Universidad Zaragoza
1474

Master's Thesis

Radio-based Multi-robot Odometry And Localization

Author

Andrés Martínez Silva

Supervisors

Fernando Caballero Benítez
Luis Merino Cabañas

Academic Supervisor

Natalia Ayuso Escuer

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2025

ACKNOWLEDGEMENTS

This document is the culmination of many dedicated hours that have been put into this work, as well as the master's program that preceded it. The last year has been particularly challenging in multiple unexpected ways, and I could not have got to this point without the support of my family and my tutors in both Zaragoza and Seville. Thanks to Natalia Ayuso, Luis Montano and Danilo Tardioli from University of Zaragoza, for helping me find a way through in a very tough situation, and of course, thanks to my tutors at Pablo de Olavide University, Fernando Caballero and Luis Merino, for providing very valuable support and introducing me to the wonderful team at the Service Robotics Lab of University Pablo de Olavide, Seville.

Abstract

Deployment of multi-robot teams in cooperative missions, like search and rescue, or exploration of large and dangerous environments (i.e. tunnels, mines, or rubble resulting from an accident), brings many unique challenges that are pushing research in exciting new ways. Particularly, localization is a crucial component which typically relies on cameras or Light Detection And Ranging (LiDAR) sensors and underperforms in presence of poor lighting or dust in interiors, and adverse weather conditions in exteriors. In this context, radio-based methods such as Ultra-Wideband (UWB) and RAdio Detection And Ranging (RADAR), which have been traditionally undersubscribed in robotics, are experiencing a boost in popularity thanks to their robustness to harsh environmental conditions and cluttered environments. Hence, this work proposes a multi-robot localization system that leverages the two technologies with inexpensive and readily-available sensors (Inertial Measurement Units, or IMUs, and wheel encoders) to estimate the position of a ground robot and an aerial robot with respect to a global reference frame, by using a pose-graph optimization framework with inter-robot constraints. The system has been developed for ROS 2 using the Ceres optimizer, and has shown promising results in simulated conditions and on a real-world dataset. Furthermore, the standard factor graph formulation also makes it easily extensible to a full Simultaneous Localization And Mapping (SLAM) problem.

Contents

Chapter 1

Introduction

1.1 Motivation

Heterogeneous multi-robot systems have gained increasing popularity in the last few years. In particular, search and rescue missions often require operation in highly cluttered scenarios, with very limited maneuverability and some areas which may become inaccessible to humans and certain types of robots. In this context, deploying a team of different robots that work cooperatively -p.e. an Autonomous Ground Vehicle (AGV) and an Unmanned Aerial Vehicle (UAV) - can bring many benefits: one robot may be allowed to access locations that are blocked to the other, enabling the team to cover more terrain in less time (see Figure 1.1a). Additionally, information sharing between robots allows for a more robust localization, which becomes essential in GPS-denied environments. Each robot must be able to determine its own position and the position of the other robot relative to itself. Although most research focuses on vision or lidar-based methods for localization, the cluttered layout of some areas, together with unfavorable environmental conditions such as fog in exteriors, dust in interiors, or poor lighting, critically impacts the performance of these methods (see Figure 1.1b). This is why radio-based methods are attracting interest in the literature, with two types of sensors emerging as promising alternatives in robotic systems: Ultra-Wideband (UWB) and RAdio Detection And Ranging (RADAR).

On the one hand, UWB sensors provide high-frequency range measurements between anchors and tags with centimeter-level precision [3]. They are also lightweight, cost-effective, and robust to harsh environmental conditions and obstructions, due to their ability to penetrate obstacles. Standard UWB localization systems require the anchors to be installed in the environment for effective triangulation, but this is often impractical. A robotic team, on the other hand, can carry transceivers onboard to obtain inter-robot distances that can be fused with different sources of odometry to localize both robots in real time. On the other hand, RADARs are able to estimate

(a) (b)

Figure 1.1: (a) UAV and AGV team in Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge [1] (b) Team of hazard detection and rescue quadruped robots [2]

both the position (in the form of range, azimuth and elevation) and radial velocity of nearby objects, by leveraging the Doppler effect. RADAR data, however, brings a particular set of challenges: it is sparser than LiDAR or vision sources, with lower angular and spatial resolution, and sensitive to noise from multiple sources, such as multipath reflections and signal scattering. Nonetheless, it has already shown good results in harsh weather and tight layouts, where the other methods generally struggle.

While inter-robot transformations have been widely covered in literature, there is limited research involving on-board UWB transceivers as a source of relative information. Moreover, radar-based localization remains still widely undocumented. In fact, these sensors have mostly been tested separately, mainly in single-robot setups, and there is very little work on heterogeneous multi-robot localization systems that exploit the advantages of both.

1.2 Objectives

The main objective of this thesis is to design and implement a cooperative localization framework for a team of two different robots -an AGV and an UAV- equipped each with multiple UWB transceivers and a RADAR sensor. This involves, first, estimating the relative transformation between the two robots based on distance measurements via nonlinear optimization, and then building a multi-robot pose-graph that simultaneously optimizes the positions of both platforms by fusing all available sensor data, including pre-processed RADAR point clouds and other readily-available sensors such as IMUs or wheel encoders. The previously estimated inter-robot transformation is integrated in the pose-graph in order to localize both robots with respect to a common frame of reference. The system will be tested in simulation, and then on a real world dataset.

1.3 Document structure

This document is structured as follows:

– **Chapter 2** reviews the state of the art on radio-based localization with a focus on inter-robot transformations, RADAR odometry and cooperative systems. It also provides a brief description of the software framework and hardware used to validate the proposed solution.

– **Chapter 3** introduces a theoretical formulation of the challenge to be addressed, from range-based estimation to using RADAR as an odometry source, and finally fusing this information in a pose-graph to simultaneously localize a team of an UAV and an AGV.

– **Chapter 4** focuses on implementation details regarding the specific setup used to validate the proposed approach, including software architecture, flow diagrams and pseudo-code.

– **Chapter 5** presents and analyzes the experimental results obtained both in simulation and in a real-world dataset.

– **Chapter 6** concludes the thesis presenting some closing statements and future lines of work.

Chapter 2

Preliminaries

2.1 Related work

The problem of estimating inter-robot relative transformations based on distance measurements has been extensively studied in [4], where the authors propose an algebraic method to compute a 6 Degrees of Freedom (DOF) rigid transformation between robot frames. A closed-form solution can be found from ten distance measurements, and a discrete set of 40 solutions from as few as six [5]. In [6], the authors particularize the case to UWB sensors and achieve a 4-DOF closed-form solution from just six measurements. However, these methods are highly sensible to outliers and noise and typically require further refinement with Nonlinear Least Squares (NLS). More recently, the authors of [7] perform an exhaustive analysis on 4-DOF relative frame transformation (RFT) estimation using UWB and visual-inertial odometry, proposing a global estimation approach to improve NLS initialization. However, these methods assume only one sensor per platform and do not exploit redundancy and frequency of multiple onboard measurements. Other research has considered up to four onboard UWB nodes [8] to compute instantaneous body frame transformations, though it relies on the availability of sufficient simultaneous measurements. In heterogeneous teams of robots, this assumption is often challenged due to limitations in sensor placement, communications quality and noise.

RADAR-based localization and SLAM have been primarily explored in the context of autonomous driving, due to its advantages over other sensor modalities in adverse weather conditions. In [9], the authors present a RADAR-only motion estimation system using a Frequency Modulated Continuous Wave (FMCW) RADAR without Doppler information. In [10], a real-time pose-graph SLAM system that leverages Iterative Closest Point (ICP)-based scan-matching to obtain relative transformations between consecutive RADAR detections is introduced. To localize the vehicle, the odometry component implements an Unscented Kalman Filter (UKF) that fuses wheel speed, yaw rate, steering wheel angle and velocity of the RADAR sensor. The latter is obtained from radial (Doppler) velocities using a RADAR ego-motion estimation algorithm [11]. In robotics, RADAR-based systems have only recently seen significant development. Notably, the authors of [12] introduced an open source Robot Operating System (ROS) package that implements real-time 6-DOF SLAM for 4D RADARs, and the authors of [13] managed to localize a robotic platform in the presence of heavy fog by fusing LiDAR and low-cost RADAR data.

Finally, cooperative localization involving a team of robots has been widely covered in the past [14, 15, 16]. However, it is difficult to find relevant work regarding teams of different robots that localize themselves with radio sensors, especially combining UWB and RADAR. To the best of our knowledge, this is the first work that integrates relative transformation estimation as an inter-robot factor in a multi-robot pose-graph optimization framework, together with RADAR odometry factors.

2.2 Methodology

The work developed during the elaboration of this thesis was tested in simulated conditions and with a real dataset. The project comprises three main modules that were developed and tested in isolation, then integrated into a full localization system: 1) relative transform estimation from UWB measurements, 2) multi-robot pose-graph optimization, and 3) RADAR odometry. This modular approach allows us to test the first two modules by feeding them virtual odometry sources and distance measurements, which are modeled with different sources of noise. Once performance achieves the expected results under different simulation conditions, we can substitute these inputs with real data from the dataset, which includes RADAR information. Due to the difficulty of realistically simulating RADAR measurements, which are very sparse and noisy, this module was only tested with real data once the rest of the system was validated. For this methodology to work, the simulated modules replicate sensor placement from the real system, as well as interfaces and message formats. With careful parameterization, this enables seamless integration with both simulated and real data.

Figure 2.1 shows a Gantt diagram of the project timeline. Note that development and experimentation follow an iterative process that runs until the expected performance is met. Then, the final results can be obtained and analyzed.

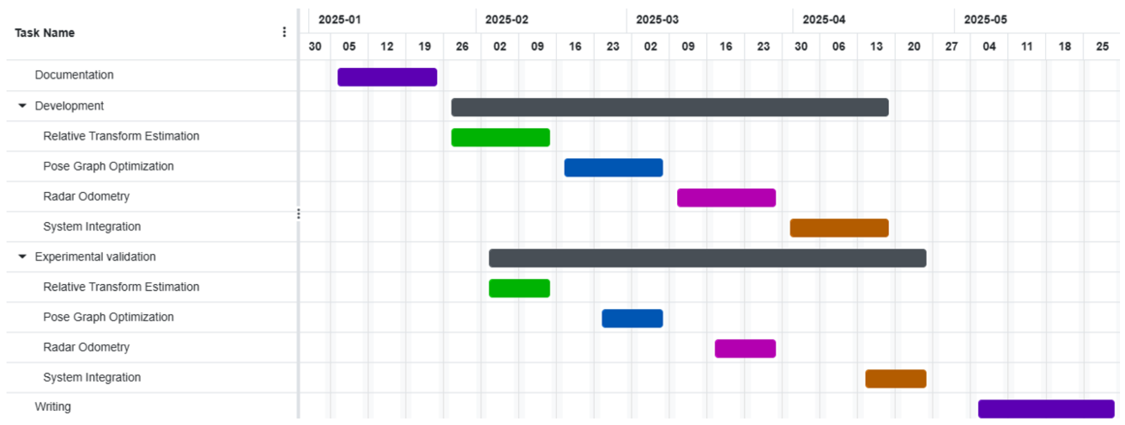


Figure 2.1: Gantt diagram showcasing the overall planning for the project. It includes documentation, development, experimentation, and writing of the final thesis.

2.3 Validation framework

2.3.1 Radio sensors

The Eliko Real-Time Location System [3] was used to obtain relative distance measurements between the two platforms. The system allows tracking any objects in 2D or 3D space, using UWB radio waves to measure distance between its two basic hardware components: anchors and tags. This technology operates by sending short radio pulses across a wide frequency spectrum. By measuring Time-of-Flight (ToF) distances, this system is able to obtain centimeter-level precision in ranges of up to 70 m even in challenging environments.

The Continental ARS548 RADAR [17] is the other sensor used in the experiments. This is a 4D RADAR that measures distance, Doppler velocity, azimuth, and elevation of detected objects. It operates on the 76-77 GHz band with real-time scanning across the full Field of View (FoV) of 20 Hz. Among its main technical features, the RADAR operates on a distance range from 0.2 to 302 m, with a distance resolution of 0.22 m, a speed resolution of 0.35 km/h and an angular resolution as small as 0.1º. On the other hand, it has an azimuth FoV of ± 60º and an elevation FoV of ± 14º, meaning that the overall FoV is quite restricted as compared to a commercial 3D LiDAR.

2.3.2 Robotic platforms

The team of robots used in the dataset experiments are the ARCO (Autonomous Robot CO-worker) ground platform developed by IDMind and the Matrice M210 V2 UAV developed by DJI. The ARCO platform is a 4-wheeled platform with an independent traction system that supports both holonomic and differential configurations. In this setup, the robot is equipped with a 9-DOF IMU, a 3D OS-1-16 LiDAR Ouster,

Figure 2.2: (a) Eliko RTLS anchor (left) and tag (right) (b) ARS548 Continental RADAR

four Eliko UWB anchors powered by PoE (Power over Ethernet), and an ARS548 Continental RADAR. A more detailed overview of the platform can be found in [18]. The M210 is a commercial lightweight quadcopter that, for the purposes of these experiments, has been equipped with two battery-powered Eliko UWB tags, a 3D OS-1-16 LiDAR Ouster and an ARS548 Continental RADAR. The platform includes dual 6-DOF IMUs and barometers that are used to provide fused velocity estimations in the absence of GPS. Additional information about the platform can be found in [19].

For more information, refer to Appendix A, where a picture highlighting the sensor placement in both platforms is included.

2.3.3 ROS2 and Ceres

The methods presented in this thesis have been developed in ROS 2 [20]. It is the second generation of the Robot Operating System, a powerful open source framework for developing robot applications. It provides a modular architecture comprising middleware, communications and developer tools. While ROS 1 was widely adopted in research, it had major drawbacks that hindered its deployment in production, including 1) a centralized architecture with a single point of failure (the *rosmaster*), 2) a custom TCP/UDP network transport protocol that struggled over faulty links, and 3) a lack of built-in security. In contrast, ROS 2 adopts a distributed design, leveraging industry-standard DDS (Data Distribution Service), event-based asynchronous communications, and emphasizes modularity and security. With ROS 1 reaching End-Of-Life in 2025, the ROS 2 design philosophy justifies the migration and makes it specially fitting for challenging applications such as multi-robot systems. In this work, ROS 2 is used to handle communications over all hardware and software modules, as well as telemetry and data visualization. Particularly, the Eliko system and ARS548 RADAR (see Section 2.3.1) interface with ROS 2 through custom drivers developed by the Social Robotics Lab and available for public use [21].

Both the ARCO and the M210 are set up to communicate with ROS 2 for control and telemetry. In this work, we subscribe to sensor measurements from wheel encoders and IMU in the case of the ARCO, and, in the case of the UAV, we obtain fused velocity data from the autopilot through ROS2 topics via the DJI SDK.

Alternatively, Ceres [22] is an open source C++ library developed by Google for modeling and solving optimization problems. Together with ROS 2, it plays a key role in this work, as both the relative transform estimation and pose-graph optimization are formulated as Nonlinear Least Squares (NLS) problems. Among the features that are useful to this work, it supports automatic differentiation, custom manifold definitions, robust loss functions, and covariance estimation. It also supports parallelization and Graphical Processing Unit (GPU) acceleration, and provides numerous performance metrics and diagnostic tools.

2.4 Chapter recap

This chapter included a review of relevant literature in the field of range-based localization in Section 2.1, then defined the general methodology in Section 2.2 and highlighted the main hardware and software tools being used in the thesis in Sections 2.3.

The next chapter delves into the theoretical formulation of the problem, including a detailed explanation of the specific challenges behind this work and the methods proposed to solve them.

Chapter 3

Problem formulation

In this chapter, the core concepts about relative transform estimation, RADAR odometry and pose-graph optimization are introduced. First, each of the individual components of the system is described in depth, and then we propose a framework where the three modules are integrated into a multi-robot localization system.

