Estimation Methods for Multi-robot Relative Localization using Ultra-wideband Radio

Charles Champagne Cossette

Department of Mechanical Engineering
McGill University, Montreal

April 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Acknowledgements

A doctorate degree is a long journey made pleasant when supported by peers and friends. My supervisors Prof. James R. Forbes and Prof. David Saussié have always been there for guidance when requested, for which I thank them both. I have spent more time with James R. Forbes, who truly is a top supervisor, and has given me an uncountable number of valuable lessons, technical or otherwise, that will remain with me far beyond this degree.

Mohammed A. Shalaby is another individual to which I owe much of the success of this thesis. Mohammed was my main collaborator on this project, and has contributed directly to many chapters of this thesis. Mohammed also contributed substantially to the experimental work, including, but not limited to, the firmware development of a custom ultra-wideband radio module. We have enjoyed significant productivity gains, and general good times, by collaborating closely. If you are reading this and are hiring, look no further than Mohammed.

Prof. Jérôme Le Ny has also played a key role in this project, being a principle investigator on the original project proposal. Prof. Le Ny and his students have also contributed to the development of the custom ultra-wideband modules used in many experiments in this thesis.

I should also not understate the importance of the people close to me outside this project and research in general. Regular shenanigans and beers were found to be critical for maintaining a steady motivation throughout the past four years. So, to all of my friends, thank you very much, and I look forward to continuing such activities.

Finally, it can be easy to forget just how lucky I am to have the constant support of my family. And of course, there is Annie Pike, who has been the *ultimate* girlfriend throughout this Ph.D.

Table of Contents

Acknowledgements ii

List of Figures vi

List of Tables x

Abstract xi

Statement of Contributions xiii

Chapter

1. Introduction 1

1.1 Background and Related Work 2
1.2 Outline 5

2. Preliminaries 6

2.1 Notation 6
2.2 Probability Density Functions 6
2.2.1 Gaussian Distributions 7
2.3 Lie Groups 8
2.3.1 Basic Definitions 10
2.3.2 $\oplus$ and $\ominus$ operators 12
2.3.3 Gaussian distributions on Lie groups 12
2.3.4 Derivatives on Lie groups 13
2.3.5 Composite groups 13
2.4 Maximum A Posteriori 14
2.4.1 Observability 15
2.5 Consistency of Bayesian Estimators 16

3. Relative Position Estimation Between Two UWB Devices with IMUs 18

3.1 Summary 18

3.2 Introduction . . . 18
3.3 Sliding Window Filtering . . . 20
3.4 Proposed Algorithm . . . 21
3.4.1 High-level architecture . . . 21
3.4.2 Attitude & Heading Reference System . . . 22
3.4.3 Relative Position Estimator . . . 23
3.4.4 Observability Analysis . . . 25
3.4.5 Keypoint Selection Strategy . . . 26
3.4.6 Multi-robot Scenario . . . 28
3.5 Simulation Results . . . 29
3.6 Experimental Results . . . 30
3.7 Conclusion and Future Work . . . 33

4. Localization with Directional Coordinates . . . 36

4.1 Summary . . . 36
4.2 Introduction . . . 36
4.3 Directional Coordinates . . . 38
4.3.1 Cartesian to Directional Coordinates . . . 39
4.3.2 Local Parameterization . . . 39
4.4 Distributions in Directional Coordinates . . . 40
4.4.1 Converting a Cartesian Gaussian Distribution to Directional Coordinates . . . 41
4.4.2 Posterior Distribution given a Range Measurement . . . 41
4.4.3 Posterior Distribution given Azimuth and Elevation Measurements 43
4.5 Directional Coordinate Kinematics . . . 44
4.5.1 Linearization . . . 45
4.6 Simulation Results . . . 46
4.7 Experimental Results . . . 48
4.8 Conclusion and Future Work . . . 49

5. Optimal Multi-Robot Formations for Relative Pose Estimation using Range Measurements . . . 52

5.1 Summary . . . 52
5.2 Introduction . . . 52
5.3 Problem Setup, Notation, and Preliminaries . . . 54
5.3.1 State Definition and Range Measurement Model . . . 54
5.4 Optimization . . . 55
5.4.1 On-manifold Cost and Gradient Descent . . . 57
5.4.2 Cost Function Implementation . . . 57
5.4.3 Optimization Results . . . 60
5.4.4 Validation on a Least-Squares Estimator . . . 61
5.5 Experimental Evaluation . . . 63
5.6 Conclusion and Future Work . . . 65

6. On-manifold Decentralized State Estimation using Pseudomeasurements and Preintegration 67

6.1 Summary 67
6.2 Introduction 68
6.3 Related Work 70
6.4 Preliminaries 71
6.4.1 Covariance Intersection 71
6.5 A Toy Problem 72
6.5.1 Solution via MAP 73
6.6 General Problem 75
6.6.1 An Observability Test 78
6.7 Efficient Odometry Sharing using Preintegration 80
6.7.1 Multi-robot preintegration 84
6.7.2 Estimating Input Biases 86
6.7.3 Autoencoding Covariance Matrices 86
6.8 Simulation with Ground Robots 89
6.9 Simulation and Experiments with Quadcopters 91
6.9.1 Simulation Results 94
6.9.2 Hardware Setup 94
6.9.3 Experimental Results 95
6.10 Conclusion and Future Work 96

7. Concluding Remarks 98

7.1 Future Work 98

Appendices 101

A. The Complex-Step Derivative Approximation on Matrix Lie Groups 102

A.1 Introduction 102
A.2 Review 103
A.3 The Complex-Step Derivative on Matrix Lie Groups 103

B. Contributed open-source software 106

List of Figures

Figure

1.1 **Left:** Custom UWB module built around the DW1000 UWB transceiver. **Right:** A customized Uvify IFO-S quadcopter, outfitted with a UWB transceiver on one of its legs. 2

1.2 The two-way-ranging protocol: the most basic method to determine distance using UWB radio. 3

1.3 **Left:** Concept diagram of traditional UWB positioning systems. **Right:** Concept diagram of a self-localizing robotic team, capable of estimating relative positions or poses from only on-board sensors. 4

2.1 Birds-eye view of a simple ground robot moving in a 2D world. 9

2.2 Distribution of robot poses, along with modelled representations using both exponential coordinates from Lie group theory, as well as traditional Cartesian coordinates. 9

3.1 An example scenario where two quadrotors possess UWB modules, providing range measurements between them. Under sufficient motion, it is possible to determine the relative position between them in a common reference frame. 19

3.2 High-level diagram of the architecture of the proposed algorithm. Each agent contains an AHRS that uses on-board accelerometer, rate gyro, and magnetometer measurements. 21

3.3 Effect of the parameter γ on the keypoint selection with $K = 12$, and the old trajectory shown as the black line. **Left:** With $\gamma = 0$, the algorithm greedily minimizes the GDOP, and results in keypoints placed far apart in time. **Middle:** With a moderate value for γ, the algorithm obtains a balance between pure GDOP minimization, and making all the keypoints as close as possible. **Right:** With a large value for γ, the minimal cost is the solution with the keypoints as close as possible, and the "plain vanilla" sliding window filter is recovered. 28

3.4 Results of 200 Monte Carlo trials. The proposed method offers an enhancement over the "plain vanilla" implementation of the sliding window filter (SWF), which simply places the keypoints at the most recent range measurements, as well as a standard extended Kalman filter (EKF), and an Iterated EKF (IEKF). 28

3.5 Histogram of the computation time of the different algorithms, when simulated on a laptop with an Intel Core i7-9750H CPU. 29

3.6 Simulation of the proposed estimation algorithm. The ground truth trajectory is shown in black, and the estimated trajectory in blue. 30
3.7 **Left:** Pozyx UWB Developer Tag mounted to a Raspberry Pi 4B, in a case. **Right:** Pozyx UWB Developer Tag mounted to a quadrotor with a Pixhawk 4. 31
3.8 Components of the position estimation error during an experimental trial, using the greedy keypoint placement method. The error compared to ground truth is shown as the blue line, with the shaded area representing the $\pm 3\sigma$ confidence bounds. 32
3.9 Comparison of the norm of the position estimation error $\mathbf{e}_r$, during an experimental trial. 33
4.1 Distribution of positions of a robot given a single distance measurement (red line) to a landmark (green pyramid) and a Gaussian prior distribution. The point cloud represents the "true" distribution, as determined by a particle filter, while the red and blue volumes are 3σ equal probability contours in Cartesian and directional coordinates, respectively. 37
4.2 Samples from a Cartesian Gaussian distribution, shown as the point cloud. The bean-shaped volume shows a 3 standard deviation equal probability contour of a Gaussian distribution in directional coordinates, but mapped back to Cartesian coordinates for visualization. 40
4.3 1000 Monte Carlo trials of a single Kalman correction given a distance measurement. Although the estimation accuracy using directional coordinates is not necessarily improved, directional coordinates offer a more consistent filter, as well as one that better resembles the particle filter. 42
4.4 Simulation results from a single trial. Black lines represent the $\pm 3\sigma$ confidence bounds. The total estimation error on the bottom figure is calculated for both filters with $\mathbf{e} = [\mathbf{r}^{\text{true}^\mathsf{T}}\ \mathbf{v}^{\text{true}^\mathsf{T}}]^\mathsf{T} - [\hat{\mathbf{r}}^\mathsf{T}\ \hat{\mathbf{v}}^\mathsf{T}]^\mathsf{T}$. 44
4.5 Average estimation error and NEES, throughout time, for 100 Monte Carlo trials. The dashed line represents the one-sided 99.7% probability boundary. 46
4.6 Pozyx UWB Developer Tag mounted to a Raspberry Pi 4B. 48
4.7 Experimental results from a single trial, with the dashed line representing the one-sided 99.7% probability boundary. 48
5.1 Problem setup and notation used. Each agent possesses a reference point a_α where α is the agent ID, as well as two tags τ_i, τ_j, where i, j are the tag IDs. $\mathbf{1}_x$ and $\mathbf{1}_y$ are vectors which represent the x and y axis of Agent 1's body frame. Throughout this chapter, the red agent is the arbitrary reference agent, and it will always be Agent 1 without loss of generality. 54
5.2 All four plots show the value of the cost with varying agent position (right agent for two-agent scenario, top agent for three-agent scenario), while maintaining fixed heading. The top row shows only the estimation cost J_{est}, while the bottom row shows the total cost J including the collision avoidance term. 59
5.3 Final locally optimal formations for 3, 4, 5, and 10 agents, with the optimization paths shown in blue. 62
5.4 **Left:** Optimal formation with sparse measurement graph. **Right:** Optimal formation with 3D position and heading as design variables. 63

5.5 Trajectory taken during optimization with superimposed 1σ equal-probability contours corresponding to the Cramér-Rao bound. The ellipsoids for the starting positions are too large to fit in the figure. 63
5.6 Cost function, along with various metrics of a least-squares estimator, obtained from 2000 Monte-Carlo trials at various points during the optimization. In the bottom two plots, the red squares denote the average norm of the respective estimation errors. 64
5.7 Experimental results using a least squares estimator. From 0 s to 30 s, the quadcopters are in a line formation and the average positioning error is 0.77 m. From 30 s to 60 s, the quadcopters are in an optimal formation and the average positioning error is 0.22 m, a 68% reduction. 64
5.8 **Top:** Three quadcopters in an initial straight line formation. **Bottom:** Quadcopters in an optimal triangle formation. 65
6.1 Three examples of decentralized estimation problems within the scope of this chapter. **Left:** A toy problem with 1D robots, each estimating both of their positions. **Middle:** A problem with an incomplete communication graph. Robots observe landmarks, have range measurements to each other, and estimate their own and neighbor absolute poses. **Right**: A more complicated experimentally-tested problem, where robots equipped with ultra-wideband radios estimate both their own absolute pose and relative poses of neighbors, in addition to IMU biases. 67
6.2 Estimation convergence for a single trial of the two-robot toy problem with $\mathbf{\Psi} = 10 \cdot \mathbf{1}$. Due to pseudomeasurements, the robot states successfully converge to a common value. 75
6.3 Results of 100 Monte Carlo trials for a four-robot version of the toy problem. The top two plots consist of a NEES plot, which is a measure of consistency. The bottom plot is the RMSE of the state. The proposed solution, which performs CI, remains statistically consistent and has reasonbly low error in many cases. 76
6.4 Concept diagram of mean-assisted autoencoder. 87
6.5 Mean percentage reconstruction error throughout training for various encoding sizes including no encoding. A single encoding number is sufficient to achieve less than 1% reconstruction error on average. 88
6.6 Visualization of preintegrated IMU noise covariance matrices along with reconstruction using mean-assisted autoencoding. 88
6.7 RMSE for the ground robot simulation. There are four blue lines for the four robots running the proposed algorithm, and four visibly coincident red lines for the naive algorithm. **Top:** State fusion occuring at 10 Hz. **Bottom:** State fusion occuring at 1Hz. 90
6.8 NEES plots for the ground robot simulation with 95% confidence bounds for the proposed vs. centralized solution. The naive solution without CI is far outside the plot. 91

6.9 Simulation estimation error of Robot 3's estimate of its own kinematic state and IMU biases. The estimate and corresponding bounds with the proposed algorithm is shown in blue, with the centralized estimate overlayed in dark grey. For attitude, the $x - y - z$ components represent roll-pitch-yaw errors respectively. Note that Robot 3 does not have GPS measurements, and therefore cannot observe the states shown in this plot without information sharing. The naive solution has rapidly diverging error and is not plotted. . 92
6.10 **Top:** Three quadcopters in flight under a motion capture system. **Bottom left:** our custom UWB module. **Bottom right:** a close up of the Uvify IFO-S quadcopter, fitted with a UWB module seen on the left leg, as well as on the opposite leg. 93
6.11 Examples of the various trajectories flown in the experimental trials, where each color represents a different quadcopter. 94
6.12 Position, velocity, and yaw RMSE for Robot 3 from one of the experimental trials. Since Robot 3 has no GPS, these quantities are unobservable without the fusion of pseudomeasurements. 95
6.13 The effect of preintegrated covariance autoencoding, as described in Section 6.7.3, on the position estimate of Robot 1. The two lines are almost identical, showing that the proposed autoencoder induces minimal error on the estimate. All other states have similar plots. 96
A.1 Variation of relative error in gradient of $f : SE(3) \to \mathbb{R}$ with step size, for both complex-step and central-difference methods. Machine precision is achievable with a sufficient reduction in step size. 105

List of Tables

Table

3.1 Noise properties for single-range and IMU simulation 31
3.2 Position RMSE of Experimental Trials 32
4.1 Simulation Noise Properties 47
6.1 Self- Positioning RMSE (m) from experimental trials. 94

Abstract

This thesis explores various methods for the autonomous estimation of the relative position and attitude between robots, using onboard sensors. This functionality is a core low-level prerequisite for a variety of multi-robot tasks that require collaboration, such as a team jointly lifting an object, exploring a space, or maintaining a formation. In particular, this thesis puts emphasis on a particular technology called Ultra-wideband radio (UWB), which is capable of generating distance measurements between a pair of transceivers with approximately 10cm of standard deviation. UWB is cheap, lightweight, and can be simultaneously used as a data transfer medium, thus making it particularly attractive for small, mobile robots. Furthermore, Lie group theory is frequently used to represent the state due to its superior ability to represent the distribution of robot poses. Centralized and decentralized algorithms are presented, each evaluated in both simulation and experiment.

Résumé

Cette thèse explore plusieurs méthodes pour l'estimation autonome de la position et de l'orientation relatives entre robots, en utilisant des capteurs embarqués. Cette fonctionnalité est un prérequis fondamental pour une variété de tâches multi-robots qui nécessitent de la collaboration, telles qu'une équipe qui soulève ensemble un objet, qui explore un espace, ou qui maintient une formation. En particulier, cette thèse se concentre sur une technique particulière appelée la radio à bande ultra-large (UWB), qui est capable de générer des mesures de distance entre une paire d'émetteurs-récepteurs avec un écart type d'environ 10 cm. L'UWB n'est pas chère, elle est légère et peut être aussi utilisée comme moyen de transfert de données entre robots, ce qui la rend particulièrement intéressante pour des petits robots mobiles. De plus, la théorie des groupes de Lie est fréquemment utilisée pour représenter l'état des robots en raison de sa capacité supérieure à représenter la distribution des poses du robot. Des algorithmes centralisés et décentralisés sont présentés, chacun étant évalué à la fois par simulation et par expérimentation.

Statement of Contributions

The following is a list of original contributions presented in this thesis.

- Chapter 3 [1]
  - A keypoint-based sliding-window filter for estimation of relative positions from single-range measurements.
  - A greedy keypoint selection strategy that is a quickly computable heuristic for an information-maximizing sliding-window filter.
- Chapter 4 [2]
  - A novel coordinate system appropriate for localization based on range, azimuth, elevation measurements.
  - An investigation into the coordinate system's consistency properties and divergence from the true distribution of positions, including a comparison to traditional Cartesian coordinates.
- Chapter 5 [3]
  - A cost function that effectively captures the estimation variance's dependence on the formation geometry of a group of robots.
  - A method for computing optimal formations based on the above cost function, and using on-manifold gradient descent.
- Chapter 6 [4]
  - A framework for decentralized multi-robot estimation based on pseudomeasurements.
  - A method for modelling common variables between robots' state definitions, and more generally, arbitrary nonlinear relationships between robot states.
  - An observability test for decentralized estimators that simultaneously accounts for local measurements and the communication topology within the team.
  - A method for sharing odometry information in a very communication-efficient way, using preintegration.
  - An autoencoder architecture that further compresses covariance information associated with preintegration.
- Appendix A [5]
  - A numerical differentiation method appropriate for functions of matrix Lie group

elements, that obtains machine-precision.

– An open-source Python software that provides general-purpose implementations of common on-manifold estimation algorithms.

Credit is given to Mohammed A. Shalaby for assistance in the experimental work present in this thesis, and for general help in idea generation that led to many of the presented contributions. However, all previously listed contributions were independently developed by the author of this thesis.

Chapter 1

Introduction

The overarching goal of this thesis is to create a team of autonomous robots that can each localize their fellow team members, by means of on-board sensors. This functionality, which ultimately produces numerical estimates of the relative position or pose between robots, is a core low-level prerequisite for a large class of multi-robot tasks. If a robotic team desires to lift an object together, explore a space together, or simply maintain a fixed formation, an accurate estimate of their relative positions will be needed.

More specifically, the goal is to achieve relative positioning in the most lightweight way possible, regarding computational and sensor requirements. Hence, there is emphasis on a specific technology called *ultra-wideband radio* (UWB). Using UWB in robotics consists of installing one or more very small UWB transceivers on each robot, and possibly more in the surrounding environment. These transceivers, often called *tags*, provide high-frequency range measurements between any pair of them, with roughly 10 cm of standard deviation. UWB transceivers are cheap, lightweight, require extremely low power, and double as a means to wirelessly transfer data between robots. It is therefore an ideal technology for small robots with limited computational resources, such as micro air vehicles. Moreover, when combined with other sensors, it is theoretically possible to even further improve positioning accuracy, thus enabling high-precision applications such as collaborative manipulation. Only recently have UWB-enabled robotic teams emerged in academic work [6–10], and the state-of-the-art still has far to go to become accurate, fast, and robust.

This thesis also aims to minimize dependence on external infrastructure, such GPS or UWB transceivers installed at known, fixed locations. Achieving this would result in a team of robots with a fully self-contained localization system that is not confined to a particular environment with such infrastructure. This, however, complicates the estimation task as there is no information from an absolute positioning systems, and sometimes not even a common reference frame between robots.

Substantial effort has been put into developing custom UWB modules for the experimental work of this thesis, which internally use a commercial chip for UWB transmission. These compact modules are seen in Figure 1.1, along with a picture of the quadcopter platform used in the later half of this work that possesses one of these modules. These custom modules have allowed for precise control over the ranging protocol, calibration procedure, and scheduling — all in the interest of maximizing the estimation accuracy. Developing custom modules has led to a unique experimental setup, where multiple robots exist in space, each possessing multiple UWB transceivers. The module firmware has been designed to not only schedule transmission times in a decentralized way, such that interference from colliding transmissions are avoided, but also to allow any one robot to "eavesdrop" on the range measurement occurring between any other two, thus itself gaining access to that range information.

Figure 1.1: **Left:** Custom UWB module built around the DW1000 UWB transceiver. **Right:** A customized Uvify IFO-S quadcopter, outfitted with a UWB transceiver on one of its legs.

Another distinctive feature of this thesis will be the heavy use of *Lie groups* to represent robot states, thus making many methods in this thesis "on-manifold" estimation algorithms. As shown in [11, 12], Lie groups provide a superior ability to model the distribution of three-dimensional robot poses under significant uncertainty, and lead to more robust estimators when the estimation error is large [13, 14]. Amongst other benefits, they provide a compact, elegant mathematical treatment of many robotic state estimation problems. Lie group theory will be briefly introduced in Chapter 2 of this thesis.

1.1 Background and Related Work

While UWB radios can often be thought of as simple "distance sensors," their actual core capability is the precise, sub-nanosecond-level timestamping of transmission or reception of radio waves [15]. This is largely achieved by sending very "sharp" impulse-like radio waves,

which correspond to a wide bandwidth in the frequency domain, thus warranting the name "ultra-wideband". The range between two transceivers is hence determined by measuring the *time-of-flight* (TOF) of a radio wave that originates from one transceiver and is received by another. The main complexity is that the clocks on each transceiver are not synchronized, and in the simplest case, their values at any point in universal time differ by a fixed offset. The *two-way-ranging* (TWR) protocol, shown in Figure 1.2, solves this problem by sending two signals, back and forth.

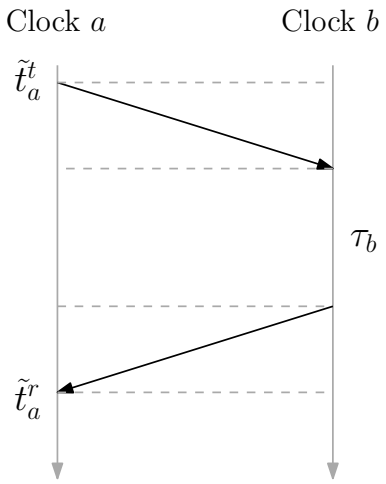

Figure 1.2: The two-way-ranging protocol: the most basic method to determine distance using UWB radio.

Assuming that the clock offset between the transceivers is constant, the range can be estimated from the total round-trip duration, cancelling out the clock offset. However, as discussed in [16], this constant offset is not always a good assumption, and it is preferred to use a first-order model featuring an offset changing at a constant rate, called the *clock skew*. An improved ranging protocol is proposed in [16] to account for this clock skew. Furthermore, [17] discusses how the intervals in the ranging protocol, such as τ_b, affect the range estimate variance, and can therefore be optimized to provide the range measurements with the lowest variance. In [18], tag clock offsets are simultaneously, dynamically estimated with the robot's pose, which can provide improved positioning accuracy. It is also possible to determine the *angle of arrival* of a UWB radio wave by having receivers with multiple antennas, as shown in [19, 20]. However, this multi-antenna hardware is not yet commercially available. In [21], the angle of arrival is directly estimated from a single antenna by examining the time-domain profile of the received signal, and mapping this to an angle using a deep neural network. However, this has poor accuracy and likely requires a hardware-specific training process.

The traditional use of UWB radio for positioning involves installing tags at fixed, known positions called *anchors*, illustrated on the left of Figure 1.3. Determining the position of the mobile tag from range measurements to these known landmarks can be done by *trilateration*,

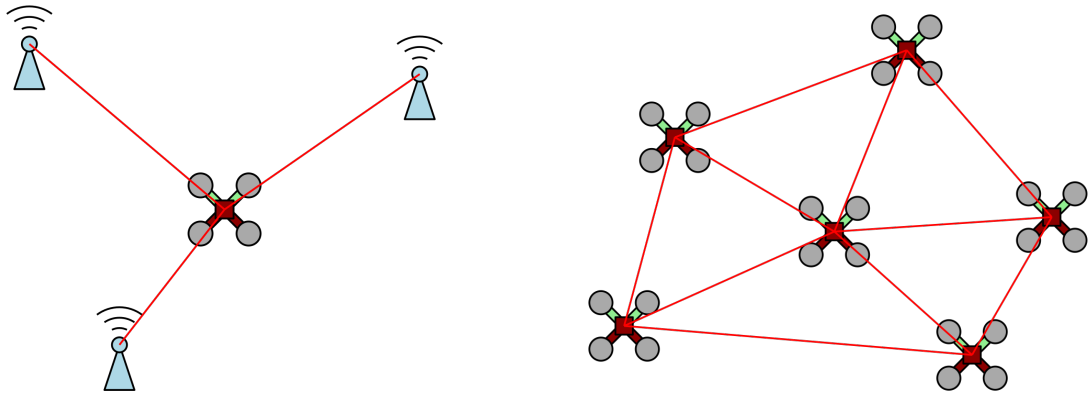

Figure 1.3: **Left:** Concept diagram of traditional UWB positioning systems. **Right:** Concept diagram of a self-localizing robotic team, capable of estimating relative positions or poses from only on-board sensors.

which involves a least-squares problem [22]. This kind of approach has been successfully used many times in robotics to provide autonomous positioning capability [23]. For example, UWB is used to localize an unmanned blimp in [24], and quadcopters in [25–27]. However, the problem of interest in this thesis is better represented by the right of Figure 1.3. In this situation, robots would only use on-board sensors for relative localization. Infrastructure independence makes the problem substantially more difficult, as it typically results in the lack of any global or absolute reference.

If the robots each possess only a single range-measuring transceiver, then at any one instant it is impossible to determine relative positions from those range measurements. The solution is ambiguous as the entire team can rotate, translate, flip and each robot can freely rotate without changing the range measurements [28]. However, if the robots are moving sufficiently, and the history of measurements is taken into account, then the relative trajectories satisfying all the measurements can be unique [29, 30]. This is simulated in [31], where velocity measurements are assumed to be available, and experimented in [6], however GPS is used to provide displacement information for the agents. In [8] a two-dimensional solution is proposed, with robots collecting velocity measurements from optical flow sensors, and sharing this data between them. Controllers are proposed in [32, 33] that ensure the motion is persistently exciting, as required. More recently, [9] proposes a convex optimization approach to the relative localization problem for two robots with single-range measurements between them.

Another approach that theoretically eliminates the persistent motion requirement is to install more than one tag on each robot. The uniqueness and observability of such a solution is presented in [28], where quadcopters possess a mixture of either one or two tags. Similarly, a large ground robot possessing four tags is shown in [34], and allows a single-tag quadcopter to localize itself relative to this ground robot. A similar setup is seen in [35].

1.2 Outline

This thesis represents the development journey toward the overarching goal of relative positioning capability, starting with a chapter dedicated to covering mathematical preliminaries, Chapter 2.

Chapter 3 describes a first approach to solving the relative positioning problem, which is to outfit each robot with a single UWB transceiver, and to exploit the robots' motion history to determine a relative position estimate. At any single instant, the single range measurement between robots is insufficient information to calculate a relative position. However, as will be shown in Chapter 3, if an inertial measurement unit (IMU) is available that provides motion information, it is possible to estimate the relative position. The main contribution of this chapter is to propose a method for selecting specific measurements in the state history that are most informative, which in turn improves accuracy over traditional approaches.

Chapter 4 builds on the single-transceiver approach by instead considering an alternate coordinate system to represent the relative positions. The main realization from previous work was that the distribution of position estimates, given range measurements, were spherically-shaped. Gaussian estimators operating in the usual Cartesian coordinates would therefore always fail to represent such a distribution, and this issue led to the solution proposed in this chapter.

Chapter 5 now switches to multiple-transceiver-per-robot approaches. While single-transceiver approaches had the benefits of compactness and keeping hardware to an absolute minimum, they have the impractical requirement of persistently exciting motion in all axes. As shown in this chapter, multi-transceiver robots can determine relative positions from range measurements alone. This chapter further studies how the actual formation geometry of a group of robots effects the estimation accuracy. This effect is modelled, quantified, and then optimized to produce formations that are optimal in terms of estimation variance.

Chapter 6 finally describes a concrete decentralized state estimator that is appropriate for relative pose estimation with inter-robot range measurements. The treatment in this chapter is general to a large variety of decentralized state estimation problems, and is not limited to range measurements. This chapter will address the fusion of arbitrary state definitions, investigate the observability of decentralized estimators, and propose methods that substantially reduce the communication cost associated with decentralized estimators.

It is also worth mentioning that many methods proposed in this thesis are not actually specific to robotics, but rather applicable to any device that contains the appropriate sensors. Examples include smartphones, smartwatches, or more custom-built electronic devices.

Chapter 2

Preliminaries

This chapter presents mathematical preliminaries necessary for the explanation, derivation, and proof of the majority of contributions presented in this thesis. More specific mathematical preliminaries relevant to individual chapters will be described therein. As such, the expert reader who is already familiar with probability, estimation, and Lie groups can move ahead to following chapters.

2.1 Notation

In this thesis, column vectors are written in lower-case bold $\mathbf{x} \in \mathbb{R}^n$, whereas matrices are in capital bold $\mathbf{X} \in \mathbb{R}^{m \times n}$. The special matrices $\mathbf{1}$ and $\mathbf{0}$ are appropriately-sized identity and zero matrices, respectively.

An arbitrary reference frame 'a' is denoted $\mathcal{F}_a$. Vectors resolved in $\mathcal{F}_a$ are denoted with a subscript, for example $\mathbf{v}_a$. The same physical vector resolved in a different frame $\mathcal{F}_b$ can be related by a direction cosine matrix (DCM), denoted $\mathbf{C}_{ab} \in SO(3)$, such that $\mathbf{v}_a = \mathbf{C}_{ab}\mathbf{v}_b$ [22]. A direction cosine matrix $\mathbf{C}_{ab}$ can be parameterized using a rotation vector, denoted $\boldsymbol{\phi} \in \mathbb{R}^3$, using $\mathbf{C}_{ab} = \exp(\boldsymbol{\phi}^\times)$, where $(\cdot)^\times$ denotes the skew-symmetric cross-product matrix operator

$$\boldsymbol{\phi}^\times = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}.$$

2.2 Probability Density Functions

A continuous random variable $\mathbf{x} \in \mathbb{R}^n$ has a probability density function (PDF) denoted $p(\mathbf{x})$ [36]. A joint PDF can always be factored as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \tag{2.1}$$

with the last two terms rewritable to produce Bayes' rule

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \tag{2.2}$$

2.2.1 Gaussian Distributions

The Gaussian distribution is a special PDF with mathematical properties that allow for very computationally-efficient estimation algorithms. In covariance form, the Gaussian distribution is parameterized by a mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{2.3}$$

Another convenient equivalent form of the Gaussian distribution is the information form. To derive the information form, the inside of the exponential in (2.3) can be manipulated to give

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})\right) \tag{2.4}$$

$$= \frac{\exp(-\frac{1}{2}(\boldsymbol{\eta}^\mathsf{T}\boldsymbol{\Lambda}^{-1}\boldsymbol{\eta}))}{\sqrt{(2\pi)^n \det \boldsymbol{\Lambda}^{-1}}} \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{x} + \boldsymbol{\eta}^\mathsf{T}\mathbf{x}\right) \tag{2.5}$$

$$= \beta \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{x} + \boldsymbol{\eta}^\mathsf{T}\mathbf{x}\right) \tag{2.6}$$

$$\triangleq \mathcal{N}^{-1}(\boldsymbol{\eta}, \boldsymbol{\Lambda}), \tag{2.7}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is defined as the *information matrix* and $\boldsymbol{\eta} = \boldsymbol{\Lambda}\boldsymbol{\mu}$ as the *information vector*.

2.2.1.1 Marginalization and Conditioning

Marginalization and conditioning are fundamental operations occurring in estimation and Bayesian inference. For the Gaussian distribution, these two operations have simple closed-form expressions [37]. That is, given the joint Gaussian probability density function

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}\right) = \mathcal{N}^{-1}\left(\begin{bmatrix} \boldsymbol{\eta}_x \\ \boldsymbol{\eta}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda}_{xx} & \boldsymbol{\Lambda}_{xy} \\ \boldsymbol{\Lambda}_{yx} & \boldsymbol{\Lambda}_{yy} \end{bmatrix}\right),$$

the marginal pdf $p(\mathbf{x}) = \int_{-\infty}^{\infty} p(\mathbf{x}, \mathbf{y})\mathrm{d}\mathbf{y}$ is given in covariance form as

$$p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx}), \tag{2.8}$$

or in information form as

$$p(\mathbf{x}) = \mathcal{N}^{-1}(\bar{\boldsymbol{\eta}}, \bar{\boldsymbol{\Lambda}}), \tag{2.9}$$

$$\bar{\boldsymbol{\eta}} = \boldsymbol{\eta}_x - \boldsymbol{\Lambda}_{xy}\boldsymbol{\Lambda}_{yy}^{-1}\boldsymbol{\eta}_y, \tag{2.10}$$

$$\bar{\boldsymbol{\Lambda}} = \boldsymbol{\Lambda}_{xx} - \boldsymbol{\Lambda}_{xy}\boldsymbol{\Lambda}_{yy}^{-1}\boldsymbol{\Lambda}_{yx}. \tag{2.11}$$

Similarly, the conditional PDF $p(\mathbf{x}|\mathbf{y})$ can be written in covariance form as

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}), \tag{2.12}$$

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y), \tag{2.13}$$

$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}, \tag{2.14}$$

or in information form as

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}^{-1}(\boldsymbol{\eta}_{x|y}, \boldsymbol{\Lambda}_{x|y}), \tag{2.15}$$

$$\boldsymbol{\eta}_{x|y} = \boldsymbol{\eta}_x - \boldsymbol{\Lambda}_{xy}\mathbf{y}, \tag{2.16}$$

$$\boldsymbol{\Lambda}_{x|y} = \boldsymbol{\Lambda}_{xx}. \tag{2.17}$$

2.3 Lie Groups

Traditional estimation theory operates on a conventional state-space model of a robot's dynamics and sensor measurements. A robot's state, usually denoted $\mathbf{x} \in \mathbb{R}^n$, will be a conventional vector of numbers that parameterize the state of interest in some way, such as position and attitude via Euler angles. Since $\mathbf{x}$ belongs to a vector space, it can be added or scaled, and the result will still be meaningful.

However, the kinematic singularities and ambiguities associated with angle parameterizations of attitude make the DCM $\mathbf{C} \in SO(n)$ a far more natural representation of attitude, where

$$SO(n) \in \{\mathbf{C} \in \mathbb{R}^{n\times n} \mid \mathbf{C}^\mathsf{T}\mathbf{C} = \mathbf{1},\ \det(\mathbf{C}) = 1\}. \tag{2.18}$$

Since the special orthogonal group $SO(n)$ is not a vector space, its elements cannot be added nor scaled with the result still belonging to $SO(n)$. In traditional estimation theory, the state belongs to a vector space, and hence addition and scaling are meaningful operations that can be applied to it. This is not the case in the majority of robotic systems, including those considered in this thesis.

To further motivate the use of Lie groups, consider a situation where a ground robot moves in a 2D plane, as shown in Figure 2.1. This robot travels forward with a fixed average velocity, and simultaneously rotates with a fixed average angular velocity, but both these velocities vary slightly about their mean in a random way. The pose of the robot after any

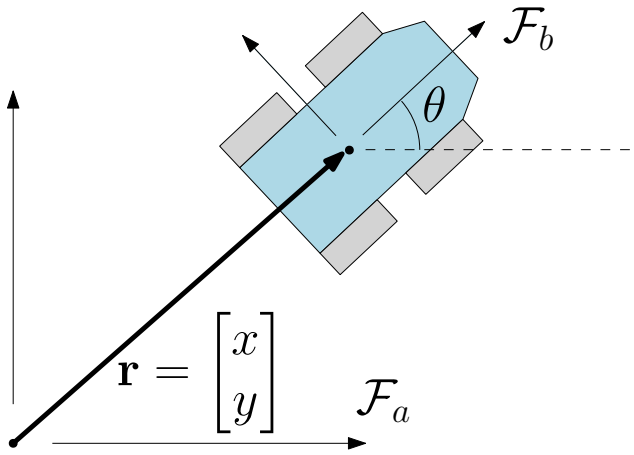

Figure 2.1: Birds-eye view of a simple ground robot moving in a 2D world.

given duration is not deterministic, but is instead random according to some distribution. It turns out that this distribution is truly "banana-shaped", as shown by the point cloud in Figure 2.2. Traditional estimation theory that assumes a Gaussian distribution would attempt to represent this "banana" with some ellipsoid, shown on the right of Figure 2.2, which can be a poor representation. On the other hand, if Lie group tools are used, it is possible to represent this distribution more accurately, as shown on the left of Figure 2.2. This behaviour, along with other mathematical reasons, leads to more robust and accurate state estimators. The next section will discuss the relevant mathematics of Lie group theory.

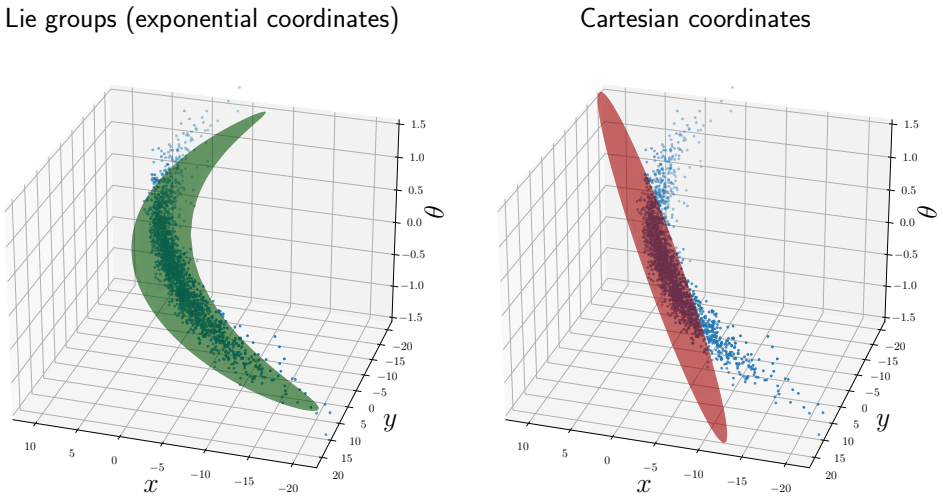

Figure 2.2: Distribution of robot poses, along with modelled representations using both exponential coordinates from Lie group theory, as well as traditional Cartesian coordinates.

2.3.1 Basic Definitions

Lie group theory is the mathematical formalism that can be used to generalize a robot's state definition into a more abstract object than a simple vector. A *group* $(G, \circ)$ is a set G, along with a *group operation* $\circ : G \times G \to G$, that for arbitrary $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in G$, satisfy the following axioms [38]:

- closure under $\circ$, meaning $\mathcal{X} \circ \mathcal{X} \in G$;
- there exists an identity element $\mathcal{I}$, defined such that $\mathcal{I} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{I} = \mathcal{X}$;
- there exists an inverse operation $\mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{I}$;
- the group operation is associative, $(\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z})$.

A Lie group is a smooth manifold whose elements simultaneously satisfy the above group axioms. The formal definition of a smooth manifold is beyond the scope of this thesis, where it suffices to think of them as abstract (hyper)-surfaces living in some higher dimensional space. For example, $\mathbf{C} \in SO(3)$ is a 3×3 matrix, hence represented by 9 numbers, but with 6 independent constraints (imposed by the fact that $\mathbf{C}^\mathsf{T}\mathbf{C} = \mathbf{1}$) that leave only 3 degrees of freedom. As a manifold, elements of $SO(3)$ can be thought of as belonging to a 3-dimensional surface living in 9-dimensional space. The key required property is the smoothness of the manifold, visualized by a lack of spikes or edges on this abstract surface, which allows for the definition of smooth functions $\mathcal{X}(t)$ with codomain entirely within the Lie group.

More concretely, this thesis will often deal with *matrix* Lie groups, which have elements as square, invertible matrices, and the group operation is regular matrix multiplication. Apart from $SO(n)$, the most common example is the special Euclidean group

$$SE(n) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} \;\middle|\; \mathbf{C} \in SO(n)\,, \mathbf{r} \in \mathbb{R}^n \right\}.$$

It is easy to verify that both $SO(n)$ and $SE(n)$ satisfy the group axioms under matrix multiplication, with the identity element simply being the identity matrix $\mathbf{1}$, and the inverse operation is the matrix inverse. For $\mathbf{X}(t) \in G$ belonging to matrix Lie group G, the relation $\mathbf{X}^{-1}(t)\mathbf{X}(t) = \mathbf{1}$ can be differentiated to give

$$\frac{\mathrm{d}\mathbf{X}^{-1}}{\mathrm{d}t}\mathbf{X} + \mathbf{X}^{-1}\dot{\mathbf{X}} = \mathbf{0}, \tag{2.19}$$

$$-\frac{\mathrm{d}\mathbf{X}^{-1}}{\mathrm{d}t}\mathbf{X} = \mathbf{X}^{-1}\dot{\mathbf{X}} \triangleq \boldsymbol{\Xi}, \tag{2.20}$$

with the argument of time omitted. The matrix $\boldsymbol{\Xi} \in \mathfrak{g}$ is said to belong to the *matrix Lie algebra* $\mathfrak{g}$, and has a form given by $\mathbf{X}^{-1}\dot{\mathbf{X}}$ for all matrix Lie groups. The Lie algebra $\mathfrak{g}$ forms a vector space and can be written as $\boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge$ where $\boldsymbol{\xi} \in \mathbb{R}^m$ and $(\cdot)^\wedge : \mathbb{R}^m \to \mathfrak{g}$ is linear. The

inverse of $(\cdot)^\wedge$ is denoted $(\cdot)^\vee : \mathfrak{g} \to \mathbb{R}^m$. Return now to (2.20) and rearrange slightly,

$$\dot{\mathbf{X}}(t) = \mathbf{X}(t)\boldsymbol{\xi}^\wedge, \tag{2.21}$$

where $\boldsymbol{\xi}^\wedge = \boldsymbol{\Xi} \in \mathfrak{g}$. If $\boldsymbol{\xi}$ is assumed to be constant, (2.21) is a matrix-valued linear ordinary differential equation with analytical solution given by

$$\mathbf{X}(t) = \mathbf{X}(0)\exp(t\boldsymbol{\xi}^\wedge), \tag{2.22}$$

where $\exp(\cdot)$ denotes the matrix exponential. Considering the case where $\mathbf{X}(0) = \mathbf{1}$ and arbitrarily fixing $t = 1$, (2.22) will give

$$\mathbf{X} = \exp(\boldsymbol{\xi}^\wedge)$$

meaning that the matrix exponential produces a group element given *any* element of its corresponding Lie algebra. Likewise, the matrix logarithm produces elements of the matrix Lie algebra given a group element

$$\boldsymbol{\xi}^\wedge = \log(\mathbf{X}).$$

There are two more important definitions that primarily appear in the derivations of Jacobians of function involving matrix Lie group elements. The first is the adjoint operator $\mathrm{Ad}_\mathbf{X} : \mathfrak{g} \to \mathfrak{g}$, defined as

$$\mathrm{Ad}_\mathbf{X}(\boldsymbol{\xi}^\wedge) = \mathbf{X}\boldsymbol{\xi}^\wedge\mathbf{X}^{-1}. \tag{2.23}$$

Since the adjoint is linear, there exists a corresponding matrix representation $\mathbf{Ad} : G \to \mathbb{R}^{m\times m}$ defined such that

$$\mathbf{Ad}(\mathbf{X})\boldsymbol{\xi} = (\mathbf{X}\boldsymbol{\xi}^\wedge\mathbf{X}^{-1})^\vee.$$

The second definition is the simple operator $(\cdot)^\odot : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ defined such that

$$\mathbf{a}^\wedge\mathbf{b} = \mathbf{b}^\odot\mathbf{a}.$$

For more abstract Lie groups, the Lie algebra is related to the group through the *exponential and logarithmic maps* $\exp : \mathfrak{g} \to G$, $\log : G \to \mathfrak{g}$. These coincide with matrix exponential and logarithm for matrix Lie groups, but have a more general definition which is beyond the scope of this thesis. Nevertheless, it is now possible to associate a group element with a vector $\boldsymbol{\xi} \in \mathbb{R}^m$ directly with

$$\mathcal{X} = \exp(\boldsymbol{\xi}^\wedge) \triangleq \mathrm{Exp}(\boldsymbol{\xi}), \qquad \boldsymbol{\xi} = \log(\mathcal{X})^\vee \triangleq \mathrm{Log}(\mathcal{X}),$$

where the shorthand notation $\mathrm{Exp} : \mathbb{R}^m \to G$ and $\mathrm{Log} : G \to \mathbb{R}^m$ has been defined. Concrete expressions for the above operations for the groups $SO(3)$, $SE(3)$, and $SE_2(3)$ can be found in [39, Ch. 2.2-2.4]. The software implementation used in this thesis can be found in Appendix B.

