Vision-Based Navigation and Large-Scale Estimation for Spacecraft Swarms

Thesis by
Kai Matsuka

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended May 26th, 2023

ACKNOWLEDGEMENTS

I am thankful for all the people who supported me throughout my Ph.D.:

To my advisor, Dr. Soon-Jo Chung, for his unwavering support, encouragement, and mentorship. In my search for a deeper understanding of knowledge, Dr. Chung's insights and feedback introduced sufficient excitation at pivotal moments and helped me avoid getting stuck on a locally optimal solution.

To the rest of my thesis committee: Dr. Fred Hadaegh, Dr. Richard Murray, and Dr. Mike Watkins. Their insightful comments and discussions helped me reorient my research in a better direction.

To my funding agencies, such as National Science Foundation Graduate Research Fellowship Program, NASA Jet Propulsion Laboratory, Boeing, and Foster and Coco Stanback Space Innovation Fund.

To members of the Autonomous Robotics and Controls Laboratory, both past and present. To Paula Mark for being always very kind and helpful. To Benjamin Riviere, Dr. Michael O'Connell, Sorina Lupu, and Patrick Spieler for being incredible friends. To Dr. Rebecca Foust, Dr. Xichen Shi, Dr. Yashwanth Nakka, Dr. Vincenzo Capuano, and Dr. Kyunam Kim for mentoring me and having an impact on my early research career. To Dr. Hiroasu Tsukamoto, Dr. Guanya Shi, Dr. Karena Cai, Dr. Ellande Tang, Dr. SooJean Han, Jimmy Ragan, Dr. Lu Gan, Dr. Matt Anderson, Connor Lee, John Lathrop, Fengze Xie, Dr. James Preiss, Joshua Cho, Dr. Wolfgang Hoenig, Dr. Anthony Fragoso, Dr. Salar Rahili, Alexei Harvard, Francesca Baldini, and Thayjes Srivas for the wonderful memories of spending time together both inside and outside of the lab.

To the rest of the GALCIT cohort, Dr. Conor Martin, Dr. Kanthasamy Ubamanyu, Jean-Sebastian Sprat, Kira Headrick, Wesley Yu, Dr. Emile Oshima, Dr. Harsha Reddy, Dr. Eric Shen, Avinash Chandra, Lucas Benoit Marechal, Mouadh Bouayad, Dr. Akshay Joshi, and Alexander Wen. I greatly benefited from this group of incredibly smart, kind, and goofy friends since the first-year program at GALCIT. I cherish the sense of camaraderie that we built over the countless nights of working on p-sets.

To other members of the Caltech community. To Michele Judd for being simply an extraordinary person and my role model of a lifetime. All the out-of-this-world experiences at the Keck Institute of Space Studies that Michele produced enriched my Caltech experience in new dimensions. To Paula Mark, Martha Salcedo, and Christine Ramirez for being always very helpful with administrative help at GALCIT.

To all my friends including, but not limited to, Amanda Ho, Swyers Family, Mar Family, Keree McGuire, Jash Banker, Joeson Wong, Cai Tong Ng, and Summer-Hangover friends for being supportive in many aspects of my life.

To my family: To my mom and dad for their unconditional support. I would not be where I am today without their sacrifices, and this thesis is dedicated to them. To Shoun and Lime for always being an important part of my life. I cherish all the memories we have together including traveling and wedding prep. To Michelle and Alan Howard for being the best American family I could ask for and always inviting me back to visit their home.

Last but not least, to my wife, Dr. Michelle Cua, who is my constant source of joy and strength. I learn from her kindness, intellectual curiosity, and determination every day, and this helped me both in my professional and personal life. I also greatly benefited from her skills in writing critiques, cooking, and baking. Obtaining a Ph.D. will be the second-best thing that happened at Caltech after meeting her.

ABSTRACT

There has been dramatic growth in the space industry over the past 20 years. Around the same time, robotics and autonomy research has advanced significantly, resulting in a plethora of new mission concepts employing autonomy, such as on-orbit inspection, mission extension, space structure assembly, and orbital debris removal becoming within the realm of possibility. Two of the key autonomous technologies that are critical to the success of these missions are (1) advanced coordination of multi-agent systems and (2) robust vision-based navigation for on-orbit servicing in close proximity. However, there are challenges to simply applying the existing technology to space systems. First, there are domain-specific challenges that are unique to space, such as orbital mechanics and harsh lighting conditions. Second, even at a theoretical level, previous works in the controls and robotics literature are limited when applied to large-scale, locally coupled systems such as spacecraft swarms. To this end, this thesis develops novel algorithms for addressing these gaps.

In the first part of the thesis, we present a decentralized, scalable algorithm for swarm localization, called the Decentralized Pose Estimation (DPE) algorithm. With the DPE algorithm, each spacecraft computes relative navigation estimates with respect to others in the swarm but achieves higher performance through the benefit of multi-agent coordination. The DPE algorithm considers both communication and relative sensing graphs and defines an observable local formation. Each spacecraft jointly localizes its local subset of spacecraft using direct and communicated measurements. Since the algorithm is local, the algorithm complexity does not grow with the number of spacecraft in the swarm. As part of the DPE, we present the Swarm Reference Frame Estimation (SRFE) algorithm, a distributed consensus algorithm to co-estimate a common Local-Vertical, Local-Horizontal frame. The DPE combined with the SRFE provides a scalable, fully-decentralized navigation solution that improves the estimation accuracy compared to when without multi-agent coordination. Numerical simulations and experiments using Caltech's robotic spacecraft simulators are presented to validate the effectiveness and scalability of the DPE algorithm. We show that DPE has much higher accuracy than the best possible estimate without any coordination, while simultaneously being scalable to an arbitrarily large number of agents.

In the second part of the thesis, we propose a novel computer vision algorithm to track the pose of an unknown and uncooperative target using multiple decentralized observers. Vision-based pose determination of an unknown target is challenging due to factors such as lack of cooperative visual markers and harsh lighting conditions of space, and the problem is even harder for distributed observers. To address this challenge, we develop the algorithm called the Multi-Spacecraft Simultaneous Estimation of Pose and Shape algorithm or MSEPS. Within MSEPS, a team of chaser spacecraft, each equipped with a monocular camera, exchange information over a local network to jointly estimate the relative kinematic state of the target and its sparse shape landmarks. In this approach, each spacecraft processes its images and extracts its own set of visual keypoints in parallel. Then, the team uses the local network to jointly estimate the target pose and shape in a distributed fashion by applying the consensus algorithm over the inter-spacecraft communication links. To the best of the authors' knowledge, this is the first cooperative vision-based algorithm for estimating the pose and shape of a space object by means of an arbitrary number of spacecraft. We validate our algorithm using simulations of relative orbits and observations captured by each chaser spacecraft and show the multiple observers successfully agree on a consistent estimate and track the target pose accurately.

In the third part of the thesis, we develop some new simulation tools that bridge the gap between robotics and space technology. When developing robotics algorithms for on-orbit systems such as DPE and MSEPS, we identified a need for new simulation tools that tightly integrate robotics algorithms with high-fidelity models of space environments such as astrodynamics effects and visual conditions. To this end, we first develop a ROS2-compatible software interface for Basilisk, the open-source astrodynamics simulation software. This tool allows running Basilisk in parallel with ROS2 in real-time and translates messages between Basilisk modules and ROS2 modules, such that control algorithms implemented in ROS2 can interact with the high-fidelity dynamics within Basilisk in a closed-loop fashion. Second, we develop a ROS2-compatible camera simulation module that uses the Neural Radiance Fields (NeRF) to rapidly generate novel images. These synthetic images are used as inputs to validate the vision-based navigation algorithm in a closed-loop fashion. To validate these simulation tools, we also developed a set of autonomous algorithms for on-orbit inspection and use the simulated measurements as inputs to the algorithm. The real-time numerical simulations demonstrate that our tools can be integrated with autonomy algorithms implemented in ROS2 in a closed-loop fashion to validate the feasibility of the mission.

In the process of addressing some lessons learned from DPE and MSEPS works, we identified that there is a gap in general frameworks for solving the optimal estimation problems for probabilistic inference of large-scale problems involving networked systems. This gap is not just applicable to spacecraft swarms, but also to a general class of large-scale, multi-agent problems in robotics and controls such as localization and mapping, wireless sensor networks, and electrical power grids. Therefore, in the fourth part of the thesis, we address this fundamental gap by developing novel algorithms for Distributed Factor Graph Optimization (DFGO) problems that arise in large-scale networked systems. We develop algorithms for both batch and real-time problems. First, for the batch DFGO problem, we derive a type of the Alternating Direction Method of Multipliers (ADMM) algorithm called the Local Consensus ADMM (LC-ADMM). LC-ADMM is fully localized; therefore, the computational effort, communication bandwidth, and memory for each agent scale like $o(1)$ with respect to the network size. We establish two new theoretical results for LC-ADMM: (1) exponential convergence when the objective is strongly convex and has a Lipschitz continuous subdifferential, and (2) $o(1/k)$ when the objective is convex and has a unique solution. We also show that LC-ADMM allows the use of non-quadratic loss functions, such as ℓ_1-norm and Huber loss. Second, we also develop the Incremental DFGO algorithm (iDFGO) for real-time problems by combining the ideas from LC-ADMM and the Bayes tree. To derive a time-scalable algorithm, we exploit the temporal sparsity of the real-time factor graph and the convergence of the augmented factors of LC-ADMM. The iDFGO algorithm incrementally recomputes estimates when new factors are added to the graph and is scalable with respect to both network size and time. We validate LC-ADMM and iDFGO in simulations with examples from multi-agent Simultaneous Localization and Mapping (SLAM) and power grids.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] K. Matsuka and S.-J. Chung, "Localized and incremental probabilistic inference for large-scale networked dynamical systems," *IEEE Transactions on Robotics (Conditionally accepted for publication)*, 2023,
K.M. conceived the project, constructed the theoretical analysis, developed and implemented algorithms and simulations, generated figures, and wrote the manuscript.

[2] K. Matsuka, C. Ohenzuwa, and S.-J. Chung, "Rapid synthetic image generation using neural radiance fields for vision-based formation flying spacecraft," in *2023 33rd AAS/AIAA Space Flight Mechanics Meeting*, AIAA, 2023,
K.M. conceived and led the project. K.M. participated in coding, running simulations, and writing the manuscript.

[3] K. Matsuka, L. Zhang, I. Ragheb, C. Ohenzuwa, and S.-J. Chung, "High-fidelity, closed-loop simulation of spacecraft vision-based relative navigation in ros2," in *2023 33rd AAS/AIAA Space Flight Mechanics Meeting*, AIAA, 2023,
K.M. conceived and led the project. K.M. participated in coding, running simulations, and writing the manuscript.

[4] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, "Decentralized formation pose estimation for spacecraft swarms," *Advances in Space Research*, vol. 67, no. 11, pp. 3527–3545, 2021. DOI: `10.1016/j.asr.2020.06.016`,
K.M. conceived the project, implemented algorithms, performed theoretical analysis, and implemented simulations. K.M. participated in the experiments and in writing the manuscripts.

[5] K. Matsuka, A. Santamaria-Navarro, V. Capuano, A. Harvard, A. Rahmani, and S.-J. Chung, "Collaborative pose estimation of an unknown target using multiple spacecraft," in *2021 IEEE Aerospace Conference*, IEEE, 2021, pp. 1–11. DOI: `10.1109/AERO50100.2021.9438352`,
K.M. led the project and developed the algorithms. K.M. participated in the technical discussion of project ideas, coding, running simulations, generating figures, and writing the manuscript.

[6] K. Matsuka, E. S. Lupu, Y. K. Nakka, R. Foust, S.-J. Chung, and F. Hadaegh, "Distributed multi-target relative pose estimation for cooperative spacecraft swarm," in *Proc. 10th International Workshop on Satellite Constellations and Formation Flying*, 2019,
K.M. conceived the project idea and developed the algorithms. K.M. participated in the experiments and in writing the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

In recent years, there has been dramatic growth in the commercial space industry. As the interest in space exploration increases, so does the demand for basic infrastructures in space to support such activities. These infrastructures include on-orbit inspection, spacecraft refueling, on-orbit construction and assembly of structures, space debris removal, and formation flying spacecraft. To enable such missions, autonomy and robotics play an increasingly important role as they can perform time-critical collision safety maneuvers during proximity operations, keep operation costs low, and scale to a large number of spacecraft.

While robotics has been applied to some space applications in the past, their usage in orbit has been much more limited. Many state-of-the-art missions rely on monolithic systems with expensive sensor suits and largely required humans in the loop. As we aim to miniaturize the spacecraft and reduce cost while also improving performance, more advanced autonomy is necessary. Advanced autonomy algorithms will also enable missions involving a large number of relatively inexpensive spacecraft, and thereby expanding the capabilities of new mission concepts.

Among the various technology gaps, there are two robotics technologies that are crucial to advancing the level of autonomy for on-orbit infrastructures. The first is *scalable, multi-agent coordination* in orbit. Multi-agent coordination refers to multiple agents collaboratively solving a joint problem. Such capabilities are relevant when a team of spacecraft aims to achieve estimation and controls performance not possible by operating each agent independently. While some existing missions such as Starlink have demonstrated the power of a large number of spacecraft, the controls and estimation problems of each agent in the constellation are largely solved independently. By enabling real-time and onboard autonomous coordination with a swarm of systems, one can deploy a large number of systems for more time-critical operations at a much closer distance with increased performance. Scalable multi-agent coordination will exploit the advantages of the swarm to its fullest potential and meet ever more demanding mission requirements.

The second technology that plays an important role in on-orbit space infrastructure is *vision-based navigation for proximity operations*. This technology is relevant for tasks such as on-orbit inspection, servicing, and debris removal. The information-rich nature of vision-based systems requires a fundamentally different set of algorithms compared to traditional spacecraft navigation systems, which rely on star trackers and GPS. While there is a substantial amount of advancement in computer vision research for terrestrial applications, there are several domain-specific challenges that make it difficult to apply them directly to space systems. For example, harsh lighting conditions make robust vision-based navigation challenging. In space, one must also consider relative orbital dynamics and attitude dynamics of the target with unknown inertia. Algorithm development is further hampered by the lack of publicly available datasets and simulation tools for testing and validation. Moreover, at the theoretical level, algorithms in robotics and control literature do not fully address the challenge of optimal estimation in a manner that scales well with swarm size. These technology gaps for on-orbit space infrastructure and robotics motivated the various research projects presented in this thesis.

1.2 Problem Statements

This thesis aims to address the following problems of using autonomous capabilities in space.

Scalable Relative Navigation of Spacecraft Swarm

Suppose a large number of spacecraft is formation-flying in a planetary orbit. In the presence of relative measurements and inter-spacecraft communications, the estimation accuracy of the spacecraft swarm may improve through coordination. The challenge is designing a distributed and localized estimation algorithm that has improved performance (e.g., in terms of estimation accuracy and situational awareness) through information sharing among agents while also ensuring that the algorithm is scalable to a large number of spacecraft.

Vision-based Pose Tracking of An Unknown, Uncooperative Target Using Multiple Spacecraft Observers

A team of multiple observer spacecraft uses computer vision to track an uncooperative and unknown target. "Uncooperative" targets do not communicate with the other spacecraft or feature any aid such as visual markers. "Unknown" is used to mean that the key information about the target such as geometry, appearance, or mass properties are not known *a priori*. For example, one can consider the application of a team of spacecraft tasked with visually inspecting and removing space debris. The challenge is to develop a distributed algorithm that computes an optimal solution while ensuring the consistency of estimates between spacecraft.

High Fidelity Simulation Tools for Robotics in Orbit

There is a lack of simulation tools for developing and testing autonomy algorithms for orbital space applications, such as vision-based navigation for on-orbit servicing. We need numerical simulation tools that bridge the gap between space systems and robotics. For this problem, we tackle the challenge of developing fast and accurate tools that can (1) model the motion of orbiting spacecraft, (2) render simulated images from onboard sensors, and (3) interface with other robotics algorithms implemented in the standardized robotics middleware Robot Operation System 2. Such simulation tools are essential for testing the autonomous algorithms for on-orbit servicing in real-time and closed-loop without requiring access to space or expensive experimental setups.

Swarm-Scalable, Time-Scalable Optimal Estimation Algorithm for Networked Systems

For this problem, we consider the optimal estimation problems for probabilistic inference of large-scale problems involving networked systems. The scope of this problem is not just spacecraft swarms, but also a class of general large-scale, locally-coupled networked systems, such as multi-agent localization and mapping, wireless sensor networks, and electrical power grids. The goal is to develop an estimation algorithm that computes the optimal solution to the centralized probabilistic inference problem in a distributed and localized fashion. The algorithm shall be scalable with both the number of agents and the length of the time horizon.

1.3 Thesis Contributions

This thesis presents our work towards advancing the multi-agent coordination and vision-based navigation of formation-flying spacecraft swarms as well as a more general class of large-scale networked systems. We briefly summarize the major contributions of each research project in this section.

Chapter 3

This chapter presents a novel algorithm for the relative positioning of a spacecraft swarm, referred to as the Decentralized Pose Estimation (DPE) algorithm. Exploiting multi-agent coordination with local neighbors, DPE improves estimation accuracy, observability, and situational awareness compared to independent systems.

Each spacecraft estimates the states of only a local subset of spacecraft; therefore, the algorithm complexity on each spacecraft does not grow with the swarm size. Based on the *ad hoc* relative sensing and communication graphs, the DPE uses the observability criteria to determine the local subset of observable spacecraft. To define a common frame for the spacecraft in DPE, we also develop the Swarm Reference Frame Estimation (SRFE) algorithm, which estimates a common Local Vertical Local Horizontal coordinate system using a distributed information consensus filter. Using these techniques, DPE is able to simultaneously achieve high estimation accuracy and scalability for spacecraft swarm estimation problems. We validate the scalability of the DPE and SRFE algorithms in numerical simulations and in a first-of-a-kind hardware experiment involving 3 spacecraft simulators, each of which is equipped with a camera.

Chapter 4

This chapter describes our work on Multi-spacecraft Simultaneous Estimation of Pose and Shape or MSEPS. MSEPS addresses the vision-based pose tracking of an unknown, uncooperative target using multiple spacecraft observers. To the best of the authors' knowledge, MSEPS is the first algorithm aimed to compute the centralized optimal estimate in a fully decentralized fashion. The goal of MSEPS is to estimate both the attitude and center of gravity (CG) of the target object using vision sensors on multiple chaser spacecraft. We propose a computer vision pipeline that is suitable for tracking a target without visual fiducial markers or any *a priori* model of geometry or appearance of the target. The use of multiple observers provides a virtual stereo configuration. For the back-end estimation problem of MSEPS, we develop a distributed algorithm based on the extended decentralized information filter. The back-end algorithm computes the approximate solution of a centralized minimum variance estimation problem, except computation is distributed and each spacecraft only communicate with neighbors. We validate the algorithm architecture through numerical simulations of relative orbits, measurements, and inter-spacecraft communications between multiple spacecraft.

Chapter 5

To validate robotics algorithms, capabilities such as modeling spacecraft motion and sensor data are crucial. In this chapter, we address the lack of simulation tools for space environments by developing two new tools: ROS-Basilisk and ROS-NeRF. ROS-Basilisk is a lightweight software that interfaces between ROS2 and the open-source astrodynamics simulation software Basilisk. This enables robotics algorithms to use the simulated dynamics from Basilisk and allows Basilisk to reflect the control commands from the ROS2 algorithm in a real-time fashion. The ROS-NeRF module provides a method to simulate the sensor output from on-board cameras. Since ROS-NeRF relies on neural networks trained on one specific scene, the rendering is much faster than traditional methods, making this module suitable for real-time, closed-loop, or Monte Carlo simulations. For space applications, the relative motion of the target with respect to light sources causes varying lighting directions and must be accounted for. Together, these ROS-Basilisk and ROS-NeRF allow us to validate autonomy and robotics algorithms for space without requiring external hardware. We demonstrate their utility in an example mission of autonomous proximity operations that require vision-based relative navigation, attitude, and formation control. We performed numerical experiments to validate both simulation tools and the autonomy algorithms and to test all the components in a real-time and closed-loop simulation.

Chapter 6

This chapter tackles the challenge of developing an optimal estimation algorithm for networked dynamical systems that scale with swarm size and time. We propose a new approach where we formulate the estimation problem as Distributed Factor Graph Optimization (DFGO) and then solve it using the Local Consensus ADMM (LC-ADMM). In addition to scalability with respect to the number of agents, LC-ADMM has various advantages. The algorithm can naturally incorporate robust loss functions such as ℓ_1 and Huber losses; agents do not need to know information about global graph topology; and there is an intuitive interpretation of the LC-ADMM algorithm steps in terms of factor graphs.

Using LC-ADMM as a backbone, we develop the Incremental Distributed Factor Graph Optimization (iDFGO) algorithm for real-time problems. The iDFGO algorithm can incrementally recompute a subset of the local problem rather than solving the optimization over the whole trajectory. The iDFGO algorithm is scalable both in network size and time.

We show two new theoretical results on the convergence rate of LC-ADMM. The first theorem shows $o(1/k)$ convergence when the objective is convex and has a unique solution. The second theorem shows LC-ADMM converges exponentially when the objective function is strongly convex and has a Lipschitz continuous subgradient.

These convergence rates are shown to hold even in the presence of additional local affine equality constraints. Finally, we perform numerical validations of LC-ADMM and iDFGO using examples from power grid monitoring and multi-agent PGO. To the best of the author's knowledge, a unified formulation of the distributed outlier rejection problem for a networked dynamical system with more than a hundred agents is demonstrated for the first time. We also empirically validate LC-ADMM and iDFGO for multi-agent PGO problems using a benchmark data set and compare our results against the state-of-the-art distributed PGO algorithm.

1.4 Related Works

We now discuss the previous works on multi-agent systems in the literature that are particularly relevant to this thesis. Multi-agent systems have the potential to be robust against loss and improved science return [4], [5]. Algorithms for multi-agent systems have been studied in different contexts such as controls, robotics, and space systems literature. Here, we review prior work on distributed estimation algorithms for large-scale systems and the role of autonomy in space applications.

Distributed Estimation

An estimation problem deals with determining the state of a system given a set of observations. While there are multiple ways to classify various types of estimation algorithms in the literature, the two classification schemes that are relevant to this thesis are centralized vs. distributed, and small- vs. large-scale. These classifications can be visualized in Fig. 1.1. The estimation algorithm is centralized when a single agent has access to all the measurements in the system, and it is primarily responsible for the computation. In contrast, distributed problems involve multiple observers that collectively estimate the states, and there is no single agent that is coordinating the collective effort. Small- and large-scale refers to the algorithm's scalability with respect to the number of state variables to be estimated. In large-scale problems, the number of variables typically grows with some system parameters, such as spatial coverage, the number of objects of interest, and time. Large-scale estimation algorithms explicitly address some specific scalability issue(s) of the problem and ensure that computation, memory storage, and information exchange (if any) are tractable. In contrast, small-scale estimation algorithms are those that do not (need to) make explicit considerations. In this thesis, we are interested in distributed estimation problems.

The early works in distributed estimation literature addressed the problem for small-

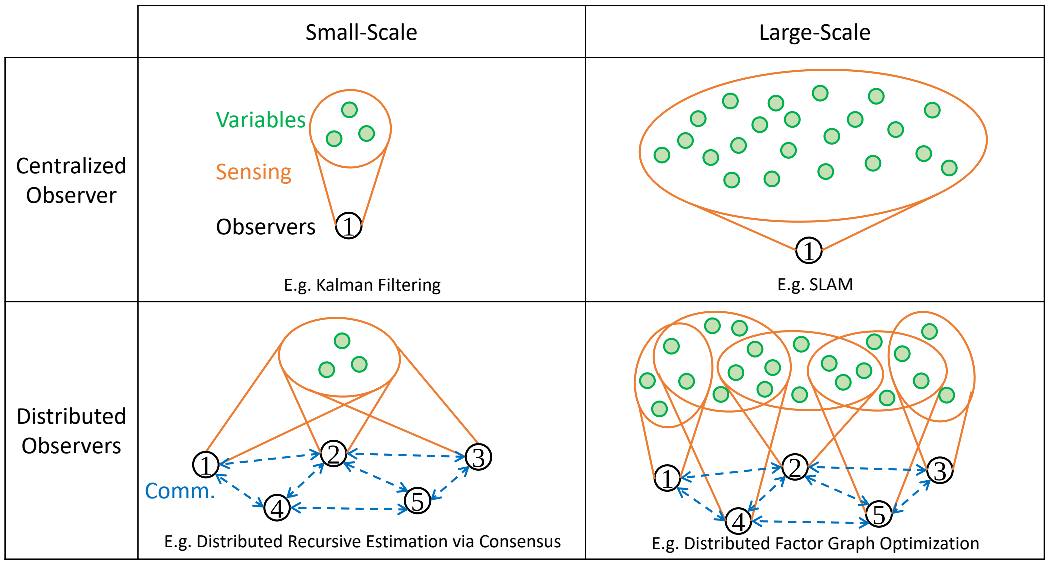


Figure 1.1: Classification of observer design problems. The two rows differentiate single observer vs. distributed problems. The two columns differentiate small- vs. large-scale problems.

scale systems [6]–[8]. Some authors developed the distributed minimum variance estimate for LTI systems [9] while others generalized the solution to nonlinear dynamical systems with non-Gaussian distribution [10]. However, while these algorithms work well for small-scale systems, they do not extend well to large-scale. One of the primary reasons is that these works employ a recursive algorithm approach, and the prior distribution is densely correlated in the recursive formulation in general. Due to this dense correlation, a pair of agents that are geospatially far from each other need to collaboratively compute their cross terms, and this becomes intractable for large-scale networks. For large-scale systems with an n-dimensional state vector, each agent needs to (1) have a vector of size n; (2) communicate an $n \times n$ matrix for the covariance inverse (or sometimes more for non-Gaussian distribution [10]); and (3) run a computation that scales with $O(n^3)$ at each time step [11]. Therefore, as the network size increases, the necessary computation and communication of recursive estimators become intractable.

Distributed and Large-Scale Estimation Problems

A smaller number of prior works in the literature aim to address the large-scale distributed observer problem, which is the bottom right category in Fig. 1.1. Large-scale problems are common in a wide variety of engineering applications, such as localization of wireless sensor networks [12], [13], tracking of electrical power grids [14], swarm robotics [15]–[17], and multi-agent Simultaneous Localization and Mapping (SLAM) [3], [18]. In distributed and large-scale problems, observers are often distributed spatially and observe a part of the overall system. The main paradigm for addressing this class of problems is to accomplish *local decomposition* such that each agent's algorithm is localized. That is to say, each agent only tracks a small subset of variables that are in proximity such that local computation, communication, and memory are all scalable with network size.

There are some works in the control literature that address this class of problem. Authors of [11] take a recursive estimation approach and developed a localized algorithm that approximates the optimal solution through iterative information exchange between neighbors. This algorithm exploits the L-banded structure of sparsely coupled information matrices to circumvent the need to directly compute the dense covariance matrix. However, this approach is specialized to linear Gaussian systems and also requires computing a good ordering of variables based on the global network topology so that one can compute a "good" L-banded information matrix. The need to have sorted variables is a relatively strong assumption and the algorithm is impractical for swarm robotics where the network topology might not be known *a priori* or could change over time.

Instead of recursive algorithms, an alternative approach is to estimate the trajectories of dynamical systems in a batch over some time horizon. Among the works in the controls literature, the line of works by System-Level Synthesis (SLS) [19] is notable for large-scale networked systems. SLS was originally motivated by optimal control of large-scale networked systems, but they extend to the estimation problems[20] which is a dual of the controls problem. SLS decomposes the generally dense, optimal LQR gains K into a product of two sparse and localized matrices Φ_u, Φ_x such that $K = \Phi_u \Phi_x^{-1}$. SLS achieves the local decomposition by first computing the solution in a batch over a time horizon, which preserves the sparsity of the network graph. However, the assumption of linear systems is baked into the derivation of SLS algorithms, and it is currently unclear how well the framework extends to general nonlinear systems. Thus, there exists a gap for distributed estimation for large-scale systems that can handle nonlinear systems or non-gaussian noise that arise in robotics problems.

Single-Agent and Multi-Agent SLAM

In parallel to the development of distributed algorithms for large-scale systems in control literature, there was also a development of estimation for large-scale problems in robotics literature. However, its motivation was slightly different from scalability to an arbitrary number of locally coupled agents. Instead, the works in robotics literature were primarily motivated to address the curse of dimensionality that arose from Simultaneous Localization And Mapping (SLAM) problems. SLAM involves estimating the state of a robot equipped with sensors while simultaneously also constructing the environment that the sensors are perceiving [21]. Since the size of the map grows as the robot traverses the environment, the problem becomes large-scale even for the single-agent problem. This is the top right category of Fig. 1.1. Thus, the challenge is to develop a scalable algorithm whose complexity does not grow with the size.

To address the curse of dimensionality, an efficient algorithm typically exploits the sparsity that arises from the spatial structure of the mapping problem [22]. In particular, factor graphs have become a key mathematical tool for exploiting this sparsity [2], [3], [23]–[25]. Factor graphs are a family of graphical models that represent the joint probability distribution, and they effectively model the spatial sparsity structure necessary for localized algorithms [26]. Factor Graph Optimization (FGO) is the problem of computing the Maximum A Posteriori (MAP) estimate of the joint probability distribution that is modeled as a factor graph. There have been substantial theoretical advancements to single-agent FGO. For example, iSAM2 [23] addresses the time-scalability issue[1] of the batch optimization approach by using incremental factor graph update by reformulating the original factor graph to a Bayes Tree. SE-Sync [2] solves a subset of FGO called Pose Graph Optimization (PGO). In PGO, all the decision variables are SE(d) and the observations consist of relative poses. SE-Sync addressed the challenges of the non-convexity of SLAM problems by developing a specialized algorithm that solves PGO with a certain global convergence guarantee. There exist well-established FGO software such as GTSAM [26] and g2o [27] to solve FGO efficiently.

There has also been some work on FGO involving multiple agents [3], [18], [25]. We refer to the problem of solving multi-agent FGO in a distributed fashion as Dis-

[1]Use of batch optimization approach, instead of recursive formulation like Kalman filter, enabled scalability to problems with a large dimensionality. However, it also re-introduced the issue of *time*-scalability. With the exponential advancement of onboard computing capabilities, the computation and memory requirements of batch algorithms have become much more tolerable for real systems.

tributed Factor Graph Optimization or DFGO. The algorithm in [25] uses distributed successive over-relaxation on a quadratic approximation of FGO. The DDF-SAM 2.0 algorithm in [18] exchanges "summarized graph" of each agent with neighbors. The DC2-PGO algorithm in [3] extends SE-Sync [2] to the multi-agent system and solves the pose graph optimization in a localized fashion with global convergence. Riemannian Block Coordinate Descent in [3] is a localized algorithm that provides a convergence guarantee to the first-order optimal solution. All of these algorithms feature local decomposition. However, these algorithms also have some limitations. For example, [25] assumes linearization and Gaussian noise. Computing the summarized graph in DDF-SAM 2.0 [18] is relatively expensive and it uses conservative approximations to make the problem tractable. The computation also requires "anti-factors" to negate the effect of factors from the previous epoch. DC2-PGO [3] is specialized to pose graph optimization. And RBCD [3] is not parallel and it is only fast enough or swarm-scalable with its "accelerated" version that requires solving the global graph coloring problem *a priori*. While there exist incremental methods for solving FGO for single-agent scenarios (e.g., iSAM2 [23]), there are no efficient incremental methods for DFGO[2]. In summary, the design of efficient algorithms to solve DFGO suitable for large-scale robotic swarms is still an active area of research.

