry, the (*robot_12*, *robot_13*) configuration exhibits the smallest exploration time gap, with both robots completing their tasks almost simultaneously.

Localization Evaluation

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_13	0.15	0.09	0.11	0.05
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_13	0.06	0.03	0.03	0.02

(a) Experiment 2.1

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.07	0.10	0.12	0.07
robot_13	0.11	0.09	0.11	0.05
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.03	0.03	0.04	0.02
robot_13	0.02	0.02	0.02	0.01

(b) Experiment 2.2

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.05	0.08	0.08	0.03
robot_13	0.14	0.11	0.12	0.05
robot_14	0.10	0.08	0.09	0.03
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.01	0.01	0.02	0.01
robot_13	0.03	0.02	0.03	0.01
robot_14	0.03	0.02	0.04	0.01

(c) Experiment 2.3

Table 5.6: Experiment 2 - Position and orientation accuracy metrics

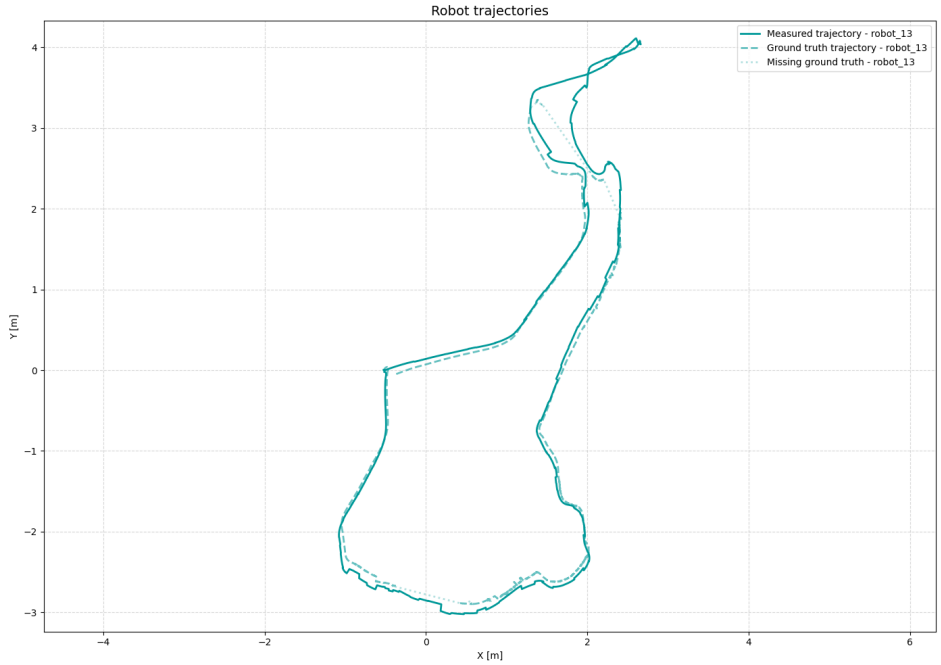


Figure 5.18: Experiment 2.1 - Localization trajectory with Vicon ground truth

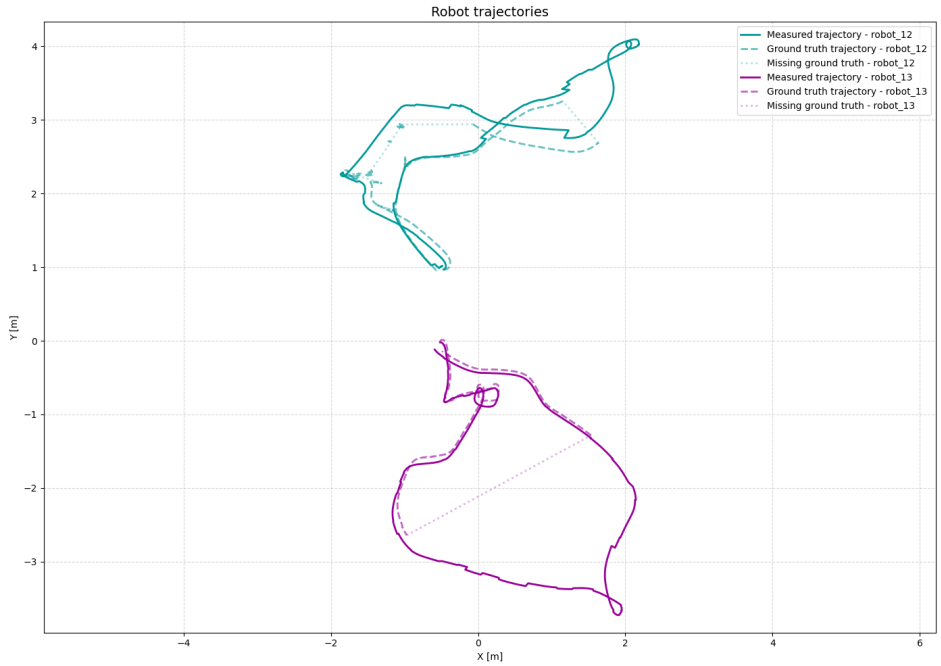


Figure 5.19: Experiment 2.2 - Localization trajectories with Vicon ground truth

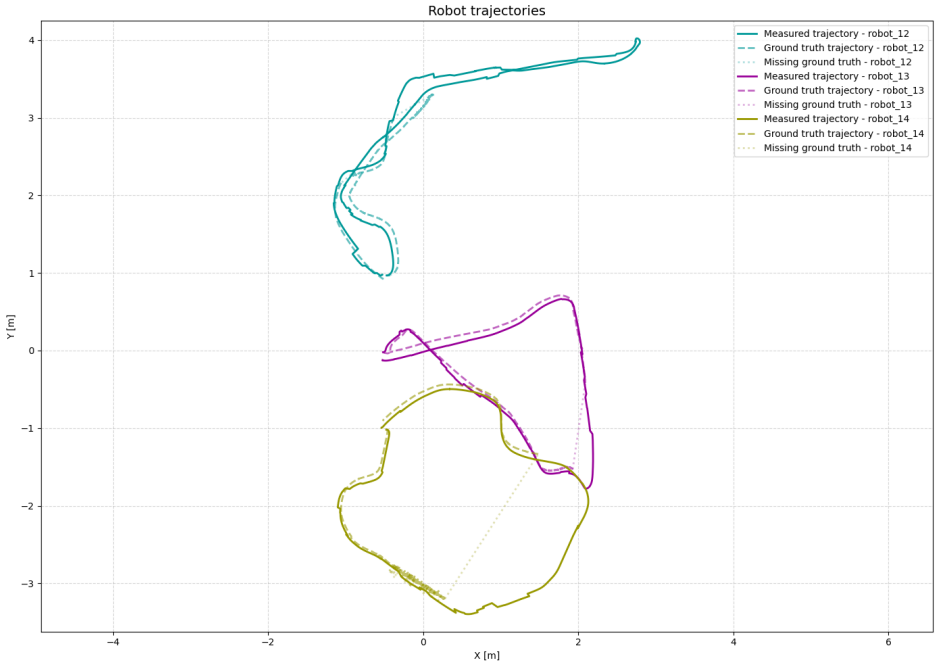


Figure 5.20: Experiment 2.3 - Localization trajectories with Vicon ground truth

Mapping evaluation

The mapping spatial resolution of the grid maps is **0.05 m/pixel**, spanning an area of **40 m** × **40 m** (**800** × **800** cells).

Robots	**Area [m²]**	**Rate [%]**
robot_13	63.71	100.00%

(a) Experiment 2.1

Robots	**Area [m²]**	**Rate [%]**
robot_12	51.63	80.72%
robot_13	51.14	79.95%
robot_12, robot_13	63.96	100.00%

(b) Experiment 2.2

Robots	**Area [m²]**	**Rate [%]**
robot_12	53.02	82.50%
robot_13	47.56	74.01%
robot_14	49.06	76.33%
robot_12, robot_13	60.42	94.01%
robot_12, robot_14	62.95	97.95%
robot_13, robot_14	53.54	83.31%
robot_12, robot_13, robot_14	64.70	100.00%

(c) Experiment 2.3

Table 5.7: Experiment 2 - Exploration coverage for each robots combination

Metric	**Exp 2.1**	**Exp 2.2**	**Exp 2.3**
Accuracy	93.9%	94.4%	94.7%
True Positive Rate (TPR)	40.5%	47.9%	53.7%
True Negative Rate (TNR)	97.3%	97.5%	97.4%
False Positive Rate (FPR)	2.7%	2.5%	2.5%
False Negative Rate (FNR)	59.5%	52.1%	46.3%

Table 5.8: Experiment 2 - Mapping accuracy confusion-based rate metrics

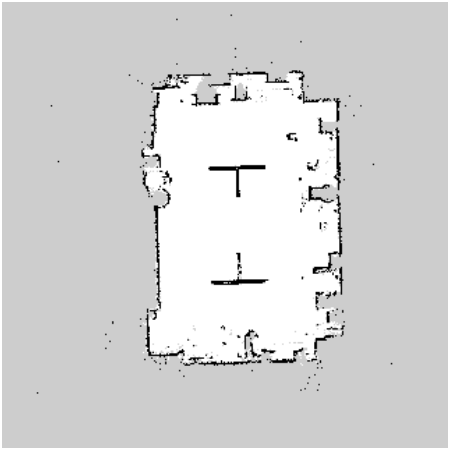
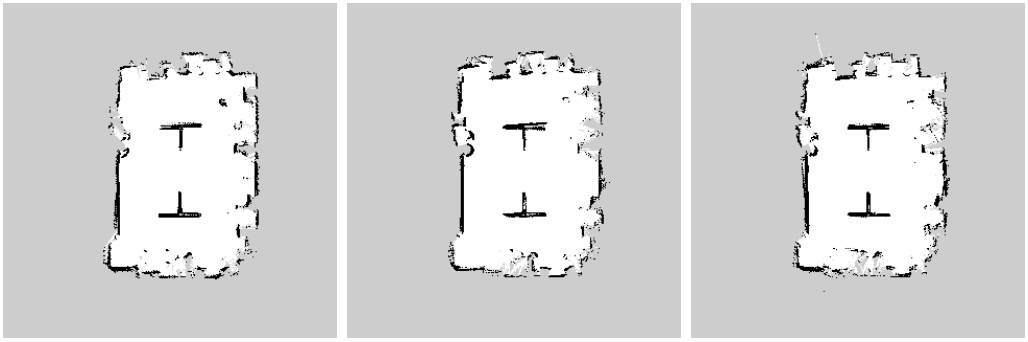
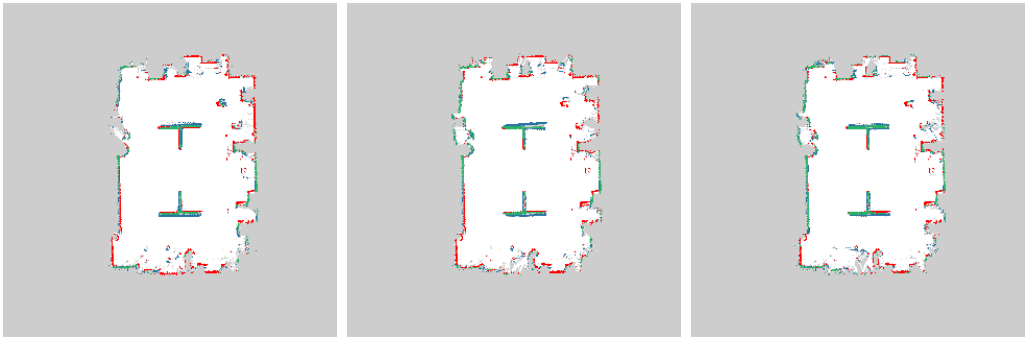