2.3.2 $\oplus$ and $\ominus$ operators

Estimation theory for vector-space states and Lie groups can be elegantly aggregated into a single mathematical treatment by defining generalized "addition" $\oplus : G \times \mathbb{R}^m \to G$ and "subtraction" $\ominus : G \times G \to \mathbb{R}^m$ operators, whose precise definitions will depend on the problem at hand. For example, possible implementations include

$$\begin{aligned} \mathcal{X} \oplus \delta\mathbf{x} &= \mathcal{X} \circ \mathrm{Exp}(\delta\mathbf{x}) && \text{(Lie group right)}, \\ \mathcal{X} \oplus \delta\mathbf{x} &= \mathrm{Exp}(\delta\mathbf{x}) \circ \mathcal{X} && \text{(Lie group left)}, \\ \mathbf{x} \oplus \delta\mathbf{x} &= \mathbf{x} + \delta\mathbf{x} && \text{(vector space)}, \end{aligned}$$

for addition and, correspondingly,

$$\begin{aligned} \mathcal{X} \ominus \mathcal{Y} &= \mathrm{Log}(\mathcal{Y}^{-1} \circ \mathcal{X}) && \text{(Lie group right)}, \\ \mathcal{X} \ominus \mathcal{Y} &= \mathrm{Log}(\mathcal{X} \circ \mathcal{Y}^{-1}) && \text{(Lie group left)}, \\ \mathbf{x} \ominus \mathbf{y} &= \mathbf{x} - \mathbf{y} && \text{(vector space)}, \end{aligned}$$

for subtraction. This abstraction is natural since a vector space technically qualifies as a Lie group with regular addition + as the group operation.

2.3.3 Gaussian distributions on Lie groups

As an example use of this abstraction, consider defining a normally-distributed Lie group element with mean $\bar{\mathcal{X}}$ and covariance $\mathbf{\Sigma}$, as done in [12], with

$$\mathcal{X} = \bar{\mathcal{X}} \circ \mathrm{Exp}(\delta\mathbf{x}), \qquad \delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}),$$

when using a right parameterization, or a similar definition for left parameterizations. This can alternatively be written in an abstract way, applicable to any group or vector space, with

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \delta\mathbf{x}, \qquad \delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}). \tag{2.24}$$

Moreover, given that $\delta\mathbf{x} = \mathcal{X} \ominus \bar{\mathcal{X}}$, it follows from (2.24) that

$$p(\mathcal{X}) = \eta \exp\left(-\frac{1}{2}(\mathcal{X} \ominus \bar{\mathcal{X}})^\intercal \mathbf{\Sigma}^{-1} (\mathcal{X} \ominus \bar{\mathcal{X}})\right) \triangleq \mathcal{N}_L(\bar{\mathcal{X}}, \mathbf{\Sigma}),$$

where η is a normalization constant that ensures $p(\mathcal{X})$ integrates to one over its domain, thus making it a valid probability density function. The definition of $\mathcal{N}_L(\bar{\mathcal{X}}, \mathbf{\Sigma})$ provides a generalized "Gaussian" distribution compatible with Lie group elements, and is useful for estimation problems.

2.3.4 Derivatives on Lie groups

Again following [40], the Jacobian of a function $f : G \to G$, taken with respect to $\mathcal{X}$ can be defined as

$$\left.\frac{Df(\mathcal{X})}{D\mathcal{X}}\right|_{\bar{\mathcal{X}}} \triangleq \left.\frac{\partial f(\bar{\mathcal{X}} \oplus \delta\mathbf{x}) \ominus f(\bar{\mathcal{X}})}{\partial \delta\mathbf{x}}\right|_{\delta\mathbf{x}=\mathbf{0}},$$

where it should be noted that the function $f(\bar{\mathcal{X}} \oplus \delta\mathbf{x}) \ominus f(\bar{\mathcal{X}})$ of $\delta\mathbf{x}$ has $\mathbb{R}^m$ as both its domain and codomain, and can thus be differentiated using any standard technique. In Appendix A, this is exploited to propose a Lie-group-compatible numerical differentiation technique that uses the complex step. This method is capable of calculating derivatives with arbitrary precision, thus being practically indiscernible from an analytical computation, except for the computation time.

With the above general definition of a derivative, it is easy to define the so-called *Jacobian of* G as $\mathbf{J} = D\mathrm{Exp}(\mathbf{x})/D\mathbf{x}$, where left/right group Jacobians are obtained with left/right definitions of $\oplus$ and $\ominus$.

2.3.5 Composite groups

A *composite group* is simply the concatenation of N other Lie groups $G_1, \ldots, G_N$ [40], with elements of the form

$$\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \ldots \mathcal{X}_N) \in G_1 \times \ldots \times G_N,$$

where $\times$ denotes the direct product of two groups. These elements form a new Lie group with group operation, inverse, identity, exponential map, and logarithmic map defined elementwise. For example,

$$\boldsymbol{\mathcal{X}} \circ \boldsymbol{\mathcal{Y}} = (\mathcal{X}_1 \circ \mathcal{Y}_1, \ldots, \mathcal{X}_N \circ \mathcal{Y}_N). \tag{2.25}$$

Furthermore, defining $\delta\mathbf{x} = [\delta\mathbf{x}_1^\mathsf{T} \ \ldots \ \delta\mathbf{x}_N^\mathsf{T}]^\mathsf{T}$ the $\oplus$, operator is given by

$$\boldsymbol{\mathcal{X}} \oplus \delta\mathbf{x} = (\mathcal{X}_1 \oplus \delta\mathbf{x}_1, \ \ldots \ , \mathcal{X}_N \oplus \delta\mathbf{x}_N) \tag{2.26}$$

and a similar definition applies to $\ominus$. Composite groups are a very practical construction, as they allow to easily make new groups out of known ones, and to possibly make composite groups of composite groups. As an example, let the state of the robot be both its pose $\mathbf{T} \in SE(3)$ and a sensor bias $\mathbf{b} \in \mathbb{R}^m$. The state can be represented with the composite element $\mathbf{x} = (\mathbf{T}, \mathbf{b})$, which has a straightforward elementwise definition of the various group and $\oplus, \ominus$ operations as per this section.

2.4 Maximum A Posteriori

Maximum a posteriori (MAP) is the standard approach taken in the robotics literature. Popular algorithms such as the extended Kalman filter (EKF), iterated EKF, sliding-window filter, and batch estimator can all be derived from a MAP approach, thus unifying them under a common theory. The traditional problem setup is to describe the robot's process model and measurement model in the standard form of

$$\begin{aligned} \mathcal{X}_k &= \mathbf{f}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}) \oplus \mathbf{w}_{k-1}, & \mathbf{w}_{k-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1}), \\ \mathbf{y}_k &= \mathbf{g}(\mathcal{X}_k) + \mathbf{v}_k, & \mathbf{v}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k), \end{aligned} \tag{2.27}$$

respectively. Here, $\mathbf{u}_{k-1}$ represents an arbitrary process model input, $\mathbf{y}_k$ represents the measurements, and both models are corrupted with "additive" Gaussian noise. As a notational convenience, let the shorthand $\mathcal{X}_{i:j} = (\mathcal{X}_i \ \dots \ \mathcal{X}_j)$ refer to a collection of arbitrary objects with indices in the range $[i, j]$, which may also form a composite group as per Section 2.3.5.

Given a history of measurements $\mathbf{y}_{0:K}$ with model as in (2.27), as well as a prior distribution $p(\mathcal{X}_0) = \mathcal{N}_L(\check{\mathcal{X}}_0, \check{\mathbf{P}}_0)$, the estimate of the state history $\boldsymbol{\mathcal{X}} = \mathcal{X}_{0:K}$ produced by the MAP approach is

$$\hat{\boldsymbol{\mathcal{X}}} = \arg\max_{\boldsymbol{\mathcal{X}}} p(\boldsymbol{\mathcal{X}} | \mathbf{y}_{0:K}). \tag{2.28}$$

Repeatedly applying Bayes' rule gives

$$\hat{\boldsymbol{\mathcal{X}}} = \arg\max_{\boldsymbol{\mathcal{X}}} \eta p(\mathcal{X}_0) \prod_{k=0}^{K} p(\mathbf{y}_k | \mathcal{X}_k) \prod_{k=1}^{K} p(\mathcal{X}_k | \mathcal{X}_{k-1}) \tag{2.29}$$

where η is a normalization constant. Based on the additive noise assumption in (2.27), it is justified to say that

$$p(\mathcal{X}_k | \mathcal{X}_{k-1}) = \mathcal{N}_L(\mathbf{f}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}), \mathbf{Q}_{k-1}), \tag{2.30}$$

$$p(\mathbf{y}_k | \mathcal{X}_k) = \mathcal{N}(\mathbf{g}(\mathcal{X}_k), \mathbf{R}_k). \tag{2.31}$$

Equivalently, minimizing the negative logarithm of (2.29) yields a nonlinear least-squares problem of the form

$$\hat{\boldsymbol{\mathcal{X}}} = \arg\max_{\boldsymbol{\mathcal{X}}} \frac{1}{2} \mathbf{e}(\boldsymbol{\mathcal{X}})^{\mathsf{T}} \mathbf{W} \mathbf{e}(\boldsymbol{\mathcal{X}}) \tag{2.32}$$

where

$$\mathbf{e}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{e}_0(\boldsymbol{\mathcal{X}}) \\ \mathbf{e}_{u,1}(\boldsymbol{\mathcal{X}}) \\ \vdots \\ \mathbf{e}_{u,K}(\boldsymbol{\mathcal{X}}) \\ \mathbf{e}_{y,0}(\boldsymbol{\mathcal{X}}) \\ \vdots \\ \mathbf{e}_{y,K}(\boldsymbol{\mathcal{X}}) \end{bmatrix}, \quad \begin{aligned} \mathbf{e}_0(\boldsymbol{\mathcal{X}}) &= \mathcal{X}_0 \ominus \check{\mathcal{X}}_0, \\ \mathbf{e}_{u,k}(\boldsymbol{\mathcal{X}}) &= \mathcal{X}_k \ominus \mathbf{f}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}), \\ \mathbf{e}_{y,k}(\boldsymbol{\mathcal{X}}) &= \mathbf{y}_k - \mathbf{g}(\mathcal{X}_k), \\ \mathbf{W} &= \operatorname{diag}(\check{\mathbf{P}}_0^{-1}, \mathbf{Q}_0^{-1}, \ldots, \mathbf{Q}_{K-1}^{-1}, \mathbf{R}_0^{-1}, \ldots, \mathbf{R}_K^{-1}). \end{aligned} \tag{2.33}$$

Using an on-manifold optimization approach, (2.32) can be solved by first parameterizing the state with $\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}} \oplus \delta\mathbf{x}$ and solving the problem

$$\delta\hat{\mathbf{x}} = \arg\max_{\delta\mathbf{x}} \mathbf{e}(\hat{\boldsymbol{\mathcal{X}}} \oplus \delta\mathbf{x})^\mathsf{T} \mathbf{W} \mathbf{e}(\hat{\boldsymbol{\mathcal{X}}} \oplus \delta\mathbf{x}). \tag{2.34}$$

Defining the Jacobian of the error vector as

$$\mathbf{H} \triangleq \left.\frac{D\mathbf{e}(\boldsymbol{\mathcal{X}})}{D\mathcal{X}}\right|_{\bar{\boldsymbol{\mathcal{X}}}} = \left.\frac{\partial \mathbf{e}(\hat{\boldsymbol{\mathcal{X}}} \oplus \delta\mathbf{x})}{\partial \delta\mathbf{x}}\right|_{\delta\mathbf{x}=\mathbf{0}}, \tag{2.35}$$

an approximate solution to (2.34) can be obtained by solving the Gauss-Newton system

$$(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})\delta\hat{\mathbf{x}} = \mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{e}(\hat{\boldsymbol{\mathcal{X}}}). \tag{2.36}$$

The above is iterated with $\hat{\boldsymbol{\mathcal{X}}} \leftarrow \hat{\boldsymbol{\mathcal{X}}} \oplus \delta\hat{\mathbf{x}}$ and typically initialized by propagating the prior mean $\check{\mathcal{X}}_0$ through the process model to compute the prior mean at each time step. A common approximation for the posterior covariance $\hat{\mathbf{P}}$ where $p(\mathcal{X}|\mathbf{y}) \approx \mathcal{N}_L(\hat{\boldsymbol{\mathcal{X}}}, \hat{\mathbf{P}})$ is given by $\hat{\mathbf{P}} = (\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})^{-1}$. The MAP estimation method is often referred to as the *batch estimator* when there are states at multiple time steps being estimated simultaneously.

2.4.1 Observability

Informally, a state-space system of the form (2.27) is said to be *observable* if, given a collection of inputs $\mathbf{u}_{0:K}$ and measurements $\mathbf{y}_{0:K}$, there exists a unique trajectory $\boldsymbol{\mathcal{X}}_k$ corresponding to those measurements. The notion of observability corresponds to the uniqueness of the solution $\hat{\boldsymbol{\mathcal{X}}}$ from (2.28), given a set of measurements and no prior knowledge [41]. In the nonlinear case, determining uniqueness is difficult in general [42], but it is still possible to make a statement of *local* uniqueness by requiring that the solution $\delta\hat{\mathbf{x}}$ to the linearized batch problem be unique. Following [43, Ch. 3.1], this should also be done with the prior term $p(\mathcal{X}_0)$ in (2.29) ignored. The uniqueness of $\delta\hat{\mathbf{x}}$ is guaranteed if the matrix $\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H}$ is full rank. Since $\mathbf{W}$ is always symmetric positive definite,

$$\operatorname{rank}(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H}) = \operatorname{rank}(\mathbf{H}^\mathsf{T}\mathbf{H}) = \operatorname{rank}(\mathbf{H}). \tag{2.37}$$

For the system given by (2.29), but with the prior term omitted, the Jacobian of the error is

$$\mathbf{H} = \begin{bmatrix} -\mathbf{F}_0 & \mathbf{1} & & \\ & \ddots & \ddots & \\ & & -\mathbf{F}_{K-1} & \mathbf{1} \\ -\mathbf{G}_0 & & & \\ & -\mathbf{G}_1 & & \\ & & \ddots & \\ & & & -\mathbf{G}_K \end{bmatrix}, \tag{2.38}$$

$$\mathbf{F}_k = \frac{D\mathbf{f}(\mathcal{X}_k, \mathbf{u}_k)}{D\mathcal{X}_k}, \tag{2.39}$$

$$\mathbf{G}_k = \frac{D\mathbf{g}(\mathcal{X}_k)}{D\mathcal{X}_k}. \tag{2.40}$$

Performing a variety of elementary column operations on (2.38), it can be shown that the rank of $\mathbf{H}$ in (2.38) is equivalent to the rank of

$$\mathcal{O} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1\mathbf{F}_0 \\ \vdots \\ \mathbf{G}_K\mathbf{F}_{K-1}\dots\mathbf{F}_0 \end{bmatrix}, \tag{2.41}$$

and hence local uniqueness requires that $\text{rank}(\mathcal{O}) = m$ where m is the degrees of freedom associated with a state $\mathcal{X}_k$.

2.5 Consistency of Bayesian Estimators

Many estimators assume that the state distribution is Gaussian and, accordingly, report both an estimated mean $\hat{\mathbf{x}}$ and covariance $\hat{\mathbf{P}}$ such that $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}})$. Given samples of the true state $\mathbf{x}$, it is frequently of interest to determine whether an estimator is statistically *consistent*. That is, whether the statistics reported by the estimator match the true statistics of the state. If $\mathbf{x}_k \in \mathbb{R}^n$ is Gaussian-distributed, then the following scalar random variable

$$\epsilon_k = (\mathbf{x}_k - \hat{\mathbf{x}}_k)^\mathsf{T} \hat{\mathbf{P}}_k^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k) \tag{2.42}$$

has a *chi-square distribution with n degrees of freedom* $\chi^2_n(\epsilon)$ and is called the *normalized estimation error squared* (NEES) [44]. A consistency test is formed by requiring that

$$P(\epsilon_k \in [r_1, r_2]) = \alpha$$

where $P(\cdot)$ denotes the probability and r_1, r_2 are the bounds of the $\alpha \in [0,1]$ probability concentration region for χ^2_n. In practice, this is checked by first evaluating the estimation error and computing NEES samples ϵ_k. Then, for a chosen α, the bounds $r_1,\ r_2$ can be computed through third-party software, and the portion of NEES samples that lie within $[r_1\ , r_2]$ is then verified to be close to α. If multiple estimation trials are conducted, typically as a Monte Carlo simulation, the N-run average NEES can be defined as

$$\bar{\epsilon}_k = \sum_{k=1}^{N} \frac{1}{N} \epsilon_k.$$

If the estimator is consistent, $N\bar{\epsilon}_k$ will be chi-squared distributed with Nn degrees of freedom, and an alternative consistency test can be constructed as

$$P(\bar{\epsilon}_k \in [r_1, r_2]) = \alpha$$

where r_1, r_2 now correspond to the α probability region of χ^2_{Nn}/N.

Chapter 3

Relative Position Estimation Between Two UWB Devices with IMUs

3.1 Summary

This chapter presents an algorithm for determining the three-dimensional relative position between two mobile robots, each using nothing more than a single ultra-wideband transceiver, an accelerometer, a rate gyro, and a magnetometer. A sliding window filter estimates the relative position at selected keypoints by combining the distance measurements with acceleration estimates, which each agent computes using an on-board attitude estimator. A key novelty is the keypoint selection strategy, which results in a positioning accuracy of less than 1 m in experiments, with the algorithm outperforming standard estimators.

3.2 Introduction

Determining a full three-dimensional relative position estimate from a single range measurement is impossible. There is an infinite set of relative positions that will produce the same range measurement. Hence, to realize an observable relative positioning solution, more information is required.

Range-based positioning is very commonly achieved by measuring distances to several landmarks with known positions, referred to as *anchors* when using UWB [18, 26]. However, the problem of estimating the relative position between moving robots, without any external reference, is much more difficult. Nevertheless, there are several papers that achieve relative positioning from range measurements, and do so by using additional sensors. For example, cameras are often used in addition to UWB [10], but vision requires substantial computational capabilities, which imposes a lower bound on the size of the agent. In [28], the presence of

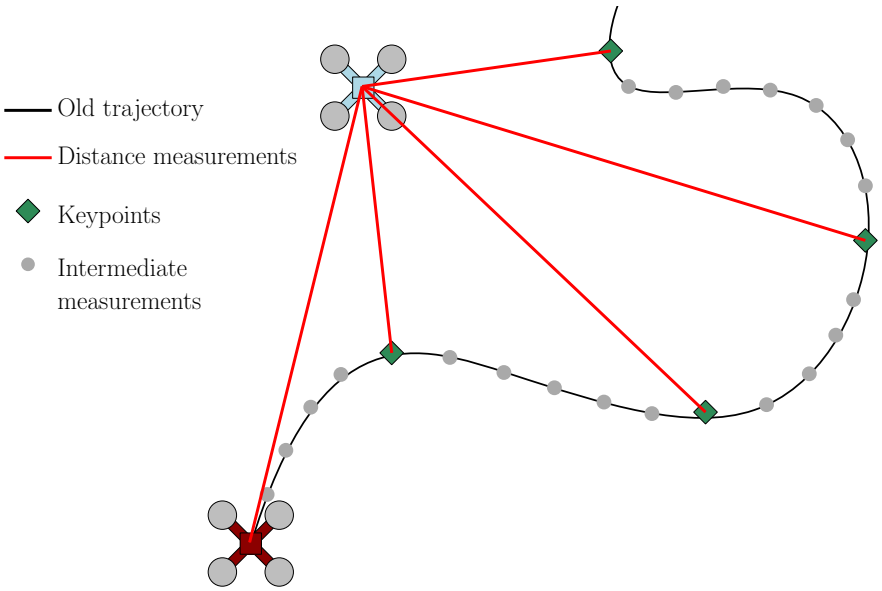

Figure 3.1: An example scenario where two quadrotors possess UWB modules, providing range measurements between them. Under sufficient motion, it is possible to determine the relative position between them in a common reference frame.

some agents with two UWB tags makes the relative positions observable. The observability analysis provided by [29] concludes that when velocity information is available in addition to single-range measurements, the relative position is observable provided that a *persistency of excitation* condition is satisfied, meaning that the agents must be in continuous relative motion. This is simulated in [31], where velocity measurements are assumed to be available, and experimented in [6], however GPS is used to provide displacement information for the agents. Cameras with optical flow are used to measure agent motion in [32, 33, 45], and [46] proposes a particle filter to estimate the relative positions of a group of mobile UWB devices, which also have IMUs. However, their solution requires a central server to perform the estimation task. The literature lacks three-dimensional solutions that use only UWB and IMU measurements.

The main contribution of this chapter is a three-dimensional relative position estimation algorithm for two mobile agents, each using nothing more than a single sensor measuring the range between them, and a 9-DOF IMU. No fixed infrastructure, GPS, cameras, or heavy computing is required, and the position is resolved in a known local frame, such as an East-North-Up reference frame. The proposed solution is easily decentralizable, and achieves sub-meter level accuracy in experiment. The algorithm requires persistency of excitation, meaning that the agents must be moving in a non-planar trajectory.

In the context of this chapter, $\mathcal{F}_b$ is associated with an agent's body frame, and $\mathcal{F}_a$ refers to some local common frame, such as an East-North-Up frame.

3.3 Sliding Window Filtering

This chapter will make use of the *sliding window filter* as a core component. The sliding window filter is a batch estimation framework with constant time and memory requirements, achieving so by *marginalizing out* older states [47, 48]. Consider a scenario where a robot travels until time step k_1, at which point it performs a full batch estimate of its state history, represented here as vector-space elements, to produce $\hat{\mathbf{x}}_{0:k_1}$. It then continues to travel until time step k_2, adding the new states $\mathbf{x}_{k_1+1:k_2}$ to its state history. The m oldest states $\mathbf{x}_{0:m-1}$ are then marginalized out, thus removing them from the optimization problem being solved at step k_2. The remaining states from the previous window's estimate are $\mathbf{x}_{m:k_1}$.

$$\underbrace{\mathbf{x}_0 \;\; \mathbf{x}_1 \;\; \cdots \;\; \mathbf{x}_{m-1} \;\; \mathbf{x}_m \;\; \cdots \;\; \mathbf{x}_{k_1}}_{\text{old window of length } K=k_1+1} \;\; \overbrace{\mathbf{x}_{k_1+1} \;\; \cdots \;\; \mathbf{x}_{k_2}}^{\text{new window of length } K=k_2-m+1}$$

The joint PDF of the new window, given the entire history of measurements until t_{k_2}, can be written as

$$p(\mathbf{x}_{m:k_2}|\check{\mathbf{x}}_0, \mathbf{u}_{0:k_2-1}, \mathbf{y}_{0:k_2}) = \eta p(\mathbf{x}_{m+1:k_2}|\mathbf{u}_{m:k_2-1}, \mathbf{y}_{m:k_2}, \mathbf{x}_m) p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1}). \tag{3.1}$$

The crux of the marginalization process is to determine an expression for $p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1})$, which takes the role of the new "prior" distribution of the new window. Performing marginalization properly, as opposed to naively removing the oldest states from the current estimation problem, is critical to maintaining an accurate, consistent estimator. To do this, consider instead the following PDF, for which an analytical expression is available,

$$p(\mathbf{x}_{0:m}|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1}) = \beta \exp(-\frac{1}{2}\mathbf{e}_m(\mathbf{x}_{0:m})^\mathsf{T} \mathbf{W}_m \mathbf{e}_m(\mathbf{x}_{0:m})), \tag{3.2}$$

where

$$\mathbf{e}_m(\mathbf{x}_{0:m}) = \begin{bmatrix} \mathbf{e}_0^\mathsf{T} & \mathbf{e}_{u,1}^\mathsf{T} & \cdots & \mathbf{e}_{u,m}^\mathsf{T} & \mathbf{e}_{y,0}^\mathsf{T} & \cdots & \mathbf{e}_{y,m-1}^\mathsf{T} \end{bmatrix}^\mathsf{T},$$
$$\mathbf{W}_m = \text{diag}(\check{\mathbf{P}}_0^{-1}, \mathbf{Q}_0^{-1}, \ldots, \mathbf{Q}_{m-1}^{-1}, \mathbf{R}_0^{-1}, \ldots, \mathbf{R}_{m-1}^{-1}),$$

β is a normalization constant, and the $\mathbf{e}_0, \mathbf{e}_{u,i}, \mathbf{e}_{y,i}$ terms are defined in (2.33), but written without arguments. This is, in general, not a Gaussian distribution due to the nonlinear nature of $\mathbf{e}_m(\cdot)$, but it can be approximated as one by linearizing $\mathbf{e}_m(\cdot)$ about a subset of the state estimates obtained from the previous window, being $\hat{\mathbf{x}}_{0:m}$. Substituting a first-order approximation of $\mathbf{e}_m(\cdot)$ into (3.2), and after some manipulation, the mean and covariance of

a Gaussian approximation to $p(\mathbf{x}_{0:m}|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1}) \approx \mathcal{N}(\boldsymbol{\mu}_{0:m}, \boldsymbol{\Sigma}_{0:m})$ can be shown to be

$$\begin{aligned}\boldsymbol{\mu}_{0:m} &= \begin{bmatrix} \boldsymbol{\mu}_{0:m-1} \\ \boldsymbol{\mu}_m \end{bmatrix} \\ &= \hat{\mathbf{x}}_{0:m} - (\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{H}_m)^{-1}\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{e}_m(\hat{\mathbf{x}}_{0:m}), \qquad (3.3)\\ \boldsymbol{\Sigma}_{0:m} &= \begin{bmatrix} \boldsymbol{\Sigma}_{0:m-1} & \boldsymbol{\Sigma}_{0:m-1,m} \\ \boldsymbol{\Sigma}_{m,0:m-1} & \boldsymbol{\Sigma}_m \end{bmatrix} = (\mathbf{H}_m^\mathsf{T}\mathbf{W}_m\mathbf{H}_m)^{-1}, \qquad (3.4)\end{aligned}$$

respectively, where

$$\mathbf{H}_m = \left.\frac{\partial \mathbf{e}_m(\mathbf{x})}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{0:m}}. \qquad (3.5)$$

It finally follows that

$$p(\mathbf{x}_m|\check{\mathbf{x}}_0, \mathbf{u}_{0:m-1}, \mathbf{y}_{0:m-1}) \approx \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m).$$

The specific step in which the marginalization occurred is when $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are extracted from (3.3) and (3.4), respectively. With an expression for the new prior now identified, one may return to (3.1) to construct a nonlinear least-squares problem in the exact same manner as the full batch problem, and solve it with the Gauss-Newton or Levenberg-Marquart algorithm.

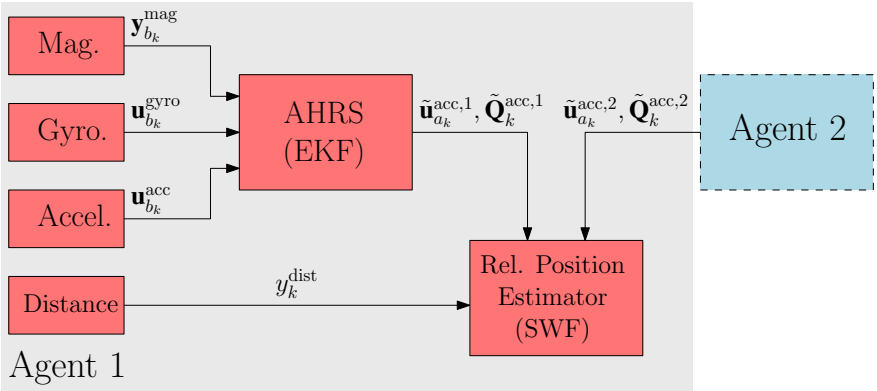

Figure 3.2: High-level diagram of the architecture of the proposed algorithm. Each agent contains an AHRS that uses on-board accelerometer, rate gyro, and magnetometer measurements.

3.4 Proposed Algorithm

3.4.1 High-level architecture

The proposed algorithm consists of two main components, with the architecture displayed graphically in Figure 3.2. Although each agent can execute the following algorithm, it will be explained from the perspective of Agent 1.

1. **Attitude & Heading Reference System (AHRS)**
   At each IMU measurement, the agents calculate their own attitudes relative to a common local frame, such as an East-North-Up reference frame, denoted $\mathcal{F}_a$. Using the attitude estimate, Agent 2 communicates an estimate of their translational acceleration, resolved in $\mathcal{F}_a$, along with a corresponding covariance.
2. **Relative Position Estimator (RPE)**
   Using the translational acceleration information of Agent 2 communicated to Agent 1, as well as a set of previous range measurements, a sliding window filter determines the position and velocity of Agent 1 relative to Agent 2 in $\mathcal{F}_a$. The high-frequency acceleration information of both agents is integrated between a set of optimized *keypoints*, which will be explained in Section 3.4.3.

The primary contribution of this chapter is the development and testing of the RPE. The choice of using a sliding window filter instead of a one-step-ahead filter, such as an extended Kalman filter (EKF), stems from the fact that position and velocity states are instantaneously unobservable given one range measurement. However, as will be shown in Section 3.4.4, the system is observable when multiple range measurements are used for state estimation, which is what the sliding window filter does.

The cascaded architecture involving the AHRS and the RPE is *loosely-coupled* due to the separation of attitude and translational state estimators. This choice comes with several advantages and disadvantages, when compared to a *tightly-coupled* framework that would estimate the attitude and translational states in one estimator. The main disadvantage is that any coupling information between the AHRS and RPE states is lost. This point represents the largest possible source of inaccuracy, as it results in the RPE performance being highly dependent on the accuracy of the attitude estimates. However, this cost is justified by the following advantages.

- The corrective step of the AHRS can be executed at an arbitrarily high frequency, thus enabling the incorporation of all possible magnetometer and accelerometer measurements for attitude correction.
- The processing can easily be distributed among the two agents, and communication requirements are heavily reduced as gyroscope and magnetometer measurements do not need to be shared.
- The relative position dynamics reduce to a linear model, which is convenient from both mathematical and computational points of view.

3.4.2 Attitude & Heading Reference System

The AHRS used in the presented simulations and experiments is an extended Kalman

filter (EKF) very similar to [22, Ch. 10], but with some modifications based on [14]. The AHRS estimates the agent attitude at time step k, denoted as $\hat{\mathbf{C}}_{ab_k}$. The estimate also has a corresponding covariance $\mathbf{P}_k^{\text{ahrs}}$, defined such that $\mathbf{C}_{ab_k} = \hat{\mathbf{C}}_{ab_k} \exp(\delta\boldsymbol{\phi}^\times)$, where $\delta\boldsymbol{\phi} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_k^{\text{ahrs}})$ and $\mathbf{C}_{ab_k}$ is the true agent attitude. The translational acceleration vector $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$, resolved in $\mathcal{F}_a$, can be calculated using the vehicle attitude and the raw accelerometer measurements $\mathbf{u}_{b_k}^{\text{acc}}$ using

$$\tilde{\mathbf{u}}_{a_k}^{\text{acc}} = \mathbf{C}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}} + \mathbf{g}_a, \tag{3.6}$$

where $\mathbf{g}_a$ is the gravity vector resolved in $\mathcal{F}_a$. However, a covariance associated with $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ is also required. Equation (3.6) is a nonlinear function of two random variables, $\mathbf{C}_{ab_k}$ and $\mathbf{u}_{b_k}^{\text{acc}}$, and as such the translational acceleration distribution can be approximated as $\tilde{\mathbf{u}}_{a_k}^{\text{acc}} \sim \mathcal{N}(\hat{\mathbf{C}}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}} + \mathbf{g}_a, \tilde{\mathbf{Q}}_k^{\text{acc}})$ where

$$\tilde{\mathbf{Q}}_k^{\text{acc}} = \mathbf{M}\mathbf{Q}_k^{\text{acc}}\mathbf{M}^\mathsf{T} + \mathbf{G}\mathbf{P}_k^{\text{ahrs}}\mathbf{G}^\mathsf{T},$$

and $\mathbf{Q}_k^{\text{acc}}$ is the covariance associated with the raw accelerometer measurements. The matrices $\mathbf{M} = \hat{\mathbf{C}}_{ab_k}$ and $\mathbf{G} = -\hat{\mathbf{C}}_{ab_k}\mathbf{u}_{b_k}^{\text{acc}^\times}$ are the Jacobians of (3.6) with respect to $\mathbf{u}_{b_k}^{\text{acc}}$ and $\delta\boldsymbol{\phi}$, respectively. Although not specified in the notation used in this section, each of the two agents computes their own, independent estimates of $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ and $\tilde{\mathbf{Q}}_k^{\text{acc}}$. They are both communicated to the relative position estimator, consisting of 3 numbers for the $\tilde{\mathbf{u}}_{a_k}^{\text{acc}}$ and 6 numbers for the symmetric covariance matrix $\tilde{\mathbf{Q}}_k^{\text{acc}}$.

3.4.3 Relative Position Estimator

Let $\mathbf{r}_{a_k}^{12}$ be the position of Agent 1, relative to Agent 2, resolved in $\mathcal{F}_a$. Let $\mathbf{v}_{a_k}^{12}$ be the velocity of Agent 1, relative to Agent 2, with respect to $\mathcal{F}_a$. The relative position estimator, which will be on-board Agent 1, estimates the state $\mathbf{x}_k = [\mathbf{r}_{a_k}^{12^\mathsf{T}} \ \mathbf{v}_{a_k}^{12^\mathsf{T}}]^\mathsf{T}$. The translational acceleration and corresponding covariances of Agents 1 and 2, being $(\tilde{\mathbf{u}}_{a_k}^{\text{acc},1}, \tilde{\mathbf{Q}}_k^{\text{acc},1})$ and $(\tilde{\mathbf{u}}_{a_k}^{\text{acc},2}, \tilde{\mathbf{Q}}_k^{\text{acc},2})$, are combined to produce a relative acceleration estimate, $\mathbf{u}_k \triangleq \tilde{\mathbf{u}}_{a_k}^{\text{acc},1} - \tilde{\mathbf{u}}_{a_k}^{\text{acc},2}$, with covariance $\tilde{\mathbf{Q}}_k = \tilde{\mathbf{Q}}_k^{\text{acc},1} + \tilde{\mathbf{Q}}_k^{\text{acc},2}$. This allows the relative position and velocity process model to be written as

$$\mathbf{x}_k = \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1}, \tag{3.7}$$

where $\Delta t_k = t_k - t_{k-1}$, $\mathbf{w}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$,

$$\mathbf{A}_{k-1} = \begin{bmatrix} \mathbf{1} & \Delta t_k\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \mathbf{B}_{k-1} = \begin{bmatrix} (\Delta t_k^2/2)\mathbf{1} \\ \Delta t_k\mathbf{1} \end{bmatrix},$$

$$\mathbf{Q}_{k-1} = \begin{bmatrix} \frac{1}{3}\Delta t_k^3\tilde{\mathbf{Q}}_{k-1} & \frac{1}{2}\Delta t_k^2\tilde{\mathbf{Q}}_{k-1} \\ \frac{1}{2}\Delta t_k^2\tilde{\mathbf{Q}}_{k-1} & \Delta t_k\tilde{\mathbf{Q}}_{k-1} \end{bmatrix}.$$

Furthermore, the RPE uses range measurements, which have a measurement model of the form

$$y_k^{\text{dist}} = \left\| \mathbf{r}_{a_k}^{12} \right\| + v_k, \tag{3.8}$$

where $v_k \sim \mathcal{N}(0, R_k)$. The Jacobian of (3.8) with respect to $\mathbf{x}_k$ is given by $\mathbf{C}_k = \begin{bmatrix} \boldsymbol{\rho}_k^{\mathsf{T}} & \mathbf{0} \end{bmatrix}$, where $\boldsymbol{\rho}_k = \mathbf{r}_{a_k}^{12} / \left\| \mathbf{r}_{a_k}^{12} \right\|$.

A naive implementation of the sliding window filter would result in including a state at every single measurement. With modern IMUs, this typically occurs at a frequency of 100 Hz - 1000 Hz, which can result in an enormous amount of states in the window if it were to span only a few seconds. The number of states in the window can be reduced by omitting all except a select few called *keypoints* [49]. The keypoints are identified as a set of K distinct time indices $\mathcal{K} = \{p_0, p_1, \ldots, p_{K-1}\}$ such that the states $\mathbf{x}_{p_0}, \mathbf{x}_{p_1}, \ldots, \mathbf{x}_{p_{K-1}}$ are the only states estimated inside the window. To do this, a process model of some sort is required that relates successive keypoint states.

A technique called *pre-integration* expresses the process model between two non-adjacent states $\mathbf{x}_i$ and $\mathbf{x}_j$ by a single relation. The use of the linear model in (3.7) makes pre-integration trivial, as direct iteration of (3.7) leads to

$$\mathbf{x}_j = \left(\prod_{k=i}^{j-1} \mathbf{A}_k \right) \mathbf{x}_i + \sum_{k=i}^{j-1} \left(\prod_{\ell=k+1}^{j-1} \mathbf{A}_\ell \right) \mathbf{B}_k \mathbf{u}_k, \tag{3.9}$$

$$\triangleq \mathbf{A}_{ji} \mathbf{x}_i + \mathbf{b}_{ji}. \tag{3.10}$$

Importantly, the terms $\mathbf{A}_{ji}$ and $\mathbf{b}_{ji}$ do not need to be recomputed at each Gauss-Newton iteration of the sliding window filter, since they are independent of the state estimate. This yields substantial computational savings compared to a case where the process model is too complicated to easily pre-integrate. With this pre-integration ability, states in the sliding window filter can be placed arbitrarily far apart in time with a very minor increase in computation time.

As such, this chapter constructs a sliding window filter where each state is a keypoint, successive keypoint states are related by the pre-integrated process model (3.10), and the measurement model at each keypoint is given by (3.8). The ability to place keypoints at arbitrary locations in an agent's state history now gives rise to an entire design problem, which is to design an appropriate keypoint selection strategy. This chapter will propose one of many possible strategies.

3.4.4 Observability Analysis

Evaluated at arbitrary keypoints, the observability rank condition given in (2.41) requires that

$$\text{rank}(\mathcal{O}) = 6, \tag{3.11}$$

where,

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{\rho}_{p_0} & \boldsymbol{\rho}_{p_1} & \boldsymbol{\rho}_{p_2} & \cdots & \boldsymbol{\rho}_{p_{K-1}} \\ \mathbf{0} & \Delta t_{p_1 p_0}\boldsymbol{\rho}_{p_1} & \Delta t_{p_2 p_0}\boldsymbol{\rho}_{p_2} & \cdots & \Delta t_{p_{K-1} p_0}\boldsymbol{\rho}_{p_{K-1}} \end{bmatrix},$$

$\boldsymbol{\rho}_k = \mathbf{r}^{12}_{a_k} / \left\|\mathbf{r}^{12}_{a_k}\right\|$, and $\Delta t_{ji} = t_j - t_i$. To ensure observability, the keypoints must be chosen to satisfy (3.11).

Theorem 3.4.1. Sufficient conditions to satisfy (3.11) are

1. $K \geq 6$,
2. $\Delta t_{p_1 p_0} \neq \Delta t_{p_2 p_0} \neq \ldots \Delta t_{p_K p_0} \neq 0$,
3. and that there exists two non-intersecting subsets of $\{\boldsymbol{\rho}_{p_0},\ \boldsymbol{\rho}_{p_1},\ \ldots,\ \boldsymbol{\rho}_{p_{K-1}}\}$, each composed of 3 linearly independent elements.

Proof. Satisfying the rank condition in (3.11) amounts to showing that there are 6 linearly independent columns in the matrix $\mathcal{O}$. Since there are K columns in $\mathcal{O}$, $K \geq 6$, which is Condition 1, is immediately required so that there are at least 6 columns. If the theorem is proven for $K = 6$, then it also holds for $K > 6$, as the addition of columns will not affect the linear independence of the first six. As such, using a change in notation for brevity, the rank condition in (3.11) requires that

$$\mathcal{O} = \begin{bmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \boldsymbol{\rho}_3 & \boldsymbol{\rho}_4 & \boldsymbol{\rho}_5 \\ \mathbf{0} & \Delta t_1\boldsymbol{\rho}_1 & \Delta t_2\boldsymbol{\rho}_2 & \Delta t_3\boldsymbol{\rho}_3 & \Delta t_4\boldsymbol{\rho}_4 & \Delta t_5\boldsymbol{\rho}_5 \end{bmatrix}$$

have rank 6. Without loss of generality, assume $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2\}$ are linearly independent. It immediately follows that the first three columns of $\mathcal{O}$ are linearly independent. Consider now $\boldsymbol{\rho}_3$, again without loss of generality. Since $\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2 \in \mathbb{R}^3$, $\boldsymbol{\rho}_3$ is linearly dependent on $\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2$ and can be written as

$$\boldsymbol{\rho}_3 = \alpha_0\boldsymbol{\rho}_0 + \alpha_1\boldsymbol{\rho}_1 + \alpha_2\boldsymbol{\rho}_2. \tag{3.12}$$

If the fourth column of $\mathcal{O}$ were linearly dependent on the first three, then there would exist $\beta_0, \beta_1, \beta_2$ such that

$$\begin{bmatrix} \boldsymbol{\rho}_3 \\ \Delta t_3\boldsymbol{\rho}_3 \end{bmatrix} = \beta_0 \begin{bmatrix} \boldsymbol{\rho}_0 \\ \mathbf{0} \end{bmatrix} + \beta_1 \begin{bmatrix} \boldsymbol{\rho}_1 \\ \Delta t_1\boldsymbol{\rho}_1 \end{bmatrix} + \beta_2 \begin{bmatrix} \boldsymbol{\rho}_2 \\ \Delta t_2\boldsymbol{\rho}_2 \end{bmatrix}. \tag{3.13}$$

Substituting (3.12) into (3.13) yields

$$\begin{bmatrix} \alpha_0\boldsymbol{\rho}_0 + \alpha_1\boldsymbol{\rho}_1 + \alpha_2\boldsymbol{\rho}_2 \\ \Delta t_3(\alpha_0\boldsymbol{\rho}_0 + \alpha_1\boldsymbol{\rho}_1 + \alpha_2\boldsymbol{\rho}_2) \end{bmatrix} = \beta_0 \begin{bmatrix} \boldsymbol{\rho}_0 \\ \mathbf{0} \end{bmatrix} + \beta_1 \begin{bmatrix} \boldsymbol{\rho}_1 \\ \Delta t_1\boldsymbol{\rho}_1 \end{bmatrix} + \beta_2 \begin{bmatrix} \boldsymbol{\rho}_2 \\ \Delta t_2\boldsymbol{\rho}_2 \end{bmatrix}. \tag{3.14}$$

The first three lines of (3.14) immediately require that $\alpha_0 = \beta_0$, $\alpha_1 = \beta_1$, $\alpha_2 = \beta_2$, which reduces the last three lines of (3.14) to

$$\Delta t_3\alpha_0\boldsymbol{\rho}_0 + \Delta t_3\alpha_1\boldsymbol{\rho}_1 + \Delta t_3\alpha_2\boldsymbol{\rho}_2 = \alpha_1\Delta t_1\boldsymbol{\rho}_1 + \alpha_2\Delta t_2\boldsymbol{\rho}_2. \tag{3.15}$$

Hence, the fourth column of $\mathcal{O}$ is linearly dependent on the first three if and only if $\alpha_0 = 0$, $\Delta t_3 = \Delta t_2$, $\Delta t_3 = \Delta t_1$. Condition 2 in Theorem 3.4.1 means $\Delta t_3 \neq \Delta t_2 \neq \Delta t_1$, hence proving the linear independence of the fourth column of $\mathcal{O}$ from the first three. Identical proofs show that the fifth and sixth columns of $\mathcal{O}$ are also linearly independent from the first three. If additionally $\{\boldsymbol{\rho}_3, \boldsymbol{\rho}_4, \boldsymbol{\rho}_5\}$ are linearly independent, then the fourth, fifth, and sixth columns of $\mathcal{O}$ will be linearly independent from each other, in addition to each being linearly independent from the first three. Hence, all 6 columns of $\mathcal{O}$ will be linearly independent, giving it a rank of 6. Since this proof was done for a completely arbitrary choice of linearly independent sets $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \boldsymbol{\rho}_2\}$ and $\{\boldsymbol{\rho}_3, \boldsymbol{\rho}_4, \boldsymbol{\rho}_5\}$, it is only required that there exists two distinct non-intersecting subsets of $\{\boldsymbol{\rho}_0, \boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_K\}$, each containing 3 linearly independent elements, which is Condition 3. □

The conditions of Theorem 3.4.1 result in the requirement that the agents' relative motion be non-planar, and that keypoints are at distinct times.