In the context of these previous works on FGO, our work in Chapter 6 can be viewed as a new approach to solving the DFGO in batch. Our algorithm is also fully localized similarly to [3], [18], but additionally, *all* the agents run the algorithm in parallel. Our approach does not require *a prior* knowledge of the global graph topology so it is suitable for *ad hoc* networks. We also develop an incremental version of the algorithm to solve DFGO that also scales with time, in addition to network size.

Finally, we can make some connections between robotics and control literature regarding their respective approaches to solving multi-agent, large-scale problems. For example, algorithms to solve DFGO are not only relevant to robotics literature (e.g., multi-agent SLAM) but they have the potential to be applicable to the estimation problems of locally coupled systems involving a large number of agents that were previously studied in controls literature [11], [20]. To the best of the author's knowledge, this connection between DFGO and optimal estimation for large-scale systems was not widely known previously in the literature. With this connection

[2]The authors of DDF-SAM2 [18] briefly mention that their local FGO problems can be solved using iSAM2 on each agent; however, in general, simply combining an iSAM2 with a distributed algorithm does not actually scale well in time overall, as we discuss in Chapter 6.

in mind, our development of a novel algorithm to solve DFGO is partly motivated by its application to networked systems with an arbitrary number of agents, and we ensure every aspect of the algorithm is scalable to the network size. Another connection is that most of the approaches that resulted in local decomposition (i.e., SLS of controls literature and DFGO variants of robotics literature) also share the similarity in formulating the estimation of a dynamical system in batch over a certain time horizon. The batch approach preserves the spatial sparsity structure which is a key ingredient for deriving a localized algorithm. In summary, these connections between controls and robotics literature provided additional insights to better understand the problems addressed in this thesis.

Alternating Direction Method of Multipliers

In Chapter 6, we present a method of solving the DFGO using the Alternating Direction Method of Multipliers (ADMM) algorithm. In distributed optimization problems, agents collectively compute the solution to a coupled objective function in a distributed fashion[3], [28]–[31]. While various algorithms exist in the literature [32]–[35], ADMM has become a popular tool as it converges for a broad class of problems [36]–[38]. In particular, the Decentralized Consensus ADMM algorithm (DC-ADMM) [37] has been shown to be applicable to multi-agent systems [39]. However, DC-ADMM assumes that all the agents have copies of the same state vector and is therefore not scalable with the size of the state vector.

Other works [40]–[42] extended DC-ADMM to localized settings where each agent's objective only depends on a small subset of decision variables. These works were scalable to large-scale problems but had some limitations. The algorithm in [40] required the graph coloring to be known. The Separable Optimization Variable ADMM (SOVA) algorithm [41], [42] removed the coloring requirement of [40], making its application to multi-agent robotics much more suitable. However, the theoretical convergence rate guarantees were limited to asymptotic convergence [43]. Because SOVA is an ADMM of decentralized consensus type, many of the existing results in the other ADMM literature do not apply directly [37]. Therefore the convergence rates and conditions under which these algorithms converge remained unknown. Some works applied ADMM to solve real-time problems by [39].

Our work on Localized Consensus ADMM (LC-ADMM) extends the results from DC-ADMM [37] and SOVA [41], [42] in two ways. First, we establish the new theoretical results convergence rate of LC-ADMM under two different sets of assumptions. We show LC-ADMM converges to the optimal solution at $o(1/k)$ rate when the objective function is convex and the problem has a unique solution (Theorem 4) and at an exponential rate when the objective is strongly convex and has a Lipschitz continuous subdifferential (Theorem 5). In both cases, the objective may be non-differentiable and the same convergence is given in the presence of local affine equality constraints. Theorem 5 is a generalization of the result of [37] to localized, non-differentiable, and affine equality-constrained settings and both Theorems 4 and 5 provide, with additional assumptions, faster rates than the asymptotic convergence given in [41]. Second, our work also provides new perspectives on LC-ADMM in the context of DFGO. During the LC-ADMM step in which each agent solves its local optimization in parallel, the local optimization problem has an intuitive interpretation as FGO. This perspective enables the development of efficient algorithms using tools from the FGO literature.

The Role of Autonomy for Orbital Space Systems

Finally, we review the role of robotics and autonomy for space systems in orbits. Recent space missions demonstrated increasingly advanced capabilities for autonomous on-orbit servicing and formation flying. For example, Orbital Express by DARPA and NASA demonstrated autonomous docking with a demo target spacecraft [44]. Years later, Mission Extension Vehicle by Northrop Grumman performed docking with a client spacecraft that is an actual commercial vehicle [45]. Other missions performed satellite-to-satellite inspection (PRISMA by ESA [46], AeroCube10 by Aerospace Corporation [47]) and proposed to perform debris removal (ELSA-d by Astroscale [48]). For these on-orbit servicing missions, vision-based navigation in proximity is critical to estimating the relative state of targets, a piece of information needed to perform precise and collision-free operations. Another type of orbital mission that requires advanced autonomy is formation flying. For example, some mission concepts use a team of formation-flying spacecraft to perform radar-based Earth observation science such as forest and cloud tomogrpahy [49], [50]. In formation flying, the control strategy of multiple spacecraft is coupled through a common control law[51], [52]. As the number of agents in a team of formation-flying spacecraft increase, the scalability of many of the existing algorithms for a coupled team of spacecraft becomes intractable quickly.

The following sections discuss the related works in these two key space robotics technologies needed to enable the advanced mission concepts: vision-based navigation and scalable estimation strategy for large-scale spacecraft swarms.

Vision-Based Navigation

One of the key technology for on-orbit servicing is a vision-based navigation algorithm for tracking the pose of the target space objects in proximity. Compared to LiDAR-based methods, the cameras have lower mass, cost, and power consumption, which makes them suitable relative navigation sensors for small spacecraft. However, vision-based navigation in space also comes with some challenges, such as harsh lighting conditions, reflective appearance, repetitive patterns of artificial satellites, and a wide range of focal distances. The properties of the target can also impact the pose estimation problem.

The complexity of the vision-based pose estimation problem depends on whether the target is *cooperative* or *uncooperative* and *known* or *unknown*. Pose estimation is easier when the target is cooperative or known. Cooperative space objects may have visual fiducial markers or be able to communicate their own information [53]–[55]. Pose estimation of *uncooperative but known* objects assumes that some critical information about the target is known, such as geometry, appearance, and mass properties. Examples of this category of algorithms include [56], [57]. In contrast, for uncooperative and unknown targets, there is no *a priori* information about the object, and the target does not communicate with the algorithm. As a result, vision-based pose estimation of *uncooperative and unknown* is much more challenging.

Vision-based pose estimation algorithms for uncooperative and unknown targets rely on model-free perception techniques such as SLAM or Structure from Motion (SfM) to infer the geometry of the target. While SLAM and SfM in terrestrial applications share similarities with vision-based pose estimation for spacecraft, there are some unique challenges for orbital systems. First, target spacecraft attitude dynamics are difficult to estimate and predict. On terrestrial SLAM, the mapped landmarks are directly attached to the inertial frame; however, on pose tracking, the landmarks on the target are attached to the target frame, which is rotating with respect to an inertial frame. Target also may have an unknown inertia matrix or active control which may need to be taken into account when formulating the pose tracking problem. Second, the visual conditions experienced in space are harsh compared to conditions commonly encountered in terrestrial SLAM problems. Adversarial visual conditions include high contrast, reflections, and repetitive features. The vision-based pose estimation algorithm for space systems must address these challenges.

Some previous work in the literature aimed to address the pose tracking of unknown targets, but they were primarily focused on using a single observer. In [58], authors developed an algorithm that uses SLAM for mapping and estimation of the pose, the center of gravity (CG), and the inertia properties of an unknown uncooperative space target and conducted experiments inside the International Space Station. The problem of estimating the CG and inertia properties and mapping was also investigated in [59] and [60]. Others presented a real-time algorithm for pose estimation based on monocular SfM [61], where a Bayesian filter is adopted to estimate the relative rotational dynamics. Some investigated the problem of feature extraction [62] and matching [63]. Some developed Simultaneous Estimation of Pose and Shape or SEPS proposed a pipeline for vision-based navigation using optical flow, and sequential filtering. Some developed SLAM-based terrain relative navigation algorithms with respect to asteroids [64]. While these works in the literature have certainly advanced the state-of-the-art of vision-based tracking by a single observer, similar research using multiple observers has been limited. Vision-based tracking of unknown targets using multiple observers can address some fundamental limitations of single-observer problems such as persistent tracking under harsh lighting conditions and harsh lighting conditions. To this end, Chapter 4 of this thesis discuss a multi-observer approach to vision-based pose tracking of an unknown target.

When evaluating the applicability of some relative navigation algorithms using proximity operation sensors, an important aspect to consider is the validation of the algorithm in a realistic, space-like environment. While there are numerous algorithms proposed in the literature, the actual number of flight missions that demonstrated these algorithms in orbit has been limited. Orbital Express demonstrated autonomous docking and undocking with a free-flying target spacecraft using a robot arm using visual servo [44]. Mission Extension Vehicle also demonstrated docking with commercial spacecraft in GEO [45], but it involved a monolithic servicing vehicle, with a collection of high-cost sensors, and a known target. Some robotic experiments such as SPHERES VERTIGO [65] and Astrobee [66] tested model-free vision-based navigation within the International Space Station (ISS), but the illumination condition inside the ISS is not an accurate representation of the harsh visual conditions of targets free-flying in orbit. Several other missions such as PRISMA [46] and AeroCube10 [47] have taken satellite-to-satellite images with sufficient proximity to possibly resolve poses, but these missions did not use the vision-based pose estimate for controlling the relative motion. Instead of flight experiments, many vision-based pose estimation algorithms proposed in the literature typically rely on validation through numerical simulations [56] or hardware experiments on the ground [64]. In Chapter 5 of this thesis, we discuss some new numerical simulation tools for validating vision-based navigation algorithms in a real-time, closed-loop fashion.

Spacecraft Swarm

Another key technology is the scalable distributed algorithms for formation flying and spacecraft swarms. A spacecraft swarm is a collection of a large number of spacecraft performing some cooperative task [67], [68]. Spacecraft swarms have the potential to revolutionize the space industry by enabling missions such as distributed aperture telescopes, space structure assemblies, and cooperative deep space explorations [69]–[71]. The mission concepts using spacecraft swarms have several advantages over monolithic satellite missions, such as robustness to individual spacecraft loss and improved science return [4], [5]. However, since it is operationally prohibitive to control all the agents in the swarm with ground-in-the-loop control, the spacecraft swarm must operate largely autonomously. Moreover, in many of the mission concepts, spacecraft swarms must fly in close proximity and in specific relative orbits, requiring them to perform controls for formation flying [51], [52] which require accurate localization knowledge. This motivates the development of accurate relative navigation strategies for spacecraft swarms such as the scalable distributed estimation algorithm for localization of spacecraft swarms discussed in Chapter 3.

ROS2 for Robotics in Space

There has been an increased interest in using the Robotics Operating System 2 (ROS2) as a middleware suit for deploying autonomous algorithms in space. ROS2 has a real-time and distributed architecture and it has been widely adopted as a standard in the robotics industry [72]. ROS/ROS2 has been implemented in a select few applications such as a lunar rover (VIPER [72]), flying robotic experiment platforms inside ISS (Astrobee [66]), a humanoid robot at ISS (Spacenaut [73]). The highly modular design, extensive features for real-time systems, and distributed communication protocols of ROS2 are all well-suited for onboard, real-time autonomous operations in orbit, even including multiple cooperative spacecraft. However, while the use of ROS2 for planetary rovers or ISS has become a reality, the use of ROS/ROS2 in orbit on free-flying spacecraft has been limited. One of the main challenges with developing autonomy in orbit using ROS2 is that there were previously no publicly accessible simulation tools such as ROS2-compatible astrodynamics simulation or highly realistic image rendering tools for proximity operation. In Chapter 5, we develop new simulation tools specifically for on-orbit servicing to bridge this gap.

1.5 Thesis Organization

The rest of this thesis is organized as follows. In this Chapter, we described the motivation, problem statements, and summary of contributions of this thesis. Chapter 2 continues by introducing preliminary concepts and mathematical notation that will be relevant for understanding the main contributions of this thesis. In Chapters 3 through 6, we discuss each of the thesis contributions in more detail. Finally, Chapter 7, provides concluding remarks.

Chapter 2

PRELIMINARIES

In this chapter, we introduce general concepts and notations that will be used throughout the thesis. First, we introduce the definitions from graph theory, which is used to describe locally coupled dynamical systems and factor graphs.

2.1 Graph Theory

In many of the chapters, graphs are used to model the network of agents. Suppose $\mathcal{V} = \{1, \dots, N\}$ is the set of N vertices in the network. The set of undirected edges, denoted as $\mathcal{E} \subseteq \{(i,j) \,|\, i,j \in \mathcal{V},\, i \neq j\}$, is defined such that $(i,j) \in \mathcal{E}$ if and only if the i-th and the j-th vertices are connected. An undirected graph is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of neighbors of the i-th vertex, $i \in \mathcal{V}$, is given by $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$. The neighborhood, or the set of inclusive neighbors, of the i-th vertex is defined as $\bar{\mathcal{N}}_i = \mathcal{N}_i \cup \{i\}$.

The adjacency matrix $A \in \mathbb{R}^{N \times N}$ of a graph $\mathcal{G}$ is defined such that

$$A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The degree of the i-th vertex is defined as $d_i = \sum_{j=1}^{N} A_{ij}$ where A is the adjacency matrix of the graph and the maximum degree Δ of a graph is defined as $\Delta = \max(d_i)$.

2.2 Locally Coupled Dynamical System

A class of systems that motivates the development of the scalable algorithms presented in this thesis is the network of locally coupled dynamical systems. Suppose a network of agents is modeled as an undirected graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ where $\mathcal{A} = \{1, \dots, N\}$ is the set of agents. We denote the state of the i-th agent as $r_i \in \mathbb{R}^{p_i}$ and the set of states by the agents in the i-th agent's neighborhood as $r_{\bar{\mathcal{N}}^i} = \{r_j \,|\, j \in \bar{\mathcal{N}}^i\}$. The dynamics and measurement equations of the i-th agent are given by

$$\begin{aligned} \dot{r}_i(t) &= a_i(r_{\bar{\mathcal{N}}^i}(t), t) + w_i(t), \quad i \in \mathcal{A}, \\ y_i(t) &= c_i(r_{\bar{\mathcal{N}}^i}(t), t) + v_i(t), \quad i \in \mathcal{A}, \end{aligned} \tag{2.2}$$

where $a_i : \prod_{j \in \bar{\mathcal{N}}^i} \mathbb{R}^{p_j} \to \mathbb{R}^{p_i}$. The evolution of the i-th agent's state depends on the states of its neighboring agents. y_i describes the measurement equation of the

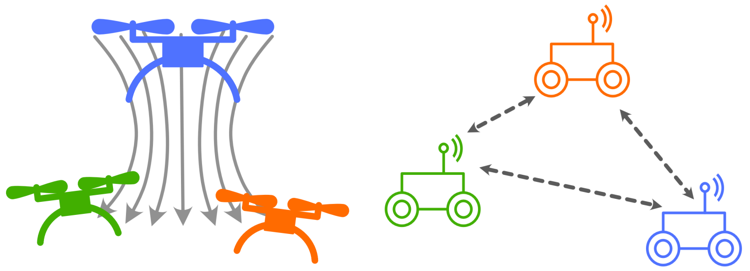


Figure 2.1: Illustration of locally coupled dynamics (left) and observations (right).

i-th agent, where $c_i : \prod_{j \in \bar{\mathcal{N}}^i} \mathbb{R}^{p_j} \to \mathbb{R}^{q_i}$. Similar to the dynamics equation, the observations made by the i-th agent may involve the neighbors' states. Additional noise terms to account for are denoted by $w_i(t) \in \mathbb{R}^{p_i}$ for process noise and $v_i(t) \in \mathbb{R}^{q_i}$ for measurement noise. The aggregated state vector of the overall system is written as $x = [r_1; \ldots; r_N]$.

The dynamical system described in (2.2) models the interactions of a variety of networked dynamical systems. Examples of locally coupled dynamics include the aerodynamics interaction between quadcopters [74] and electrical power grid [19], and examples of locally coupled measurements include range and bearing observation between agents. These notions of local coupling are illustrated in Fig. 2.1.

In this thesis, we develop distributed observer algorithms to collaboratively estimate the states of the systems including (2.2). The main challenge of the observer design problem is that the locally coupled dynamical systems described in (2.2) are typically *large-scale* problems where the dimension of the state vector grows with network size. Suppose the dimension of the overall state vector is $p \triangleq \sum_{i \in \mathcal{A}} p_i$. If we were to employ a classical distributed estimation algorithm such as distributed consensus Kalman filter whose algorithm complexity scales like $O(p^3)$, the problem becomes quickly intractable as the number of agents in the network increase. In this thesis, we show in Chapter 3 and 6 that the key to making the algorithm tractable is to develop a *localized* algorithm where each agent only maintains the information of the small subset of the state pertaining to its neighborhood.

2.3 Factor Graph Optimization

One of the mathematical tools used in robotics estimation problems is Factor Graph Optimization or FGO. Factor Graphs can be used to model the spatiotemporal sparsity of the estimation, which results in scalability in large-scale problems. Later in Chapter 6, we introduce novel algorithms to solve the FGO problem involving

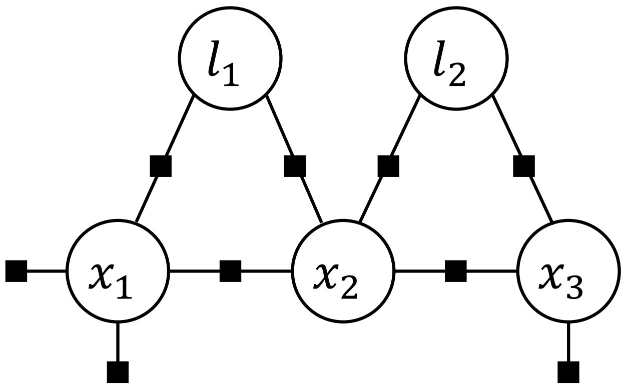


Figure 2.2: An example of a factor graph involving robot state at three time steps and two landmarks.

multi-agent systems. This section introduces the basic concept of factor graphs. For a more detailed background on factor graphs, readers are directed to [26].

To motivate the discussion of factor graphs, consider an example estimation problem where there is a state for a robot at three different time steps and two landmark variables, as illustrated in Fig. 2.2. The objective is to estimate the set of unknowns $\mathcal{X} = \{x_1, x_2, x_3, l_1, l_2\}$ given a set of independent random observations $\mathcal{Y} = \{y_1, \ldots, y_6\}$ and other information such as priors and dynamics. One way to estimate the unknown variables is to formulate the problem as a maximum a posteriori (MAP) estimation problem. The MAP estimator computes an estimate $\mathcal{X}^{\text{MAP}}$ such that it maximizes the posterior probability distribution, as follows:

$$\begin{aligned}\mathcal{X}^{\text{MAP}} &= \arg\max_{\mathcal{X}} p(\mathcal{X} \mid \mathcal{Y}) \\ &= \arg\max_{\mathcal{X}} \frac{p(\mathcal{Y} \mid \mathcal{X}) p(\mathcal{X})}{p(\mathcal{Y})} \\ &= \arg\max_{\mathcal{X}} l(\mathcal{X}; \mathcal{Y}) p(\mathcal{X}).\end{aligned} \tag{2.3}$$

In the manipulation above, the second equality uses the Bayes rule, and the third equality uses the fact that (a) the set of measurements $\mathcal{Y}$ is given, and (b) $p(\mathcal{Y} \mid \mathcal{X}) \propto l(\mathcal{X}; \mathcal{Y})$. Oftentimes, the probability distribution in the objective function of (2.3) can be further *factorized* into a product of independent probability distributions. For example, in the toy problem illustrated in Fig. 2.2, this factorization is given by

$$\begin{aligned}p(\mathcal{X}|\mathcal{Y}) \propto\ & p(x_1) p(x_2|x_1) p(x_3|x_2) \\ & \times l(x_1; y_1) l(x_3; y_2) \\ & \times l(x_1, l_1; y_3) l(x_2, l_1; y_4) l(x_2, l_2; y_5) l(x_3, l_2; y_6).\end{aligned} \tag{2.4}$$

Each term in the factorization is called a *factor*. The equation (2.4) contains different types of factors such as

1. Markov chain $p(x_1)p(x_2|x_1)p(x_3|x_2)$,
2. Likelihood function corresponding to absolute measurements $l(x_1; y_1)$ and $l(x_3; y_2)$, and
3. Relative measurements of landmarks $l(x_1, l_1; y_3)$, $l(x_2, l_1; y_4)$, $l(x_2, l_2; y_5)$, and $l(x_3, l_2; y_6)$.

Factor graphs are a family of sparse graphical models that describe the factorization of joint probability distributions. Formally, a factor graph is defined as a bipartite graph $\mathcal{G}_F = (\mathcal{X}, \mathcal{F}, \mathcal{E}_F)$ where $\mathcal{X}$ is the set of variables, $\mathcal{F}$ is the set of factors, and $\mathcal{E}_F$ is the set of edges. An edge $e_{ij} \in \mathcal{E}_F$ always connect one variable $x_i \in \mathcal{X}$ and one factor $\phi_j \in \mathcal{F}$. Coming back to our example, a factor graph representation of the joint probability distribution is depicted in Fig. 2.2. The circles correspond to $\mathcal{X}$, the squares correspond to $\mathcal{F}$, and the lines connecting them correspond to $\mathcal{E}_F$ in the figure. *Factor graph optimization* is the problem of computing the MAP estimate in (2.3) by exploiting the sparse structure of factor graphs.

We can make some connection between the Factor Graph Optimization and recursive estimator such as Kalman Filter. Consider the following dynamical system.

$$\begin{aligned} x_{t+1} &= f(x_t) + w_t, \quad t = [0, T-1], \\ y_t &= h(x_t) + v_t, \quad t = [0, T]. \end{aligned} \tag{2.5}$$

The process noise $w_t \sim \mathcal{N}(0, W)$ and measurement noise $v_t \sim \mathcal{N}(0, V)$ are given by zero-mean, Gaussian, i.i.d. random noise. The set of state variables is $\mathcal{X} = \{x_t \,|\, t = [0, T]\}$ and the set of measurements is $\mathcal{Y} = \{y_t \,|\, t = [0, T]\}$. Given a zero-mean, independent, Gaussian random prior information on x_0, denoted as $\hat{x}_0 \sim \mathcal{N}(0, P_0)$, the factor graph representation of the joint probability distribution is shown as Fig. 2.3, and the MAP estimation is given by

$$\mathcal{X}^{\text{MAP}} = \arg\min_{\mathcal{X}} \|x_0 - \hat{x}_0\|_{P_0} + \sum_{t=0}^{T-1} \|x_{t+1} - f(x_t)\|_W + \sum_{t=0}^{T} \|y_t - h(x_t)\|_V \tag{2.6}$$

For a linear system with Gaussian noise, the solution to the Kalman filter coincides with (2.6). The advantage of the Kalman Filter is that the optimal solution is computed in a recursive fashion and therefore algorithm is scalable in time. On

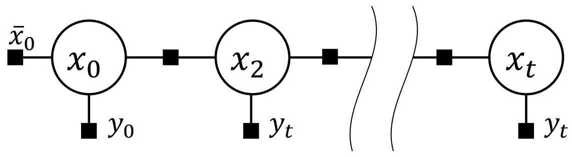


Figure 2.3: A factor graph for the dynamical system in (2.5).

the other hand, factor graph optimization solves (2.6) in batch, so the problem size increases as the time horizon increases. In Chapter 6, we discuss how a batch optimization approach preserves the sparsity structure of factor graph which is critical to deriving distributed and localized algorithms. Then, we overcome the time scalability by proposing an incremental approach to solving the distributed factor graph optimization problem.

2.4 Chapter Summary

In this chapter, we introduced some notation and concepts in graph theory, locally coupled dynamical systems, and factor graph optimization. In the following chapters, we use these concepts in the description and development of our theory and algorithms.

Chapter 3

DECENTRALIZED POSE ESTIMATION

This chapter contains material from the following publications:

[1] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, "Decentralized formation pose estimation for spacecraft swarms," *Advances in Space Research*, vol. 67, no. 11, pp. 3527–3545, 2021. DOI: `10.1016/j.asr.2020.06.016`,

[2] K. Matsuka, E. S. Lupu, Y. K. Nakka, R. Foust, S.-J. Chung, and F. Hadaegh, "Distributed multi-target relative pose estimation for cooperative spacecraft swarm," in *Proc. 10th International Workshop on Satellite Constellations and Formation Flying*, 2019,

3.1 Introduction

In this chapter, we develop a distributed and localized relative navigation algorithm for spacecraft swarms called the Decentralized Pose Estimation (DPE) algorithm. This is the first work in this thesis towards addressing the challenge of scalability in cooperative estimation problems for locally coupled systems.

Cooperative localization of spacecraft swarm is challenging. First, in some multi-agent localization algorithms for small-scale swarms, the time complexity scales at least linearly with the formation size [75], [76]. Hence, these algorithms are

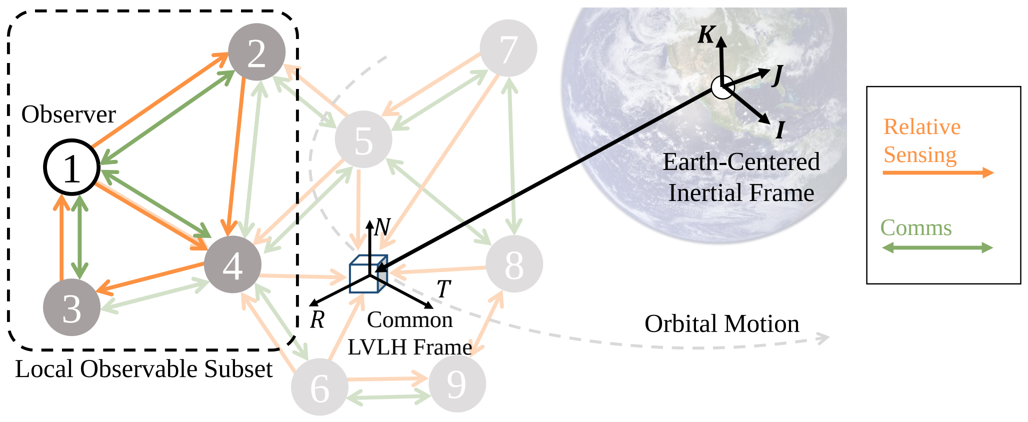


Figure 3.1: A spacecraft swarm and its relative sensing and communication networks. The DPE estimates the spacecraft poses in the local observable subset with respect to the common Local-Horizontal, Local-Vertical (LVLH) frame.

not suitable for large-scale swarms. Another challenge is the requirement that each spacecraft must estimate the absolute orbit of a reference spacecraft in order to define a common Local-Horizontal, Local-Vertical (LVLH) frame estimate (see Fig. 3.1). This estimation is challenging for large-scale swarms as some of the spacecraft in the swarm may not make a direct measurement of this reference spacecraft [10], [77]. Any algorithm suitable for swarm localization requires a novel approach that explicitly addresses these challenges.

DPE addresses the issue of scalability by being a distributed and localized estimation algorithm. Using the DPE, each spacecraft estimates the states of only a local subset of the swarm. Neighbor spacecraft cooperate to improve the pose estimates by sharing their measurements. The DPE algorithm uses the result of nonlinear observability analysis to determine the local observable subset of spacecraft given the *ad hoc* relative sensing and communication graphs for each spacecraft. Then, it jointly localizes the spacecraft in the local observable subset by fusing the measurements that are collected over the communication network. Since each spacecraft only requires a local subset of information, DPE is easily scalable to larger swarm sizes. DPE also offers some advantages over algorithms without collaboration by improving the estimation accuracy and increasing the number of observable spacecraft.

For our specific implementation, we represent the attitude of the spacecraft with a quaternion and use an Extended Kalman Filter (EKF) to estimate the error in the attitude state at each time step. As part of the DPE, we also present the Swarm Reference Frame Estimation (SRFE) algorithm, which allows each spacecraft to co-estimate the common LVLH frame of the swarm in a decentralized manner. The SRFE applies the decentralized consensus filter [9], [10] to estimate the reference spacecraft that may be visible to only a subset of the spacecraft. The DPE combined with the SRFE provides a fully decentralized navigation solution that can be used in swarm motion planning.

The DPE algorithm was verified in simulations and in real-time robotic experiments. The DPE performance was compared against that of an Individual EKF, wherein each spacecraft uses only its measurements to estimate only those spacecraft it directly measures, and a Centralized EKF, which has access to all the information in the swarm. The robotic experiment was conducted on Caltech's robotic spacecraft dynamics simulators, the Multi-Spacecraft Testbed for Autonomy Research (M-STAR) [78], [79]. The relative pose of each spacecraft was estimated using vision on-board each spacecraft. We validated the DPE estimate against the ground truth obtained from a motion capture system. In summary, this chapter presents a scalable, decentralized algorithm for swarm localization that is appropriate for onboard implementation.

3.2 Problem Statement

Prior to describing our DPE algorithm, we first provide a brief summary of the mathematical notations used in our work, as well as some relevant background information on orbital dynamics and error state estimation.

Let $\mathcal{G}^{\mathrm{s}} = (\mathcal{A}, \mathcal{E}^{\mathrm{s}})$ denote a directed graph that describes the *relative sensing graph*, with $\mathcal{A} = \{1, \ldots, N\}$ the set of agents (spacecraft) and $\mathcal{E}^{\mathrm{s}}$ the set of edges. An edge (i, j) is in $\mathcal{E}^{\mathrm{s}}$ when the i-th spacecraft measures the relative pose of the j-th spacecraft. Similarly, let $\mathcal{G}^{\mathrm{c}} = (\mathcal{A}, \mathcal{E}^{\mathrm{c}})$ denote the *communication graph*, an undirected graph for the communication topology. We say $(i, j) \in \mathcal{E}^{\mathrm{c}}$ if there is a communication link between the i-th and the j-th spacecraft. Note that the measurement graph and the communication graph may be different in general. The out-neighbors of a node i in a graph $\mathcal{G}$ are defined as $\mathcal{N}_i = \{j \in \mathcal{A} \mid (i, j) \in \mathcal{E}(\mathcal{G})\}$ and we use subscripts s and c to distinguish the neighbors for relative sensing and communication graphs, respectively. The neighborhood of node i is defined as $\bar{\mathcal{N}_i} = \mathcal{N}_i \cup \{i\}$. The degree of a node in a graph is defined as $d_i = \sum_{j=1}^{N} A_{ij}$ where A is the adjacency matrix of the graph and the maximum degree Δ of a graph is defined as $\Delta = \max(d_i)$.