Figure 5.21: Experiment 2 - RTAB-Map ground truth



(a) Experiment 2.1 **(b)** Experiment 2.2 **(c)** Experiment 2.3

Figure 5.22: Experiment 2 - Mapping results



(a) Experiment 2.1 **(b)** Experiment 2.2 **(c)** Experiment 2.3

Figure 5.23: Experiment 2 - Confusion error metric (Green – TP, White – TN, Blue – FP, Red – FN)

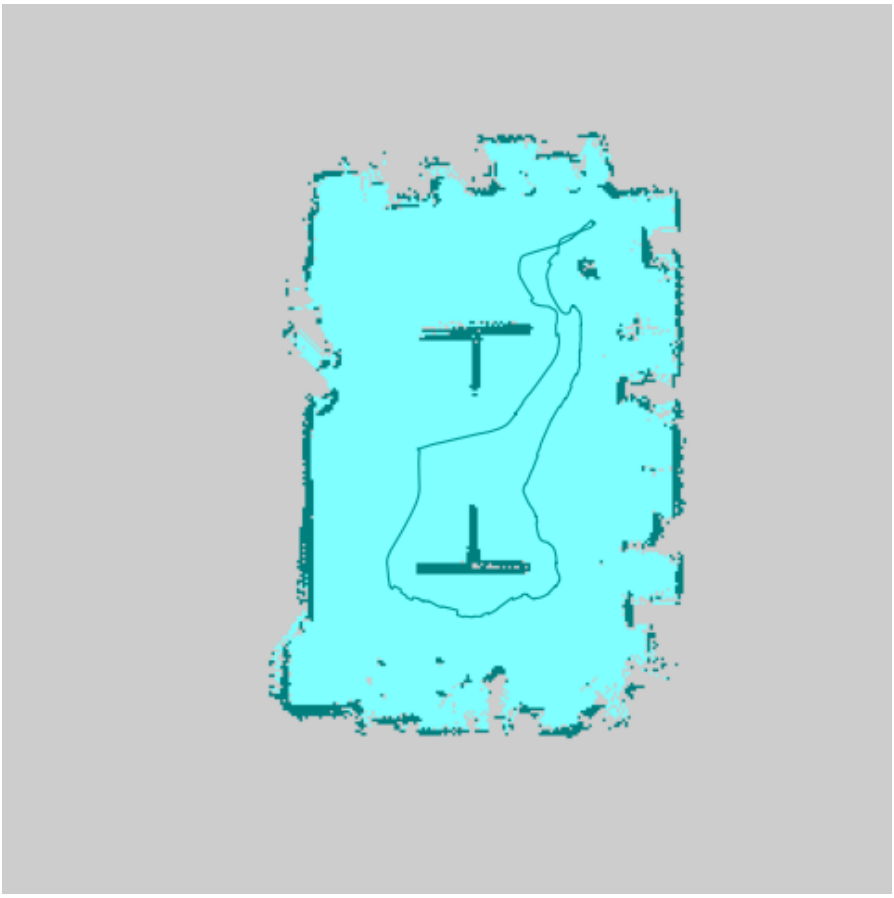
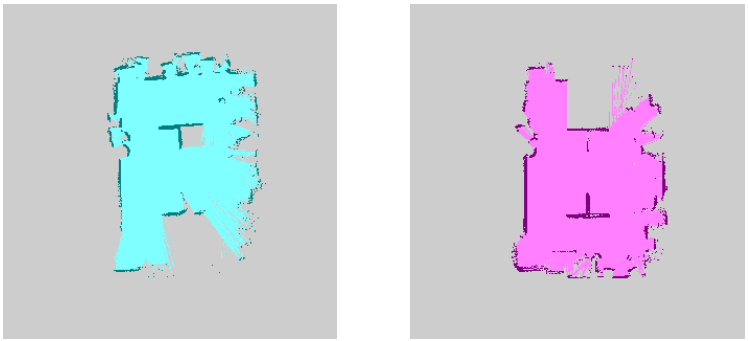


Figure 5.24: Experiment 2.1 - Robot exploration and trajectory



(a) *robot_12* map **(b)** *robot_13* map

Figure 5.25: Experiment 2.2 - Single robots exploration maps

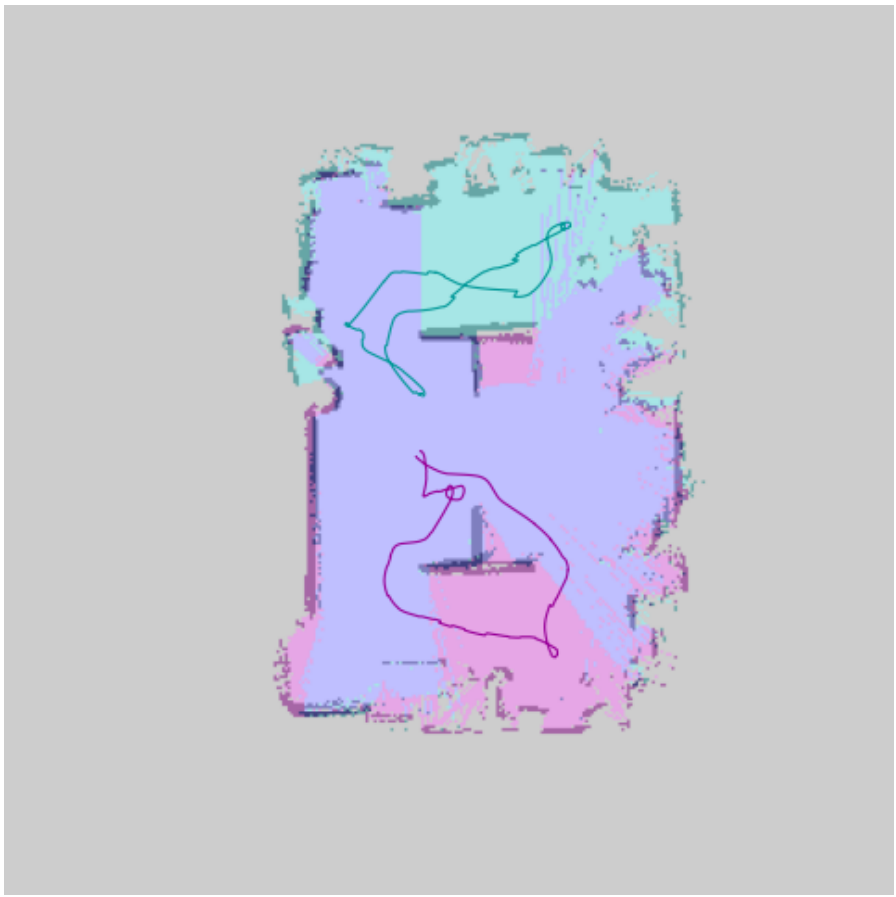
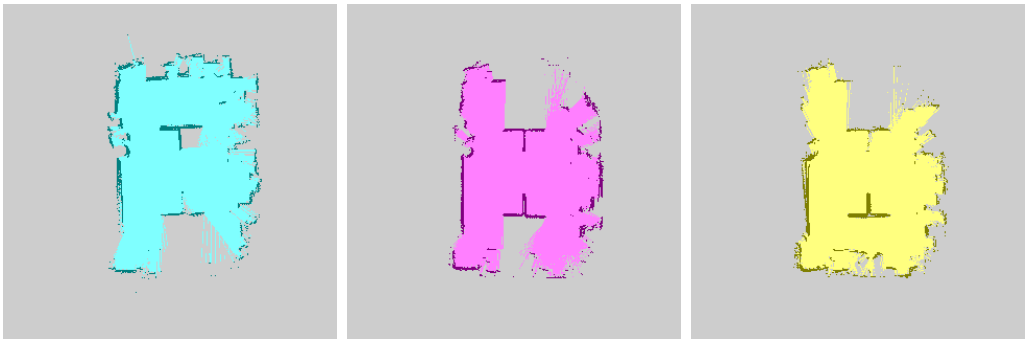