3.1 Relative localization

Suppose a heterogeneous team of two robots equipped with UWB transceivers are moving in proximity, each of them describing a trajectory in their own local reference. The UAV (robot 1) is equipped with two tags, and the AGV (robot 2) is equipped with four anchors - see Figure 3.1. For more information on the basics of UWB localization, refer to Appendix B.

Let $\tilde{\mathbf{p}} = [x, y, z, 1]^T \in \mathbb{R}^4$ denote the homogeneous 3D coordinates of a point in space. Let ${}^{\mathbf{B_1}}\tilde{\mathbf{p}}_{\mathbf{ti}} \in \mathbb{R}^4$ be the known homogeneous coordinates of UWB tag i in the body frame of robot 1, and ${}^{\mathbf{B_2}}\tilde{\mathbf{p}}_{\mathbf{aj}} \in \mathbb{R}^4$ the known homogeneous coordinates of UWB anchor j in the body frame of robot 2, and let ${}^{\mathbf{O_1}}\mathbf{p_1}$ and ${}^{\mathbf{O_2}}\mathbf{p_2} \in$ SE(3) be the poses of robots 1 and 2 in their own local odometry frames, respectively. Consequently, the homogeneous coordinates of tag i in the odometry frame of robot 1, ${}^{\mathbf{O_1}}\tilde{\mathbf{p_{ti}}} \in \mathbb{R}^4$, and the homogeneous coordinates of anchor j in the odometry frame of robot 2, ${}^{\mathbf{O_2}}\tilde{\mathbf{p_{aj}}} \in \mathbb{R}^4$, are known. In order to simplify the problem, we reduce the problem to 4 DOF by making the assumption that the roll and pitch of each robot's local odometry frame are known and fixed. The IMUs onboard most robotic platforms are able to calculate roll and pitch states very accurately thanks to the integration of gyroscopes and accelerometers, and they are fully observable. Thus, only the translation components and the yaw angle are expected to exhibit significant drift.

In this way, we can define a state vector $\mathbf{x} = [\text{x,y,z,}\psi]^T$, comprised of three translation components and the yaw angle ψ, and T($\mathbf{x}$, θ, ϕ) $\in$ SE(3) as the rigid

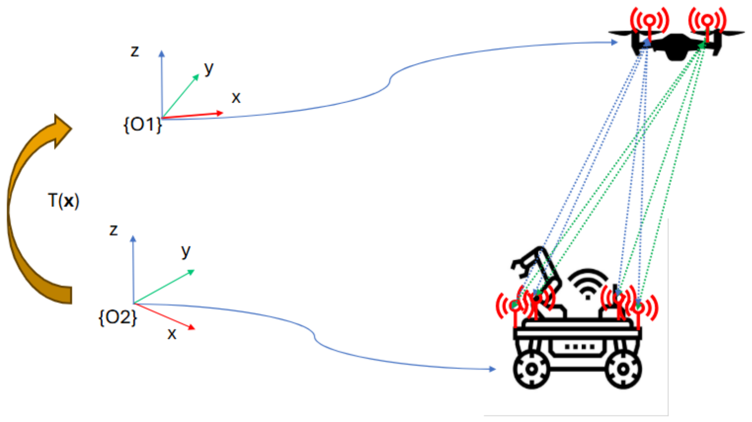


Figure 3.1: Each robot describes a trajectory in its own local odometry frame while collecting range measurements (dashed lines) between two tags and four anchors. Measurement from different tags are represented in different colors.

transformation between the odometry frames defined by the state vector, and the known relative roll θ and pitch ϕ. If both robots start their trajectories on the horizontal plane, these values are set to zero. If not, they can be measured precisely (e.g. through IMU readings at the starting points) and introduced as fixed values. For ease of notation, from now on we will refer to this transformation as T($\mathbf{x}$). With this information, it is possible to describe a measurement model, where the predicted distance $\hat{d}(t_i, a_j)$ between tag i and anchor j is defined as follows:

$$\hat{d}(t_i, a_j) = ||^{\mathbf{O_1}}\tilde{\mathbf{p_{ti}}} - T(\mathbf{x})^{\mathbf{O_2}}\tilde{\mathbf{p_{aj}}}|| + \epsilon \tag{3.1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ represents an independent, identically distributed Gaussian noise with $\sigma = 10$ cm, which is the uncertainty associated to the measurement. This value has been selected according to the accuracy level stated in the technical specifications of the commercial Eliko UWB system [3].

The objective is to find the inter-robot transformation T($\mathbf{x}$) between the odometry frames of both platforms, which allows us to map the local positions of robot 2 to the local reference frame of robot 1, according to the anchor-tag range measurements d. This is an optimization problem that can be formulated in terms of Nonlinear Least Squares (NLS). NLS methods aim to find the minimum of an objective function J by performing a succession of linear approximations from an initial estimate until a convergence criterion is met. The objective function is defined according to the general NLS formulation outlined in [22], where J minimizes the sum of squared residuals over one or more cost functions, which are the different terms in the objective function. Each cost function defines a residual based on the difference between a prediction -expressed as a function of the optimization parameters- and an observation, often normalized by the uncertainty associated to that observation.

In this problem, the objective function is defined as:

$$J = \min_{\mathbf{x}} \sum_{W} (\sum_{i,j} (\frac{\hat{d}(t_i, a_j) - d}{\sigma})^2 + ||e_{4D}(\mathbf{T}(\mathbf{x}), \mathbf{p})||^2_{\Sigma}) \tag{3.2}$$

The first term is a scalar that, for each anchor-tag pair, minimizes the difference between the predicted and the actual measurement, which is normalized by dividing by its standard deviation σ. The second term is used to anchor the estimated transform to a prior $\mathbf{p} \in SE(3)$, where $e_{4D}(.)$ is an error function that obtains a 4D residual by comparing the translational and yaw components of two SE(3) poses. More information on how this is performed is presented in Appendix C. Finally, $||\mathbf{e}||_{\Sigma}$ is the squared Mahalanobis distance $\mathbf{e^T \Sigma^{-1} e}$ with covariance matrix Σ, and W is a distance-based sliding window comprising from 2 to 10 m of each robot trajectory.

When working with noisy sensor data, the presence of outliers can lead to the computation of artificially large residuals that can potentially compromise the performance of the optimization. To address this, the cost functions are robustified, meaning that a robust loss function - in this case, the Cauchy loss [23] - is used to reduce the influence of outliers in residual computation. In this way, only residuals within an acceptable range (two or three times the expected standard deviation) are used in the optimization. This is specified as a parameter when defining the cost functions and handled internally by the Ceres Solver (more details on the implementation can be found in Chapter 4).

3.1.1 Dilution of precision

Dilution of Precision (DoP) is a well-known phenomenon in range-based localization that plays a key role in the performance of the system. In UWB localization, it refers to how the geometry configurations of anchors and tags affect the uncertainty of the position estimate. For the problem to be well-constrained, the transceivers (i.e. anchors and tags) must be placed as far apart as possible, and covering as much as possible of the available space in both the horizontal and vertical directions. Otherwise, poor geometric diversity leads to poor observability of the states that we want to estimate, making the optimization problem ill-conditioned. This arises from the nature of omnidirectional range measurements, which the optimizer uses to trilaterate the position of the transceiver and can be visualized as spheres around the anchors. When the anchors are spaced enough, the intersection between the spheres - which yields the location of the tag - results in a smaller area of uncertainty. In contrast, when

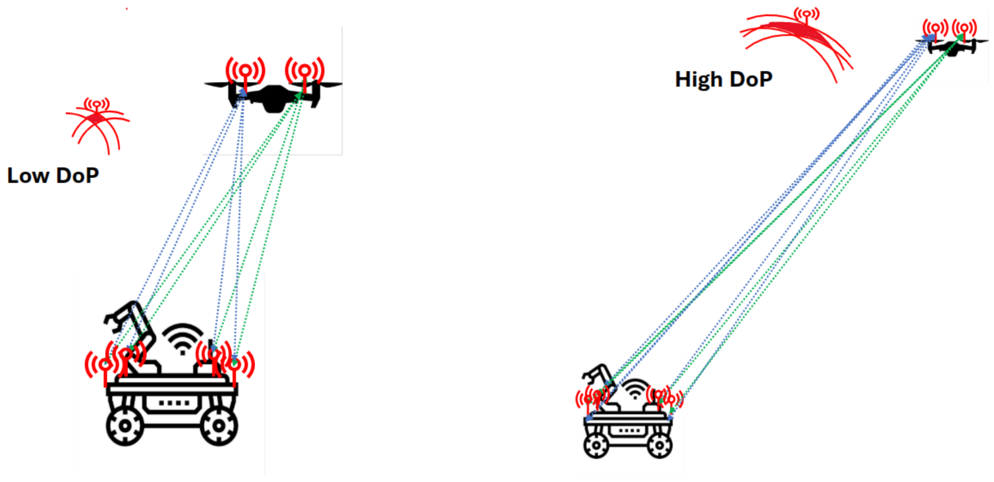


Figure 3.2: The effect of Dilution of Precision as the robots move far away from each other. When the distance between the transceivers is too small with respect to the distance between the robots, the uncertainty of the trilaterated position (highlighted in red) increases, and the problem becomes ill-conditioned.

the distance between the anchors is too small compared to the distance to the tag, the spheres become nearly tangent, producing a larger region of intersection and thus greater uncertainty. This is illustrated in Figure 3.2.

In the setup used for this work, the ground vehicle is equipped with four anchors arranged in a 3D rectangular prism with dimensions 64x60x55 cm (see Appendix A.2). The drone carries two tags placed horizontally with a 48 cm separation. This distribution is limited by the space available on the platform, imposing a practical bound on the operational range of the system. As a result, the proposed UWB localization system is only expected to perform at reasonable distances given these bounds (approximately below 10 m separation).

3.1.2 Closed-form prior

NLS methods are by definition sensitive to initial conditions, where poor values can increase the risk of converging to local minima, or not converging at all. One of the most popular methods for solving NLS is Levenberg-Marquardt, which leverages step size depending on the magnitude of the gradient of the cost function. However, as mentioned in Section 2.1, even then this method struggles under a suboptimal initial guess, often being used as a refinement step. To test the robustness of the proposed solution, the closed-form linear solution proposed in [6] is incorporated as a prior. This method solves an over-determined linear system from the set of available measurements, using a RANSAC estimator to handle outliers. The first time the optimization is run, the linear algorithm is used to find a potential candidate to initialize the state, and added as a robustified cost term. For future solves, the initial state is set to the previous optimized state in order to have a smooth estimate, but the linear solution, if available, is still incorporated in the cost function (see second term of Equation 3.2). This setup allows us to analyze whether the prior contributes to convergence, or if the system is robust enough without it.

3.2 RADAR odometry

Section 3.1 discussed inter-robot localization (this is, determining the position of one robot relative to the other). Another fundamental component of a localization system is odometry, which estimates a robot's motion with respect to its own local frame. In the previous section, we assumed this information was available and sufficiently accurate. However, in practice, odometry derived from wheel encoders and IMUs tends to accumulate drift over time. While LiDAR and vision-based methods are extensively used for localization via scan-matching or image correspondences, as mentioned in Section 1.1, they are particularly sensitive to challenging environmental conditions. RADARs, on the other hand, are robust to harsh weather and lighting conditions and can be used for localization using similar techniques. They provide range, azimuth, elevation and radial or Doppler velocity (i.e. the component of the object velocity along the sensor line of sight) with each detection. This information is converted into a sparse pointcloud with per-point velocity estimates. Figure 3.3 shows a basic architecture of a RADAR-based odometry or, by extension, SLAM system. The raw RADAR data is first pre-processed on a front-end to obtain intermediate representations that can be used in the backend to estimate the robot positions (and the map, in the case of SLAM) in a factor graph optimization framework. In SLAM, the front-end is also used to detect loop closures.

In this case, the pre-processing involves a filter to remove false positives and dynamic objects, as well as ego-motion estimation (this is, estimating the velocity of the sensor platform) by fusing radial velocity and IMU data. This information can then be used as a constraint for registration-based scan-matching techniques. The concatenation of consecutive relative transforms estimated with these methods can be used as a standalone odometry system, or fused with additional measurements in a larger localization problem.

3.2.1 Ego-Motion estimation

The first step is to filter the RADAR point cloud and compute a velocity estimation of the sensor platform (also called ego-motion estimation). To do this, we use the

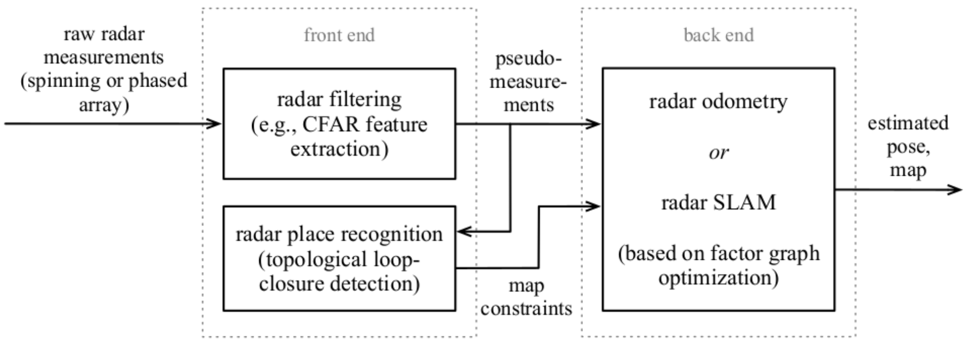

raw radar measurements (spinning or phased array)
front end
radar filtering (e.g., CFAR feature extraction)
radar place recognition (topological loop-closure detection)
pseudo-measurements
map constraints
back end
radar odometry
or
radar SLAM
(based on factor graph optimization)
estimated pose, map

Figure 3.3: Basic architecture of a RADAR Odometry and/or SLAM system, as proposed in [24]

existing work presented in [25], which is a modified version of the original ego-motion estimation algorithm introduced in [11], adapted for holonomic ground robots. The original method defines the Doppler velocity v_D of each detection in a RADAR scan as the projection of the robot velocity vector $\mathbf{v}$ onto the unit vector $\mathbf{h}$ aimed towards the detected point, which is expressed as a dot product:

$$v_D = \mathbf{h} \cdot \mathbf{v} = h_x v_x + h_y v_y + h_z v_z \tag{3.3}$$

For a RADAR scan with N detections, it is possible to build a linear system that can be solved via linear least squares:

$$\begin{bmatrix} v_{D,1} \\ v_{D,2} \\ \vdots \\ v_{D,N} \end{bmatrix} = \begin{bmatrix} h_{1x} & h_{1y} & h_{1z} \\ h_{2x} & h_{2y} & h_{2z} \\ \vdots & \vdots & \vdots \\ h_{Nx} & h_{Ny} & h_{Nz} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{3.4}$$

For ground-based holonomic robots, where motion is typically constrained to the longitudinal (X) and lateral (Y) directions, one degree of freedom can be removed by using the IMU-read roll and pitch to establish a relationship between the three velocity variables:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} \cos(\theta) v_x \\ \cos(\phi) v_y \\ \sin(\theta) v_x + \sin(\phi) v_y \end{bmatrix} \tag{3.5}$$

This formulation improves robustness by accounting for feasible velocities and reduces noise in the Z component, where RADAR data is particularly noisy due to low vertical resolution and reduced FoV. However, for aerial platforms, this loosely coupled formulation does not hold as the robot can exhibit significant velocity changes in all three directions, and the three components need to be estimated independently.

This method works under the assumption of staticity, thus, any moving obstacle in the scene (p.e. the other robot, in this particular application) must be filtered out beforehand. In [11], a RANSAC estimator is used to remove detections corresponding to dynamic elements. An additional filter in [25] is used to discard points by range and height, retaining only those within reasonable bounds.