A column concatenation of vectors $x_1, \ldots, x_n$ is written as $\overline{x} = [x_1; \ldots; x_n]$ or $\overline{x} = \|_{i=1,\ldots,n} x_i$, where the bar over the variable denotes an augmented variable defined as a concatenation of variables. The positions and velocities of an object a in frame b are denoted $p_{a,b}, v_{a,b}$. The attitude and angular rate of the frame a with respect to the frame b are denoted $q_{a,b}$ and $\omega_{a,b}$. The function $R(\cdot)$ maps a quaternion onto a rotation matrix such that $x_b = R(q_{a,b}) x_a$ where x_a, x_b are vectors expressed in frame a and b, respectively. We use ˆ above variables to denote their estimates.

Relative Orbital Dynamics

This section reviews the equation of motion for the relative orbital dynamics. For the rest of the paper, we assume that each spacecraft is in a near-circular orbit, there are no perturbations, and that all the spacecraft are in proximity such that the relative orbital dynamics can be linearized to the Hill-Clohessy-Wiltshire (HCW) equations [80].

There exist some dynamics models that include eccentricity or other perturbation effects [67], [81], [82]; however, we choose the HCW model dynamics to illustrate more clearly the decentralized aspects of the algorithm. Suppose $p_{i,L}, v_{i,L}$ are the position and velocity vectors of the i-th spacecraft with respect to an LVLH frame that is commonly known among all the spacecraft in the swarm. The dynamics of translational states using the HCW equations are described by

$$\begin{bmatrix} \dot{p}_{i,L} \\ \dot{v}_{i,L} \end{bmatrix} = A_t \begin{bmatrix} p_{i,L} \\ v_{i,L} \end{bmatrix} + B_t w_t \tag{3.1}$$

where state matrix A_t and actuation matrix B_t are given by

$$\begin{aligned} A_t &= \begin{bmatrix} 0_{3\times3} & I_3 \\ A_{vp} & A_{vv} \end{bmatrix}, \quad B_t = \begin{bmatrix} 0_{3\times3} \\ I_3 \end{bmatrix} \\ A_{vp} &= \begin{bmatrix} 3n^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -n^2 \end{bmatrix}, \quad A_{vv} = \begin{bmatrix} 0 & 2n & 0 \\ -2n & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.2}$$

where n is the mean anomaly of the reference spacecraft orbit and $w_t \sim \mathcal{N}(0_{3\times1}, W_t)$ is assumed to be a zero mean Gaussian process noise.

Attitude Dynamics

For DPE, we choose to represent the attitude components of the spacecraft state as quaternions [83]. A quaternion is defined as $q = [q_v; q_s]$ where $q_v \in \mathbb{R}^3$ is the vector and $q_s \in \mathbb{R}$ is the scalar components of the quaternion, respectively. A unit quaternion $q \in S^3$ satisfies the constraint $q^\top q = 1$. Quaternion multiplication is denoted with the group operator $\otimes$ and is defined as

$$q' \otimes q = \begin{bmatrix} q'_s q_v + q_s q'_v - q'_v \times q_v \\ q'_s q_s - q'_v \cdot q_v \end{bmatrix}. \tag{3.3}$$

The inverse of a quaternion is defined as

$$q^{-1} = \frac{1}{||q||} \begin{bmatrix} -q_v \\ q_s \end{bmatrix}. \tag{3.4}$$

A small attitude perturbation $\delta q \in S^3$ can be represented in a minimal coordinate $a \in \mathbb{R}^3$ where the mapping from $a \in \mathbb{R}^3$ to δq is defined as follows

$$\delta q(a) = \frac{1}{2} \begin{bmatrix} a \\ \sqrt{4 - a^\top a} \end{bmatrix}. \tag{3.5}$$

Suppose $q_{i,I}$ and $q_{i,I}^{\mathrm{ref}}$ denote the true and reference attitude for the i-th spacecraft, respectively. Then, the attitude error $a_{i,I}$ and angular rate error $\delta\omega_{i,I}$ are defined such that

$$\delta q(a_{i,I}) = q_{i,I} \otimes (q_{i,I}^{\mathrm{ref}})^{-1}, \tag{3.6}$$

$$\delta\omega_{i,I} = \omega_{i,I} - \omega_{i,I}^{\mathrm{ref}}. \tag{3.7}$$

The kinematic differential equation for the attitude quaternion for the i-th spacecraft body frame can be expressed in the following two equivalent forms:

$$\dot{q}_{i,I} = \frac{1}{2}\Omega(\omega_{i,I})q_{i,I} = \frac{1}{2}\Theta(q_{i,I})\omega_{i,I} \tag{3.8}$$

where $\Omega(\omega)$ and $\Theta(q)$ are matrices defined as

$$\Omega(\omega) = \begin{bmatrix} -\omega^{\times} & \omega \\ -\omega^{\top} & 0 \end{bmatrix}, \quad \Theta(q) = \begin{bmatrix} q_s I_3 + q_v^{\times} \\ -q_v^{\top} \end{bmatrix}. \tag{3.9}$$

The superscript $\times$ denotes a skew-symmetric matrix. The attitude rate of the i-th spacecraft is propagated via the following equation

$$\dot{\omega}_{i,I} = -J^{-1}{\omega_{i,I}}^{\times} J\omega_{i,I} + J^{-1}w_a \tag{3.10}$$

where J is the inertia tensor of the spacecraft in the body frame, and $w_a \sim \mathcal{N}(0_{3\times1}, W_a)$ is the attitude process noise modeled as a torque perturbation with zero-mean white Gaussian noise.

Review of Error State Estimation

Because the standard EKF does not strictly enforce the manifold constraint for quaternions, we estimate the error state for the attitude components. This is similar to conventional attitude estimation techniques such as the Multiplicative EKF (MEKF) [84]. The main idea is to estimate attitude error in a minimal coordinate at each step while using a quaternion to provide a non-singular attitude representation overall. The algorithm involves three steps: time update, measurement update, and reset. This section reviews the time update of the error state and its covariance, as well as the reset step.

The state variables of the i-th spacecraft x_i are defined as

$$x_i = [p_{i,L};\ \ v_{i,L};\ \ q_{i,I};\ \ \omega_{i,I}] \tag{3.11}$$

where $p_{i,L}$ and $v_{i,L}$ are relative positions and velocities of the i-th spacecraft with respect to the swarm reference LVLH frame. Attitude parameters $q_{i,I}$ and $\omega_{i,I}$ are the quaternion and the angular rate of the i-th spacecraft with respect to the Earth-Centered Inertial (ECI) frame, respectively. Precisely speaking, translational states and rotational states are expressed with respect to different frames (i.e., the LVLH and the ECI frames). This is a convenient choice that was made in order to simplify the relative orbital and attitude dynamics models.

Following the definitions in (3.6) and (3.7), the attitude parameters are decomposed into reference and error terms: $q_{i,I} = \delta q(a_{i,I}) \otimes q_{i,I}^{\text{ref}}$ and $\omega_{i,I} = \omega_{i,I}^{\text{ref}} + \delta\omega_{i,I}$. Non-singular representation of x_i is denoted by the reference state vector

$$x_i^{\text{ref}} = [p_{i,L};\;\; v_{i,L};\;\; q_{i,I}^{\text{ref}};\;\; \omega_{i,I}^{\text{ref}}]. \tag{3.12}$$

At each filtering time step, the actual state to be estimated is the minimal coordinate representation of state with respect to $q_{i,I}^{\text{ref}}$ and $\omega_{i,I}^{\text{ref}}$ defined as

$$x_i^{\min} = [p_{i,L};\;\; v_{i,L};\;\; a_{i,I};\;\; \delta\omega_{i,I}]. \tag{3.13}$$

We refer to this as the minimal state vector of the i-th spacecraft, denoted by the superscript min. At each step, $q_{i,I}^{\text{ref}}$ and $\omega_{i,I}^{\text{ref}}$ are selected such that the prior estimate of $a_{i,I}$ and $\delta\omega_{i,I}$ are identically zero.

The state vector x_i resides on a manifold $\mathcal{M} = \mathbb{R}^6 \times S^3 \times \mathbb{R}^3$ and we can extend the notion of group operator to states in $\mathcal{M}$ as follows. Suppose $x', x \in \mathcal{M}$. Then the group operator $\boxplus$ is defined as

$$x' \boxplus x = \begin{bmatrix} p' + p \\ v' + v \\ q' \otimes q \\ \omega' + \omega \end{bmatrix}. \tag{3.14}$$

Suppose the error state between two states is defined as $\Delta x = x' \boxplus x^{-1} \in \mathcal{M}$, whose components are denoted by $\Delta x = [\Delta p;\;\; \Delta v;\;\; \Delta q;\;\; \Delta\omega]$. This error state can be parameterized by a minimal state error $\Delta\chi \in \mathbb{R}^{12}$ defined as

$$\Delta\chi = [\Delta p;\;\; \Delta v;\;\; a(\Delta q);\;\; \Delta\omega]. \tag{3.15}$$

Finally, the two states, x and x', and the state error $\Delta\chi$ are related by

$$x' = \Delta x(\Delta\chi) \boxplus x \tag{3.16}$$

where $\Delta x(\cdot) : \mathbb{R}^{12} \to \mathcal{M}$ is the map from $\Delta\chi$ to Δx. To obtain the posterior state vector for each spacecraft, the DPE evaluates (3.16) at each reset step to apply the correction $\Delta\chi$, which is expressed in minimal coordinates.

Next, we derive the equation of motion for attitude error states, which is necessary for computing the covariance time update. Taking the time derivative of the error quaternion given by (3.6) and substituting (3.8), one can obtain

$$2\dot{\delta q}(a_{i,I}) = \begin{bmatrix} \omega_{i,I} \\ 0 \end{bmatrix} \otimes \delta q(a_{i,I}) - \delta q(a_{i,I}) \otimes \begin{bmatrix} \omega_{i,I}^{\text{ref}} \\ 0 \end{bmatrix}. \tag{3.17}$$

Substituting this into (3.5) leads to the equation of motion for attitude error

$$\dot{a}_{i,I} = \left(2\dot{\delta q}(a_{i,I})\right)_{1:3} = \frac{1}{2}\left((4 - a_{i,I}^\top a_{i,I})^{\frac{1}{2}}\delta\omega_{i,I} - (2\omega_{i,I}^{\text{ref}} + \delta\omega_{i,I}) \times a_{i,I}\right). \tag{3.18}$$

Similarly, the equation of motion for angular rate error can be derived from (3.7) and (3.10)

$$\dot{\delta\omega}_{i,I} = -J^{-1}\left(\delta\omega_{i,I}^\times J(\omega_{i,I}^{\text{ref}} + \delta\omega_{i,I}) + \omega_{i,I}^{\text{ref}\times} J\delta\omega_{i,I}\right) + J^{-1}w_a. \tag{3.19}$$

(3.18) and (3.19) together represent the attitude error dynamics. The Jacobian of error attitude dynamics in (3.18) and (3.19) with respect to error attitude variables is given by

$$\left.\frac{\partial \dot{a}_{i,I}}{\partial a_{i,I}}\right|_{a_{i,I},\delta\omega_{i,I}=0} = \left.\left(\frac{-\delta\omega_{i,I}^\top a_{i,I}}{2(4 - a_{i,I}^\top a_{i,I})^{\frac{1}{2}}} - \omega_{i,I}^{\text{ref}\times} - \frac{1}{2}\delta\omega_{i,I}^\times\right)\right|_{a_{i,I},\delta\omega_{i,I}=0} = -\omega_{i,I}^{\text{ref}\times} \tag{3.20}$$

$$\left.\frac{\partial \dot{a}_{i,I}}{\partial \delta\omega_{i,I}}\right|_{a_{i,I}=0} = \left.\left(\frac{1}{2}(4 - a_{i,I}^\top a_{i,I})^{\frac{1}{2}} I_3 + a_{i,I}^\times\right)\right|_{a_{i,I}=0} = I_3 \tag{3.21}$$

$$\frac{\partial \dot{\delta\omega}_{i,I}}{\partial a_{i,I}} = 0_{3\times 3} \tag{3.22}$$

$$\begin{aligned} \left.\frac{\partial \dot{\delta\omega}_{i,I}}{\partial \delta\omega_{i,I}}\right|_{\delta\omega_{i,I}=0} &= -J^{-1}\left.\left(\delta\omega_{i,I}^\times J - (J(\omega_{i,I}^{\text{ref}} + \delta\omega_{i,I}))^\times + \omega_{i,I}^{\text{ref}\times} J\right)\right|_{\delta\omega_{i,I}=0} \\ &= J^{-1}\left((J\omega_{i,I}^{\text{ref}})^\times - \omega_{i,I}^{\text{ref}\times} J\right). \end{aligned} \tag{3.23}$$

Finally, the covariance can be computed by solving the differential Lyapunov equation

$$\dot{P} = AP + PA^\top + BWB^\top \tag{3.24}$$

where A is Jacobian of propagation of states.

$$A = \begin{bmatrix} A_t & 0_{6\times 6} \\ 0_{6\times 6} & A_a \end{bmatrix}, \quad B = \begin{bmatrix} B_t & 0_{6\times 3} \\ 0_{6\times 3} & B_a \end{bmatrix}, \quad W = \begin{bmatrix} W_t & 0_{3\times 3} \\ 0_{3\times 3} & W_a \end{bmatrix} \tag{3.25}$$

where A_t and B_t are given by HCW equations in (3.2) and A_a and B_a are given by

$$A_a = \begin{bmatrix} -\omega_{i,I}^{\text{ref}\times} & I_3 \\ 0_{3\times 3} & J^{-1}\left((J\omega_{i,I}^{\text{ref}})^\times - \omega_{i,I}^{\text{ref}\times} J\right) \end{bmatrix}, \quad B_a = \begin{bmatrix} 0_{3\times 3} \\ J^{-1} \end{bmatrix}. \tag{3.26}$$

Because the translational and rotational dynamics are decoupled, the state matrix preserves a block diagonal structure.

In this section, we briefly described various mathematical notations and background knowledge on orbital dynamics and error state estimation. This information was foundational in the development of our DPE algorithm, which we describe in the next section.

3.3 Decentralized Pose Estimation Algorithm

In this section, we describe the DPE algorithm. The DPE estimates the poses of a local observable subset of spacecraft in a swarm, given the relative sensing and communication network topologies. First, each spacecraft measures the poses of itself and its neighbors. Each spacecraft then communicates its measurements and the associated measurement noise covariances to its communication neighbors $j \in \mathcal{N}_i^{\mathrm{c}}$. Based on the available communication and relative sensing networks at the given time, the augmented state vector is modified to add newly detected spacecraft and subtract the spacecraft that became unobservable. Finally, each spacecraft jointly estimates the poses of the local spacecraft. This algorithm is summarized in Algorithm 1 and the following sections explain the steps in detail. A copy of the same algorithm is implemented on each spacecraft.

First, we clearly define the set of spacecraft to be estimated by the i-th spacecraft.

Definition 1. *The local observable set $\mathcal{V}_i \subset \mathcal{A}$ for the i-th spacecraft is defined as the union of the sensing neighborhood over the communication neighborhood. That is*

$$\mathcal{V}_i := \bigcup_{j \in \bar{\mathcal{N}}_i^{\mathrm{c}}} \bar{\mathcal{N}}_j^{\mathrm{s}}. \tag{3.27}$$

This is the set of agents detected by the i-th spacecraft either via communication or via relative sensing in one communication step. The goal of DPE is for each spacecraft $i \in \mathcal{A}$ to estimate the state of each detected spacecraft $j \in \mathcal{V}_i$. Suppose the cardinality of the local observable set is $N_i = \mathrm{card}(\mathcal{V}_i)$. We define the reference augmented state vector $\bar{x}_i^{\mathrm{ref}} \in \mathcal{M}^{N_i}$ and the minimal state augmented vector $\bar{x}_i^{\mathrm{min}} \in \mathbb{R}^{12N_i}$ for the i-th spacecraft as the column concatenation of all states for $j \in \mathcal{V}_i$. That is

$$\bar{x}_i^{\mathrm{ref}} = \|_{j \in \mathcal{V}_i} x_j^{\mathrm{ref}}, \quad \bar{x}_i^{\mathrm{min}} := \|_{j \in \mathcal{V}_i} x_j^{\mathrm{min}} \tag{3.28}$$

where x_j^{ref} and x_j^{min} correspond to the full and minimal state for the j-th spacecraft as defined in (3.11) and (3.13).

Absolute and Relative Measurement Models

Each spacecraft $i \in \mathcal{A}$ is assumed to have an absolute pose measurement y_i with respect to the Earth-Centered Inertial (ECI) frame.

$$y_i = h^{\mathrm{a}}(x_i, p_{L,I}, q_{L,I}) + \psi_i^{\mathrm{a}} = \begin{bmatrix} R(q_{L,I})p_{i,L} + p_{L,I} \\ q_{i,I} \end{bmatrix} + \psi_i^{\mathrm{a}} \tag{3.29}$$

where ψ_i^{a} denotes measurement noise. $p_{L,I}$ and $q_{L,I}$ describe the LVLH to ECI transformation and are treated as fixed parameters known from the SRFE algorithm. This measurement is available from GPS and a star tracker. We denote the position and the attitude components of this observation as $y_i = [p_{i,I}^{\mathrm{obs}}; q_{i,I}^{\mathrm{obs}}]$. Since the attitude measurement is given as a quaternion, it is convenient to transform the observation to a pseudo-measurement form [84]

$$\tilde{y}_i = \begin{bmatrix} R(q_{L,I})(p_{i,I}^{\mathrm{obs}} - p_{L,I}) \\ 2\left(q_{i,I}^{\mathrm{obs}} \otimes (q_{i,I})^{-1}\right)_{1:3} \end{bmatrix}. \tag{3.30}$$

Then this measurement can be modeled as

$$\tilde{y}_i = \tilde{h}^{\mathrm{a}}(x_i^{\mathrm{min}}) + \tilde{\psi}_i^{\mathrm{a}} = \begin{bmatrix} p_{i,L} \\ a_{i,I} \end{bmatrix} + \tilde{\psi}_i^{\mathrm{a}} \tag{3.31}$$

where $\tilde{\psi}_i^{\mathrm{a}} \sim \mathcal{N}(0_{6\times}, \tilde{\Psi}_i)$ is the absolute pseudo-measurement noise vector.

In addition to its absolute measurement, each spacecraft may have relative measurements, possibly multiple at a given time. Each relative measurement is assumed to be a pose measurement provided by a monocular camera. The availability of relative measurements depends on the physical constraints of the given sensors such as range, field-of-view (FOV), and lighting. This information is captured by the edges in the relative sensing graph $\mathcal{G}_s$. The relative measurement is assumed to give the relative pose of the observed spacecraft with respect to the observer. Let $y_{j,i} = [p_{j,i}^{\mathrm{obs}}; q_{j,i}^{\mathrm{obs}}]$ denote the pose of the j-th spacecraft relative to the i-th spacecraft. The relative attitude can be written in terms of reference and error attitude as follows

$$\begin{aligned} q_{j,i} &= q_{j,I} \otimes \left(q_{i,I}\right)^{-1} \\ &= \delta q(a_{j,I}) \otimes q_{j,i}^{\mathrm{ref}} \otimes \delta q(-a_{i,I}) \end{aligned} \tag{3.32}$$

where $q_{j,i}^{\mathrm{ref}} = q_{j,I}^{\mathrm{ref}} \otimes \left(q_{i,I}^{\mathrm{ref}}\right)^{-1}$. In the same way as absolute measurement, it is more convenient to transform the relative measurement to a minimal parameterization.

We define the relative pseudo measurement by

$$\tilde{y}_{j,i} = \begin{bmatrix} p^{\text{obs}}_{j,i} \\ a^{\text{obs}}_{j,i} \end{bmatrix} \tag{3.33}$$

where $a^{\text{obs}}_{j,i} = 2\left(q^{\text{obs}}_{j,i} \otimes \left(q^{\text{ref}}_{j,i}\right)^{-1}\right)_{1:3}$. Therefore, the minimal relative pose measurement of the j-th spacecraft with respect to the i-th spacecraft is given by

$$\tilde{y}_{j,i} = \tilde{h}^{\text{r}}\left(x^{\min}_{i}, x^{\min}_{j}, x^{\text{ref}}_{i}\right) + \tilde{\psi}^{\text{r}}_{j,i}. \tag{3.34}$$

where $\tilde{\psi}^{\text{r}}_{j,i}$ is relative pseudo-measurement noise and the measurement model is given by

$$\tilde{h}^{\text{r}}\left(x^{\min}_{i}, x^{\min}_{j}, x^{\text{ref}}_{i}\right) = \begin{bmatrix} R(\delta q(a_{i,I}))R(q^{\text{ref}}_{i,I})R(q_{L,I})^{\top}(p_{j,L} - p_{i,L}) \\ 2\left(\delta q(a_{j,I}) \otimes q^{\text{ref}}_{j,I} \otimes \delta q(-a_{i,I}) \otimes \left(q^{\text{ref}}_{j,I}\right)^{-1}\right)_{1:3} \end{bmatrix}. \tag{3.35}$$

From this, the Jacobian of relative pseudo-measurement with respect to each minimal state variable can be computed as follows

$$\frac{\partial \tilde{h}^{\text{r}}}{\partial p_{i,L}} = \begin{bmatrix} -R(q^{\text{ref}}_{i,I})R(q_{L,I})^{\top} \\ 0_{3\times 3} \end{bmatrix} \tag{3.36}$$

$$\frac{\partial \tilde{h}^{\text{r}}}{\partial p_{j,L}} = \begin{bmatrix} R(q^{\text{ref}}_{i,I})R(q_{L,I})^{\top} \\ 0_{3\times 3} \end{bmatrix} \tag{3.37}$$

$$\frac{\partial \tilde{h}^{\text{r}}}{\partial a_{i,I}} = \begin{bmatrix} \left[R(q^{\text{ref}}_{i,I})R(q_{L,I})^{\top}(p_{j,L} - p_{i,L})\right]^{\times} \\ -R(q^{\text{ref}}_{j,i}) \end{bmatrix} \tag{3.38}$$

$$\frac{\partial \tilde{h}^{\text{r}}}{\partial a_{j,I}} = \begin{bmatrix} 0_{3\times 3} \\ I_3 \end{bmatrix}. \tag{3.39}$$

Communication and Augmented Sensing

At every communication step, each spacecraft broadcasts its sensing information, including both its absolute and relative measurements. Absolute sensing information is defined as $\mathcal{M}^{\text{a}}_{i} = (y_i, \Psi_i, i)$. For each edge in the relative sensing graph $\mathcal{G}_s$, the relative sensing information is defined as $(y_{j,i}, \Psi_{j,i}, (i,j))$. We define the set $\mathcal{M}^{\text{r}}_{i}$ to be the set of relative sensing information for all of the direct measurements the i-th spacecraft makes:

$$\mathcal{M}^{\text{r}}_{i} = \{(y_{j,i}, \Psi_{j,i}, (i,j)) \mid j \in \mathcal{N}^{\text{s}}_{i}\} \tag{3.40}$$

where $\mathcal{N}^{\text{s}}_{i}$ denotes the neighbors of the i-th spacecraft in the relative sensing graph $\mathcal{G}^{\text{s}}$. At each communication time step, each spacecraft broadcasts $\mathcal{M}^{\text{a}}_{i}$ and $\mathcal{M}^{\text{r}}_{i}$ to its communication neighbors.

Each spacecraft also collects the information broadcasted by its neighbors. The set of all of the relative sensing edges collected by the i-th spacecraft is

$$\bar{\mathcal{E}}_i^{\mathrm{s}} := \{(j,k) \in \mathcal{E}^{\mathrm{s}} \mid j \in \bar{\mathcal{N}}_i^{\mathrm{c}}\}. \tag{3.41}$$

The augmented relative observation, measurement model, and noise are defined as

$$\begin{gathered} \overline{y}_i^{\mathrm{r}} := \|_{(j,k)\in\bar{\mathcal{E}}_i^{\mathrm{s}}} \tilde{y}_{k,j}, \quad \overline{h}_i^{\mathrm{r}}(\overline{x}_i^{\min}, \overline{x}_i^{\mathrm{ref}}) := \|_{(j,k)\in\bar{\mathcal{E}}_i^{\mathrm{s}}} \tilde{h}^{\mathrm{r}}(x_j^{\min}, x_k^{\min}, x_j^{\mathrm{ref}}) \\ \overline{\psi}_i^{\mathrm{r}} := \|_{(j,k)\in\bar{\mathcal{E}}_i^{\mathrm{s}}} \tilde{\psi}_{k,j}^{\mathrm{r}} \end{gathered} \tag{3.42}$$

which are column concatenations over all of the relative sensing edges available to the i-th spacecraft. Similarly, each spacecraft collects all of the absolute measurements

$$\overline{y}_i^{\mathrm{a}} := \|_{j\in\bar{\mathcal{N}}_i^{\mathrm{c}}} \tilde{y}_j, \quad \overline{h}_i^{\mathrm{a}}(\overline{x}_i^{\min}) := \|_{j\in\bar{\mathcal{N}}_i^{\mathrm{c}}} \tilde{h}^{\mathrm{a}}(x_j^{\min}), \quad \overline{\psi}_i^{\mathrm{a}} := \|_{j\in\bar{\mathcal{N}}_i^{\mathrm{c}}} \tilde{\psi}_j^{\mathrm{a}}. \tag{3.43}$$

The total augmented measurement is the collection of all of the relative and absolute measurements. That is, $\overline{y}_i = [\overline{y}_i^{\mathrm{a}}; \overline{y}_i^{\mathrm{r}}]$, $\overline{h}_i = [\overline{h}_i^{\mathrm{a}}; \overline{h}_i^{\mathrm{r}}]$, and $\overline{\psi}_i = [\overline{\psi}_i^{\mathrm{a}}; \overline{\psi}_i^{\mathrm{r}}]$, such that the augmented measurement equation becomes

$$\overline{y}_i = \overline{h}_i(\overline{x}_i^{\min}, \overline{x}_i^{\mathrm{ref}}) + \overline{\psi}_i. \tag{3.44}$$

The corresponding Jacobian linearized around the estimates $\hat{\overline{x}}_i^{\min}$ and $\hat{\overline{x}}_i^{\mathrm{ref}}$ becomes

$$\overline{H}_i = \left.\frac{\partial \overline{h}_i(\overline{x}_i^{\min}, \hat{\overline{x}}_i^{\mathrm{ref}})}{\partial \overline{x}_i^{\min}}\right|_{\overline{x}_i^{\min} = \hat{\overline{x}}_i^{\min}}. \tag{3.45}$$

Since all of the measurement models depend only on one or two spacecraft states at a time, each row of $\overline{H}_i$ will be sparse.

For each spacecraft $i \in \mathcal{A}$, we have propagation models for the full and minimal state vectors

$$\dot{x}_i^{\mathrm{ref}} = f(x_i^{\mathrm{ref}}) \tag{3.46}$$

$$\dot{x}_i^{\min} = f^{\min}(x_i^{\min}, x_i^{\mathrm{ref}}) \tag{3.47}$$

where the reference state model is given by collecting (3.2), (3.8), and (3.10) and the minimal propagation model is given by (3.2), (3.18), and (3.19). Recall the augmented state for the i-th spacecraft is $\overline{x}_i^{\mathrm{ref}}$. Then, the augmented dynamical system for all spacecraft $j \in \mathcal{V}_i$ is given by

$$\dot{\overline{x}}_i^{\mathrm{ref}} = \overline{f}_i(\overline{x}_i^{\mathrm{ref}}) := \|_{j\in\mathcal{V}_i} f(x_j^{\mathrm{ref}}) \tag{3.48}$$

$$\dot{\overline{x}}_i^{\min} = \overline{f}_i^{\min}(\overline{x}_i^{\min}, \overline{x}_i^{\mathrm{ref}}) := \|_{j\in\mathcal{V}_i} f^{\min}(x_j^{\min}, x_j^{\mathrm{ref}}). \tag{3.49}$$

Integrating (3.48) propagates the previous prior estimate $\hat{\overline{x}}_i^{\text{ref}+}(t-1)$ to the current posterior estimate $\hat{\overline{x}}_i^{\text{ref}-}(t)$, where the superscript $-$ and $+$ denotes prior and posterior, respectively. (3.49), instead, is used to define the augmented Jacobian

$$\overline{A}_i = \frac{\partial \overline{f}_i^{\min}(\overline{x}_i^{\min}, \hat{\overline{x}}_i^{\text{ref}})}{\partial \overline{x}_i^{\min}}\Bigg|_{\overline{x}_i^{\min}=\hat{\overline{x}}_i^{\min}}. \tag{3.50}$$

Since the propagation of each state is decoupled, (3.50) is a block diagonal where the diagonal block corresponding to $j \in \mathcal{V}_i$ is given by $A(x_j^{\text{ref}})$ from (3.26). Using these equations, the augmented posterior covariance from the previous step can be updated to the current prior covariance using

$$\dot{\overline{P}}_i = \overline{A}_i\overline{P}_i + \overline{P}_i\overline{A}_i^\top + \overline{B}_i\overline{W}_i\overline{B}_i^\top. \tag{3.51}$$

While the augmented state matrix $\overline{A}_i$ and the process noise covariance term $\overline{B}_i\overline{W}_i\overline{B}_i^\top$ are block diagonal, (3.51) has to be solved simultaneously because $\overline{P}_i$ is not diagonal.

At the measurement update of the DPE, the Kalman gain, the posterior covariance, and the state correction terms are computed similarly to the standard EKF:

$$K_i = \overline{P}_i^-\overline{H}_i^\top\left(\overline{H}_i\overline{P}_i^-\overline{H}_i^\top + \overline{\Psi}_i\right)^{-1} \tag{3.52}$$

$$\overline{P}_i^+ = \left(I - K_i\overline{H}_i\right)\overline{P}_i^- \tag{3.53}$$

$$\Delta\bar{\chi}_i = K_i(\bar{y}_i - \overline{h}_i(\overline{x}_i^{\min}, \overline{x}_i^{\text{ref}})). \tag{3.54}$$

Because the correction term $\Delta\bar{\chi}_i$ is expressed in the minimal coordinate $\mathbb{R}^{12N_i}$, we apply the reset step to the augmented reference state to recover the posterior estimate of the state

$$\overline{x}_i^{\text{ref}+} = \Delta\bar{x}_i(\Delta\bar{\chi}_i) \boxplus \overline{x}_i^{\text{ref}-}. \tag{3.55}$$

The definition of the group operator $\boxplus$ and the mapping between the tangent space are extended to those for the augmented vector by simply applying the operations for each of $j \in \mathcal{V}_i$.