Figure 5.26: Experiment 2.2 - Robots exploration and trajectories



(a) *robot_12* map **(b)** *robot_13* map **(c)** *robot_14* map

Figure 5.27: Experiment 2.3 - Single robots exploration maps

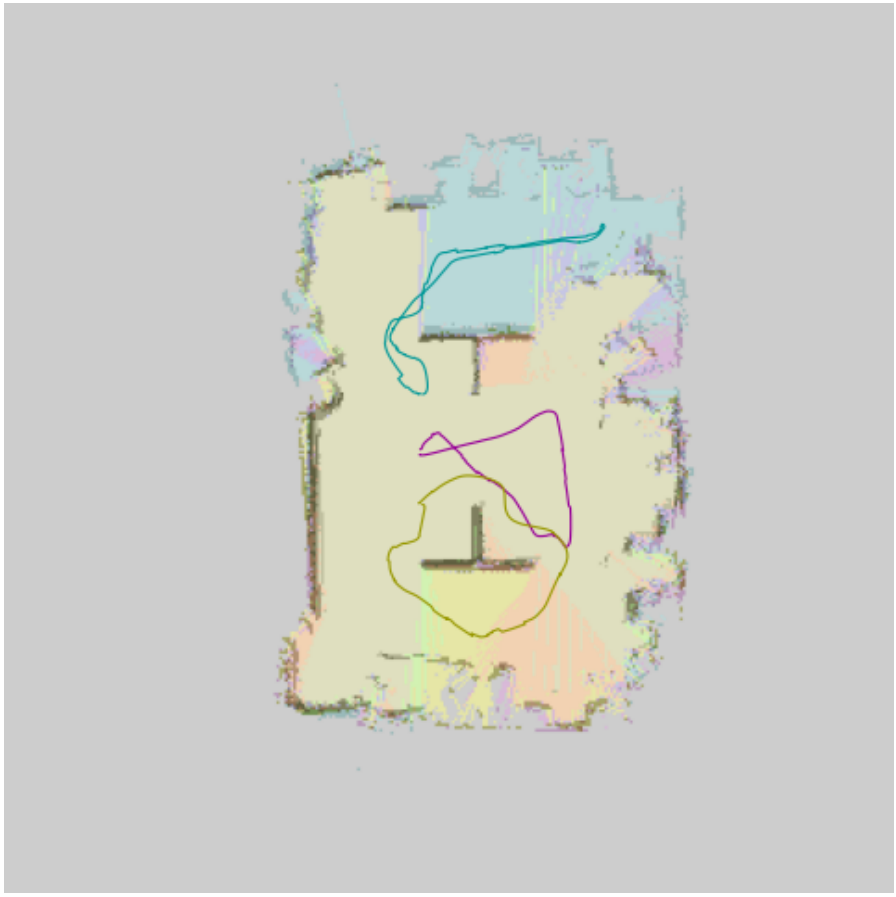


Figure 5.28: Experiment 2.3 - Robots exploration and trajectories

5.5.3 Experiment 3

In the following description of results, the sub-experiments and their respective robot configurations are indicated as follows:

- **Experiment 3.1:** (*robot_13*)
- **Experiment 3.2:** (*robot_12*, *robot_13*)
- **Experiment 3.3:** (*robot_12*, *robot_13*, *robot_14*)

Timing evaluation

Robot	**Exp 3.1**	**Exp 3.2**	**Exp 3.3**
robot_12	–	66.002 s	86.699 s
robot_13	114.411 s	68.999 s	70.359 s
robot_14	–	–	66.339 s
Total time	**114.411 s**	**68.999 s**	**86.699 s**
Gap time	–	2.997 s	20.360 s

Table 5.9: Experiment 3 - Timing evaluation

With four initial frontiers to be explored, the environment shown in Figure 5.4 challenges the robots to select frontiers coherently in order to maximize exploration efficiency. Interestingly, despite the increased number of frontiers with respect to the last experiment, the optimal performance is achieved with the (*robot_12*, *robot_13*) configuration, which demonstrates a highly balanced exploration pattern. Each robot covers a distinct zone of the map, navigating around the obstacle on its side and efficiently exploring the entire area. In this case as well, the time gap between the two robots is remarkably small.

Localization Evaluation

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_13	0.09	0.12	0.13	0.06
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_13	0.02	0.03	0.03	0.02

(a) Experiment 3.1

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.13	0.08	0.09	0.04
robot_13	0.08	0.11	0.12	0.05
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.06	0.03	0.04	0.02
robot_13	0.03	0.03	0.03	0.02

(b) Experiment 3.2

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.12	0.09	0.10	0.04
robot_13	0.11	0.09	0.11	0.05
robot_14	0.12	0.09	0.10	0.04
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.06	0.02	0.03	0.02
robot_13	0.05	0.04	0.05	0.02
robot_14	0.05	0.03	0.04	0.02

(c) Experiment 3.3

Table 5.10: Experiment 3 - Position and orientation accuracy metrics

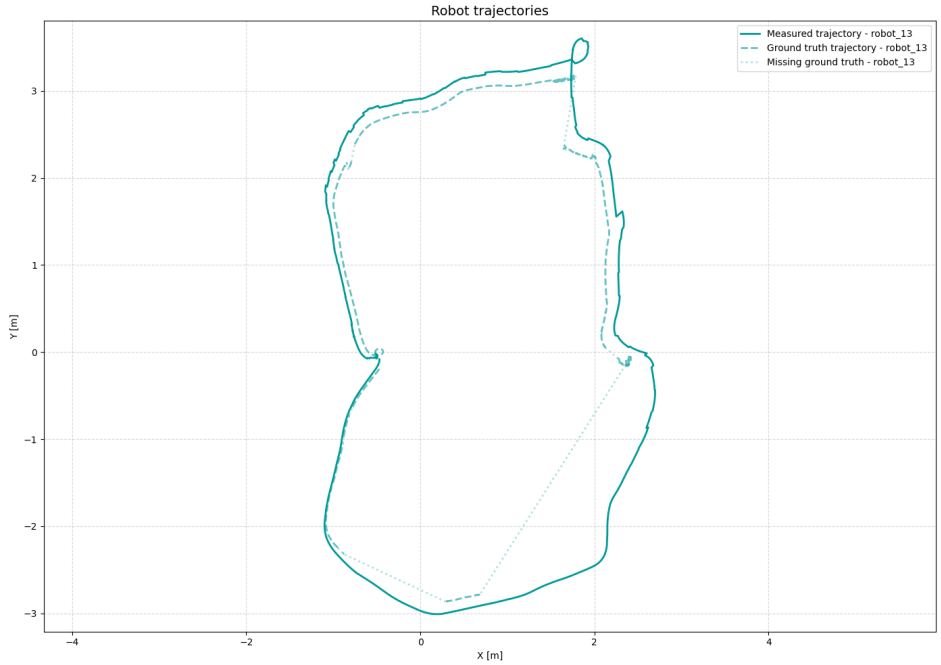


Figure 5.29: Experiment 3.1 - Localization trajectory with Vicon ground truth

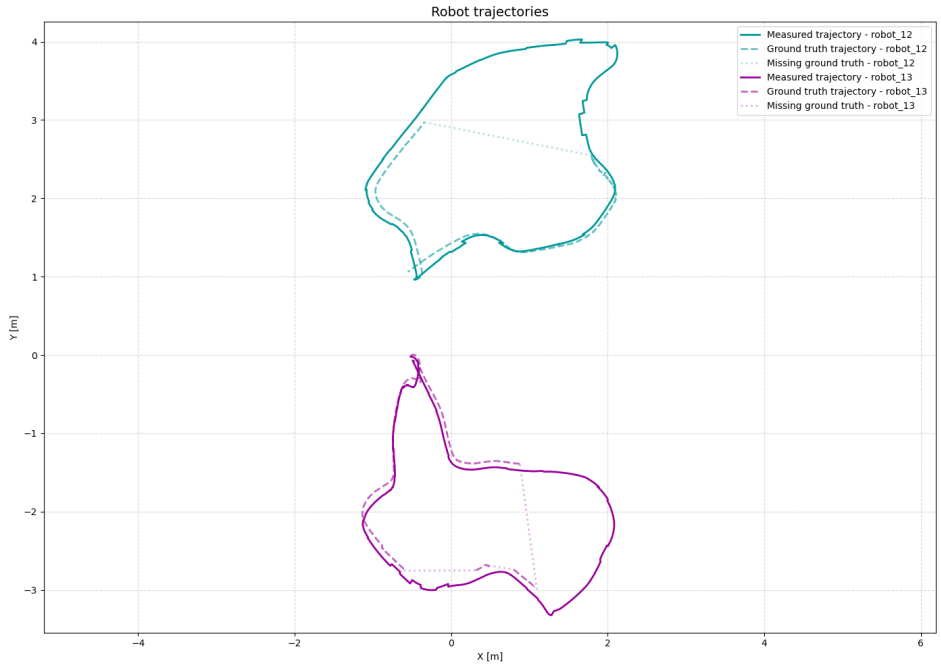


Figure 5.30: Experiment 3.2 - Localization trajectories with Vicon ground truth

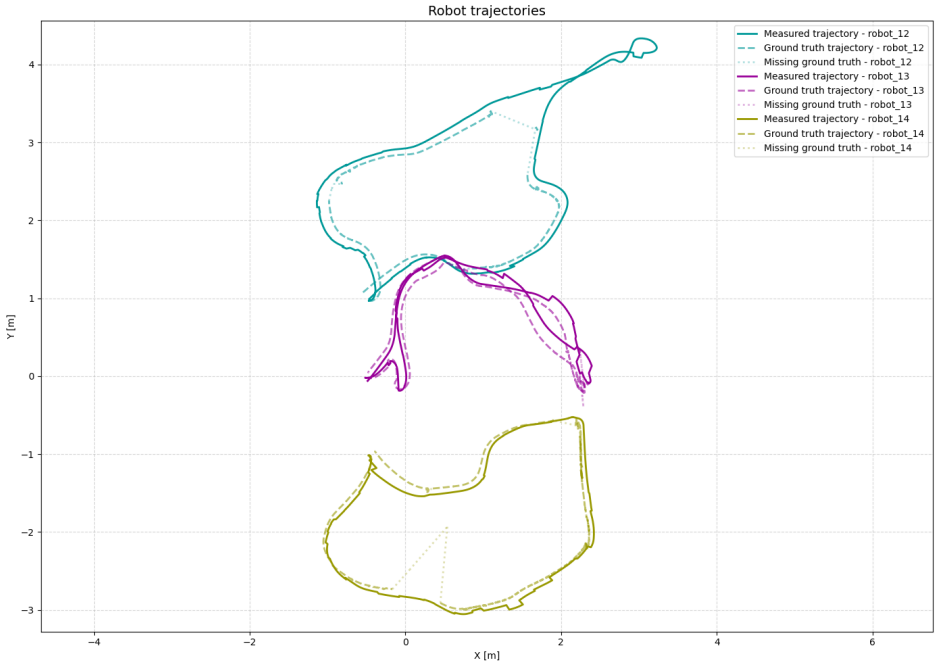


Figure 5.31: Experiment 3.3 - Localization trajectories with Vicon ground truth

Mapping evaluation

The mapping spatial resolution of the grid maps is **0.05 m/pixel**, spanning an area of **40 m** × **40 m** (**800** × **800** cells).