3.2.2 Scan matching

Iterative Closest Point (ICP) is arguably the most popular scan-matching algorithm due to its simplicity, and a good trade-off between accuracy and efficiency. Even though it has known many variants since it was originally introduced, its basic structure remains unchanged. The original point-to-point ICP consists on an iteration over two steps:

1. For a target point cloud $\mathbf{P}$, and a source point cloud $\mathbf{Q}$ transformed with the current transformation matrix $\mathbf{T}$, find the correspondence set $\mathcal{K} = (\mathbf{p}, \mathbf{q})$
2. Update the transformation matrix $\mathbf{T}$ by minimizing the distance between each pair of correspondences, using cost function $\mathrm{E}(\mathbf{T})$.

$$E(\mathbf{T}) = \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}} \|\mathbf{p} - \mathbf{T}\mathbf{q}\|^2 \tag{3.6}$$

The variant used in this work is Generalized ICP (GICP) [26], which is better suited for handling noisy and sparse scans like those produced by RADAR sensors -see Figure 3.4 for a comparison between RADAR and LiDAR scans. GICP extends the standard ICP by modeling measured points as samples from Gaussian distributions centered at the true points, which are assumed to be in perfect correspondence. This way, each point p_i in the target cloud and q_i in the source cloud are modeled as Gaussian-distributed variables: $p_i \sim \mathcal{N}(\hat{p}_i, \mathbf{C}_i^P)$ and $q_i \sim \mathcal{N}(\hat{q}_i, \mathbf{C}_i^Q)$, where $\hat{p}_i$ and $\hat{q}_i$ are the true values and $\mathbf{C}_i^P$, $\mathbf{C}_i^Q$ are the respective covariance matrices.

The new cost function becomes a Mahalanobis distance between corresponding point distributions:

$$E(\mathbf{T}) = \sum_i \mathbf{d}_i^{(\mathbf{T})^T} \left(\mathbf{C}_i^B + \mathbf{T}\mathbf{C}_i^A\mathbf{T}^T\right)^{-1} \mathbf{d}_i^{(\mathbf{T})} \tag{3.7}$$

where $\mathbf{d}_i^{(\mathbf{T})} = p_i - \mathbf{T}q_i$ is the residual between corresponding points.

In practice, open-source libraries like *PCL* [27] provide standard implementations of ICP and its variants, including GICP. *PCL* includes high-level methods that estimate the relative transform from a source and a target point cloud, providing fitness and performance metrics. Covariance computation is abstracted from the user

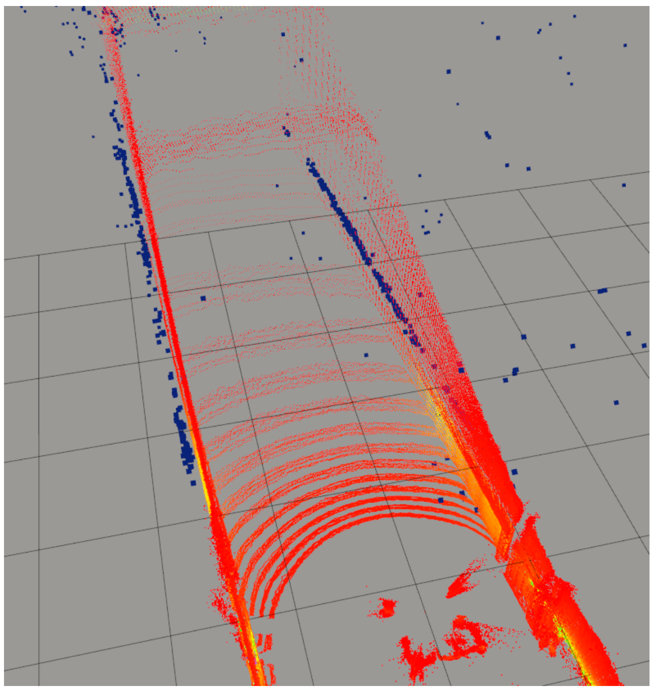


Figure 3.4: Comparison of scans from a 3D OS1 LiDAR (red) and a 4D ARS548 RADAR (blue). Extracted from [21]

and performed internally using a k-nearest neighbors (kNN) search. The user only needs to provide parameters regarding convergence criteria, and number of neighbours. For improved efficiency, the *small_gicp* library [28] provides optimized, parallelized implementations of GICP and other ICP variants that serve as drop-in replacements for the equivalent PCL classes. This is especially valuable in multi-robot localization or SLAM systems that operate in real time, where performance is critical.

3.3 Multi-robot pose optimization

In the previous Sections, UWB relative localization and RADAR odometry provided a source for inter-robot transformations and local position estimates for each robot, respectively. However, these estimates must somehow be integrated consistently over time for both robots. To overcome this challenge, factor graph optimization is a widely documented approach to localization and SLAM [24, 29] that offers a robust framework for fusing data from multiple sensor modalities in a global estimation problem. In what follows, the basic concepts behind factor graph localization are introduced, and a multi-robot extension is presented.

Solving a factor graph consists of maximizing the posterior density $p(\mathbf{x}|\mathbf{z})$ of states $\mathbf{x}$ given the relative measurements coming from observations $\mathbf{z}$. Through Bayes' theorem, the posterior can be expressed as the product of the measurement density $p(\mathbf{z}|\mathbf{x})$ and

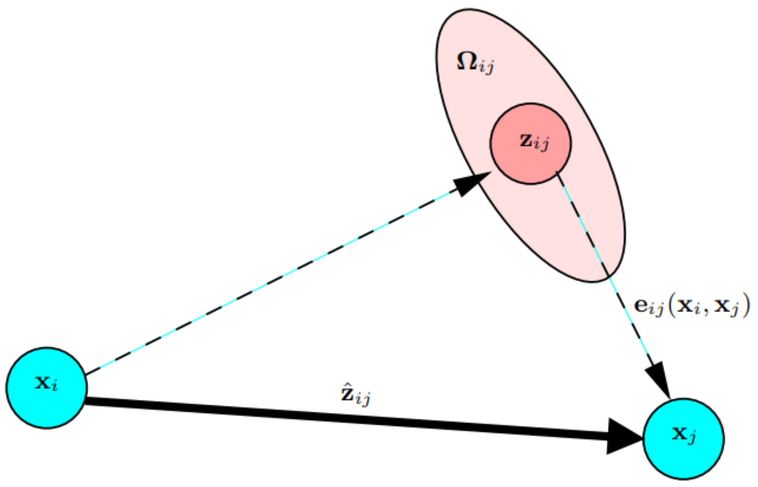


Figure 3.5: Basic elements of a pose-graph formulation: nodes, or robot positions (in blue) are joined by edges, which are predicted relative transformations which arise from observations coming from uncertain measurements (in red, where Ω is the associated information matrix). Extracted from [29].

.

the prior over the states $p(\mathbf{x})$, divided by a normalization term that can be dropped as it does not involve the states:

$$\mathbf{x}^{MAP} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) = \arg\max_{\mathbf{x}} \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} = \arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) \tag{3.8}$$

Multiple types of factor graphs are described in literature depending on the problem to be solved. In the general SLAM formulation, states $\mathbf{x}$ are both robot positions and landmarks in the map. Pose-graphs are a particular case of factor graphs where the states are uniquely robot positions, without landmarks. In pose-graph SLAM, robot poses are connected by loop closures. When there are no loop closures, the problem is reduced to pose-graph localization.

A pose-graph is composed of robot positions, which are represented as nodes, and observations coming from sensor measurements, which are represented as edges or factors (see Figure 3.5). In the case of a single robot, edges are often relative pose measurements that serve as constraints between two consecutive nodes, in the case of odometry, or two arbitrary nodes, in the case of loop closing in SLAM.

Even if real-world sensor noise is often complex, with the presence of outliers and multimodal distributions, a common assumption when solving a pose-graph SLAM or localization problem is that noise models are Gaussian. This is because, under this assumption, the problem can be formulated as nonlinear least squares of the form:

$$X^* = \arg\min_{\mathbf{x}} \sum_{i,j} ||\mathbf{z_{ij}} - \hat{\mathbf{z}}_{\mathbf{ij}}(\mathbf{x_i}, \mathbf{x_j})||^2_{\Sigma_i} = \arg\min_{\mathbf{x}} \sum_{i,j} ||\mathbf{e}(\hat{\mathbf{z}}_{\mathbf{ij}}, \mathbf{z_{ij}})||^2_{\Sigma_i} \tag{3.9}$$

Where $\mathbf{z_{ij}}$ is a relative measurement between two nodes and $\mathbf{\hat{z}_{ij}}(\mathbf{x_i}, \mathbf{x_j})$ is the predicted measurement for a configuration of those two nodes with covariance Σ_i. In many cases, including this work, these values correspond to relative transformations, and $\mathbf{e}(\mathbf{z_{ij}}, \mathbf{\hat{z}_{ij}})$ would be an error function that compares the prediction and the observation. To compensate for the errors that can be induced in the least-squares estimations due to the Gaussian assumption, outliers should be handled through robustified cost functions or consensus-based rejection methods like RANSAC.

Pose-graphs can be extended to multi-robot systems with a few considerations. In this work, we adopt the concept of encounters and anchor nodes introduced in [14]. Encounters are edges that connect the poses of two robots, and anchor nodes define the transformation between the local odometry frame of each robot and a global frame of reference - or, equivalently, they define where each robot starts their mission. In this way, robot poses are expressed in their own local frames, and encounter constraints involve the anchor nodes to express the inter-robot measurement in a global reference frame. A schematic representation of this framework is shown in Figure 3.6. To reduce the problem to 4-DOF we can make use of the error function $e_{4D}(.)$ presented in Equation (3.2), and detailed in Appendix C. This yields the following expression:

$$X^* = \arg\min_{\mathbf{x}} \sum_{W} \sum_{r=0}^{R-1} \Big(||e_{4D}(\mathbf{x_0^r}, \mathbf{p^r})||^2_{\Sigma_0} + \sum_{i=1}^{Nr-1} ||e_{4D}(\mathbf{\hat{z}_{i,i-1}}, \mathbf{z_{i,i-1}})||^2_{\Sigma_{i,i-1}} \Big) + \sum_{i,j \in C} ||e_{4D}(\mathbf{c_{ij}}, \mathbf{z_{ij}})||^2_{\Sigma_{ij}} \quad (3.10)$$

The first term anchors the first pose of each robot to a prior, which is commonly set at the origin. This is a common method to address gauge freedom - the fact that the trajectories are globally underconstrained by themselves, as they are composed of chaining relative transformations. The second term groups the odometry constraints between the consecutive poses of each robot along their respective trajectories, for a total of Nr relevant poses or keyframes. The third term refers to encounter constraints, where $\mathbf{c}$ is the predicted relative transformation between pose $\mathbf{x_i}$ of robot r, and pose $\mathbf{x_j}$ of robot r', as explained by the anchor nodes $(\mathbf{\Delta r}, \mathbf{\Delta r'})$. The general formulation states:

$$\mathbf{c_{ij}} = (\mathbf{\Delta r} \oplus \mathbf{x_i^r}) \ominus (\mathbf{\Delta r'} \oplus \mathbf{x_j^{r'}}) \quad (3.11)$$

Even then, the overall system still faces a gauge freedom that is addressed by adding an additional prior to at least one of the anchor nodes.

With this formulation, we can introduce multiple sources of sensor information to estimate the local poses of each robot (RADAR, IMU, wheel odometry) by deriving

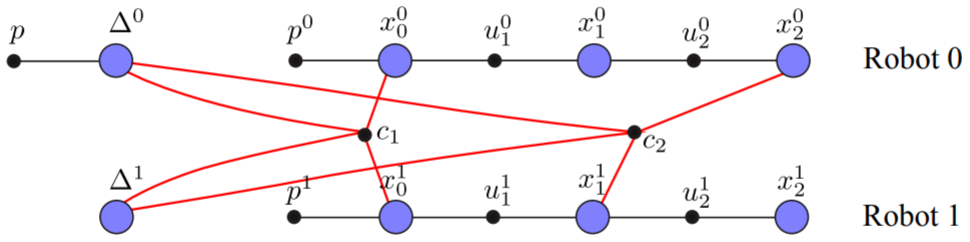


Figure 3.6: Multi-robot pose-graph formulation for two robots. Nodes are represented in blue, including robot positions $\mathbf{x}$ and anchor nodes Δ. Odometry constraints $\mathbf{u}$ connect consecutive robot nodes, while encounter constraints $\mathbf{c}$ connect the poses of different robots at any time. Priors $\mathbf{p}$ anchor each robot trajectory to their own local frame, and at least one of the anchor nodes to a global frame. Extracted from [14]

.

relative pose constraints between keyframes. Anchor nodes, on the other hand, handle inter-robot constraints in the graph, such as those coming from UWB relative estimation. Because robot poses are still expressed locally, each robot can build its own graph without seeing the others. Only when encounters are produced, global consistency is enforced.

3.4 Chapter recap

This chapter covered a theoretical analysis regarding the main components of the localization system: relative localization based on UWB ranging in Section 3.1, RADAR odometry in Section 3.2 and multi-robot pose-graph optimization in Section 3.3.

In what follows, the practical implementation of these concepts is presented, including the proposed software architecture, algorithms and flow diagrams of the developed solution.

Chapter 4

Implementation

In this Chapter, an overview of the developed localization system is presented. First, we provide a description of the software architecture as well as implementation details regarding inter-process communication and pseudo-code. This includes a practical overview of the ROS 2 nodes that were developed and how they communicate with each other. Finally, we provide an explanation of how these modules were tested and validated in experiments, both in simulated conditions and with the dataset. The full code is available in a public GitHub[1] repository.

4.1 Software description

The proposed solution was developed by taking a modular approach, according to the formulation in Chapter 3. This approach allows to test each component independently before integrating the full system, facilitating experimental validation. Additionally, the architecture was designed to be used indistinctly with real or simulated sensor information. A general schematic of the full system can be seen in Figure 4.1. The arrows represent the flow of data between nodes through ROS topics.

4.1.1 Relative transformation node

The first node of the pipeline implements the Relative Transform Estimation (RTE) introduced in Section 3.1. A pseudo-code of the implementation of this module is provided in Algorithm 1. This node takes the poses of each robot in their respective local frames by subscribing to their available odometry topics. It also takes pairwise anchor-tag range measurements from the Eliko RTLS. In order to make this information available through ROS 2 topics, we use the Eliko driver developed by the Service Robotics Lab of Pablo de Olavide University. Among other functionalities, the driver synchronizes and publishes all available range measurements at a given time using

[1]https://github.com/amartinezsilva/mr-radio-localization.git

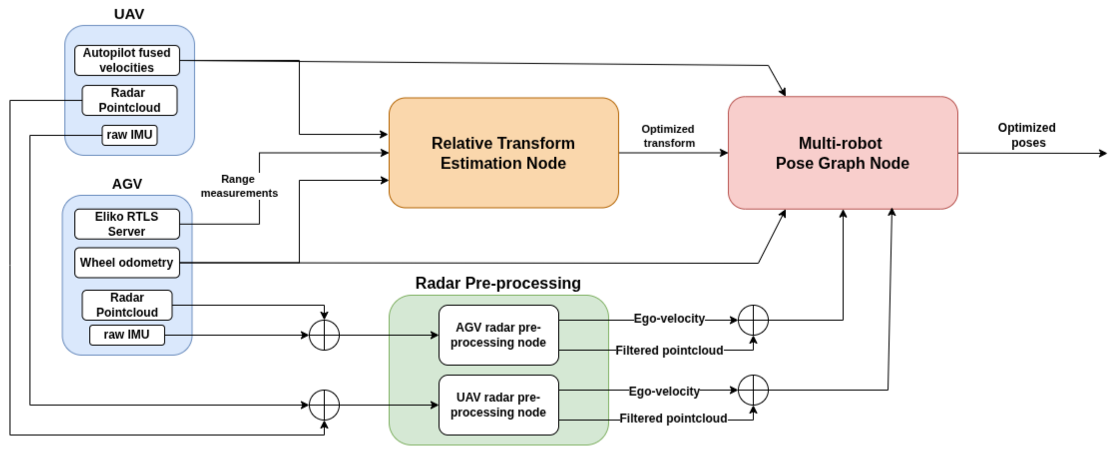


Figure 4.1: Overall architecture of the developed system. The symbol $\oplus$ is a graphic convention to simplify the diagram, and represents that both topics are used as input. For example, RADAR pre-processing subscribes to each robot IMU and RADAR pointcloud topics.

a custom ROS 2 message. The output is the estimated relative pose between the two robots as a homogeneous transformation matrix with an associated covariance, in a custom *PoseStampedWithCovariance* message. To do this, the node pre-processes input data by synchronizing the latest range measurements with the latest odometry. This step is necessary to compute the positions of the UWB transceivers in the local frame of each robot, which are used by the Ceres optimizer to compute the residuals. To ensure that the measurements provide enough observability for the estimated states, a distance-based measurement window W is employed. This window only includes measurements that comprise between two and ten meters of each robot trajectory. Measurements corresponding to situations where neither robot has moved are rejected in order to avoid accumulating highly correlated data.