For this chapter, we implemented the DPE algorithm with specific definitions for the state, measurement, and dynamics models. However, the strategy of defining the augmented state vector and measurements can be extended to different scenarios.

Adding and Subtracting Nodes to Set $\mathcal{V}_i$

The local observable set $\mathcal{V}_i$ from (3.27) may vary at each time step, based on the relative sensing and communication graphs $\mathcal{G}^{\text{s}}$ and $\mathcal{G}^{\text{c}}$ at the given time. The DPE modifies the augmented state vector $\overline{x}_i^{\mathrm{ref}}$ and its associated covariance $\overline{P}_i$ if the set $\mathcal{V}_i$ has changed over time.

A new spacecraft j is added to $\mathcal{V}_i$ at time t if a measurement of the new spacecraft becomes available at the new time step. A new measurement becomes available to an agent either when (i) the agent itself or one of its communication neighbors detects a new spacecraft or (ii) the measurement becomes available through the addition of a new communication link. The DPE waits for two consecutive pose measurements, such that the velocities of the new spacecraft are computed by numerical differentiation of the two pose measurements.

DPE adds the new spacecraft states for $j \in \mathcal{V}_i$ to $\hat{\overline{x}}_i$ by adding a new state x_j^{ref} directly computed from the positions and velocities. A block column and a block row are added to the covariance matrix when initializing the state. Assuming that x_j^{ref} is independent of $\hat{\overline{x}}_i$ at the previous time state, the augmented covariance matrix is created by adding a new set of rows and columns with a prescribed specified initial uncertainty. The off-diagonals are zeros since x_j^{ref} and $\hat{\overline{x}}_i$ are independent.

The observer spacecraft may also stop estimating a spacecraft if a previously estimated spacecraft becomes unobservable. Depending on the application, the dynamics of the unobservable states may be propagated by the dynamics model without the measurement updates for a fixed maximum number of rounds. If a new measurement becomes available for the spacecraft before the maximum number of rounds, the measurement update is applied and the count is reset. The spacecraft state and associated covariance blocks are deleted if the count exceeds the specified maximum number.

The DPE algorithms explained in the above sections can be summarized in Algorithm 1.

Nonlinear Observability

Assuming that the swarm has limited sensing and limited communication, it is important to determine which subset of spacecraft in the swarm is observable. The observer system for the i-th spacecraft in terms of $\overline{x}_i$ is constructed from (3.44) and (3.48). As usual, the following observability analysis assumes the deterministic nonlinear observer system.

$$\begin{cases} \dot{\overline{x}}_i = \overline{f}_i(\overline{x}_i) \\ \overline{y}_i = \overline{h}_i(\overline{x}_i) \end{cases} \tag{3.56}$$

Algorithm 1: The DPE Algorithm.

```
Result: Estimate $\hat{\bar{x}}_i^{\mathrm{ref}+}(t)$ and $\overline{P}_i^+(t)$
Initialize $\hat{\bar{x}}_i^{\mathrm{ref}+}(0)$ and $\overline{P}_i^+(0)$
while true do
    $\hat{\bar{x}}_i^{\mathrm{ref}+}(t-1), \overline{P}_i^+(t-1) = \mathrm{Reassign}(\hat{\bar{x}}_i^{\mathrm{ref}+}(t), \overline{P}_i^+(t))$
    Get measurements $\mathcal{M}_i^{\mathrm{a}}, \mathcal{M}_i^{\mathrm{r}}$
    for $j \in \mathcal{N}_i^{\mathrm{c}}$ do
        Exchange measurements $(\mathcal{M}_i^{\mathrm{a}}, \mathcal{M}_i^{\mathrm{r}})$ and $(\mathcal{M}_j^{\mathrm{a}}, \mathcal{M}_j^{\mathrm{r}})$
    end
    Collect measurements: $\bar{\mathcal{M}}_i^{\mathrm{a}} = \bigcup_{j \in \bar{\mathcal{N}}_i^{\mathrm{c}}} \mathcal{M}_j^{\mathrm{a}}, \bar{\mathcal{M}}_i^{\mathrm{r}} = \bigcup_{j \in \bar{\mathcal{N}}_i^{\mathrm{c}}} \mathcal{M}_j^{\mathrm{r}}$
    Update $(\hat{\bar{x}}_i^{\mathrm{ref}+}(t-1), \overline{P}_i^+(t-1))$ according to $\mathcal{V}_i$
    $\hat{\bar{x}}_i^{\mathrm{ref}-}(t), \overline{P}_i^-(t) = \mathrm{Time\ Update}(\hat{\bar{x}}_i^{\mathrm{ref}+}(t-1), \overline{P}_i^+(t-1))$
    $\Delta\bar{\chi}_i, \overline{P}_i^+(t) = \mathrm{Measurement\ Update}(\hat{\bar{x}}_i^-, \overline{P}_i^-, \bar{\mathcal{M}}_i^{\mathrm{a}}, \bar{\mathcal{M}}_i^{\mathrm{r}})$
    $\hat{\bar{x}}_i^{\mathrm{ref}+}(t) = \mathrm{Reset}(\Delta\bar{\chi}_i, \hat{\bar{x}}_i^{\mathrm{ref}-})$
end
```

We analyze (3.56) to determine its nonlinear observability. First, we define the terminology to make the discussion more concrete.

Definition 2. *Suppose $i, j \in \mathcal{A}$. We say agent j is observable to agent i if $j \in \mathcal{V}_i$ and x_j, a subset of state vector $\bar{x}_i$, is observable to i.*

Definition 3. *We say a set of agents $\mathcal{S}_i \subseteq \mathcal{A}$ is an observable set with respect to agent i if agent j is observable to agent i for all $j \in \mathcal{S}_i$*

Recall that $\bar{x}_i = \|_{j \in \mathcal{V}_i} x_j$ where $\mathcal{V}_i \subseteq \mathcal{A}$. Any agent $j \notin \mathcal{V}_i$ is not observable to agent i because it is not a part of the local dynamical system. Therefore $j \in \mathcal{V}_i$ is a necessary condition for agent j to be observable to agent i. Using this definition, we have the following proposition.

Proposition 1. *Suppose $j \in \bar{\mathcal{N}}_i^{\mathrm{c}}$ and $k \in \bar{\mathcal{N}}_j^{\mathrm{s}}$ for some $i \in \mathcal{A}$. Then agents j and k are observable to agent i.*

Proof. We have that $j, k \in \mathcal{V}_i$ by Definition 1, so x_j and x_k are both parts of the state $\bar{x}_i$ estimated by i-th agent in the nonlinear system (3.56). Now we consider the part of (3.56) pertaining to agents j and k. We define $w = [x_j; x_k]$.

$$\begin{cases} \dot{w} = f^p(w) \\ z = h^p(w) \end{cases} \tag{3.57}$$

where $f^p(w) = [f(x_j); f(x_k)]$ and $h^p(w) = [h^{\mathrm{a}}(x_j); h^{\mathrm{r}}(x_j, x_k)]$. The measurement model can be written as

$$h^p(w) = \begin{bmatrix} R(q_{L,I})^\top p_{j,L} + p_{L,I} \\ q_{j,I} \\ R(q_{j,I})R(q_{L,I})^\top (p_{k,L} - p_{j,L}) \\ q_{k,I} \otimes \left(q_{j,I}\right)^{-1} \end{bmatrix} \tag{3.58}$$

where $q_{L,I}$ and $p_{L,I}$ are known fixed parameters. The zeroth- and first-order Lie derivatives of h^p are given by

$$\mathfrak{L}^0 h^p(w) = h^p(w), \tag{3.59}$$

$$\mathfrak{L}^1_{f^p} h^p(w) = \nabla_w h^p(w) \cdot f^p(w). \tag{3.60}$$

Based on the Lie derivatives above, the observability matrix is defined as follows

$$O = \left\{ \nabla_w \mathfrak{L}^l_{f^p} h^p(w) \mid l \in \mathbb{N} \right\}. \tag{3.61}$$

The observability rank condition [85] states that if the observability matrix O is full column rank, the nonlinear system (3.57) is locally weakly observable. One can compute the gradient of the zeroth-order Lie derivative to get

$$\begin{aligned} &\nabla_w \mathfrak{L}^0 h^p(w) = \nabla_w h^p(w) \\ &= \begin{bmatrix} R(q_{L,I})^\top & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & I_4 & 0_{4\times3} & 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & 0_{4\times3} \\ -R(q_{i,I})R(q_{L,I})^\top & 0_{3\times3} & \Phi_1 & 0_{3\times3} & R(q_{i,I})R(q_{L,I})^\top & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & \Phi_2 & 0_{4\times3} & 0_{4\times3} & 0_{4\times3} & \Phi_3 & 0_{4\times3} \end{bmatrix} \end{aligned} \tag{3.62}$$

where $\Phi_1(p_{j,L}, p_{k,L}, q_{j,I})$ and $\Phi_2(q_{k,I})$ are some functions that are generally non-zero and $\Phi_3 = \Phi_3(q_{j,I})$ is given by

$$\Phi_3(q) = \begin{bmatrix} q_s I_3 + q_v^\times & -q_v \\ q_v^\top & q_s \end{bmatrix}. \tag{3.63}$$

The gradient of the first-order Lie derivative is

$$\begin{aligned} &\nabla_w \mathfrak{L}^1_{f^p} h^p \\ &= \begin{bmatrix} 0_{3\times3} & R(q_{L,I})^\top & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & * & \frac{1}{2}\Theta(q_{j,I}) & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ * & * & * & * & * & R(q_{j,I})R(q_{L,I})^T & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & * & * & 0_{3\times3} & 0_{3\times3} & * & \frac{1}{2}\Phi_3(q_{j,I})\Theta(q_{k,I}) \end{bmatrix} \end{aligned} \tag{3.64}$$

where an asterisk denotes some non-zero block element of matched dimensions. Because the nonlinear system (3.57) is infinitely smooth, O has an infinite number of rows in general. However, it is sufficient to show that a finite number of rows are linearly independent to determine local weak observability. With this in mind, we consider only the rows corresponding to the zeroth and first Lie derivatives.

$$O = \begin{bmatrix} \nabla_w \mathfrak{L}^0 h^p \\ \nabla_w \mathfrak{L}^1_{fp} h^p \end{bmatrix} \tag{3.65}$$

After applying block row elimination, O reduces to

$$\begin{bmatrix} R(q_{L,I})^\top & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & I_4 & 0_{4\times3} & 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & 0_{4\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & R(q_{j,I})R(q_{L,I})^\top & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & 0_{4\times3} & 0_{4\times3} & 0_{4\times3} & \Phi_3(q_{j,I}) & 0_{4\times3} \\ 0_{3\times3} & R(q_{L,I})^\top & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & \frac{1}{2}\Theta(q_{j,I}) & 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & 0_{4\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times4} & 0_{3\times3} & 0_{3\times3} & R(q_{j,I})R(q_{L,I})^\top & 0_{3\times4} & 0_{3\times3} \\ 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & 0_{4\times3} & 0_{4\times3} & 0_{4\times3} & 0_{4\times4} & \frac{1}{2}\Phi_3(q_{j,I})\Theta(q_{k,I}) \end{bmatrix}. \tag{3.66}$$

Given $||q|| = 1$, $R(q)$, $\Phi_3(q)$, and $\Theta(q)$ have full column rank. Therefore O has full column rank for arbitrary $j \in \bar{\mathcal{N}}_i^{\mathrm{c}}$ and $k \in \bar{\mathcal{N}}_j^{\mathrm{s}}$. The observability rank condition [85] tells that the nonlinear system from (3.57) is locally weakly observable. □

Finally, we arrive at the following theorem.

Theorem 1. *Suppose the detected set of agents $\mathcal{V}_i \subseteq \mathcal{A}$ for agent i is defined as in* (3.27). *Then $\mathcal{V}_i$ is the largest observable set in $\mathcal{A}$.*

Proof. Suppose $j \in \bar{\mathcal{N}}_i^{\mathrm{c}}$. Proposition 1 implies that $\bar{\mathcal{N}}_j^{\mathrm{s}}$ is an observable set with respect to the agent i. Moreover since $\mathcal{V}_i$ is defined as the union of all $\bar{\mathcal{N}}_j^{\mathrm{s}}$ over $\forall j \in \bar{\mathcal{N}}_i^{\mathrm{c}}$, $\mathcal{V}_i$ is also an observable set with respect to agent i. Recall that $j \in \mathcal{V}_i$ is a necessary condition for agent j to be observable to agent i because it has no information on $l \in \mathcal{A} \setminus \mathcal{V}_i$. Therefore, we conclude that $j \in \mathcal{V}_i$ is a necessary and sufficient condition for agent j to be observable to agent i. □

Theorem 1 states that all the detected sets of spacecraft $\mathcal{V}_i$ are observable in this problem formulation. Moreover, no other spacecraft $j \in \mathcal{A} \setminus \mathcal{V}_i$ is observable to the i-th spacecraft, given the measurement models and the one-hop communication limitation assumed in this problem. In other words, the DPE algorithm estimates the states for all the agents in the largest local observable subset $j \in \mathcal{V}_i$.

3.4 Consensus Estimation of Swarm Reference Frame

This section details the Swarm Reference Frame Estimation (SRFE) algorithm. Sophisticated motion planning algorithms typically require a common local reference (e.g., LVLH frame); however, finding such a reference frame is a non-trivial estimation task. All of the spacecraft in the swarm must have an estimate, and the swarm must reach a consensus on the common local reference frame. We apply an information consensus filter [9], [10], which is a decentralized algorithm where a sensor network co-estimates a state vector using the consensus algorithm [86].

In the development of the SRFE algorithm, we make the following assumptions:

- the communication graph $\mathcal{G} = (\mathcal{A}, \mathcal{E}^{\mathrm{c}})$ is undirected and connected at each time step;
- the subset of agents in the swarm has the measurements of the absolute pose of the reference spacecraft;
- the degree of the communication graph is upper bounded by a finite bound $d^{\max}$. That is $\mathrm{Card}(\mathcal{N}_i^{\mathrm{c}}) < d^{\max}$ for all $i \in \mathcal{A}$ for some $d^{\max} < +\infty$.

The assumption that the communication graph is undirected may be relaxed so long as the graph is balanced [10], [86]. To estimate the common LVLH frame, the state that needs to be estimated is the absolute position and velocity of a reference spacecraft in the ECI frame. Denote ξ to be the reference spacecraft translational state where $\xi = [p_{L,I}; v_{L,I}]$ and $p_{L,I}$ and $v_{L,I}$ are the position and velocity of the reference spacecraft in the ECI. Suppose a subset of spacecraft $\mathcal{W} \subseteq \mathcal{A}$ measures the absolute pose of the reference spacecraft. These absolute pose measurements may be obtained by combining GPS and relative pose measurements, which are assumed to be available for the DPE. Then, the discrete-time dynamics for the whole swarm are given by the following set of equations

$$\xi(t+1) = f^s(\xi(t)) + w^s, \quad t = 1, 2, \dots \qquad \xi(0) = \xi_0 \tag{3.67}$$

$$\eta_i(t) = H^s \xi + \psi_i^s, \quad i \in \mathcal{W}, \tag{3.68}$$

where $H^s = [I_3 \ \ 0_{3\times3}]$ is the absolute measurement model for measurement η_i. Propagation is modeled by a nonlinear function f^s. The process and measurement noises are denoted as $w^s \sim \mathcal{N}(0_{6\times1}, W^s)$ and $\psi_i^s \sim \mathcal{N}(0_{3\times1}, \Psi^s)$, respectively, and they are assumed to be independent. We define the estimate of ξ by $\hat{\xi}_i$ for a spacecraft $i \in \mathcal{A}$. The objective of the SRFE is to estimate $\hat{\xi}_i$ in an optimal fashion for all the spacecraft. To this end, we apply the information consensus filter [9] to the distributed dynamical system given by (3.67)) and (3.68).

The SRFE first computes the proposal information vector u_i^0 and the proposal information matrix U_i^0, according to (3.71) and (3.72). N denotes the size of the swarm $N = \text{Card}(\mathcal{A})$. This is computed on each spacecraft $i \in \mathcal{A}$ from the prior estimate $\hat{\xi}_i^-$ and the information matrix J_i^-. Next, the swarm communicates the consensus proposals to its neighbors and iteratively applies consensus K times. For each iteration k, the SRFE uses the consensus to compute a posterior information vector and information matrix according to (3.73) and (3.74), respectively. The consensus coefficient ϵ must satisfy a stable upper bound $\epsilon < 1/d$ where $d(\mathcal{G}^c)$ is the maximum degree of the communication graph. By the assumption that the communication graph has a finite degree, we have $d(\mathcal{G}^c) < d^{\max}$. Then, we choose ϵ such that $\epsilon < 1/d^{\max}$ to guarantee convergence. *A posteriori* information state and information matrix are computed according to (3.75) and (3.76). The algorithm is modified such that it uses nonlinear dynamics for time propagation of state. The Jacobian of $f^s(\xi)$ around $\hat{\xi}_i$ is defined to be $F_i = \left.\frac{\partial f^s(\xi)}{\partial \xi}\right|_{\xi=\hat{\xi}_i}$ and this is used for the covariance time propagation. The SRFE algorithm is summarized in Algorithm 2.

The SRFE has multiple properties that make it advantageous for the common LVLH estimation. First, the information consensus filter asymptotically approaches the optimal centralized estimate as $K \to \infty$, assuming the dynamical system is linear [9]. The optimal centralized estimate refers to the Kalman filter solution given that the centralized nodes have access to all the measurements (3.68), where f^s in (3.67) is linear. In practice, it is known that the information consensus filter achieves near-optimal value even if K is small [9]. The algorithm is strictly local and decentralized, such that the spacecraft only requires local information exchange. Each of the agents has an estimate of the reference trajectory even if some of the spacecraft do not make a direct measurement. Also, this approach is agnostic to whether the reference spacecraft or target is cooperative or uncooperative, so long as some of the spacecraft in the swarm can measure the absolute position of the reference in the ECI frame. In addition, the LVLH estimation has a unique requirement that the reference trajectory obeys the orbital dynamics. This is because the relative orbital dynamics such as HCW assume that the reference trajectory follows the modeled

Algorithm 2: The SRFE Algorithm.

```
Result: Estimate $\hat{\xi}_i^+$ and $J_i^+$ for each $i \in \mathcal{A}$
Input: $\hat{\xi}_i(0) = \hat{\xi}_{i0}$, $J_i(0) = J_{i0}$
while true do
    Propagate dynamics
        $\hat{\xi}_i^-(t) = f^s(\hat{\xi}_i^+(t-1))$    (3.69)
        $J_i^-(t) = \left(F_i(J_i^+(t-1))^{-1}F_i^\top + W^s\right)^{-1}$    (3.70)
    Get measurements $\eta_i$
    Compute consensus proposal vector $u_i^0$ and matrix $U_i^0$
        $u_i^0 = \frac{1}{N}J_i^-(t)\hat{\xi}_i^- + H^{s\top}\Psi^s\eta_i$    (3.71)
        $U_i^0 = \frac{1}{N}J_i^-(t) + H^{s\top}\Psi^s H^s$    (3.72)
    Perform consensus on $u_i^0$ and $U_i^0$
    for $k = 1$ to $K$ do
        Communicate $u_i^k$ and $U_i^k$ to all neighbors $j \in \mathcal{N}_i^{\mathrm{c}}$
        Update:
            $u_i^k = u_i^{k-1} + \epsilon \sum_{j \in \mathcal{N}_i^{\mathrm{c}}} (u_j^{k-1} - u_i^{k-1})$    (3.73)
            $U_i^k = U_i^{k-1} + \epsilon \sum_{j \in \mathcal{N}_i^{\mathrm{c}}} (U_j^{k-1} - U_i^{k-1})$    (3.74)
    end
    Compute a posteriori state and information matrix
        $\hat{\xi}_i^+(t) = (U_i^K)^{-1}u_i^K$    (3.75)
        $J_i^+(t) = NU_i^K$    (3.76)
end
```

dynamics. The SRFE algorithm estimates the orbital states of an actual orbiting body; therefore, an unbiased estimate of ξ will also approximately satisfy the orbital dynamics.

3.5 Validation of DPE and SRFE by Numerical Simulations

In the following three subsections, we discuss how the performance of the DPE and the SRFE was verified. Using a satellite inspection mission scenario as an example, we first illustrate the estimator convergence. Second, we quantitatively compare the computational time and the estimation error for an increasing number of spacecraft in the swarm. Finally, the DPE was implemented in a robotic experiment using Caltech's robotic spacecraft simulators called the Multi-spacecraft Testbed for Autonomy Research (M-STAR). The experiment considers time-varying relative sensing and communication graphs.

Numerical Simulation Example

This section verifies the performance of the DPE algorithm in a 6DOF numerical simulation example. We consider an example mission scenario in a circular, Low Earth Orbit (LEO) where three spacecraft cooperatively inspect one target spacecraft, such as a defunct satellite. The target is uncooperative in the sense that it does not communicate any information with the other spacecraft in the swarm. The three inspector spacecraft are placed in periodic, thrust-free relative spacecraft trajectories referred to as Passive Relative Orbits (PROs) [51] such that the centers of the PROs coincide with the target. Each spacecraft has an elliptical relative orbit with radii of 10×20 meters. All the spacecraft are initialized in the same orbital plane. The ground truth dynamics of each spacecraft were modeled using (nonlinear) Keplerian dynamics with no perturbations. The DPE uses the linearized HCW dynamics as the propagation model for the relative dynamics and the SRFE integrates the Keplerian dynamics to propagate the target state. For the attitude motion, the spacecraft follow a constant slew rate matching the negative of the mean motion, such that the attitude makes one rotation with one orbit. The absolute sensing uncertainties are selected to be 5 meters in position and 1 degree in attitude. The relative measurement uncertainties are 0.1 meters in position and 0.1 degrees in attitude. The simulation is run for one orbit. We assume that each spacecraft has the relative measurements of the target and the other spacecraft, and that all of the inspecting spacecraft communicate with each other. The sensing and communication graphs are fixed in this simulation example; the dynamic graphs are considered in the

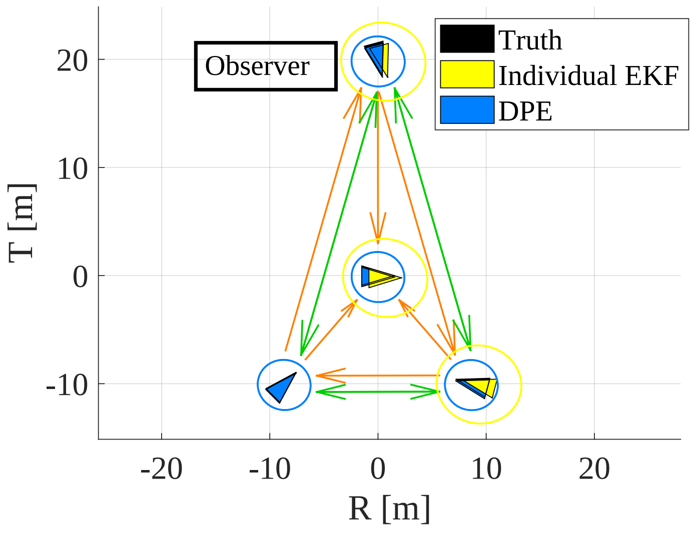


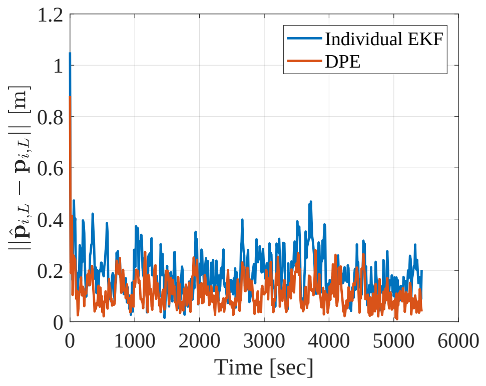
Figure 3.2: DPE and Individual EKF pose estimates and relative sensing and communication graphs.

robotic experiment. The relative sensing and communication graphs are shown in Fig. 3.2.

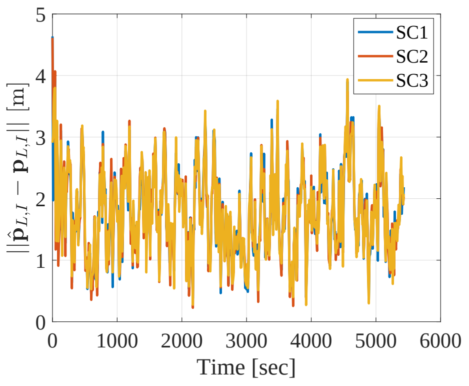
The inspector spacecraft first uses the SRFE to estimate the target spacecraft orbit state in the ECI frame. Since the target is uncooperative, the team of inspector spacecraft uses the SRFE to collectively estimate the target trajectory, which is then used to define the LVLH frame. Each inspector spacecraft uses its absolute measurement and the relative measurement of the target to create a pseudo-measurement of the target absolute position. The maximum degree of the communication graph is $d = 2$, so the consensus coefficient is selected to be $\epsilon = 0.49$. Each spacecraft uses the target estimate from the SRFE to define the LVLH frame. Next, the DPE estimates the formation pose with respect to the common LVLH frame. The absolute pose measurements are transformed from the ECI frame to the LVLH frame.

As a point of comparison, we also implement the Individual EKF where each spacecraft estimates poses only using its own absolute and relative measurements. For the estimation parameters such as measurement and process noise, the same parameters were used for the Individual EKF as those of the DPE. This represents the case where there is no communication between the spacecraft.

Fig. 3.2 shows the formation pose estimate obtained in the DPE after one orbit. The triangles represent the poses of the spacecraft in the swarm: black for ground truth, blue for the DPE estimate, and yellow for the Individual EKF estimate. The corresponding error ellipse represents the 99.7 percent confidence of the respective



(a) Individual EKF and DPE estimation error



(b) SRFE estimation error

Figure 3.3: Position estimation error of DPE and SRFE in 6DOF numerical simulations.

position estimates. Fig. 3.2 shows that the position error covariance for the DPE is much smaller than that of the Individual EKF. The local relative sensing graph of the DPE can be viewed as a pose graph. In this example, the DPE can estimate the states for all of the spacecraft in the swarm even though there are only two direct relative measurements available to each spacecraft.

In order to compare the Individual EKF and the DPE, the error metric was defined as the Euclidean norm of the position estimation error: $||\hat{p}_{i,L} - p_{i,L}||$. Fig. 3.3a shows the time history of the position estimation error for an inspector spacecraft estimated by another inspector spacecraft. The position error for the DPE is smaller compared to that of the Individual EKF at a steady state. Fig. 3.3a also shows that the estimate converges quickly.

Fig. 3.3b shows the performance of the SRFE algorithm for each of the three inspector spacecraft. The figure plots the error metric $||\hat{p}_{L,I} - p_{L,I}||$ which is the norm of the position estimation error of the reference spacecraft with respect to the inertial frame. The figure shows that the SRFE estimates quickly converge to the true trajectory for all the spacecraft. The high correlation between the three estimate errors is due to the fact that the three estimates converge to a commonly agreed estimate.

Scalability Analysis

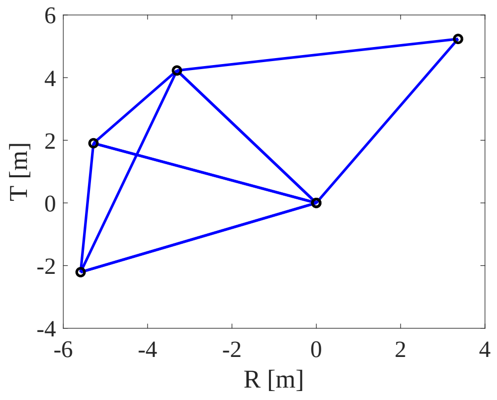
We verify the scalability of the DPE in numerical simulations for an increasing number of spacecraft in the swarm. The swarm sizes considered are 5, 100, 150, 200, 250, and 300 spacecraft. For each simulation, we study the estimation accuracy and the computational time of the DPE on each spacecraft. The performance of the DPE is compared against two other filters: the Centralized EKF, which has access to all of the measurements by all of the spacecraft, and the Individual EKF, where each individual spacecraft estimates the states only using its own measurements. The same parameters as those of the DPE were used for the Individual EKF and the Centralized EKF, except for the assumption on the communication. This scalability analysis does not include the computational time of obtaining the measurements. Because the scalability of the SRFE algorithm is established in prior work [9], [10], we assume the absolute reference state of the SRFE is given for this section only.

Given the relative positions, the relative velocities are selected such that all of the spacecraft have a concentric PRO. The relative measurements are obtained as described by (3.34). The simulation is run for 3,000 sec, which is approximately half of an orbit.

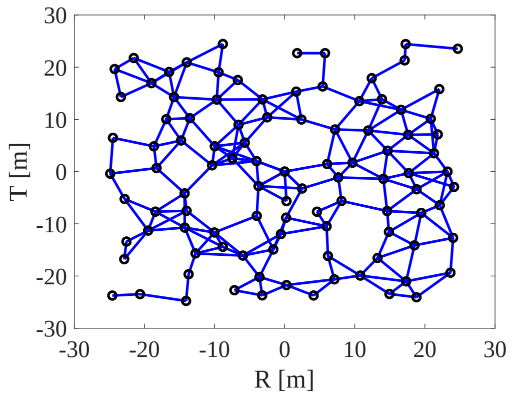
The relative positions of spacecraft are initialized randomly. We specify a minimum separation distance between spacecraft. The spatial density of the swarm is kept constant for a varying number of spacecraft. The graph edges between spacecraft are created if the Euclidean distance between each pair of spacecraft is below the detection threshold. The relative sensing and communication networks are given by the same graph. We enforce a maximum degree on each agent by pruning edges off the nodes with too many edges. A limited degree physically corresponds to each spacecraft having a restricted number of communication links. With this assumption, the DPE has a bounded number of elements in the augmented state vector regardless of the total number of spacecraft in the swarm. Moreover, we ensure that the communication graph is fully connected in all of the scenarios considered. Fig. 3.4a through 3.4c show the example graphs for 5, 100 and 300 spacecraft.

The performance of DPE is compared against the Centralized and Individual EKFs. The Centralized EKF is a global observer which has access to all of the measurements available in the swarm. While the Centralized EKF is prohibitive for large formations in terms of communication and computation, it represents the best possible estimate given all the information available in the network. The performance of DPE, the Centralized EKF, and the Individual EKF are compared in Fig. 3.5a and 3.5c.