Robots	**Area [m²]**	**Rate [%]**
robot_13	64.46	100.00%

(a) Experiment 3.1

Robots	**Area [m²]**	**Rate [%]**
robot_12	53.37	82.73%
robot_13	50.19	77.80%
robot_12, robot_13	64.90	100.00%

(b) Experiment 3.2

Robots	**Area [m²]**	**Rate [%]**
robot_12	50.67	79.43%
robot_13	46.79	73.35%
robot_14	49.42	77.47%
robot_12, robot_13	57.02	89.38%
robot_12, robot_14	63.11	98.93%
robot_13, robot_14	55.82	87.50%
robot_12, robot_13, robot_14	64.28	100.00%

(c) Experiment 3.3

Table 5.11: Experiment 3 - Exploration coverage for each robots combination

Metric	**Exp 3.1**	**Exp 3.2**	**Exp 3.3**
Accuracy	93.2%	93.6%	93.4%
True Positive Rate (TPR)	42.5%	40.6%	43.8%
True Negative Rate (TNR)	96.5%	97.3%	96.6%
False Positive Rate (FPR)	3.5%	2.7%	3.3%
False Negative Rate (FNR)	57.5%	59.4%	56.2%

Table 5.12: Experiment 3 - Mapping accuracy confusion-based rate metrics

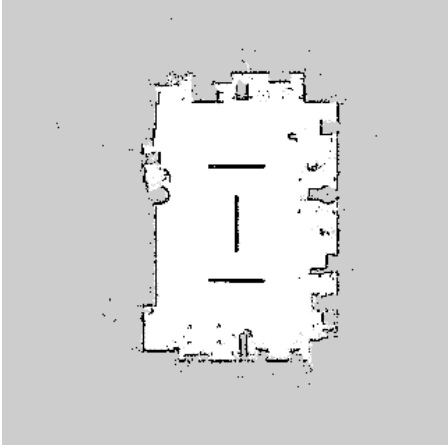
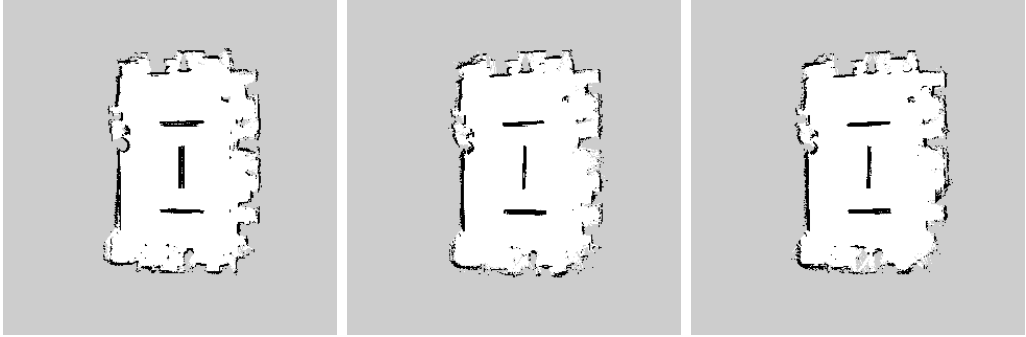
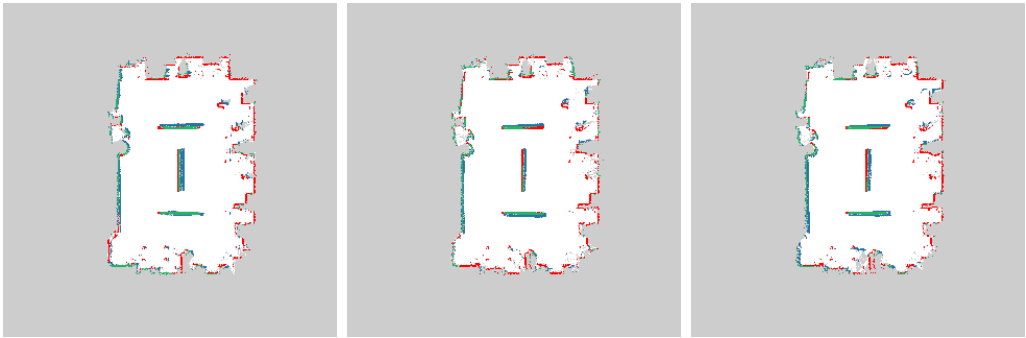


Figure 5.32: Experiment 3 - RTAB-Map ground truth



(a) Experiment 3.1 **(b)** Experiment 3.2 **(c)** Experiment 3.3

Figure 5.33: Experiment 3 - Mapping results



(a) Experiment 3.1 **(b)** Experiment 3.2 **(c)** Experiment 3.3

Figure 5.34: Experiment 3 - Confusion error metric (Green – TP, White – TN, Blue – FP, Red – FN)

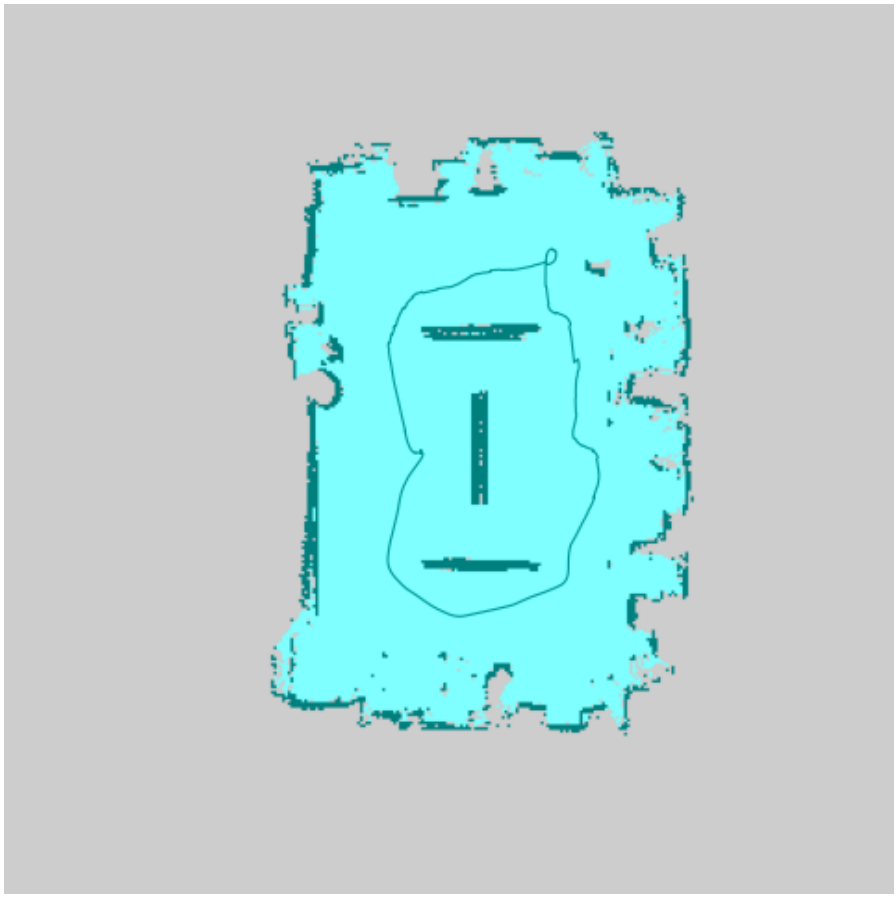
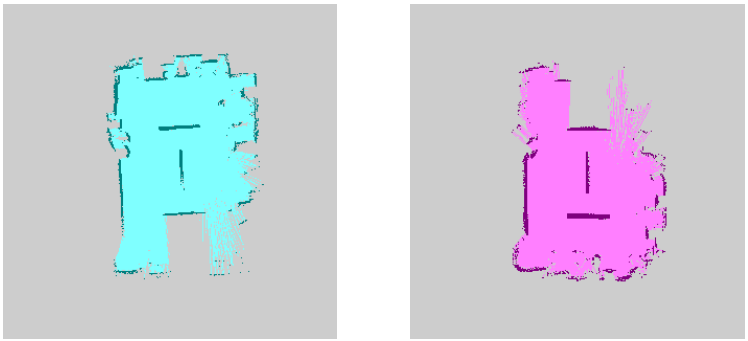


Figure 5.35: Experiment 3.1 - Robot exploration and trajectory



(a) *robot_12* map **(b)** *robot_13* map

Figure 5.36: Experiment 3.2 - Single robots exploration maps

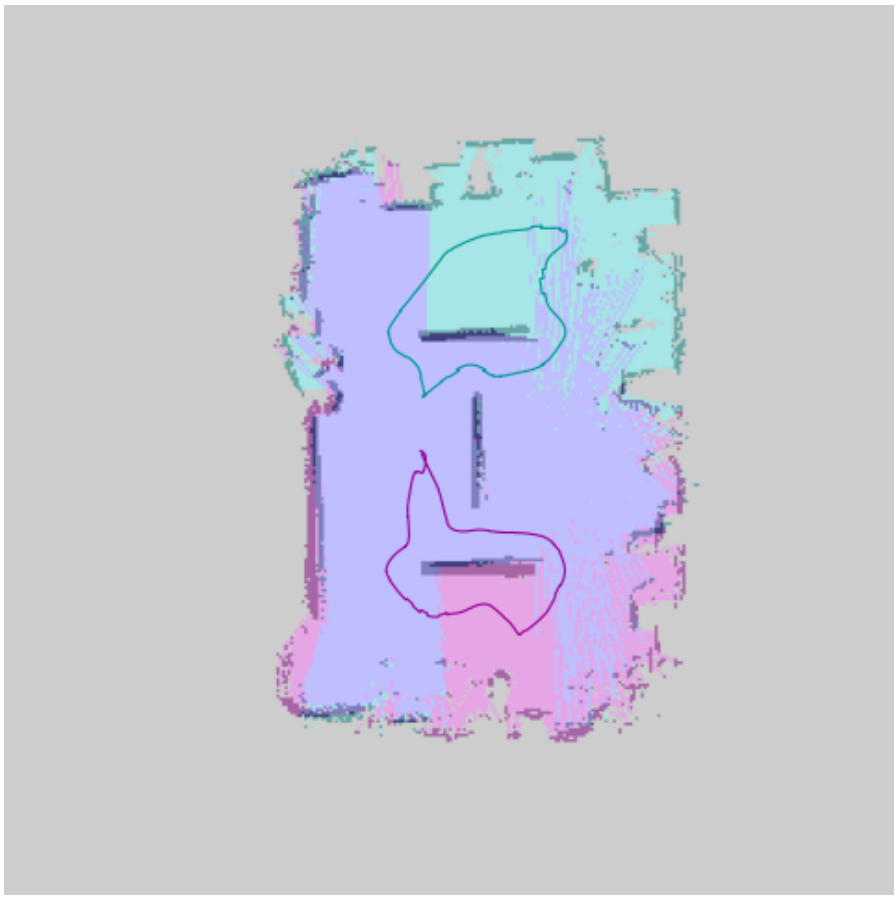
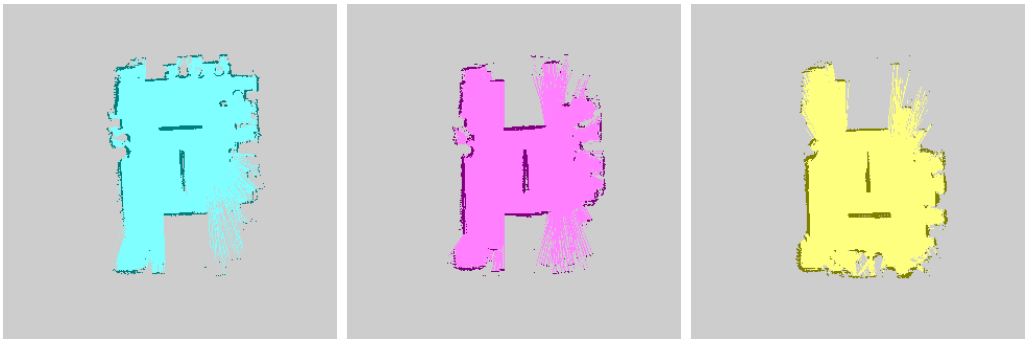