The optimization runs on a 10 Hz timer, however, it is only performed under the following conditions: 1) that the robots have moved or rotated a minimal amount since the last optimization, 2) that the window contains enough sufficiently informative measurements (corresponding to different points of both robots trajectories, and coming from both tags).

4.1.2 RADAR odometry nodes

Section 3.2 discussed that RADAR odometry systems often require the raw point cloud to be filtered in order to remove dynamic objects. Additionally, the pre-processing step can be used to estimate the linear velocity of the platform on which the RADAR is mounted, by leveraging filtered pointclouds and IMU data. This step is performed in the RADAR pre-processing nodes. A pseudo-code of this implementation is provided

```
Algorithm 1: Relative Transform Estimation Node
  Inputs : Odometry poses ${}^{\mathbf{O_1}}\mathbf{p_1}, {}^{\mathbf{O_2}}\mathbf{p_2} \in SE(3)$;
           UWB range measurements $d(t_i, a_j)$;
           Fixed tag/anchor positions ${}^{\mathbf{B_1}}\tilde{\mathbf{p}}_{\mathbf{ti}}, {}^{\mathbf{B_2}}\tilde{\mathbf{p}}_{\mathbf{aj}}$;
           Parameters: initial conditions, motion threshold $\epsilon$, noise $\sigma$, $W_{size}$.
  Output: Estimated relative transform $T(\mathbf{x})$ and covariance $\Sigma$
1 Function RangeMeasurementsCallback ():
2     if both ${}^{\mathbf{O_1}}\mathbf{p_1}$ and ${}^{\mathbf{O_2}}\mathbf{p_2}$ are recent then
3         Compute ${}^{\mathbf{O_1}}\tilde{\mathbf{p}}_{\mathbf{ti}}$ and ${}^{\mathbf{O_2}}\tilde{\mathbf{p}}_{\mathbf{aj}}$;
4         Append $\left(d(t_i, a_j), {}^{\mathbf{O_1}}\tilde{\mathbf{p}}_{\mathbf{ti}}, {}^{\mathbf{O_2}}\tilde{\mathbf{p}}_{\mathbf{aj}}\right)$ to sliding window $W$;
5 Function OptimizationTimerCallback () ;    // Runs periodically (10
  Hz)
6 :
7     if $\|\Delta\mathbf{p}_1\| > \epsilon$ and $\|\Delta\mathbf{p}_2\| > \epsilon$ then
8         Prune outdated or distant measurements from window $W$;
9         if $|W|$ is sufficient then
10            (Optional) Initial Guess:;
11            Solve closed-form linear system to obtain prior pose $\mathbf{p}$;
12            Nonlinear Optimization (Ceres);
13            Solve for $\mathbf{x} = [x, y, z, \psi]^T$ by minimizing cost function $J(\mathbf{x})$ defined
               in Eq. (3.2);
14            if converged then
15                Estimate covariance $\Sigma$ via marginalization;
16                Apply moving average to $\mathbf{x}$;
17                Publish $T(\mathbf{x})$ as PoseWithCovarianceStamped;
```

in Algorithm 2. These nodes subscribe to the raw IMU data and RADAR Point Clouds of each platform through their respective topics, and produce a new, filtered pointcloud and an ego-motion estimation. The latter is published via a *Twist* message, where the angular information is set to zero. In order to communicate with the RADAR sensors through ROS 2, the ARS_548 driver [21] developed by the Service Robotics Lab of Pablo de Olavide University is used. This driver converts the RADAR detection into a ROS 2 *PointCloud2* message with Doppler velocity information.

This work uses the pre-processing presented in [25] which has been adapted to handle both robots at the same time. While the original work proposed a loosely coupled solution for holonomic and non-holonomic ground robots, this implementation is only valid for the AGV. In the case of the UAV, the original estimation by [11], which computes the three linear velocities, is used instead. Algorithm 2 provides an overview of the RADAR pre-processing algorithm.

Algorithm 2: RADAR Pre-processing Node

```
Input: RADAR point cloud $P$, IMU orientation quaternion $q_{\text{IMU}}$,
       transformation $T_{\text{RADAR}\to\text{imu}}$, robot_type (holonomic AGV or UAV),
       range/azimuth/elevation/RCS thresholds, inlier threshold $\epsilon$, median
       threshold $\tau$
Output: Ego velocity vector $\mathbf{v}_r$, filtered RADAR point cloud
1  IMU Callback:;
2      Update current orientation: $q_{\text{curr}} \leftarrow q_{\text{IMU}}$;
3  RADAR Callback:;
4      Transform all points $p_i \in P$ using $T_{\text{RADAR}\to\text{imu}}$;
5      Extract $\{\mathbf{x}_i, \text{doppler}_i, \text{RCS}_i\}$ fields from cloud;
6      Compute orientation change: $\Delta q$;
7      if $\textit{estimate}(P, \Delta q, \textit{robot\_type})$ succeeds then
8          Publish $\mathbf{v}_r$ as TwistWithCovarianceStamped msg;
9      Filter cloud with range and height thresholds;
10     Publish filtered cloud;
11 Function: estimate($P, \Delta q, \text{robot\_type}$);
12     Apply SNR, range, azimuth, and elevation filters $\Rightarrow$ valid targets $V$ with
         radial velocities $\mathbf{v}_D$ ;
13     if $\textit{median}(|\mathbf{v}_D|) < \tau$ then
14         Set $\mathbf{v}_r \leftarrow \mathbf{0}$;
15         return true;
16     else
17         Run RANSAC:;
18         foreach iteration do
19             Randomly sample $n$ targets from V with radial velocity $\mathbf{v}_D$;
20             Solve for $\mathbf{v}_r$ using Eq. (3.4), depending on robot_type;
21             Compute predicted radial velocity $\hat{\mathbf{v}}_D$ using Eq. (3.3);
22             Mark target $i$ as inlier if $|\mathbf{v}_D - \hat{\mathbf{v}}_D| < \epsilon$;
23         Solve for $\mathbf{v}_r$ using largest inlier set;
24         return true if inliers exist;
```

4.1.3 Pose-Graph Optimization node

The node at the end of the pipeline implements the multi-robot pose-graph optimization framework that was described theoretically in Section 3.3. This node expects inputs from all the previous nodes: namely, it subscribes to odometry from the AGV and the UAV, the optimized transform from the relative estimation node, and the filtered point clouds and ego-velocity estimations from the RADAR pre-processing nodes. The outputs are the optimized local trajectories of each robot as an array of poses, and the optimized anchors, which are transformations that map each of the local trajectories to a global reference frame. The graph is published in real time, and exported to a *.csv* file at the end of execution. Figure 4.2 shows the pose-graph architecture particularized to the AGV-UAV use case.

The nodes of each of the local pose-graphs are represented with a different color: blue for the UAV poses and green for the AGV poses. Each of the robots constructs its own pose-graph locally, adding a new keyframe after enough displacement has been detected through odometry observations. When a new keyframe is added, all the available measurements (in Figure 4.2, represented as ellipses with dashed lines) are synchronized to compute the relative constraints between the previous keyframes.

Constraints are represented with dashed lines joining the different nodes. There are four types: first, odometry constraints are relative transformations between consecutive nodes based on observations from wheel odometry or autopilot velocities. Secondly, ICP constraints are relative transformations resulting from scan-matching using the filtered RADAR pointclouds and the ego-velocity estimates. Notice that scan-matching is performed over a set of previous keyframes. This is specially useful when working with RADAR, as filtered pointclouds are very sparse and noisy, and comparing only with the previous keyframe may not provide a strong enough constraint. This approach considers temporal redundancy, which helps handle occlusions and spurious correspondences. Next, encounter constraints are created by the estimated relative transforms based on UWB measurements. Whenever a relative transform observation is available, the predicted observation involves the current active keyframes as well as the anchor nodes, and tries to align the nodes according to the relative transformation. As it was explained in Section 3.3, anchor nodes are variables that represent where each of the robot starts their mission with respect to a common reference (either provided by a map, or chosen arbitrarily in the case of SLAM), and provide global consistency to the pose-graph. In Figure 4.2, anchor nodes are represented with the letter Δ, and encounter constraints are highlighted in red. Finally, prior constraints are used to address gauge freedom in two ways, as discussed in Section 3.3: to fix the first local poses to a (local) known value $\mathbf{p_0}$, and to fix the AGV anchor to a (global) known value $\mathbf{p_\Delta}$.

Figure 4.3 shows a flow diagram of the process. At initialization, the priors for each of the trajectories are set, as well as the prior for one of the anchor nodes. Then, each robot adds its first keyframe, which is initialized with odometry values. After this, the optimization runs on a timer which executes its callback at a rate of 10 Hz. It manages the keyframes of each robot independently, and adds global encounter constraints whenever a new inter-robot transformation is estimated by the RTE node. Similarly to the RTE node, it only adds a new keyframe if enough motion has been detected (this can be inferred from any of the proprioceptive measurements which are

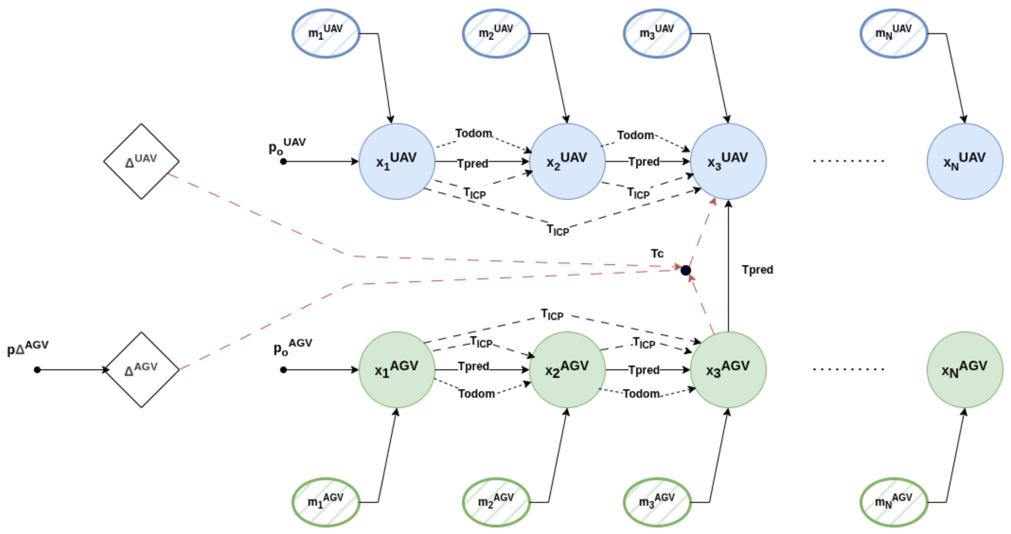


Figure 4.2: Multi-robot pose-graph design in the AGV-UAV scenario.

usually published at a stable rate, let it be wheel odometry, autopilot velocities, or raw IMU). In order to store the graph efficiently, only a short history of raw measurements is kept in memory. This way, constraints between nodes are stored as a structure containing the ids of the nodes that they connect, the relative transformation between them, and its associated covariance.

4.1.4 Simulation

A simulation framework was developed to test the functionality of the ROS 2 nodes described in 4.1.1 and 4.1.3. The simulator consists of two additional ROS nodes: an odometry simulator and a measurement simulator. Figure 4.4 shows the modified architecture with the new components. In what follows, their basic functionality is described:

– **Odometry simulator**: this node creates two trajectories as a sequence of linear and angular velocity commands for two robotic platforms, integrates them and publishes the corresponding poses consistently over time. The node aims to simulate a real odometry system by providing local pose estimates, like those coming from wheel encoders in the case of the AGV, or the autopilot in the case of the aircraft. It also models odometry drift over time, by injecting each pose with a bias that grows with the distance traveled, and a random zero-mean Gaussian noise that is added in a per-step basis. It takes as parameters the origin of each trajectory with respect to a common frame, the maximum and minimum allowed velocities and accelerations, and a factor to model odometry noise based on total distance and angle traveled. This node publishes both the ground truth positions of the robots, and the simulated odometry positions injected with noise.

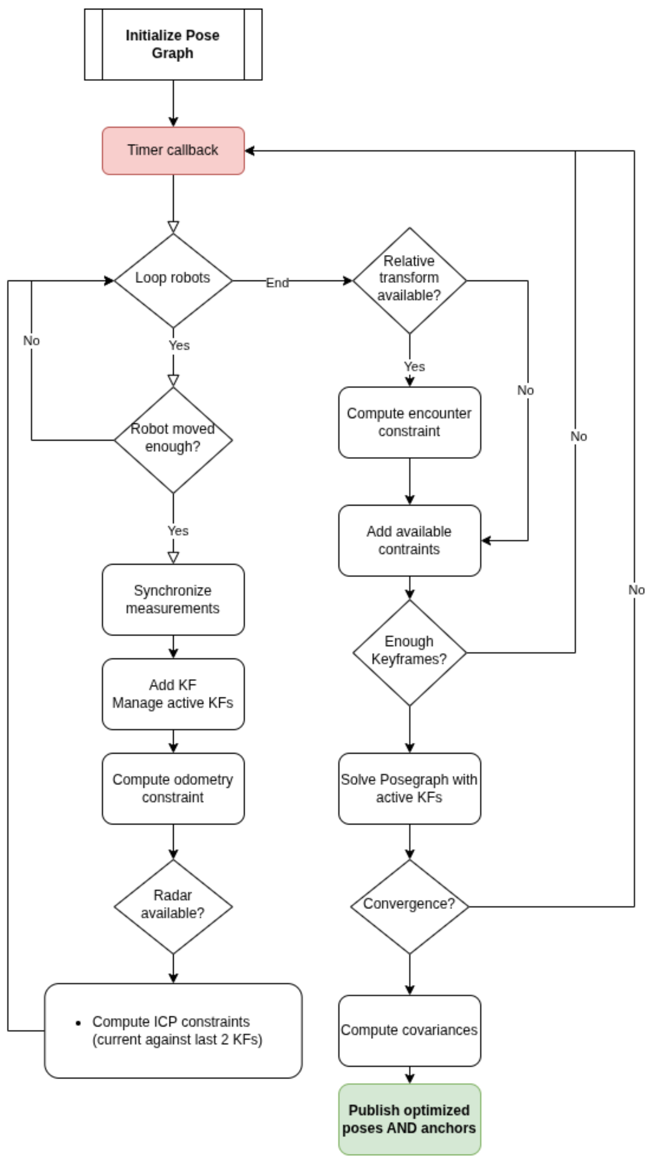


Figure 4.3: Flow diagram detailing the pose-graph optimization cycle

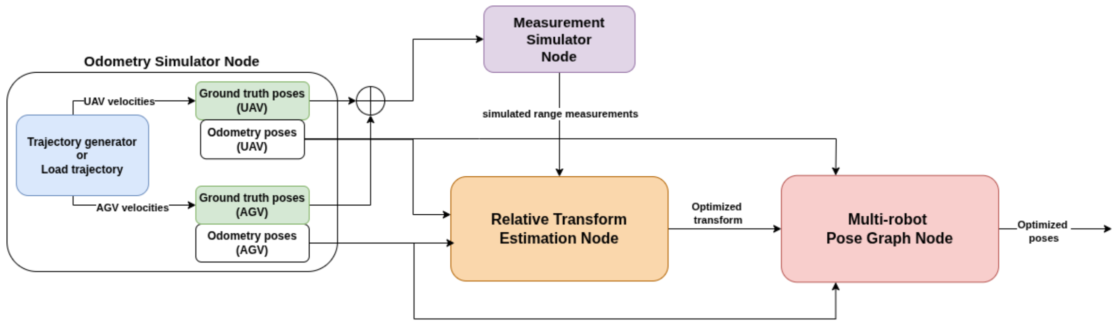


Figure 4.4: Architecture of the system in simulation conditions. Notice that ground truth poses are only used to generate the simulated measurements, while the estimation node takes odometry poses and range measurements, which is the information available to the robots.