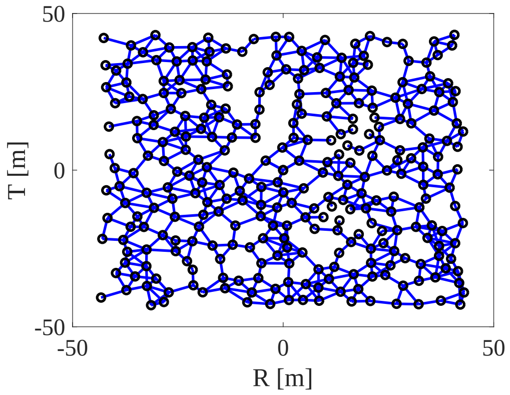
Fig. 3.5a shows the standard deviation of the position estimation error for different numbers of spacecraft in the swarm. The position estimation error, which is computed once the steady state has been achieved, is the time-averaged and



(a) 5 spacecraft



(b) 100 spacecraft



(c) 300 spacecraft

Figure 3.4: The spatial distribution of teams of 5, 100, and 300 spacecraft in the LVLH frame and their connectivity.

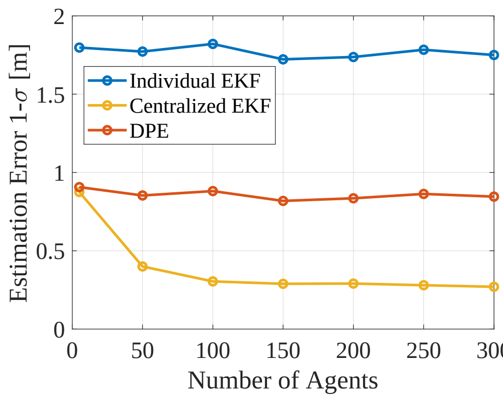
swarm-averaged standard deviation of the absolute position estimate. The position estimation error has significantly improved compared to the Individual EKF case. It is not surprising that multi-agent coordination improves estimation accuracy compared to a baseline without information exchange, but the amount of improvement is illuminating. The fact that estimation accuracy improves so much by including measurements only from the immediate neighborhood implies that the majority of all the relevant information is concentrated in the neighborhood of the agent. In other words, the incremental information gain due to an additional spacecraft is diluted by uncertainties accumulated over the relative sensing graph hops.

Fig. 3.5a also shows that the DPE estimation is not as accurate as that of Centralized EKF. This is because DPE only uses information from the local observable subset. However, this raises a follow-up research question: is there a localized and scalable algorithm that coincides with the centralized optimal solution? Later in Chapter 6, we develop a new algorithm to address this optimality gap.

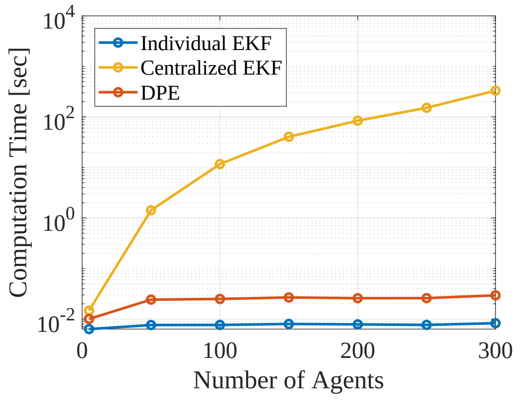
Fig. 3.5b shows the computation time required by each spacecraft for the DPE, the Centralized EKF, and the Individual EKF. For DPE and the Individual EKF, the computation time is averaged over all of the spacecraft in the swarm. The computation time for DPE remains constant as the number of spacecraft in the swarm increases. This is expected as DPE is a local algorithm, and the graphs have a fixed degree; therefore, the size of the state to be estimated is bounded. Fig. 3.5c compares the number of spacecraft estimated by each estimation algorithm. For DPE, this is the number of spacecraft that is included in the local relative sensing graph. For the Individual EKF, this is the size of the relative sensing neighborhood. The bars represent the minimum and maximum number of neighbors among all of the spacecraft in the swarm. Note that DPE has a significantly larger (but bounded) number of spacecraft observed by each spacecraft. On average, the number of estimated spacecraft increases by more than a factor of two. The maximum number of spacecraft observed is high for DPE. This occurs at the part of the swarm where many spacecraft are close to each other. In this example, we restricted the communication and relative sensing graphs to have a maximum degree of $d^{\max} = 6$. This ensures that the number of nodes in the local relative sensing graph is bounded. The impact of the restriction is also empirically confirmed by the fact that the maximum size of the local sensing graph does not grow as the overall swarm size increases. Note that this maximum size can be reduced by choosing a smaller upper bound for the maximum degree of the graph. Also, the simulation did not include the time required to obtain the pose measurements. While the computational complexity for vision-based pose extraction likely has a larger constant than the DPE algorithm, it is also constant with respect to the swarm size.

3.6 Experimental Validation Using Spacecraft Simulators

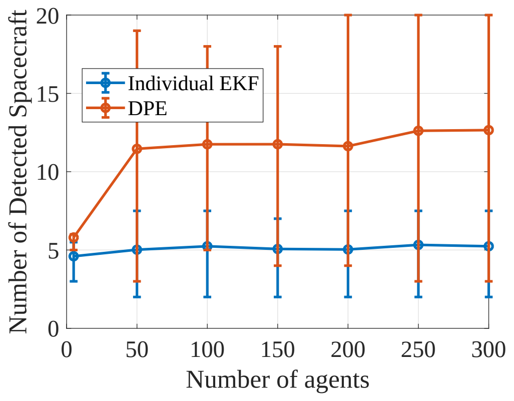
The DPE algorithm was implemented on-board Caltech's robotic spacecraft simulators, the M-STAR. In the experiment, each spacecraft obtained relative pose measurements using a monocular camera and a computer vision algorithm. This experiment tested the real-time performance of the DPE with time-varying graphs, where the relative sensing graph changed depending on which spacecraft were in the camera's field-of-view (FOV). Because the experiment was constrained to planar motions, the DPE algorithm formulation was modified from the 6-DOF to its 3-DOF analog. With the 3-DOF formulation, the state for the i-th spacecraft is selected to be $x_i = [p_{i,L}; v_{i,L}; \theta_{i,L}; \omega_{i,L}]$ where $p_{i,L}$ and $v_{i,L}$ denote 2D position and velocity



(a) Position error averaged over agents.



(b) Computation time averaged over agents.



(c) Number of other vehicles in proximity detected (average, minimum, and maximum across all agents).

Figure 3.5: The performance of DPE as the number of vehicles in swarm increases from 5 to 300 spacecraft.

vectors with respect to the LVLH frame and $\theta_{i,L}$ and $\omega_{i,L}$ denote the 1D attitude and rotation rate of the spacecraft. The time-varying communication graph was simulated by masking part of the available communication packages.

Experimental Setup

Each spacecraft simulator used air-bearing and on-board air-based thrusters to simulate frictionless dynamics similar to that in space. We used the 3-DOF configuration where the simulators translate and rotate only in a planar motion. Each spacecraft was equipped with a Jetson TX2 computer, a monocular camera with a high FOV lens, and ArUco visual markers [87] on each side as seen in Fig. 3.6. Example images of the detected markers are shown in Fig. 3.7.

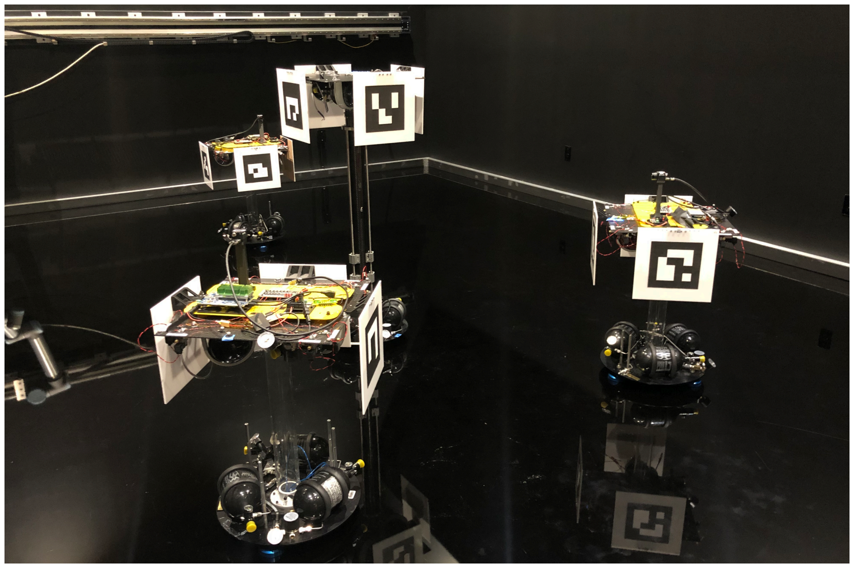


Figure 3.6: M-STAR with the ArUco markers on the flat epoxy floor.

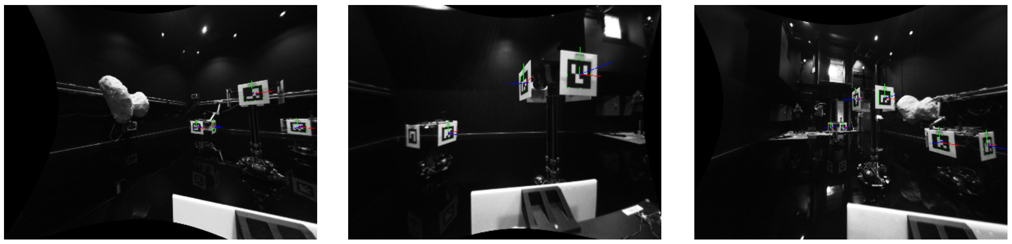


Figure 3.7: Detected markers as seen from the cameras on the spacecraft simulators.

Each spacecraft simulator used a thruster-based controller to follow a prescribed HCW trajectory using the ground truth pose from the motion capture system. We implemented the same formation as the numerical simulation case, where three inspector spacecraft (labeled 1 through 3) orbiting around an uncooperative target spacecraft (labeled 4). The attitude dynamics again assume that the spacecraft rotate at the rate of the negative of the mean motion. The resulting trajectories were such that each spacecraft measured at least two spacecraft persistently throughout the trajectory. The time of each orbit was scaled to 262 sec, short enough so that the experiment would be completed without depleting compressed-air for the thrusters used to follow the trajectory.

The images from the monocular camera were processed on board each spacecraft using a standard computer vision algorithm [87] to detect the ArUco markers and estimate their full pose. While the ArUco-based algorithm does not address some of the relative pose estimation challenges that result from using electro-optical sensors in a space environment [88], there exist various other vision-based relative pose estimation algorithms [89]–[91] that implement application-specific solutions. Because DPE has a decentralized architecture for abstract communication and sensing networks, DPE can be used in conjunction with most vision-based algorithms available. For this reason, we simplified the experiments by using the ArUco markers and focus on the aspects relevant to DPE.

The absolute pose measurement $[p_{i,L}; \theta_{i,L}]$ is given from a motion capture system. The motion capture system is typically used to provide the ground truth position of the spacecraft to a sub-millimeter level accuracy; therefore, additional noise was artificially added to make the absolute measurement more realistic. The noise was modeled as zero-mean Gaussian with 0.2 meters standard deviation in translation and 2 degrees standard deviation in rotation.

The on-board computers sent and received information over a wireless network to simulate inter-spacecraft communication. In order to test the DPE's performance under a time-varying communication graph, some shared information was artificially masked to simulate time-varying inter-satellite communication links. Specifically, we prescribed the edges of the undirected communication graph to be

$$\mathcal{E}^{\mathrm{c}} = \begin{cases} \varnothing, & \text{if } t < 20 \text{ sec} \\ \{(1,2),(1,3)\}, & \text{if } 20 \text{ sec} \le t < 50 \text{ sec} \\ \{(1,2),(1,3),(2,3)\}, & \text{if } 50 \text{ sec} \le t. \end{cases} \tag{3.77}$$

The measurement graph was also allowed to vary between time steps, depending on whether a neighbor spacecraft was visible in the FOV or not.

Table 3.1 includes information about the parameters used by the DPE in the experiment. The relative measurement covariance is scaled with the squared Euclidean distance between spacecraft centers to model a larger uncertainty for relative measurement at a large separation distance.

Software Architecture

The Robot Operating System (ROS) was used for interfacing with the sensors and communicating measurements across robots. A block diagram of the software architecture can be seen in Fig. 3.8. After obtaining the pose of the ArUco markers in the camera's FOV, each marker pose was transformed into the corresponding spacecraft body frame. This frame transformation was estimated by an extrinsic calibration procedure described in next section. The relative and absolute measurements and their covariances were then communicated to the prescribed neighbors over the

Table 3.1: Parameters specified in the DPE.

Parameters	Values
Process noise std dev	
Translation	0.03 m, 0.01 m/sec
Attitude	0.2 deg, 0.05 deg/sec
Absolute measurement std dev	
Translation	0.2 m
Attitude	2 deg
Relative measurement std dev	
Translation	0.1 m
Attitude	10 deg
Initial uncertainty std dev	
Translation	2 m, 0.03 m/sec
Attitude	15 deg, 2 deg/sec
Control interval	1 sec

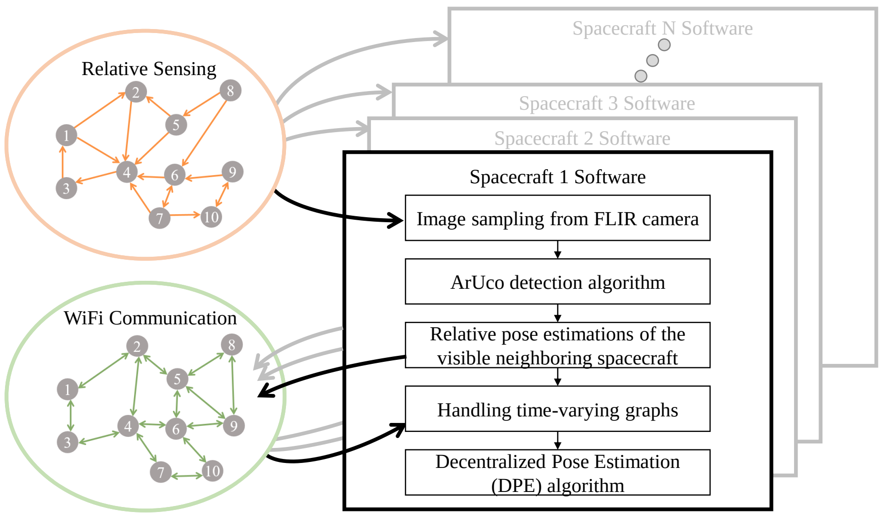

Relative Sensing
WiFi Communication
Spacecraft N Software
Spacecraft 3 Software
Spacecraft 2 Software
Spacecraft 1 Software
Image sampling from FLIR camera
ArUco detection algorithm
Relative pose estimations of the visible neighboring spacecraft
Handling time-varying graphs
Decentralized Pose Estimation (DPE) algorithm

Figure 3.8: DPE software architecture.

wireless network. When the set of observable agents changed due to time-varying measurement or communication graphs, the method described in Section 3.3 was used to modify the augmented states and covariances.

Camera Calibration

We performed intrinsic and extrinsic calibrations for each pairing of a monocular camera and a spacecraft simulator. For the intrinsic calibrations, a pinhole camera model with 6 radial distortion coefficients, 2 tangential distortion coefficients, and 4 thin prism distortion coefficients was used to model the high FOV lens camera.

A checkerboard pattern was used as a visual target, and a motion capture system collected the camera and the target poses.

Extrinsic calibration was performed to determine the relative pose between the camera frame and the spacecraft body frame. The camera was rigidly attached to the spacecraft body. The camera frame was defined such that its origin coincided with the camera optical center. To perform this calibration, we first used a monocular camera and a computer vision algorithm to extract the relative pose of the ArUco targets. Sufficient pose measurements were collected while the ArUco target was moved around the workspace. At each time step, the relative pose between the spacecraft and the ArUco target was also obtained using the motion capture system. We solved a least-square optimization to retrieve the camera to spacecraft body calibration.

For each relative sensing edge, the relative pose measurement error was computed as detected pose minus the ground truth from the motion capture system. After camera calibration, the standard deviation of relative measurement error was 3.0 cm along the line of sight, 2.3 cm in the in-plane perpendicular direction, and 2.9 deg for attitude.

Table 3.1 includes information about the parameters used by the DPE in the experiment. The relative measurement covariance is scaled with the squared Euclidean distance between spacecraft centers to model a larger uncertainty for relative measurement at a large separation distance.

Experimental Results

The three inspector spacecraft, shown in Fig. 3.6, ran the DPE algorithm in synchronized rounds once every 1 sec, of which the ArUco detection took approximately 0.1 sec. Measurement collection and communication were allotted 0.3 sec. After the measurement was collected, the DPE step took approximately 10^{-3} sec. Fig. 3.9 shows the varying sensing and communication network used by Spacecraft 2 for estimation in the experiments. The orange edges correspond to the relative sensing graph, while the green edges correspond to the communication graph. Initially, there is no communication between any of the spacecraft, and Spacecraft 2 only has access to the measurements it collects: relative measurements obtained using ArUco pose estimation and a measurement of its own absolute pose. At $t = 20$ sec, Spacecraft 2 begins to communicate with Spacecraft 1. This allows Spacecraft 2 to have additional relative and absolute measurements. At $t = 50$ sec, Spacecraft

2 starts communicating with Spacecraft 3, adding more measurements to its graph. Fig. 3.10 shows the estimation error as a function of time during the experiment. A 2-σ uncertainty envelope is plotted around the error, using the covariance matrix. The dashed vertical lines indicate that new communication links are introduced at t = 20 sec and t = 50 sec. For every additional communication link, the covariance size decreases, confirming that added relative and absolute measurements provided by communication reduce uncertainty.

Fig. 3.10 also shows that, even when the communication graph remains unchanged, the measurement graph also changes due to some spacecraft entering and exiting the camera FOV. Time-varying relative sensing topologies are representative of realistic sensing constraints for a spacecraft swarm. For instance, at t = 37 sec an edge is lost and then regained at t = 40 sec. Fig. 3.10 shows that there is a temporary increase in the uncertainty during this period, followed by a decrease when the measurement is restored. This observation supports that additional relative measurements generally help the DPE reduce the estimate uncertainty. There are some other events where relative sensing edges are added and lost (after the second communication link is established at t = 50 sec), but the covariance did not change noticeably. This is likely explained by the added measurements available to Spacecraft 2 after the communication links to both Spacecraft 1 and 3 are established. These extra measurements provide more observation paths from Spacecraft 1 to Spacecraft 4, adding redundancy so that the loss of a single measurement is not as impactful. These redundant measurements from communication are another advantage of cooperative estimation using the DPE.

3.7 Chapter Summary

We present the Decentralized Pose Estimation (DPE) algorithm that solves the swarm localization problem for the formation flying spacecraft. The DPE considers *ad hoc* relative sensing and communication networks to determine a set of observable spacecraft and shares these spacecraft's measurements to jointly estimate their poses with respect to the LVLH frame at each time step. As a part of the DPE, the Swarm Reference Frame Estimation (SRFE) algorithm applies the information consensus filter to estimate the common LVLH frame in a decentralized fashion. The DPE is a local, decentralized algorithm that has a constant complexity with respect to the swarm size. Numerical simulations verify that the estimation errors of the DPE are improved compared to those for no cooperation cases and that the computation time remains constant as the swarm size increases. An experimental result using

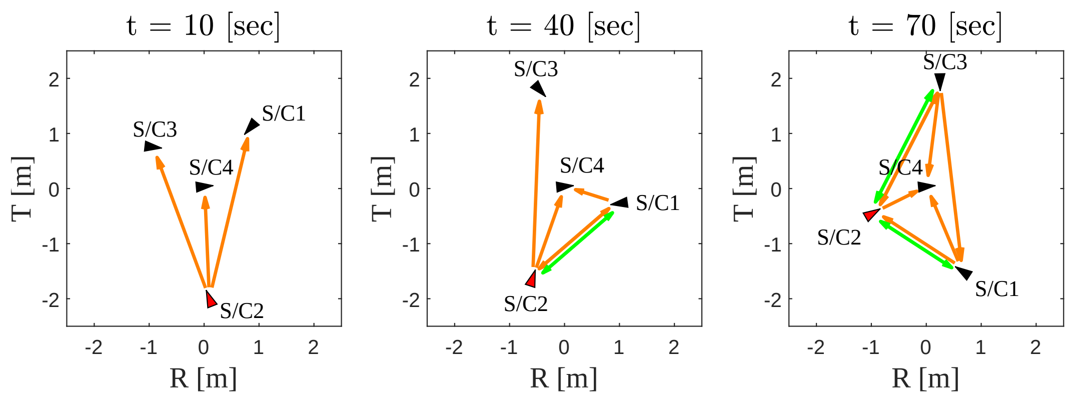


Figure 3.9: The relative sensing and communication edges as seen by the observer (Spacecraft 2) at different times during the experiment

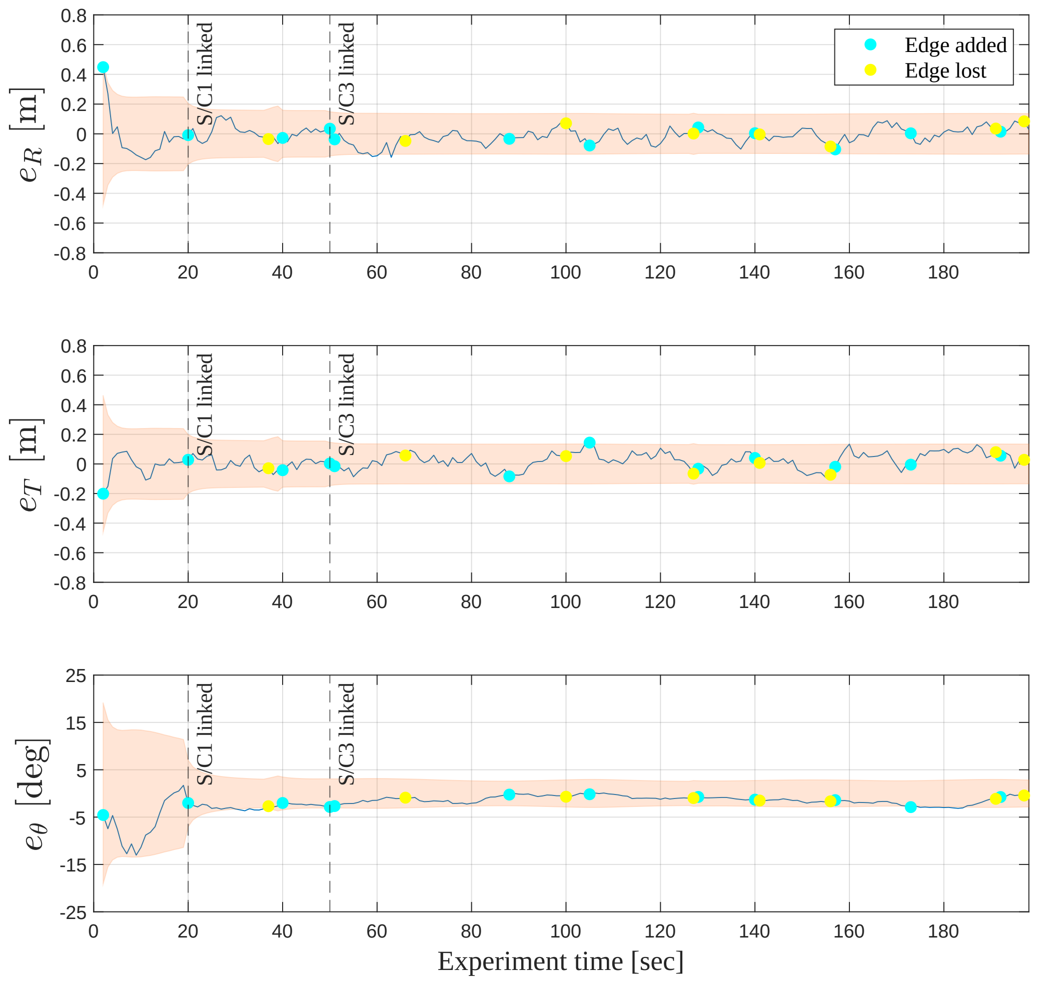


Figure 3.10: The estimation error of the target (Spacecraft 4) by the observer (Spacecraft 2).

air-bearing spacecraft simulators demonstrates good DPE performance using vision-based relative pose measurements with *ad hoc* networks.

While we demonstrated that DPE improves estimation accuracy through coordination with neighbors in the local subset, the optimality analysis (Fig. 3.5a in Section 3.5) also identified that the DPE estimate is not as accurate as the best possible estimation from a centralized, but not scalable, estimator. This insight motivated our follow-up work in Chapter 6 where we address the question: can we develop a distributed and localized estimation algorithm that computes *centralized optimal* estimation algorithm? In Chapter 6, we tackle this optimality gap by utilizing a new theoretical framework for Distributed Factor Graph Optimization.

Chapter 4

MULTI-SPACECRAFT SIMULTANEOUS ESTIMATION OF POSE AND SHAPE

This chapter contains material from the following publication:

[1] K. Matsuka, A. Santamaria-Navarro, V. Capuano, A. Harvard, A. Rahmani, and S.-J. Chung, "Collaborative pose estimation of an unknown target using multiple spacecraft," in *2021 IEEE Aerospace Conference*, IEEE, 2021, pp. 1–11. DOI: 10.1109/AERO50100.2021.9438352,

In this chapter, we tackle the second key technology with robotics in orbit; robust vision-based navigation. Previously in Chapter 3, the hardware experiment for DPE used vision to compute the relative pose of neighbor spacecraft, but it used fiducial markers which made the pose estimation problem much simpler. However, the target objects are often *unknown* and *uncooperative* in real on-orbit servicing missions; we do not know the geometry or appearance of the target *a priori*, and there is no information sharing between the target and observers. Vision-based navigation is more challenging in this case, as it requires more complex robotics perception techniques such as Simultaneous Localization And Mapping (SLAM). In addition, previous works in the literature focused on a single observer; leaving multi-agent observer problems largely unexplored.

In this chapter, we develop the Multi-spacecraft Simultaneous Estimation of Pose and Shape (MSEPS) algorithm. MSEPS is designed to track the pose of unknown, uncooperative spacecraft with multiple observers. MSEPS estimates CG of the target object using vision sensors on multiple chaser spacecraft. tackle the problem in the distributed sensor network paradigm where the team of chaser spacecraft can exchange information over the local inter-spacecraft communication links. MSEPS can be viewed as a multi-observer extension of SEPS algorithm [90].

We exploit the recent development in distributed estimation theory [9], [10] where the approximate solution of a minimum variance estimate given the global information is computed in a distributed, iterative fashion. We highlight that the framework for a distributed sensor network is distinct from that of the cooperative SLAM in terrestrial applications, which assumes infrequent, event-based information exchange

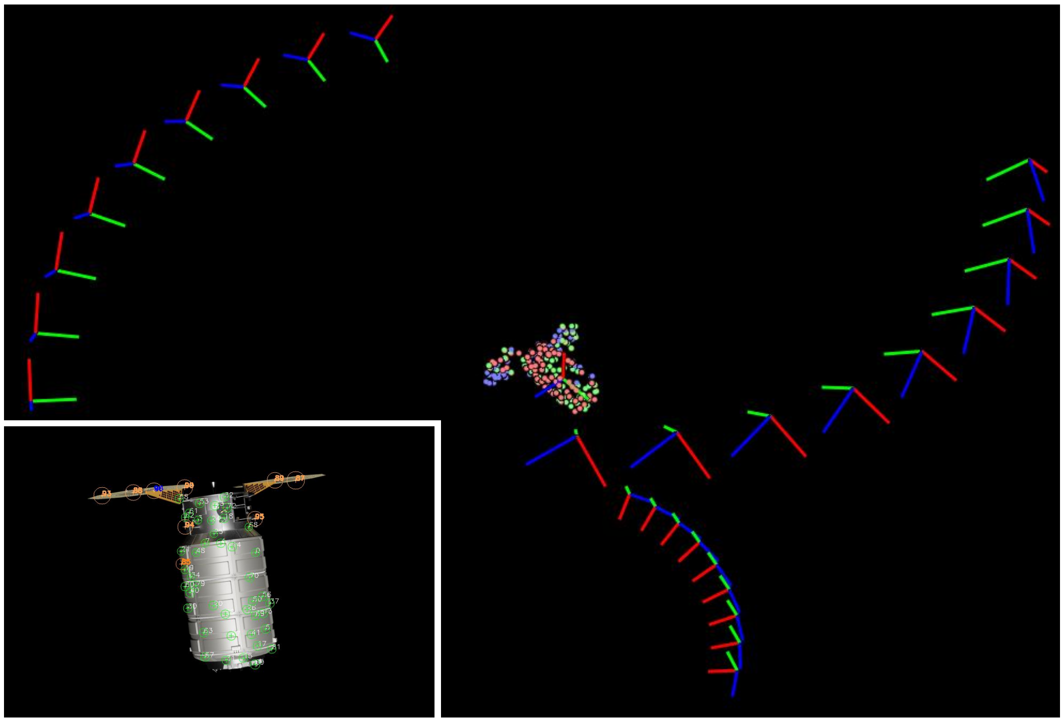
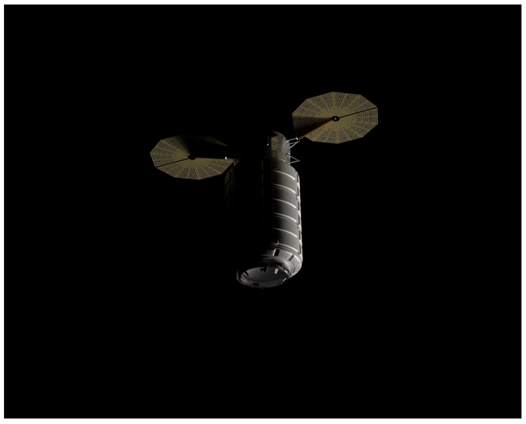


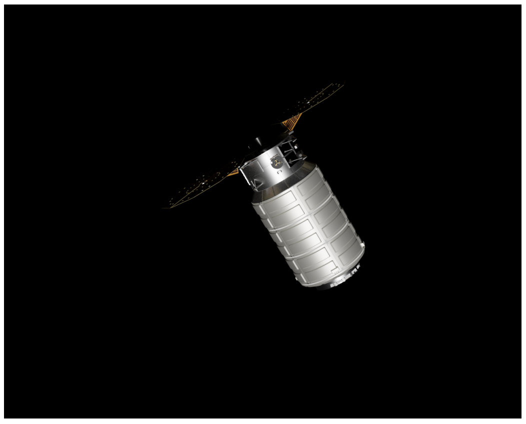
Figure 4.1: Target shape reconstructed from three cooperative spacecraft with monocular cameras. The bottom left image shows the keypoints seen by one of the chasers.