Figure 5.37: Experiment 3.2 - Robots exploration and trajectories



(a) *robot_12* map **(b)** *robot_13* map **(c)** *robot_14* map

Figure 5.38: Experiment 3.3 - Single robots exploration maps

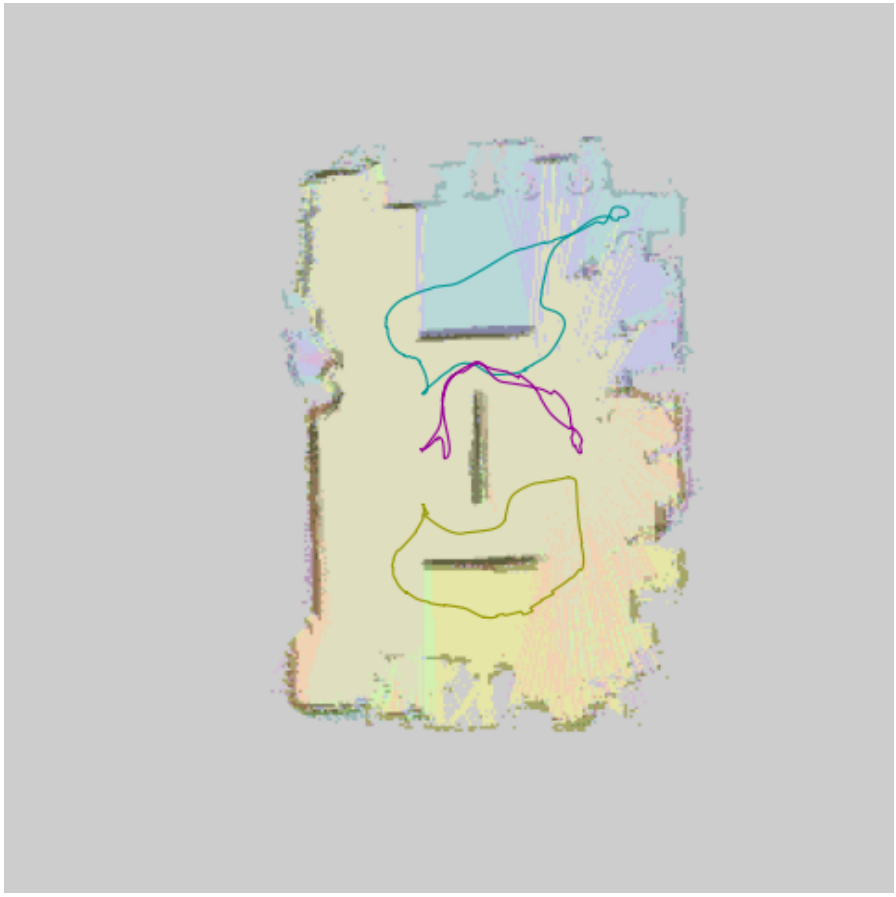


Figure 5.39: Experiment 3.3 - Robots exploration and trajectories

5.5.4 Experiment 4

In the following description of results, the sub-experiments and their respective robot configurations are indicated as follows:

- **Experiment 4.1:** (*robot_13*)
- **Experiment 4.2:** (*robot_12*, *robot_13*)
- **Experiment 4.3:** (*robot_12*, *robot_13*, *robot_14*)

Timing evaluation

Robot	**Exp 4.1**	**Exp 4.2**	**Exp 4.3**
robot_12	–	102.321 s	106.500 s
robot_13	220.707 s	113.801 s	99.906 s
robot_14	–	–	96.816 s
Total time	**220.707 s**	**113.801 s**	**106.500 s**
Gap time	–	11.480 s	9.684 s

Table 5.13: Experiment 4 - Timing evaluation

The environment shown in Figure 5.5 is designed to evaluate the behavior of the multi-robot system in a more complex scenario, characterized by a labyrinth-style map, numerous frontiers to be discovered, and several blind spots. This configuration presents a significant challenge for coordination and task allocation among the robots, as it requires efficient frontier selection and path planning to avoid redundant exploration and ensure full area coverage, while avoiding collisions between robots in strict areas.

In this experiment, the (*robot_12*, *robot_13*, *robot_14*) configuration proved to be noticeably more efficient, exhibiting a well-balanced distribution of the robots across distinct regions of the environment.

Localization Evaluation

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_13	0.03	0.11	0.13	0.06
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_13	0.01	0.03	0.04	0.02

(a) Experiment 4.1

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.01	0.07	0.09	0.06
robot_13	0.17	0.13	0.14	0.06
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.06	0.04	0.05	0.02
robot_13	0.09	0.09	0.09	0.03

(b) Experiment 4.2

Robot	**Final dist. error [m]**	**MAE [m]**	**RMSE [m]**	**STD [m]**
robot_12	0.16	0.09	0.11	0.05
robot_13	0.05	0.10	0.11	0.05
robot_14	0.01	0.07	0.09	0.04
Robot	**Final yaw error [rad]**	**MAE [rad]**	**RMSE [rad]**	**STD [rad]**
robot_12	0.07	0.04	0.04	0.02
robot_13	0.05	0.05	0.05	0.01
robot_14	0.04	0.03	0.03	0.01