3.4.5 Keypoint Selection Strategy

The keypoint selection strategy aims to find a suitable keypoint index set $\mathcal{K}$. The proposed strategy is a greedy algorithm, which places keypoints one-by-one, searching linearly through all possible candidate keypoint locations, and selecting the location with the lowest cost. The chosen cost function is based on the *geometric dilution of precision* (GDOP) [22, Ch. 8.5], and is given by

$$J(\mathcal{K}) = \operatorname{tr}\left((\mathbf{D}(\mathcal{K})^{\mathsf{T}}\mathbf{D}(\mathcal{K}))^{-1}\right) + \gamma \sum_{i=1}^{|\mathcal{K}|} \Delta t_{p_i p_{i-1}}, \tag{3.16}$$

where $\mathbf{D}(\mathcal{K}) = \begin{bmatrix} \boldsymbol{\rho}_{p_0} & \boldsymbol{\rho}_{p_1} & \boldsymbol{\rho}_{p_2} & \cdots & \boldsymbol{\rho}_{p_{|\mathcal{K}|}} \end{bmatrix}^{\mathsf{T}}$, $|\mathcal{K}|$ denotes the number of elements in $\mathcal{K}$, and γ is a tuning parameter. The first term in (3.16) is the square of the GDOP, also noting that the form of the $\mathbf{D}(\mathcal{K})$ matrix shows a natural similarity to the observability matrix shown in (3.11). Minimizing this first term avoids a keypoint selection that will make the observability matrix close to rank deficient. The second term in (3.16) penalizes the keypoints for being

Algorithm 1 Greedy Keypoint Selection. Given the current time step k, the total number of keypoints K, and the keypoint index set of the previous window $\mathcal{K}_{\text{old}}$, calculate the new keypoint index set $\mathcal{K}$. A set of candidate indices $\mathcal{C}$ is maintained.

1: **function** GETKEYPOINTS($k, K, \mathcal{K}_{\text{old}}$)
2: $\quad\mathcal{K} \leftarrow \{k-3, k-2, k-1, k\}$
3: $\quad\mathcal{C} \leftarrow \mathcal{K}_{\text{old}} \cup \{\max(\mathcal{K}_{\text{old}})+1, \ldots, k-4\}$
4: $\quad$**for** $i = 1, \ldots, K-4$ **do**
5: $\quad\quad J_{\text{best}} \leftarrow \infty$
6: $\quad\quad$**for all** $p \in \mathcal{C}$ **do**
7: $\quad\quad\quad\mathcal{K}_{\text{temp}} \leftarrow \mathcal{K} \cup \{p\}$
8: $\quad\quad\quad$**if** $J(\mathcal{K}_{\text{temp}}) \leq J_{\text{best}}$ **then**
9: $\quad\quad\quad\quad J_{\text{best}} \leftarrow J(\mathcal{K}_{\text{temp}})$
10: $\quad\quad\quad\quad p_{\text{best}} \leftarrow p$
11: $\quad\quad\quad$**end if**
12: $\quad\quad$**end for**
13: $\quad\quad\mathcal{K} \leftarrow \mathcal{K} \cup \{p_{\text{best}}\}$
14: $\quad\quad\mathcal{C} \leftarrow \mathcal{C} \setminus \{p_{\text{best}}\}$
15: $\quad$**end for**
16: $\quad$**return** $\mathcal{K}$
17: **end function**

placed too far apart in time. Excessively old keypoints will require long pre-integration between their adjacent states, which increases the covariance of the estimate. The tuning parameter γ is used to balance the two terms, and its effect is illustrated in Figure 3.3.

Finally, it is also possible to recursively compute the matrix $\mathbf{\Lambda} = (\mathbf{D}(\mathcal{K})^{\mathsf{T}}\mathbf{D}(\mathcal{K}))^{-1}$, which avoids recomputing the inverse when a new keypoint is added to $\mathcal{K}$. Defining $\tilde{\mathcal{K}} = \mathcal{K} \cup \{p\}$, it can be shown that

$$\left(\mathbf{D}(\tilde{\mathcal{K}})^{\mathsf{T}}\mathbf{D}(\tilde{\mathcal{K}})\right)^{-1} = \left(\mathbf{D}(\mathcal{K})^{\mathsf{T}}\mathbf{D}(\mathcal{K}) + \boldsymbol{\rho}_p\boldsymbol{\rho}_p^{\mathsf{T}}\right)^{-1} \tag{3.17}$$

$$= \mathbf{\Lambda} - \frac{\mathbf{\Lambda}\boldsymbol{\rho}_p\boldsymbol{\rho}_p^{\mathsf{T}}\mathbf{\Lambda}}{1 + \boldsymbol{\rho}_p^{\mathsf{T}}\mathbf{\Lambda}\boldsymbol{\rho}_p}, \tag{3.18}$$

where, in (3.18), the Woodbury matrix identity is used. Equation (3.18) is particularly useful when evaluating the **if** statement in line 8 of Algorithm 1, which shows the details of the proposed keypoint placement strategy. The algorithm initializes $\mathcal{K}$ with the 4 most recent indices in order to produce an invertible $(\mathbf{D}(\mathcal{K})^{\mathsf{T}}\mathbf{D}(\mathcal{K}))^{-1}$ matrix, after which the cost function naturally leads to a selection of older keypoints.

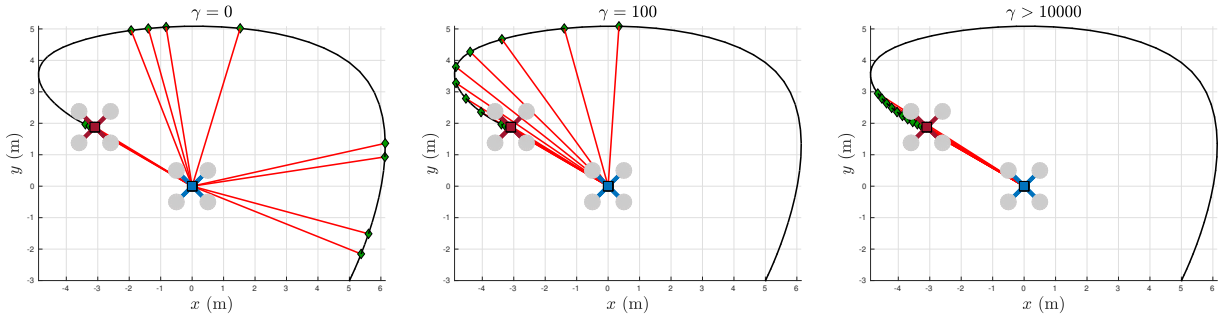

Figure 3.3: Effect of the parameter γ on the keypoint selection with $K = 12$, and the old trajectory shown as the black line. **Left:** With $\gamma = 0$, the algorithm greedily minimizes the GDOP, and results in keypoints placed far apart in time. **Middle:** With a moderate value for γ, the algorithm obtains a balance between pure GDOP minimization, and making all the keypoints as close as possible. **Right:** With a large value for γ, the minimal cost is the solution with the keypoints as close as possible, and the "plain vanilla" sliding window filter is recovered.

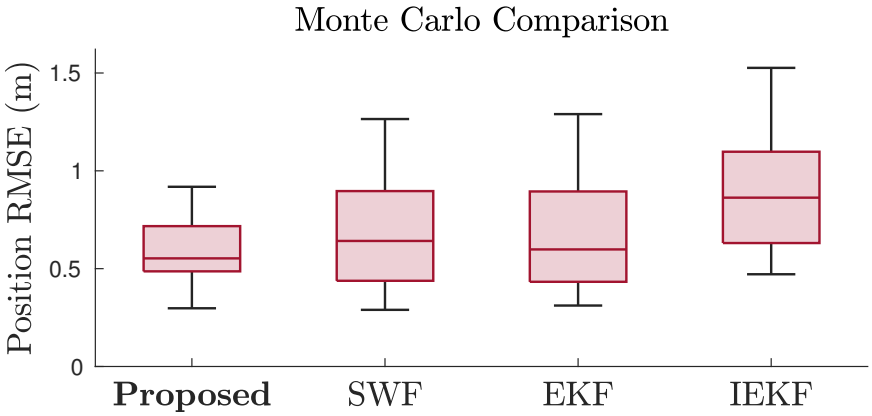

Figure 3.4: Results of 200 Monte Carlo trials. The proposed method offers an enhancement over the "plain vanilla" implementation of the sliding window filter (SWF), which simply places the keypoints at the most recent range measurements, as well as a standard extended Kalman filter (EKF), and an Iterated EKF (IEKF).

3.4.6 Multi-robot Scenario

Although this chapter focuses on a scenario with two agents, one way to extend the proposed algorithm to multiple agents is to naively copy the estimator for each pair of agents. In any case, it is possible to leverage the proposed framework to drastically simplify the inter-agent communication. By recalling that $\mathbf{u}_k = \tilde{\mathbf{u}}_{a_k}^{\text{acc},1} - \tilde{\mathbf{u}}_{a_k}^{\text{acc},2}$, it is straightforward to show that $\mathbf{b}_{ji}$ in (3.10) can be written as

$$\mathbf{b}_{ji} = \mathbf{b}_{ji}^{\text{acc},1} - \mathbf{b}_{ji}^{\text{acc},2},$$

where $\mathbf{b}_{ji}^{\text{acc},1}$ and $\mathbf{b}_{ji}^{\text{acc},2}$ only depend on measurements from Agents 1 and 2, respectively. This has a significant implication: the agents do not need to communicate their acceleration measurements at high frequency, but can instead preintegrate them in advance to produce

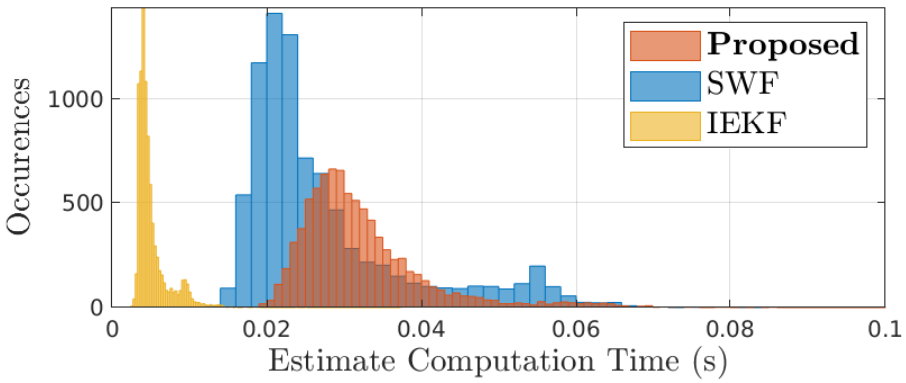

Figure 3.5: Histogram of the computation time of the different algorithms, when simulated on a laptop with an Intel Core i7-9750H CPU.

$\mathbf{b}_{ji}^{\text{acc},1}$ and $\mathbf{b}_{ji}^{\text{acc},2}$, then communicate these quantities at an arbitrarily lower frequency. This is of substantial practical interest, as communicating all high-frequency IMU measurements could quickly exceed the inter-agent communication capacity, especially for multi-robot scenarios.

3.5 Simulation Results

The proposed algorithm is tested in simulation, and compared against a "plain vanilla" sliding window filter (*Vanilla SWF* or *SWF* in the figures), which simply uses the K most recent range measurements as the keypoint times, as shown on the right of Figure 3.3. A comparison is also made to using a standard EKF as the relative position estimator. The simulation is repeated for 200 Monte Carlo trials, where each trial uses a different, randomized trajectory. Table 3.1 shows the values used when generating zero-mean Gaussian noise on each sensor.

Figure 3.4 shows a box plot of the position root-mean-squared error (RMSE) for the 200 Monte Carlo trials, where the RMSE $= \sqrt{(1/N)\sum_{k=1}^{N} \mathbf{e}_{r_k}^{\mathsf{T}} \mathbf{e}_{r_k}}$, $\mathbf{e}_{r_k} = \hat{\mathbf{r}}_{a_k}^{12} - \mathbf{r}_{a_k}^{12}$.

Using the greedy keypoint placement scheme shows a 9% reduction in average position RMSE when compared to the Vanilla SWF, 7% compared to the EKF, and 32% compared to the iterated EKF [43, Ch. 4.2.5]. An interesting observation is that EKF can perform as well, if not better than the SWF and the iterated EKF (IEKF). The SWF and IEKF both both iterate the state estimate until a locally minimal least-squares cost is found, while the EKF does not. This implies that there exists a state with lower overall process and measurement error, yet higher true estimation error. Many instances have been observed where the least-squares cost decreases while true estimation error increases, a possible consequence of the non-unique or unobservable nature of this problem. The authors therefore suspect that the

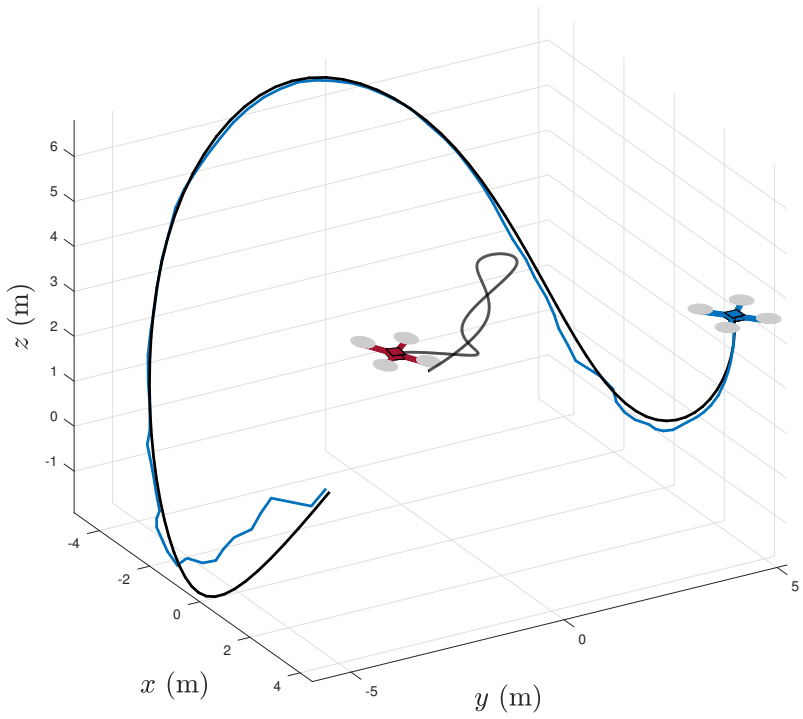

Figure 3.6: Simulation of the proposed estimation algorithm. The ground truth trajectory is shown in black, and the estimated trajectory in blue.

iterative algorithms can occaisionally iterate towards incorrect local minimums, thus iterating away from the true state. The proposed algorithm seems to be less prone to this issue.

The proposed algorithm takes an average of 0.035 s per estimate computation, whereas the Vanilla SWF takes 0.027 s, the IEKF takes 0.005 s, and the EKF takes 0.0006 s. A histogram of the computation time is shown in Figure 3.5. It is worth mentioning that the computation times were evaluated in MATLAB, and the implementations were not optimized for performance. Large speed improvements could be obtained by using a compiled language.

3.6 Experimental Results

The proposed algorithm is also tested in a real experiment, with two different hardware setups, both shown in Figure 3.7. The first consists of a Raspberry Pi 4B, with an LSM9DS1 9-DOF IMU and a Pozyx UWB Developer Tag, providing range measurements to an identical device. Although the cost of the actual sensors embedded in one of these prototypes is $15 USD, each cost a total of approximately $270 USD. However, this is mainly due to the cost of the Raspberry Pi 4B, the battery pack, and the Pozyx Developer Tag. The second setup consists of a Pozyx Developer tag mounted to quadrotors possessing a Pixhawk 4, which provides IMU and magnetometer data. In both setups, two identical devices are randomly

Table 3.1: Noise properties for single-range and IMU simulation

Specification	Value	Units
Magnetometer std. dev.	1	μF
Gyroscope std. dev.	0.001	rad/s
Accelerometer std. dev.	0.01	m/s^2
Distance std. dev	0.1	m
Initial position std. dev.	0.8	m
Initial velocity std. dev.	0.1	m/s
Initial attitude std. dev.	0.001	red
Accel/Gyro/Mag frequency	100	Hz
Distance frequency	10	Hz
RPE window size K	20	-
RPE keypoint selection parameter γ	100	-
RPE frequency	10	Hz

moved around a room by hand, in an overall volume of roughly 5 m $\times$ 4 m $\times$ 2 m. When using the quadrotors, the motors are spinning without propellors to examine the effect of vibrations and magnetic interference. Ground truth position and attitude measurements are collected using an OptiTrack optical motion capture system.

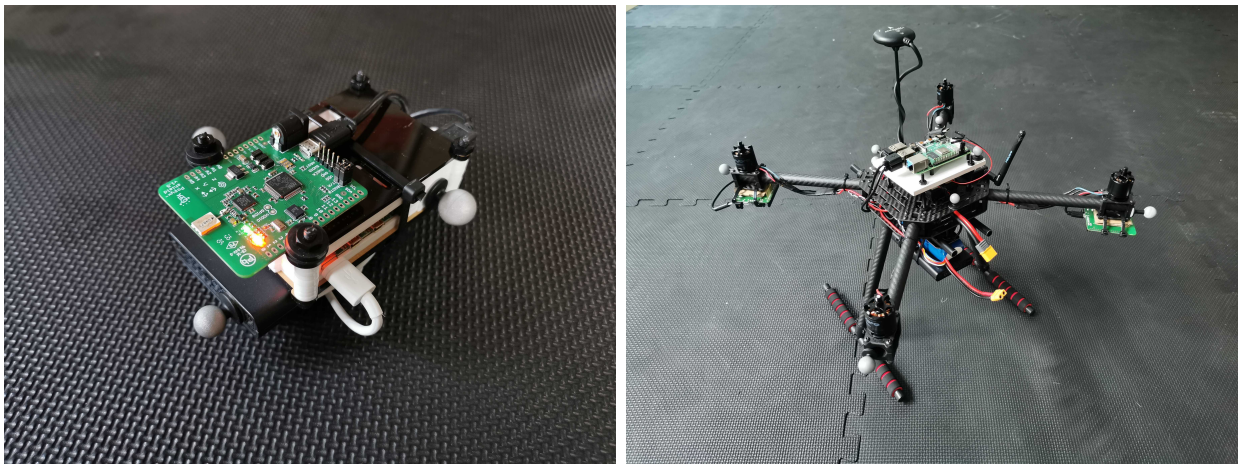

Figure 3.7: **Left:** Pozyx UWB Developer Tag mounted to a Raspberry Pi 4B, in a case. **Right:** Pozyx UWB Developer Tag mounted to a quadrotor with a Pixhawk 4.

Table 3.2 shows the position RMSE of the different algorithms, for each trial. The proposed method outperforms "plain vanilla" implementations of the SWF and the EKF by a large margin, in all of the trials, showing the value of the greedy keypoint selection strategy. Figure 3.8 shows the position estimation error of the proposed method, and that the estimate stays within the $\pm 3\sigma$ confidence bounds, except for two very brief moments. Finally, Figure 3.9 shows the norm of the position estimation error in one of the trials. During the periods of large error, between 20 and 40 seconds in Figure 3.9, the least squares cost for all three

algorithms did not dramatically increase relative to other periods. This implies that the EKF and the Vanilla SWF have converged to ambiguous states which also have low process and measurement error, again a possible consequence of unobservability.

Table 3.2: Position RMSE of Experimental Trials

	Proposed	**Vanilla SWF**	**EKF**
Trial 1 (one static agent)	**0.68 m**	1.41 m	2.11 m
Trial 2 (both moving)	**0.55 m**	1.22 m	1.71 m
Trial 3 (both moving)	**0.86 m**	1.56 m	2.69 m
Trial 4 (spinning motors)	**0.87 m**	1.34 m	1.12 m

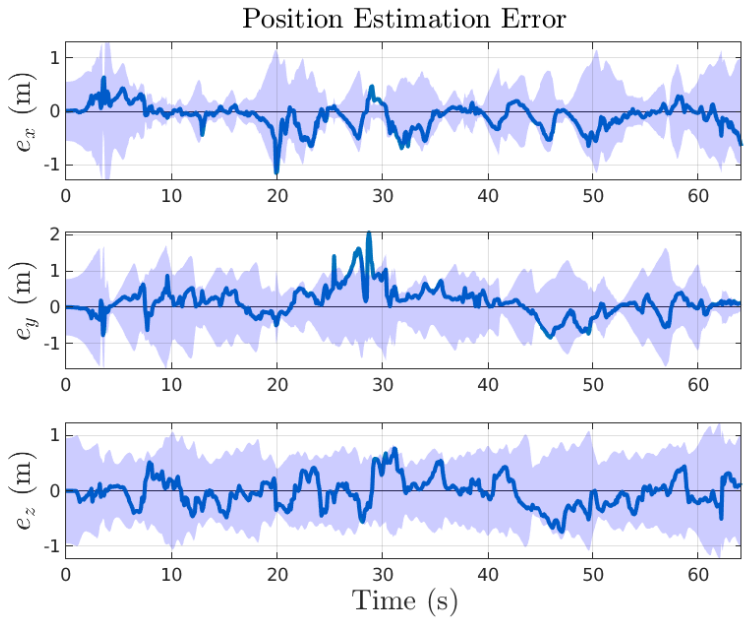

Figure 3.8: Components of the position estimation error during an experimental trial, using the greedy keypoint placement method. The error compared to ground truth is shown as the blue line, with the shaded area representing the $\pm 3\sigma$ confidence bounds.

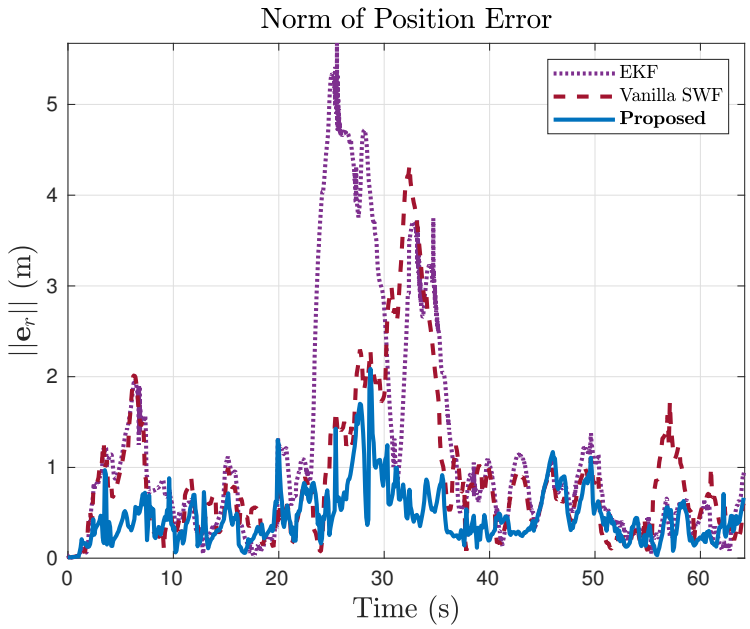

Figure 3.9: Comparison of the norm of the position estimation error $\mathbf{e}_r$, during an experimental trial.

3.7 Conclusion and Future Work

This chapter shows that it is possible to estimate the relative position of one UWB-equipped agent relative to another, provided each agent is endowed with a 9-DOF IMU and there is sufficient motion between the agents. The proposed algorithm shows improvement over standard estimators in both simulation and experiment, where there is 2- to 3-fold reduction in positioning error. This gain is largely achieved due to the ability of choosing specific "keypoints" in the previous state history, which is enabled by preintegration, and is an idea inspired by the SLAM literature [49, 50]. The keypoint selection strategy is a subproblem within this estimation task, and the proposed solution uses a greedy algorithm that attempts to minimize the geometric dilution of precision, as well as the preintegration duration.

While this algorithm is promising, there are a number of important ways in which it could be improved.

- **Eliminating the magnetometer.**
  The experiments from this chapter made it clear that the magnetometer is an extremely problematic sensor to use indoors, due to magnetic perturbations. This work, unfortunately, relies on a magnetometer so that each robot may estimate their attitude relative to a common world reference frame, such as an East-North-Up reference frame. This sensor could be eliminated by using the robots's body frames themselves as the reference

frames, eliminating the need for a common world reference frame entirely.

- **Using a tightly-coupled formulation.**
  In this chapter, a seperate AHRS was used for due to the convenience associated with the modularity, and so that magnetometer measurements could be fused into the attitude estimate at high frequency. However, this loosely-coupled formulation theoretically induces a performance loss as cross-correlations between attitude and translational states are completely neglected. In a new version of this algorithm, a suggested alternative is to use relative extended poses to represent the state, which would have the form

$$\mathbf{T}_{12} = \begin{bmatrix} \mathbf{C}_{12} & \mathbf{v}_1^{21} & \mathbf{r}_1^{21} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SE_2(3), \tag{3.19}$$

  where $SE_2(3)$ is referred to as the group of "extended" poses [51]. Using this formulation would also address the above point on eliminating the magnetometer, since any reference to a world frame is now completely eliminated. Furthermore, tightly-coupled IMU preintegration is a solved problem, and therefore it would still be possible to use the proposed keypoint selection with this new state defintion.

- **Using an improved cost function for keypoint selection.**
  The cost function in (3.16) is heuristically designed in an attempt to reduce the estimation error variance. However, a cost function that more directly reflects the estimation variance would be

$$J(\mathbf{x}) = -\log\det(\mathbf{H}(\mathbf{x})^{\mathsf{T}}\mathbf{W}\mathbf{H}(\mathbf{x})). \tag{3.20}$$

  It turns out that $\mathbf{H}(\mathbf{x})^{\mathsf{T}}\mathbf{W}\mathbf{H}(\mathbf{x})$, when evaluated at the true state, is equivalent to the *Fisher information matrix*, and its inverse represents a lower bound on the estimation variance known as the *Cramer-Rao bound*. Hence, maximizing the Fisher information could lead to lower-variance estimates, and to an entire class of general estimators called *Fisher-information-maximizing sliding window filters*. The main suspected difficulty with this approach, however, is that this new cost would be slower to compute than (3.16), and hence either a clever computation technique or other heuristic must be employed.

- **Incorporating zero-velocity updates.**
  Any improvement to the motion prediction based on IMU measurements is expected to significantly improve the final estimation accuracy. As such, especially for ground robots or hand-held devices, the IMU prediction could be improved by detecting moments of near-zero velocity [52–54]. These moments could then be incorporated as

pseudomeasurements in the sliding-window filter.

However, even if all these improvements were implemented, there would still be the requirement of persistent non-planar relative motion between the agents. This is impractical, since if there are long static or planar moments, the estimator could diverge. This is due to the fact that such a situation causes the state to become unobservable, thus leading to high estimation error variance, and hence eventually to catastrophically high linearization errors. The next chapter will attempt to deal with these situations of very high uncertainty.

Chapter 4

Localization with Directional Coordinates

4.1 Summary

A coordinate system is proposed that replaces the usual three-dimensional Cartesian x, y, z position coordinates, for use in robotic localization applications. The coordinate system consists of a range and direction represented by an element of $SO(3)$. Range, azimuth, and elevation measurement models become greatly simplified, and, unlike spherical coordinates, the proposed coordinates do not suffer from the same kinematic singularities and angle wrap-around. When compared to Cartesian coordinates, the proposed coordinate system results in a significantly enhanced ability to represent the true distribution of robot positions, ultimately leading to large improvements in state estimation consistency.

4.2 Introduction

As highlighted in Chapter 3, performing relative position estimation from single-range measurements requires persistent, non-planar relative motion between the two robots. If this condition is not met, the problem becomes unobservable, and the estimator is prone to divergence.

The original line of thinking when pursuing the idea in this chapter was to design an estimator that could maintain "consistency despite unobservability," and thus hopefully not diverge during static moments. Ideally, the reported variance of this estimator would simply increase in a statistically-consistent way during static moments, and decrease when there is motion. The goal was to design an estimator that can simply report that such an unobservable situation is occurring and continue operating, as opposed to breaking down entirely. In this regard, the goal was not met, but the idea has shown to be useful for a slightly different problem: when directional information is available in addition to the range.

Specifically, this chapter considers the problem of estimating the position of a robot,

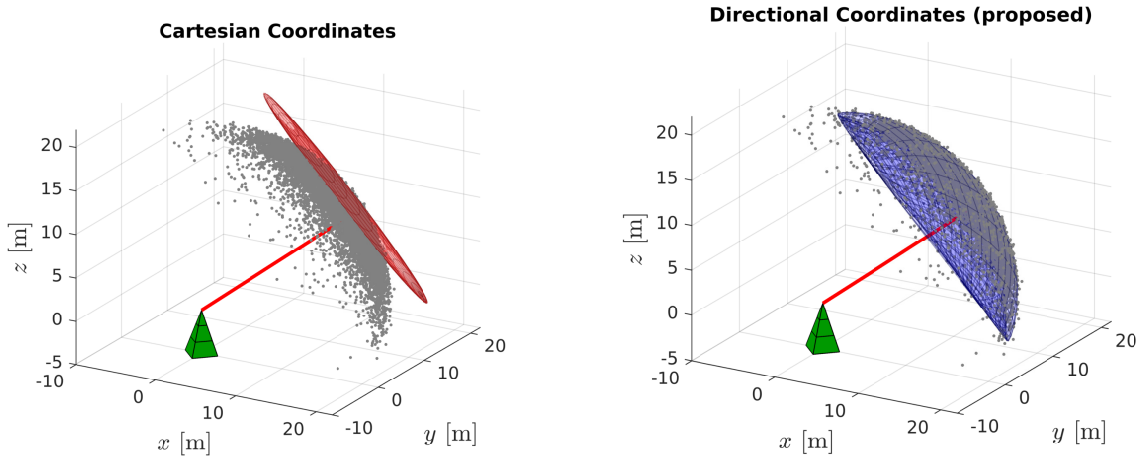

Figure 4.1: Distribution of positions of a robot given a single distance measurement (red line) to a landmark (green pyramid) and a Gaussian prior distribution. The point cloud represents the "true" distribution, as determined by a particle filter, while the red and blue volumes are 3σ equal probability contours in Cartesian and directional coordinates, respectively.

relative to some landmark or reference point, using range, azimuth, and elevation (RAE) measurements. RAE measurements can be generated from a variety of sensors, such as radar used to track aircraft and spacecraft, or point cloud measurements obtained from LIDAR [43, Ch. 6.4.3]. Recently, ultra-wideband (UWB) radio has also been used to provide angle of arrival measurements [19], which is essentially an azimuth measurement, in addition to the usual distance measurements obtained between two UWB transceivers. However, single-antenna UWB angle of arrival measurements can have high variance [21].

A typical probabilistic estimation algorithm will use the familiar Cartesian x, y, z coordinates, with Gaussian estimators reporting the mean and covariance of the estimated distribution. However, as will be shown in this chapter, Cartesian coordinates are generally a poor way of parameterizing the position of a robot when relying heavily on RAE measurements, especially when these measurements are very noisy, or if the robot position is highly uncertain. For example, the distribution of positions given a range measurement generally appears spherical, as shown in Figure 4.1, which greatly differs from a standard Gaussian ellipsoid. For this reason, this chapter proposes an alternate coordinate system, referred to herein as *directional coordinates*. These coordinates are shown to provide a much more accurate representation of the position distribution, ultimately leading to improved state estimate consistency.

The idea of changing the coordinate system to better represent a distribution is not new. The "banana distribution" associated with robot position uncertainty, primarily resulting from attitude uncertainty and sensor noise, has motivated the use of *exponential coordinates* [11]. A Gaussian distribution in exponential coordinates transforms to a "banana" distribution when mapped back to Cartesian coordinates. These exponential coordinates are naturally exploited

when using matrix Lie groups in robotic estimation problems [13, 43]. Spherical coordinates use range, azimuth, and elevation as the coordinates themselves, and have been used in state estimation [55, 56]. Spherical coordinates are often modified to avoid singularities when the range is zero, such as using the logarithm of the range in [56], or its reciprocal [55]. However, spherical coordinates possess an additional kinematic singularity at an elevation of $\varepsilon = \pm\pi/2$ rad, and also suffer from angle wrap-around, which must be overcome with explicit checks in the estimator implementation. Another strategy specific to RAE measurements is simply to directly convert them to a position measurement [57, 58], thus creating a linear measurement model. However, this requires careful conversion of the original noise statistics to equivalent noise statistics on the new measurement, which can introduce inaccuracies.

The new *directional coordinate* system proposed in this chapter uses a range and direction cosine matrix (DCM) to parameterize position. The result is a smooth parameterization with 1) no angle wrap-around issues, 2) no kinematic singularities associated with the elevation, 3) RAE measurement models that are linear, or have constant, state-estimate-independent Jacobians, and 4) a filter that is significantly more consistent than a standard Cartesian coordinate filter, when noise levels are high.

One drawback of the proposed use of directional coordinates is that a previously linear process model in Cartesian coordinates becomes nonlinear, an example being a simple "velocity input" process model. However, in applications with measurements arriving at a regular frequency, this added complexity is justified by the simplified measurement models, and enhanced ability to capture the true distributions. The proposed directional coordinates are evaluated in an extended Kalman filter (EKF) framework, and compared to an EKF that uses the usual Cartesian coordinates.

4.3 Directional Coordinates

In this chapter, *directional coordinates* are defined as the pair $(\rho, \mathbf{C})$, where $\rho \in \mathbb{R}_{\geq 0}$ and $\mathbf{C} \in SO(3)$ is a direction cosine matrix (rotation matrix). The idea is to use these coordinates as an alternate representation of the components of a Cartesian position vector, denoted $\mathbf{r} \in \mathbb{R}^3$. This is done with the defining relation

$$\mathbf{r} = \rho \mathbf{C} \mathbf{e}_1, \tag{4.1}$$

where $\mathbf{e}_1 = [1\ 0\ 0]^\mathsf{T}$. The *range* $\rho = \|\mathbf{r}\|$ is the length of the position vector, whereas $\mathbf{C}$ can be interpreted as a direction cosine matrix (DCM) that "rotates" the vector $\mathbf{e}_1$ to produce a *unit direction vector* $\boldsymbol{\varrho} = \mathbf{C}\mathbf{e}_1 = \mathbf{r}/\|\mathbf{r}\|$ collinear to $\mathbf{r}$.

Directional coordinates are similar to spherical coordinates, but the azimuth and elevation

angles are instead represented with a DCM $\mathbf{C}$, but with one of its three degrees of freedom fixed to zero to avoid an over-parameterization. This avoids the kinematic singularities associated with spherical coordinates, as well as any angle wrap-around issues.

4.3.1 Cartesian to Directional Coordinates

Given a position vector $\mathbf{r} = [r_x \; r_y \; r_z]^\mathsf{T}$ expressed in Cartesian coordinates, equivalent directional coordinates $(\rho, \mathbf{C})$ satisfying (4.1) can be obtained from the well-known axis-angle parameterization of a DCM. The range is straightforwardly obtained with $\rho = \|\mathbf{r}\|_2$, while the direction cosine matrix $\mathbf{C}$ can be obtained by first defining $\mathbf{a}$ and ψ as

$$\mathbf{a} = \frac{\rho \mathbf{e}_1^\times \mathbf{r}}{\left\|\rho \mathbf{e}_1^\times \mathbf{r}\right\|} = \begin{bmatrix} 0 \\ -r_z \\ r_y \end{bmatrix} \Bigg/ \left\| \begin{bmatrix} 0 \\ -r_z \\ r_y \end{bmatrix} \right\|,$$

$$\psi = \arccos \frac{\rho \mathbf{e}_1^\mathsf{T} \mathbf{r}}{\|\rho \mathbf{e}_1\| \, \|\mathbf{r}\|} = \arccos \frac{\rho \mathbf{e}_1^\mathsf{T} \mathbf{r}}{\rho^2} = \arccos \frac{r_x}{\rho},$$

from which $\mathbf{C}$ is then given by $\mathbf{C} = \exp(\psi \mathbf{a}^\times)$, which is well defined except when $r_z = r_y = 0$. In this case, setting $\rho = \|\mathbf{r}\|$ and $\mathbf{C} = \mathbf{1}$ will satisfy the definition given by (4.1). As such, a continuous function $\mathbf{h} : \mathbb{R}^3 \to \mathbb{R} \times SO(3)$ is defined that performs the above operations to go from $\mathbf{r}$ to $(\rho, \mathbf{C})$. That is,

$$(\rho, \mathbf{C}) = \mathbf{h}(\mathbf{r}). \tag{4.2}$$

4.3.2 Local Parameterization

Estimation tools that operate on $SO(3)$ directly are very well established [11, 13, 43], most of which require the use of a *local parameterization* of $SO(3)$. In this chapter, this is given by $\boldsymbol{\phi} = [\phi_1 \; \phi_2]^\mathsf{T} \in \mathbb{R}^2$, defined such that

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge),$$

where the *wedge* operator $(\cdot)^\wedge : \mathbb{R}^2 \to \mathfrak{so}(3)$ is given by

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_2 & \phi_1 \\ \phi_2 & 0 & 0 \\ -\phi_1 & 0 & 0 \end{bmatrix},$$

and otherwise expressed as $\boldsymbol{\phi}^\wedge = ([0 \; \boldsymbol{\phi}^\mathsf{T}]^\mathsf{T})^\times$. An inverse mapping, named the *vee* operator, $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^2$ can be defined such that $((\boldsymbol{\phi})^\wedge)^\vee = \boldsymbol{\phi}$. It is also useful to define an operator $(\cdot)^\odot : \mathbb{R}^3 \to \mathbb{R}^{3\times 2}$ such that the identity $\boldsymbol{\phi}^\wedge \mathbf{a} = \mathbf{a}^\odot \boldsymbol{\phi}$ holds. With the above definition of the

wedge operator, it can be shown that the $(\cdot)^{\odot}$ operator must be

$$\mathbf{a}^{\odot} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}^{\odot} = \begin{bmatrix} a_3 & -a_2 \\ 0 & a_1 \\ -a_1 & 0 \end{bmatrix}. \tag{4.3}$$

A few useful identities can be derived, for $\mathbf{b}, \mathbf{c} \in \mathbb{R}^2$, by simple expansion into components,

$$\mathbf{b}^{\wedge^{\mathsf{T}}} = -\mathbf{b}^{\wedge}, \quad (\mathbf{b}+\mathbf{c})^{\wedge} = \mathbf{b}^{\wedge} + \mathbf{c}^{\wedge}, \quad (\mathbf{b}^{\wedge}\mathbf{c}^{\wedge})^{\vee} = \mathbf{0}. \tag{4.4}$$

4.4 Distributions in Directional Coordinates

Consider the "mean" or "nominal" directional coordinates $(\check{\rho}, \check{\mathbf{C}})$ and the perturbations $\delta\rho$ and $\delta\boldsymbol{\phi}$ such that

$$\rho = \check{\rho} + \delta\rho, \tag{4.5}$$

$$\mathbf{C} = \check{\mathbf{C}} \exp(\delta\boldsymbol{\phi}^{\wedge}). \tag{4.6}$$

Define

$$\delta\mathbf{x} = \begin{bmatrix} \delta\rho \\ \delta\boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^3. \tag{4.7}$$

If $\delta\mathbf{x}$ is a random variable drawn from $\delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \check{\mathbf{P}})$, this will induce a distribution over the values of $(\rho, \mathbf{C})$.

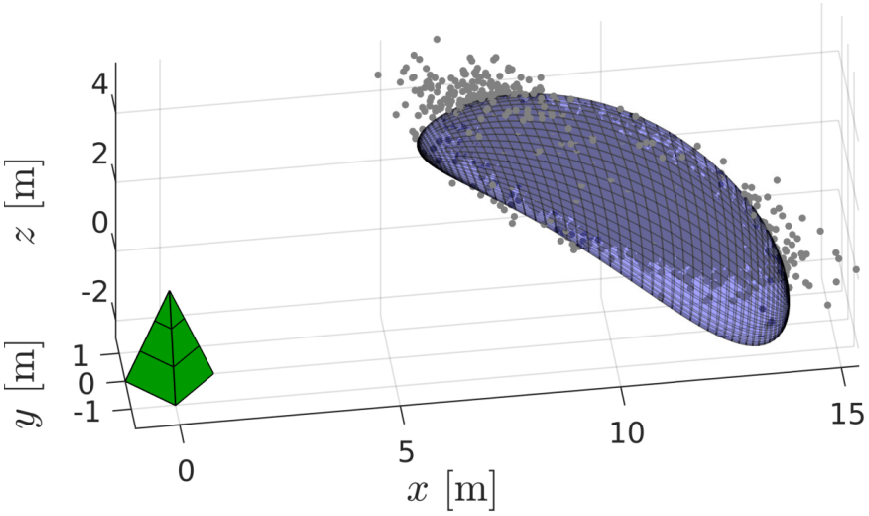

Figure 4.2: Samples from a Cartesian Gaussian distribution, shown as the point cloud. The bean-shaped volume shows a 3 standard deviation equal probability contour of a Gaussian distribution in directional coordinates, but mapped back to Cartesian coordinates for visualization.

4.4.1 Converting a Cartesian Gaussian Distribution to Directional Coordinates

A common requirement will be to convert a Gaussian distribution over Cartesian coordinates $\mathbf{r} \sim \mathcal{N}(\check{\mathbf{r}}, \check{\mathbf{P}}_r)$, into an equivalent, or approximate, distribution in directional coordinates, represented with nominal point $(\check{\rho}, \check{\mathbf{C}})$ and corresponding distribution $\delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \check{\mathbf{P}})$. Fortunately, a mapping $\mathbf{h}(\cdot)$ has been identified in Section 4.3.1, which allows for many well-known methods of passing a Gaussian through a nonlinearity [43, Ch. 2.2.8]. This chapter uses sigma points for this step exclusively.

A variety of sigma point methods, such as the *unscented transform*, *spherical cubature*, or *Gauss-Hermite cubature* [37, Ch. 6], can be used to generate a set of sigma points and corresponding weights $(\mathbf{s}_i, w_i),\ i = 1, \ldots, N$ drawn from the prior distribution $\mathcal{N}(\check{\mathbf{r}}, \check{\mathbf{P}}_r)$. The Cartesian sigma points are transformed to directional coordinates through $\mathbf{h}(\cdot)$,

$$(\rho_i, \mathbf{C}_i) = \mathbf{h}(\mathbf{s}_i), \qquad i = 1, \ldots, N.$$

The covariance associated with the directional coordinates can then be approximated as

$$\check{\mathbf{P}} \approx \sum_{i=1}^{N} w_i \delta\boldsymbol{\xi}_i \delta\boldsymbol{\xi}_i^{\mathsf{T}}, \qquad \delta\boldsymbol{\xi}_i = \begin{bmatrix} \rho_i - \check{\rho} \\ \ln(\check{\mathbf{C}}^{\mathsf{T}} \mathbf{C}_i)^{\vee} \end{bmatrix}.$$

In standard sigma point methods, $(\check{\rho}, \check{\mathbf{C}})$ would be obtained from the weighted mean of the transformed sigma point values $(\rho_i, \mathbf{C}_i)$ [37, Ch. 6], which would require an optimization procedure to find the mean $\check{\mathbf{C}}$ on $SO(3)$ [59]. However, as suggested by [60], and verified extensively through simulation in this chapter, it is sufficient to simply obtain the nominal points $(\check{\rho}, \check{\mathbf{C}})$ by passing $\check{\mathbf{r}}$ through the nonlinear model (4.2), $(\check{\rho}, \check{\mathbf{C}}) = \mathbf{h}(\check{\mathbf{r}})$, without an apparent loss in accuracy. An example of the results of this sigma point procedure can be visualized in Figure 4.2.

4.4.2 Posterior Distribution given a Range Measurement

Consider now the task of estimating the distribution of $(\rho, \mathbf{C})$ given a prior distribution and a single range measurement y from the origin. The prior distribution's mean is denoted $(\check{\rho}, \check{\mathbf{C}})$, and is distributed as per (4.5)-(4.6) with $\delta\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \check{\mathbf{P}})$. In directional coordinates, the range measurement model is linear,

$$y = \rho + \nu, \qquad \nu \sim \mathcal{N}(0, R).$$

This leads to a trivial linearization procedure, where small changes in δy are related to small $\delta\mathbf{x}$ through $\delta y = \mathbf{H}\delta\mathbf{x} + \mathbf{M}\nu$, with $\mathbf{H} = [1\ 0\ 0]$ and $\mathbf{M} = 1$. This provides a setup to use a *multiplicative extended Kalman filter* [43] correction step to estimate the posterior mean

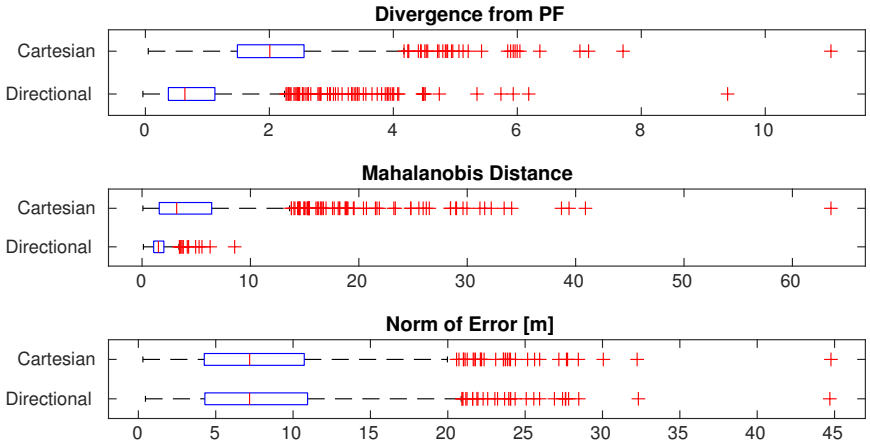

Figure 4.3: 1000 Monte Carlo trials of a single Kalman correction given a distance measurement. Although the estimation accuracy using directional coordinates is not necessarily improved, directional coordinates offer a more consistent filter, as well as one that better resembles the particle filter.