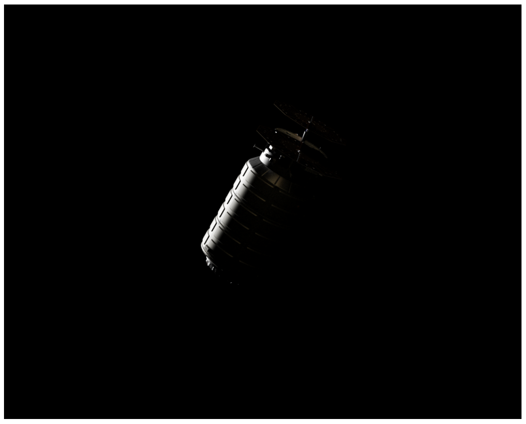
upon rendezvous. Distributed Kalman information filter, which poses clear advantages for multi-agent robotics perception, thus we adopt the same strategy in this work.

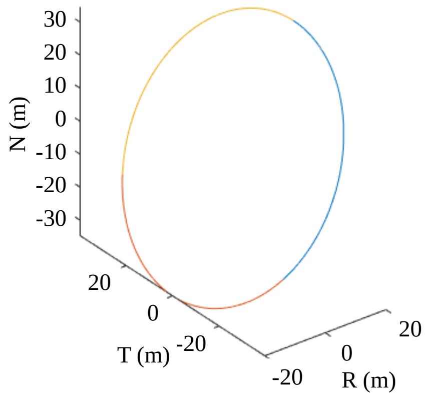
The contributions of this chapter are as follows: a) We present the first-of-the-kind multi-spacecraft algorithm architecture for the cooperative vision-based pose estimation of the uncooperative and unknown target, b) we apply the distributed estimation of sensor networks to develop the extended decentralized information filter for the MSEPS, we propose an improvement in dynamics update of the information matrix, leveraging the special structure that arises in the MSEPS problem, and d) we validate the algorithm architecture through numerical simulations of relative orbits, measurements, and inter-spacecraft communications.

The rest of the paper is organized as follows. The estimation problem is formally stated in Section 4.1. Section 4.2 discusses the overall architecture of the MSEPS, which includes keypoint extraction, matching, and optical flow components. Section 4.3 discusses the decentralized information filter algorithm, which is a back-end filter that fuses distributed sensing information. Section 4.4 presents the validation of the approach through a simulation setup. Finally, conclusions are drawn in


(a) View from Chaser 1


(b) View from Chaser 2


(c) View from Chaser 3



(d) Chasers' orbits with respect to the target at origin

Figure 4.2: Example images seen by chasers and their relative orbit with respect to the target.

Section 4.5.

4.1 Problem Statement

The use of multiple observers has multiple advantages. First, the chasers produce larger coverage of the target when they are placed in well spatially distributed relative orbits. The group of spacecraft can maintain a formation such that a persistent estimate can be produced. In this way, a virtual distributed camera system is created. This is of special interest in those cases where some of the chasers cannot have good visual conditions but receive reasonable updates from the other chasers. The observations from different perspectives also improve the depth estimation, convergence rate, and accuracy. Moreover, if one of the chasers fails during the mission, the inspection task can still be completed by the other chasers. We present a cooperative strategy that generalizes to an arbitrary number of the

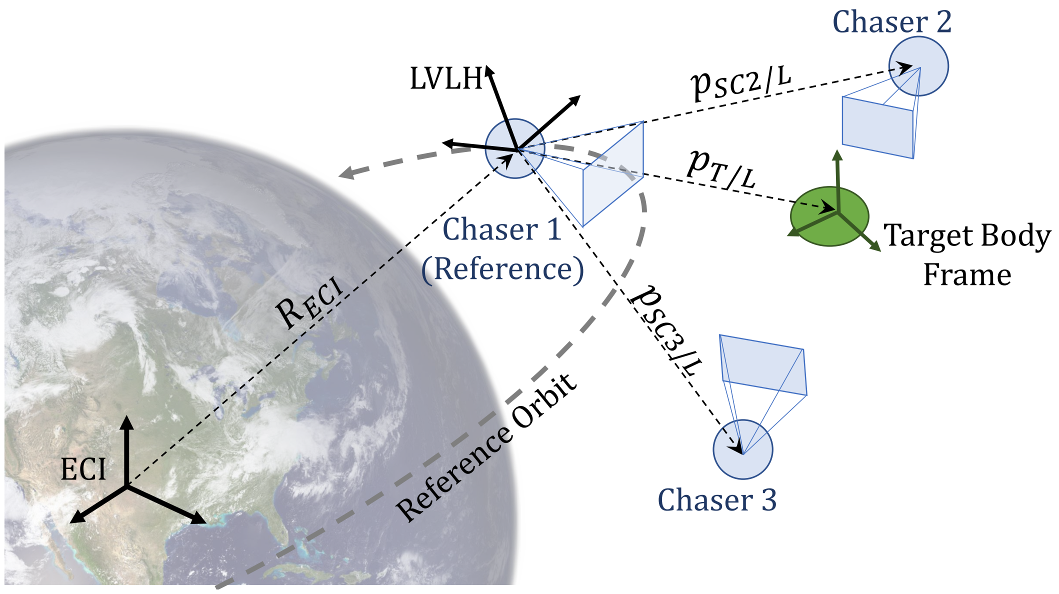


Figure 4.3: Local-Vertical Local-Horizontal (LVLH) frame, defined on the reference spacecraft, and target's and chasers' body coordinate frames.

chaser spacecraft.

Fig. 4.3 illustrates the conventions for the reference frames used in this chapter. The absolute orbital motion of each chaser and target is described with respect to the Earth-Centered Inertial (ECI) frame. One of the chasers is chosen as a reference spacecraft, which defines a local-vertical local-horizontal (LVLH) coordinate frame. The target body frame is defined such that its origin coincides with its center of gravity (CG), which is to be estimated by MSEPS. The orientation of the target frame is arbitrarily selected during the initialization since the target is unknown, and the attitude trajectory is described in terms of the relative attitude.

The MSEPS estimates the state vector

$$x = [x_T; l^1; \cdots; l^N]$$

where x_T is the target state and l^i is the landmark states for the i-th chaser. The target state is defined as $x_T \triangleq [p_{T/L}; v_{T/L}; q_{T/I}]$ includes the relative positions and velocities of the target expressed in the LVLH frame, and its attitude in ECI frame. The angular velocity of the target is not a part of the state vector. Instead, they are obtained from the optical flow module explained in Section 4.2. The i-th chaser's landmark state is defined as $l^i = [p^i_{1/T}; \cdots; p^i_{n_i/T}]$ where $p^i_{j/T}$ is the j-th landmark visible to the i-th chaser, expressed in the target body frame. The resulting nonlinear chaser system is written as

$$x_{Tk} = f_T(x_{Tk-1}, k) + w_{Tk}, \tag{4.1}$$

$$p^i_{j/T_k} = p^i_{j/T_{k-1}}, \quad \forall i \in \mathcal{A}, j \in \{1, \dots, n^i\}, \tag{4.2}$$

$$y_{i,k} = c_i(x_{Tk}, l^i_k) + v_{i,k}, \quad \forall i \in \mathcal{A}. \tag{4.3}$$

The nonlinear discrete-time target dynamics model f_T may include any relevant environmental forces and control actuation. In this chapter, we assume the coupling between the relative orbital mechanics and the attitude dynamics are negligible. The relative orbital propagation model includes the fully nonlinear equations for the Earth's point gravity and we assume torque-free attitude dynamics for the attitude propagation. The measurement $y_{i,k}$ are pixel coordinates of keypoints extracted from the image taken by the i-th spacecraft at k using the computer vision algorithms. The measurement model c_i projects the 3D landmarks to image plane and it is a function of the chaser pose (even though they are not included as the arguments to the function in (4.3) because these parameters are assumed to be known). The standard pinhole camera model is used for the camera projection model throughout this chapter. $w_{Tk} \sim \mathcal{N}(0, W_{Tk})$ and $v_{i,k} \sim \mathcal{N}(0, V^i_k)$ for $\forall i \in \mathcal{A}$ are process noise for target dynamics and measurement noise for each keypoint measurements, respectively.

Equations (4.1)-(4.3) can be equivalently written as a general nonlinear dynamical system, in terms of the full state vector x, as follows

$$x_k = f(x_{k-1}, k-1) + w_k, \tag{4.4}$$

$$y_k = c(x_k, k) + v_k. \tag{4.5}$$

where $w_k = [w_{Tk}; 0; \dots; 0]$ and $v_k = [v^1_k; \dots; v^N_k]$ are the augmented noise vectors. If this process was to be estimated in a centralized algorithm, one can design a straightforward nonlinear observer such as EKF. The challenge of the distributed system is that each measurement y_i is only available on the i-th chaser. The goal of the MSEPS is to approximate the minimum variance posterior estimate x^+_k using the local communication network in a distributed fashion.

To simplify the formulation, we make the following assumptions. Each chaser is equipped with a star tracker and a Global Navigation Satellite System (GNSS) receiver [92] and all chaser poses relative to the reference spacecraft are known. In order to have access to the real-time position of the reference spacecraft position, we assume an external estimation approach specifically designed for tracking

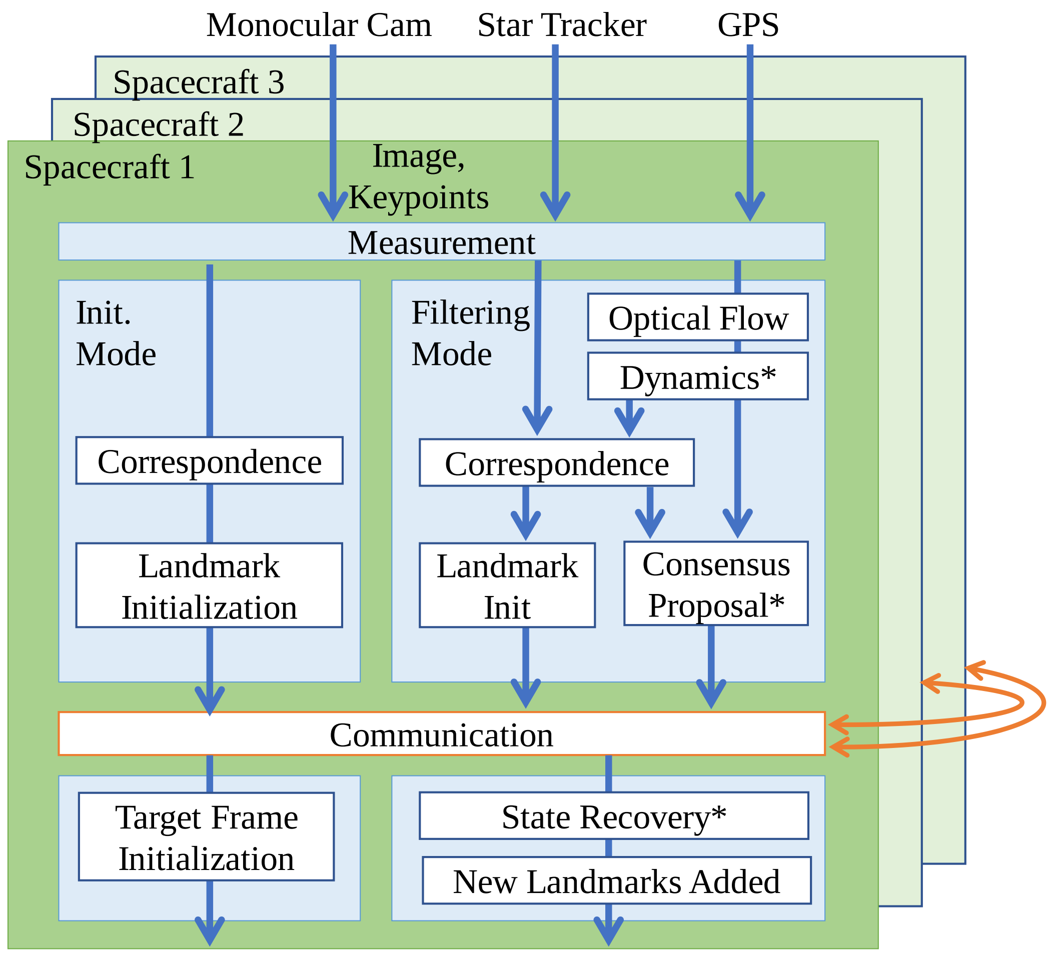


Figure 4.4: The collaborative pose estimation algorithm architecture. Each chaser has its own copy of the decentralized algorithm.

formation. Next, we assume that the target is not applying relative translational maneuvers or such maneuver is negligible compared to the assumed process noise. Finally, we assume the chaser spacecraft have a connected but possibly time-varying communication graph.

4.2 MSEPS Architecture Overview

The Multi-spacecraft Simultaneous Estimation of Pose and Shape (MSEPS) consists of multiple algorithm modules as shown in Fig. 4.4. This section describes each of these modules, except the decentralized back-end filter, which is discussed in more detail in Section 4.3.

The MSEPS can be primarily separated into *Initialization Mode* and *Filtering Mode* which are shown as two columns in Fig. 4.4. We assume that the target CG and geometry are unknown or only partially known prior to the Initialization Mode.

Therefore, Initialization Mode calculates both the initial set of landmarks and the target pose from a small batch of images. Also, the target body coordinate frame is initialized such that its origin is coincident with the estimated CG and it has an arbitrary orientation fixed with the target. During Filtering Mode, the state at the new time step is predicted by propagating the target dynamics and comparing it against the new observations. The Filtering Mode will continuously provide the relative pose estimate of the target such that a guidance and control module can use the product to proximity operation maneuvers in real-time. The modules with asterisks in Fig. 4.4 are part of the extended decentralized information filter described in Section 4.3.

The algorithms in each Initialization and Filtering Modes are also split between the *pre-communication*, *communication*, and *post-communication* steps, where the communication aggregates information shared across multiple spacecraft. The following sections visit each module and discuss its functionality.

Keypoint Extraction and Correspondence

The strategies for keypoint correspondences are different for Initialization and Filtering Modes. During the Initialization Mode, the two sets of extracted keypoints from two different epochs are matched by all-to-all, brute force matching, followed by ratio-test. In addition, we apply a random sample consensus (RANSAC) algorithm with a 5-point Stewenius algorithm to compute the 3D landmark position, which is discussed in the next section. During the Filtering Mode, the extracted keypoints are matched with projected landmarks. The search region is reduced by using the predicted keypoint location and covariance of the landmarks. We assume the standard pinhole camera model to describe the projection of the 3D landmarks onto the image plane.

Each chaser searches for correspondences only among images taken by itself and not across the multiple spacecraft. This architectural choice is motivated by a few reasons. First, finding correspondences of descriptors across multiple spacecraft requires communicating the set of descriptors, which increases the bandwidth. Second, even with feasible communication, the chance of detecting a correspondence is small across multiple spacecraft, given the large variation in the visual conditions when the chasers are spatially well distributed.

Target Frame Initialization

The primary function of the Initialization Mode is to initialize the target frame, which involves selecting the origin and orientation of the frame consistent among all chasers and defining all the detected inliers in the target frame. The target frame initialization involves three steps: individual initialization, communication, and transformation into a common frame.

First, individual spacecraft initialize the respective landmarks expressed in its camera frame prior to the communication step. Given the camera pose in ECI at each epoch is known from GNSS and the star tracker, the target pose trajectory is solved along with their landmarks. To solve this, the 5-point Stewenius algorithm [93] with random sample consensus (RANSAC) is used to determine inliers, followed by least-square optimization. With relative pose transformation from the chaser's camera to the reference's camera, the landmark positions are initialized in the reference spacecraft frame. At this time, the reference spacecraft also initializes the target frame by selecting an arbitrary attitude and a coarse estimate of the target CG. The coarse estimate of the target CG is defined as one of the landmark features observed.

Second, spacecraft exchange information in the communication step. The landmarks estimated from each chaser and the target initial frame from the reference spacecraft are shared with all the spacecraft. Finally, each spacecraft applies the pose transformation to obtain landmarks in the target reference frame.

Optical Flow

The optical flow module uses the sequence of images to compute the angular velocity in a similar way as done in the previous work [90]. When the initial frame and the inertia matrix of the target are unknown, direct observation of angular measurement has advantages such as simpler propagation of attitude quaternion and avoiding estimation of the inertia matrix. The optical flow may be obtained by classical methods such as Lucas-Kanade [94] or more recent methods such as using a convolutional neural network trained with sequences of images [95].

Communication

The information shared among neighbor spacecraft is different for Initialization and Filtering Modes. The communication module manages the information exchange between neighboring spacecraft. Given the edges in the communication graph $\mathcal{E}^k$, the communication link is established between $(i, j) \in \mathcal{E}^k$. We assume that the communication rate is higher than the estimation rate, i.e., there may be multiple communication exchanges at each filter time epoch.

Landmark Initialization

If there is a new keypoint correspondence that was not previously tracked, the landmark state is added to the filter. First, individual spacecraft detects new landmarks prior to the communication step. The new landmarks' 3D positions are broadcast to all the spacecraft, and finally, all the new landmarks from the epoch are added to the state. This step is implemented in a similar way to SEPS [90].

4.3 Extended Decentralized Information Filter

Section 4.2 provided an overview of sub-modules that enable the computer vision pipeline for MSEPS. This section discusses the back-end filter extended decentralized information filter (EDIF), a nonlinear extension of the decentralized information filter.

Should there be a centralized node that has access to all the measurements in the network, a standard extended information filter in Algorithm 3 is sufficient to fuse measurements from multiple sensors. In Algorithm 3, j_k and J_k denote the information vector and information matrix at time k and F_k and H_k are Jacobians of dynamics and measurement functions. W_k and V_k are covariance matrices for the process noise and the measurement noise, respectively. The superscript "+" and "−" denotes posterior and prior estimates. Notice that the information filter form of the Kalman filter admits a simple summation form for the measurement equations ((4.9) and (4.10)) facilitating a distributed implementation.

Algorithm 3 in the current form has some disadvantages, however. First, the algorithm is still not distributed. Second, the matrix inverse operation of the information matrix is computationally intensive as the state vector size becomes large. This is a particular concern for MSEPS where the state includes landmark states. We will exploit the Decentralized Information Filter [10] which obtains the approximate solution to the minimum variance estimate for a linear system. EDIF will extend this result to nonlinear dynamics and measurement models. The following sections discuss how to distribute the measurement update. We also develop an improvement to the dynamics update of the information matrix which leverages the special structure of the MSEPS problem.

Algorithm 3: Extended Information Filter.

Result: x_k^+, J_k^+

Prediction step

$$x_k^- = f(x_{k-1}^+, k), \tag{4.6}$$

$$(J_k^-)^{-1} = F_k (J_{k-1}^+)^{-1} F_k^\top + W_k, \tag{4.7}$$

Compute the information vector

$$j_k^- = J_k^- x_k^- \tag{4.8}$$

Get measurement y_k

Measurement update

$$j_k^+ = j_k^- + H_k^\top V_k^{-1} \left(y_k - c(x_k^-, k) + H_k x_k^- \right), \tag{4.9}$$

$$J_k^+ = J_k^- + H_k^\top V_k^{-1} H_k \tag{4.10}$$

Recover the state vector

$$x_k^+ = (J_k^+)^{-1} j_k^+ \tag{4.11}$$

Measurement Update and Consensus

When the overall measurement y_k consists of independent measurements, the measurement update becomes a simple sum of the measurement contributions from all the observations.

$$j_k^+ = j_k^- + \sum_{i \in \mathcal{A}} {H_k^i}^\top {V_k^i}^{-1} (y_i - c_i(x_k^-, k) + H_k^i x), \tag{4.12}$$

$$J_k^+ = J_k^- + \sum_{i \in \mathcal{A}} {H_k^i}^\top {V_k^i}^{-1} H_k^i, \tag{4.13}$$

where y_i and c_i denote the measurement and measurement model of the keypoints as seen by the i-th chaser. The information update has a block-sparse structure that allows further simplifications. Recall that the state vector is divided into sub-blocks $x = [x_T; l^1; \cdots; l^N]$. After sub-dividing the information vector and matrix into corresponding block elements, we can equivalently write (4.12) and (4.13) as following. For simplicity, we drop the subscript k which denoted the time.

$$J_{TT}^{+} = J_{TT}^{-} + \sum_{i \in \mathcal{A}} H_{iT}^{\top} V_k^{i\,-1} H_{iT}, \tag{4.14}$$

$$J_{ii}^{+} = J_{ii}^{-} + H_{ii}^{\top} V_k^{i\,-1} H_{ii}, \quad \forall i \in \mathcal{A}, \tag{4.15}$$

$$J_{iT}^{+} = J_{iT}^{-} + H_{ii}^{\top} V_k^{i\,-1} H_{iT}, \quad \forall i \in \mathcal{A}, \tag{4.16}$$

$$J_{ij}^{+} = J_{ij}^{-}, \quad \forall i, j \in \mathcal{A}, i \neq j. \tag{4.17}$$

The linear form of the measurement update in the information filter leads to the distributed implementation. For a formation with a small number of chasers, (4.12)-(4.13) may be implemented by simply relaying the contributions from individual terms. For a formation with a large number of spacecraft, a consensus algorithm may be used to iteratively converge to average. To see this, we manipulate the measurement update equation to

$$j_k^{+} = \frac{1}{N} \sum_{i \in \mathcal{A}} \left(j_k^{-} + N H_k^{i\,\top} V_k^{i\,-1} (y_i - c_i(x_k^{-}, k) + H_k^{i} x) \right), \tag{4.18}$$

$$J_k^{+} = \frac{1}{N} \sum_{i \in \mathcal{A}} \left(J_k^{-} + N H_k^{i\,\top} V_k^{i\,-1} H_k^{i} \right), \tag{4.19}$$

where $N = \mathrm{card}(\mathcal{A})$. Assuming an undirected graph for the communication topology, the decentralized information filter applies the following consensus protocol to compute the above equations.

$$u_{c+1}^{i} = u_c^{i} + \epsilon \sum_{j \in \mathcal{N}(i)} \left(u_c^{j} - u_c^{i} \right), \tag{4.20}$$

$$U_{c+1}^{i} = U_c^{i} + \epsilon \sum_{j \in \mathcal{N}(i)} \left(U_c^{j} - U_c^{i} \right), \tag{4.21}$$

where ϵ is the constant design parameter called the consensus coefficient. Later, Theorem 2 shows that ϵ must be sufficiently small to guarantee convergence. At the first iteration of consensus, the u_c^i and U_c^i are initialized as

$$u_0^{i} = j_k^{-} + N H_k^{i\,\top} V_k^{i\,-1} (y_i - c_i(x_k^{-}, k) + H_k^{i} x), \tag{4.22}$$

$$U_0^{i} = J_k^{-} + N H_k^{i\,\top} V_k^{i\,-1} H_k^{i}. \tag{4.23}$$

The following theorem guarantees the convergence of the consensus protocol given that ϵ is bounded by a function of the degree of the graph.

Theorem 2 (Convergence of consensus protocol [96])**.** *Consider a network of agents with communication graph $\mathcal{G}$ applying the consensus algorithms* (4.20)-(4.21) *where*

$0 < \epsilon < \frac{1}{\Delta}$ and Δ is the maximum degree of the network. If the digraph is balanced, an average-consensus is asymptotically reached. That is $\lim_{c\to\infty} u_c^i = \frac{1}{N}\sum_{j\in\mathcal{A}} u^j$ for $\forall i \in \mathcal{A}$.

The maximum degree of the network may be controlled by specifying the maximum degree for each node, such that ϵ may be selected prior to the mission. Also, in practice, such a consensus algorithm has been shown to have sufficient convergence with finite iterations [17].

Information Time-Update

One drawback of using the information filter as opposed to the Kalman filter is the time-update of the information matrix requires the additional inversion of the covariance matrix. The computational complexity of inverse scales with $O(n^3)$ in a fully dense matrix where n is the dimension of the state vector. For a large scale problem like MSEPS where n is large, reducing the inversion operation is desirable. With this in mind, we introduce the following proposition in which the structure of the MSEPS problem is used to reduce the complexity of computation.

Proposition 2. *Assume J_{k-1}^+ and W_{Tk} are positive definite matrices. Suppose F_{Tk} and F_k corresponds to Jacobians of f_T and f, respectively, and W_k is the process noise covariance at k. Then, the time-update step of the information matrix can be re-written as*

$$J_k^- = \bar{J} - \bar{J}\omega(I + \omega^\top \bar{J}\omega)^{-1}\omega^\top \bar{J} \tag{4.24}$$

where

$$\bar{J} := F_k^{-\top} J_{k-1}^+ F_k^{-1}, \tag{4.25}$$

$$\omega := [(W_{Tk})^{\frac{1}{2}}; 0; \cdots; 0]. \tag{4.26}$$

Proof. The most general time-update equation of the information matrix is given by

$$(J_k^-)^{-1} = F_k (J_{k-1}^+)^{-1} F_k^\top + W_k \tag{4.27}$$

where F is the Jacobian of the overall dynamics $f(x)$. We leverage that the process noise is introduced only to the target state x_T and not on the landmarks. Suppose W_T is the process noise covariance corresponding to the target propagation. Then, the process noise covariance for overall dynamics may be written as $W = \omega\omega^\top$ where ω is defined above with zero matrices, appropriately sized. Let $\bar{J} = F^{-\top} J_{k-1}^+ F^{-1}$, where dynamics of Jacobian has block diagonal structure $F = \text{diag}(F_T, I, \cdots, I)$. Using the definitions of ω and $\bar{J}$, (4.27) may be equivalently written as

$$\begin{aligned} J_k^- &= (\bar{J}^{-1} + \omega\omega^\top)^{-1} \\ &= \bar{J} - \bar{J}\omega(I + \omega^\top \bar{J}\omega)^{-1}\omega^\top \bar{J}. \end{aligned} \tag{4.28}$$

The second equality holds by using the matrix inversion lemma. □

We note that the computational complexity of (4.24) with respect to landmark size scales better than (4.27). Jacobian F has a block diagonal structure with mostly identity elements, so the inverse of Jacobian is simply $F^{-1} = \text{diag}(F_T^{-1}; I; \cdots; I)$. The computation of ω does not depend n, and computation $\bar{J}$ and $\bar{J}\omega$ both scales linearly, i.e., $O(n)$. The matrix inversion $(I + \omega^\top \bar{J}\omega)^{-1}$ is order of $n_T \times n_T$ matrix and does not depend on n. The operation that requires most computation is $\omega(I + \omega^\top \bar{J}\omega)^{-1}\omega^\top \bar{J}$ which requires $O(n^2)$ computation. Therefore this manipulation eliminates the inversion of the full information matrix.

State Recovery

Once the information vector and the information matrix are computed, we need to compute the state vector, which involves the inversion of the information matrix

$$x = (J^+)^{-1} j^+. \tag{4.29}$$

Tracked Landmarks

The newly discovered landmark states and respective covariances are initialized after the measurement update step. This takes place after the communication step, ensuring that new landmarks detected by any of the chasers are included in the new state vector and the information matrix. The information matrix is updated by placing the inverse of the initial covariance of the new landmarks on the extended diagonal block.

This completes the discussion of all the necessary components of the EDIF algorithm. EDIF algorithm is summarized in Algorithm 4 for clarity. In the context of the overall architecture shown in Fig. 4.4, the modules that are primarily involved with the EDIF are denoted with asterisks.

4.4 Simulation

The part of the MSEPS architecture described in Section 4.2-4.3 is validated in a computer simulation. The purpose of the simulation is to validate the multi-spacecraft architecture, the distributed algorithm, and the dynamic allocation of

```
Algorithm 4: Extended Decentralized Information Filter.
Result: $x_k^+, J_k^+$
Prediction step
    $$x_{T_k}^- = f_T(x_{T_{k-1}}^+), \quad l_{i,k}^- = l_{i,k-1}^+ \quad (4.30)$$
    $$\bar{J} = F^{-\top} J_{k-1}^+ F^{-1} \quad (4.31)$$
    $$J_k^- = \bar{J} - \bar{J}\omega(I + \omega^\top \bar{J}\omega)^{-1}\omega^\top \bar{J} \quad (4.32)$$
Get measurement $y_i$
Compute consensus proposals
    $$u_0^i = j^{i^-} + N H_k^{i\top} V_i^{-1}(y_i - c_i(x_k^-, k) + H_k^i x_k^-) \quad (4.33)$$
    $$U_0^i = J^{i^-} + N H_k^{i\top} V_i^{-1} H_k^i \quad (4.34)$$
while $c \leq C$ do
    Perform consensus
        $$u_c^i = u_{c-1}^i + \epsilon \sum_{j \in \mathcal{N}_i} \left(u_{c-1}^j - u_{c-1}^i\right) \quad (4.35)$$
        $$U_c^i = U_{c-1}^i + \epsilon \sum_{j \in \mathcal{N}_i} \left(U_{c-1}^j - U_{c-1}^i\right) \quad (4.36)$$
end
Compute the posterior state and information matrix
    $$x_k^+ = \left(U_C^i\right)^{-1} u_C^i \quad (4.37)$$
    $$J^+ = U_C^i \quad (4.38)$$
Add new landmark states and covariances
```

landmark states and covariances. As such, we do not use the images to extract the keypoints and synthetic images are only used for visualization purposes. We assume that the keypoints extraction, keypoint correspondence, and optical flow are solved by functional sub-modules. We refer the reader to [90], [95] for further implementation details. Future work includes validation of the algorithm with the computer vision algorithm in the loop using the realistic synthetic images.

Extracted keypoints in this simulation are obtained by projecting a set of pre-defined 3D landmarks attached to the exterior of the spacecraft model to each chaser's image plane. A set of keypoints was simulated for each chaser based on the relative pose between the chaser and the target at the time. Only the landmarks that have line-

Orbital Parameter	Value
Semi-major axis	42167.0 km
Eccentricity	2.9E-4
Inclination	8.1E-2 deg
Argument of perigee	354.2 deg
RAAN	68.8 deg
True anomaly	240.8 deg

Table 4.1: Geostationary orbital parameters.

Relative State	Chaser 1	Chaser 2	Chaser 3
Position [m]	20.0	-10.0	-10.0
	0.0	-34.6	34.6
	34.6	-17.3	-17.3
Velocity [mm/s]	0.0	-1.26	1.26
	-2.92	-1.46	1.46
	0.0	-2.19	2.19

Table 4.2: Chaser initial states in target-LVLH frame.

of-sight are considered visible. The keypoints corresponding to the same landmark have the same descriptors at different epoch, but the descriptors are different across different spacecraft. At each epoch, each chaser processes the observation and runs an iteration of the filter. The swarm of spacecraft only shared the variables over the simulated inter-spacecraft communication. The algorithms on all the chaser spacecraft are assumed to run in a synchronized fashion via clock synchronization obtained from the GNSS. The differential GNSS and the star tracker measurements used in the formation flight estimation are simulated by the ground truth relative state.

Orbital Mechanics and Attitude Dynamics

The target and chaser spacecraft are placed in geostationary orbits and the target's initial orbital parameters are tabulated in Table 4.1. The initial positions and velocities of the chaser spacecraft were selected such that they are in a formation with respect to the target and each other, as shown in Table 4.2. The resulting trajectories of the chasers form concentric circular orbits with respect to the target as shown in Fig. 4.2d. The absolute orbit for each spacecraft was computed by propagating the respective spacecraft in the Earth-Centered Inertial (ECI) frame. The target spacecraft simulated for this validation has a slight tumble which makes the CG position in spin-parallel direction still observable over a longer period of time.