(c) Experiment 4.3

Table 5.14: Experiment 4 - Position and orientation accuracy metrics

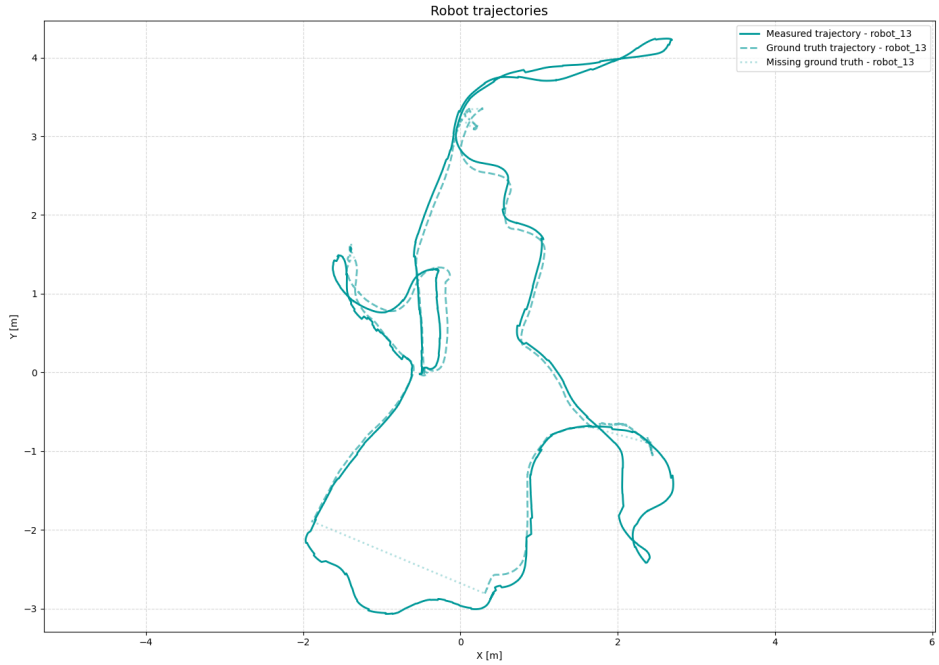


Figure 5.40: Experiment 4.1 - Localization trajectory with Vicon ground truth

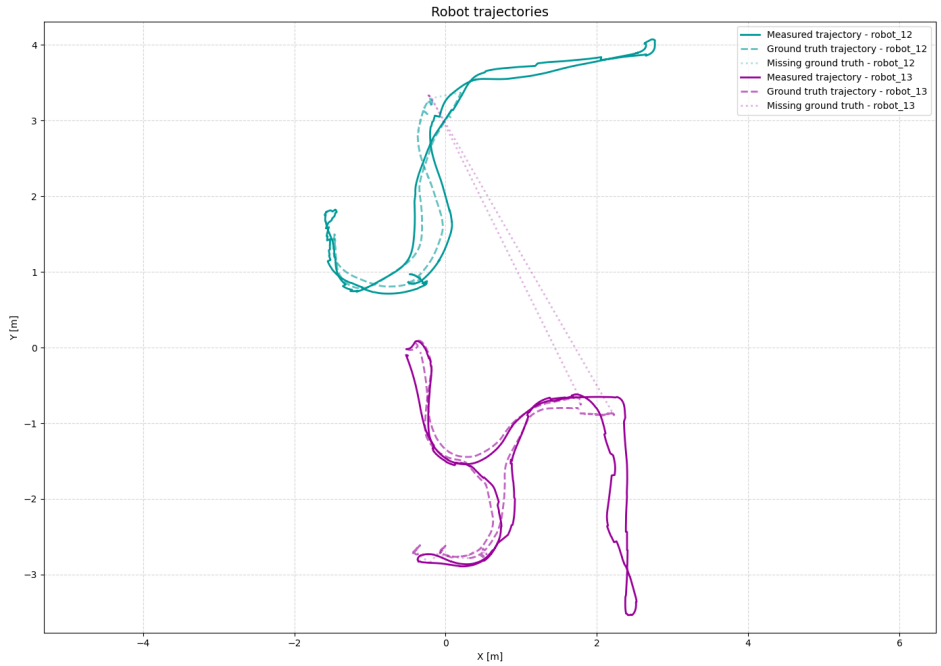


Figure 5.41: Experiment 4.2 - Localization trajectories with Vicon ground truth

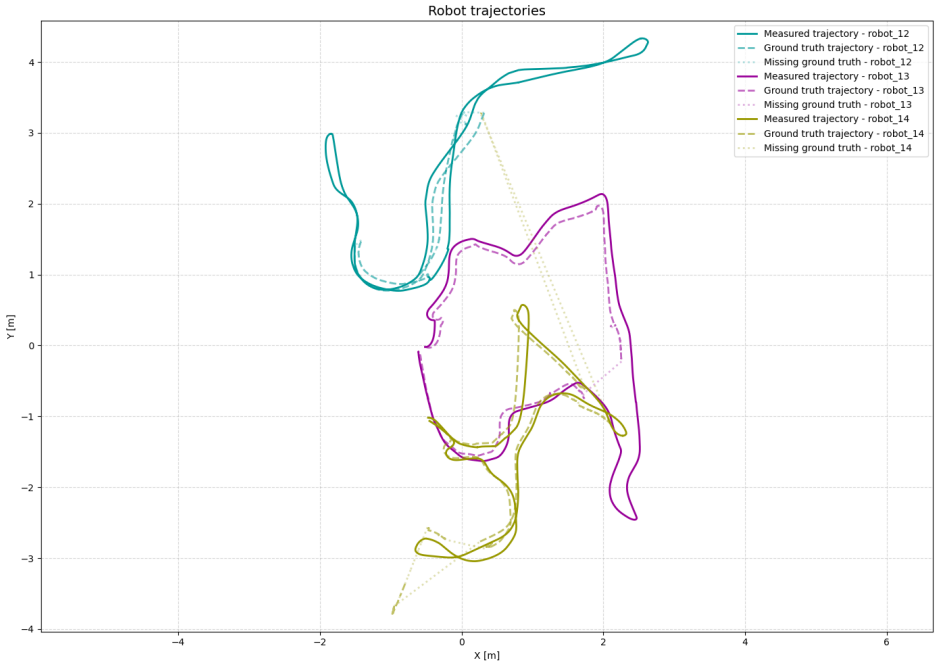


Figure 5.42: Experiment 4.3 - Localization trajectories with Vicon ground truth

Mapping evaluation

The mapping spatial resolution of the grid maps is **0.05 m/pixel**, spanning an area of **40 m** × **40 m** (**800** × **800** cells).

Robots	**Area [m²]**	**Rate [%]**
robot_13	64.65	100.00%

(a) Experiment 4.1

Robots	**Area [m²]**	**Rate [%]**
robot_12	47.59	73.21%
robot_13	52.89	81.37%
robot_12, robot_13	65.00	100.00%

(b) Experiment 4.2

Robots	**Area [m²]**	**Rate [%]**
robot_12	40.88	65.29%
robot_13	46.97	75.02%
robot_14	48.67	77.74%
robot_12, robot_13	57.75	92.24%
robot_12, robot_14	60.01	95.84%
robot_13, robot_14	54.74	87.42%
robot_12, robot_13, robot_14	63.16	100.00%

(c) Experiment 4.3

Table 5.15: Experiment 4 - Exploration coverage for each robots combination

Metric	**Exp 4.1**	**Exp 4.2**	**Exp 4.3**
Accuracy	92.0%	91.9%	92.5%
True Positive Rate (TPR)	32.1%	33.3%	44.8%
True Negative Rate (TNR)	96.8%	96.7%	96.2%
False Positive Rate (FPR)	3.2%	3.2%	3.8%
False Negative Rate (FNR)	67.9%	66.7%	55.2%

Table 5.16: Experiment 4 - Mapping accuracy confusion-based rate metrics

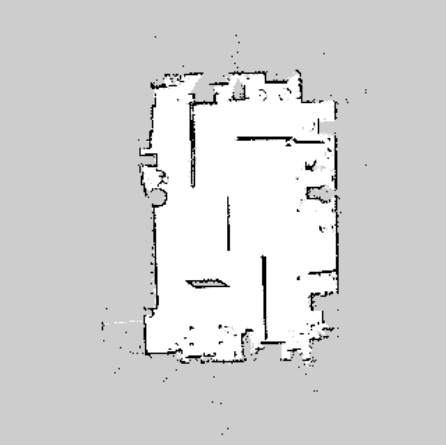
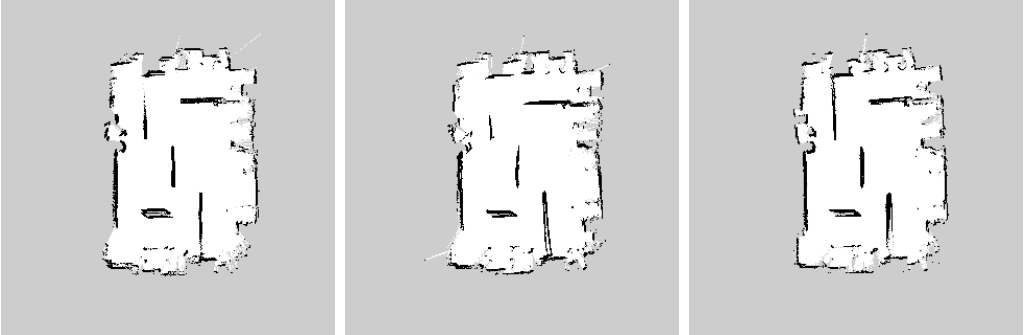
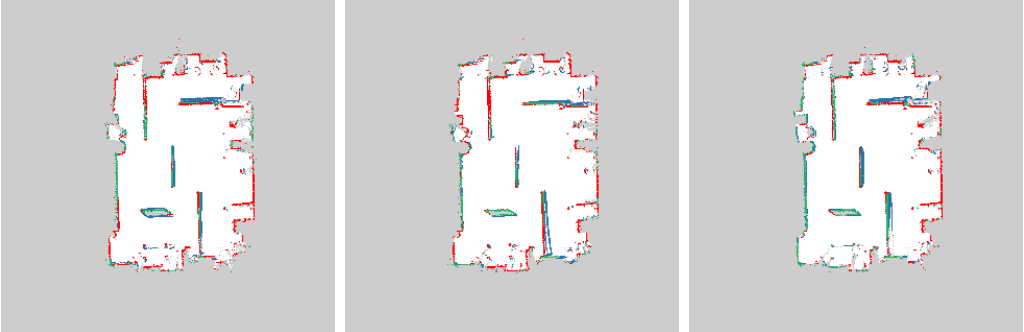


Figure 5.43: Experiment 4 - RTAB-Map ground truth



(a) Experiment 4.1 **(b)** Experiment 4.2 **(c)** Experiment 4.3

Figure 5.44: Experiment 4 - Mapping results



(a) Experiment 4.1 **(b)** Experiment 4.2 **(c)** Experiment 4.3

Figure 5.45: Experiment 4 - Confusion error metric (Green – TP, White – TN, Blue – FP, Red – FN)

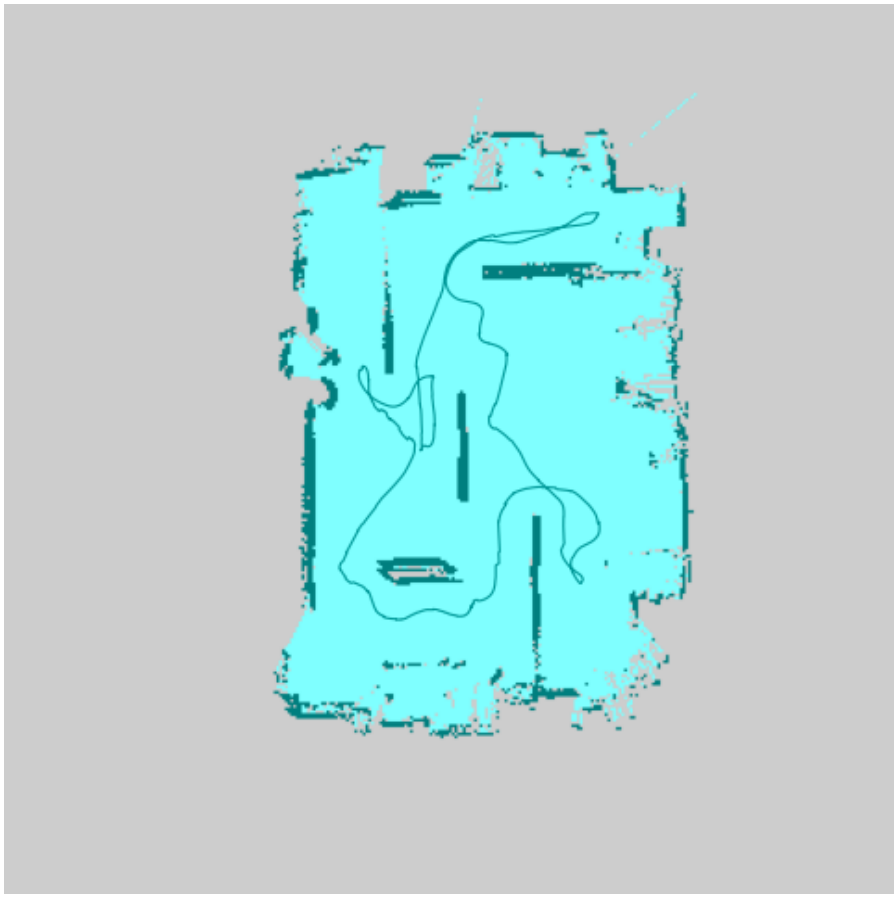
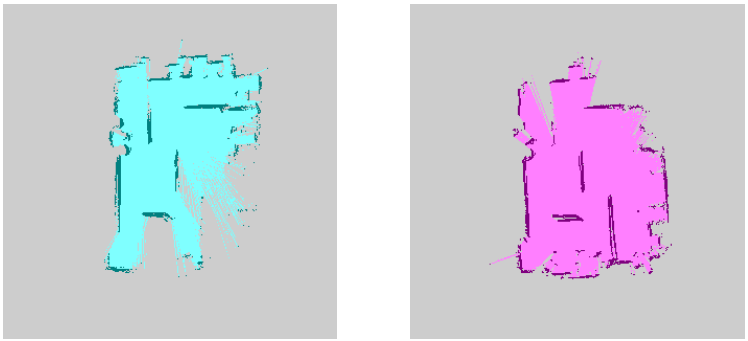