$(\hat{\rho}, \hat{\mathbf{C}})$ and covariance $\hat{\mathbf{P}}$ with

$$\mathbf{K} = \check{\mathbf{P}}\mathbf{H}^{\mathsf{T}}(\mathbf{H}\check{\mathbf{P}}\mathbf{H}^{\mathsf{T}} + \mathbf{M}R\mathbf{M}^{\mathsf{T}})^{-1}, \tag{4.8}$$

$$\delta\check{\mathbf{x}} = \begin{bmatrix} \delta\check{\rho}, \\ \delta\check{\boldsymbol{\phi}} \end{bmatrix} = \mathbf{K}\mathbf{z}, \qquad \mathbf{z} = y - \check{\rho}, \tag{4.9}$$

$$\hat{\rho} = \check{\rho} + \delta\check{\rho}, \tag{4.10}$$

$$\hat{\mathbf{C}} = \check{\mathbf{C}}\exp(\delta\check{\boldsymbol{\phi}}^{\wedge}), \tag{4.11}$$

$$\hat{\mathbf{P}} = (\mathbf{1} - \mathbf{K}\mathbf{H})\check{\mathbf{P}}(\mathbf{1} - \mathbf{K}\mathbf{H})^{\mathsf{T}} + \mathbf{K}\mathbf{M}R\mathbf{M}^{\mathsf{T}}\mathbf{K}^{\mathsf{T}}. \tag{4.12}$$

Figure 4.1 visualizes the results of this single EKF correction step when it is performed in Cartesian coordinates and directional coordinates, for the same Cartesian Gaussian prior. The point cloud is the distribution determined by a standard bootstrap particle filter [37, Ch. 7], and is considered the closest approximation to the true distribution. In Cartesian coordinates, the measurement model is given by $y = \|\mathbf{r}\|_2 + \nu$, which is nonlinear, and requires linearization to produce a state-dependent measurement Jacobian. Linearization errors aside, Figure 4.1 clearly shows the degree to which the distribution is non-Gaussian, when expressed in Cartesian coordinates. As such, under high uncertainty, there is no Cartesian Gaussian estimator that could ever accurately represent this distribution.

This simple single-step correction experiment is repeated for 1000 Monte Carlo trials, where the prior mean $\check{\mathbf{r}}$ and covariance $\check{\mathbf{P}}_r$ are randomized, and the true position $\mathbf{r}^{\text{true}}$ is randomly sampled from this prior distribution. The prior distribution is converted to directional coordinates using Section 4.4.1, and the posterior is calculated using Section 4.4.2. Figure 4.3 displays various metrics of the 1000 Monte Carlo trials. The "dissimilarity", or

divergence, from the particle filter is calculated using the method in [61], which calculates an approximate Kullback-Leibler (KL) divergence between two point clouds, created by sampling their respective distributions. The Mahalanobis distance for the directional coordinates is calculated with

$$D = \sqrt{\delta\boldsymbol{\xi}^{\mathsf{T}}\hat{\mathbf{P}}^{-1}\delta\boldsymbol{\xi}}, \qquad \delta\boldsymbol{\xi} = \begin{bmatrix} \rho^{\text{true}} - \hat{\rho} \\ \ln(\hat{\mathbf{C}}^{\mathsf{T}}\mathbf{C}^{\text{true}})^{\vee} \end{bmatrix}.$$

Although, on a single step, there is no accuracy improvement when using directional coordinates, there are substantially enhanced consistency properties, as reflected by the significantly lower Mahalanobis distances and divergence from the particle filter.

4.4.3 Posterior Distribution given Azimuth and Elevation Measurements

Another set of measurements that are naturally well-suited to directional coordinates are azimuth and elevation (AE) measurements, denoted α and ε respectively. Azimuth and elevation measurements are defined to produce a direction vector $\boldsymbol{\varrho}$ where

$$\boldsymbol{\varrho} = \mathbf{C}_z(\alpha)^{\mathsf{T}}\mathbf{C}_y(-\varepsilon)^{\mathsf{T}}\mathbf{e}_1, \tag{4.13}$$

and $\mathbf{C}_z(\cdot)$, $\mathbf{C}_y(\cdot)$ are principle rotation DCMs about the z and y axes, respectively. Hence, if the direction vector resulting from (4.13) is considered to be the measurement itself, the measurement model

$$\mathbf{y} = \mathbf{C}\mathbf{e}_1 + \boldsymbol{\nu}, \qquad \boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

is obtained. The covariance $\mathbf{R}$ can be obtained from covariances associated with α and ε using a standard linearization procedure [43, Ch. 2.2.8]. Next, a measurement innovation $\mathbf{z}$ is defined following inspiration from the *invariant extended Kalman filter* (IEKF) literature [13]. Moreover, only two components of $\mathbf{z}$ are used to avoid creating a fictitious third measurement, when only two are actually measured, that being α and ε. This is done through use of the projection matrix $\mathbf{E}$ in (4.14), where the innovation is defined as

$$\begin{aligned} \mathbf{z} &\triangleq \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{E}} \check{\mathbf{C}}^{\mathsf{T}}(\mathbf{y} - \check{\mathbf{y}}) \qquad (4.14) \\ &= \mathbf{E}\check{\mathbf{C}}^{\mathsf{T}}\mathbf{C}\mathbf{e}_1 + \mathbf{E}\check{\mathbf{C}}^{\mathsf{T}}\boldsymbol{\nu} - \mathbf{E}\check{\mathbf{C}}^{\mathsf{T}}\check{\mathbf{C}}\mathbf{e}_1 \\ &\approx \mathbf{E}(\mathbf{1} + \delta\boldsymbol{\phi}^{\wedge})\mathbf{e}_1 + \mathbf{E}\check{\mathbf{C}}^{\mathsf{T}}\boldsymbol{\nu} - \mathbf{E}\mathbf{e}_1 \\ &= \mathbf{E}\mathbf{e}_1^{\odot}\delta\boldsymbol{\phi} + \mathbf{E}\check{\mathbf{C}}^{\mathsf{T}}\boldsymbol{\nu} \triangleq \mathbf{H}_{\phi}\delta\boldsymbol{\phi} + \mathbf{M}\boldsymbol{\nu}, \qquad (4.15) \end{aligned}$$

where $\mathbf{C} = \check{\mathbf{C}}\exp(\delta\boldsymbol{\phi}^{\wedge})$, and the first-order approximation $\exp(\delta\boldsymbol{\phi}^{\wedge}) \approx (\mathbf{1} + \delta\boldsymbol{\phi}^{\wedge})$ has been made. As with the range measurements, the measurement Jacobian $\mathbf{H} = [\mathbf{0}\ \mathbf{H}_{\phi}]$ is constant and state-estimate independent, and would otherwise be highly nonlinear in Cartesian

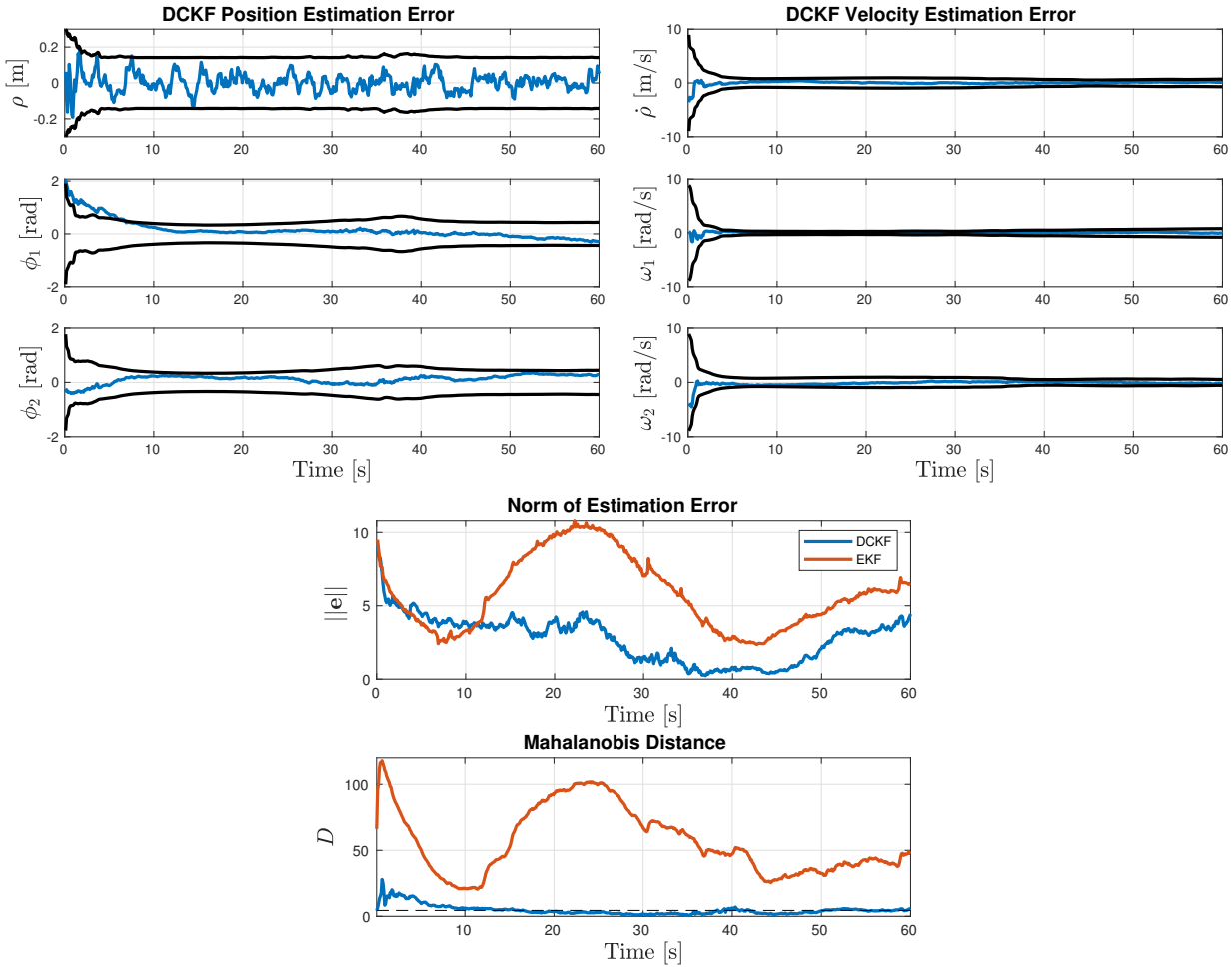

Figure 4.4: Simulation results from a single trial. Black lines represent the $\pm 3\sigma$ confidence bounds. The total estimation error on the bottom figure is calculated for both filters with $\mathbf{e} = [\mathbf{r}^{\text{true}^\mathsf{T}}\ \mathbf{v}^{\text{true}^\mathsf{T}}]^\mathsf{T} - [\hat{\mathbf{r}}^\mathsf{T}\ \hat{\mathbf{v}}^\mathsf{T}]^\mathsf{T}$.

coordinates. One may return to (4.8)-(4.12) to compute a Kalman filter correction step, with new definitions for $\mathbf{y}$, $\mathbf{z}$, $\mathbf{H}$, $\mathbf{M}$, and $\mathbf{R}$.

4.5 Directional Coordinate Kinematics

To execute dynamic filtering tasks, a process model is required, and it will usually be necessary to relate the rate of change of the directional coordinates $(\dot{\rho}, \dot{\mathbf{C}})$ to a Cartesian velocity input $\mathbf{v} = \dot{\mathbf{r}}$. Differentiating (4.1) with respect to time,

$$\dot{\mathbf{r}} = \dot{\rho}\mathbf{C}\mathbf{e}_1 + \rho\dot{\mathbf{C}}\mathbf{e}_1. \tag{4.16}$$

The time rate of change of $\mathbf{C}$ is $\dot{\mathbf{C}} = \mathbf{C}\boldsymbol{\omega}^\wedge$, where $\boldsymbol{\omega} \in \mathbb{R}^2$. Substituting $\dot{\mathbf{C}} = \mathbf{C}\boldsymbol{\omega}^\wedge$ into (4.16), and using the $(\cdot)^\odot$ operator defined in Section 4.3.2 results in

$$\begin{aligned}
\dot{\mathbf{r}} &= \dot{\rho}\mathbf{C}\mathbf{e}_1 + \rho\mathbf{C}\boldsymbol{\omega}^\wedge\mathbf{e}_1 \\
&= \dot{\rho}\mathbf{C}\mathbf{e}_1 + \rho\mathbf{C}\mathbf{e}_1^\odot\boldsymbol{\omega} \\
&= \begin{bmatrix} \mathbf{C}\mathbf{e}_1 & \rho\mathbf{C}\mathbf{e}_1^\odot \end{bmatrix} \begin{bmatrix} \dot{\rho} \\ \boldsymbol{\omega} \end{bmatrix} \\
&= \underbrace{\mathbf{C}\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_1^\odot \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \rho\mathbf{1} \end{bmatrix}}_{\mathbf{S}(\rho,\mathbf{C})} \begin{bmatrix} \dot{\rho} \\ \boldsymbol{\omega} \end{bmatrix}.
\end{aligned} \tag{4.17}$$

To obtain the time rate of change of the directional coordinates, $\mathbf{S}$ must be invertible. Fortunately, assuming $\rho > 0$, $\mathbf{S}$ has an analytical inverse given by

$$\mathbf{S}(\rho, \mathbf{C})^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & (1/\rho)\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{e}_1^\mathsf{T} \\ \mathbf{e}_1^{\odot\mathsf{T}} \end{bmatrix} \mathbf{C}^\mathsf{T}.$$

It follows that

$$\begin{bmatrix} \dot{\rho} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{S}(\rho, \mathbf{C})^{-1}\mathbf{v}, \tag{4.18}$$

which is well defined provided $\rho > 0$. Equation (4.18) can be easily split into its two constituent equations, leading to the time rate of change of the directional coordinates, that being

$$\dot{\rho} = \mathbf{e}_1^\mathsf{T}\mathbf{C}^\mathsf{T}\mathbf{v}, \qquad \dot{\mathbf{C}} = \frac{1}{\rho}\mathbf{C}(\mathbf{e}_1^{\odot\mathsf{T}}\mathbf{C}^\mathsf{T}\mathbf{v})^\wedge. \tag{4.19}$$

The only kinematic singularity is located at $\rho = 0$, as opposed to spherical coordinates, which have an additional singularity at an elevation of $\varepsilon = \pm\pi/2$ rad. This singularity is not of concern provided the robot device does not move near the origin. This is often the case in practice, as if another robot or measurement device is located at the origin, the hardware itself forces a minimum separating distance.

4.5.1 Linearization

It is necessary to linearize the directional coordinate kinematics in order to execute an EKF prediction step. This is done by applying small perturbations to $\rho, \mathbf{C}, \mathbf{v}$ with $\rho = \hat{\rho} + \delta\rho$, $\mathbf{C} \approx \hat{\mathbf{C}}(\mathbf{1} + \delta\boldsymbol{\phi}^\wedge)$, and $\mathbf{v} = \hat{\mathbf{v}} + \delta\mathbf{v}$. Equation (4.19) becomes

$$\dot{\hat{\rho}} + \delta\dot{\rho} \approx \mathbf{e}_1^\mathsf{T}\left(\hat{\mathbf{C}}(\mathbf{1} + \delta\boldsymbol{\phi}^\wedge)\right)^\mathsf{T}(\hat{\mathbf{v}} + \delta\mathbf{v}). \tag{4.20}$$

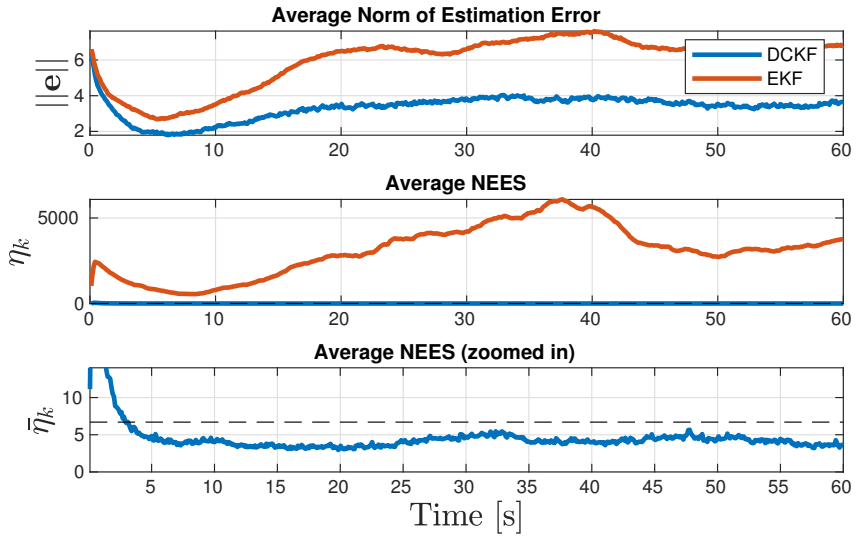

Figure 4.5: Average estimation error and NEES, throughout time, for 100 Monte Carlo trials. The dashed line represents the one-sided 99.7% probability boundary.

Expanding (4.20), neglecting higher-order terms, and applying identities (4.4), eventually gives

$$\delta\dot{\rho} \approx -\mathbf{e}_1^\mathsf{T}(\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}})^\odot\delta\boldsymbol{\phi} + \mathbf{e}_1^\mathsf{T}\hat{\mathbf{C}}^\mathsf{T}\delta\mathbf{v}.$$

Linearizing the DCM kinematics gives

$$\frac{\mathrm{d}(\mathbf{1}+\delta\boldsymbol{\phi}^\wedge)}{\mathrm{d}t} \approx \frac{\mathrm{d}(\hat{\mathbf{C}}^\mathsf{T}\mathbf{C})}{\mathrm{d}t} = \dot{\hat{\mathbf{C}}}^\mathsf{T}\mathbf{C} + \hat{\mathbf{C}}^\mathsf{T}\dot{\mathbf{C}}, \tag{4.21}$$

$$\delta\dot{\boldsymbol{\phi}}^\wedge \approx \frac{-1}{\hat{\rho}}(\mathbf{e}_1^{\odot\mathsf{T}}\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}})^\wedge\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{C}}(\mathbf{1}+\delta\boldsymbol{\phi}^\wedge) + \frac{1}{\rho}\hat{\mathbf{C}}^\mathsf{T}\mathbf{C}(\mathbf{e}_1^{\odot\mathsf{T}}\mathbf{C}^\mathsf{T}\mathbf{v})^\wedge. \tag{4.22}$$

Again, after expansion, neglecting higher-order terms, applying identities (4.4), and application of the $(\cdot)^\vee$ to both sides results in

$$\delta\dot{\boldsymbol{\phi}} \approx \frac{1}{\hat{\rho}}\mathbf{e}_1^{\odot\mathsf{T}}\hat{\mathbf{C}}^\mathsf{T}\delta\mathbf{v} - \frac{1}{\hat{\rho}}\mathbf{e}_1^{\odot\mathsf{T}}(\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}})^\odot\delta\boldsymbol{\phi} - \frac{1}{\hat{\rho}^2}\mathbf{e}_1^{\odot\mathsf{T}}\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}}\delta\rho. \tag{4.23}$$

4.6 Simulation Results

Two different EKFs are tested in simulation. The only difference between the two EKFs is the coordinate system used for the position parameterization. The standard EKF uses Cartesian coordinates, whereas the *directional coordinate Kalman filter* (DCKF) uses directional coordinates. In these simulations, it is assumed that the acceleration $\mathbf{a} = \dot{\mathbf{v}} = \ddot{\mathbf{r}}$ of some point is measured with noise, along with noisy RAE measurements. The EKF state consists of $[\mathbf{r}^\mathsf{T}\ \mathbf{v}^\mathsf{T}]^\mathsf{T}$, while the DCKF state is $(\rho, \mathbf{C}, \mathbf{v})$, where $\mathbf{v} \in \mathbb{R}^3$ is a Cartesian velocity.

For the DCKF, the continuous-time linearized process model can be written as $\delta\dot{\mathbf{x}} =$

Table 4.1: Simulation Noise Properties

Specification	Value	Units
Range std. dev.	0.1	m
AE std. dev.	0.8	rad
Accel. std. dev	0.1	$\mathrm{m/s}^2$
Init. pos. std. dev.	5	m
Init. vel. std. dev.	3	m/s
Meas. frequency	10	Hz

$\mathbf{A}\delta\mathbf{x} + \mathbf{L}\delta\mathbf{a}$ where $\delta\mathbf{x} = [\delta\rho\ \delta\boldsymbol{\phi}^\mathsf{T}\ \delta\mathbf{v}^\mathsf{T}]^\mathsf{T}$,

$$\mathbf{A} = \begin{bmatrix} 0 & -\mathbf{e}_1^\mathsf{T}(\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}})^\odot & \mathbf{e}_1^\mathsf{T}\hat{\mathbf{C}}^\mathsf{T} \\ -\frac{1}{\bar{\rho}^2}\mathbf{e}_1^{\odot\mathsf{T}}\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}} & -\frac{1}{\bar{\rho}}\mathbf{e}_1^{\odot\mathsf{T}}(\hat{\mathbf{C}}^\mathsf{T}\hat{\mathbf{v}})^\odot & \frac{1}{\bar{\rho}}\mathbf{e}_1^{\odot\mathsf{T}}\hat{\mathbf{C}}^\mathsf{T} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{4.24}$$

and $\mathbf{L} = [0\ \mathbf{0}\ \mathbf{1}]^\mathsf{T}$. The prediction step of the DCKF integrates (4.19) and $\dot{\mathbf{v}} = \mathbf{a}$ forward in time with simple Euler integration, but any method can be used. An equivalent discrete-time linearized model $\delta\mathbf{x}_k = \mathbf{A}_{k-1}\delta\mathbf{x}_{k-1} + \mathbf{w}_{k-1},\ \mathbf{w}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ can be obtained for the DCKF using a standard discretization technique such as a zero-order-hold. The matrices $\mathbf{A}_{k-1}$ and $\mathbf{Q}_{k-1}$ are used to propagate the filter covariance forwards with the standard expression, $\check{\mathbf{P}}_k = \mathbf{A}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{A}_{k-1}^\mathsf{T} + \mathbf{Q}_{k-1}$. The correction step for the DCKF is performed using equations (4.8)-(4.12), with $\mathbf{y}$, $\mathbf{z}$, $\mathbf{H}$, $\mathbf{M}$, and $\mathbf{R}$ matrices defined in Sections 4.4.2 for a range measurement, or 4.4.3 for azimuth-elevation measurements. The $\mathbf{H}$ matrices are augmented with zeros due to the additional state $\mathbf{v}$.

Figure 4.4 shows an example run with no covariance tuning, displaying the error behavior and consistency of the proposed DCKF. Figure 4.5 contains results for 100 Monte Carlo trials with mild covariance tuning. The *normalized estimation error squared* (NEES) [44, Ch. 5.4] η_k^i is a consistency metric, calculated for the directional coordinates at time step k, on trial i, with

$$\eta_k^i = \delta\boldsymbol{\xi}_k^\mathsf{T}\hat{\mathbf{P}}_k^{-1}\delta\boldsymbol{\xi}_k, \qquad \delta\boldsymbol{\xi}_k = \begin{bmatrix} \rho_k^{\text{true}} - \hat{\rho}_k \\ \ln(\hat{\mathbf{C}}_k^\mathsf{T}\mathbf{C}_k^{\text{true}})^\vee \\ \mathbf{v}_k^{\text{true}} - \hat{\mathbf{v}}_k \end{bmatrix}. \tag{4.25}$$

Under large noise and initial uncertainty, the DCKF boasts an average 44% reduction in estimation error compared to the EKF and remains consistent, whereas the EKF is completely inconsistent, even with dedicated covariance tuning.

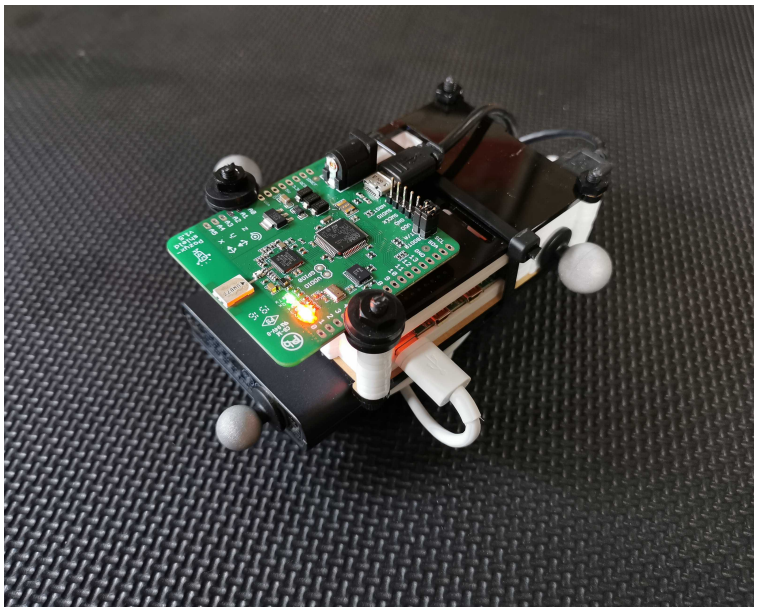

Figure 4.6: Pozyx UWB Developer Tag mounted to a Raspberry Pi 4B.

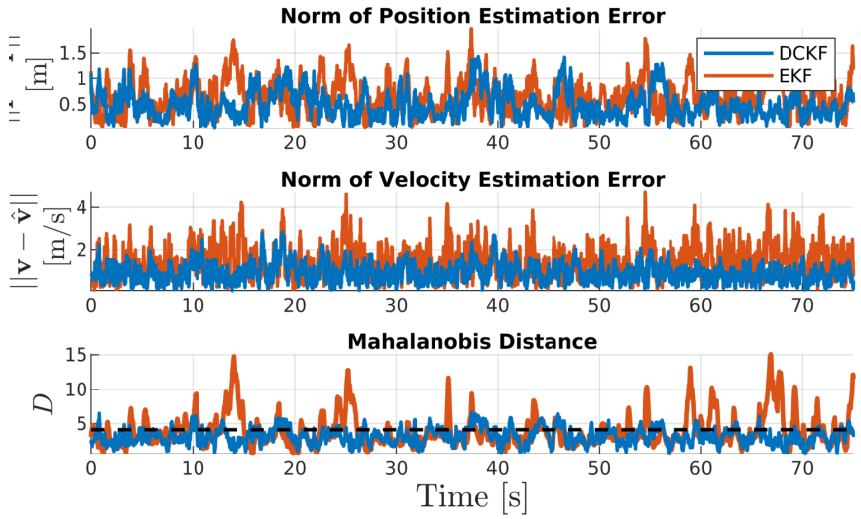

Figure 4.7: Experimental results from a single trial, with the dashed line representing the one-sided 99.7% probability boundary.

4.7 Experimental Results

The EKF and DCKF are also compared in a real experiment. Figure 4.6 shows the Raspberry Pi 4B that was used with an LSM9DS1 IMU, providing accelerometer and gyroscope measurements, and a Pozyx UWB Developer Tag, providing distance measurements to a Pozyx UWB Creator Anchor on the floor. Because the particular UWB modules used do not provide azimuth and elevation measurements, azimuth and elevation measurements were

simulated using ground-truth data, with artificially added zero-mean Gaussian noise with a standard deviation of 0.8 rad. This noise level is intentionally high, to induce a performance difference between the DCKF and EKF, and highlight the DCKF's improvement under large noise. The device moved in an overall volume of roughly 5 m $\times$ 4 m $\times$ 2 m. Ground truth position and attitude measurements are collected using an OptiTrack optical motion capture system.

Figure 4.7 shows various performance and consistency metrics associated with the experimental trial. The position and velocity root-mean-squared error (RMSE) was 0.55 m, 0.87 m/s for the DCKF, and 0.97 m, 1.56 m/s for the EKF, respectively. This corresponds to a 43% reduction in position RMSE, and a 44% reduction in velocity RMSE. Once again, the DCKF is both more accurate and more consistent, when compared to the EKF.

4.8 Conclusion and Future Work

This chapter introduces a new coordinate system for parameterizing positions, which consists of using a range ρ and a direction, represented by a direction cosine matrix $\mathbf{C} \in SO(3)$. This chapter shows how this is particularly well-suited for RAE measurements, as the directional coordinate system captures the posterior distribution much more effectively than Cartesian coordinates, which culminates in a more accurate and significantly more consistent estimator.

One of the main limitations of this idea is that it is quite specific to the described class of measurements. For example, if the ranging device is not collocated with the IMU, and there is a significant moment arm, then the measurement is no longer linear and this important desired property may be lost. There are, however, interesting future directions that have emerged as a result of this investigation.

- **Representing the range coordinate as an element of the multiplicative group.** In this chapter, the range coordinate has been treated as a simple element of $\mathbb{R}_{>0}$, but it is never explicitly enforced that the number be strictly positive. Since range errors are computed by simple subtraction, and the Kalman correction is additive, the range could theoretically become negative. However, if the range is declared to be an element of the multiplicative group $(\mathbb{R}, \times)$ where $\times$ is regular scalar multiplication, then the range will naturally be strictly positive. The exponential map for this group is simply the regular exponential function

  $$\rho = \exp(\xi)$$

  with the logarithmic map as the natural logarithm. The time rate of change is simply

of the form

$$\dot{\rho} = \rho\xi$$

where $\xi \in \mathbb{R}$. Under assumption of constant ξ, this can be discretized exactly with $\rho_k = \rho_{k-1}\exp(\Delta t \xi_{k-1})$. The ranging model remains $y = \rho + v$, but the linearization would be done differently, in a multiplicative way

$$\begin{aligned} \bar{y} + \delta y &= \bar{\rho}\exp(\delta\xi) + v && (4.26)\\ &\approx \bar{\rho}(1 + \delta\xi) + v, && (4.27)\\ \delta y &\approx \bar{\rho}\delta\xi, && (4.28) \end{aligned}$$

implying that the Jacobian of the ranging model is now given by $\bar{\rho}$, which has undesired dependence on the state. However, if a invariant innovation $z = \bar{\rho}^{-1}(y - \bar{\rho})$ is used, the Jacobian of the innovation would simply be 1, and state-independence is recovered.

- **Generalizing coordinate definition for representation of arbitrary distributions.** This chapter is yet another demonstration that simply changing the coordinate system in which a Gaussian distribution is defined, but leaving the estimator unchanged, can give significantly different estimation results [11, 12]. Hence, perhaps, an optimal coordinate system could be found for an arbitrary problem, which yields the best results estimation results according to a designed metric. Concretely, suppose there exists a familiar coordinate system $\mathbf{x}$, such as Cartesian coordinates, with a process and measurement model given by the standard form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) && (4.29)\\ \mathbf{y}_k &= \mathbf{g}(\mathbf{x}_k, \mathbf{v}_k) && (4.30) \end{aligned}$$

with $\mathbf{w}_k, \mathbf{v}_k$ being Gaussian noise terms. The task would be to find a smooth, bijective mapping $\boldsymbol{\varphi}(\cdot)$ relating the states to a new coordinate system

$$\begin{aligned} \boldsymbol{\xi} &= \boldsymbol{\varphi}(\mathbf{x}), && (4.31)\\ \mathbf{x} &= \boldsymbol{\varphi}^{-1}(\boldsymbol{\xi}). && (4.32) \end{aligned}$$

It is straightforward to rewrite the process and measurement model as a function of the new coordinates $\boldsymbol{\xi}$ with

$$\begin{aligned} \boldsymbol{\xi}_k &= \boldsymbol{\varphi}(\mathbf{f}(\boldsymbol{\varphi}^{-1}(\boldsymbol{\xi}_{k-1}), \mathbf{u}_{k-1}, \mathbf{w}_{k-1})), && (4.33)\\ \mathbf{y}_k &= \mathbf{g}(\boldsymbol{\varphi}^{-1}(\boldsymbol{\xi}_k), \mathbf{v}_k), && (4.34) \end{aligned}$$

which can then be employed in a variety of standard Bayesian estimators. The mapping $\boldsymbol{\varphi}(\cdot)$ could be a parametric model, such as a neural network, that is optimized/trained using ground truth state data to minimize the mean squared estimation error. This

potential method would be appropriate for systems where $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are fundamentally good models, but the coordinate system is not amenable to representing the true posterior distribution with a Gaussian distribution.

As previously stated, calculating the azimuth or elevation from ultra-wideband radio is an emergent technology that is rare to find in a commercial UWB product, and was not available for the experiments in this thesis. These directional measurements, even if they had very high variance, were found to be critical for the DCKF estimator to be statistically consistent. Recall that the original reason directional coordinates were considered was to achieve "consistency despite unobservability" when given only single-range measurements. The range measurement model becomes linear with this new coordinate system, and independent of the unobservable directions, which are the directional degrees of freedom associated with $\mathbf{C}$. However, when there is non-negligible process noise, the covariance associated with the directional states has been observed to incorrectly decrease when the true state remains static. This eventually leads to filter inconsistency, unless azimuth-elevation measurements are also present.

In light of the results of the previous two chapters, which strongly suggest that robots will need to have persistent relative motion when single-range measurements are used, the next two chapters will feature more sensors. In particular, more ultra-wideband tags will be added to each robot.

Chapter 5

Optimal Multi-Robot Formations for Relative Pose Estimation using Range Measurements

5.1 Summary

In this chapter, the problem of estimating relative poses from a set of inter-robot range measurements is investigated. Specifically, it is shown that the estimation accuracy is highly dependent on the true relative poses themselves, which prompts the desire to find multi-robot formations that provide the best estimation performance. By direct maximization of Fisher information, it is shown in simulation and experiment that improvements in estimation accuracy can be obtained by optimizing the formation geometry of a team of robots. The approach parameterizes the problem using body-frame-resolved relative poses, and uses on-manifold gradient descent to optimize the formation geometry.

5.2 Introduction

To obtain statically-observable relative position or pose estimates between robots, more relative measurements are required than just a single range measurement. For example, cameras can be used with object detection [62], or possibly infrared emitters/receivers [63]. In [64], mobile ground robots equipped with LIDAR serve as mobile anchors for quadcopters equipped with UWB. However, vision-based systems are prone to poor visual conditions, and require substantial computational resources, while LIDAR is typically expensive, bulky, and power-hungry. Furthermore, vision-based systems require robots to remain within each other's field of view, whereas UWB is omnidirectional.

An alternative is to place more tags on each robotic agent [65–67], which can create statically-observable relative poses as shown by [28]. When combined with an inertial measurement unit (IMU) and a magnetometer, the agents' individual attitudes can be estimated relative to a world frame, allowing relative positions to be resolved in the world frame. However, magnetometer sensor measurements are substantially disturbed in the presence of metallic structures indoors [68, 69], which degrades estimation accuracy. Another challenge is that there are certain formation geometries that cause the relative positions to be unobservable, such as when all the UWB tags lie on the same line [28]. This is closely tied to the well-known general dependence of positioning accuracy on the geometry of the tags, and arises even with the presence of anchors [70].

To avoid divergence of the state estimator, multi-robot missions relying on inter-robot range measurements for relative position estimation must avoid these aforementioned unobservable formation geometries. This imposes a constraint on planning algorithms. A planning solution to avoid unobservable positions is proposed in [71], where a cost function based on the Cramér-Rao bound quantifies the estimation accuracy as a function of robot positions. A similar approach is presented in [72] for multi-tag robots. Limitations of these approaches include the requirement of the presence of anchors, as well as the lack of explicit consideration of agent attitudes.

This contribution of the chapter is a method for computing locally optimal formations for relative pose estimation, especially in the absence of anchors. Furthermore, it is shown that with two-tag agents, both the relative position and relative heading of the agents are locally observable from range measurements alone. The problem setup is deliberately formulated in the agents' body frames, thus being completely invariant to any arbitrary world frame, eliminating the need for a magnetometer. This chapter further differs from [72] by using $SE(n)$ pose transformation matrices to represent the relative poses, avoiding the complications associated with angle parameterizations of attitude. This leads to the use of an *on-manifold gradient descent* procedure to determine optimal formations. Simulations and experiments show that the variance of estimation error does indeed decrease as the agents approach their optimal formations.

The proposed cost function is general to 2D or 3D translations, arbitrary measurement graphs, and any number of arbitrarily-located tags. Moreover, the proposed cost function goes to infinity when the agents approach unobservable configurations, meaning that its use naturally avoids such unobservable formation geometries. For these reasons, the cost function is amenable to a variety of future planning applications, such as to impose an inequality constraint on an indoor exploration planning problem.

The chapter is outlined as follows. The problem setup, notation, and other preliminaries

are described in Section 5.3. The optimization setup and results are described in Section 5.4. The optimal formations are evaluated experimentally in Section 5.5.

5.3 Problem Setup, Notation, and Preliminaries

Consider N agents along with M ranging tags distributed amongst them. Let $\tau_1, \tau_2, \ldots, \tau_M$ consist of unique physical points collocated with the ranging tags. Let $a_1, \ldots, a_N$ represent reference points on the agents themselves. The inter-tag range measurements are represented by a measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \ldots, M\}$ is the set of nodes, which is equivalent to the set of tag IDs, and $\mathcal{E}$ is the set of edges corresponding to the range measurements. Defining the set of agent IDs as $\mathcal{A} = \{1, \ldots, N\}$, it is convenient to define a simple "lookup function" $\ell : \mathcal{V} \to \mathcal{A}$ that returns the agent ID on which any particular tag is located. For example, if τ_i is physically on agent α, then $\ell(i) = \alpha$. An example scenario with three agents using this notation is shown in Figure 5.1.

5.3.1 State Definition and Range Measurement Model

Since the agents are rigid bodies, an orthonormal reference frame attached to their bodies can be defined. A position vector representing the position of point z, relative to point w,

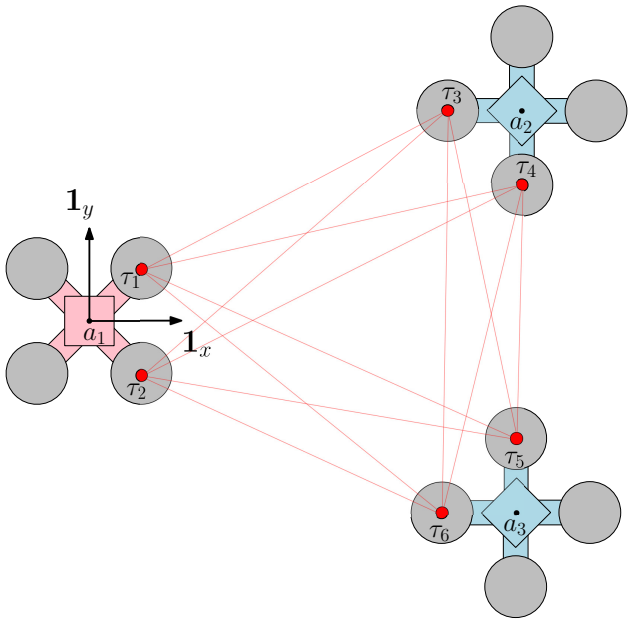

Figure 5.1: Problem setup and notation used. Each agent possesses a reference point a_α where α is the agent ID, as well as two tags τ_i, τ_j, where i, j are the tag IDs. $\mathbf{1}_x$ and $\mathbf{1}_y$ are vectors which represent the x and y axis of Agent 1's body frame. Throughout this chapter, the red agent is the arbitrary reference agent, and it will always be Agent 1 without loss of generality.

resolved in the body frame of agent α is denoted $\mathbf{r}_\alpha^{zw} \in \mathbb{R}^n$. The attitude of the body frame on agent α relative to the body frame on agent β is represented with a rotation matrix $\mathbf{C}_{\alpha\beta} \in SO(n)$ such that $\mathbf{r}_\alpha^{zw} = \mathbf{C}_{\alpha\beta}\mathbf{r}_\beta^{zw}$. The relative position and attitude between agents α and β, $(\mathbf{r}_\alpha^{a_\beta a_\alpha}, \mathbf{C}_{\alpha\beta})$ define the relative pose between them, and can be packaged together in a pose transformation matrix

$$\mathbf{T}_{\alpha\beta} = \begin{bmatrix} \mathbf{C}_{\alpha\beta} & \mathbf{r}_\alpha^{a_\beta a_\alpha} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(n). \tag{5.1}$$

Throughout this chapter, Agent 1 will be the arbitrary reference agent, such that the poses of all the other agents are expressed relative to Agent 1

$$\mathbf{x} = (\mathbf{T}_{12}, \ldots, \mathbf{T}_{1N}). \tag{5.2}$$

A generic range measurement between tag i and tag j is modelled as a function of the state $\mathbf{x} = (\mathbf{T}_{12}, \ldots, \mathbf{T}_{1N})$ with

$$y_{ij}(\mathbf{x}) = ||\mathbf{C}_{1\alpha}\mathbf{r}_\alpha^{\tau_i a_\alpha} + \mathbf{r}_1^{a_\alpha a_1} - (\mathbf{C}_{1\beta}\mathbf{r}_\beta^{\tau_j a_\beta} + \mathbf{r}_1^{a_\beta a_1})|| + v_{ij}, \tag{5.3}$$

where $\alpha = \ell(i)$, $\beta = \ell(j)$, and $v_{ij} \sim \mathcal{N}(0, \sigma_{ij}^2)$. This can be written compactly with the pose transformation matrices,

$$\begin{aligned} y_{ij}(\mathbf{x}) &= \left\| \mathbf{D}\mathbf{T}_{1\alpha} \begin{bmatrix} \mathbf{r}_\alpha^{\tau_i a_\alpha} \\ 1 \end{bmatrix} - \mathbf{D}\mathbf{T}_{1\beta} \begin{bmatrix} \mathbf{r}_\beta^{\tau_j a_\beta} \\ 1 \end{bmatrix} \right\| + v_{ij} \\ &\triangleq \left\| \mathbf{D}\mathbf{T}_{1\alpha}\mathbf{p}_i - \mathbf{D}\mathbf{T}_{1\beta}\mathbf{p}_j \right\| + v_{ij}, \end{aligned} \tag{5.4}$$

where $\mathbf{D} = [\mathbf{1}\ \mathbf{0}]$. The state $\mathbf{x} = (\mathbf{T}_{12}, \ldots, \mathbf{T}_{1N})$, written here as a tuple of poses, is an element of a composite Lie group of its own,

$$\mathbf{x} \in SE(n) \times \ldots \times SE(n) \triangleq SE(n)^{N-1}.$$

The group operation for $SE(n)^{N-1}$ is the elementwise matrix multiplication of the pose matrices in two arbitrary tuples, and the group inverse is the elementwise matrix inversion of the elements of the tuple $\mathbf{x}$. The $\oplus$ operator is defined here as

$$\mathbf{x} \oplus \delta\mathbf{x} = \left(\mathbf{T}_{12}\exp(\delta\boldsymbol{\xi}_2^\wedge),\ \ldots\ , \mathbf{T}_{1N}\exp(\delta\boldsymbol{\xi}_N^\wedge)\right), \tag{5.5}$$

where $\delta\boldsymbol{\xi}_i \in \mathbb{R}^m$, $\delta\mathbf{x} = [\delta\boldsymbol{\xi}_2^\mathsf{T}\ \ldots\ \delta\boldsymbol{\xi}_N^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{m(N-1)}$, and will be used throughout the chapter.

5.4 Optimization

The goal is to find the relative agent poses that, with respect to some metric, provide the best relative pose estimation results if the estimation were to be done exclusively using the

range measurements. The metric chosen in this chapter is based on Fisher information and the Cramér-Rao bound, which will be recalled here.

Definition 5.4.1 (Fisher information matrix [44])**.** Let $\mathbf{y} \in \mathbb{R}^q$ be a continuous random variable that is conditioned on a nonrandom variable $\mathbf{x} \in \mathbb{R}^n$. The *Fisher information matrix* (FIM) is defined as

$$\mathbf{I}(\mathbf{x}) = \mathbb{E}\left[\left(\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}}\right)^{\mathsf{T}}\left(\frac{\partial \log p(\mathbf{y}|\mathbf{x})}{\partial \mathbf{x}}\right)\right] \in \mathbb{R}^{n\times n}, \tag{5.6}$$

where $\mathbb{E}[\cdot]$ is the expectation operator and $p(\cdot)$ denotes a probability density function.