– **Measurement simulator**: the purpose of this node is to simulate range measurements by using the ground truth relative transformation $\mathbf{T_{gt}}$ between the local odometry frames of the robots, which is available through the odometry simulator. By recovering the measurement model defined in Equation 3.1, we are able to create a simulated range measurement $d(t_i, a_j)$ between tag ti and anchor aj:

$$d(t_i, a_j) = ||{}^{\mathbf{O_1}}\tilde{\mathbf{p_{ti}}} - \mathbf{T_{gt}}{}^{\mathbf{O_2}}\tilde{\mathbf{p_{aj}}}|| + \epsilon \ \sim \mathcal{N}(0, \sigma^2) \tag{4.1}$$

The simulated odometry and range measurements can be used as inputs to the Relative Transformation Node as well as the Pose-Graph Optimization node without any changes to their design or functionality. This allows to easily switch between simulated and real inputs, which is especially valuable in experimentation. Finally, due to the complexity of accurately modeling RADAR measurements, the RADAR odometry module was not included in the simulated framework, and only tested with real measurements. Because of this, it was incorporated only after the rest of the components were verified both in simulation an in the dataset.

4.2 Chapter recap

This chapter covered an implementation overview of the proposed localization system, detailing the functionality of the different individual modules, which were implemented as ROS 2 nodes, and how they communicate with each other. Section 4.1.1 covered relative transformation estimation, Section 4.1.2 detailed the pre-processing of RADAR point clouds including ego-motion estimation, Section 4.1.3 presented the global pose optimization at the end of the pipeline, and finally, Section 4.1.4 showcased the architecture of the system adapted for simulation.

In what follows, experimental results, which were obtained using the proposed system, are presented and analyzed in both simulation and a real-world dataset.

Chapter 5

Experimental analysis

This Chapter presents and analyzes results from a set of experiments that were conducted to validate the designed multi-robot localization framework. The first two experiments correspond to the simulation framework, where the functionality of the relative transformation estimation node, as well as the pose-graph optimization node, are tested using virtual measurements. We generate two different set of trajectories that model two different scenarios of cooperative navigation. Then, results from a real world experiment are presented, where we validate the functionality of the RADAR odometry module using data captured from the ARS548 RADARs onboard both of the robots.

5.1 Experimental results on simulation

Two sets of trajectories were generated using the odometry simulator node (see Section 4.1.4) to cover different use cases of cooperative navigation between a ground vehicle and an aerial vehicle. In the first case (see Figure 5.1a), both robots describe a planar parametric trajectory in the shape of an '8' at a constant velocity of 0.5 m/s. Both robots start separated by a distance of 1 me in the horizontal plane and 2 m in the vertical plane, with a relative orientation of 30º in the vertical axis. The trajectories are considered to be completed when 100 m are covered by both of the robots, which results in a series of loops around the closed circuit. During this trajectory, the relative distance between the robots is expected to remain consistent with both platforms staying relatively close to each other. In this scenario, we assume that the odometry is accurate enough to not accumulate significant drift over time.

The second set of trajectories is generated randomly, and can be seen in Figure 5.1b. In this experiment, the robots start separated by 0.5 m in the horizontal plane, 2 m in the vertical plane, and $7°$ in orientation. The trajectories are also 100 m long but covered at variable velocities, using feasible dynamic constraints to generate realistic

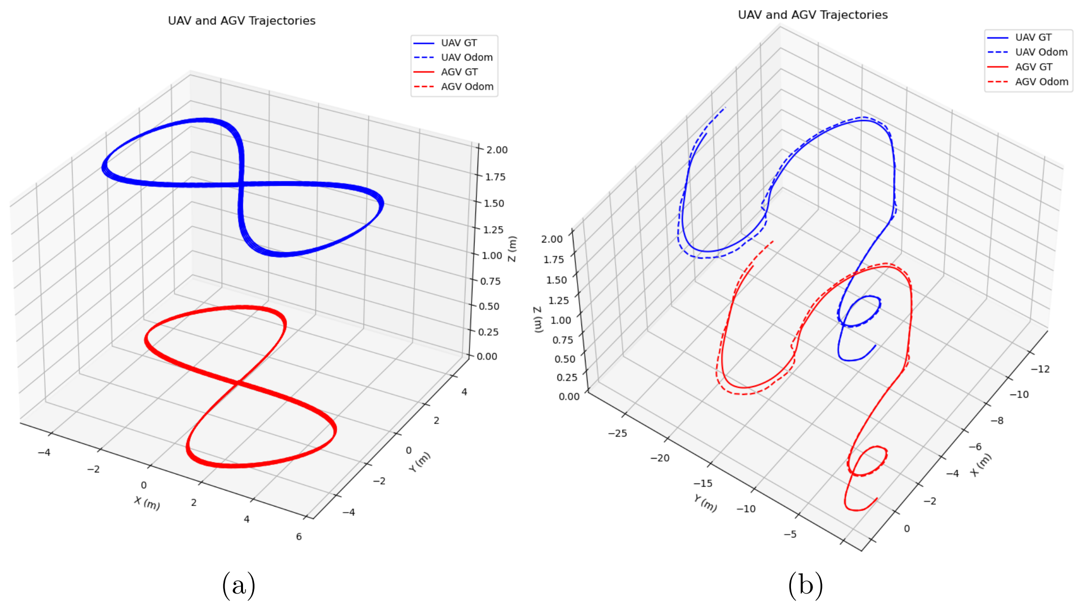


Figure 5.1: Two sets of simulated trajectories for the AGV (red) and the UAV (blue) generated for the experiments: (a) Parametric trajectories (b) Randomly generated trajectories with 2% odometry drift

curvatures. In this case, the yaw offset will make the robots progressively separate from each other. Additionally, we simulate a 2% odometry drift which is expected to degrade the quality of measurements over time. The purpose of this experiment is to illustrate the behavior of the system in a more realistic scenario, and to observe the impact of DoP in UWB-based relative localization, which will manifest as the robots get further away from each other.

In order to evaluate the algorithms, the Absolute Trajectory Error (ATE) is used to compute the error in translation, and the Root Mean Squared Error is used to compute the errors in the yaw angle. On the other hand, Ceres provides a 4x4 covariance matrix Σ_i for each estimated 4-DOF pose. To quantify the overall uncertainty of the estimated trajectory, the traces - sum of diagonal elements - are computed and then averaged. To give the result some physical meaning, the translation and yaw parts are computed separately. This way, for N estimated poses in a robot :

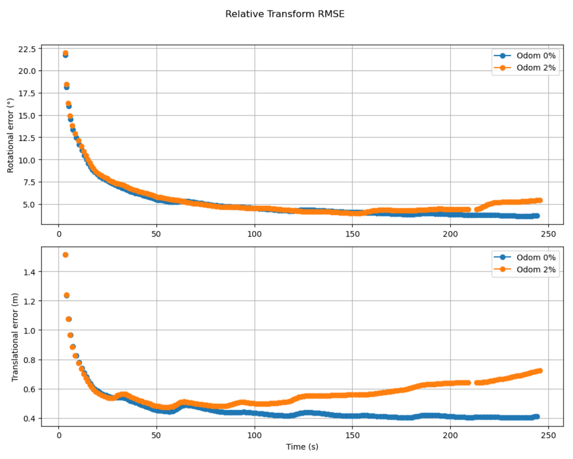
$$\begin{aligned} \mathrm{ATE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{p}_i^{\mathrm{est}}-\mathbf{p}_i^{\mathrm{gt}}\right\|^2}, \quad \bar{\sigma}^2_{\mathrm{xyz}} = \frac{1}{N}\sum_{i=1}^{N}\left(\sigma^2_{xx,i}+\sigma^2_{yy,i}+\sigma^2_{zz,i}\right) \\ \mathrm{RMSE}_{\mathrm{yaw}} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathrm{wrap}_{[-\pi,\pi)}\left(\psi_i^{\mathrm{est}}-\psi_i^{\mathrm{gt}}\right)\right)^2}, \quad \bar{\sigma}^2_{\psi} = \frac{1}{N}\sum_{i=1}^{N}\sigma^2_{\psi\psi,i} \end{aligned} \tag{5.1}$$

To evaluate the relative transformation computation, the RMSE of the angular and translational components of the transformation error are used: this is, the rotational and translational difference between the ground truth relative transformation and the estimated one, averaged over the whole trajectory. Figure 5.2a shows the transformation error in Experiment 1. Two cases are represented, when the odometry estimations are precise enough (each robot can estimate its own position without significant drift), and when they are not (in cases where only wheel odometry or IMU integration is available, which accumulate drift). In the first case, the error quickly converges from the initial solution (in this case, the Identity transformation) to very small values (≈ 0.4 cm in translation, and $\approx 3^\circ$ in rotation). In the presence of drift, once it becomes significant towards the middle of the trajectory, the error starts growing again, reaching nearly 0.8 cm in translation.

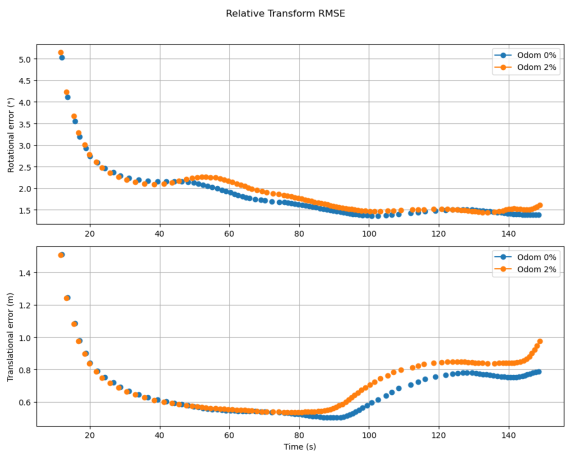
A 3D representation of the resulting optimized trajectories for Experiment 1 can be seen in Figure 5.3a. Alternatively, the final optimized global positions for the two robots can be seen in Figure 5.4, already transformed by the anchors. Notably, the most significant error is seen in the vertical (Z) component, which can be attributed to the following causes: (1) trajectories are planar with minimal relative motion in the vertical direction, and (2) very small vertical spread on the anchors ($\approx 0.5m$) and no vertical separation on the tags lead to almost identical vertical components of the anchor-tag vectors when the UAV maintains a fixed altitude of 2 m above the AGV. These conditions lead to poor observability of the vertical direction. However, this behavior is expected, and a direct consequence of DoP and the weak coupling between the measurement model and the Z-component of the relative transformation.

Regarding the second set of trajectories, Figure 5.2b shows the RMSE of the angular and translational components of the relative transformation error, with and without drift. In the first half of the trajectories, where the robots remain close together and odometry estimates are accurate enough, the behavior is similar to Experiment 1. In contrast, from the middle point onward, both cases start to accumulate a significant error predominantly in translation, which grows faster upon noisy odometry. In both cases, the large robot separation makes it difficult for the estimator to converge due to high DoP in the later parts of the trajectories. This effect is not seen in rotation, which is probably due to the initial solution being very close to the real yaw offset imposed on the trajectories.

The 3D representation of the optimized trajectories can be seen in 5.3b and the temporal evolution of the states is shown in Figure 5.5. While odometry drift degrades the quality of the estimated trajectories in the four components, the divergence is again predominant in the vertical direction, now leading to significant errors (>0.5 m). This is due to the same phenomenon that was noticed in Experiment 1, in this case potentiated by bad odometry estimates and higher DoP.

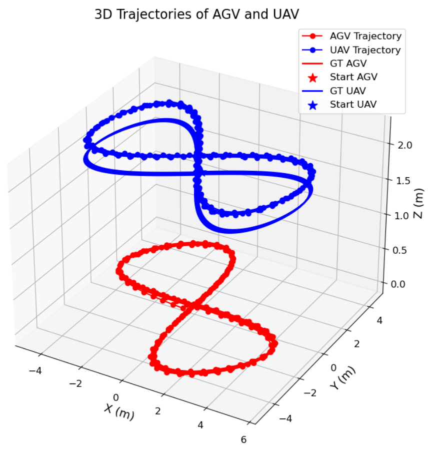


(a)

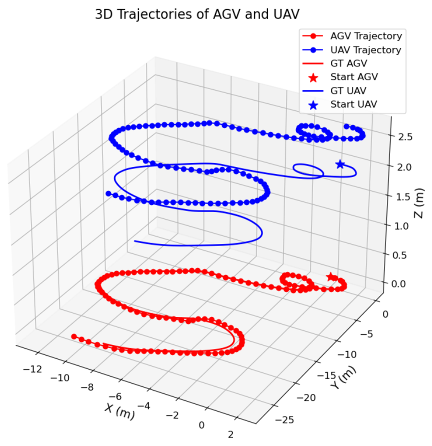


(b)

Figure 5.2: Relative Transformation RMSE over time, with and without odometry drift. (a) Experiment 1 (parametric trajectory) (b) Experiment 2 (random trajectory)



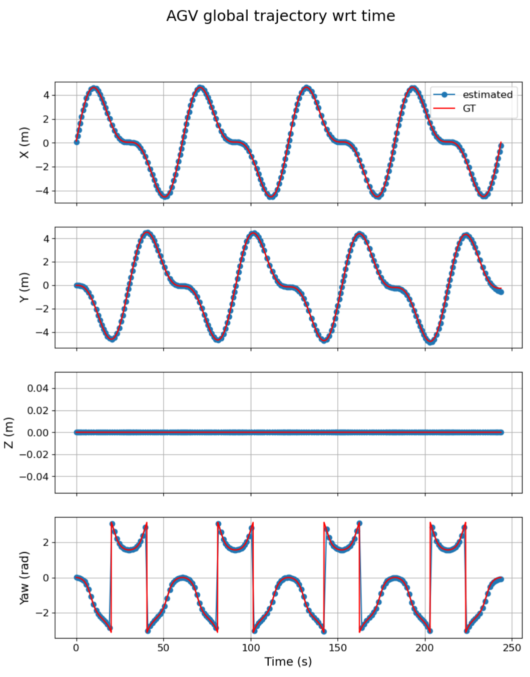
(a)



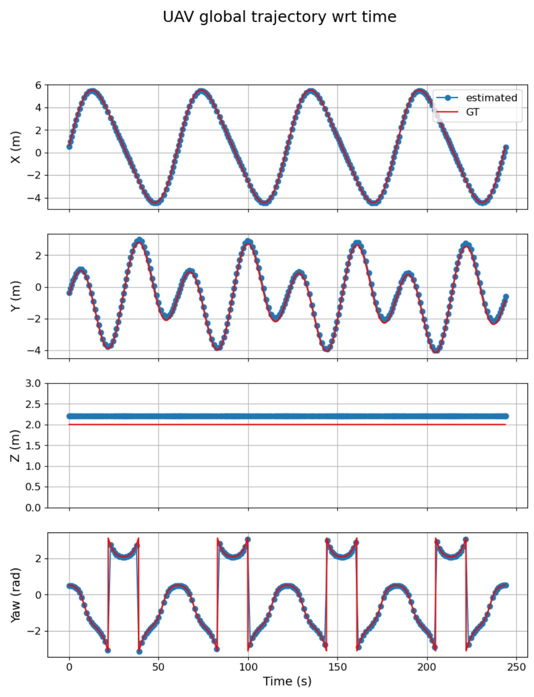
(b)

Figure 5.3: Optimized 3D poses of the AGV(red) and the UAV(blue) after pose-graph optimization: (a) Parametric trajectories (b) Randomly generated trajectories with 2% odometry drift

Table 5.1 shows quantitative results of the pose-graph experiments. Errors on the AGV are significantly lower, which is due to the fact that the global reference is set to coincide with the local origin of the AGV trajectory, with the UAV anchor carrying the error of the relative transformation. However, in Trajectory 1, after 100 m of trajectory, we are able to obtain errors as low as 0.27 m in translation and 1.2º in rotation using precise odometry. In the presence of drift, we manage an error as low as 1.23 m in translation and 13.4° in rotation in Trajectory 2. Finally, regarding the influence of the linear prior, while it contributed to a 15 % decrease in error under perfect odometry conditions (Trajectory 1), it did not impact the results in any noticeable way under noisy odometry (Trajectory 2) with respect to error or convergence times.