Figure 5.46: Experiment 4.1 - Robot exploration and trajectory



(a) *robot_12* map **(b)** *robot_13* map

Figure 5.47: Experiment 4.2 - Single robots exploration maps

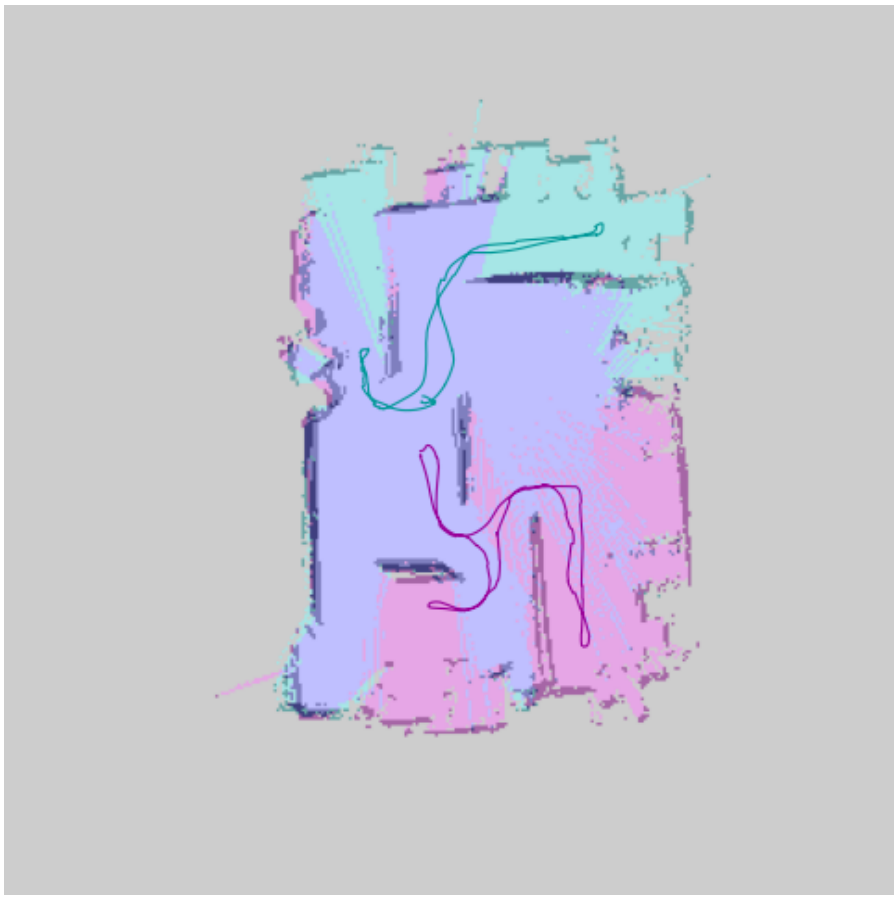
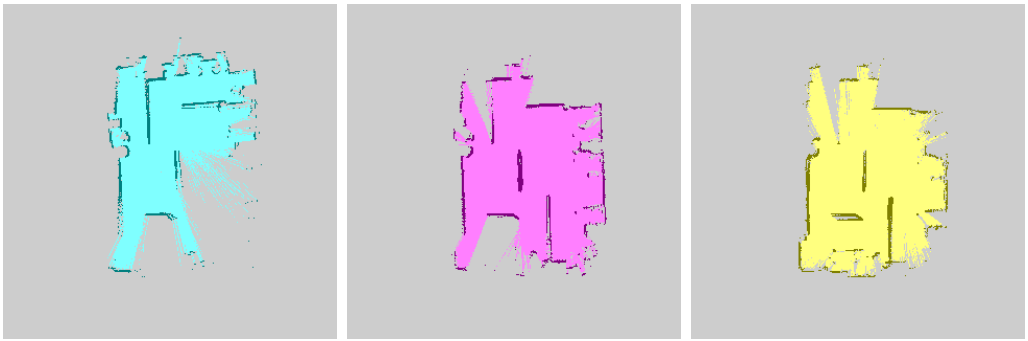


Figure 5.48: Experiment 4.2 - Robots exploration and trajectories



(a) *robot_12* map **(b)** *robot_13* map **(c)** *robot_14* map

Figure 5.49: Experiment 4.3 - Single robots exploration maps

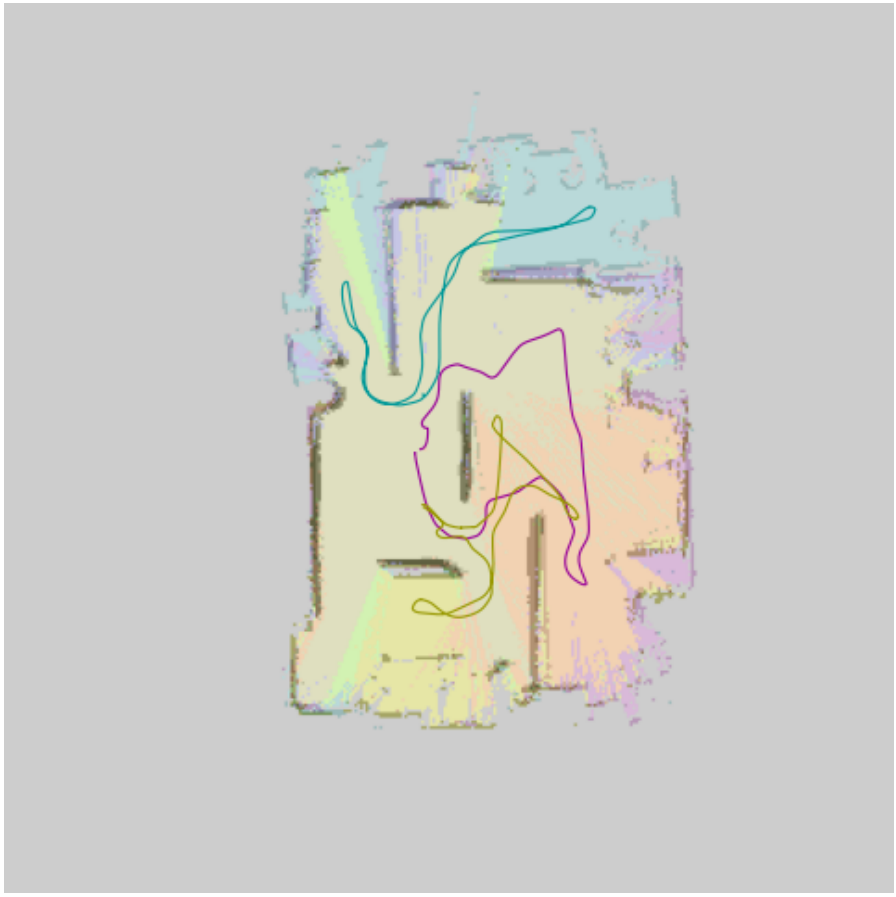


Figure 5.50: Experiment 4.3 - Robots exploration and trajectories

5.5.5 Experiment 5

In the following description of results, a single experiment was performed with the robot configuration:(*robot_12*, *robot_13*, *robot_14*).

This experiment aims to highlight the efficiency of the multi-robot system in simultaneously mapping different sections of a large environment. The objective is to assess the system's ability to autonomously identify multiple exploration frontiers and assign them to different robots in a coordinated and efficient manner.

The results demonstrate that the system is capable of recognizing the main distinct frontiers and effectively directing each robot toward separate unexplored regions, thereby maximizing overall coverage while minimizing redundant paths. This coordinated behavior leads to a significant reduction in the individual coverage required from each robot, confirming the scalability and robustness of the proposed multi-robot exploration strategy in large environments.

Timing evaluation

The execution time for this mapping experiment is 223.219 s.

Mapping evaluation

Robots	**Area [m²]**	**Rate [%]**
robot_12	67.26	67.52%
robot_13	60.80	61.04%
robot_14	50.88	51.06%
robot_12, robot_13	86.02	86.35%
robot_12, robot_14	87.29	87.63%
robot_13, robot_14	75.78	76.08%
robot_12, robot_13, robot_14	99.61	100.00%

Table 5.17: Experiment 5 - Exploration coverage for each robots combination

Metric	**Value**
Accuracy	90.2%
True Positive Rate (TPR)	21.9%
True Negative Rate (TNR)	95.3%
False Positive Rate (FPR)	4.7%
False Negative Rate (FNR)	78.1%