Theorem 5.4.1 (Cramér-Rao Bound [44])**.** Let $\mathbf{y} \in \mathbb{R}^q$ be a continuous random variable that is conditioned on $\mathbf{x} \in \mathbb{R}^n$. Let $\hat{\mathbf{x}}(\mathbf{y})$ be an unbiased estimator of $\mathbf{x}$, i.e., $\mathbb{E}[\mathbf{e}(\mathbf{x})] = \mathbb{E}[\hat{\mathbf{x}}(\mathbf{y}) - \mathbf{x}] = \mathbf{0}$. The *Cramér-Rao lower bound* states that

$$\mathbb{E}\left[\mathbf{e}(\mathbf{x})\mathbf{e}(\mathbf{x})^{\mathsf{T}}\right] \geq \mathbf{I}^{-1}(\mathbf{x}). \tag{5.7}$$

Theorem 5.4.2 (FIM for a Gaussian PDF)**.** Consider the nonlinear measurement model with additive Gaussian noise,

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{v}, \qquad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{5.8}$$

The Fisher information matrix is given by

$$\mathbf{I}(\mathbf{x}) = \mathbf{H}(\mathbf{x})^{\mathsf{T}}\mathbf{R}^{-1}\mathbf{H}(\mathbf{x}), \tag{5.9}$$

where $\mathbf{H}(\mathbf{x}) = \partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x}$.

The Cramér-Rao bound represents the minimum variance achievable by any unbiased estimator. Hence, motivated by Theorem 5.4.1, an estimation cost function J_{est} is defined

$$J_{\text{est}}(\mathbf{x}) = -\log\det \mathbf{I}(\mathbf{x}), \tag{5.10}$$

which will be minimized with the agent relative poses $\mathbf{x}$ as the optimization variables. The logarithm of the determinant of $\mathbf{I}(\mathbf{x})$ is one option amongst many choices of matrix norms, such as the trace or Frobenius norm. We have found the chosen cost function to behave well in terms of numerical optimization and, most importantly, goes to infinity when the FIM becomes non-invertible. The state $\mathbf{x}$ is locally observable from measurements $\mathbf{y}$ if the measurement Jacobian $\mathbf{H}(\mathbf{x})$ is full column rank, which also makes the FIM full rank. As will be seen in Section 5.4.2, non-invertibility of the FIM also corresponds to formations that result in unobservable relative poses, which should be avoided.

To create a measurement model in the form of (5.8), the range measurements are all

concatenated into a single vector

$$\mathbf{y}(\mathbf{x}) = \underbrace{[\ldots y_{ij}(\mathbf{x}) \ldots]^{\mathsf{T}}}_{\triangleq \mathbf{g}(\mathbf{x})} + \mathbf{v}, \quad \forall (i,j) \in \mathcal{E},\ \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}),$$

where $\mathbf{R} = \text{diag}(\ldots, \sigma_{ij}^2, \ldots)$. It would be possible to directly descend the cost in (5.10) with an optimization algorithm such as gradient descent, if not for the fact that the state $\mathbf{x}$ does not belong to Euclidean space $\mathbb{R}^n$ but rather $SE(n)^{N-1}$. As such, the expression $\partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x}$ is meaningless unless properly defined.

5.4.1 On-manifold Cost and Gradient Descent

The modification employed in this chapter is to reparameterize the measurement model by defining $\mathbf{x} = \bar{\mathbf{x}} \oplus \delta\mathbf{x}$, leading to

$$\mathbf{y} = \mathbf{g}(\bar{\mathbf{x}} \oplus \delta\mathbf{x}) + \mathbf{v} \triangleq \bar{\mathbf{g}}(\delta\mathbf{x}). \tag{5.11}$$

The state $\bar{\mathbf{x}}$ will represent the current optimization iterate, which will be updated using $\delta\mathbf{x}$.

Since the argument of the new measurement model $\bar{\mathbf{g}}(\delta\mathbf{x})$ now belongs to Euclidean space $\mathbb{R}^{m(N-1)}$, it is possible to compute the "local" approximation to the FIM [73] at $\mathbf{x} = \bar{\mathbf{x}}$ with $\mathbf{I}(\bar{\mathbf{x}}) = \mathbf{H}(\bar{\mathbf{x}})^{\mathsf{T}} \mathbf{R}^{-1} \mathbf{H}(\bar{\mathbf{x}})$ where

$$\mathbf{H}(\bar{\mathbf{x}}) = \left. \frac{\partial \mathbf{g}(\bar{\mathbf{x}} \oplus \delta\mathbf{x})}{\partial \delta\mathbf{x}} \right|_{\delta\mathbf{x}=\mathbf{0}},$$

and evaluate the cost function $J_{\text{est}}(\bar{\mathbf{x}}) = -\log\det \mathbf{I}(\bar{\mathbf{x}})$. Finally, an on-manifold gradient descent step with step size γ can be taken with

$$\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} \oplus \left(-\gamma \left. \frac{\partial J_{\text{est}}(\bar{\mathbf{x}} \oplus \delta\mathbf{x})}{\partial \delta\mathbf{x}} \right|_{\delta\mathbf{x}=\mathbf{0}} \right)^{\mathsf{T}}. \tag{5.12}$$

The proposed gradient descent procedure is a standard approach to optimization on matrix manifolds [74]. From a differential-geometric point of view, an approximation to the FIM is computed in the tangent space of the current optimization iterate $\bar{\mathbf{x}}$, which is a familiar Euclidean vector space. A gradient descent step is computed in the tangent space, and the result is *retracted* back to the manifold $SE(n)^{N-1}$ using the retraction $\mathcal{R}_{\bar{\mathbf{x}}}(\delta\mathbf{x}) = \bar{\mathbf{x}} \oplus \delta\mathbf{x}$.

5.4.2 Cost Function Implementation

Creating an implementable expression for the cost function $J_{\text{est}}(\bar{\mathbf{x}}) = -\log\det\left(\mathbf{H}(\bar{\mathbf{x}})^{\mathsf{T}} \mathbf{R}^{-1} \mathbf{H}(\bar{\mathbf{x}})\right)$ eventually amounts to computing the Jacobian of the range measurement model (5.4) with respect to $\delta\boldsymbol{\xi}_\alpha$ and $\delta\boldsymbol{\xi}_\beta$. To see this,

$$\mathbf{H}(\bar{\mathbf{x}}) = \begin{bmatrix} \vdots \\ \mathbf{H}^{ij}(\bar{\mathbf{x}}) \\ \vdots \end{bmatrix},$$
$$\mathbf{H}^{ij}(\bar{\mathbf{x}}) = [\mathbf{0} \dots \mathbf{H}^{ij}_{\alpha}(\bar{\mathbf{x}}) \dots \mathbf{H}^{ij}_{\beta}(\bar{\mathbf{x}}) \dots \mathbf{0}],$$

where

$$\mathbf{H}^{ij}_{\alpha}(\bar{\mathbf{x}}) \triangleq \left.\frac{\partial y_{ij}(\bar{\mathbf{x}} \oplus \delta\mathbf{x})}{\partial \delta\boldsymbol{\xi}_{\alpha}}\right|_{\delta\mathbf{x}=\mathbf{0}},$$
$$\mathbf{H}^{ij}_{\beta}(\bar{\mathbf{x}}) \triangleq \left.\frac{\partial y_{ij}(\bar{\mathbf{x}} \oplus \delta\mathbf{x})}{\partial \delta\boldsymbol{\xi}_{\beta}}\right|_{\delta\mathbf{x}=\mathbf{0}}.$$

The row matrix $\mathbf{H}^{ij}(\bar{\mathbf{x}}) \in \mathbb{R}^{1\times m(N-1)}$ represents the Jacobian of a single range measurement y_{ij} with respect to the full state perturbation $\delta\mathbf{x}$. This resulting matrix will be zero everywhere except for two blocks $\mathbf{H}^{ij}_{\alpha}(\bar{\mathbf{x}}) \in \mathbb{R}^{1\times m}$ and $\mathbf{H}^{ij}_{\beta}(\bar{\mathbf{x}}) \in \mathbb{R}^{1\times m}$, respectively located at the α^{th} and β^{th} block columns, and have closed-form expressions derived as follows. Let $y_{ij}(\mathbf{x}) = y_{ij}(\bar{\mathbf{x}}) + \delta y_{ij} \triangleq \bar{y}_{ij} + \delta y_{ij}$, $\mathbf{T}_{1\alpha} = \bar{\mathbf{T}}_{1\alpha}\exp(\delta\boldsymbol{\xi}^{\wedge}_{\alpha})$, $\mathbf{T}_{1\beta} = \bar{\mathbf{T}}_{1\beta}\exp(\delta\boldsymbol{\xi}^{\wedge}_{\beta})$, and $v_{ij} = 0$. The terms $\delta y_{ij}, \delta\boldsymbol{\xi}_{\alpha}, \delta\boldsymbol{\xi}_{\beta}$ are assumed to be small quantities, which motivates, for example, the approximation $\exp(\delta\boldsymbol{\xi}^{\wedge}_{\alpha}) \approx \mathbf{1} + \delta\boldsymbol{\xi}^{\wedge}_{\alpha}$. Equation (5.4) becomes

$$(\bar{y}_{ij} + \delta y_{ij})^2 = \left(\mathbf{D}\bar{\mathbf{T}}_{1\alpha}(\mathbf{1} + \delta\boldsymbol{\xi}^{\wedge}_{\alpha})\mathbf{p}_i - \mathbf{D}\bar{\mathbf{T}}_{1\beta}(\mathbf{1} + \delta\boldsymbol{\xi}^{\wedge}_{\beta})\mathbf{p}_j\right)^{\mathsf{T}} \left(\mathbf{D}\bar{\mathbf{T}}_{1\alpha}(\mathbf{1} + \delta\boldsymbol{\xi}^{\wedge}_{\alpha})\mathbf{p}_i - \mathbf{D}\bar{\mathbf{T}}_{1\beta}(\mathbf{1} + \delta\boldsymbol{\xi}^{\wedge}_{\beta})\mathbf{p}_j\right),$$

which, after expanding and neglecting higher-order terms, leads to

$$2\bar{y}_{ij}\delta y_{ij} = 2(\mathbf{p}_i^{\mathsf{T}}\bar{\mathbf{T}}_{1\alpha}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}} - \mathbf{p}_j^{\mathsf{T}}\bar{\mathbf{T}}_{1\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}})\mathbf{D}\bar{\mathbf{T}}_{1\alpha}\delta\boldsymbol{\xi}^{\wedge}_{\alpha}\mathbf{p}_i + 2(\mathbf{p}_j^{\mathsf{T}}\bar{\mathbf{T}}_{1\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}} - \mathbf{p}_i^{\mathsf{T}}\bar{\mathbf{T}}_{1\alpha}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}})\mathbf{D}\bar{\mathbf{T}}_{1\beta}\delta\boldsymbol{\xi}^{\wedge}_{\beta}\mathbf{p}_i. \quad (5.13)$$

Next, it is straightforward to define a simple operator $(\cdot)^{\odot}$, as per [43], such that $\boldsymbol{\xi}^{\wedge}\mathbf{p} = \mathbf{p}^{\odot}\boldsymbol{\xi}$. Rearranging (5.13) yields

$$\delta y_{ij} = \frac{(\mathbf{p}_i^{\mathsf{T}}\bar{\mathbf{T}}_{1\alpha}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}} - \mathbf{p}_j^{\mathsf{T}}\bar{\mathbf{T}}_{1\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}})}{\bar{y}_{ij}}\mathbf{D}\bar{\mathbf{T}}_{1\alpha}\mathbf{p}_i^{\odot}\delta\boldsymbol{\xi}_{\alpha} - \underbrace{\frac{(\mathbf{p}_i^{\mathsf{T}}\bar{\mathbf{T}}_{1\alpha}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}} - \mathbf{p}_j^{\mathsf{T}}\bar{\mathbf{T}}_{1\beta}^{\mathsf{T}}\mathbf{D}^{\mathsf{T}})}{\bar{y}_{ij}}}_{\triangleq\boldsymbol{\rho}_{ij}^{\mathsf{T}}}\mathbf{D}\bar{\mathbf{T}}_{1\beta}\mathbf{p}_j^{\odot}\delta\boldsymbol{\xi}_{\beta}. \quad (5.14)$$

The term $\boldsymbol{\rho}_{ij}$ is the physical unit direction vector between tags i and j, resolved in Agent 1's body frame. From (5.14) it then follows that

$$\frac{\partial y_{ij}}{\partial\delta\boldsymbol{\xi}_{\alpha}} = \boldsymbol{\rho}_{ij}^{\mathsf{T}}\mathbf{D}\bar{\mathbf{T}}_{1\alpha}\mathbf{p}_i^{\odot}, \qquad \frac{\partial y_{ij}}{\partial\delta\boldsymbol{\xi}_{\beta}} = -\boldsymbol{\rho}_{ij}^{\mathsf{T}}\mathbf{D}\bar{\mathbf{T}}_{1\beta}\mathbf{p}_j^{\odot}.$$

The cost function J_{est} is visualized for varying agent position in the top row of Figure 5.2, where the red dot shows the minimum found within that view. Looking at the top-left plot

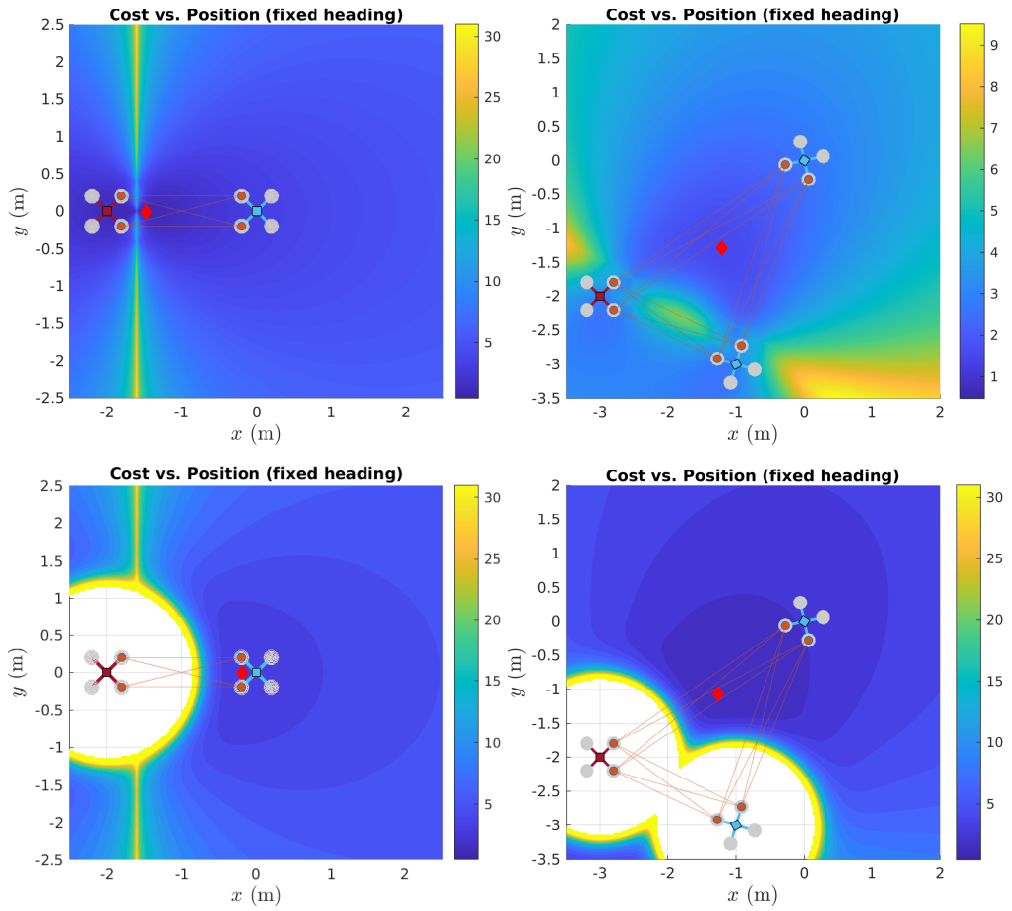

Figure 5.2: All four plots show the value of the cost with varying agent position (right agent for two-agent scenario, top agent for three-agent scenario), while maintaining fixed heading. The top row shows only the estimation cost J_{est}, while the bottom row shows the total cost J including the collision avoidance term.

of Figure 5.2, there is a vertical line of high cost near the agent on the left, corresponding exactly to when all four tags line up, leading to an unobservable formation. Similarly, the three-agent scenario in the top-right plot of Figure 5.2 shows a high cost when the agents are nearly all on the same line, which is a situation of near-unobservability. However, as can be seen in the top-left plot, the minimum is unacceptably close to the left agent, which would cause them to collide. Indeed, we have observed that naively descending the cost J_{est} alone leads to all the agents collapsing into each other. An explanation for this behavior is that when agents are closer together, changes in attitude result in larger changes in the range measurements, which increases Fisher information. Nevertheless, in practice, collisions must be avoided, and this is done by augmenting the cost with an additional collision avoidance term $J_{\text{col}}(\mathbf{x})$, such that the total cost $J(\mathbf{x})$ is

$$J(\mathbf{x}) = J_{\text{est}}(\mathbf{x}) + J_{\text{col}}(\mathbf{x}), \qquad J_{\text{col}}(\mathbf{x}) = \sum_{\substack{\alpha,\beta\in\mathcal{A} \\ \alpha\neq\beta}} J_{\text{col}}^{\alpha\beta}(\mathbf{x}), \tag{5.15}$$

where a collision avoidance cost from [75] is used,

$$J_{\text{col}}^{\alpha\beta}(\mathbf{x}) = \left(\min \left\{ 0, \frac{\left\| \mathbf{r}_1^{a_\alpha a_\beta} \right\|^2 - R^2}{\left\| \mathbf{r}_1^{a_\alpha a_\beta} \right\|^2 - d^2} \right\} \right)^2 . \tag{5.16}$$

The term R represents an "activation radius" and d is the safety collision avoidance radius. In this chapter, the agent relative position is expressed as a function of pose matrices with

$$\mathbf{r}_1^{a_\alpha a_\beta} = \mathbf{D}\mathbf{T}_{1\alpha}\mathbf{b} - \mathbf{D}\mathbf{T}_{1\beta}\mathbf{b},$$

where $\mathbf{D} = [\mathbf{1}\ \mathbf{0}]$, $\mathbf{b} = [\mathbf{0}\ 1]^\mathsf{T}$. The new cost function J is plotted on the bottom row of Figure 5.2, showing the effect of the collision avoidance term. Finally, one is now ready to descend the cost directly with

$$\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} \oplus \left(-\gamma \left. \frac{\partial J(\bar{\mathbf{x}} \oplus \delta\mathbf{x})}{\partial \delta\mathbf{x}} \right|_{\delta\mathbf{x}=\mathbf{0}} \right)^\mathsf{T} . \tag{5.17}$$

In this work, the Jacobian of J_{est} is computed numerically with finite difference [5], and the optimization is only done offline for the following reasons. The solution to the optimization problem is only a function of some physical properties, the measurement graph $\mathcal{G}$, and the number of robots N. For any experiments that use the same hardware, the physical properties such as the safety radius, tag locations, and measurement covariances, all remain constant. The measurement graph $\mathcal{G}$ can often also be assumed to be constant and fully connected. Even though full-connectedness is not required in the proposed approach, technologies such as UWB often have a ranging limit that is well beyond the true ranges between all robots in the experiment. Hence, it is straightforward to precompute optimal formations for varying robot numbers N with fully-connected measurement graphs, and to store the solutions in memory onboard each robot. The computation time for this method, for 10 agents or less, is on the order of a few minutes.

Nevertheless, a distributed, real-time implementation is required for varying measurement graphs, which is likely to arise in the presence of obstacles that block line-of-sight. Such a scenario requires simultaneously satisfying obstacle avoidance constraints and perhaps other planning objectives, which is beyond the scope of this chapter.

5.4.3 Optimization Results

The gradient descent in (5.17) is performed with a step size of $\gamma = 0.1$, an activation radius of $R = 2$ m, and a safety radius of $d = 1$ m. Each agent has two tags located at

$$\mathbf{r}_\alpha^{\tau_i a_\alpha} = [0.2\ 0.2]^\mathsf{T}, \quad \text{and} \quad \mathbf{r}_\alpha^{\tau_j a_\alpha} = [0.2\ -0.2]^\mathsf{T},$$

where $\alpha = \ell(i) = \ell(j)$ and the units are in meters. Figure 5.3 shows quadcopter formations at convergence for 3, 4, 5, and 10 agents, each with a fully-connected measurement graph $\mathcal{G}$,

except for edges corresponding to two tags on the same agent. The results shown here are intuitive, with the three- and four-agent scenarios corresponding to an equilateral triangle and square, respectively. However, with increasing agent numbers, regular polygon formations are no longer optimal, as can be seen in the five- and ten-agent scenarios.

Since the treatment in this chapter is general to an arbitary measurement graph $\mathcal{G}$, provided the FIM remains maximum rank, optimization is also performed for a non-fully-connected measurement graph. The results for this along with a 3D scenario are shown in Figure 5.4. In 3D, the robot relative poses are represented with elements of $SE(3)$, but with four degrees of freedom corresponding to the three translational components and heading. This is because two-tag robots are used in these simulations, making relative roll and pitch between robots unobservable without more sensors. Hence, roll and pitch are excluded from the optimization and their values are fixed to zero. Moreover, from an application standpoint, both ground vehicles and quadcopter-type aerial vehicles only have heading as a rotational degree of freedom available for planning.

The cost function J has many local minima associated with the various geometric symmetries contained in the problem. That is, for two-tag agents, a "flip" of 180 degrees in heading for each agent leads to a local minimum, as do symmetries in the formation positions. The solutions presented are the result of a trial-and-error process with hand-picked initial guesses, where all local minimums encountered in this process provided sufficiently low covariance. Therefore, it can be argued that finding the global minimum is not necessary, especially if only avoiding the unobservable formations of high cost is required.

5.4.4 Validation on a Least-Squares Estimator

To validate the claim that descending the cost improves the estimation performance, a non-linear least-squares estimator is used. At regular iterates $\bar{\mathbf{x}}$ of the optimization trajectory, a small 2000-trial Monte Carlo experiment is performed, where in each trial a set of range measurements are generated with $\mathbf{y} = \mathbf{g}(\bar{\mathbf{x}}) + \mathbf{v}$, $\mathbf{v} = \mathcal{N}(\mathbf{0}, \mathbf{R})$. Then, an on-manifold Gauss-Newton procedure [43] is used to solve

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \sum_{\alpha=2}^{N} \left\| \log(\mathbf{C}_{1\alpha}^{\mathsf{T}} \check{\mathbf{C}}_{1\alpha})^{\vee} \right\|_{\check{\mathbf{P}}_{\alpha}}^{2} + \frac{1}{2} \left\| \mathbf{y} - \mathbf{g}(\mathbf{x}) \right\|_{\mathbf{R}}^{2}, \tag{5.18}$$

where $\|\mathbf{e}\|_{\mathbf{M}}^{2} = \mathbf{e}^{\mathsf{T}} \mathbf{M}^{-1} \mathbf{e}$ denotes a squared Mahalanobis distance, and an attitude prior with "mean" $\check{\mathbf{C}}_{1\alpha}$ and covariance $\check{\mathbf{P}}_{\alpha}$ is also included for each agent. It turns out that minimization of only the second term in (5.18) yields unacceptably poor estimation performance, as the solution often converges to local minimums depending on the initial guess. The inclusion of a low-covariance attitude prior, which is practically obtained by dead-reckoning on-board

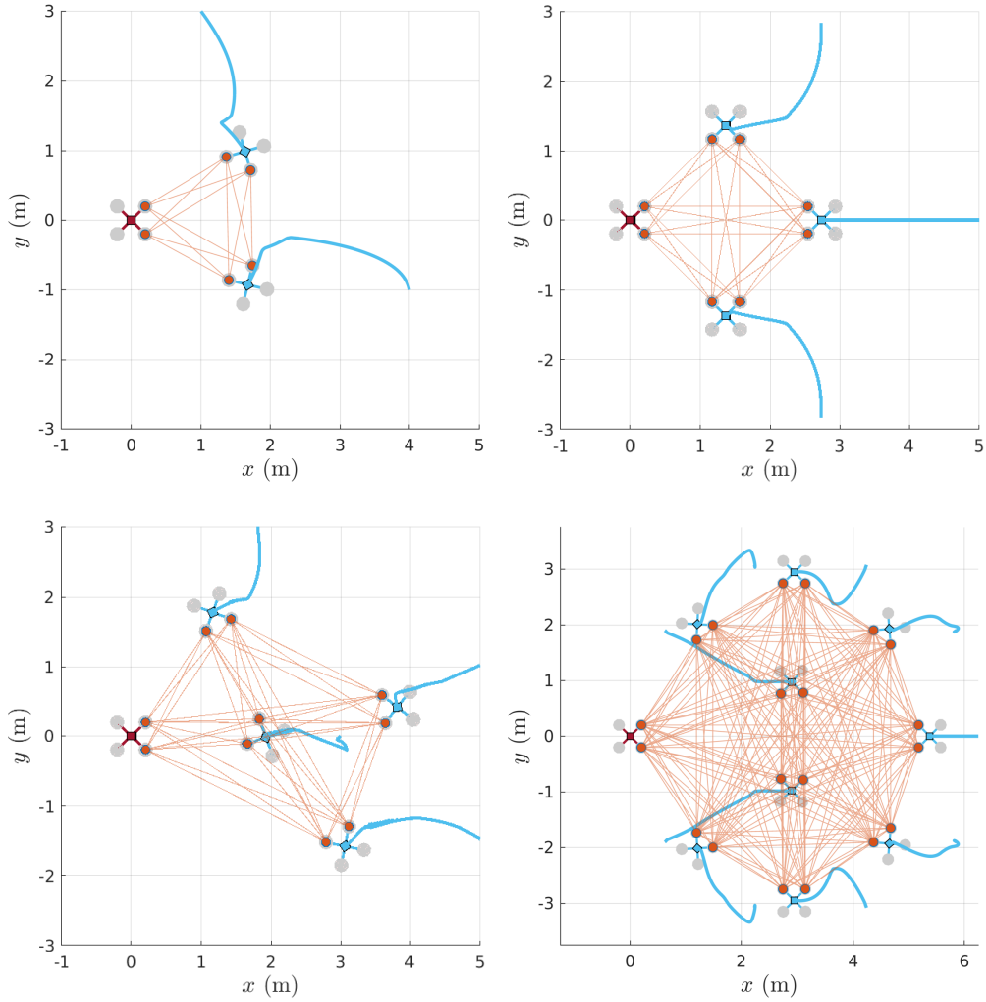

Figure 5.3: Final locally optimal formations for 3, 4, 5, and 10 agents, with the optimization paths shown in blue.

gyroscope measurements, is critical for obtaining low estimation error.

Figure 5.6 shows the value of the cost throughout the optimization trajectory, as well as the mean squared estimation error over the $K = 2000$ Monte Carlo trials per optimization step. The true agent poses are initialized in a near-straight line, as shown in Figure 5.5, and the covariances used are $\mathbf{R} = 0.1^2\mathbf{1}$ m^2, $\check{\mathbf{P}}_\alpha = 0.08^2$ rad^2. The mean squared estimation error (MSE) is calculated with

$$\text{MSE} = \frac{1}{K}\sum_{k=1}^{K} \delta\boldsymbol{\xi}^\mathsf{T}\delta\boldsymbol{\xi}, \qquad \delta\boldsymbol{\xi} = \begin{bmatrix} \log(\bar{\mathbf{T}}_{12}^{-1}\hat{\mathbf{T}}_{12})^\vee \\ \vdots \\ \log(\bar{\mathbf{T}}_{1N}^{-1}\hat{\mathbf{T}}_{1N})^\vee \end{bmatrix}, \tag{5.19}$$

and shows a clear correlation with the cost function. This provides evidence for the fact that descending the proposed cost function also reduces the estimation error.

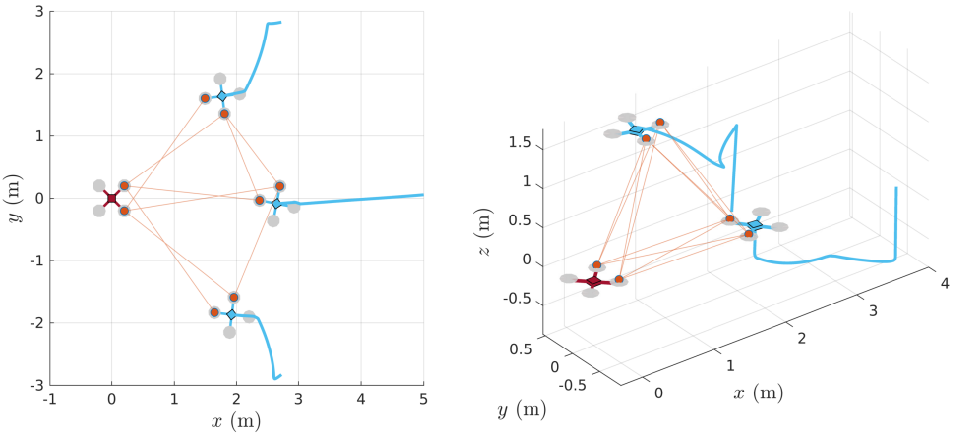

Figure 5.4: **Left:** Optimal formation with sparse measurement graph. **Right:** Optimal formation with 3D position and heading as design variables.

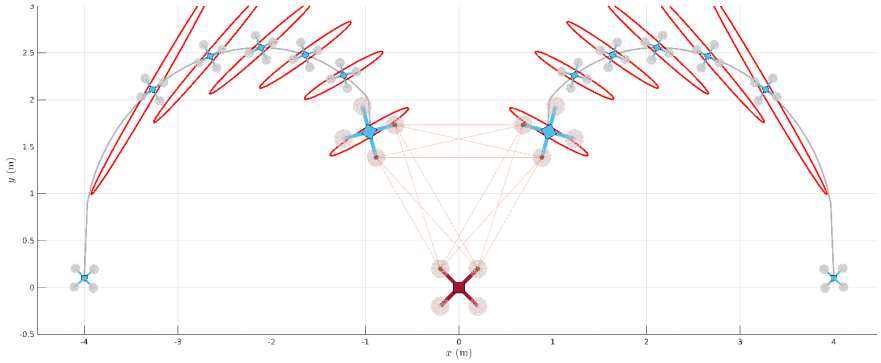

Figure 5.5: Trajectory taken during optimization with superimposed 1σ equal-probability contours corresponding to the Cramér-Rao bound. The ellipsoids for the starting positions are too large to fit in the figure.

5.5 Experimental Evaluation

An estimator is also run with three PX4-based Uvify IFO-S quadcopters in order to experimentally validate the claim that descending the proposed cost function results in improved estimation performance. The quadcopters start by flying in a line formation and, after 30 seconds, proceed to a triangle formation computed using the proposed framework for another 30 seconds, as shown in Figure 5.8. Figure 5.7 shows the position estimation error using the least-squares estimator presented in Section 5.4.4. Real gyroscope measurements are used to obtain an attitude prior at all times. Range measurements are synthesized with a standard deviation of 10 cm using ground truth vehicle poses obtained from a motion capture system. The UWB tags are simulated to be 17 cm apart, corresponding to extremities of the propeller arms. As can be seen in Figure 5.7, moving to the optimal triangle formation, from one of the worst starting formations results in a 68% reduction in estimation variance.

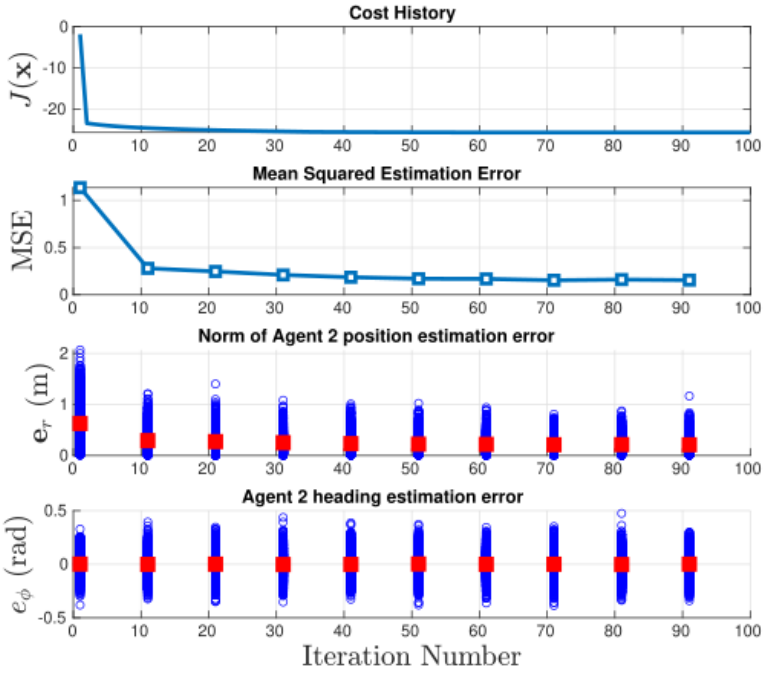

Figure 5.6: Cost function, along with various metrics of a least-squares estimator, obtained from 2000 Monte-Carlo trials at various points during the optimization. In the bottom two plots, the red squares denote the average norm of the respective estimation errors.

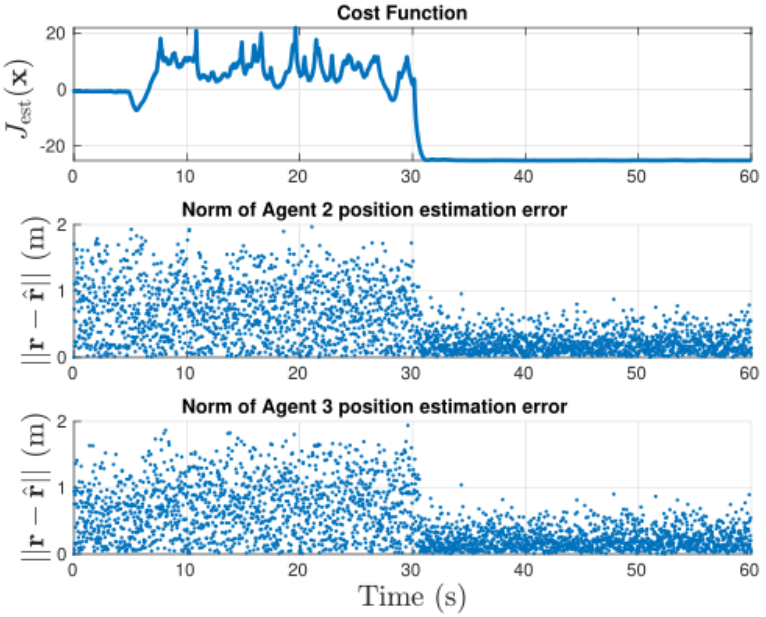

Figure 5.7: Experimental results using a least squares estimator. From 0 s to 30 s, the quadcopters are in a line formation and the average positioning error is 0.77 m. From 30 s to 60 s, the quadcopters are in an optimal formation and the average positioning error is 0.22 m, a 68% reduction.

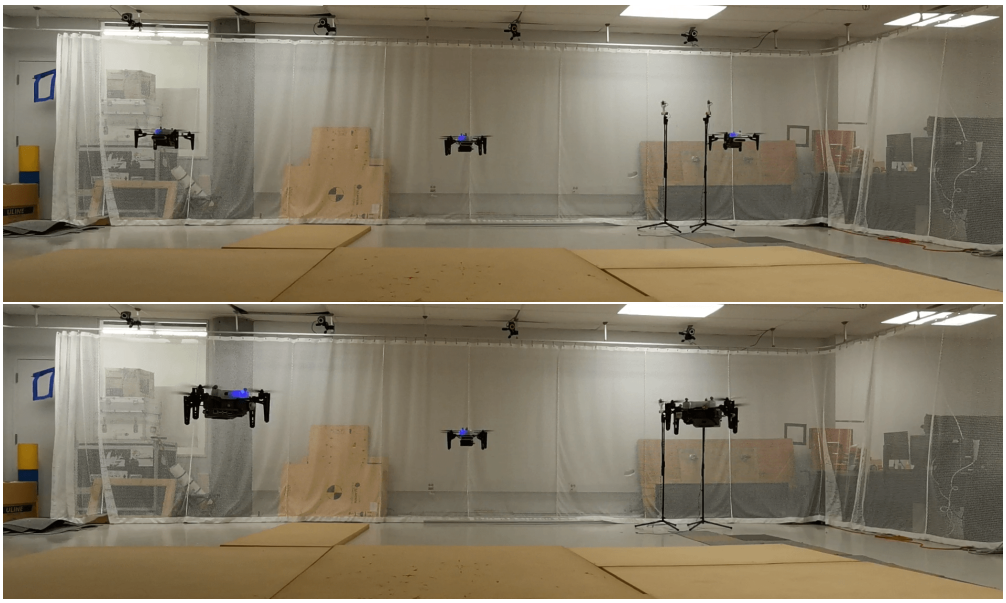

Figure 5.8: **Top:** Three quadcopters in an initial straight line formation. **Bottom:** Quadcopters in an optimal triangle formation.

5.6 Conclusion and Future Work

This chapter proposes a method for computing optimal formations that minimize relative pose estimation variance. This is done by introducing a cost function that maximizes Fisher information and also penalizes cases where robots are too close to each other, in order to avoid collisions. On-manifold gradient descent is used to descend the cost, and both simulation and experiment show that large improvements in estimation various can be seen when robots move away from singular formations to their optimal formations. A key feature of the proposed cost function is that it goes to infinity when formations are singular/unobservable formations. Moreover, the results show that the largest reductions in estimation variance occur immediately when moving away from singular formations, and that there are diminishing returns when approaching the optimal formations.

Motivated by the results of this chapter, the following ideas could lead to interesting, practical uses.

- **Using the cost function as an inequality constraint.**
  This point is especially motivated by the observation that the largest accuracy gains are obtained when moving away from unobservable formations, and that estimation variance is approximately the same in a reasonably large region around the optimal formations. Hence, a natural use case for the proposed cost function in (5.15) is to use it as an inequality constraint on some other application-oriented planning problem

described by cost $P(\mathbf{x})$. That is,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} P(\mathbf{x}) \tag{5.20}$$

$$\text{such that } J(\mathbf{x}) < \epsilon \tag{5.21}$$

where ϵ is some arbitrary threshold. An example is to design trajectories that move a group of agents through hallways and around obstacles, while maintaining the cost below this threshold.

- **Decentralized computation of the cost.**
  This chapter assumes that all robot poses are available for computation of the cost function $J(\mathbf{x})$. However, in cases where some robots only have access to a subset of $\mathbf{x}$, a decentralized computation would be needed. This could be performed in a way similar to [71].

Implicitly from the fact that the Fisher information matrix is in general not singular, this chapter shows that the relative position and heading is statically observable from range measurements alone, provided that robots have two UWB tags on them. The next chapter will exploit this to finally design a concrete, decentralized estimator that can fuse range measurements with other lightweight, cheap sensors on-board each robot. Furthermore, the treatment will be mathematically general to any process and measurement model, therefore applicable to a large variety of robotics problems.

Chapter 6

On-manifold Decentralized State Estimation using Pseudomeasurements and Preintegration

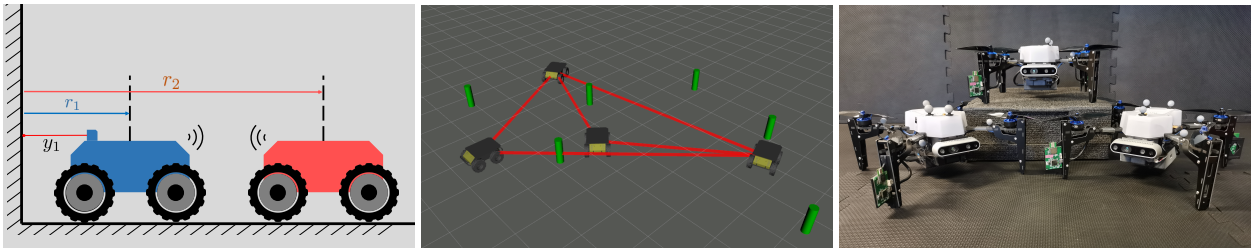

Figure 6.1: Three examples of decentralized estimation problems within the scope of this chapter. **Left:** A toy problem with 1D robots, each estimating both of their positions. **Middle:** A problem with an incomplete communication graph. Robots observe landmarks, have range measurements to each other, and estimate their own and neighbor absolute poses. **Right**: A more complicated experimentally-tested problem, where robots equipped with ultra-wideband radios estimate both their own absolute pose and relative poses of neighbors, in addition to IMU biases.

6.1 Summary

This chapter addresses the problem of decentralized, collaborative state estimation in robotic teams. In particular, this chapter considers problems where individual robots estimate similar physical quantities, such as each other's position relative to themselves. The use of *pseudomeasurements* is introduced as a means of modelling such relationships between robots' state estimates, and is shown to be a tractable way to approach the decentralized state estimation problem. Moreover, this formulation easily leads to a general-purpose observability test that simultaneously accounts for measurements that robots collect from their own sensors, as well as the communication structure within the team. Finally, input preintegration is

proposed as a communication-efficient way of sharing odometry information between robots, and the entire theory is appropriate for both vector-space and Lie-group state definitions. The proposed framework is evaluated on three different simulated problems, and one experiment involving three quadcopters.

6.2 Introduction

The previous chapter established that having robots outfitted with UWB tags led to observable relative poses, and the effect of the relative poses themselves on the estimation accuracy was studied. However, this was all performed in a centralized way, where all measurements were assumed to be available to a central computer. To run an algorithm on each robot's processor, the estimator needs to be fundamentally decentralized. Decentralized state estimation is a fundamental requirement for real-world multi-robot deployments. Whether the task is collaborative mapping, relative localization, or collaborative dead-reckoning, the multi-robot estimation problem seeks to estimate the state of each robot given the measurements obtained locally by sensors on each robot in the team. This problem is made difficult by the fact that not all robots can communicate with each other and, furthermore, that high-frequency sensor measurements would require substantial communication bandwidth to simply share across the team. The *centralized estimator* is a computing unit that can somehow collect all the sensor measurements on each robot, and then fuse them all to jointly estimate the states of every robot in one large system. Theoretically, the centralized estimator has the lowest possible estimation error variance, and all decentralized implementations attempt to match its performance.

A common approach is for robots to share their current state and associated covariance rather than of a history of measurement values [76–78]. This approach has the benefit of low communication cost and fixed message size, but suffers from a well-known issue of not being able to compute cross-correlations between the robots' state estimates [79]. Furthermore, in certain problems, robots may be estimating the same physical quantities. As an example, consider two robots estimating each other's position, in addition to their own positions, as shown in Figure 6.1 (left). Their state vectors are both robots' position, and therefore both seek to estimate the same physical quantities, a situation referred to here as *full state overlap*. When robot states have similar, if not identical state definitions, it is straightforward to compute the error between their state estimates using simple subtraction. However, for more complicated problems, especially those with state definitions belonging to arbitrary Lie groups, a generalized measure of error between different robots' state estimates must be introduced.

This chapter introduces a new framework for formulating decentralized multi-robot estimation problems. The proposal is to use *pseudomeasurements* as a tractable and effective way to model any generic nonlinear relationship between robot state definitions, including full or partial state overlap as special cases. A pseudomeasurement is introduced for each edge in the communication graph, and the framework also naturally leads to a novel observability test that takes into account both the local measurements obtained by each robot in addition to the communication structure between them.

Secondly, the use of preintegration is proposed as a means of sharing odometry information over an arbitrary duration of time, in a lossless matter. The naive solution is for robots to share a history of odometry measurements since the last time they communicated, which has processing, memory, and communication requirements that grow linearly with the time interval between communications. Preintegration provides a constant-time, constant-memory, and constant-communication alternative that is algebraically identical to simply sharing the input measurements themselves. This makes preintegration a natural choice for multi-robot estimation problems. Moreover, preintegration preserves statistical independence assumptions that typical Kalman filtering prediction steps rely on. Preintegration is best known from the visual-inertial odometry literature [50, 80], where the same concept, adapted for relative pose estimation is introduced in [81]. This chapter generalizes the concept to other common process models in robotics, presents a solution for simultaneous input bias estimation, and also further proposes a deep autoencoder to compress the associated covariance information.

Finally, the theory here is developed for general on-manifold state definitions, including familiar linear vector spaces. The proposed solution is general to any state definition, process model, and measurement model subject to typical Gaussian noise assumptions. This chapter does not focus on the treatment of cross-correlations, and hence employs the simple, well-known covariance intersection (CI) [76, 77] method. This allows for an arbitrary communication graph within the robot team, while remaining lightweight and avoiding cumbersome bookkeeping. The main drawback is that CI is proven to yield sub-optimal estimation. However, in practice, the performance can remain adequate, and sometimes very comparable to centralized estimation, as shown here from simulated and experimental results.

The remainder of this chapter is as follows. In Section 6.3, related work is discussed. Mathematical preliminaries and notation are shown in Section 6.4. The chapter then starts with a simplified "toy" problem showcasing the proposed method in Section 6.5, and the theory is generalized in Section 6.8. Finally, Section 6.8 contains an application to a ground robot simulation, Section 6.9 applies the method to a more complicated experimental quadcopter problem.

6.3 Related Work

There are many sub-problems associated with the overall decentralized state estimation problem. Even if communication links between robots are assumed to be lossless and have infinite bandwidth, there is still the issue of propagating information over general, incomplete communication graphs. For two robots, it is straightforward to compute centralized-equivalent estimators on each robot as done in [82], which use information filters to accumulate the information from a series of measurements, and then communicate this quantity. In [82], centralized-equivalent solutions are also derived for fully connected graphs, as well as tree-shaped graphs. However, they show that for generic graphs, it is impossible to obtain a centralized-equivalent estimate with only neighboring knowledge, and that more knowledge of the graph topology is required.