The target's initial angular velocity is 1 degree per second. While the target attitude is numerically propagated by torque-free dynamics, each chaser is assumed to track a smooth attitude trajectory that satisfies its pointing requirement. Specifically, each chaser points its camera optical axis towards the target spacecraft such that the target is always entirely within the field of view. We assume that reaction wheels will provide the necessary slew maneuvers. Note that because chaser spacecraft inertial poses are assumed to be known from the GNSS and the star tracker, the MSEPS does not need to model the chasers' dynamics on-board.

Simulated Measurements

Each chaser receives the GPS and star tracker measurements as the pose with respect to the ECI frame. These measurements are simulated as the ground truth positions. Note that this is not a strong assumption, as in GEO and above, although above the GPS constellation, it is possible to track GPS signals for navigation using high sensitivity receivers [97]. Millimeter accuracy has been proved in LEO filtering differential GPS observations with a model of the spacecraft dynamics. The same accuracy is theoretically also possible in higher orbits, such as in GEO, by tightly fusing the GPS observations with measurements of another sensor [92].

We also assume that the keypoint detection and correspondence are solved by the front-end computer vision algorithm, instead of extracting the keypoints from synthetic images using the computer vision algorithms.

A set of keypoints are simulated by projecting the landmark to the camera origin given the relative pose of the target with respect to the camera at each time epoch. A set of pre-defined 3D landmarks are placed on the exterior of the spacecraft model. A pinhole camera model is used and the landmarks are only visible when there is no obstruction on their line of sight. We artificially add Gaussian noise to each keypoint observation at each epoch. The image size is 1024-by-1280 pixels and the focal length is $f = 2400$ pixels.

Results

The reconstructed shape of the target is shown as a 3D point cloud in Fig. 4.5. Even though the target spacecraft (Cygnus) has a complex geometry including deployed solar panels, the reconstructed point cloud closely resembles the target shape. The red, blue, and green markers denote the landmarks that are visible by Chaser 1, 2, and 3, respectively at the epoch. Because three spacecraft are spatially distributed, they cover different surfaces of the Cygnus, showing the advantage of the multi-

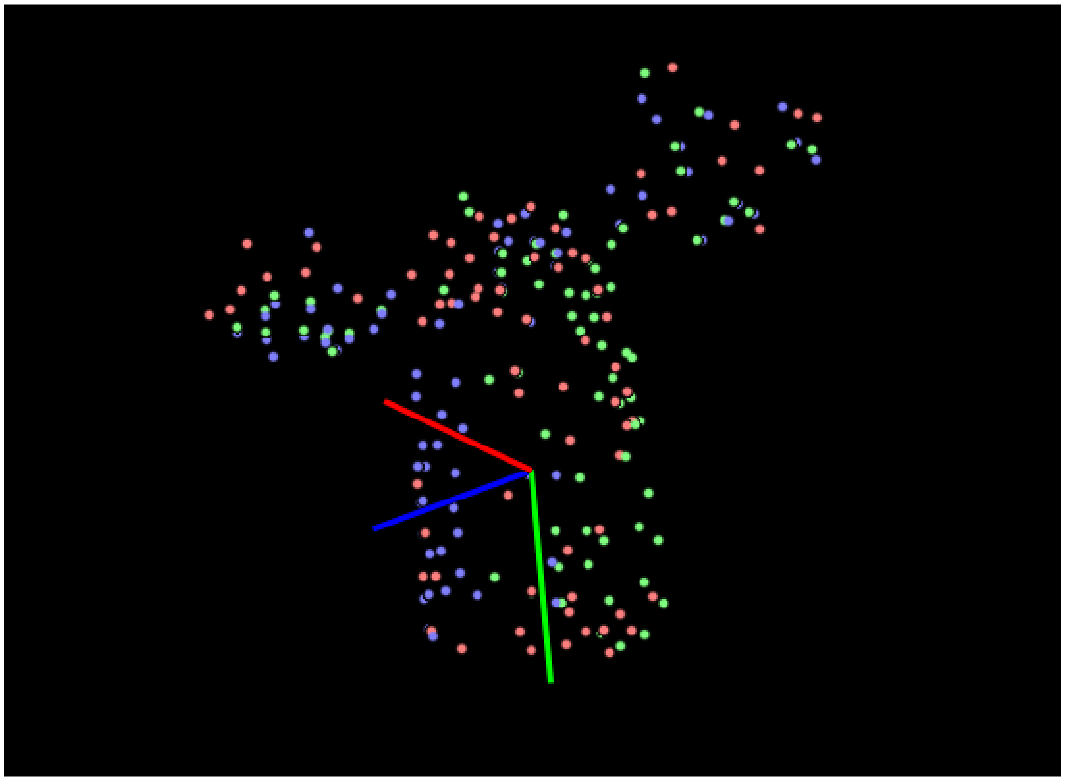


Figure 4.5: Reconstructed shape obtained from the tracked landmarks. Red, blue, and green markers denote landmarks seen by Chasers 1, 2, and 3, respectively.

spacecraft approach. Fig. 4.1 shows the same reconstructed shape from another perspective. The figure also includes the camera pose trajectory expressed in the estimated target reference frame. Even though the relative trajectories of the chasers were designed to be concentric circular orbit in the LVLH frame (Fig. 4.2d), the chaser trajectories are more complex when expressed in the target reference frame due to the target's own rotation.

The projected keypoints of Chaser 1 is shown along with the synthetic image in Fig. 4.6. For each keypoint, the corresponding landmark covariance projected onto the image plane is shown as a circle around the keypoint. Green color indicates that the keypoint was tracked for 10 or more consecutive frames, while orange color indicates the point was tracked for less. The figure shows that the landmarks with longer tracks have smaller covariances as expected.

The results of target pose tracking are shown in Fig. 4.7 and 4.8. Fig. 4.7 shows the quaternion values of both truth and estimated target attitude, with respect to the initial frame. The estimated attitude quaternion follows closely of the true attitude. It also shows that the estimate obtained by all chasers agree with each other. Fig. 4.8

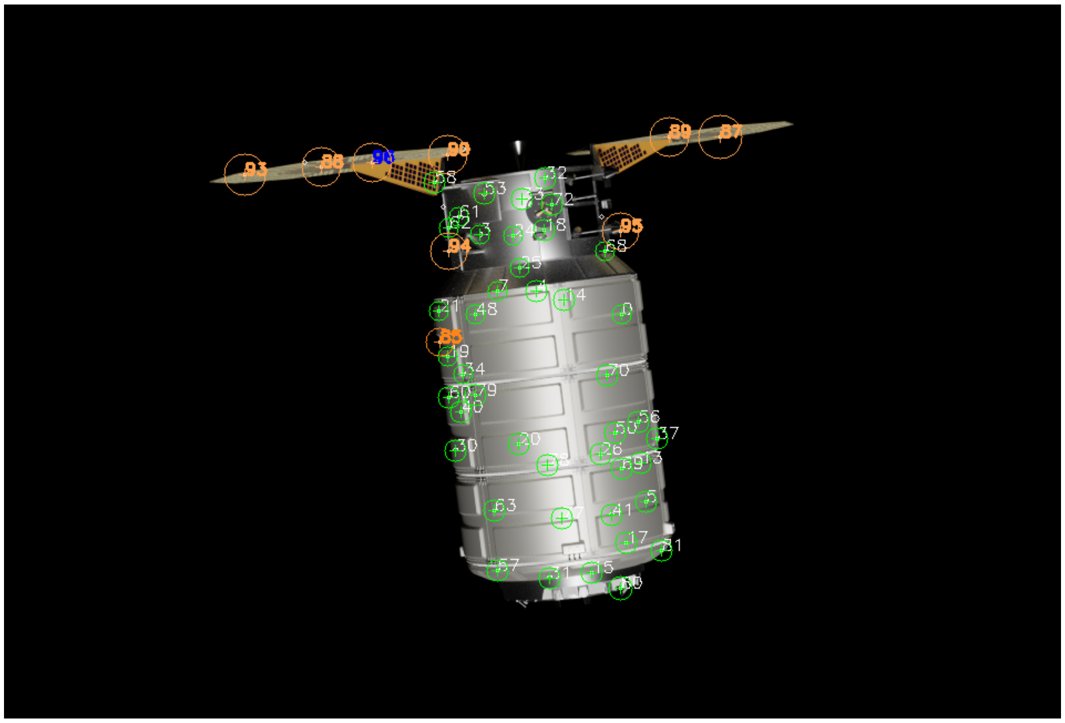


Figure 4.6: Tracked landmark overlaid on the synthetic image. Circles around each landmark indicate the size of uncertainties. Green landmarks indicate states that are tracked for more than 10 consecutive frames.

shows the estimation error for target's CG. The error is defined with respect to the ground truth target CG which was used to generate the simulation. The CG location estimation errors are described in terms of the parallel and perpendicular directions to the target spin-axis. This projection is selected because the observability of the CG location in the perpendicular direction is expected to be higher than the direction parallel for any rotating object with constant or slowly varying angular velocity vector. Fig. 4.8 verifies these behaviors where it shows a quick initial convergence in both parallel and perpendicular directions. Then the position estimate continues to converge (approach towards zero) with higher error in the parallel direction. The CG position estimate remains bounded over time.

4.5 Chapter Summary

We presented the cooperative vision-based pose estimation algorithm called Multi-spacecraft Simultaneous Estimation of Pose and Shape (MSEPS). MSEPS is posed as a distributed sensor network paradigm. Using inter-spacecraft communication, MSEPS tightly integrates the vision-based feature tracking problem with a distributed estimation framework. We provided an overview of the multi-spacecraft

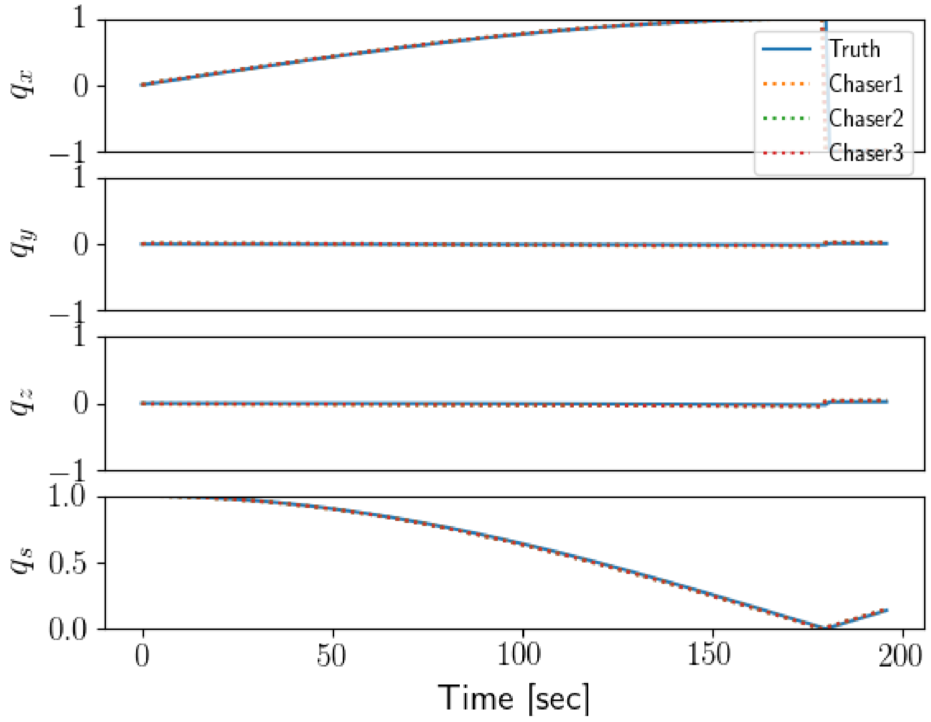


Figure 4.7: Target attitude quaternion tracking result.

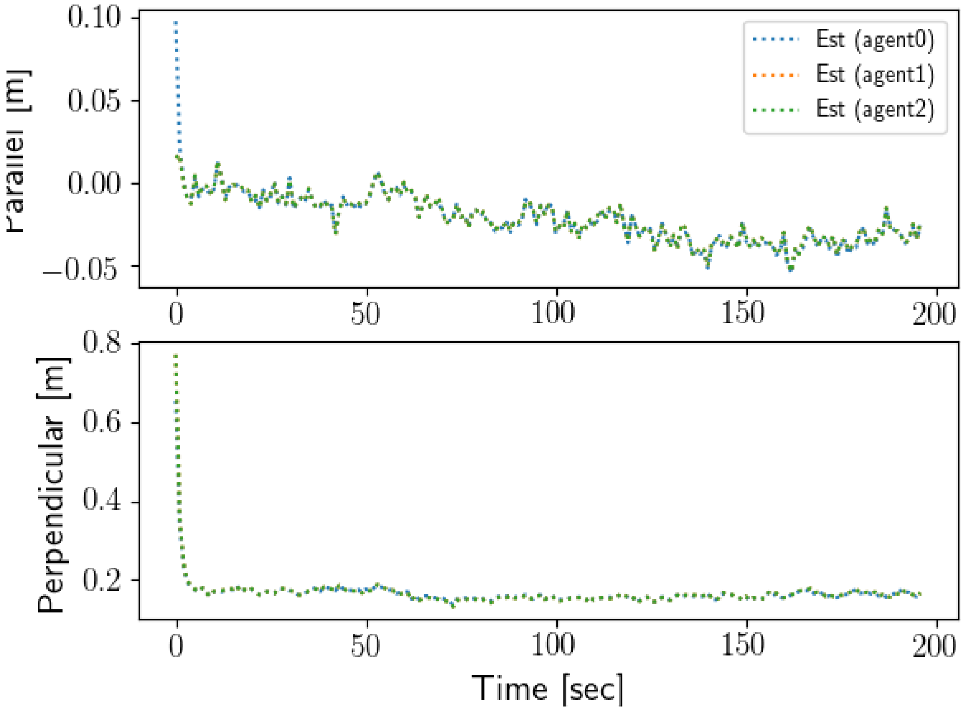


Figure 4.8: Target center of gravity estimation error.

algorithm architecture that consists of the computer vision pipeline, communication, and back-end filtering. We proposed the extended decentralized information filter that approximately solves the minimum variance estimate of the global information in a distributed fashion. We made algorithm improvement that exploits the special structure of the MSEPS problem. We validated the distributed algorithm and some of the algorithm pipelines using the simulation.

Chapter 5

HIGH-FIDELITY SIMULATION TOOLS FOR SPACECRAFT VISION-BASED RELATIVE NAVIGATION

This chapter contains material from the following publications:

[1] K. Matsuka, L. Zhang, I. Ragheb, C. Ohenzuwa, and S.-J. Chung, "High-fidelity, closed-loop simulation of spacecraft vision-based relative navigation in ros2," in *2023 33rd AAS/AIAA Space Flight Mechanics Meeting*, AIAA, 2023,

[2] K. Matsuka, C. Ohenzuwa, and S.-J. Chung, "Rapid synthetic image generation using neural radiance fields for vision-based formation flying spacecraft," in *2023 33rd AAS/AIAA Space Flight Mechanics Meeting*, AIAA, 2023,

The previous two chapters presented the DPE and MSEPS algorithms which are designed to enable spacecraft to perform robotics operations in orbit. As we developed new algorithms that bridge the gap between robotics and space systems, we noticed the lack of publicly available simulation tools that integrated robotics technologies with space systems, such as those used to test vision-based navigation algorithms. When developing robotics applications, it is helpful to utilize Robot Operating System 2 (ROS2) which is a standardized, open-source, software development kit with a rich algorithm library. However, ROS2 currently does not have much support for high-fidelity simulations of space environments. On the other hand, there exist high-fidelity astrodynamics simulation tools for guidance, navigation, and control flight software for spacecraft; however, these tools do not have support to interact with robotics software such as those implemented ROS2. To address this gap between robotics and space systems, we developed new simulation tools that help with the development of vision-based navigation algorithms and proximity operations. These tools allow us to test autonomous algorithms implemented in ROS2 in real-time and closed-loop fashion without requiring access to space or expensive experimental setups.

This chapter presents two new simulation tools. First, we present ROS-Basilisk, a ROS2-compatible software interface for Basilisk [98]. Basilisk is an open-source software that is capable of performing real-time astrodynamics simulation and is useful for guidance, navigation, and control in aerospace applications. While Basilisk provides a rich framework for astrodynamics simulation, it does not currently interface with ROS, which would be required to validate the real-time performance of algorithms. We designed our ROS-Basilisk interface in a manner that preserved the flexibility and computational efficiency of Basilisk while also allowing algorithms implemented in ROS2 to interact with the dynamics simulation in Basilisk.

Second, we also developed ROS-NeRF, which takes the simulated spacecraft poses of the observer and target spacecraft and simulates the images of the target that the observer's relative navigation camera would acquire. ROS-NeRF is implemented based on Neural Radiance Fields or NeRF [99], a recent technique for learning-based 3D view synthesis. ROS-NeRF rapidly renders high-fidelity images of target spacecraft from novel viewpoints based on the relative position and orientation of the target and observer spacecraft. ROS-NeRF also expands NeRF to be applicable to variable lighting directions, where the target direction of the sun relative to the target may change due to the target's rotational motion. We do this by explicitly incorporating the lighting direction as an input parameter in neural network architecture. During an offline phase, we first train NeRF on high-fidelity synthetic images of spacecraft. During the online phase, the trained model is used to generate synthetic images from a new camera, and this online part is implemented as a ROS2 node. With this approach, ROS-NeRF can rapidly render high-quality images of target spacecraft and publish them to other ROS2 nodes, such as vision-based navigation algorithms.

To demonstrate the use cases of these simulation tools, we also develop an integrated set of autonomy algorithms for on-orbit inspection and formation flying scenarios. We demonstrate that ROS-Basilisk and ROS-NeRF can be used to validate the performance of autonomy algorithms that are implemented in ROS2.

The rest of the chapter is organized as follows. Section 5.1 discusses the motivation for developing the new simulation tools. Section 5.2 discusses the details of ROS-Basilisk. Section 5.3 describes the details of ROS-NeRF. Section 5.4 briefly describes the set of autonomous algorithms we developed for on-orbit inspection scenarios. Section 5.5 discusses the numerical simulation results. Finally, Section 5.6 makes concluding remarks about this chapter.

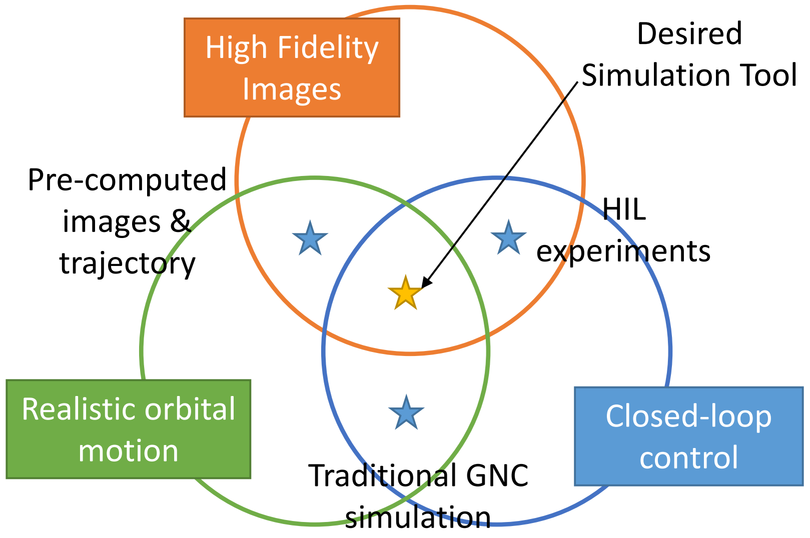


Figure 5.1: Characterization of common verification and validation strategies for on-orbit inspection systems vs. ideal simulation tool that was developed in Chapter 5.

5.1 Motivation

In order to validate vision-based navigation algorithms for on-orbit inspection, people commonly perform either numerical simulations, hardware experiments on the ground, or some combination of both. However, common verification techniques have some disadvantages as illustrated in the Venn diagram in Fig. 5.1. For example, even though traditional GNC simulation tools (e.g., Basilisk [98]) are capable of performing closed-loop simulations with realistic dynamics, they have previously lacked the ability to simulate information-rich sensors (e.g., camera) rapidly and at high fidelity. Hardware experiments involving real cameras and platforms, such as robotic arms [100] or air-bearing robots [78], can perform closed-loop simulations with real images. However, these experimental platforms are often limited in their range of motion, costs, and feasible simulation duration. Pre-computing a trajectory and rendering a set of synthetic images can be made as realistic as the accuracy of models, but the simulations are not closed-loop as it does not reflect the controls computed based on the vision-based navigation estimates. In order to reduce engineering time and cost, an ideal tool should model realistic orbital motions and synthetic images and be able to perform closed-loop simulations.

5.2 ROS-Basilisk: Astrodynamics Simulation

This section discusses ROS-Basilisk, a software layer that bridges ROS2 and Basilisk [98]. Basilisk features a large library of high-fidelity force models, hardware

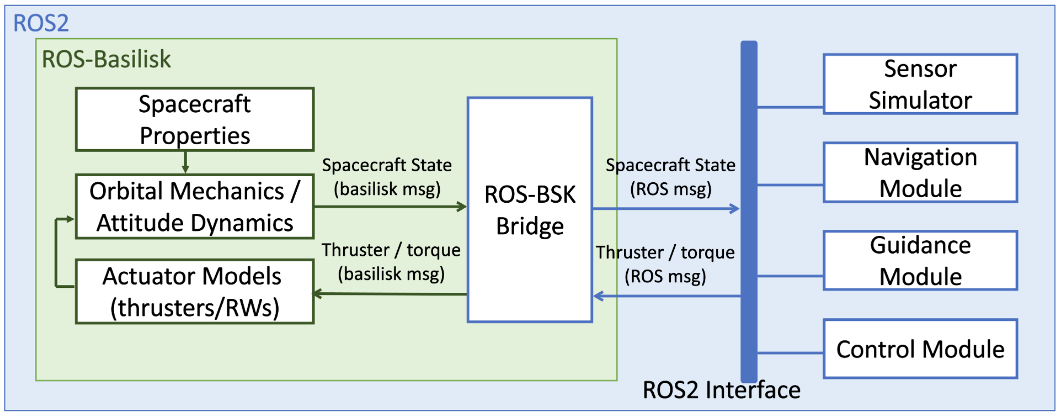

ROS2
ROS-Basilisk
Spacecraft Properties
Orbital Mechanics / Attitude Dynamics
Actuator Models (thrusters/RWs)
Spacecraft State (basilisk msg)
Thruster / torque (basilisk msg)
ROS-BSK Bridge
Spacecraft State (ROS msg)
Thruster / torque (ROS msg)
Sensor Simulator
Navigation Module
Guidance Module
Control Module
ROS2 Interface

Figure 5.2: Diagram of ROS-Basilisk interacting with other modules via ROS2 messaging.

components, visualization tools, an efficient back-end implementation in C/C++, and a user-friendly Python interface for astrodynamics simulation. Basilisk can be used to perform closed-loop simulations of space mission scenarios with G&C algorithms, such as formation flying [101]. ROS-Basilisk acts as a lightweight interface software such that ROS2 and Basilisk can run side-by-side, and modules implemented in ROS2 can interact with dynamics simulations in Basilisk.

The architecture of ROS-Basilisk and its interaction with ROS2 and Basilisk is shown in Fig. 5.2. From the ROS2 perspective, ROS-Basilisk is a ROS2 node that simulates spacecraft dynamics and actuators. ROS-Basilisk "subscribes" to control messages (eg. thruster burn time, reaction wheel torques), simulates the actuation and spacecraft dynamics, and "publishes" the resulting states (eg. spacecraft poses). Other ROS2 modules can use the ground truth states from ROS-Basilisk to simulate sensors (e.g., camera, GPS, star tracker), which are in turn used by other navigation and control algorithms implemented in ROS2. From Basilisk's perspective, there is a custom Basilisk module (ROS-BSK Bridge in Fig. 5.2) that interacts with the spacecraft dynamics and actuator models. ROS-BSK Bridge translates from Basilisk messages to ROS2 messages and vice versa. From the rest of the Basilisk modules' point of view, ROS-BSK Bridge is a Basilisk node that handles sensor simulations and autonomy algorithms. With this lightweight interface implementation, simulations within Basilisk and autonomy algorithms within ROS2 can interact with each other.

ROS-Basilisk also runs ROS2 and Basilisk simultaneously in parallel. More specifically, ROS-Basilisk is implemented as a ROS2 node with a timer-based callback, and it propagates the Basilisk simulation for one simulation step at a time at each callback to emulate real-time simulation. ROS-Basilisk uses timestamps on ROS2 messages to handle synchronization. Running one simulation at a time using timer-callback coarsely aligns Basilisk and ROS simulations; however, the exact timestamps may not be aligned precisely due to possible delays such as the time it takes to compute the simulation. We address this synchronization issue by appending each ROS2 message with a timestamp from the Basilisk simulation time.

In summary, ROS-Basilisk enables high-fidelity astrodynamics simulation capabilities for robotics applications with a bi-directional ability to interact with other ROS2 modules. ROS-Basilisk preserves the flexibility and extensive capabilities of Basilisk and can accommodate various mission scenarios beyond the example considered in this chapter. Since the interface only deals with translating between ROS and Basilisk messages, one can incorporate various actuator models and high-fidelity environmental forces in the Basilisk simulation scenario. The next section presents an example of using the simulation output from ROS-Basilisk where the relative navigation camera simulation module computes the image of the target based on the spacecraft poses.

5.3 ROS-NeRF: Fast Camera Rendering via Neural Radiance Fields

In addition to ROS-Basilisk, we also developed ROS-NeRF, a separate ROS2 module based on Neural Radiance Fields (NeRF) [102] that rapidly renders synthetic images of target spacecraft in real-time along with ROS2. NeRF is a learning-based, novel-view synthesis technique. In NeRF, a neural network is first trained from a relatively small number of images. This trained model can then be used to render images from novel views. Since NeRF was first published, various extensions to NeRF have been explored. For example, some extensions aim to improve the training and rendering speed [103]–[105], scale up to large scenes [106], [107], accommodate lighting variation [108], enable multi-resolution rendering [103], [109], and estimate the camera pose jointly [110]. In ROS-NeRF, we applied NeRF to spacecraft target simulation applications by training it on a custom dataset of high-fidelity, synthetic images of spacecraft. We also developed a ROS2 software interface so other vision-based navigation algorithms implemented in ROS2 can utilize the images rendered from NeRF in real time. One challenge with applying NeRF to space applications is that the spacecraft target can move relative to the light sources in the scene; to address this, we also present a strategy to specify lighting variations that exploits the conditions expected in the orbital environment.

To learn a 3D scene from a set of 2D images, NeRF uses Multilayer Perceptron (MLP) to model the mapping from a 5D coordinate (i.e., a spatial position $x \in \mathcal{R}^3$ and viewing direction $d_v \in \mathcal{S}^2$) to a volume density $\sigma(x) \in \mathcal{R}$ and view-dependent emitted radiance $c(x, d_v) \in \mathcal{R}^3$ at each location. Given the neural network weights Θ, we can write the mapping that is learned by the neural network as

$$F_\Theta : (x, d_v) \rightarrow (\sigma, c), \tag{5.1}$$

where the mapping F_Θ is modeled by a Multi-Layer Perceptron (MLP). Once NeRF memorizes the mapping to $\sigma(x)$ and $c(x, d_v)$, the image from a new pose can be reconstructed.

The rendered image is computed by evaluating the volume density and emitted radiance along the camera ray. Suppose a ray $r : \mathbb{R} \rightarrow \mathbb{R}^3$, $r(t) = o + pt$ has the origin $o \in \mathbb{R}^3$ and ray direction $p \in \mathbb{R}^3$. The values of $\sigma(r(t))$ and $c(r(t), d_v)$ are sampled between the near and far bounds $t \in [t_n, t_f]$ along the ray. Then, the expected color along the ray is computed by the following integral:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d_v)dt, \tag{5.2}$$

where

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right). \tag{5.3}$$

The color value $C(r)$ is computed for each pixel to generate an image from a novel view.

The mapping F_Θ in (5.1) as formulated in the original NeRF paper [102] is suitable for modeling a static scene with time-invariant lighting conditions. In relative navigation applications for spacecraft, however, the lighting conditions with respect to the target body frame may change rapidly due to the rotation of the target. There are some prior works in the literature that addressed the time-varying lighting conditions by additionally including hidden variables which can be trained to capture non-static effects [72]. This is a relatively generalizable approach where the variation of lighting source is unknown. In contrast, there are additional exploitable structures when considering the lighting variation for spacecraft relative navigation. These exploitable structures include (1) the primary source of illumination in orbit is the sun; (2) the sun's position is well-known; and (3) the ray from the sun is mostly collimated and constant. When these assumptions hold, we can modify NeRF such that we train the MLP for *explicit* mapping from the lighting direction $d_l \in \mathcal{S}^2$ to view-dependent, *lighting-dependent* emitted radiance $c(x, d_v, d_l)$. That is to define the modified mapping $\tilde{F}_\Theta$ as

$$\tilde{F}_\Theta : (x, d_v, d_l) \to (\sigma, c). \tag{5.4}$$

Note that the density $\sigma(x)$ is still invariant to lighting direction d_l. Explicitly incorporating known structures to the neural network makes the training more efficient and results more accurate. Therefore, ROS-NeRF trains for the mapping in (5.4).

The workflow of ROS-NeRF is divided into two phases: an offline training phase and an online rendering phase. In the offline phase, we first generate a set of high-fidelity images of a target spacecraft with varying poses and lighting conditions using Blender. We then train the neural network using a sparse set of generated images with ground truth poses and lighting conditions. In the online phase, ROS-NeRF uses the pre-trained, neural-network weights to render synthetic images of the target spacecraft. Given a ground truth spacecraft pose for the target and servicing spacecraft in an inertial frame at a particular time step, NeRF renders an image of the target spacecraft. This online rendering part of the algorithm is implemented as a ROS2 node. In this work, ROS-NeRF is implemented with NVIDIA's Neural Graphics Primitives (instant-NGP) [103]. We chose instant-NGP due to its speed and usability; however, the basic idea behind using NeRF for online image rendering tools can be implemented with other NeRF variants in the literature. Next, we discuss the implementation details of ROS-NeRF in the following sections.

Offline Phase – Data Generation and Training

The offline phase of ROS-NeRF includes data generation and neural network training. A sparse set of realistic synthetic images of the target spacecraft are generated using the conventional ray-tracing rendering software Blender [111]. For each image, a camera pose T and a lighting direction d_l are randomly sampled. The camera pose is expressed in the target body frame and is sampled randomly from the uniform distribution on a sphere with a radius R with the camera pointing at the target object. The lighting direction d_l is sampled uniformly from a unit sphere. The set of training images has 4 channels (RGBA), where the alpha-channel specifies the transparency. The background of the synthetic images is set to be transparent, as this helps improve the training of NeRF. Generally, this training dataset generation step is slow and computationally intensive, and that is acceptable. The idea is to generate a set of highly realistic images based on physics-based rendering only once in an offline phase.