Table 5.18: Experiment 5 - Mapping accuracy confusion-based rate metrics

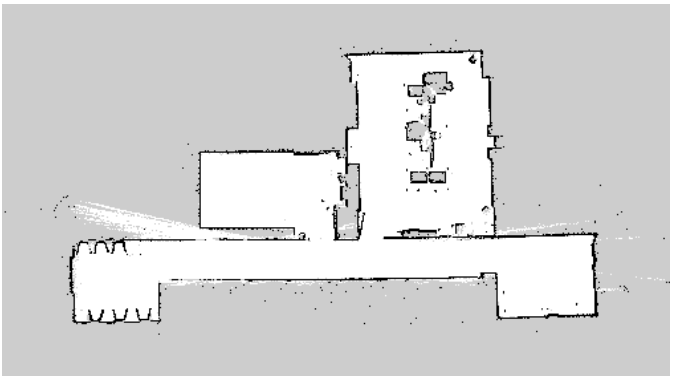


Figure 5.51: Experiment 5 - RTAB-Map ground truth

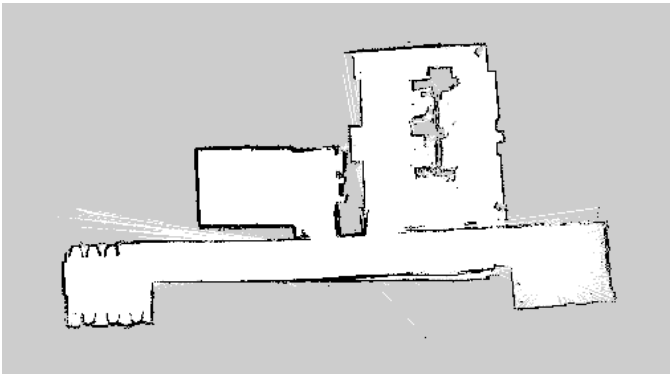


Figure 5.52: Experiment 5 - Mapping results

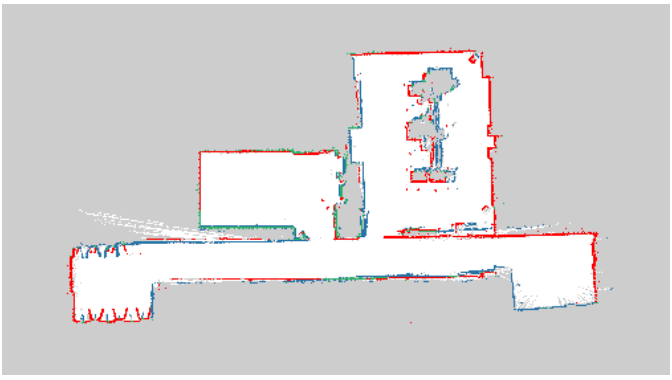
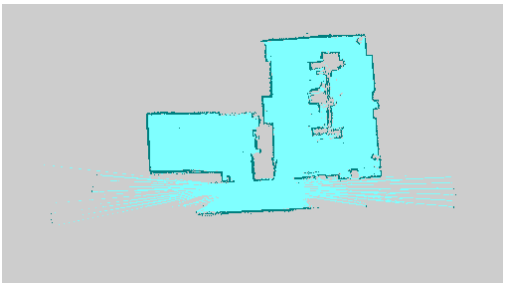
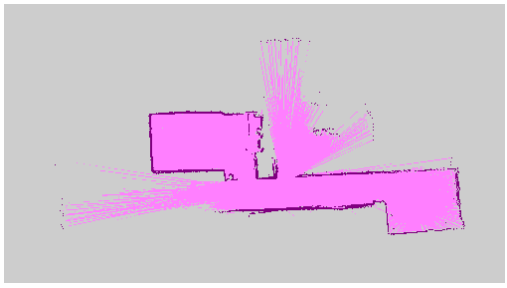


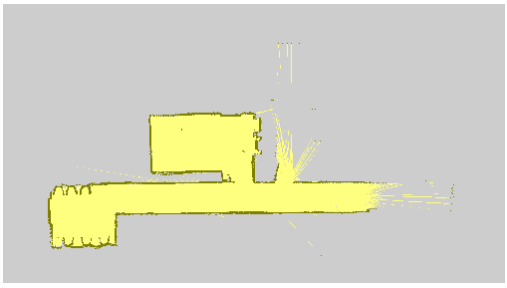
Figure 5.53: Experiment 5 - Confusion error metric (Green – TP, White – TN, Blue – FP, Red – FN)



(a) *robot_12* map



(b) *robot_13* map



(c) *robot_14* map

Figure 5.54: Experiment 5 - Single robots exploration maps

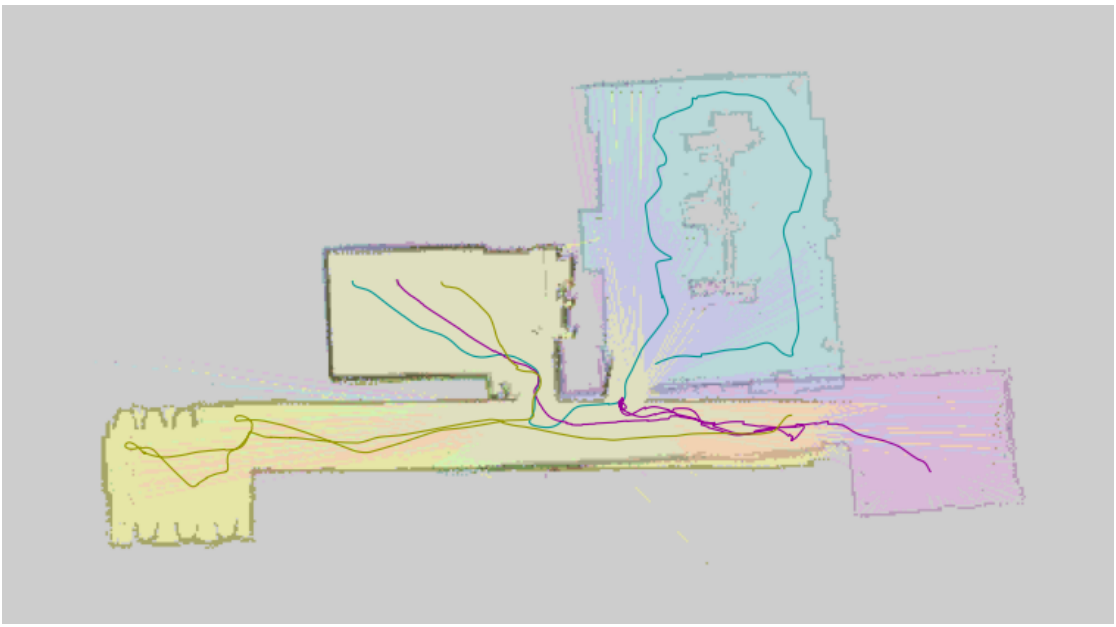


Figure 5.55: Experiment 5 - Robots exploration and trajectories

Chapter 6

Conclusions and Future Works

This thesis has focused on the development of a centralized multi-robot Active Collaborative SLAM (AC-SLAM) framework, with the primary objective of establishing a solid baseline for future research and improvement in multi-robot mapping, localization, decision-making, and planning. The framework also emphasizes the use of factor-graph formulations as the core optimization tool for representing and solving complex estimation and coordination problems in a structured and modular way.

Factor graphs provide a clear and flexible representation of the relationships between robot states, sensor measurements, and control inputs. Their modularity allows new constraints or measurement types to be seamlessly integrated without reformulating the entire problem. Moreover, the ease of interpretation and implementation of factor-graph-based methods makes them well suited for large-scale multi-robot systems, where the same mathematical structure can be applied to localization, mapping, and trajectory optimization within a unified framework.

The centralized AC-SLAM approach developed in this work enables multiple ground rovers to share information and collaboratively build a globally consistent map while maintaining accurate localization. Through centralized inference, the system ensures coherence among all agents, facilitating coordinated exploration and efficient coverage of large or complex environments. The framework also optimizes robot trajectories to prevent collisions and redundant exploration, leading to smoother motion and improved efficiency across the team.

The experimental results demonstrate that the proposed framework can successfully coordinate multiple robots operating in semi-structured environments. The system achieves smooth and dynamically feasible trajectories, good localization accuracy, and globally consistent mapping performance. The centralized optimization balances local autonomy with global coherence, reducing uncertainty in both individual robot poses and the shared map. These results confirm the potential of factor-graph-based centralized approaches as a solid foundation for large-scale,

cooperative multi-robot exploration and mapping.

Nevertheless, while the framework establishes an effective foundation, it also reveals further research to enhance scalability, robustness, and autonomy. In particular, scalability remains a key challenge, as the centralized architecture may face computational and communication limitations when deployed with a very large number of robots. Future developments should therefore explore distributed inference and collaborative sensing strategies to overcome these limitations and push the boundaries of fully autonomous, cooperative multi-robot systems.

6.1 Future Works

- **Collaborative components in SLAM:** Incorporating inter-robot measurements and sharing robust landmarks between robots would reduce drift, and improve localization accuracy. This approach enhances map consistency and robustness, even in scenarios with limited communication, noisy sensors, or partial observations, enabling more reliable multi-robot coordination and cooperative exploration.

- **Distributed and hybrid architectures:** Transitioning from a purely centralized to a distributed or hybrid optimization framework would improve scalability and robustness. Techniques such as Gaussian Belief Propagation or distributed factor graph optimization could allow robots to perform local computations while maintaining global consistency through asynchronous updates, thereby reducing reliance on a central node.

- **Heterogeneous multi-robot systems:** Adapting the framework to heterogeneous teams combining aerial and ground robots would improve coverage in complex scenarios such as large-scale agriculture or search and rescue scenarios.

- **Learning-based optimization and decision-making:** Integrating reinforcement learning could enable adaptive frontier selection, exploration policies, and communication strategies, allowing the system to learn from prior experiences and improve efficiency over time.

Bibliography

[1] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. «iSAM: Incremental Smoothing and Mapping». In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378. DOI: 10.1109/TRO.2008.2006706 (cit. on p. 7).

[2] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. «iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering». In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641 (cit. on p. 7).

[3] Yetong Zhang, Ming Hsiao, Jing Dong, Jakob Engel, and Frank Dellaert. «MR-iSAM2: Incremental Smoothing and Mapping with Multi-Root Bayes Tree for Multi-Robot SLAM». In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8671–8678. DOI: 10.1109/IROS51168.2021.9636687 (cit. on p. 7).

[4] Frank Dellaert. «Factor Graphs: Exploiting Structure in Robotics». In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 (May 2021). DOI: 10.1146/annurev-control-061520-010504 (cit. on p. 14).

[5] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos. *A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers*. 2023. arXiv: 2207.00254 [cs.RO]. URL: https://arxiv.org/abs/2207.00254 (cit. on p. 17).

[6] B. Al-Tawil, T. Hempel, A. Abdelrahman, and A. Al-Hamadi. «A review of visual SLAM for robotics: evolution, properties, and future applications». In: *Frontiers in Robotics and AI* 11 (2024), p. 1347985. DOI: 10.3389/frobt.2024.1347985 (cit. on p. 18).

[7] Muhammad Farhan Ahmed, Khayyam Masood, Vincent Fremont, and Isabelle Fantoni. «Active SLAM: A Review on Last Decade». In: *Sensors* 23.19 (Sept. 2023), p. 8097. ISSN: 1424-8220. DOI: 10.3390/s23198097. URL: http://dx.doi.org/10.3390/s23198097 (cit. on p. 19).

[8] Phillip Quin, Dac Nguyen, Thanh Vu, Alen Alempijevic, and Gavin Paul. «Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration». In: *Frontiers in Robotics and AI* 8 (Feb. 2021), p. 616470. DOI: 10.3389/frobt.2021.616470 (cit. on p. 20).

[9] Aalok Patwardhan and Andrew J. Davison. «A Distributed Multi-Robot Framework for Exploration, Information Acquisition and Consensus». In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 12062–12068. DOI: 10.1109/ICRA57147.2024.10610185 (cit. on p. 23).

[10] Aalok Patwardhan, Riku Murai, and Andrew J. Davison. «Distributing Collaborative Multi-Robot Planning With Gaussian Belief Propagation». In: *IEEE Robotics and Automation Letters* 8.2 (Feb. 2023), pp. 552–559. ISSN: 2377-3774. DOI: 10.1109/lra.2022.3227858. URL: http://dx.doi.org/10.1109/LRA.2022.3227858 (cit. on p. 48).