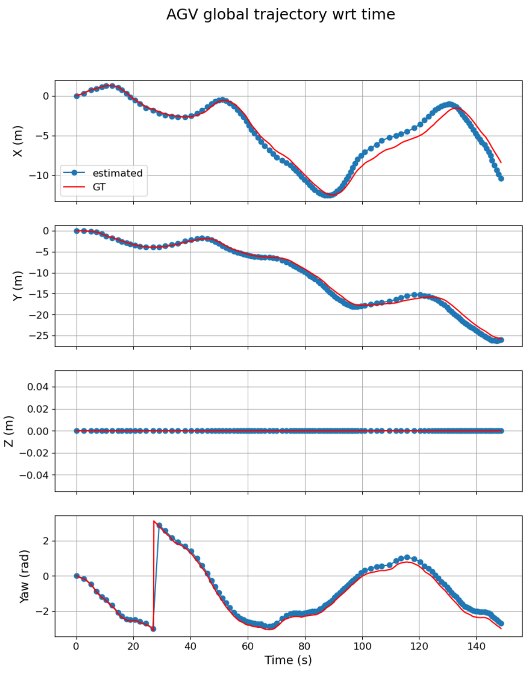


(a)

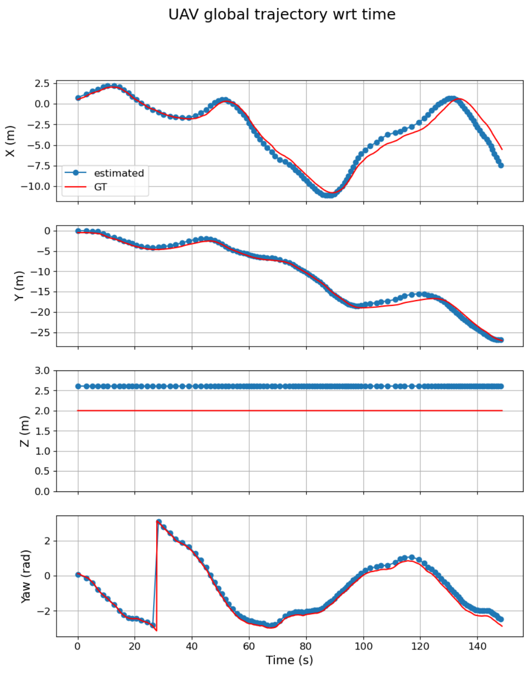


(b)

Figure 5.4: Temporal evolution of the estimated states in Experiment 1 (a) AGV (b) UAV



(a)



(b)

Figure 5.5: Temporal evolution of the estimated states in Experiment 2 (a) AGV (b) UAV

5.2 Experimental results on real data

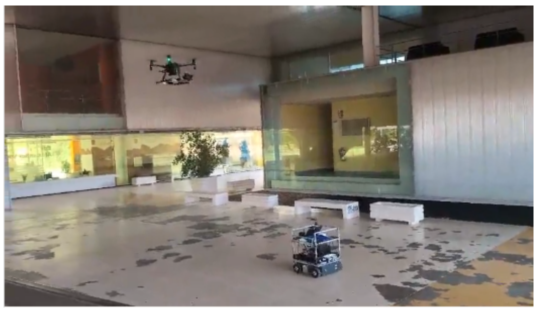
5.2.1 Dataset description

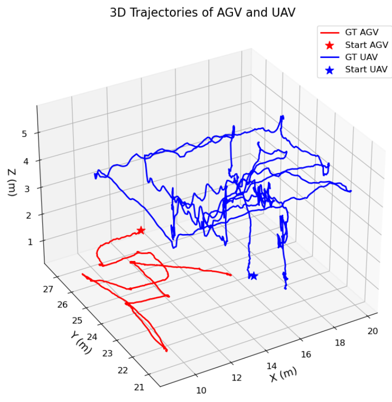
The dataset experiment was conducted at building 45 in Pablo de Olavide University (see Figure 5.6a). Figure 5.6b shows the reference trajectories being used for comparison, estimated via Direct LiDAR Localization (DLL)[30] using 3D LiDARs onboard both robots. The AGV trajectory spans 30 m in the horizontal plane and includes frequent fast heading changes, while the UAV trajectory covers 102 m in 3D space with overall smoother motions. Although out of the scope of this work, it is worth noting that DLL relies on pointcloud-to-map registration for position tracking and is quite sensitive to LiDAR scan quality - making it less precise than other systems like Optitrack motion capture. When evaluating these references, it was noticed that, while the UAV reference proved consistent when checked against the map, the AGV reference showed some issues due to fast rotations and lack of informative range data X direction of the map, which is a hallway. Conversely, in the UAV case, this issue does not manifest as much, as the robot is facing a nearby wall in this direction during most of the trajectory (as seen in Figure 5.6a), and performs overall smoother motions. These issues help explain the larger errors observed with respect to the method proposed, especially towards the end of the trajectories.

5.2.2 Results

Figure 5.7a shows the estimated relative transformation converging to the ground truth solution, and Figure 5.7b indicates the rotational and translational components of the RMSE of the estimated transformation. After nearly 20 seconds of trajectory, when both of the robots have advanced enough, the error quickly converges to ≈ 1.1 m in translation and $\approx 5°$ in rotation. Considering measurement error, which can comprise from 10 to 20 cm at these operating ranges, and odometry error (in this case, only wheel and IMU odometry are being fed to the optimizer, both of which tend to accumulate significant drift), the results are favorable and consistent with what was obtained in simulation. Additionally, the linear method from [6] was implemented but RANSAC failed to find a prior with enough consensus. Consequently, even in the absence of a good initial solution, the proposed method is robust enough to compute a decent estimate in translation and yaw, given that measurements are sufficiently informative in terms of relative motion. Finally, Appendix A includes an additional representation of raw range data gathered during the experiment, as compared with the estimated and ground truth inter-robot distances.

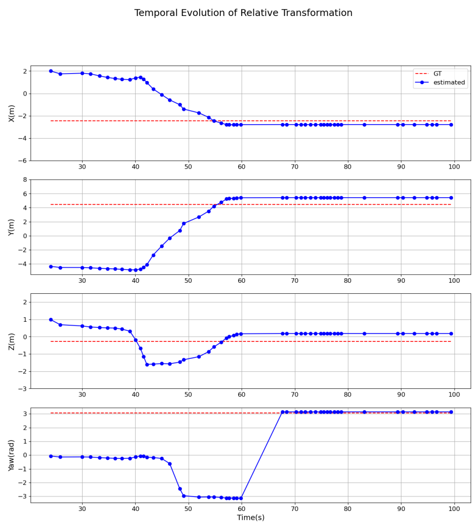
In Figure 5.8, the linear velocities estimated by the RADAR ego-motion algorithm


(a)

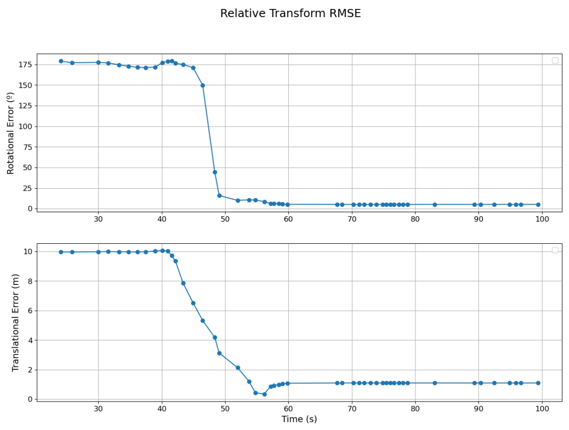


(b)

Figure 5.6: (a) Real experiment taking place in building 45 of Pablo de Olavide University (b) Reference trajectories of the AGV and UAV, estimated with Direct LiDAR Localization. A star marks the beginning of each trajectory.



(a)



(b)

Figure 5.7: (a) Temporal evolution of the estimated relative transform with predicted motion uncertainty. (b) RMSE of the estimated transform with respect to ground truth from DLL, showing rotational and translational components.

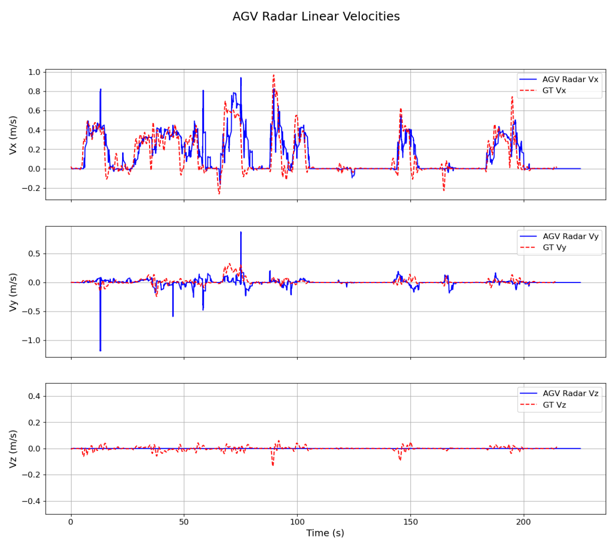
are represented, after being smoothed with a 10 sample moving average filter. Ground truth linear velocities are estimated by computing the derivative of the DLL pose estimation, and similarly smoothing the result. In this experiment, despite its holonomic configuration, the robot moves predominantly in the forward direction, where the RADAR is able to capture significant changes in velocity ($\approx$ 0.5 m/s). In the Y axis, while there is some movement in the early part of the trajectory, the RADAR is not able to capture subtle changes in velocity ($\leq$ 0.2 m/s). The UAV case, which proves more challenging as there is individual motion in the three directions, the method is also able to provide a consistent estimate. In this case, the ground truth estimation is sufficiently accurate and it is possible to compare directly against it.

Figure 5.9 shows the final optimized trajectories of the UAV and the AGV in 3D, and Figure 5.10 shows the four optimized states of both trajectories with respect to time. The AGV trajectory is anchored to the map while the UAV trajectory is not, which means the relative localization error is manifested on this side. This is consistent with Figure 5.7a, where the bias in the Y direction matches the errors found in the relative transformation (≈ 1.1 m). Additionally, it was found that RADAR scan-matching frequently produced inconsistent translations in the vertical direction that contaminated the estimation, which was compensated by setting a higher variance in this direction to the relative transformation constraint computed by GICP. This is a well-known issue in RADAR-based systems due to the low vertical resolution of the sensor [25], and is most notably seen in the aerial vehicle trajectory. Conversely, on the AGV side, the error is mostly manifested in the direction of the hallway, especially when the robot rotates in place. Table 5.1 shows the global trajectory error and average uncertainty of the system.

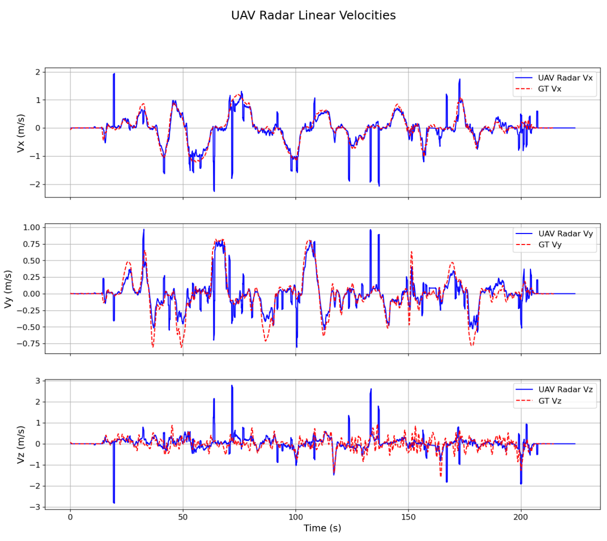
Finally, performance metrics for all optimization experiments are provided in Table 5.2. All experiments were performed in real time from a stream of data stored in a .bag file, and run on a Gigabyte AORUS 5 KE laptop with a 12th Gen Intel® Core™ i7-12700H CPU and 32GB of RAM. In both cases, the optimization is parallelized with four threads. The temporal metric is provided by Ceres and corresponds to the total time spent in the solver. The relative transformation estimation is overall less demanding due to mostly scalar residuals, can be performed in 200 ms with a consistent variance. Pose-graph optimization shows more variance across experiments, which can be attributed to the variable number of 4D residuals being included in the optimization problem depending on the availability of sensor measurements. However, the average time per solve in the case of the dataset is very similar for both optimization problems, and imply that the system would be able to run at rates from 5 to 10 Hz, which are quite acceptable for real-time localization.

5.3 Chapter recap

This chapter presented and evaluated results obtained by testing the proposed localization system across a set of scenarios comprising simulation, in Section 5.1, and a real-world experiment in Section 5.2. In the next chapter, this thesis is concluded with some final remarks as well as promising lines for future work.