In [83], a general centralized-equivalent algorithm is presented for arbitrary time-varying graphs, and is formulated over distributions directly, hence allowing inference using any algorithm such as an extended or sigma-point Kalman filter. Robots must still share raw measurements with each other, therefore requiring substantial bookkeeping. The approach is extended to the SLAM problem in [84]. In [85], the centralized Kalman filter equations are decomposed using a singular value decomposition to generate independent equations that each robot can compute. Provided that robots have broadcasting ability, and obtain direct pose measurements of their neighbors, a centralized-equivalent solution can be obtained. The consensus Kalman filter developed in [86, 87] aims to asymptotically send the state of n arbitrary nodes to a common value, which is effectively a problem with full state overlap. Alternate consensus approaches such as "consensus on information" or "consensus on measurements" are shown in [88]. Moreover, the problem does not consider the fact that robots collect their own, seperate odometry measurements.

As previously mentioned, one of the simplest solutions to the decentralized estimation problem is to use covariance intersection, presented in [76, 77]. CI conservatively assumes maximum correlation between robot estimates. Although the performance is theoretically suboptimal, the implementation is extremely simple, and imposes no constraints whatsoever on the communication frequency or graph topology. In [78], covariance intersection is applied to a collaborative localization problem, where each robot estimates their own absolute state given direct relative pose measurements to other robots. Covariance intersection has recently been exploited for the fusion of poses on Lie groups in [89]. In [90], an EKF-like filter is used for decentralized estimation where cross-correlations are also explicitly tracked for both the prediction step and the fusion of local measurements. When relative measurements are encountered, an improved approximation to the joint covariance matrix is developed, which

outperforms CI. The approach in [90] assumes that process model inputs between robots are uncorrelated, which is not applicable in some of the problems in this chapter. The work in [91] builds off of [90] to solve a full 3D relative pose estimation problem where each robot has a camera and an IMU.

Another approach using scattering theory has recently been presented for two robots in [92, 93], with the objective of reducing the communication cost associated with the communication of high-rate sensor measurements. Also making reference to the IMU preintegration technique in [50, 80], covariance pre-computations are derived in [92] and extended to also include the mean in [93]. It is shown that by sharing pre-computed matrices with the same size as the state vector, a centralized-equivalent state estimate can be directly obtained with no measurement reprocessing. However, the generalization to more robots does not seem straightforward.

6.4 Preliminaries

This chapter will address problems where an individual robot's motion and measurements are modelled in the standard form of

$$\begin{aligned} \mathcal{X}_{i_k} &= \mathbf{f}(\mathcal{X}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}, \mathbf{w}_{i_{k-1}}), & \mathbf{w}_{i_{k-1}} &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{i_{k-1}}), \\ \mathbf{y}_{i_k} &= \mathbf{g}(\mathcal{X}_{i_k}) + \mathbf{v}_{i_k}, & \mathbf{v}_{i_k} &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{i_k}), \end{aligned} \tag{6.1}$$

for Robot i, where $\mathbf{u}_{i_k} \in \mathbb{R}^{n_u}$ is the process input at time step k, $\mathbf{y}_{i_k} \in \mathbb{R}^{n_y}$ are the measurements, and $\mathcal{X}_{i_k} \in G$ denotes the robot state belonging to any Lie group G. As a notational convenience, the shorthand $\mathcal{X}_{i:j} = \{\mathcal{X}_i \ \dots \ \mathcal{X}_j\}$ will refer to a collection of arbitrary objects with indices in the range $[i, j]$.

6.4.1 Covariance Intersection

Covariance intersection (CI) is a tool introduced by [76] for the purposes of decentralized data fusion under unknown cross-correlations, and can be summarized with the following theorem.

Theorem 6.4.1 (Consistency of Covariance Intersection)**.** The inequality

$$\begin{bmatrix} \frac{1}{w}\boldsymbol{\Sigma}_{xx} & \mathbf{0} \\ \mathbf{0} & \frac{1}{1-w}\boldsymbol{\Sigma}_{yy} \end{bmatrix} \geq \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{xy}^{\mathsf{T}} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}, \tag{6.2}$$

which applies in the positive definite sense, holds for all $w \in (0, 1)$, where $\boldsymbol{\Sigma}_{xx}, \boldsymbol{\Sigma}_{yy}$, and the right-hand-side of (6.2) are positive definite.

6.5 A Toy Problem

Consider first one of the simplest multi-robot estimation problems, shown on the left of Figure 6.1. Two robots are located at positions r_1 and r_2, respectively, and both robots seek to estimate both robots' positions. By design, each robot carries distinct, conceptually independent estimates, even though their states represent the same true *physical* variables. This mimics exactly what will occur in implementation, as each robot's processor will have a live estimate of both robots' positions. Their state vectors can therefore be defined as

$$\mathbf{x}_1 = \begin{bmatrix} r_1^{[1]} & r_2^{[1]} \end{bmatrix}^\mathsf{T}, \qquad \mathbf{x}_2 = \begin{bmatrix} r_1^{[2]} & r_2^{[2]} \end{bmatrix}^\mathsf{T}, \tag{6.3}$$

where the square bracket superscript $(\cdot)^{[i]}$ is used when necessary to denote Robot i's estimate or "instance" of a common physical variable. Each robot also collects local measurements from its sensors. Robot 1 is capable of measuring its own position,

$$y_1 = \mathbf{G}_1\mathbf{x}_1 + v_1, \quad v_1 \sim \mathcal{N}(0, R_1), \quad \mathbf{G}_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \tag{6.4}$$

while Robot 2 is only capable of measuring its position relative to Robot 1,

$$y_2 = \mathbf{G}_2\mathbf{x}_2 + v_2, \quad v_2 \sim \mathcal{N}(0, R_2), \quad \mathbf{G}_2 = \begin{bmatrix} -1 & 1 \end{bmatrix}. \tag{6.5}$$

To keep things simple for this demonstrative problem, robots are assumed to have access to each other's input measurements, such as wheel odometry. This allows them to predict their state forward in time using a conventional Kalman filter. However, a more communication-efficient solution will be proposed in Section 6.7. Neither robot is capable of estimating their full state vector from local measurements only, meaning that some form of communication will be required.

To reflect the knowledge that the two robots' state vectors are physically the same, a key design choice of this chapter is to incorporate a *pseudomeasurement* of the form

$$\mathbf{y}_{12} = \mathbf{x}_1 - \mathbf{x}_2 + \mathbf{v}_{12}, \qquad \mathbf{v}_{12} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Psi}), \tag{6.6}$$

whose "measured" value is always exactly zero. This pseudomeasurement can be viewed as a soft constraint on the problem, inversely weighted by the user-chosen pseudomeasurement covariance $\mathbf{\Psi}$. The estimation problem is now to compute, as accurately as possible, the posterior distribution

$$p(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}). \tag{6.7}$$

6.5.1 Solution via MAP

Applying MAP to this simplified problem is to say that

$$\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 = \arg\max_{\mathbf{x}_1, \mathbf{x}_2} p(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}). \tag{6.8}$$

Assuming that $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_{12}$ are all independent random variables allows the use of Bayes' rule to write

$$p(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}) = \eta p(\mathbf{y}_1 | \mathbf{x}_1) p(\mathbf{y}_2 | \mathbf{x}_2) p(\mathbf{y}_{12} | \mathbf{x}_1, \mathbf{x}_2) p(\mathbf{x}_1, \mathbf{x}_2). \tag{6.9}$$

Next, assume that the prior distributions of the robots are independent and Gaussian, possibly as a result of using CI,

$$p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_1) p(\mathbf{x}_2) = \mathcal{N}(\check{\mathbf{x}}_1, \check{\mathbf{P}}_1) \mathcal{N}(\check{\mathbf{x}}_2, \check{\mathbf{P}}_2). \tag{6.10}$$

Substituting (6.10) into (6.9) and grouping terms into those available to each robot yields

$$p(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}) = \eta p(\mathbf{y}_{12} | \mathbf{x}_1, \mathbf{x}_2) \Big(p(\mathbf{y}_1 | \mathbf{x}_1) p(\mathbf{x}_1) \Big) \Big(p(\mathbf{y}_2 | \mathbf{x}_2) p(\mathbf{x}_2) \Big). \tag{6.11}$$

Since the local measurement models are linear, it is straightforward to exactly compute the terms

$$p(\mathbf{y}_i | \mathbf{x}_i) p(\mathbf{x}_i) = \eta_i p(\mathbf{x}_i | \mathbf{y}_i) = \eta_i \mathcal{N}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{P}}_i), \quad i = 1, 2, \tag{6.12}$$

using the regular Kalman filter equations. The means and covariances $\tilde{\mathbf{x}}_i, \tilde{\mathbf{P}}_i$ (with tildes) represent the distribution of each robot's state conditioned on only local measurements, without the information that the robots' states are physically the same. Substituting (6.12) into (6.11) yields a simplified expression for the posterior, and the optimization problem (6.8) now leads to the least-squares problem

$$\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 = \arg\min_{\mathbf{x}_1, \mathbf{x}_2} \frac{1}{2} \mathbf{e}(\mathbf{x}_1, \mathbf{x}_2)^\mathsf{T} \mathbf{W} \mathbf{e}(\mathbf{x}_1, \mathbf{x}_2), \tag{6.13}$$

where

$$\mathbf{e}(\mathbf{x}_1, \mathbf{x}_2) = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \\ \mathbf{1} & -\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \mathbf{0} \end{bmatrix} \triangleq \mathbf{H}\mathbf{x} - \mathbf{z},$$

$$\mathbf{W} = \operatorname{diag}(\tilde{\mathbf{P}}_1^{-1}, \tilde{\mathbf{P}}_2^{-1}, \boldsymbol{\Psi}^{-1}).$$

In this linear case the unique solution $\hat{\mathbf{x}}$ is given by

$$\hat{\mathbf{x}} = (\mathbf{H}^\mathsf{T} \mathbf{W} \mathbf{H})^{-1} \mathbf{H}^\mathsf{T} \mathbf{W} \mathbf{z}, \tag{6.14}$$

where

$$\mathbf{H}^\top\mathbf{W}\mathbf{H} = \begin{bmatrix} \tilde{\mathbf{P}}_1^{-1} + \boldsymbol{\Psi}^{-1} & -\boldsymbol{\Psi}^{-1} \\ -\boldsymbol{\Psi}^{-1} & \tilde{\mathbf{P}}_2^{-1} + \boldsymbol{\Psi}^{-1} \end{bmatrix}, \tag{6.15}$$

$$\mathbf{H}^\top\mathbf{W}\mathbf{z} = \begin{bmatrix} \tilde{\mathbf{P}}_1^{-1}\tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{P}}_2^{-1}\tilde{\mathbf{x}}_2 \end{bmatrix}. \tag{6.16}$$

Various applications of the Sherman-Morrison-Woodbury (SMW) identities [43] can be used to analytically invert the information matrix $\mathbf{H}^\top\mathbf{W}\mathbf{H}$, as well as solve for the mean $\hat{\mathbf{x}}$ in (6.14). After some manipulation, the result is

$$\begin{aligned} \hat{\mathbf{x}} &\triangleq \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}}_1 + \mathbf{K}_1(\tilde{\mathbf{x}}_2 - \tilde{\mathbf{x}}_1) \\ \tilde{\mathbf{x}}_2 + \mathbf{K}_2(\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2) \end{bmatrix}, \\ \hat{\mathbf{P}} &\triangleq (\mathbf{H}^\top\mathbf{W}\mathbf{H})^{-1} \\ &= \begin{bmatrix} (\mathbf{1} - \mathbf{K}_1)\tilde{\mathbf{P}}_1 & -\mathbf{K}_1\tilde{\mathbf{P}}_2 \\ -\mathbf{K}_2\tilde{\mathbf{P}}_1 & (\mathbf{1} - \mathbf{K}_2)\tilde{\mathbf{P}}_2 \end{bmatrix}, \\ \mathbf{K}_1 &\triangleq \tilde{\mathbf{P}}_1(\boldsymbol{\Psi} + \tilde{\mathbf{P}}_2 + \tilde{\mathbf{P}}_1)^{-1}, \\ \mathbf{K}_2 &\triangleq \tilde{\mathbf{P}}_2(\boldsymbol{\Psi} + \tilde{\mathbf{P}}_2 + \tilde{\mathbf{P}}_1)^{-1}, \end{aligned} \tag{6.17}$$

and hence that $p(\mathbf{x}_1, \mathbf{x}_2|\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}) = \mathcal{N}(\hat{\mathbf{x}}, \hat{\mathbf{P}})$. The final individual estimates are obtained by marginalizing out the other robots' states, which is trivial to do in covariance form by simply extracting the corresponding blocks from $\hat{\mathbf{x}}, \hat{\mathbf{P}}$, yielding

$$p(\mathbf{x}_1|\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}) = \mathcal{N}(\hat{\mathbf{x}}_1, (\mathbf{1} - \mathbf{K}_1)\tilde{\mathbf{P}}_1), \tag{6.18}$$

$$p(\mathbf{x}_2|\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_{12}) = \mathcal{N}(\hat{\mathbf{x}}_2, (\mathbf{1} - \mathbf{K}_2)\tilde{\mathbf{P}}_2). \tag{6.19}$$

Conditioning on the pseudomeasurement $\mathbf{y}_{12}$ has introduced cross-correlation terms in (6.17), which are feasible to keep track of for this two-robot scenario, but introduce substantial complexity for an arbitrary multi-robot scenario. Therefore, this chapter simply employs the CI approximation as required. Figure 6.2 shows the estimation error of each robot as multiple pseudomeasurements are fused in succession. The two robots' estimates not only converge to zero error, but also to a common value, which is the main effect of the pseudomeasurent. In Figure 6.3, 100 Monte-Carlo trials are performed on a simulation of this toy problem, but extended to four robots. The root-mean-squared error (RMSE) and normalized estimation error squared (NEES), calculated as per [44, Ch. 5.4], are plotted through time. The lines marked "Proposed" fuse pseudomeasurements as described, and use CI before each state fusion. The naive solution is identical, but does not utilize covariance intersection before state fusion, thus completely neglects cross-correlations. Although the use of CI does introduce error compared to the centralized solution, it is still vastly better than the naive approach.

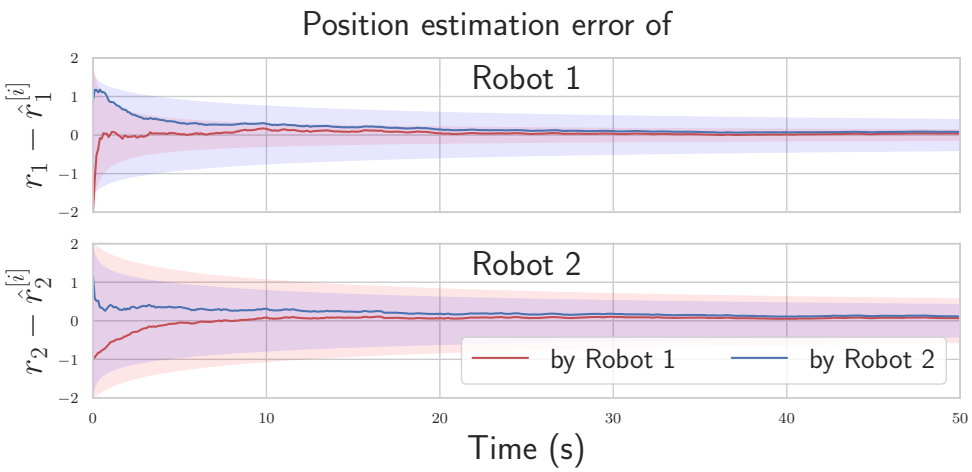

Figure 6.2: Estimation convergence for a single trial of the two-robot toy problem with $\mathbf{\Psi} = 10 \cdot \mathbf{1}$. Due to pseudomeasurements, the robot states successfully converge to a common value.

6.6 General Problem

Consider now N robots, which communicate in correspondance with an arbitrary undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes or robot IDs, and $\mathcal{E}$ is the set of edges. The robots have states $\mathcal{X}_i \in G_i,\ i \in \mathcal{V}$ belonging to possibly different groups. Pseudomeasurement models $\mathbf{c}_{ij} : G_i \times G_j \to \mathbb{R}^c$ are now defined in a generic way

$$\mathbf{y}_{ij_k} = \mathbf{c}_{ij}(\mathcal{X}_{i_k}, \mathcal{X}_{j_k}) + \mathbf{v}_{ij_k}, \qquad \mathbf{v}_{ij_k} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Psi}), \tag{6.20}$$

for each pair $(i, j) \in \mathcal{E}$. Using MAP, the general state fusion problem is now

$$\hat{\mathcal{X}}_{1:N} = \arg\max_{\mathcal{X}_{1:N}} p(\mathcal{X}_{1:N} | \mathbf{y}_{ij}), \quad \text{for all } (i, j) \in \mathcal{E}. \tag{6.21}$$

It is easier to instead consider a single pseudomeasurement at a time, such as, without loss of generality, $\mathbf{y}_{12}$. The posterior distribution given only $\mathbf{y}_{12}$ is

$$p(\mathcal{X}_{1:N} | \mathbf{y}_{12}) = \eta p(\mathbf{y}_{12} | \mathcal{X}_1, \mathcal{X}_2) p(\mathcal{X}_{1:N}) \tag{6.22}$$

$$= \eta\, \mathcal{N}(\mathbf{c}_{12}(\mathcal{X}_1, \mathcal{X}_2), \mathbf{\Psi}) \prod_{i=1}^{N} \mathcal{N}_L(\tilde{\mathcal{X}}_i, \tilde{\mathbf{P}}_i) \tag{6.23}$$

where robot state priors have again been assumed to be independent, as a result of using covariance intersection. Due to this independence, the variables $\mathcal{X}_{3:N}$ can be removed from the optimization problem since their optimal values are simply $\hat{\mathcal{X}}_{3:N} = \tilde{\mathcal{X}}_{3:N}$ and have no effect on $\mathcal{X}_1, \mathcal{X}_2$. Minimizing the negative logarithm of (6.23), omitting terms involving $\mathcal{X}_{3:N}$,

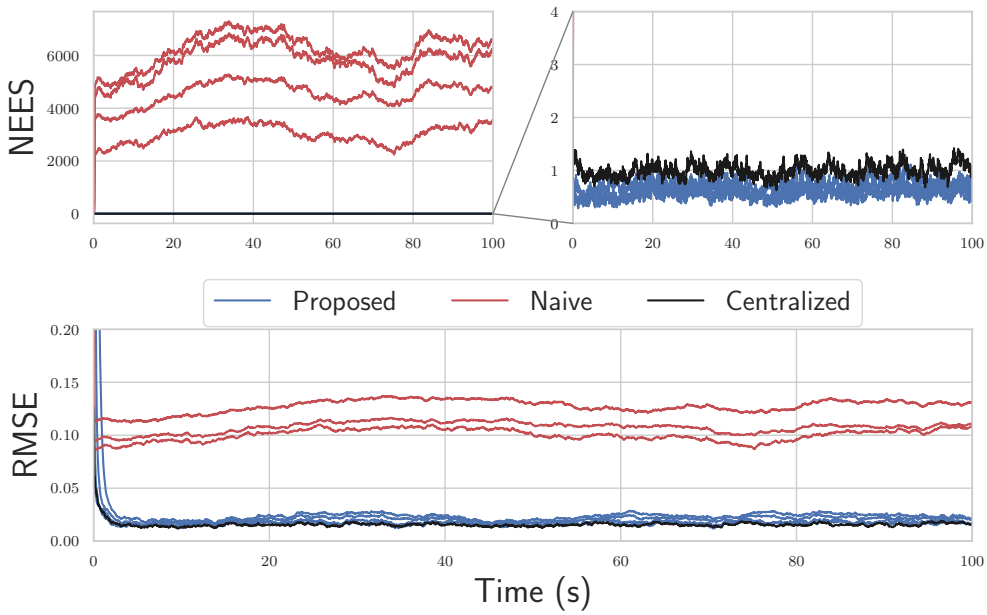

Figure 6.3: Results of 100 Monte Carlo trials for a four-robot version of the toy problem. The top two plots consist of a NEES plot, which is a measure of consistency. The bottom plot is the RMSE of the state. The proposed solution, which performs CI, remains statistically consistent and has reasonbly low error in many cases.

leads to a least-squares problem with error vector given by

$$\mathbf{e}(\mathcal{X}_1, \mathcal{X}_2) = \begin{bmatrix} \mathcal{X}_1 \ominus \tilde{\mathcal{X}}_1 \\ \mathcal{X}_2 \ominus \tilde{\mathcal{X}}_2 \\ -\mathbf{c}_{12}(\mathcal{X}_1, \mathcal{X}_2) \end{bmatrix}, \tag{6.24}$$

and weight $\mathbf{W} = \text{diag}(\tilde{\mathbf{P}}_1^{-1}, \tilde{\mathbf{P}}_2^{-1}, \boldsymbol{\Psi}^{-1})$. Defining $\mathbf{J}_i$ as the group Jacobian associated with G_i, as well as $\mathbf{S}_i, \mathbf{S}_j$ being the jacobians of $\mathbf{c}_{ij}$ with respect to $\mathcal{X}_i, \mathcal{X}_j$, respectively, the Jacobian of the error vector is

$$\mathbf{H} = \begin{bmatrix} \mathbf{J}_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2^{-1} \\ -\mathbf{S}_1 & -\mathbf{S}_2 \end{bmatrix}, \tag{6.25}$$

which is written without arguments $(\mathcal{X}_1, \mathcal{X}_2)$ for brevity. The relevant terms of the Gauss-Newton system are

$$\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H} = \begin{bmatrix} \mathbf{J}_1^{-\mathsf{T}}\mathbf{P}_1^{-1}\mathbf{J}_1^{-1} + \mathbf{S}_1^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{S}_1 & \mathbf{S}_1^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{S}_2 \\ \mathbf{S}_2^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{S}_1 & \mathbf{J}_2^{-\mathsf{T}}\mathbf{P}_2^{-1}\mathbf{J}_2^{-1} + \mathbf{S}_2^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{S}_2 \end{bmatrix},$$

$$\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{e}(\mathcal{X}_1, \mathcal{X}_2) = \begin{bmatrix} \mathbf{J}_1^{-\mathsf{T}}\mathbf{P}_1^{-1}(\mathcal{X}_1 \ominus \tilde{\mathcal{X}}_\infty) - \mathbf{S}_1^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{c}_{12}(\mathcal{X}_1, \mathcal{X}_2) \\ \mathbf{J}_2^{-\mathsf{T}}\mathbf{P}_2^{-1}(\mathcal{X}_2 \ominus \tilde{\mathcal{X}}_\in) - \mathbf{S}_2^\mathsf{T}\boldsymbol{\Psi}^{-1}\mathbf{c}_{12}(\mathcal{X}_1, \mathcal{X}_2) \end{bmatrix},$$

which, by substantial manipulation with the SMW identities, can be used to analytically compute $\delta\hat{\mathbf{x}} = (\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})^{-1}\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{e}(\hat{\mathcal{X}}_1, \hat{\mathcal{X}}_2)$, producing on-manifold iterated-EKF-like expressions. The result is

$$\begin{aligned}
\delta\hat{\mathbf{x}}_1 &= -\mathbf{J}_1(\hat{\mathcal{X}}_1 \ominus \tilde{\mathcal{X}}_1) + \mathbf{K}_1\mathbf{z}, \\
\delta\hat{\mathbf{x}}_2 &= -\mathbf{J}_2(\hat{\mathcal{X}}_2 \ominus \tilde{\mathcal{X}}_2) + \mathbf{K}_2\mathbf{z}, \\
\mathbf{K}_1 &= \mathbf{J}_1\tilde{\mathbf{P}}_1\mathbf{J}_1^\mathsf{T}\mathbf{S}_1^\mathsf{T}\mathbf{V}^{-1}, \\
\mathbf{K}_2 &= \mathbf{J}_2\tilde{\mathbf{P}}_2\mathbf{J}_2^\mathsf{T}\mathbf{S}_2^\mathsf{T}\mathbf{V}^{-1}, \\
\mathbf{z} &= -\mathbf{c}_{12}(\hat{\mathcal{X}}_1, \hat{\mathcal{X}}_2) + \mathbf{S}_1\mathbf{J}_1(\hat{\mathcal{X}}_1 \ominus \tilde{\mathcal{X}}_1) \\
&\quad + \mathbf{S}_2\mathbf{J}_2(\hat{\mathcal{X}}_2 \ominus \tilde{\mathcal{X}}_2), \\
\mathbf{V} &= \mathbf{\Psi} + \mathbf{S}_1\mathbf{J}_1\tilde{\mathbf{P}}_1\mathbf{J}_1^\mathsf{T}\mathbf{S}_1^\mathsf{T} + \mathbf{S}_2\mathbf{J}_2\tilde{\mathbf{P}}_2\mathbf{J}_2^\mathsf{T}\mathbf{S}_2^\mathsf{T},
\end{aligned} \tag{6.26}$$

where iteration is done with $\hat{\mathcal{X}}_i \leftarrow \hat{\mathcal{X}}_i \oplus \delta\hat{\mathbf{x}}_i$ after initialization with $\hat{\mathcal{X}}_i \leftarrow \tilde{\mathcal{X}}_i$. The marginal posterior covariances of Robots 1 and 2 are obtained from the corresponding diagonal blocks of $(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})^{-1}$, and can be shown to be

$$\begin{aligned}
\hat{\mathbf{P}}_1 &= (\mathbf{1} - \mathbf{K}_1\mathbf{S}_1)\mathbf{J}_1\tilde{\mathbf{P}}_1\mathbf{J}_1^\mathsf{T}, \\
\hat{\mathbf{P}}_2 &= (\mathbf{1} - \mathbf{K}_2\mathbf{S}_2)\mathbf{J}_2\tilde{\mathbf{P}}_2\mathbf{J}_2^\mathsf{T}.
\end{aligned} \tag{6.27}$$

The above fusion step introduces cross-correlations between the state estimates of $\mathcal{X}_1$ and $\mathcal{X}_2$, and hence require a covariance intersection step

$$\hat{\mathbf{P}}_1 \leftarrow \frac{1}{w}\hat{\mathbf{P}}_1, \qquad \hat{\mathbf{P}}_2 \leftarrow \frac{1}{1-w}\hat{\mathbf{P}}_2, \qquad w \in (0,1). \tag{6.28}$$

The next step is to make the approximation that $p(\mathcal{X}_{1:N}|\mathbf{y}_{12}) \approx \prod_{i=1}^N \mathcal{N}_L(\hat{\mathcal{X}}_i, \hat{\mathbf{P}}_i)$, and proceed with the fusion of a second pseudomeasurement, using $p(\mathcal{X}_{1:N}|\mathbf{y}_{12}, \mathbf{y}_{13}) = \eta p(\mathbf{y}_{13}|\mathcal{X}_1, \mathcal{X}_3)p(\mathcal{X}_{1:N}|\mathbf{y}_{12})$. This new posterior can again be approximated as Gaussian using the expressions in (6.26) and (6.27), and the process is repeated until all pseudomeasurements are incorporated. Algorithm 2 summarizes the general-purpose decentralized estimation algorithm from the point of view of an arbitrary robot. The algorithm is presented in a callback format, describing how the current state estimate is updated when various events occur. In practice the pseudomeasurement covariance $\mathbf{\Psi}$ can be made very small, such as $\mathbf{\Psi} = 10^{-6} \cdot \mathbf{1}$, to enforce instantaneous matching of state estimates.

Algorithm 2 Decentralized estimation with no input sharing.

For Robot i with process and measurement models given by (6.1), the following points break down how to compute the estimate when various events occur. Robot i's estimate is initialized with $\check{\mathcal{X}}_{i_0}, \check{\mathbf{P}}_{i_0}$. The matrices $\mathbf{F}_{i_k}, \mathbf{L}_{i_k}$ are the Jacobians of $\mathbf{f}$ with respect to $\mathcal{X}_{i_k}$ and $\mathbf{w}_{i_k}$, respectively. The matrix $\mathbf{G}_{i_k}$ is the Jacobian of $\mathbf{g}$, and $\mathbf{S}_{i_k}, \mathbf{S}_{j_k}$ are the Jacobians of $\mathbf{c}_{ij}$ with respect to $\mathcal{X}_{i_k}, \mathcal{X}_{j_k}$, respectively.

- On the reception of an input measurement $\mathbf{u}_{i_{k-1}}$:

$$\check{\mathcal{X}}_{i_k} = \mathbf{f}(\hat{\mathcal{X}}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}, \mathbf{0}), \tag{6.29}$$

$$\check{\mathbf{P}}_{i_k} = \mathbf{F}_{i_{k-1}} \hat{\mathbf{P}}_{i_{k-1}} \mathbf{F}_{i_{k-1}}^\mathsf{T} + \mathbf{L}_{i_{k-1}} \mathbf{Q}_{k-1} \mathbf{L}_{i_{k-1}}^\mathsf{T}. \tag{6.30}$$

- On the reception of a local measurement $\mathbf{y}_{i_k}$:

$$\begin{aligned}
\mathbf{K} &= \hat{\mathbf{P}}_{i_k} \mathbf{G}_{i_k}^\mathsf{T} (\mathbf{G}_{i_k} \hat{\mathbf{P}}_{i_k} \mathbf{G}_{i_k}^\mathsf{T} + \mathbf{R}_{i_k})^{-1}, \\
\delta\tilde{\mathbf{x}} &= \mathbf{K}(\mathbf{y}_{i_k} - \mathbf{g}(\check{\mathcal{X}}_{i_k})), \\
\tilde{\mathcal{X}}_{i_k} &= \check{\mathcal{X}}_{i_k} \oplus \delta\tilde{\mathbf{x}}, \\
\tilde{\mathbf{P}}_{i_k} &= (\mathbf{1} - \mathbf{K}\mathbf{G}_{i_k}) \check{\mathbf{P}}_{i_k}.
\end{aligned}$$

- On the reception of a neighbors' state estimate $\tilde{\mathcal{X}}_{j_k}, \tilde{\mathbf{P}}_{j_k}$:

$$\begin{aligned}
\tilde{\mathbf{P}}_i &\leftarrow \frac{1}{w} \tilde{\mathbf{P}}_i, \qquad \tilde{\mathbf{P}}_j \leftarrow \frac{1}{1-w} \tilde{\mathbf{P}}_j, \\
\mathbf{K} &= \check{\mathbf{P}}_{i_k} \mathbf{S}_{i_k}^\mathsf{T} (\boldsymbol{\Psi} + \mathbf{S}_{i_k} \tilde{\mathbf{P}}_{i_k}^\mathsf{T} \mathbf{S}_{i_k}^\mathsf{T} + \mathbf{S}_{j_k} \tilde{\mathbf{P}}_{j_k}^\mathsf{T} \mathbf{S}_{j_k}^\mathsf{T})^{-1}, \\
\delta\hat{\mathbf{x}} &= -\mathbf{K}(\mathbf{c}_{ij}(\tilde{\mathcal{X}}_{i_k}, \tilde{\mathcal{X}}_{j_k})), \\
\hat{\mathcal{X}}_{i_k} &= \tilde{\mathcal{X}}_{i_k} \oplus \delta\hat{\mathbf{x}}, \\
\hat{\mathbf{P}}_{i_k} &= (\mathbf{1} - \mathbf{K}\mathbf{G}_{i_k}) \tilde{\mathbf{P}}_{i_k},
\end{aligned}$$

  and optionally further iterate *both* robot estimates with (6.26), (6.27).
- At any time, send the current state estimate $\tilde{\mathcal{X}}_{j_k}, \tilde{\mathbf{P}}_{j_k}$ to neighbors.

6.6.1 An Observability Test

In a decentralized state estimation context, the full system state is the concatenation of all the robots' states. Hence, observability of this system refers to the ability to recover the state trajectory of *each* robot given the inputs and measurements of *all* robots. An observability test for decentralized estimators should therefore take into account all the local measurements collected by each robot *and* what information is communicated between them. Consider again the simple toy problem described in Section 6.5, but with neither robot collecting absolute position measurements, thus making both robot's states unobservable. Performing a standard observability test for an individual robot, treating the other robot's estimate as a "measurement", will falsely conclude that the system is observable. On the other hand, the upcoming observability test will correctly indicate that there is an unobservable degree of

freedom in the joint system state, through rank deficiency of an observability matrix.

The pseudomeasurements encode the communication structure between the robots. An advantage of this proposed approach is that it allows the straightforward application of common observability tests to the entire team of robots, viewed as a single system. A *local* observability test can be formed by considering the MAP problem on an entire trajectory simultaneously, but without prior information on the initial state ([41]). Let the bolded $\boldsymbol{\mathcal{X}}_k = (\mathcal{X}_{1_k}, \ldots, \mathcal{X}_{N_k})$ denote the state of all robots at time step k, $\mathbf{y}_k = [\mathbf{y}_{1_k}^\mathsf{T} \ \ldots \ \mathbf{y}_{N_k}^\mathsf{T}]^\mathsf{T}$ denote a stacked vector containing all the robots' local measurements at time step k. Let $\boldsymbol{\psi}_k = [\ \ldots\ \mathbf{y}_{ij}^\mathsf{T}\ \ldots\ ]^\mathsf{T}$, $(i,j) \in \mathcal{E}$ denote the stacked pseudomeasurements between all robots at time step k. The MAP problem is

$$\hat{\boldsymbol{\mathcal{X}}}_{0:K} = \arg\max_{\boldsymbol{\mathcal{X}}_{0:K}} p(\boldsymbol{\mathcal{X}}_{0:K} | \mathbf{y}_{0:K}, \boldsymbol{\psi}_{0:K}) \tag{6.31}$$

with

$$p(\boldsymbol{\mathcal{X}}_{0:K} | \mathbf{y}_{0:K}, \boldsymbol{\psi}_{0:K}) = \eta \prod_{k=0}^{K} p(\mathbf{y}_0 | \boldsymbol{\mathcal{X}}_0) p(\boldsymbol{\psi}_0 | \boldsymbol{\mathcal{X}}_0) \prod_{k=1}^{K} p(\boldsymbol{\mathcal{X}}_k | \boldsymbol{\mathcal{X}}_{k-1}),$$

leading to a nonlinear least squares problem with weight $\mathbf{W}$ and error vector

$$\begin{aligned}
\mathbf{e}(\boldsymbol{\mathcal{X}}_{0:K}) &= [\ \ldots\ \mathbf{e}_{u,k}\ \ldots\ \mathbf{e}_{y,k}\ \ldots\ \mathbf{e}_{\psi,k}\ \ldots]^\mathsf{T}, \\
\mathbf{e}_{u,k} &= [\ldots (\mathcal{X}_{i_k} \ominus \mathbf{f}(\mathcal{X}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}))^\mathsf{T} \ldots], \\
\mathbf{e}_{y,k} &= [\ \ldots\ (\mathbf{y}_{i_k} - \mathbf{g}(\mathcal{X}_{i_k}))^\mathsf{T}\ \ldots\ ], \quad i = 1, \ldots, N, \\
\mathbf{e}_{\psi,k} &= [\ \ldots\ -\mathbf{c}_{ij}(\mathcal{X}_{i_k}, \mathcal{X}_{j_k})^\mathsf{T}\ \ldots\ ], \quad (i,j) \in \mathcal{E}.
\end{aligned}$$

The error Jacobian is

$$\mathbf{H} = \begin{bmatrix}
-\mathbf{F}_0 & \mathbf{1} & & & \\
& \ddots & \ddots & & \\
& & -\mathbf{F}_{K-1} & \mathbf{1} & \\
-\mathbf{G}_0 & & & & \\
& -\mathbf{G}_1 & & & \\
& & \ddots & & \\
& & & -\mathbf{G}_K & \\
-\mathbf{\Phi}_0 & & & & \\
& -\mathbf{\Phi}_1 & & & \\
& & \ddots & & \\
& & & -\mathbf{\Phi}_K &
\end{bmatrix}, \tag{6.32}$$

$$\mathbf{F}_k = \text{diag}\left(\ldots, \frac{D\mathbf{f}(\mathcal{X}_{i_k}, \mathbf{u}_{i_k})}{D\mathcal{X}_{i_k}}, \ldots\right), \tag{6.33}$$

$$\mathbf{G}_k = \text{diag}\left(\ldots, \frac{D\mathbf{g}(\mathcal{X}_{i_k})}{D\mathcal{X}_{i_k}}, \ldots\right), \quad i = 1, \ldots, N, \tag{6.34}$$

$$\mathbf{\Phi}_k = -\frac{D\mathbf{e}_{\psi,k}(\boldsymbol{\mathcal{X}}_k)}{D\boldsymbol{\mathcal{X}}_k}, \tag{6.35}$$

with all undisplayed entries in $\mathbf{H}$ equal to zero. For the solution to the MAP problem to be unique, then to first order, $(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H})$ must be invertible, and thus full rank. Fortunately, $\mathbf{W}$ is always positive definite regardless of any cross-correlations that would add off-diagonal entries. Hence, $\text{rank}(\mathbf{H}^\mathsf{T}\mathbf{W}\mathbf{H}) = \text{rank}(\mathbf{H})$, and it is thus required that $\mathbf{H}$ be full column rank. This implies that the proposed observability test is unaffected by the approximation induced by CI, or any cross-correlation terms that may or may not be successfully tracked. In a similar way to [36, Ch 3.1.4], it can be shown that by performing a variety of elementary row/column operations, the rank of $\mathbf{H}$ is equivalent to the rank of

$$\mathcal{O} = \begin{bmatrix} \mathbf{M}_0 \\ \mathbf{M}_1\mathbf{F}_0 \\ \vdots \\ \mathbf{M}_K\mathbf{F}_{K-1}\ldots\mathbf{F}_0 \end{bmatrix}, \qquad \mathbf{M}_k = \begin{bmatrix} \mathbf{G}_k \\ \mathbf{\Phi}_k \end{bmatrix}. \tag{6.36}$$

Hence, if $\mathcal{O}$ has maximum rank, the solution to the MAP problem is locally unique, and the system is said to be observable. Note that this test easily allows for time-varying graphs, which would yield a different $\mathbf{\Phi}_k$ for each time step k.

6.7 Efficient Odometry Sharing using Preintegration

Many problems, especially those where robots estimate their neighbors' positions, will require robots to have access to their neighbors' process-model input values $\mathbf{u}$. Until now, it has been assumed that all robots have unrestricted access to each other's inputs. In robot state estimation applications, the input is often the odometry measurements, such as wheel encoder or IMU measurements. These can occur at frequencies of 100 - 1000 Hz, and can therefore be infeasible to share in real time, especially if multiple robots are to simultaneously share measurements at high frequency. This could quickly reach a bandwidth limit on the common communication channel, such as ultra-wideband radio.

The proposed solution to this problem is to use *preintegration*. That is, robots will instead share preintegrated input measurements over an arbitrary duration of time instead of

individual input measurements. Specifically, consider the following generic process model

$$\mathcal{X}_k = \mathbf{f}(\mathcal{X}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}). \tag{6.37}$$

The action of preintegration is to directly iterate this process model by repeated compositions in order to, after algebraic manipulation, generate a new *preintegrated process model* $\mathbf{f}_{pq}$ that relates two states at arbitrary time steps $k = p$ and $k = q$. That is,

$$\mathcal{X}_{p+1} = \mathbf{f}(\mathcal{X}_p, \mathbf{u}_p, \mathbf{w}_p), \tag{6.38}$$

$$\mathcal{X}_{p+2} = \mathbf{f}(\mathbf{f}(\mathcal{X}_p, \mathbf{u}_p, \mathbf{w}_p), \mathbf{u}_{p+1}, \mathbf{w}_{p+1}), \tag{6.39}$$

$$\vdots \tag{6.40}$$

$$\mathcal{X}_q = \mathbf{f}(\mathbf{f}(\ \dots\ \mathbf{f}(\mathcal{X}_p, \mathbf{u}_p, \mathbf{w}_p)\ \dots\ ), \mathbf{u}_{q-1}, \mathbf{w}_{q-1}) \tag{6.41}$$

$$\triangleq \mathbf{f}_{pq}(\mathcal{X}_i, \Delta\mathcal{X}_{pq}) \oplus \mathbf{w}_{pq}, \tag{6.42}$$

where $\Delta\mathcal{X}_{pq}$ is the *relative motion increment* (RMI), which in general may also belong to a Lie group, and $\mathbf{w}_{pq} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{pq})$ is the *preintegrated noise*. The advantages of preintegration will stem from the careful choice of RMI definition, which is ideally done such that the RMI has the following properties.

1. The RMI is determined from the input measurements exclusively, and is independent of the state estimate:

$$\Delta\mathcal{X}_{pq} = \Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}). \tag{6.43}$$

2. Far fewer numbers are required to represent the RMI than the $(q-p)$ raw measurements that occurred during the preintegration interval:

$$\dim(\Delta\mathcal{X}_{pq}) << (q-p)\dim(\mathbf{u}_k). \tag{6.44}$$

If the above points are true, communicating $\Delta\mathcal{X}_{pq}, \mathbf{Q}_{pq}$ instead of $\mathbf{u}_{p:q-1}$ will not only reduce the communication cost, but will also result in a fixed message size and ability to directly predict the state forward over a long duration of time, instead of sequentially processing the measurements. It turns out that many common process models in robotics are amenable to preintegration, and furthermore are typically extremely fast to preintegrate incrementally as input measurements are obtained. That is, there exists a function $\textsc{Increment}(\cdot)$ such that

$$\Delta\mathcal{X}_{pq}, \mathbf{Q}_{pq} = \textsc{Increment}(\Delta\mathcal{X}_{p:q-1}, \mathbf{Q}_{p:q-1}, \mathbf{u}_{q-1}).$$

A few examples now follow, which describe concrete implementations of $\Delta\mathcal{X}_{pq}$, $\mathbf{f}_{pq}(\cdot)$, and $\textsc{Increment}(\cdot)$.

Example 6.7.1 (Linear preintegration)**.** The linear process model

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{L}_{k-1}\mathbf{u}_{k-1} \tag{6.45}$$

can be directly iterated to yield

$$\mathbf{x}_q = \left(\prod_{k=p}^{q-1} \mathbf{F}_k\right) \mathbf{x}_p + \sum_{k=p}^{q-1} \left(\prod_{\ell=k+1}^{q-1} \mathbf{F}_\ell\right) \mathbf{L}_k \mathbf{u}_k \tag{6.46}$$

$$\triangleq \mathbf{F}_{pq} \mathbf{x}_p + \Delta \mathbf{x}_{pq}, \tag{6.47}$$

where (6.47) defined $\mathbf{f}(\cdot)$. Assuming that noise enters the model additively through the input $\mathbf{u}_k = \bar{\mathbf{u}}_k + \mathbf{w}_k$, where $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, (6.47) becomes $\mathbf{x}_j = \mathbf{F}_{pq} \mathbf{x}_i + \Delta \bar{\mathbf{x}}_{pq} + \mathbf{w}_{pq}$ where

$$\mathbf{w}_{pq} \triangleq \sum_{k=p}^{q-1} \left(\prod_{\ell=k+1}^{q-1} \mathbf{F}_\ell\right) \mathbf{L}_k \mathbf{w}_k, \tag{6.48}$$

$$= \mathbf{F}_{q-1} \mathbf{w}_{pq-1} + \mathbf{L}_{q-1} \mathbf{w}_{q-1}. \tag{6.49}$$

The RMI $\Delta \mathbf{x}_{pq}$ and corresponding covariance are therefore built incrementally with

$$\Delta \mathbf{x}_{pq} = \mathbf{F}_{q-1} \Delta \mathbf{x}_{pq-1} + \mathbf{L}_{q-1} \mathbf{u}_{q-1}, \tag{6.50}$$

$$\mathbf{Q}_{pq} = \mathbf{F}_{q-1} \mathbf{Q}_{pq-1} \mathbf{F}_{q-1}^\mathsf{T} + \mathbf{L}_{q-1} \mathbf{Q}_{q-1} \mathbf{L}_{q-1}^\mathsf{T}, \tag{6.51}$$

which together define the INCREMENT$(\cdot)$ function.