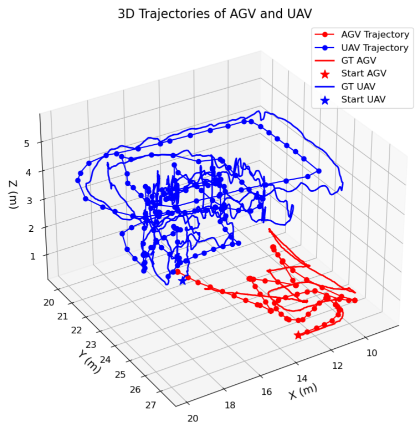


(a)

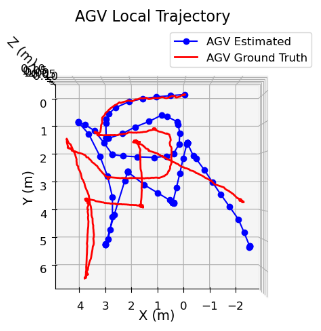


(b)

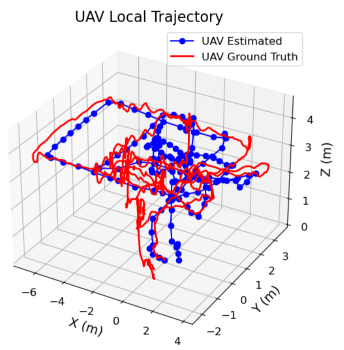
Figure 5.8: RADAR-estimated velocities as compared with DLL velocities in the AGV (a) and the UAV (b)



(a)

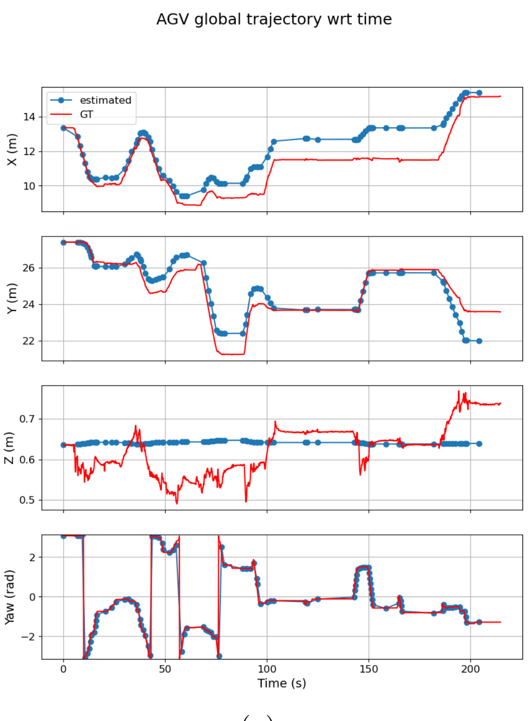


(b)



(c)

Figure 5.9: (a) 3D representation of the global optimized poses with anchor transformation, (b) and (c): Optimized local AGV and UAV trajectories, respectively, starting from the origin without anchor transformation.



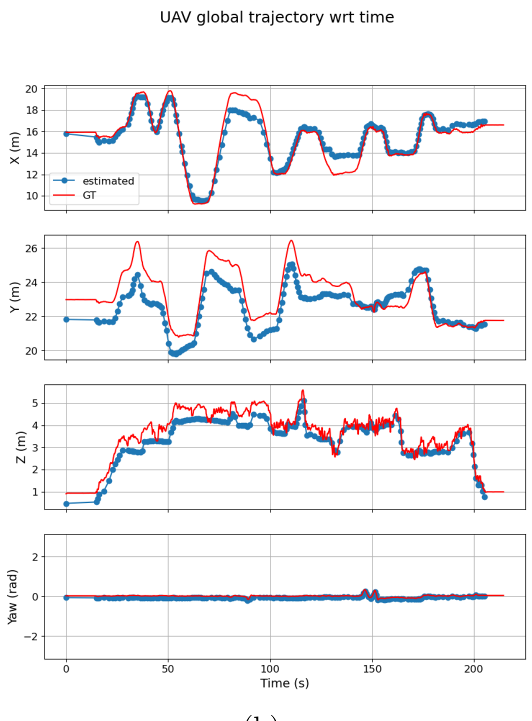


Figure 5.10: Estimated global positions and yaw orientation with respect to time in the AGV case (a) and the UAV (b), already transformed by the anchors, as compared with ground truth. Both trajectories are globally anchored to the AGV origin, which means the relative transformation error is manifested in the UAV trajectory.

Table 5.1: ATE and average uncertainty (trace of covariance) after pose-graph optimization

Experiment	Robot	ATE (m/°)	$\bar{\sigma}^2$ (m^2 / rad^2)
Trajectory 1 (no drift)	UAV	0.27 / 1.2	0.028 / 0.009
	AGV	0.07 / 1.3	0.028 / 0.009
Trajectory 2 (2% drift)	UAV	1.28 / 12.0	0.215 / 0.016
	AGV	1.22 / 13.4	0.154 / 0.021
Dataset	UAV	1.28 / 4.50	0.080 / 0.023
	AGV	1.28 / 9.87	0.28 / 0.09

Table 5.2: Optimization performance for the Relative Transform Estimation (RTE) and Pose-Graph optimization problems (avg. runtime ± std. dev, in s), with number of runs in parentheses

Experiment	RTE	Pose-Graph
Trajectory 1	0.287 ± 0.094 (264)	1.076 ± 0.807 (120)
Trajectory 2	0.282 ± 0.108 (102)	0.695 ± 0.710 (127)
Dataset	0.140 ± 0.141 (50)	0.118 ± 0.238 (226)

Chapter 6

Conclusions

This work showed the potential of radio-based localization methods in GPS-denied environments where LiDAR and vision may fail - e.g., in poorly lit scenes or in places with harsh weather conditions. A multi-robot cooperative localization framework using UWB and RADAR was proposed to localize a team of a ground platform and an aerial platform. UWB was used for inter-robot localization by aligning the trajectories using short-range anchor-tag measurements gathered along over a few meters of trajectory. With a configuration of two tags on the aerial platform and four anchors on the ground robot, both translation and yaw were reliably estimated even without an initial guess. However, performance degraded at longer distances due to limited transceiver geometry and DoP. On the other hand, RADAR odometry estimates were obtained via ego-motion estimation and scan-matching. Even though it requires careful pre-processing and outlier rejection due to the sparsity and multi-modal noise of RADAR scans, per-point Doppler velocity estimates proved useful to obtain a consistent estimation of the robot linear velocity, which can in turn be used as an initial solution for scan-matching.

This information was then incorporated into a multi-robot pose-graph optimization as relative pose constraints, and fused with proprioceptive measurements which tend to accumulate drift over time. Finally, the relative pose-graph formulation allowed to introduce global consistency to the overall problem with the inclusion of two anchor nodes and a new type of encounter constraint that aligns the robot trajectories whenever they are seeing each other, or sharing a view of the scene. Simulation and dataset results proved the robustness of relative localization and RADAR ego-motion estimation, which allowed to track long, complex trajectories of two different robots with relative localization errors of approximately 1.3 % in the most complex case, which is the UAV, even in the presence of noisy odometry.

6.1 Future work

This work addresses a localization problem, and this means that there are no explicit place recognition or loop closing strategies in effect. While the fusion of different sensor modalities and inter-robot measurements across a sliding window of keyframes help reduce the accumulation of uncertainty over time, it is still expected to grow as the robots explore the environment.

With this in mind, the main promising line of work becomes extending this system to a full multi-robot, radio-based SLAM system. Adopting a factor graph formulation facilitates this, as pose-graph localization can be considered a simplified version of pose-graph SLAM, where only odometry constraints are being considered. In this context, RADAR place recognition would be a major challenge due to the small FoV of the sensors, and the low resolution and visual aliasing induced by noise in RADAR scans. The small FoV also means that the same place will appear very different when looked at from different positions. To address this, learning-based approaches are already being used to obtain meaningful embeddings of RADAR scans for place recognition by leveraging the filtered static scan resulting from ego-velocity estimation [31]. In a multi-robot scenario, encounters would enable loop closures not only when a robot revisits a location, but also when any of the robots sees a location previously seen by others. Finally, while UWB can't be used for place recognition, it can still serve its original purpose by providing a global alignment constraint between the robot trajectories.

Bibliography

[1] IEEE Spectrum. Video friday: CMU DARPA subterranean challenge, 2021.

[2] Caroline Rees. Hazard detection & rescue quadruped robot released, 2022.

[3] Eliko Location Technologies. Eliko – next-generation location tracking technology, 2025.

[4] Nikolas Trawny, Xun S Zhou, Ke Zhou, and Stergios I Roumeliotis. Interrobot transformations in 3-d. *IEEE Transactions on Robotics*, 26(2):226–243, 2010.

[5] Nikolas Trawny, Xun S Zhou, and Stergios I Roumeliotis. 3d relative pose estimation from six distances. In *Robotics: Science and systems*, 2009.

[6] Francisco Molina Martel, Juri Sidorenko, Christoph Bodensteiner, Michael Arens, and Urs Hugentobler. Unique 4-dof relative pose estimation with six distances for uwb/v-slam-based devices. *Sensors*, 19(20):4366, 2019.

[7] Thien Hoang Nguyen and Lihua Xie. Relative transformation estimation based on fusion of odometry and uwb ranging data. *IEEE Transactions on Robotics*, 39(4):2861–2877, 2023.

[8] Zhiqiang Cao, Ran Liu, Chau Yuen, Achala Athukorala, Benny Kai Kiat Ng, Muraleetharan Mathanraj, and U-Xuan Tan. Relative localization of mobile robots with multiple ultra-wideband ranging measurements. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5857–5863. IEEE, 2021.

[9] Sarah H Cen and Paul Newman. Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6045–6052. IEEE, 2018.

[10] Martin Holder, Sven Hellwig, and Hermann Winner. Real-time pose graph slam based on radar. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1145–1151. IEEE, 2019.

[11] Dominik Kellner, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Klaus Dietmayer. Instantaneous ego-motion estimation using doppler radar. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 869–874. IEEE, 2013.

[12] Jun Zhang, Huayang Zhuge, Zhenyu Wu, Guohao Peng, Mingxing Wen, Yiyao Liu, and Danwei Wang. 4dradarslam: A 4d imaging radar slam system for large-scale environments based on pose graph optimization. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8333–8340. IEEE, 2023.

[13] David Alejo, Rafael Rey, José Antonio Cobano, Fernando Caballero, and Luis Merino. Data fusion of radar and lidar for robot localization under low-visibility conditions in structured environments. In *Iberian Robotics conference*, pages 301–313. Springer, 2022.

[14] Been Kim, Michael Kaess, Luke Fletcher, John Leonard, Abraham Bachrach, Nicholas Roy, and Seth Teller. Multiple relative pose graphs for robust cooperative mapping. In *2010 IEEE International Conference on Robotics and Automation*, pages 3185–3192. IEEE, 2010.

[15] Isaac Deutsch, Ming Liu, and Roland Siegwart. A framework for multi-robot pose graph slam. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 567–572. IEEE, 2016.

[16] Vadim Indelman, Erik Nelson, Nathan Michael, and Frank Dellaert. Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 593–600. IEEE, 2014.

[17] Continental Engineering Services. Ars 548 rdi radar sensor, 2025.

[18] Universidad Pablo de Olavide Service Robotics Lab. Arco dataset: Ros2 datasets for localization and mapping algorithms, 2023.

[19] DJI. *MATRICE 200 SERIES V2 User Manual*. DJI, v1.2 edition, April 2019. Accessed: 2025-05-03.

[20] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics*, 7(66):eabm6074, 2022.

[21] Fernando Fernández-Calatayud, Lucía Coto, David Alejo, José Javier Carpio, Fernando Caballero, and Luis Merino. ars548_ros: An ars 548 rdi radar driver for ros. *SoftwareX*, 30:102111, 2025.

[22] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 10 2023.

[23] Ceres Solver Team. Modeling non-linear least squares, 2025. Accessed: 2025-06-21.

[24] Luca Carlone, Ayoung Kim, Frank Dellaert, Timothy Barfoot, and Daniel Cremers. *SLAM Handbook. From Localization and Mapping to Spatial Intelligence.* Cambridge University Press.

[25] Lucia Coto Elena, Fernando Caballero, and Luis Merino. Loosely coupled 4d-radar-inertial odometry for ground robots. *arXiv preprint arXiv:2411.17289*, 2024.

[26] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[27] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[28] Kenji Koide. small_gicp: Efficient and parallel algorithms for point cloud registration. *Journal of Open Source Software*, 9(100):6948, August 2024.

[29] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[30] Fernando Caballero and Luis Merino. Dll: Direct lidar localization. a map-based localization approach for aerial robots. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5491–5498. IEEE, 2021.

[31] Guohao Peng, Heshan Li, Yangyang Zhao, Jun Zhang, Zhenyu Wu, Pengyu Zheng, and Danwei Wang. Transloc4d: Transformer-based 4d radar place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17595–17605, 2024.

[32] HAGC Premachandra, Ran Liu, Chau Yuen, and U-Xuan Tan. Uwb radar slam: An anchorless approach in vision denied indoor environments. *IEEE Robotics and Automation Letters*, 8(9):5299–5306, 2023.

[33] Elizabeth R Boroson, Robert Hewitt, Nora Ayanian, and Jean-Pierre de la Croix. Inter-robot range measurements in pose graph optimization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4806–4813. IEEE, 2020.

[34] Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3(6):1, 2010.

[35] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[36] MapScaping. Triangulation vs trilateration – what's the difference?, 2021. Accessed: 2025-06-14.

List of Figures

List of Tables

List of Acronyms

List of acronyms.

Abbreviation	Definition
AGV	Autonomous Ground Vehicle
ATE	Absolute Trajectory Error
CPU	Central Processing Unit
DOF	Degree Of Freedom
DoP	Dilution of Precision
FMCW	Frequency-Modulated Continuous Wave
FoV	Field of View
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
NLS	Nonlinear Least Squares
RADAR	RAdio Detection And Ranging
RF	Radio Frequency
RFT	Relative Frame Transformation
RTE	Relative Transform Estimation
RMSE	Root Mean Squared Error
ROS	Robot Operating System
SE(3)	Special Euclidean Group in 3 dimensions
SLAM	Simultaneous Localization And Mapping
ToF	Time of Flight
UAV	Unmanned Aerial Vehicle
UWB	Ultra-Wideband

Appendix

Appendix A

Anchor-tag placement and raw range data

This appendix presents additional information regarding the experimental setup for the real-world scenario showcased in the dataset that is being used as reference in this work, and also replicated in simulation. Figure A.1 highlights the location of the relevant sensors onboard both of the robots, including the four UWB anchors on the ARCO platform and the two UWB tags on the UAV platform, as well as the two ARS548RDI RADARs. Notice that the RADAR in the aerial platform is oriented backwards and tilted to point below the robot, while the RADAR on the ARCO points forward in the same direction as the robot.

On the other hand, Figure A.2 shows, in black, the range measurements reported by the UWB transceivers during the experiment described in Section 5.2. In red, the inter-robot range is represented as inferred from DLL ground truth, and in blue, the range resulting from the relative transformation estimated in this work. The initial values show the optimized transformation converging from the identity to the solution, which closely matches both the observations and ground truth. Additionally, it shows a period when the transceivers stop reporting ranges when the robots get too far apart (over 10 m).

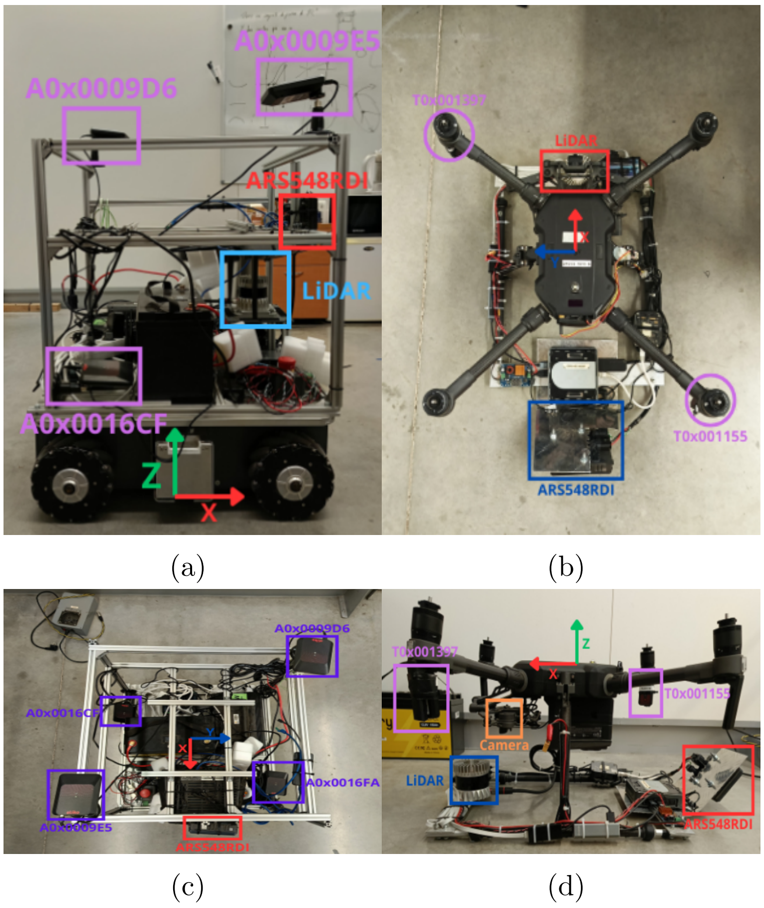


Figure A.1: Left column: overview of the ARCO platform and sensor setup. Right column: overview of the M210 platform and sensor setup. The purple identifiers refer to the IDs of the onboard tags on the UAV and anchors on the AGV. The RADARs are highlighted in red.