Example 6.7.2 (Wheel odometry preintegration on $SE(2)$)**.** Given a robot pose $\mathbf{T} \in SE(2)$, the wheel odometry process model is given by

$$\mathbf{T}_k = \mathbf{T}_{k-1} \mathrm{Exp}(\Delta t \mathbf{u}_{k-1}), \tag{6.52}$$

where $\mathbf{u} = [\omega \; v \; 0]^\mathsf{T}$, ω is the robot's heading rate-of-change, and v is its forward velocity in its own body frame. Direct iteration yields the preintegrated process model $\mathbf{f}_{pq}(\cdot)$ given by

$$\mathbf{T}_q = \mathbf{T}_p \underbrace{\prod_{k=p}^{q-1} \mathrm{Exp}(\Delta t \mathbf{u}_k)}_{\triangleq \Delta \mathbf{T}_{pq}}. \tag{6.53}$$

Noise $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ is again assummed to enter additively through the input, and a series

of first-order approximations lead to

$$\begin{aligned}\Delta\mathbf{T}_{pq} &= \prod_{k=p}^{q-1} \mathrm{Exp}(\Delta t(\bar{\mathbf{u}}_k + \mathbf{w}_k)) \\ &\approx \prod_{k=p}^{q-1} \mathrm{Exp}(\Delta t\bar{\mathbf{u}}_k)\mathrm{Exp}(\Delta t\mathbf{J}_k\mathbf{w}_k) \\ &\approx \Delta\bar{\mathbf{T}}_{pq} \underbrace{\prod_{k=p}^{q-1} \mathrm{Exp}(\Delta t\mathbf{Ad}(\Delta\mathbf{T}_{k+1j}^{-1})\mathbf{J}_k\mathbf{w}_k)}_{\mathrm{Exp}(\mathbf{w}_{pq})},\end{aligned}$$

where $\mathbf{J}_k \triangleq \mathbf{J}(\Delta t\bar{\mathbf{u}}_k)$ is the right Jacobian of $SE(2)$. Having identified an expression for $\mathrm{Exp}(\mathbf{w}_{pq})$, under the assumption that $\mathbf{w}_{pq}$ is small,

$$\begin{aligned}\mathbf{w}_{pq} &\approx \sum_{k=p}^{q-1} \Delta t\mathbf{Ad}(\Delta\mathbf{T}_{k+1q}^{-1})\mathbf{J}_k\mathbf{w}_k \\ &= \sum_{k=p}^{q-2} \Delta t\mathbf{Ad}(\Delta\mathbf{T}_{k+1q}^{-1})\mathbf{J}_k\mathbf{w}_k \\ &\qquad + \Delta t \underbrace{\mathbf{Ad}(\Delta\mathbf{T}_{qq}^{-1})}_{\mathbf{1}}\mathbf{J}_{q-1}\mathbf{w}_{q-1} \\ &= \underbrace{\mathbf{Ad}(\Delta\mathbf{T}_{q-1q}^{-1})}_{\triangleq\mathbf{F}_{q-1}}\mathbf{w}_{pq-1} + \underbrace{\Delta t\mathbf{J}_{q-1}}_{\triangleq\mathbf{L}_{q-1}}\mathbf{w}_{q-1},\end{aligned}$$

and the defining operations of the INCREMENT$(\cdot)$ function follow,

$$\Delta\mathbf{T}_{pq} = \Delta\mathbf{T}_{pq-1}\mathrm{Exp}(\Delta t\mathbf{u}_k), \tag{6.54}$$

$$\mathbf{Q}_{pq} = \mathbf{F}_{q-1}\mathbf{Q}_{pq-1}\mathbf{F}_{q-1}^{\mathsf{T}} + \mathbf{L}_{q-1}\mathbf{Q}_{q-1}\mathbf{L}_{q-1}^{\mathsf{T}}. \tag{6.55}$$

Example 6.7.3 (IMU preintegration)**.** Being the most well-known usage of preintegration, a complete reference for IMU preintegration on the $SO(3) \times \mathbb{R}^3 \times \mathbb{R}^3$ manifold can be obtained from [50], and alternatively for the $SE_2(3)$ group from [36, 51]. Either approach can be used with the framework in this chapter. However in Section 6.9 of this chapter, $\mathbf{T}_{wi} \in SE_2(3)$ matrices are used to represent the extended pose of Robot i relative to an inertial world frame w. Following [81] and [51], it can be shown that the discrete-time IMU kinematic equations can be written in the form

$$\mathbf{T}_{wi_k} = \mathbf{G}_{k-1}\mathbf{T}_{wi_{k-1}}\mathbf{U}_{k-1}, \tag{6.56}$$

where

$$\mathbf{T}_{wi_k} = \begin{bmatrix} \mathbf{C} & \mathbf{v} & \mathbf{r} \\ \mathbf{0} & 1 & 0 \\ \mathbf{0} & 0 & 1 \end{bmatrix}$$

$$\mathbf{G}_{k-1} = \begin{bmatrix} \mathbf{1} & \Delta t\mathbf{g} & -\frac{\Delta t^2}{2}\mathbf{g} \\ 0 & 1 & -\Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{U}_{k-1} = \begin{bmatrix} \exp(\Delta t\boldsymbol{\omega}^\wedge) & \Delta t\mathbf{J}(\Delta t\boldsymbol{\omega})\mathbf{a} & \frac{\Delta t^2}{2}\mathbf{N}(\Delta t\boldsymbol{\omega})\mathbf{a} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{N}(\boldsymbol{\phi}) = \mathbf{z}\mathbf{z}^\mathsf{T} + 2\left(\frac{1}{\phi} - \frac{\sin\phi}{\phi^2}\right)\mathbf{z}^\wedge + 2\frac{\cos\phi - 1}{\phi^2}\mathbf{z}^\wedge\mathbf{z}^\wedge$$

$$\phi = \|\boldsymbol{\phi}\|, \ \mathbf{z} = \boldsymbol{\phi}/\phi,$$

and $\mathbf{C} \in SO(3)$, $\mathbf{v}$, $\mathbf{r}$ respectively represent attitude, velocity, position relative to frame w, $\boldsymbol{\omega}$ is the IMU's unbiased gyro measurement, $\mathbf{a}$ is the IMU's unbiased accelerometer measurement, $\mathbf{g}$ is the gravity vector resolved in frame w, and $\mathbf{J}(\boldsymbol{\phi})$ is the left Jacobian of $SO(3)$. Preintegration of these kinematics is easily achieved by direct iteration with

$$\mathbf{T}_{wi_q} = \left(\prod_{k=p}^{q-1}\mathbf{G}_{k-1}\right)\mathbf{T}_{wi_p}\left(\prod_{k=p}^{q-1}\mathbf{U}_{k-1}\right) \tag{6.57}$$

$$\triangleq \Delta\mathbf{G}_{pq}\mathbf{T}_{wi_p}\Delta\mathbf{U}_{pq}, \tag{6.58}$$

where $\Delta\mathbf{U}_{pq}$ is the RMI, and the noise statistics can be propagated through this preintegration process by a standard linearization procedure. [81] present the details of the noise propagation, as well as how to adapt this formulation for relative poses.

6.7.1 Multi-robot preintegration

In the context of multi-robot estimation problems, an individual robot's process model may involve the input values of many neighboring robots. To reflect this, rewrite the process model for Robot i as

$$\mathcal{X}_{i_k} = \mathbf{f}(\mathcal{X}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}, \mathbf{u}_{j_{k-1}}), \qquad j \in \mathcal{N}_i, \tag{6.59}$$

where $\mathbf{u}_{i_k}$ denotes an input measured by Robot i and $\mathcal{N}_i$ denotes the set of neighbor IDs of Robot i. The preintegrated process model would now be written as

$$\mathcal{X}_{i_q} = \mathbf{f}_{pq}(\mathcal{X}_{i_p}, \Delta\mathcal{X}_{i_{pq}}, \Delta\mathcal{X}_{j_{pq}}), \qquad j \in \mathcal{N}_i, \tag{6.60}$$

where $\Delta\mathcal{X}_{i_{pq}}$ denotes an RMI calculated from the input measurements of Robot i.

A complication is that the RMIs from neighboring Robots $\Delta\mathcal{X}_{j_{pq}}$ are only available asynchronously, meaning it is not always possible to evaluate (6.60) directly. To deal with this, assume that the preintegrated process model $\mathbf{f}_{pq}$ is compatible with

$$\mathfrak{X}_{i_{k-1}} = \mathbf{f}(\mathcal{X}_{i_{k-2}}, \mathbf{u}_{i_{k-2}}, \mathbf{0}), \tag{6.61}$$

$$\mathfrak{X}_{i_k} = \mathbf{f}(\mathfrak{X}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}, \mathbf{0}), \tag{6.62}$$

$$\mathcal{X}_{i_k} = \mathbf{f}_{pq}(\mathfrak{X}_{i_k}, \mathcal{I}, \Delta\mathcal{X}_{j_{pq}}), \tag{6.63}$$

where $\mathcal{I}$ is the identity element. The variable $\mathfrak{X}_{i_k}$ represents an intermediate, non-physical state that is continuously propagated using the process model without input information from neighboring robots. Sometime later, at arbitrary time step $k = q$, the RMI from a neighboring robot $\Delta\mathcal{X}_{j_{pq}}$ is received and this intermediate state $\mathfrak{X}_{i_q}$ is propagated back into a physically meaningful quantity $\mathcal{X}_{i_q}$. A concrete example of this asynchronous intermediate state updating is shown in Section 6.8, and a summary is shown in Algorithm 3.

Algorithm 3 Decentralized estimation with preintegration.

The setup is identical to Algorithm 2, except that the process model requires input information from neighboring robots as in (6.59), (6.60).

- On the reception of a local input measurement $\mathbf{u}_{i_{k-1}}$:

$$\begin{aligned}\mathfrak{X}_{i_k} &= \mathbf{f}(\hat{\mathcal{X}}_{i_{k-1}}, \mathbf{u}_{i_{k-1}}, \mathbf{0}) \\ \mathfrak{P}_{i_k} &= \mathbf{F}_{i_{k-1}}\hat{\mathbf{P}}_{i_{k-1}}\mathbf{F}_{i_{k-1}}^{\mathsf{T}} + \mathbf{L}_{i_{k-1}}\mathbf{Q}_{k-1}\mathbf{L}_{i_{k-1}}^{\mathsf{T}}.,\end{aligned}$$

  and increment Robot i's own RMI,

$$\Delta\mathcal{X}_{i_{pk}}, \mathbf{Q}_{i_{pk}} = \textsc{Increment}(\Delta\mathcal{X}_{i_{pk-1}}, \mathbf{Q}_{i_{pk-1}}, \mathbf{u}_{i_{k-1}}).$$

- Whenever required, send $\Delta\mathcal{X}_{i_{pq}}, \mathbf{Q}_{i_{pq}}$ to neighbors.
- On the reception of a neighboring robot's RMI and covariance $\Delta\mathcal{X}_{j_{pq}}, \mathbf{Q}_{j_{pq}}$,

$$\begin{aligned}\mathcal{X}_{i_q} &= \mathbf{f}_{pq}(\mathfrak{X}_{i_q}, \mathcal{I}, \Delta\mathcal{X}_{j_{pq}}), \\ \mathbf{F}_{pq} &= \left.\frac{D\mathbf{f}_{pq}(\mathfrak{X}, \mathcal{I}, \Delta\mathcal{X}_{j_{pq}})}{D\mathfrak{X}}\right|_{\mathfrak{X}_{i_q}}, \\ \check{\mathbf{P}}_{i_q} &= \mathbf{F}_{pq}\mathfrak{P}_{i_{k-1}}\mathbf{F}_{pq}^{\mathsf{T}} + \mathbf{Q}_{j_{pq}}.\end{aligned}$$

- On the reception of a local measurement $\mathbf{y}_{i_k}$, proceed as per Algorithm 2.
- On the reception of a neighbors' state estimate $\tilde{\mathcal{X}}_j, \tilde{\mathbf{P}}_j$, proceed as per Algorithm 2.
- At any time, send the current state estimate $\tilde{\mathcal{X}}_{j_k}, \tilde{\mathbf{P}}_{j_k}$ to neighbors.

6.7.2 Estimating Input Biases

For some problems, it may be desired to estimate an input bias $\mathbf{b}$ as part of the overall state $\mathcal{X}$, a setup commonly occuring in inertial navigation where accelerometer and rate gyro biases are estimated. The difficulty lies in the frequent inability to express RMIs independently of the bias values, thus leaving RMIs in the form of

$$\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{b}_{p:q-1}). \tag{6.64}$$

In the context of the multi-robot estimation scheme shown in Algorithm 3, computing RMIs this way causes inconsistency in the filter, since the RMIs are now correlated with the robot states. Accounting for this would require maintaining the cross-correlation between a robot's state and their neighbors' biases.

A simpler alternate solution is to have robots estimate their neighbors' input biases in addition to their own. This requires to exploit the fact that biases are usually modelled to follow a random walk, and therefore have a constant mean in the absence of any correcting information. This motivates the approximation $\mathbf{b}_p \approx \mathbf{b}_{p+1} \approx \ldots \approx \mathbf{b}_q$ and hence

$$\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{b}_{p:q-1}) \approx \Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{b}_q). \tag{6.65}$$

When robots receive input measurements, they increment their RMIs with raw (biased) inputs to produce $\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{0})$. At an appropriate time, they share their current biased RMIs, which is corrected for bias by the receiving robot using the first-order approximation

$$\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{b}_q) \approx \Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{0}) \oplus \mathbf{B}_{pq}\mathbf{b}_q, \tag{6.66}$$

$$\mathbf{B}_{pq} \triangleq \frac{D}{D\mathbf{b}_q}\Big(\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1}, \mathbf{b}_q)\Big), \tag{6.67}$$

where $\mathbf{B}_{pq}$ is defined as the bias jacobian. Equation (6.66) is an approximation that contributes unmodelled errors to the estimation problem, relying on an assumption that $\mathbf{b}_q$ is small. This can be enabled by a proper offline calibration procedure that removes any large bias, and only small deviations from this are estimated online. Such a procedure is used for the quadcopter problem in Section 6.9, where good, consistent estimation results are still obtained despite the approximation in (6.66).

6.7.3 Autoencoding Covariance Matrices

As shown in Algorithm 3, predicting state estimates that are a function of neighbor inputs requires the RMI $\Delta\mathcal{X}_{pq}$ along with a corresponding covariance $\mathbf{Q}_{pq}$. As is, these two quantities must be shared between robots. This section proposes an optional method that further reduces communication costs by eliminating the requirement to share the preintegrated

covariance $\mathbf{Q}_{pq}$. This method is applicable to any preintegrated process model, although it was only employed for IMU preintegration in this work.

The key insight is that $\Delta\mathcal{X}_{pq}(\mathbf{u}_{p:q-1})$ and $\mathbf{Q}_{pq}(\mathbf{u}_{p:q-1})$ are both calculated from the same input values $\mathbf{u}_{p:q-1}$. Hence, if an alternate mapping $\mathbf{Q}_{pq} = \mathbf{h}(\Delta\mathcal{X}_{pq})$ existed, then it would be sufficient to share $\Delta\mathcal{X}_{pq}$ only, and the receiving robot could infer $\mathbf{Q}_{pq}$ directly from the RMI. In the absence of analytic expressions for $\mathbf{h}(\cdot)$, this chapter approximates the function with a neural network, trained on purely synthetic RMI-covariance pairs. An additional complication is that such a function $\mathbf{h}(\cdot)$ may not always exist since an RMI can correspond to many possible covariances depending on the input values. In this case $\mathbf{h}(\cdot)$ is not a true function since it is one-to-many, and its definition is modified to also accept a low-dimensional *encoding* $\boldsymbol{e}(\mathbf{Q}_{pq})$, leading to $\mathbf{Q}_{pq} = \mathbf{h}(\Delta\mathcal{X}_{pq}, \boldsymbol{e}(\mathbf{Q}_{pq}))$. This leads to an architecture here referred to as *mean-assisted autoencoding*, depicted in Figure 6.4. The flattened lower-triangular half of $\mathbf{Q}_{pq}$ is given to a simple fully-connected *encoder* network with GELU activation functions and a single hidden layer. The output of this network is the encoding, which can be as small as one or two numbers. This encoding is then concatenated with a parameterization of the RMI $\Delta\mathcal{X}_{pq}$ and fed to a similar *decoder* network. The decoder network outputs the flattened lower-triangular half of a reconstructed covariance matrix denoted $\hat{\mathbf{Q}}_{pq}$. For training, the loss function simply uses the Frobenius norm,

$$\mathcal{L}(\mathbf{Q}_{pq}, \hat{\mathbf{Q}}_{pq}) = ||\mathbf{Q}_{pq} - \hat{\mathbf{Q}}_{pq}||_{\mathsf{F}}. \tag{6.68}$$

Figure 6.5 shows the training convergence history for various encoding sizes, applied to IMU preintegration. The Adam optimizer is chosen with an initial learning rate of 10^{-3} that is scheduled to decrease once the loss plateaus. The encoder/decoder networks have hidden layers with 256 neurons, and the training process takes just a few minutes on a laptop CPU to achieve less than 1% average reconstruction error on a validation dataset from experimental

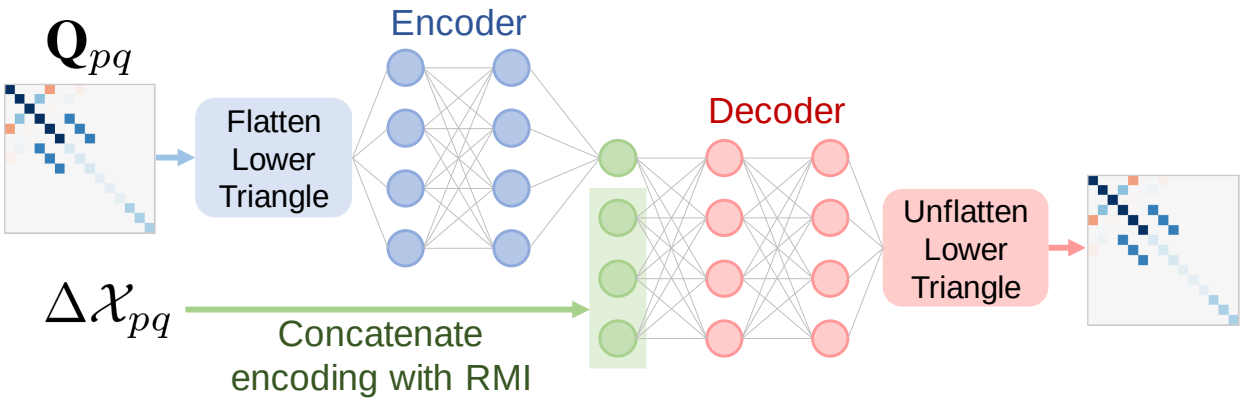

Figure 6.4: Concept diagram of mean-assisted autoencoder.

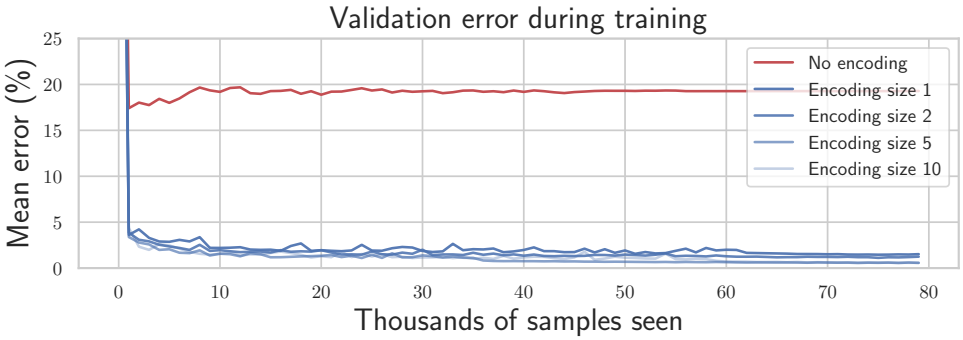

Figure 6.5: Mean percentage reconstruction error throughout training for various encoding sizes including no encoding. A single encoding number is sufficient to achieve less than 1% reconstruction error on average.

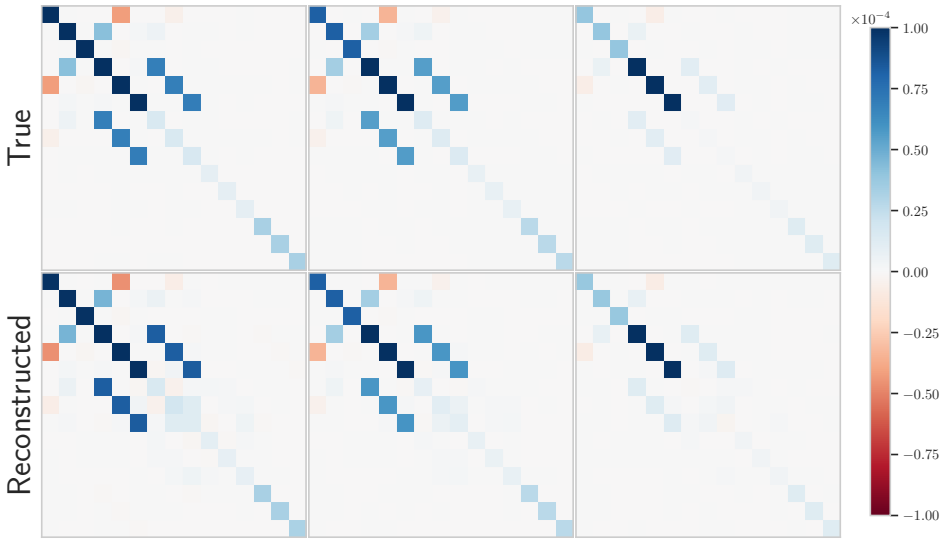

Figure 6.6: Visualization of preintegrated IMU noise covariance matrices along with reconstruction using mean-assisted autoencoding.

data. The training data is purely synthetic, where RMIs are constructed from a random amount of random IMU measurements, with values covering the realistic range of real IMU measurements. Since the length of the dataset is infinite, the risk of overfitting is completely eliminated, and the autoencoder can immediately generalize to different real physical IMUs. A visualization of the reconstructed covariance matrices can be seen in Figure 6.5, using an encoding size of only 1 number. Section 6.9 will employ this method "in the loop" for a real quadcopter problem. It will be shown that the reconstruction error is so small that the impact on the estimation results are negligible.

Concretely, using IMU preintegration as an example, the RMI itself must be communicated, which requires 10 floating-point numbers. However, the covariance matrix is 15×15, which would require communication of an additional 120 floating-point numbers to represent one of its triangular halves. With the proposed autoencoder, these 120 numbers are replaced with an encoding consisting of *one* number, thus dramatically reducing the communication cost.

6.8 Simulation with Ground Robots

The proposed algorithm is tested in a simulation with ground robots, shown in the center of Figure 6.1. Each robot estimates their own pose and their neighbors' poses relative to a world frame. Denoting the pose of Robot i relative to the world frame w as $\mathbf{T}_{wi} \in SE(2)$, the state of an arbitrary robot is given by

$$\mathcal{X}_{i_k} = \left(\mathbf{T}^{[i]}_{wi_k}, \mathbf{T}^{[i]}_{wj_k}, \dots\right), \qquad j \in \mathcal{N}_i, \tag{6.69}$$

where, again, the $(\cdot)^{[i]}$ superscript indicates Robot i's estimate or "instance" of that physical quantity. Each robot collects wheel odometry at 100 Hz, providing $\mathbf{u}_{i_k} = [\omega_{i_k}\ v_{i_k}\ 0]^\mathsf{T}$ as input measurements, where ω_{i_k} is Robot i's angular velocity and v_{i_k} is its forward velocity in its own body frame. The pose kinematics for any single robot along with its preintegration are shown in Example 6.7.2. When Robot i receives an input measurement, it updates the part of its state corresponding to its own pose to create

$$\mathfrak{X}_{i_k} = \left(\mathbf{T}^{[i]}_{iw_{k-1}}\mathrm{Exp}(\Delta t \mathbf{u}_{i_{k-1}}), \mathbf{T}^{[i]}_{wj_{k-1}}, \dots\right). \tag{6.70}$$

The neighbor poses $\mathbf{T}_{wj_{k-1}}$ are now out of date, as neighboring odometry information is not yet accessible to Robot i, and this partially out-of-date state is non-physical and given the symbol $\mathfrak{X}_{i_k}$. When a neighbor's RMI $\Delta\mathbf{T}_{j_{pq}}$ is received at some later time step $k = q$, the state is updated with

$$\mathcal{X}_{i_q} = \left(\mathbf{T}^{[i]}_{iw_q}, \mathbf{T}^{[i]}_{wj_p}\Delta\mathbf{T}_{j_{pq}}, \dots\right) \tag{6.71}$$

where p represents the time step index of the last time a neighbor RMI was received.

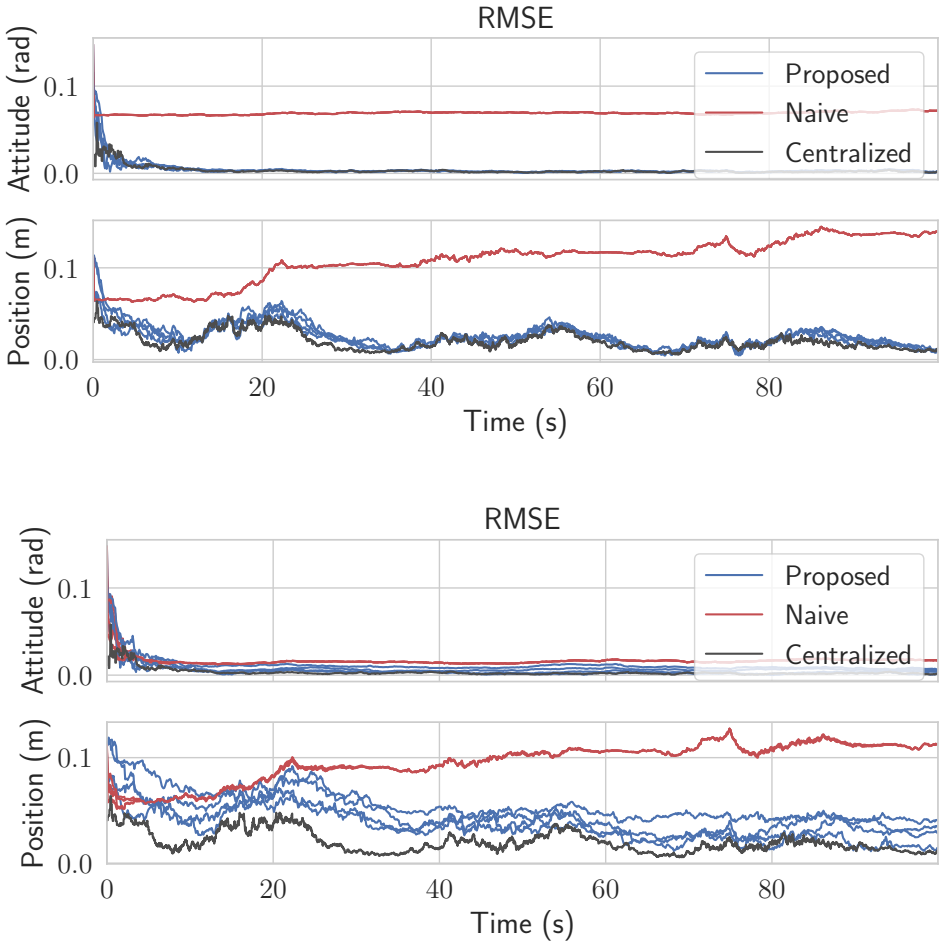

Figure 6.7: RMSE for the ground robot simulation. There are four blue lines for the four robots running the proposed algorithm, and four visibly coincident red lines for the naive algorithm. **Top:** State fusion occuring at 10 Hz. **Bottom:** State fusion occuring at 1Hz.

Each robot also collects range measurements to its neighbors at 10 Hz, and only two robots collect relative position measurements to known landmarks at 10 Hz. At an arbitrary separate frequency, each robot sends its current state and covariance to its neighbors, allowing the neighbors to compute pseudomeasurements of the form

$$\mathbf{c}_{ij}(\mathcal{X}_i, \mathcal{X}_j) = \begin{bmatrix} \operatorname{Log}\left(\mathbf{T}_{wi}^{[i]^{-1}}\mathbf{T}_{wi}^{[j]}\right) \\ \operatorname{Log}\left(\mathbf{T}_{wj}^{[i]^{-1}}\mathbf{T}_{wj}^{[j]}\right) \\ \operatorname{Log}\left(\mathbf{T}_{w\ell}^{[i]^{-1}}\mathbf{T}_{w\ell}^{[j]}\right) \\ \vdots \end{bmatrix}, \qquad \ell \in \mathcal{N}_i \cap \mathcal{N}_j. \tag{6.72}$$

A simulation is performed with 4 robots each executing Algorithm 3, with root-mean-squared error (RMSE) and normalized estimation error squared (NEES) plots shown in

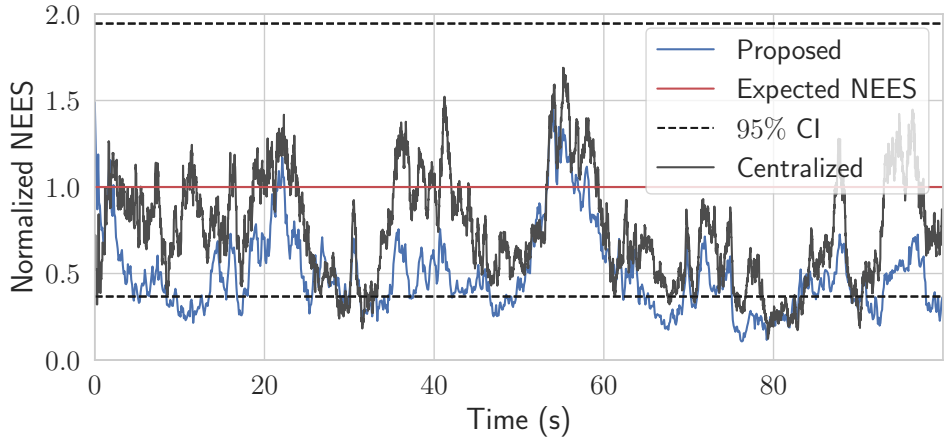

Figure 6.8: NEES plots for the ground robot simulation with 95% confidence bounds for the proposed vs. centralized solution. The naive solution without CI is far outside the plot.

Figure 6.7 and 6.8, respectively. The results show that all four robots' estimation errors successfully stabilize and remain low, despite only two robots having sufficient sensors to make their states observable. Due to covariance intersection, the decentralized solution is slightly underconfident, but this is preferred compared to overconfidence. If state fusion is done at a sufficiently high frequency, performance is even comparable to the centralized estimator.

6.9 Simulation and Experiments with Quadcopters

To demonstrate the flexibility of the proposed framework, consider a new problem involving quadcopters. The kinematic state of each quadcopter is modelled using extended pose matrices $\mathbf{T} \in SE_2(3)$ ([51]). Each robot estimates both their absolute pose relative to the world frame $\mathbf{T}_{wi} \in SE_2(3)$, their own IMU bias $\mathbf{b}_i$, as well as the *relative* poses of their neighbors $\mathbf{T}_{ij} \in SE_2(3)$ and their IMU bias $\mathbf{b}_j$. The full state of Robot i is then given by

$$\mathcal{X}_i = (\mathbf{T}_{wi}, \mathbf{b}_i^{[i]}, \mathbf{T}_{ij}, \mathbf{b}_j^{[i]}, \ldots), \qquad j \in \mathcal{N}_i. \tag{6.73}$$

The pose of Robot j relative to Robot i $\mathbf{T}_{ij}$ has kinematics involving the IMU measurements of both robots, and are given in discrete time by

$$\mathbf{T}_{ij_k} = \mathbf{U}_{i_{k-1}}^{-1} \mathbf{T}_{ij_{k-1}} \mathbf{U}_{j_{k-1}}, \tag{6.74}$$

where $\mathbf{U}_{j_{k-1}}$ has an identical definition as in (6.56), but computed from Robot j's IMU measurements. When Robot i receives input measurements from its own IMU $\mathbf{u}_k$, it predicts the part of its own state corresponding to its own pose, and additionally performs a partial

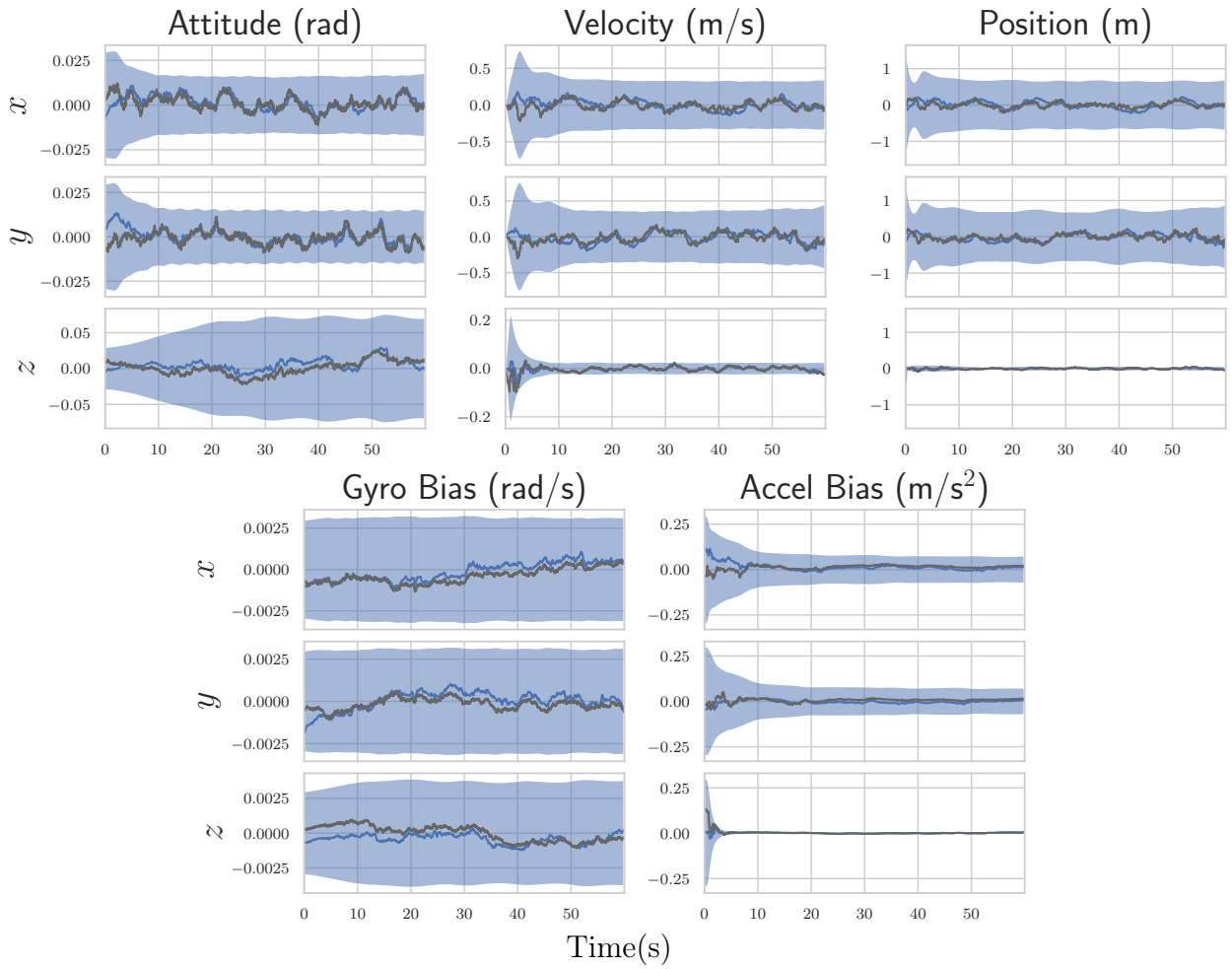

Figure 6.9: Simulation estimation error of Robot 3's estimate of its own kinematic state and IMU biases. The estimate and corresponding bounds with the proposed algorithm is shown in blue, with the centralized estimate overlayed in dark grey. For attitude, the $x-y-z$ components represent roll-pitch-yaw errors respectively. Note that Robot 3 does not have GPS measurements, and therefore cannot observe the states shown in this plot without information sharing. The naive solution has rapidly diverging error and is not plotted.

prediction on the relative poses with

$$\mathfrak{X}_{i_k} = (\mathbf{G}_{k-1}\mathbf{T}_{wi_{k-1}}\mathbf{U}_{i_{k-1}}, \mathbf{b}^{[i]}_{i_{k-1}}, \mathbf{U}^{-1}_{i_{k-1}}\mathbf{T}_{ij_{k-1}}, \mathbf{b}^{[i]}_{j_{k-1}}, \ldots). \tag{6.75}$$

The terms $\mathfrak{T}_{ij_{k-1}} \triangleq \mathbf{U}^{-1}_{i_{k-1}}\mathbf{T}_{ij_{k-1}}$, which are the partially-predicted neighbor poses, are a strange, non-physical intermediate state. Only when the neighbor's RMI $\Delta\mathbf{U}_{j_{pq}}$ is received do the neighbor poses regain meaning with $\mathbf{T}_{ij_q} = \mathfrak{T}_{ij_p}\Delta\mathbf{U}_{j_{pq}}$. However, since biases are also being estimated in this problem, Robot i must first correct the neighbor's raw RMIs $\Delta\mathbf{U}_{j_{pq}}(\mathbf{u}_{j_{p:q-1}}, \mathbf{0})$ using its estimate of the neighbor's IMU bias, as described in Section 6.7.2. That is,

$$\Delta\mathbf{U}_{j_{pq}} \approx \Delta\mathbf{U}_{j_{pq}}(\mathbf{u}_{j_{p:q-1}}, \mathbf{0}) \oplus \mathbf{B}_{j_{pq}}\mathbf{b}^{[i]}_{j_q}, \tag{6.76}$$

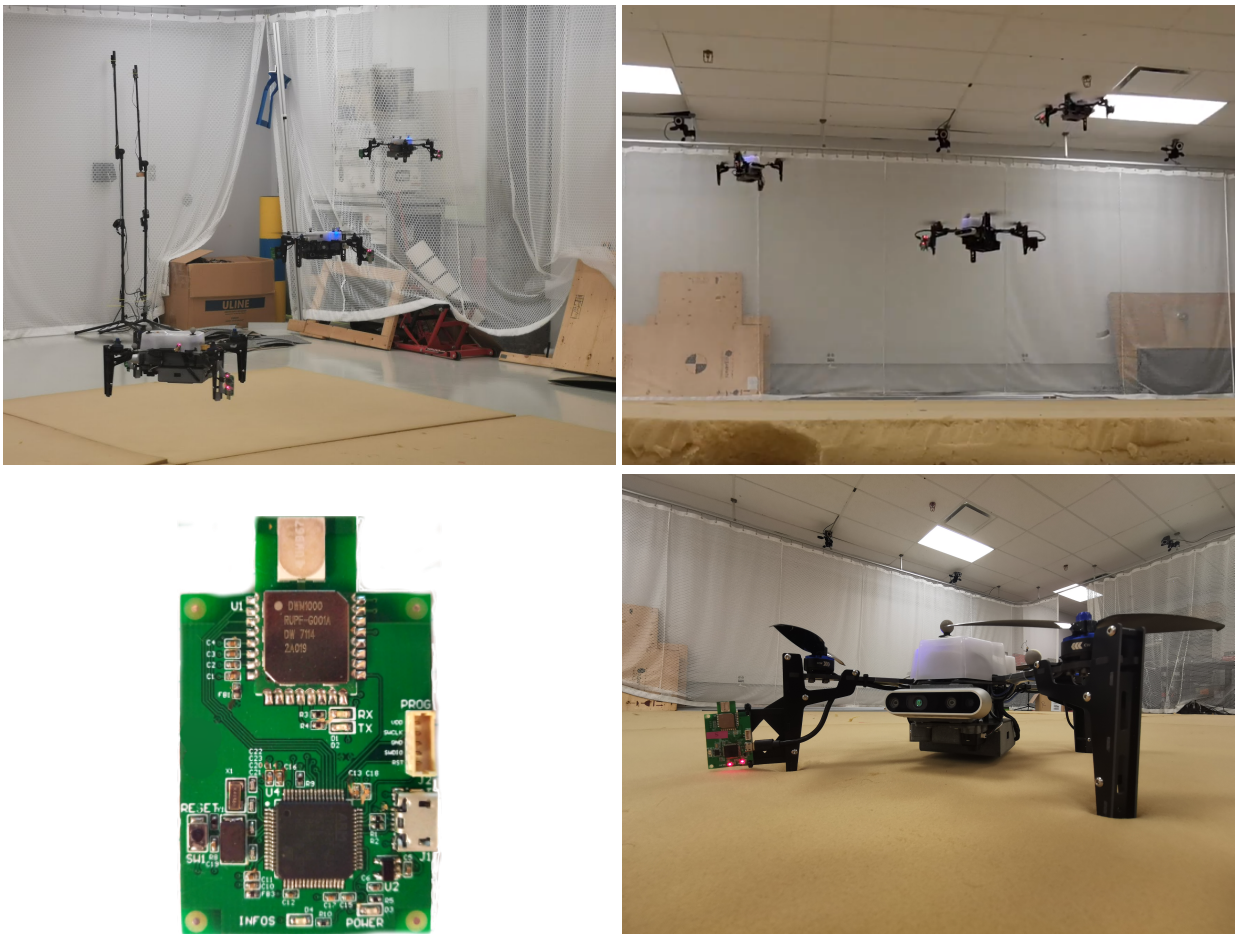

Figure 6.10: **Top:** Three quadcopters in flight under a motion capture system. **Bottom left:** our custom UWB module. **Bottom right:** a close up of the Uvify IFO-S quadcopter, fitted with a UWB module seen on the left leg, as well as on the opposite leg.

leading to the full state update given by

$$\mathcal{X}_{i_q} = (\mathbf{T}_{wi_q}, \mathbf{b}_q^{[i]}, \mathfrak{T}_{ij_p}\Delta\mathbf{U}_{j_{pq}}, \mathbf{b}_{j_q}^{[i]} \ldots). \tag{6.77}$$

Finally, the pseudomeasurements chosen for this problem are

$$\mathbf{c}_{ij}(\mathcal{X}_i, \mathcal{X}_j) = \begin{bmatrix} \operatorname{Log}\left(\mathbf{T}_{wi}\mathbf{T}_{ij}\mathbf{T}_{wj}^{-1}\right) \\ \operatorname{Log}\left(\mathbf{T}_{ij}\mathbf{T}_{ji}\right) \\ \mathbf{b}_i^{[i]} - \mathbf{b}_i^{[j]} \\ \mathbf{b}_j^{[i]} - \mathbf{b}_j^{[j]} \\ \operatorname{Log}\left(\mathbf{T}_{ij}\mathbf{T}_{j\ell}\mathbf{T}_{i\ell}^{-1}\right) \\ \vdots \end{bmatrix}, \qquad \ell \in \mathcal{N}_i \cap \mathcal{N}_j. \tag{6.78}$$

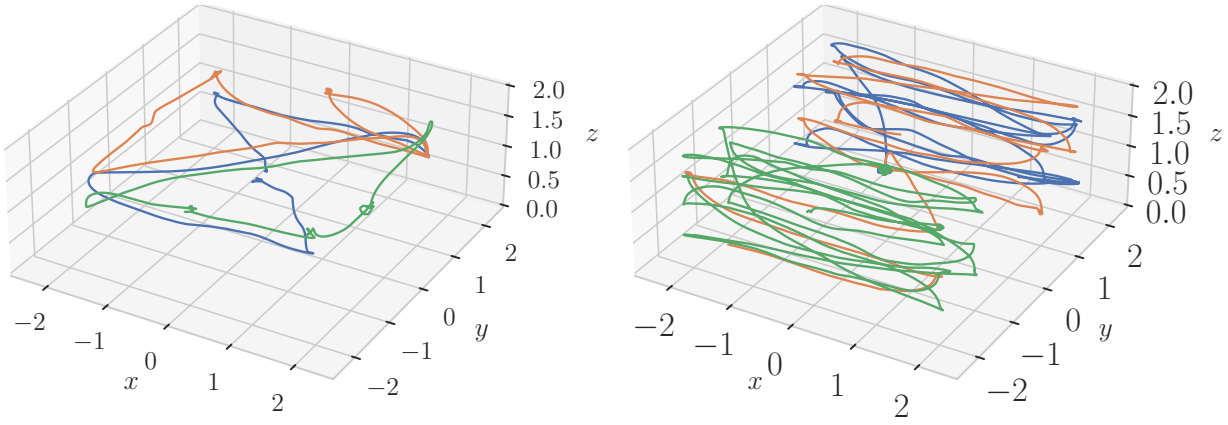

Figure 6.11: Examples of the various trajectories flown in the experimental trials, where each color represents a different quadcopter.

Table 6.1: Self- Positioning RMSE (m) from experimental trials.

Trial #	Centralized			Proposed			Error Reduction		
	Robot 1	Robot 2	Robot 3	Robot 1	Robot 2	Robot 3	Robot 1	Robot 2	Robot 3
1	0.43	0.49	0.55	0.22	0.22	0.61	-48%	-54%	10%
2	0.18	0.26	0.34	0.16	0.18	0.40	-13%	-28%	16%
3	0.17	0.24	0.68	0.16	0.17	0.45	-10%	-28%	-33%
4	0.20	0.25	0.31	0.26	0.28	0.48	32%	-13%	55%

6.9.1 Simulation Results

The algorithm is first tested with simulated versions of the described quadcopters, and the estimation results for Robot 3's absolute pose and bias are shown in Figure 6.9. Although there are many other states associated with the simulation, these states are the most interesting as they are the ones that are unobservable without incorporation of the pseudomeasurements. Figure 6.9 shows that Robot 3 is capable of estimating its own absolute pose and bias, using information from sensors located on Robots 1 and 2. Furthermore, the errors remain within the 3-sigma confidence bounds, even with the first-order RMI bias correction, indicating statistical consistency.