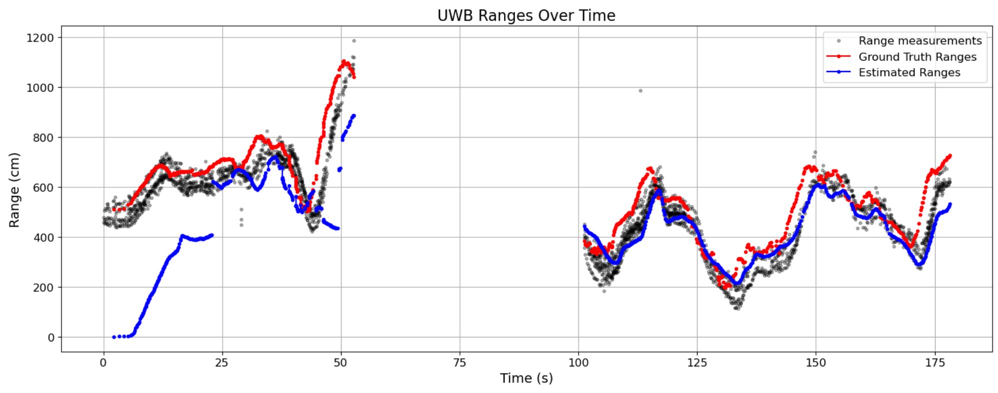


Figure A.2: Comparison of all pairwise range measurements obtained in the experiment (black) with the ground truth inter-robot distance estimated by DLL (red) and the presented method (blue). The captured data includes a segment where connection is temporarily lost due to the transceivers going out of range. Additionally, one of the tags fails to connect in the second part of the trajectory, which results in non-observable yaw.

Appendix B

Basics of trilateration

Range-based localization methods, including Ultra-Wideband (UWB) localization, rely on the principle of trilateration. Trilateration is the geometric process of determining the position of an unknown point of interest in space (in this case, the tag), based on the distances or ranges to reference points whose positions are known (in this case, the anchors). Suppose we have four anchors a_i, i $\in \{1, 4\}$ distributed in 3D space, and we want to know the position of a tag t. Each anchor can measure its distance d_i to the tag. If we assume the measurements have no error, this means that:

– With only one anchor a_1, the tag can be anywhere on the surface of a sphere centered at a_1 with radius d_1.

– Adding another anchor a_2 and its corresponding range d_2 reduces the possible positions of the tag to the intersection of the two spheres (a circle).

– Introducing a third anchor a_3 and its distance d_3 restricts the possible tag locations to two points in space.

– Finally, with a fourth anchor a_4 and its distance d_4 the location of the tag is finally obtained.

Figure B.1 shows a graphical representation of this example. In the real world, however, the problem becomes more complex as range measurements have an uncertainty associated to them. This uncertainty is influenced by sensor specifications and geometry (namely, DoP, which is described in Section 3.1.1), which means that adding more measurements is usually required to reduce the area of uncertainty.

This method is often confused with triangulation, which uses angles instead of distances. For more information on the differences between the two, refer to [36].

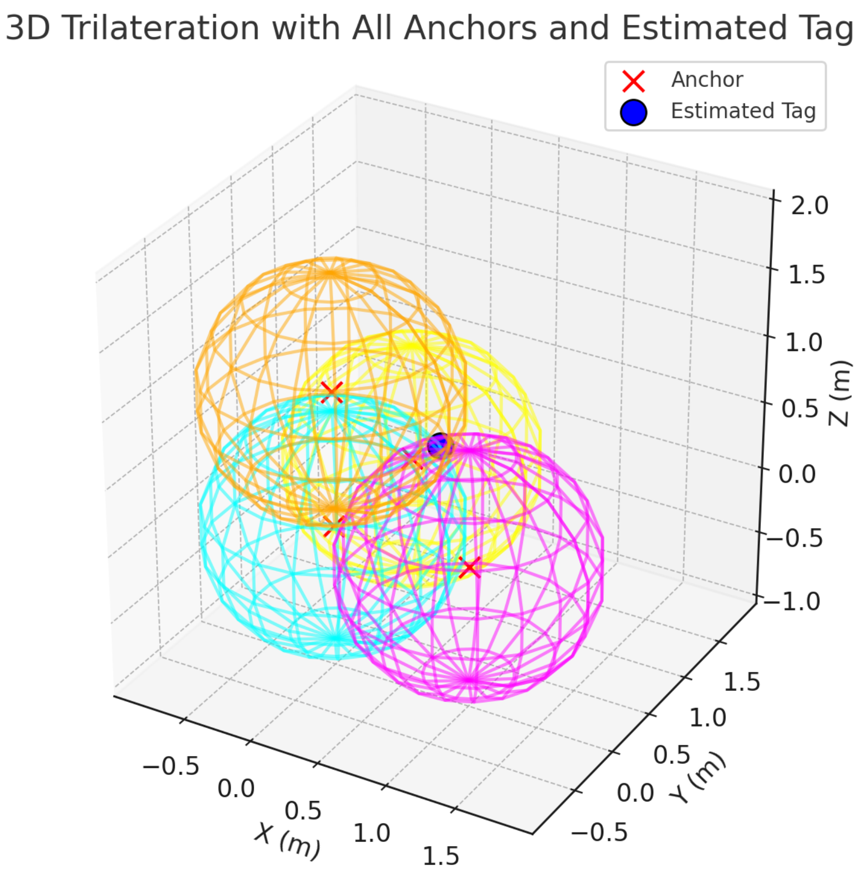


Figure B.1: Graphical representation of the trilateration process used to locate a tag with four anchors.

B.1 Inter-robot transformations based on distances

The previous section covered the general case of estimating the location of a tag in a static sensor network. Now, suppose two robots are moving in 3D. One robot is carrying one or more anchors, and the other carries one or more tags, which provide range measurements between them. Additionally, each of them is able to estimate their position and orientation in their own global frame (p.e. via dead-reckoning). In cooperative localization, sensor-fusion algorithms generally require that all measurements are expressed with respect to a common frame of reference. This implies knowing the relative transformation between said global frames, which can be inferred from relative range measurements taken at different vantage points of the robot trajectories as they move. This problem is known as motion-induced robot-to-robot extrinsic calibration, and has been previously covered in [4]. In this seminal work, the authors provide an observability analysis to demonstrate that the robots must move between observations in order to reliably estimate a closed-form 6-DOF relative pose (expressed as a translation vector $\mathbf{p_0}$ and a quaternion $\mathbf{q_0}$), and an algebraic method to compute it. In what follows, a brief explanation of the general formulation presented in that work is provided. For more information, please refer to [4].

Let $\mathbf{p_{1,k}}$ and $\mathbf{p_{2,k}}$ be the positions of robot 1 and 2 at time k in their global frames (so that $\mathbf{p_{2,k}} = [x_{2,k}, y_{2,k}, z_{2,k}]^T$), and d_k the range measurement at time k, the scalar term ε_k is defined as a function of known or measured values:

$$\varepsilon_k := \frac{1}{2}\left(d_0^2 + \mathbf{p_{1,k}^T p_{1,k}} + \mathbf{p_{2,k}^T p_{2,k}} - d_k^2\right) \tag{B.1}$$

From this, the authors formulate a homogeneous linear system where the vector of unknowns $\bar{\mathbf{x}}_{17x1}$ includes: a vector $\bar{\mathbf{q}}^T$ of 10 quadratic monomials formed from $\mathbf{q_0}$; the relative translation vector from the global frame of robot 2 to robot 1 $\mathbf{p_0^T}$; the rotated translation vector $\mathbf{r_0} = \mathbf{C_0}^T\mathbf{p_0}$ - where $\mathbf{C_0}$ is the rotation matrix from the global frame of robot 2 to robot 1- and the scalar $\rho = 1$. In this way, the system is defined as follows:

$$\mathbf{M}\bar{\mathbf{q}} + \mathbf{A}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{r}_0\end{bmatrix} - \boldsymbol{\varepsilon} = \mathbf{0} \Rightarrow \mathcal{M}\bar{\mathbf{x}} = \mathbf{0} \tag{B.2}$$

where,

$$\mathbf{A} := \begin{bmatrix}\mathbf{p}_{1,1}^T & -\mathbf{p}_{2,1}^T\\ \vdots & \vdots\\ \mathbf{p}_{1,n-1}^T & -\mathbf{p}_{2,n-1}^T\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \quad \boldsymbol{\varepsilon} := \begin{bmatrix}\varepsilon_1\\ \vdots\\ \varepsilon_{n-1}\\ 1\end{bmatrix} \tag{B.3}$$

$$\mathcal{M} := \begin{bmatrix}\mathbf{M} & \mathbf{A} & -\boldsymbol{\varepsilon}\end{bmatrix}, \quad \bar{\mathbf{x}} := \begin{bmatrix}\bar{\mathbf{q}}^T & \mathbf{p}_0^T & \mathbf{r}_0^T & \rho\end{bmatrix}^T \tag{B.4}$$

$$\mathbf{M}(k,:) = \mathbf{p}_{1,k}^T \left[\begin{matrix} x_{2,k} & 2y_{2,k} & 2z_{2,k} & 0 & -x_{2,k}\\ -y_{2,k} & 2x_{2,k} & 0 & 2z_{2,k} & y_{2,k}\\ -z_{2,k} & 0 & 2x_{2,k} & -2y_{2,k} & -z_{2,k}\end{matrix}\right.$$

$$\left.\begin{matrix} 0 & -2z_{2,k} & -x_{2,k} & 2y_{2,k} & x_{2,k}\\ 2z_{2,k} & 0 & -y_{2,k} & -2x_{2,k} & y_{2,k}\\ 2y_{2,k} & 2x_{2,k} & z_{2,k} & 0 & z_{2,k}\end{matrix}\right], \quad k = 1, \ldots, n-1$$

$$\mathbf{M}(n,:) = \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1\end{bmatrix}$$

More recently, the authors in [6] followed this work with a simplified method that estimates 4 DOF of the relative transformation (the translation component $\boldsymbol{\varepsilon} = [u, v, w]^T$ and an elemental rotation around the vertical axis α) which showed better performance compared to the original 6 DOF method in [4]. In this case, a reduced vector of unknowns $\bar{\mathbf{x}} = [u, v, w, cos\alpha, sin\alpha, L_1, L_2, L_3]^T$ is defined, where L_1, L_2, L_3 are auxiliary variables:

$$L_1 = ucos\alpha + vsin\alpha \tag{B.5}$$

$$L_2 = vcos\alpha - usin\alpha \tag{B.6}$$

$$L_3 = u^2 + v^2 + w^2 \tag{B.7}$$

Recovering the previous notation, at each vantage point k in the trajectories, the positions of robot 1 and robot 2 in their own global frames ($\mathbf{p_{1,k}} = [x_{1,k}, y_{1,k}, z_{1,k}]^T$ and $\mathbf{p_{2,k}} = [x_{2,k}, y_{2,k}, z_{2,k}]^T$, respectively) are known, as well as the distance measured between them d_k. With this information, a linear system of equations is formulated:

$$\mathbf{M\bar{x}} - \mathbf{b} = \mathbf{0} \tag{B.8}$$

where,

$$\mathbf{M}(k,:) = \left[\begin{array}{cccc} -2x_{1,k} & -2y_{1,k} & 2z_{2,k} - 2z_{1,k} & -2(x_{1,k}x_{2,k} + y_{1,k}y_{2,k}) \end{array}\right.$$
$$\left.\begin{array}{cccc} 2(x_{1,k}y_{2,k} - y_{1,k}x_{2,k}) & 2x_{2,k} & 2y_{2,k} & 1 \end{array}\right], \quad k = 0, \ldots, n-1$$

$$b_k = d_k^2 - \mathbf{p_{1,k}^T p_{1,k}} - \mathbf{p_{2,k}^T p_{2,k}} + 2z_{1,k}z_{2,k}, \quad k = 0, \ldots, n-1 \tag{B.9}$$

While both methods aim to solve the same problem, this thesis uses the latter to obtain a prior for NLS, because of its simpler formulation and better suitability to the use case covered in this work (4-DOF relative estimation with UWB sensors). For more information on this particularized method, including comparisons with the original work in [4], please refer to [6].

Appendix C

4-DOF error function

At many points in this work, it is necessary to perform operations with SE(3) poses. Particularly, it is important to 1) concatenate poses by applying relative transformations from odometry, 2) compute the difference or relative transformation between two poses, and 3) project the poses to the degrees of freedom that we are interested in estimating. To do this, we adopt the motion composition operators, as well as the exponential and logarithmic mappings. These operations are associated with Lie Groups and their corresponding Lie Algebras, which are widely used in localization and SLAM problems. This appendix discusses the basic concepts that directly relate to the work presented in this document. For more information, please refer to [35] and [34].

Let **a**, **b** and **c** be SE(3) poses, this is, 6D poses belonging to the Lie Group known a Special Euclidean Group in 3D. Poses have different parameterizations, and one of the most commonly used in localization and SLAM is 4x4 homogeneous transformation matrices. Assuming this is the representation we are using, we define the operators:

$$\mathbf{c} = \mathbf{a} \oplus \mathbf{b} = \mathbf{a} \cdot \mathbf{b} \tag{C.1}$$

$$\mathbf{b} = \mathbf{c} \ominus \mathbf{a} = \mathbf{a}^{-1} \cdot \mathbf{c} \tag{C.2}$$

Here, $\oplus$ is the motion composition operator, which applies a perturbation **b** (typically, a relative motion resulting from odometry) to a pose **a** expressed in a global frame. This concatenation yields a new pose **c** also expressed in the global frame. Conversely, the inverse motion composition $\ominus$ expresses **c** locally in the frame of **a**. It can be used to retrieve the relative motion, or difference, between two poses expressed in a common frame (for example, a prediction and an observation).

Other useful operations when working with Lie Groups are the exponential and logarithmic mappings. In SE(3), the logarithmic map $\xi(.)$ maps an SE(3) element to its minimal 6D representation in the associated Lie Algebra $se(3)$, which contains three translational components and three rotational components. This operation is invertible through the exponential map, with maps a minimal representation in $se(3)$ back to its corresponding element in SE(3).

The logarithmic mapping is used in this work to obtain a minimal representation of a 4x4 transformation matrix that can be passed to the optimizer, reducing the number of parameters from 16 down to 6 in the case of full 3D poses. In our particular scenario we only are interested in translation and rotation along the vertical axis, so the optimizer must work with just 4 variables, and the remaining degrees of freedom must be held fixed. To do this, we retrieve the three translational components and the last angular component of the minimal representation to compute a 4D residual. Recovering the notation in Equation C, for a predicted pose $\mathbf{a}$ and observed pose $\mathbf{b}$, the error function becomes:

$$e_{4D}(\mathbf{a}, \mathbf{b}) = \xi(\mathbf{a} \ominus \mathbf{b})_{[1:3,6]} \tag{C.3}$$

This definition results in a smooth and differentiable error that is invariant to the global coordinate frame and consistent with pose composition. Additionally, it can very easily be extended to full 6-DOF estimation by using the whole six elements of the logarithmic map.