6.9.2 Hardware Setup

The hardware setup in these experiments can be seen in Figure 6.10. Three Uvify IFO-S quadcopters are used that each possess an IMU at 200Hz, a 1D LIDAR height sensor at 30Hz, and magnetometers at 30Hz. Additionally, two ultra-wideband (UWB) transceivers are installed on the quadcopter legs, producing inter-robot distance measurements at 90Hz for each robot. The UWB transceivers are custom-printed modules that use the DW1000 UWB

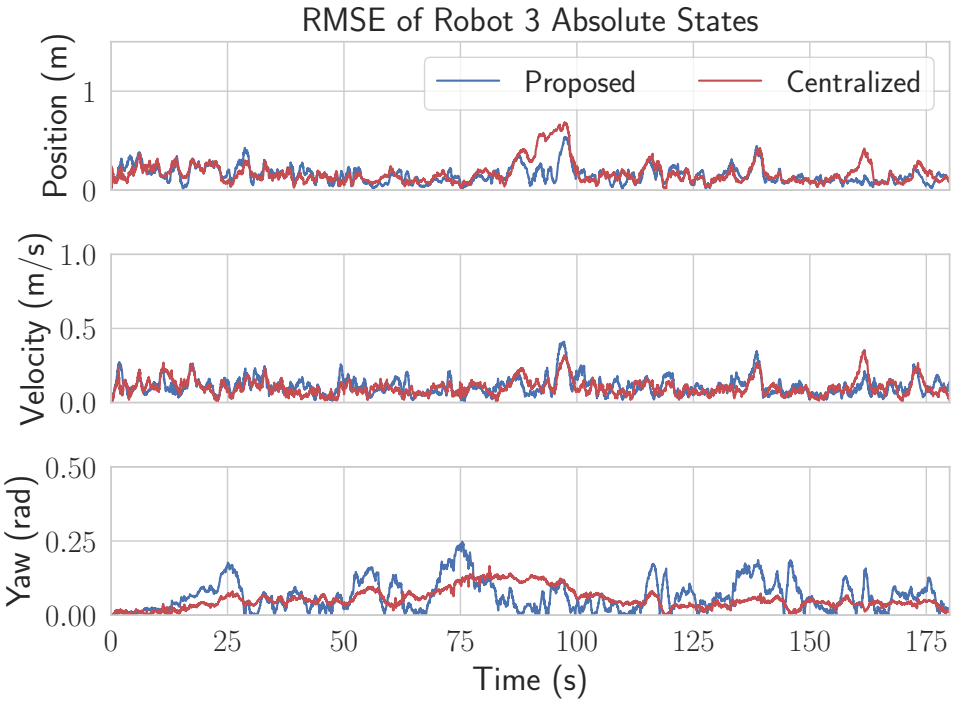

Figure 6.12: Position, velocity, and yaw RMSE for Robot 3 from one of the experimental trials. Since Robot 3 has no GPS, these quantities are unobservable without the fusion of pseudomeasurements.

transceiver. The firmware for these modules has been written in C, implementing a double-sided two-way-ranging protocol with details described by [16]. [16] also describe the power-based bias calibration and noise characterization procedure used in these experiments. Since all transceivers operate on the same frequency in these experiments, only one can transmit at a time to avoid interference. A decentralized scheduler is therefore implemented that continuously cycles through all transceiver pairs one at a time, obtaining range measurements and potentially transmitting other useful data.

A Vicon motion capture system is used to collect ground truth, from which synthesized GPS measurements with a standard deviation of 0.3 m are generated for Robots 1 and 2 only. Robot 3 does not possess GPS measurements, nor any magnetometer measurements, and therefore has no absolute pose information available without communication with the other two robots. Example trajectories for some of the experimental trials are shown in Figure 6.11.

6.9.3 Experimental Results

Multiple experimental runs are performed on different days, with the absolute positioning results for each robot viewable in Table 6.1. In some cases, the proposed algorithm even

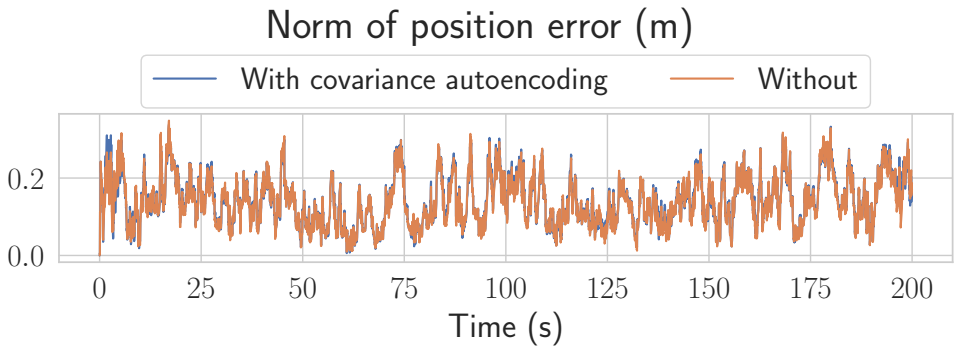

Figure 6.13: The effect of preintegrated covariance autoencoding, as described in Section 6.7.3, on the position estimate of Robot 1. The two lines are almost identical, showing that the proposed autoencoder induces minimal error on the estimate. All other states have similar plots.

outperforms the centralized solution, which is theoretically optimal. However, the real world contains many unmodelled sources of error, such as frame misalignments, timestamping errors, vibrations, and UWB ranging outliers. Even after substantially increasing the datasheet-provided covariances associated with the IMU measurements, it appears that the results benefit from the covariance inflation resulting from CI. For both estimators, the IMU is calibrated to compensate for large biases and scaling factors. The normalized-innovation-squared test ([44]) is also used to reject UWB outliers in both estimators,

A plot of RMSE versus time for Robot 3's absolute states can be seen in Figure 6.12, which are states that are unobservable from Robot 3's own measurements. Again, Figure 6.12 shows that error magnitudes lie in similar ranges for both the centralized and decentralized estimators. Figure 6.13 compares two decentralized estimator runs, with one using the mean-assisted autoencoder from Section 6.7.3. As desired, the lines are identical, and the plot shows that the estimate is unaffected by the autoencoding. This means that the autoencoder is highly effective at compressing the covariance matrix with minimal reconstruction error.

6.10 Conclusion and Future Work

This chapter presents a general-purpose algorithm for decentralized state estimation in robotics. The algorithm is the result of a new way to formulate the decentralized state estimation problem, specifically with the assistance of pseudomeasurements that allow the definition of arbitrary nonlinear relationships between robot states. Another core contribution is the use of preintegration when robots require access to each other's process model input information. Using the proposed preintegration, along with a proposed covariance autoencoding method, communication requirements between robots are reduced substantially. The algorithm is tested on three different problems, each involving a variety of

state definitions, process models, and measurements. Thanks to covariance intersection, the algorithm is appropriate for arbitrary, potentially time-varying graphs, and does not require any bookkeeping, growing memory, buffering of measurements, or reprocessing of data. At the same time, the approximation made by covariance intersection makes the proposed method suboptimal in terms of estimation variance, as it is well-known to be overly-conservative. Nevertheless, in the specific problems shown in this chapter, the results using CI have been satisfactory provided that the fusion frequency is high enough.

Future improvements to the framework proposed in this chapter include the following.

- **Improving the approximation made by covariance intersection.**
  While covariance intersection can sometimes be effective, and is arguably the simplest non-diverging solution, it can lead to poor performance for some problems. A first improvement could be to use *split covariance intersection* (SCI) as shown in [77], or a more recent approach presented in [90] that more directly tracks cross-correlations, but still only approximately.
- **Decentralized batch and sliding-window estimators.**
  It should be possible to also derive decentralized full-batch and sliding-window estimators, often termed *smoothers*, given that these both originate from the MAP approach, which is also what is done in this chapter. Sliding-window filters and incremental batch approaches have become popular in the SLAM and visual-inertial odometry (VIO) literature [50, 94, 95]. Developing decentralized smoothers could allow for tight-coupling of relative states within a robot's VIO algorithm.

Chapter 7

Concluding Remarks

This thesis presents various algorithms for relative position or pose estimation between robots. The algorithms are analyzed and evaluated in simulation and experimental data, and an effort has been made to limit the sensor choice to small, cheap, easy-to-process sensors. For this reason, the use of cameras and LIDAR are notable omissions from the sensor choices in this thesis.

A summary of contributions can be found in the front matter to this thesis. The final section of each chapter expands more on the contributions made in that chapter, provides a comprehensive discussion on the limitations of the corresponding algorithm, and finally suggests many future research directions that could address these limitations.

The most accurate real relative positioning estimates obtained in this thesis were in the range of 0.19 m - 0.26 m, obtained in Chapter 6, with multiple UWB tags per quadcopter, as well as height, IMU, and GPS measurements for 2 out of 3 robots. This accuracy may be impressive given the limited sensors used, but is still significantly more error than the results reported in [10], for example, which reports an error of 0.05 m with the assistance of stereo cameras with direct visual detection of neighboring robots. The easiest way to improve the accuracy of the relative position estimates would be to add more sensors to each robot. Chapter 6 is fortunately general to arbitrary measurement models, and hence the proposed method is still applicable even when more sensors are added. Nevertheless, even with the existing sensor set used in this thesis, there were still many practical issues that degraded accuracy. This will be discussed in the next section.

7.1 Future Work

While UWB has many attractive features, it remains a challenging sensor to work with. Apart from the fact that a single distance measurement is not a lot of information, UWB is

prone to ranging error as a result of unmodelled antenna delays, variable antenna radiation patterns, non-line-of-sight conditions, and multipath propagation. An attempt at calibrating-out these effects has been documented in recent work [16], but it could be possible to further improve this calibration procedure so that the range measurements themselves become more accurate.

Most results in this thesis rely on inertial measurement units (IMU) as a source of odometry. IMUs are extremely cheap and lightweight, but require a double integration procedure to produce position priors, and may have biases that must be estimated online. Furthermore, as seen in flying experiments, vibrations heavily degrade the dead-reckoned estimate obtained from the IMU. It was necessary to dramatically increase the associated covariances, compared to the IMU's datasheet, in order to obtain consistent estimates. If the odometry can be improved in any way, it will likely result in significantly more accurate estimates.

- **Relative-pose-dependent UWB bias correction.**
  Recent co-authored work in [16] fits a simple polynomial model that maps received signal power to a ranging bias. This method is effective and simple to use, since it does not require any robot state information. However, inspired by [96], further improvements could be obtained by learning a relationship between the relative poses between the tags and the bias. One downside of this approach, is that the bias calibration would become a function of the state, and must therefore be directly in the loop with the estimator. This could yield less predictable, more complex behavior.
- **Improved inertial dead-reckoning for quadcopters.**
  Inertial dead-reckoning on quadcopters is heavily degraded by the presence of vibrations, and the solution drifts to hundreds of meters of error in a matter of seconds. Apart from rigorous IMU calibration, one way to improve the dead-reckoning would be to identify moments of near-zero velocity [53], which can then lead to the creation of zero-velocity pseudomeasurements, that can be incorporated into any of the filters described in this thesis. Another approach could be to learn displacements or velocity from IMU measurements directly using a data-driven model, as done in [97].
- **Gaussian sum filtering for range-based relative pose estimation.**
  This idea is particularly applicable to a setup similar to that described in Chapter 5. That is, relative pose estimation where each robot possesses two UWB tags. In this case, while the relative poses are *locally* unique given the range measurements, there are still many other discrete formations that produce the same measurements. This, in turn, gives rise to a multimodal distribution as the posterior distribution of relative poses. As such, this problem is a natural use-case for the Gaussian sum filter [98]. A

Gaussian sum filter could be initialized with an estimate at each mode, which would then collapse to a single-mode distribution once the the robots move sufficiently.

Appendices

Appendix 1

The Complex-Step Derivative Approximation on Matrix Lie Groups

A.1 Introduction

Path planning, state estimation, and control algorithms often require Jacobian computations with respect to attitude and pose. Often these Jacobians are computed analytically, by hand while adhering to the matrix Lie group structure of the problem [43]. However, in some cases analytical computation of Jacobians may be impractical, necessitating a numerical procedure. Numerical computation of Jacobians is also useful for quickly comparing algorithms that require Jacobians, before investing effort into one specific algorithm and the associated analytically derived Jacobians. Numerical Jacobians can also be used to verify Jacobians that are derived by hand.

A variety of numerical differentiation techniques appropriate for matrix Lie groups can be found in the literature. In [74] a forward-difference method is described for general matrix manifolds, a method that is used in the open-source software MANOPT [99]. A central-difference method is employed in the open-source software GTSAM [100], and automatic differentiation methods are presented in [101, 102]. The Python-based software PYMANOPT [103] is an open-source optimization toolbox for matrix manifolds that employs automatic differentiation. SOPHUS [104] is another open-source C++ package that exploits the automatic differentiation functionality available in CERES [105], a nonlinear least-squares library developed by Google. However, automatic differentiation can be time consuming to implement and finite-differencing is prone to subtractive cancellation errors, thus limiting precision [106]. The complex-step derivative approximation is a numerical method for computing first derivatives that does not suffer from subtractive cancellation errors [106]. One of the earlier appearances of the complex-step derivative can be found in [107], where

the derivatives of scalar functions of real variables are evaluated. In [106], the complex-step derivative is investigated further, along with its use in Fortran, C/C++, and other languages. An application to a multidisciplinary design optimization problem is also shown. This method has gained popularity due to its ability to realize machine-precision accuracy of derivative computations, and doing so without tuning the step size, since it can be reduced to an arbitrarily small value. The complex-step derivative also requires only one complex function evaluation.

This appendix considers the formulation and application of the complex-step derivative approximation to functions of matrix Lie group elements. The aforementioned advantages of the standard complex-step derivative remain present, while the proposed method can be used to compute both left and right Jacobians.

A.2 Review

Consider the complex-differentiable function $f : \mathbb{C} \to \mathbb{C}$ perturbed about the nominal point $\bar{x}$ by jh where $\bar{x}, h \in \mathbb{R}$ and $j = \sqrt{-1}$. A Taylor series expansion yields

$$f(\bar{x} + jh) = f(\bar{x}) + \left.\frac{\partial f(z)}{\partial z}\right|_{z=\bar{x}} jh - \frac{1}{2}\left.\frac{\partial^2 f(z)}{\partial z^2}\right|_{z=\bar{x}} h^2 - \frac{1}{3!}\left.\frac{\partial^3 f(z)}{\partial z^3}\right|_{z=\bar{x}} jh^3 \dots \tag{A.1}$$

If $f(\bar{x})$ is assumed to be real for all real $\bar{x}$, then, to first order, taking the imaginary portion of (A.1) yields [107]

$$\left.\frac{\partial f(z)}{\partial z}\right|_{z=\bar{x}} = \frac{\mathrm{Im}\{f(\bar{x} + jh)\}}{h} + \mathcal{O}(h^2).$$

This is valid as long as $f(\bar{x}) \in \mathbb{R}$ for all $\bar{x} \in \mathbb{R}$, and that derivatives are evaluated at strictly real nominal points. From a practical standpoint, a user is often attempting to find derivatives of $f : \mathbb{R} \to \mathbb{R}$. Providing that this can be extended to $f : \mathbb{C} \to \mathbb{C}$ such that f is complex-differentiable, then with a minor abuse of notation, this can construct a derivative approximation for $f(x)$ as written in [106, 107],

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} \approx \frac{\mathrm{Im}\{f(x + jh)\}}{h}.$$

Since there are no subtractive cancellation errors, the complex-step derivative approximation can produce machine-precision approximations by reducing h to an arbitrarily small step size.

A.3 The Complex-Step Derivative on Matrix Lie Groups

Consider a complex-differentiable function $f : G \to \mathbb{C}$, $\mathbf{X}(\boldsymbol{\epsilon}^R) = \bar{\mathbf{X}} \exp(\boldsymbol{\epsilon}^{R^\wedge})$ is parametrizable by a perturbation $\boldsymbol{\epsilon}^R = [\epsilon_1^R \; \epsilon_2^R \dots \epsilon_n^R]^\mathsf{T} \in \mathbb{C}^n$ on the right, and $\bar{\mathbf{X}} \in \mathbb{R}^{m \times m}$ is some nominal

value of $\mathbf{X}$. Consider perturbing $f(\mathbf{X}(\boldsymbol{\epsilon}^R))$ by $\boldsymbol{\epsilon}^R = \mathbf{0} + jh\mathbf{1}_i$, where $\mathbf{1}_i$ is the i^{th} column of the appropriately-dimensioned identity matrix $\mathbf{1}$. The composition $f(\mathbf{X}(\boldsymbol{\epsilon}^R))$ has essentially recast f as $f : \mathbb{C}^n \to \mathbb{C}$, from which a Taylor series expansion yields [108, Ch. 11.3]

$$f\left(\bar{\mathbf{X}}\exp((jh\mathbf{1}_i)^\wedge)\right) = f(\bar{\mathbf{X}}) + \left.\frac{\partial f(\mathbf{X}(\boldsymbol{\epsilon}^R))}{\partial \epsilon_i^R}\right|_{\boldsymbol{\epsilon}^R=\mathbf{0}} jh - \frac{1}{2}\left.\frac{\partial^2 f(\mathbf{X}(\boldsymbol{\epsilon}^R))}{\partial \epsilon_i^{R2}}\right|_{\boldsymbol{\epsilon}^R=\mathbf{0}} h^2 + \mathcal{O}(h^3). \quad \text{(A.2)}$$

Since it is assumed that $f(\bar{\mathbf{X}}) \in \mathbb{R}$, taking the imaginary component of (A.2) yields an approximation for the derivative

$$\frac{\partial f(\mathbf{X}(\boldsymbol{\epsilon}^R))}{\partial \epsilon_i^R} \approx \frac{\text{Im}\{f\left(\bar{\mathbf{X}}\exp((jh\mathbf{1}_i)^\wedge)\right)\}}{h}. \quad \text{(A.3)}$$

The right Jacobian $\partial f(\mathbf{X}(\boldsymbol{\epsilon}^R))/\partial \boldsymbol{\epsilon}^R$ can be obtained by individually computing the derivatives using (A.3) with $i = 1, 2, \ldots, n$. The left Jacobian can identically be obtained by instead parametrizing $\mathbf{X}$ with $\mathbf{X}(\boldsymbol{\epsilon}^L) = \exp(\boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{X}}$. This leads to

$$\frac{\partial f(\mathbf{X}(\boldsymbol{\epsilon}^L))}{\partial \epsilon_i^L} \approx \frac{\text{Im}\{f\left(\exp((jh\mathbf{1}_i)^\wedge)\bar{\mathbf{X}}\right)\}}{h}. \quad \text{(A.4)}$$

Note that the superscripts on $\boldsymbol{\epsilon}^R$ and $\boldsymbol{\epsilon}^L$ are simply labels that correspond to right and left perturbations, respectively, as opposed to exponents.

Example: Consider the function

$$f(\mathbf{T}) = \mathbf{v}^\mathsf{T}\mathbf{T}\mathbf{y},$$

where $\mathbf{T} \in SE(3)$ and $\mathbf{v}, \mathbf{y} \in \mathbb{R}^4$. The left Jacobian can be determined analytically using the first-order approximation $\mathbf{T} = \exp(\boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{T}} \approx (\mathbf{1} + \boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{T}}$ and a Taylor series expansion. To this end,

$$\begin{aligned} f(\exp(\boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{T}}) &= \mathbf{v}^\mathsf{T}\exp(\boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{T}}\mathbf{y} \\ &\approx \mathbf{v}^\mathsf{T}(\mathbf{1} + \boldsymbol{\epsilon}^{L\wedge})\bar{\mathbf{T}}\mathbf{y} \\ &= \mathbf{v}^\mathsf{T}\bar{\mathbf{T}}\mathbf{y} + \underbrace{\mathbf{v}^\mathsf{T}(\bar{\mathbf{T}}\mathbf{y})^\odot}_{\left.\frac{\partial f(\mathbf{T}(\boldsymbol{\epsilon}^L))}{\partial \boldsymbol{\epsilon}^L}\right|_{\boldsymbol{\epsilon}^L=\mathbf{0}}} \boldsymbol{\epsilon}^L. \end{aligned} \quad \text{(A.5)}$$

The elements of $\partial f(\mathbf{T}(\boldsymbol{\epsilon}^L))/\partial \boldsymbol{\epsilon}^L$ are computed using (A.4) with varying step sizes h, and the results are compared with a central-difference scheme in Fig. A.1. The error is computed by taking the relative 2-norm of the difference between the analytical and numerical solutions. Like the standard complex-step derivative, the complex-step derivative tailored to the matrix Lie group $SE(3)$ is able to achieve analytic accuracy, up to machine precision, for small enough h, while the central-difference derivative is not.

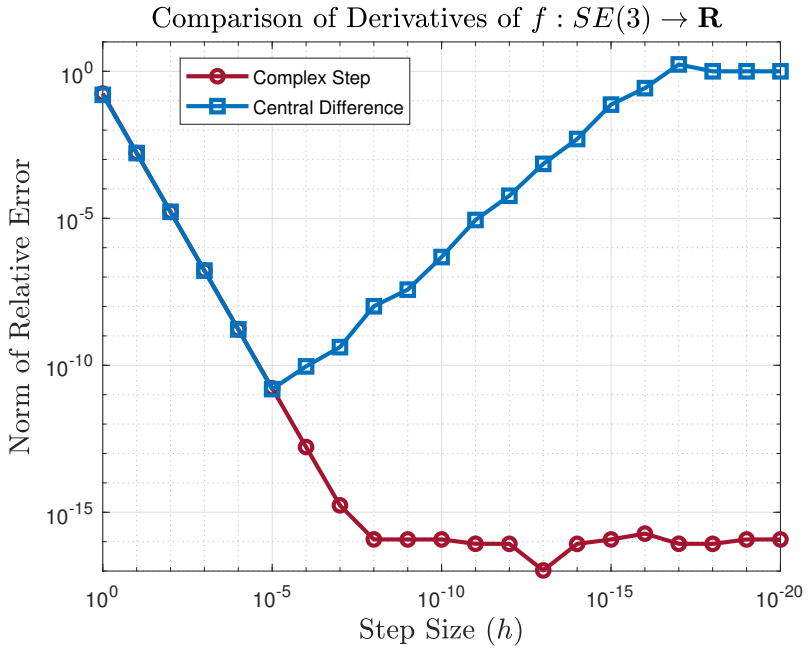

Figure A.1: Variation of relative error in gradient of $f : SE(3) \rightarrow \mathbb{R}$ with step size, for both complex-step and central-difference methods. Machine precision is achievable with a sufficient reduction in step size.

Appendix 2

Contributed open-source software

This thesis has also led to the development of two open-source Python packages that can be used for general state estimation purposes:

- **pylie - Lie groups operations in Numpy, Jax, and C++.**
  `https://github.com/decargroup/pylie`
  Pylie is an instantiation-free Python package for common matrix Lie group operations implemented as pure static classes. Using pure static classes keeps the usage extremely simple while still allowing for abstraction and inheritance. The package does not introduce new objects, with everything operating directly on Numpy or Jax arrays. This allows users to implement their own more sophisticated objects using these classes as back-end mathematical implementations.
- **pynav - An on-manifold state estimation library.**
  `https://github.com/decargroup/pynav`
  The core idea behind this project is to use abstraction and polymorphism such that a single estimator implementation can operate on a variety of state manifolds, such as the usual vector space, and any Lie group. This is done by defining a set of abstract classes that the user must implement, corresponding to a state definition, process model, and measurement model. A number of common general-purpose estimation algorithms are available, such as the EKF, Iterated EKF, and sigma-point filters, in addition to a variety of common process and measurement models.

Bibliography

[1] C. C. Cossette, M. Shalaby, D. Saussie, J. R. Forbes, and J. Le Ny, "Relative Position Estimation between Two UWB Devices with IMUs", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4313–4320, 2021. arXiv: 2104.10730.

[2] C. C. Cossette, M. Shalaby, D. Saussie, and J. R. Forbes, "Localization with Directional Coordinates", in *IEEE International Conference on Intelligent Robots and Systems*, Prague, 2021, pp. 2253–2258. arXiv: 2109.09229.

[3] C. C. Cossette, M. A. Shalaby, D. Saussie, J. L. Ny, and J. R. Forbes, "Optimal Multi-robot Formations for Relative Pose Estimation Using Range Measurements", in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Kyoto, 2022, pp. 2431–2437.

[4] C. C. Cossette, M. A. Shalaby, D. Saussié, and J. R. Forbes, "On-manifold Decentralized State Estimation using Pseudomeasurements and Preintegration", vol. 14, no. 8, pp. 1–15, 2023. arXiv: 2304.04036. [Online]. Available: http://arxiv.org/abs/2304.04036.

[5] C. C. Cossette, A. Walsh, and J. R. Forbes, "The Complex-Step Derivative Approximation on Matrix Lie Groups", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 906–913, 2020.

[6] K. Guo, Z. Qiu, W. Meng, L. Xie, and R. Teo, "Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments", *Intl. Jrnl. of Micro Air Vehicles*, vol. 9, no. 3, pp. 169–186, 2017.

[7] T. M. Nguyen, A. H. Zaini, C. Wang, K. Guo, and L. Xie, "Robust Target-Relative Localization with Ultra-Wideband Ranging and Communication", in *IEEE Intl. Conf. on Robotics and Automation*, Brisbane, 2018, pp. 2312–2319. arXiv: 1802.08953.

[8] K. Guo, X. Li, and L. Xie, "Ultra-wideband and odometry-based cooperative relative localization with application to multi-UAV formation control", *IEEE Trans. on Cybernetics*, vol. 50, no. 6, pp. 2590–2603, 2020.

[9] T. H. Nguyen and L. Xie, "Relative Transformation Estimation Based on Fusion of Odometry and UWB Ranging Data", pp. 1–15, 2022. arXiv: 2202.00279. [Online]. Available: http://arxiv.org/abs/2202.00279.

[10] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm", in *IEEE Intl. Conf. on Robotics and Automation*, Paris, 2020, pp. 8776–8782. arXiv: 2003.05138.

[11] A. W. Long, C. K. Wolfe, M. J. Mashner, and G. S. Chirikjian, "The banana distribution is Gaussian: A localization study with exponential coordinates", *Robotics: Science and Systems*, vol. 8, pp. 265–272, 2013.

[12] T. D. Barfoot and P. T. Furgale, "Associating Uncertainty with Three-Dimensional Poses for use in Estimation Problems", *IEEE Trans. on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[13] A. Barrau and S. Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer", *IEEE Trans. on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2017. arXiv: 1410.1465.

[14] A. Barrau and S. Bonnabel, "Three examples of the stability properties of the invariant extended Kalman filter", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 431–437, 2017. [Online]. Available: https://doi.org/10.1016/j.ifacol.2017.08.061.

[15] Z. Sahinoglu, S. Gezici, and I. Guvenc, *Ultra-wideband Positioning Systems*. New York, NY: Cambridge University Press, 2008.

[16] M. A. Shalaby, C. C. Cossette, J. R. Forbes, and J. L. Ny, "Calibration and Uncertainty Characterization for Ultra-Wideband Two-Way-Ranging Measurements (preprint)", 2022. arXiv: 2210.05888. [Online]. Available: https://arxiv.org/abs/2210.05888v2.

[17] M. A. Shalaby, C. Champagne Cossette, J. R. Forbes, and J. Le Ny, "Reducing Two-way Ranging Variance by Signal-Timing Optimization (preprint)", 2022. arXiv: 2211.00538v1. [Online]. Available: https://arxiv.org/abs/2211.00538.

[18] J. Cano, S. Chidami, and J. Le Ny, "A Kalman Filter-Based Algorithm for Simultaneous Time Synchronization and Localization in UWB Networks", in *IEEE Intl. Conf. on Robotics and Automation*, Montreal, 2019, pp. 1431–1437.

[19] I. Dotlic, A. Connell, H. Ma, J. Clancy, and M. McLaughlin, "Angle of Arrival Estimation Using Decawave DW1000 Integrated Circuits", *Workshop on Positioning, Navigation and Communications*, 2017.

[20] N. Patwari, J. N. Ash, S. Kyperountas, A. O. H, R. L. Moses, and N. S. Correal, "Locating the Nodes", *IEEE Signal Processing Magazine*, pp. 54–69, 2005.

[21] A. Ledergerber, M. Hamer, and R. D'Andrea, "Angle of Arrival Estimation based on Channel Impulse Response Measurements", in *IEEE International Conference on Intelligent Robots and Systems*, Macau, 2019, pp. 6686–6692.

[22] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensors*, 1st. The McGraw-Hill Companies, 2008.

[23] S. Krishnan, P. Sharma, Z. Guoping, and O. H. Woon, "A UWB based localization system for indoor robot navigation", in *International Conference on Ultra-Wideband*, Singapore, 2007, pp. 77–82.

[24] V. Mai *et al.*, "Local Positioning System Using UWB Range Measurements for an Unmanned Blimp", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2971–2978, 2018.

[25] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing Ultra-wideband Range Measurements with Accelerometers and Rate Gyroscopes for Quadrocopter State Estimation", in *International Conference on Robotics and Automation*, Seattle, 2015, pp. 1730–1736.

[26] A. Ledergerber, M. Hamer, and R. D'Andrea, "A Robot Self-Localization System using One-Way Ultra-Wideband Communication", in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Hamburg, 2015, pp. 3131–3137.

[27] A. Goudar, K. Pereida, and A. P. Schoellig, "Ultra-Wideband Aided Inertial Navigation: Observability Analysis and Sensor Extrinsics Calibration", in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Las Vegas, 2020. [Online]. Available: `http://www.dynsyslab.org`.

[28] M. Shalaby, C. C. Cossette, J. R. Forbes, and J. Le Ny, "Relative Position Estimation in Multi-Agent Systems Using Attitude-Coupled Range Measurements", *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4955–4961, 2021.

[29] P. Batista, C. Silvestre, and P. Oliveira, "Single Range Aided Navigation and Source Localization: Observability and Filter design", *Systems and Control Letters*, vol. 60, no. 8, pp. 665–673, 2011.

[30] F. She, Y. Zhang, D. Shi, H. Zhou, X. Ren, and T. Xu, "Enhanced Relative Localization Based on Persistent Excitation for Multi-UAVs in GPS-Denied Environments", *IEEE Access*, vol. 8, pp. 148 136–148 148, 2020.

[31] I. Sarras, J. Marzat, S. Bertrand, and H. Piet-Lahanier, "Collaborative multiple micro air vehicles' localization and target tracking in GPS-denied environment from range–velocity measurements", *International Journal of Micro Air Vehicles*, vol. 10, no. 2, pp. 225–239, 2018. [Online]. Available: `https://us.sagepub.com/en-us/nam/open-access-at-sage`.

[32] T. M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, "Distance-Based Cooperative Relative Localization for Leader-Following Control of MAVs", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3641–3648, 2019.

[33] T. M. Nguyen, Z. Qiu, T. H. Nguyen, M. Cao, and L. Xie, "Persistently Excited Adaptive Relative Localization and Time-Varying Formation of Robot Swarms", *IEEE Trans. on Robotics*, vol. 36, no. 2, pp. 553–560, 2020.

[34] T. M. Nguyen, T. H. Nguyen, M. Cao, Z. Qiu, and L. Xie, "Integrated UWB-vision approach for autonomous docking of UAVS in GPS-denied environments", in *International Conference on Robotics and Automation*, Montreal, 2019, pp. 9603–9609.

[35] Y. Xianjia, L. Qingqing, J. P. Queralta, J. Heikkonen, and T. Westerlund, "Cooperative UWB-based localization for outdoors positioning and navigation of UAVs aided by ground robots", in *IEEE International Conference on Autonomous Systems*, Montreal, 2021. arXiv: 2104.00302.

[36] T. D. Barfoot, *State Estimation for Robotics*, 2nd. Cambridge University Press, 2023.

[37] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2010.

[38] B. Hall, *Lie Groups, Lie Algebras and Representations*, second. New York, NY: Springer, 2015.

[39] J. Arsenault, "Practical Considerations and Extensions of the Invariant Extended Kalman Filtering Framework", M.A.Sc. Thesis, McGill University, 2019.

[40] J. Solà, J. Deray, and D. Atchuthan, *A Micro Lie Theory for State Estimation in Robotics*, 2018. arXiv: 1812.01537. [Online]. Available: http://arxiv.org/abs/1812.01537.

[41] M. L. Psiaki, "The blind tricyclist problem and a comparative study of nonlinear filters: A challenging benchmark for evaluating nonlinear estimation methods", *IEEE Control Systems Magazine*, vol. 33, no. 3, pp. 40–54, 2013.

[42] R. Hermann and A. J. Krener, "Nonlinear Controllability and Observability", *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.

[43] T. Barfoot, *State Estimation for Robotics*, 1st. Toronto, ON: Cambridge University Press, 2019.

[44] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: John Wiley & Sons, Inc., 2001.

[45] S. van der Helm, K. N. McGuire, M. Coppola, and G. de Croon, "On-board Range-based Relative Localization for Micro Aerial Vehicles in Indoor Leader-Follower Flight", *Autonomous Robots*, vol. 44, pp. 415–441, 2020.

[46] R. Liu, C. Yuen, T. N. Do, D. Jiao, X. Liu, and U. X. Tan, "Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information", in *IEEE Intl. Conf. on Robotics and Automation*, Singapore, 2017, pp. 5623–5629. arXiv: 1704.01397.

[47] G. Sibley, "A Sliding Window Filter for SLAM", University of Southern California, Tech. Rep., 2006.

[48] T. C. Dong-Si and A. I. Mourikis, "Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis", in *IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 5655–5662.

[49] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization", *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[50] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry", *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[51] M. Brossard, A. Barrau, P. Chauchat, and S. Bonnabel, "Associating Uncertainty to Extended Poses for on Lie Group IMU Preintegration With Rotating Earth", *IEEE Trans. Robotics*, vol. 38, no. 2, pp. 998–1015, 2022.

[52] M. Zhang, M. Zhang, Y. Chen, and M. Li, "IMU Data Processing For Inertial Aided Navigation: A Recurrent Neural Network Based Approach", in *IEEE International Conference on Robotics and Automation*, Xi'an, 2021, pp. 3992–3998. arXiv: 2103.14286.

[53] B. Wagstaff and J. Kelly, "LSTM-Based Zero-Velocity Detection for Robust Inertial Navigation", in *International Conference on Indoor Positioning and Indoor Navigation*, Nantes, 2018, pp. 24–27. arXiv: 1807.05275.

[54] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU Dead-Reckoning", *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020. arXiv: 1904.06064.

[55] A. Boberg, A. N. Bishop, and P. Jensfelt, "Robocentric mapping and localization in modified spherical coordinates with bearing measurements", in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Melbourne: IEEE, 2009, pp. 139–144.

[56] M. Mallick, L. Mihaylova, S. Arulampalam, and Y. Yan, "Angle-only Filtering in 3D using Modified Dpherical and Log Spherical Coordinates", in *Intl. Conf. on Information Fusion*, Chicago, 2011.

[57] D. Lerro and Y. Bar-Shalom, "Tracking With Debiased Consistent Converted Measurements Versus EKF", *IEEE Trans. on Aerospace and Electronic Systems*, vol. 29, no. 3, pp. 1015–1022, 1993.

[58] S. V. Bordonaro, "Converted Measurement Trackers for Systems with Nonlinear Measurement Functions", Ph.D. Thesis, University of Connecticut, 2015.

[59] Y. Dai, J. Trumpf, H. Li, N. Barnes, and R. Hartley, "Rotation averaging with application to camera-rig calibration", in *Asian Conference on Computer Vision*, vol. 5995 LNCS, Berlin, 2009, pp. 335–346.

[60] M. Brossard, A. Barrau, and S. Bonnabel, "A Code for Unscented Kalman Filtering on Manifolds (UKF-M)", in *International Conference on Robotics and Automation*, Paris, 2020.

[61] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence Estimation for Multidimensional Densities via κ-Nearest-Neighbor Distances", *IEEE Trans. on Information Theory*, vol. 55, no. 5, pp. 2392–2405, 2009.

[62] S. Li, C. De Wagter, and G. C. De Croon, "Self-supervised Monocular Multi-robot Relative Localization with Efficient Deep Neural Networks", in *IEEE International Conference on Robotics and Automation*, Philadelphia, 2022, pp. 9689–9695. arXiv: `2105.12797`. [Online]. Available: `http://arxiv.org/abs/2105.12797`.

[63] L. Mao, J. Chen, Z. Li, and D. Zhang, "Relative localization method of multiple micro robots based on simple sensors", *International Journal of Advanced Robotic Systems*, vol. 10, pp. 1–9, 2013.

[64] J. P. Queralta, Q. Li, F. Schiano, and T. Westerlund, "VIO-UWB-Based Collaborative Localization and Dense Scene Reconstruction within Heterogeneous Multi-Robot Systems", in *IEEE International Conference on Advanced Robotics and Mechatronics*, Guilin, 2022, pp. 87–94. arXiv: `2011.00830`. [Online]. Available: `http://arxiv.org/abs/2011.00830`.

[65] H. Xu *et al.*, "Omni-Swarm: A Decentralized Omnidirectional Visual-Inertial-UWB State Estimation System for Aerial Swarms", *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3374–3394, 2022. arXiv: `2103.04131`. [Online]. Available: `http://arxiv.org/abs/2103.04131`.

[66] S. Guler, M. Abdelkader, and J. S. Shamma, "Infrastructure-free Localization of Aerial Robots with Ultrawideband Sensors", in *American Control Conference*, Philadelphia, 2019, pp. 13–18. arXiv: `1809.08218`.

[67] B. Hepp, T. Nägeli, and O. Hilliges, "Omni-directional Person Tracking on a Flying Robot using Occlusion-robust Ultra-wideband Signals", in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Daejeon, 2016, pp. 189–194.

[68] M. Kok and A. Solin, "Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps", in *Intl. Conf. on Information Fusion*, Cambridge, 2018, pp. 1353–1360. arXiv: 1804.01926.

[69] A. Solin, M. Kok, N. Wahlstrom, T. B. Schon, and S. Sarkka, "Modeling and Interpolation of the Ambient Magnetic Field by Gaussian Processes", *IEEE Trans. on Robotics*, vol. 34, no. 4, pp. 1112–1127, 2018. arXiv: 1509.04634.

[70] W. Zhao, M. Vukosavljev, and A. P. Schoellig, "Optimal Geometry for Ultra-wideband Localization using Bayesian Optimization", *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 481–15 488, 2020.

[71] J. Le Ny and S. Chauvière, "Localizability-Constrained Deployment of Mobile Robotic Networks with Noisy Range Measurements", in *American Control Conference*, Milwaukee, 2018, pp. 2788–2793.

[72] J. Cano and J. Le Ny, "Improving Ranging-Based Location Estimation with Rigidity-Constrained CRLB-Based Motion Planning", in *Intl. Conf. on Robotics and Automation*, Xi'an, 2021.

[73] S. Bonnabel and A. Barrau, "An intrinsic Cramér-Rao bound on lie groups", in *Geometric Science of Information*, vol. 9389, Springer International Publishing, 2015, pp. 664–672. arXiv: 1506.05662. [Online]. Available: https://arxiv.org/abs/1506.05662.

[74] P. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton Uni. Press, 2008.

[75] Y. Xia, X. Na, Z. Sun, and J. Chen, "Formation control and collision avoidance for multi-agent systems based on position estimation", *ISA Transactions*, vol. 61, no. 1, pp. 287–296, 2016.

[76] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations", in *American Control Conference*, vol. 4, Albuquerque, 1997, pp. 2369–2373.

[77] S. Julier, "General Decentralized Data Fusion with Covariance Intersection (CI)", in *Multisensor Data Fusion*, 2001, ch. 12, pp. 269–294.

[78] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, and S. I. Roumeliotis, "Decentralized multi-robot cooperative localization using covariance intersection", in *International Conference on Intelligent Robots and Systems*, Tokyo, 2013, pp. 1412–1417.

[79] M. Shalaby, C. C. Cossette, J. Le Ny, and J. R. Forbes, "Cascaded Filtering Using the Sigma Point Transformation", *IEEE Robotics and Automation Letters*, 2021.

[80] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions", *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[81] M. Shalaby, C. C. Cossette, J. R. Forbes, and J. Le Ny, "Multi-Robot Relative Pose Estimation and IMU Preintegration Using Passive UWB Transceivers (preprint)", 2023. arXiv: `2203.11004`. [Online]. Available: `https://arxiv.org/abs/2203.11004`.

[82] S. Grime and H. F. Durrant-Whyte, "Data fusion in decentralized sensor networks (Information Filter)", *Control Engineering Practice*, vol. 2, no. 5, pp. 849–863, 1994.

[83] K. Y. Leung, T. D. Barfoot, and H. H. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach", *IEEE Trans. Robotics*, vol. 26, no. 1, pp. 62–77, 2010.

[84] K. Y. Leung, T. D. Barfoot, and H. H. Liu, "Decentralized cooperative SLAM for sparsely-communicating robot networks: A centralized-equivalent approach", *J Intell. and Robot Syst.*, vol. 66, no. 3, pp. 321–342, 2012.

[85] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization", *IEEE Trans. Robotics and Auto.*, vol. 18, no. 5, pp. 781–795, 2002.

[86] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays", *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[87] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters", *IEEE Conf. Decis. Control. Eur. Control Conf.*, pp. 8179–8184, 2005.

[88] G. Battistelli, L. Chisci, G. Mugnai, A. Farina, and A. Graziano, "Consensus-Based Linear and Nonlinear Filtering", *IEEE Trans. on Automatic Control*, vol. 60, no. 5, pp. 1410–1415, 2015.

[89] L. Li and M. Yang, "Joint localization based on split covariance intersection on the lie group", *IEEE Trans. Robotics*, vol. 37, no. 5, pp. 1508–1524, 2021.

[90] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized localization for multi-robot systems with asynchronous pairwise communication", *Intl Journal of Robotics Research*, vol. 37, no. 10, pp. 1152–1167, 2018.

[91] R. Jung, C. Brommer, and S. Weiss, "Decentralized Collaborative State Estimation for Aided Inertial Navigation", in *Intl. Conf. on Robotics and Automation*, Paris, 2020, pp. 4673–4679.

[92] E. Allak, R. Jung, and S. Weiss, "Covariance Pre-Integration for Delayed Measurements in Multi-Sensor Fusion", in *Intl. Conf. on Intelligent Robots and Systems*, Macau, 2019, pp. 6642–6649.

[93] E. Allak, A. Barrau, R. Jung, J. Steinbrener, and S. Weiss, "Centralized-Equivalent Pairwise Estimation with Asynchronous Communication Constraints for two Robots", in *Intl. Conf. on Intelligent Robots and Systems*, Kyoto, 2022, pp. 8544–8551.

[94] Z. Huai and G. Huang, "Robocentric visual–inertial odometry", *International Journal of Robotics Research*, 2019.

[95] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping using the Bayes tree", *International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[96] W. Zhao, J. Panerati, and A. P. Schoellig, "Learning-Based Bias Correction for Time Difference of Arrival Ultra-Wideband Localization of Resource-Constrained Mobile Robots", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3639–3646, 2021. arXiv: `2103.01885`.

[97] W. Liu *et al.*, "TLIO: Tight Learned Inertial Odometry", *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020. arXiv: `2007.01867`.

[98] D. L. Alspach and H. W. Sorenson, "Nonlinear bayesian estimation using gaussian sum approximations", *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, 1972.

[99] N. Boumal, B. Mishra, P.-A Absil, and R. Sepulchre, "Manopt, a Matlab Toolbox for Optimization on Manifolds", *Machine Learning Research*, vol. 15, pp. 1455–1459, 2014.

[100] GTSAM, *Math of GTSAM*, 2019. [Online]. Available: `https://github.com/borglab/gtsam` (visited on 09/03/2019).

[101] K. Röbenack, J. Winkler, and S. Wang, "LIEDRIVERS - A Toolbox for the Efficient Computation of Lie Derivatives Based on the Object-Oriented Algorithmic Differentiation Package ADOL-C", in *Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, ETH Zürich, 2011, pp. 57–66.

[102] H. Sommer, C. Pradalier, and P. T. Furgale, "Automatic Differentiation on Differentiable Manifolds", in *Robotics Research. Springer Tracts in Advanced Robotics*, vol. 114, 2016, pp. 505–520.

[103] J. Townsend and S. Weichwald, "Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation", *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1–5, 2016.

[104] H. Strasdat, *Sophus - Lie groups for 2D/3D Geometry*, 2019. [Online]. Available: `https://strasdat.github.io/Sophus/` (visited on 11/19/2019).

[105] S. Agarwal and K. Mierle, *Ceres Solver*. [Online]. Available: `http://ceres-solver.org/`.

[106] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso, "The Complex-Step Derivative Approximation", *ACM Trans. on Mathematical Software*, vol. 29, no. 3, pp. 245–262, 2003.

[107] W. Squire and G. Trapp, "Using Complex Variables to Estimate Derivatives of Real Functions", *SIAM Review*, vol. 40, no. 1, pp. 110–112, 1998.

[108] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups - Volume 2*. Birkhäuser Boston, 2009.