Once the dataset generation is completed, we then train the NeRF neural network with a sparse set of synthetic images. This set of images sparsely samples the 3D scene. Each image has a known camera pose and lighting direction as trainable parameters. After N iterations, we store the trained weights of the neural network. These weights represent the 3D scene of the target spacecraft.

Online Phase – Rendering

In the online phase, ROS-NeRF uses the pre-trained weights to render synthetic images of the target spacecraft. Given a ground-truth pose for the target and servicing spacecraft and a lighting direction, ROS-NeRF renders an image of the target spacecraft as seen by a relative navigation camera. For each pair of target and camera poses, we render a 4-channel, RGBA image using the previously trained NeRF model. Since the background of the rendered image is transparent, a different background can be incorporated to generate a 3-channel RGB image. The output of ROS-NeRF is a synthesized image of the target spacecraft from an arbitrary viewpoint at a particular time. The online portion of the algorithm is packaged as a ROS2 node so that the rendered images can be published as ROS2 messages.

ROS-NeRF can also be used in scenarios where there are multiple objects to be simulated. When there are multiple objects in the view, our approach can be extended by simply using multiple NeRF models to render individual targets in parallel. Using information from the alpha channel, which specifies transparency, these images can then be combined into a single composite image containing possibly multiple objects. This approach may be a reasonable approximation when the interactions of light between objects are negligible compared to the light from the sun. The online portion of the ROS-NeRF pipeline is illustrated in Fig. 5.3.

In the next section, we describe experiments to evaluate the performance of the images rendered via NeRF on some basic computer vision tasks such as object detection and keypoint matching.

Evaluation of NeRF-Rendered Images

This section provides the evaluation results for ROS-NeRF. For each of the following experiments, images were rendered using both Blender and ROS-NeRF. We first present qualitative comparisons of the images. Next, we evaluate the rendering

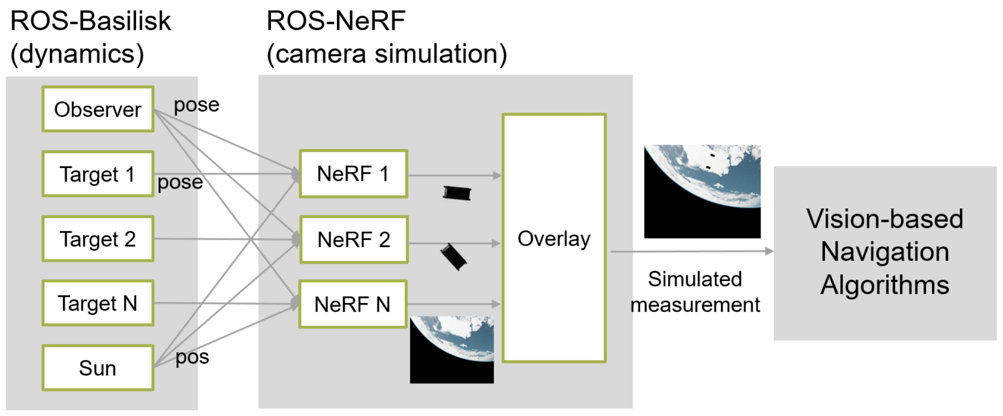


Figure 5.3: The architecture of ROS-NeRF.

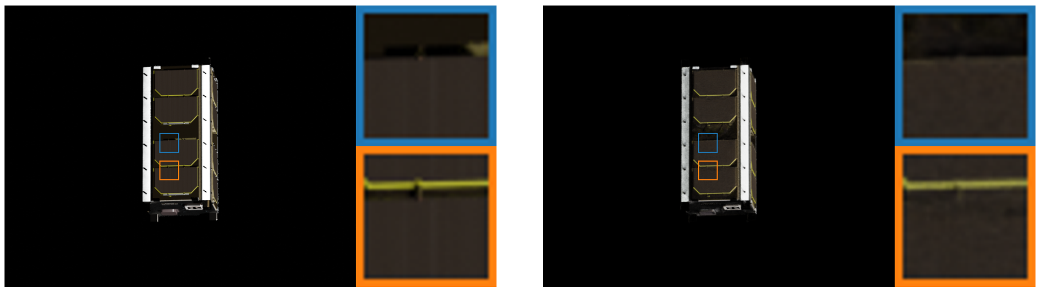
fidelity by comparing the performance of object detection and keypoint matching algorithms applied to the rendered images.

The experiments were carried out on a desktop with a CPU of Intel i7-8700 @ 3.20GHz and a GPU of GTX 1070 (8GB memory). The rendering speed depends on the image size and relative size of the object in the image. For a 640 by 512 pixel image, the average rendering time per image was 10.1 sec for Blender and 2.2 sec for instant-NGP. We also note that the rendering speed of Blender could be much slower depending on the complexity of the 3D rendering model.

Qualitative Comparison Between NeRF And Blender

First, we qualitatively compare the synthetic images generated from Blender and NeRF. Figures 5.4a and 5.4b provide a comparison of images rendered by Blender and NeRF with the same camera pose and lighting direction. The image is generated under a previously unseen view and lighting; i.e., the specific image from Blender was not included as a part of the dataset for training NeRF. Despite some subtle differences in appearance, overall, NeRF renders an image that very is similar to the physics-based rendering given by Blender.

Next, Fig. 5.5 shows images rendered by Blender and NeRF while varying lighting conditions. The figure shows the network successfully renders high-quality images of the spacecraft from the same viewpoint but with varying lighting conditions. With our approach to explicitly incorporate lighting direction in the neural networks, the model learns the variation in appearance well.



(a) Blender (b) NeRF

Figure 5.4: Comparison of target spacecraft images rendered via Blender and NeRF.

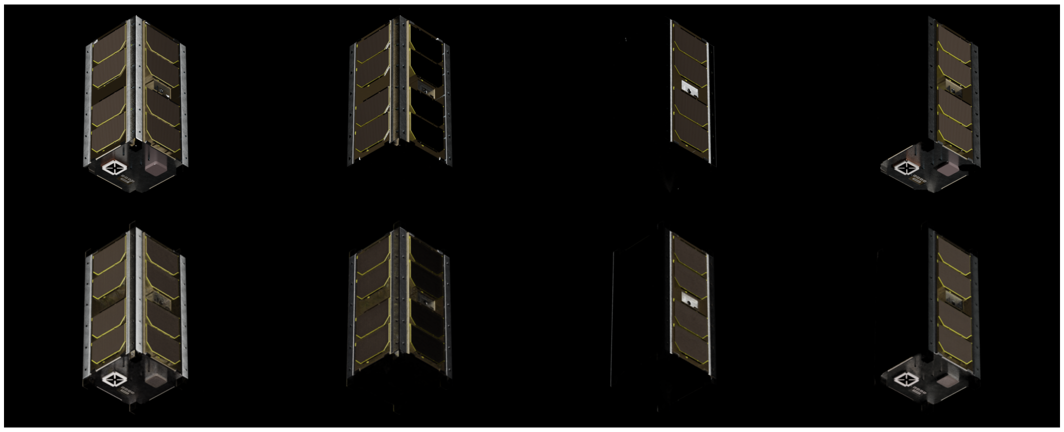


Figure 5.5: Images of a target spacecraft from the same camera pose but 4 different lighting conditions. Images by Blender are on the top row and images from NeRF are on the bottom row.

Object Detection on NeRF-rendered Images

For this evaluation, we applied an object detection algorithm for spacecraft on images rendered via NeRF and Blender and compared their results. The object detection algorithm uses the network architecture of YOLOv5 [112] and is specifically trained for detecting spacecraft. The detection algorithm is described in more detail in the "Vision-Based Spacecraft Detection" section. An example of an object detection result is shown in Fig. 5.6. To compose the image, the target CubeSat was rendered by NeRF, and then overlayed with a background image of the earth. The object detection algorithm correctly identified the object in a scene as a spacecraft with 95% probability.

We compared the performance of the object detection algorithm on two sets of images; one generated by NeRF and the other generated by Blender. For each method, we generate a target spacecraft image with randomized camera pose and lighting. The background image may randomly have Earth either fully or partially

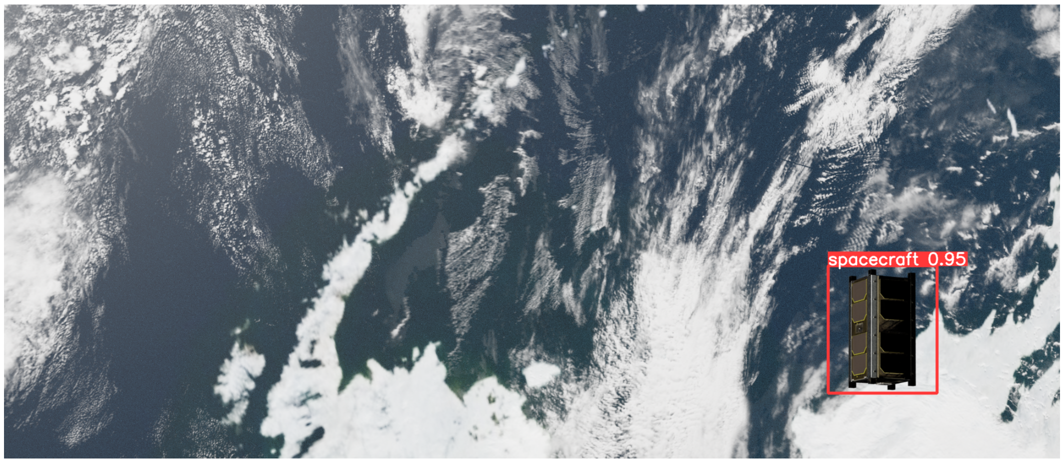


Figure 5.6: An example of the spacecraft detection algorithm. The image of CubeSat, minus the background image, was rendered by NeRF.

in the view, or not in the view at all. The placement of the satellite in the image, in terms of position, orientation, and scale, was also randomized. The scale of the spacecraft varied anywhere from 2% to 80% of the overall image, making the detection task challenging for some of the images. The two sets of images—one by Blender and one by NeRF—were generated using the same seed for the pseudo-random number generator. In other words, both sets of images have the same randomization; therefore the only differences between the two sets of images are the fact that spacecraft images were rendered by Blender or NeRF. Each set contains 500 images.

Based on the object detection results, we compute the precision and recall rates of the object detection algorithm. The resulting precision-recall curves are shown in Fig. 5.7. While there are small differences between the two curves, the object detection algorithm was able to detect the target object similarly for both sets of data. This suggests that NeRF-generated images are a suitable alternative for validating object detection tasks.

Keypoint Detection and Matching of NeRF-derived Images

In the next experiment, we applied visual keypoint detection and matching on a sequence of NeRF-rendered images. Visual keypoints serve as a basic building block for the computer vision pipeline when applying Simultaneous Localization And Mapping (SLAM)-like approaches to the spacecraft pose estimation problem [113]–[116]. The objective of this experiment was to evaluate the performance of keypoint matching on the spacecraft images rendered by NeRF. To generate the set of images,

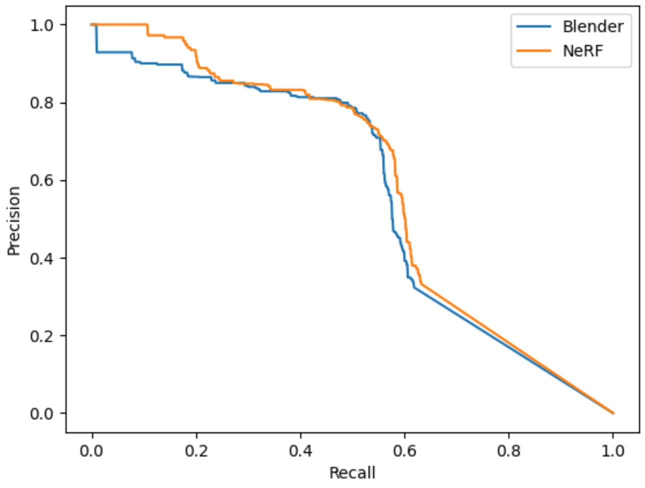


Figure 5.7: Precision vs. recall for the same spacecraft detection algorithm run on images rendered by Blender and NeRF.

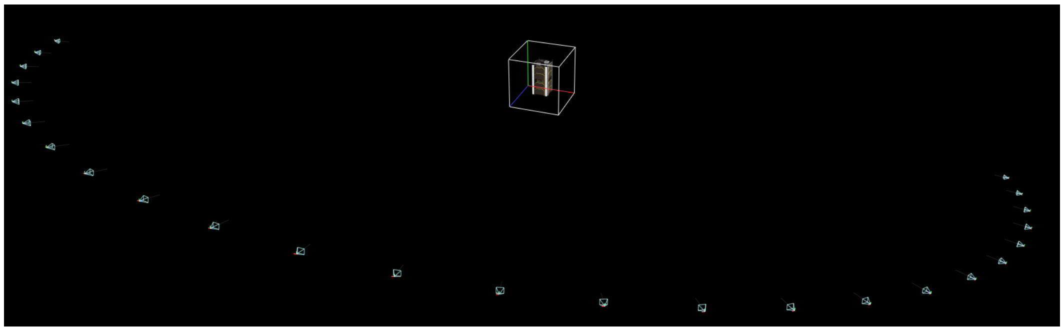


Figure 5.8: Sequence of camera views with respect to the target in frame-to-frame visual feature matching experiment. In this scenario, each camera view is separated by 6 degrees separation.

the camera trajectory was designed such that the camera moves around the target in a circular motion at a specified separation angle while pointed at the target. An example camera trajectory is shown in Fig. 5.8. We also applied the same keypoint detection and matching to an almost identical set of images generated by Blender for comparison.

At each step in the camera trajectory, we applied the keypoint matching between the pair of images from two consecutive camera poses. The Scale-Invariant-Feature-Transform (SIFT) algorithm was used for keypoint detection, and standard filtering methods such as Lowe's ratio test and RANSAC were applied to remove outliers. An example of keypoint matching is shown in Fig. 5.9.

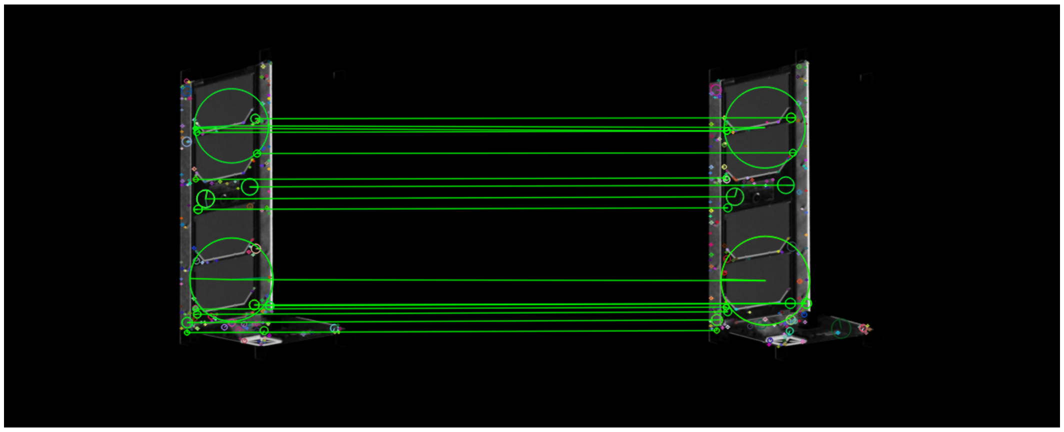
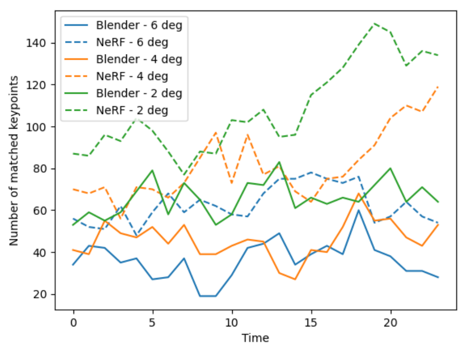
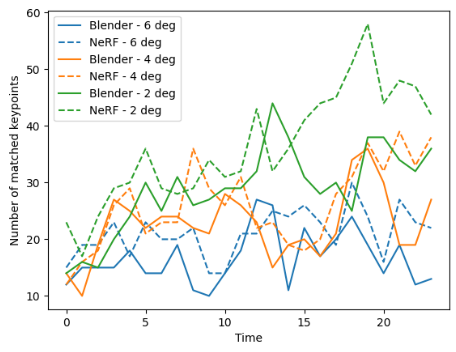


Figure 5.9: Keypoint match between two successive NeRF images.



(a) No filtering based on keypoint size



(b) Keypoint size greater than 4 pixels

Figure 5.10: Number of good keypoint matches between images rendered via NeRF and Blender.

To quantitatively evaluate the performance of the keypoint matching algorithm, we plotted the number of matches at each time step as the camera pose advances along the trajectory. We ran experiments for different separation angles of 2, 4, and 6 degrees between each frame. Fig. 5.10 provides plots that show the number of detected matches as a function of the time step for two different filtering schemes. The different separation angles are shown in different colors, with results for Blender and NeRF in solid and dashed line types, respectively. For Fig. 5.10a, the keypoint matching process was implemented according to the above description. The NeRF images had a larger number of matched features in comparison to the Blender images. This is partly because the algorithm detected more keypoints in the NeRF-rendered images. These additional keypoints were often small in size. The hypothesis for the additional detected keypoints is due to the difference in the rendering of textureless surfaces (e.g., smooth metal)—Blender results are relatively smooth and do not exhibit much variation in intensity, whereas NeRF results contain minor local variations due to the imprecision in the learned model. Because the artifacts learned as a part of the NeRF model have consistent appearances when viewed from a similar perspective, they are detected and matched as coherent keypoints.

Since many of these additional detected features were small, we additionally filtered keypoints based on a minimum threshold size of 4. The number of matched keypoints after applying this filter is shown in Fig. 5.10b. With this filter in place, the number of matched keypoints is more similar between NeRF-based and Blender-based images.

Discussion

Overall, the images generated via ROS-NeRF appear reasonably photo-realistic and have the potential to be used as tools to validate vision-based navigation algorithms in space. The appearance variations due to lighting variations can be specified and accurately rendered by augmenting the inputs of the MLP with lighting direction. When we compared the ROS-NeRF-based images against Blender-based "ground truth" images, vision-based navigation tasks performed similarly for both sets of images.

When comparing the object detection and keypoint matching tasks, the object detection algorithm performed better on the ROS-NeRF images. The results of the object detection performed on ROS-NeRF-based images were on par with those performed on ground truth images. Conversely, keypoint matching algorithm results showed some differences between ROS-NeRF-based and Blender-based images. This is likely explained by the fact that the object detection task infers an object from larger patches of the input image than keypoint detection. Because keypoint matching is more sensitive to local appearance variation, the algorithm is more affected by minor imprecisions from the ROS-NeRF model.

Finally, while ROS-NeRF has the potential to be used as a camera rendering module for vision-based navigation tasks (and more generally to be used as a part of vision-based navigation and mapping tasks), there are still challenges to overcome with this approach. For example, some rendered images may contain "floaters"—artifacts that are not part of the actual 3D scene and often look like they are floating in space. One can also incorporate various NeRF extensions for faster training and rendering speed [104] and multi-resolution [109] support.

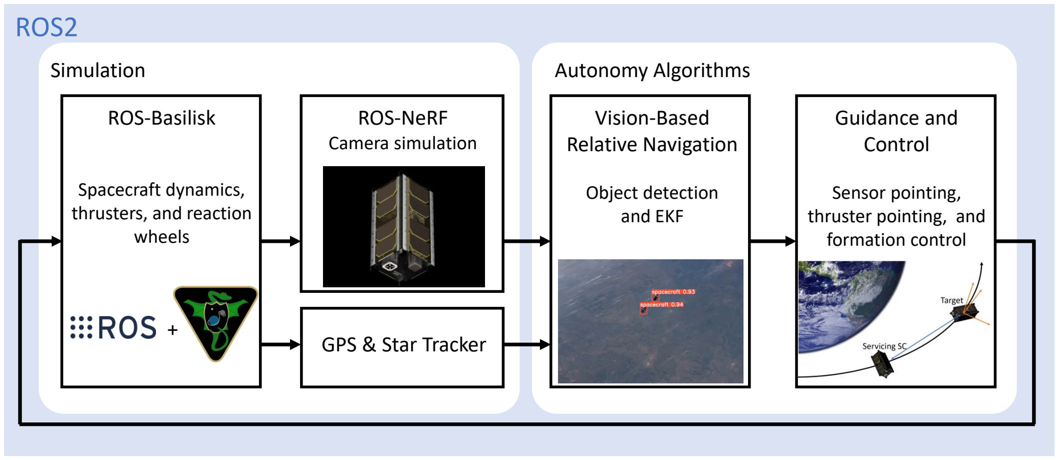


Figure 5.11: Closed-loop simulation for on-orbit inspection scenario.

5.4 Autonomous Algorithm Stack for On-Orbit Inspection

In the previous section, we described two new simulation tools: ROS-Basilisk (Section 5.2) and ROS-NeRF (Section 5.3). In this next section, we demonstrate how these tools can be applied to aid in the development of autonomy algorithms for orbital space systems. We do this by developing an example algorithm pipeline for on-orbit inspection mission scenarios and testing the algorithms in simulations using ROS-Basilisk and ROS-NeRF in a closed-loop fashion. The high-level idea of this fully integrated simulation is shown Fig. 5.11.

The example mission scenario we consider is the following. An observer spacecraft is inspecting another uncooperative, free-orbiting CubeSat in Low Earth Orbit. The observer spacecraft is equipped with a camera, a GPS, a star tracker, a thruster, and a set of three-axis reaction wheels. The observer uses a relative navigation camera sensor to visually track the target. The observer also uses the thruster to maneuver and maintain a tight formation at a 30 m separation distance.

The autonomous on-orbit inspection algorithm consists of the following components. First, the vision-based spacecraft detection module detects the target spacecraft. The outputs of the spacecraft detection, GPS, and star tracker measurements are fused in an Extended Kalman Filter (EKF) to estimate the 3D positions and velocities of the servicing and target spacecraft. Then, a formation-keeping controller uses the navigation estimate and Linear Quadratic Regulator (LQR) to maintain formation. The attitude guidance module switches between two tasks: pointing a camera to the target and pointing the thruster to the delta-V axis. The attitude controller tracks the desired pointing. All of these algorithms are packaged in ROS2 modules.

The following sections describe the implementation details for vision-based spacecraft detection, relative and absolute navigation module, and formation keeping and attitude control.

Vision-based Spacecraft Detection

We apply a CNN-based object detection technique to spacecraft detection task. We use the state-of-the-art object detection network YOLOv5 [112] and train it on a custom dataset specialized for spacecraft detection tasks. For more efficient training, we apply transfer learning, where we initialize the neural network training with the weights pre-trained on a generic object detection dataset.

The vision-based spacecraft detection algorithms used consists of an offline and online phase. In the offline phase, we applied transfer learning; that is we additionally trained a pre-trained neural network for spacecraft detection tasks using a custom dataset. In the online phase, we used the trained neural network to detect the uncooperative target spacecraft in the real-time stream of images obtained from the (simulated) camera sensor. The pre-trained neural network was provided by standard object detection tools from computer vision.

To train the object detection algorithm for spacecraft detection tasks, we generated a custom-labeled dataset. First, we generated a set of synthetic images of spacecraft with transparent backgrounds. They are generated in part using ray-tracing software, and in part by augmenting images of generic spacecraft scraped from the internet. Second, we render a set of background images of Earth using Blender, with a variety of camera poses, lighting conditions, and cloud appearances in Low Earth Orbit. Then, we overlay the spacecraft image onto the background image, with randomly determined spacecraft size and position. A total of 2000 images were generated, each having the spacecraft labeled with a tight bounding box. Fig. 5.12 summarizes the process of generating the labeled dataset. We consider a single classification category of “spacecraft” for detection.

Next, the YOLOv5 network is trained with the labeled dataset of spacecraft. We initialize the training with weights pre-trained on the COCO dataset [117]. After the neural network was trained, the model was validated on an unseen validation dataset, including some real images obtained from previous missions. For instance, Fig. 5.13 shows the object detection results on a real, space-borne image of MinXSS and CADRE spacecraft being deployed from ISS.

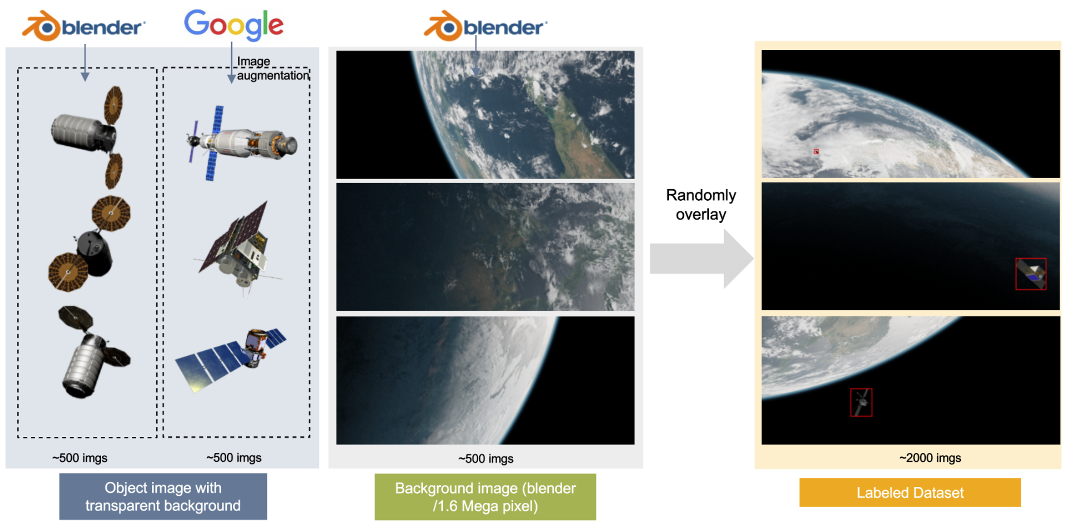


Figure 5.12: Generating a labeled dataset for training the spacecraft detection algorithm.

Finally, we implemented the spacecraft detection algorithm with the ROS2 interface. Module subscribes to the images (either from a real camera or from a simulated camera module like ROS-NeRF) and publishes the bounding boxes corresponding to detected spacecraft.

Relative and Absolute Navigation Module

After the object is correctly detected, the next step of the autonomy algorithm is to estimate the 3D positions and velocities of the spacecraft. The navigation module jointly estimates the 3D positions and velocities of both the servicing and the target spacecraft via an Extended Kalman Filter (EKF). The available information is the noisy GPS measurements of the servicing spacecraft and the object detection results. The navigation module uses the noisy star-tracker measurements to estimate the servicing spacecraft's attitude. Assuming that *a priori* information on the target spacecraft scale is available, we used the size of the detected bounding box to compute the coarse range estimate while the center of the bounding box is for bearing. Since the likelihood of error in object detection is non-trivial, we heuristically rejected the outliers by thresholding at a confidence bound calculated by projecting covariance estimates onto bearing measurement space.

Formation Keeping and Attitude and Control

The position and velocity estimates from the navigation module were inputted into the formation keeping and attitude control. The module is responsible for (1)

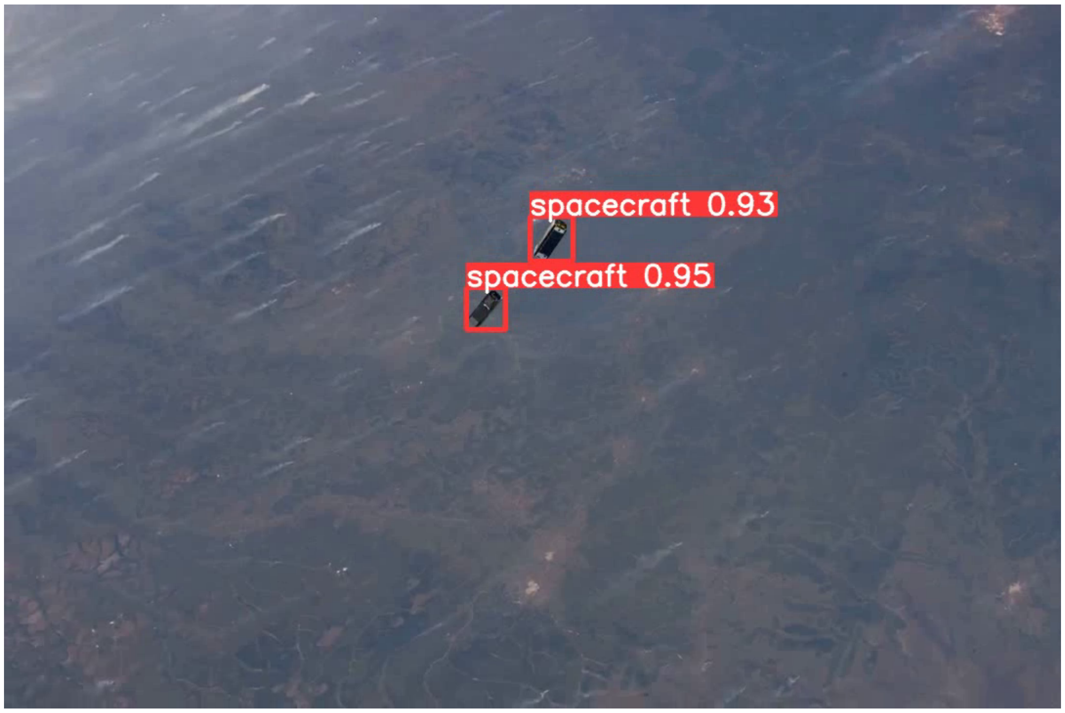


Figure 5.13: Spacecraft detection algorithm classifying MinXSS (bottom) and CADRE spacecraft (top) from the real images taken from the International Space Station.

maintaining along-track formation with respect to the target spacecraft by applying periodic burns and (2) pointing the camera at the target whenever possible. For translational positions and velocities control, we applied a Linear Quadratic Regulator (LQR) with linearized relative orbital dynamics at a low rate of every 200 sec. Since the servicing spacecraft had a single thruster, the attitude controller had to reorient the spacecraft to align the thrust vector to the desired direction during the burn phase. Given the desired delta-V maneuvers, the attitude control system balanced between pointing the camera at the target spacecraft and pointing the thruster in the delta-V direction. The main idea is that the spacecraft briefly "looks away" from the target for the delta-V maneuver and returns to pointing the camera once the burn is complete. This attitude-pointing schedule strategy is shown in the table in Fig. 5.14. Finally, given the attitude-pointing commands and the current attitude estimate from the navigation module, the attitude controller computed the desired torque using attitude error feedback and allocated torque to each of the three reaction wheels.

The formation-keeping and attitude control system algorithm was implemented as the G&C module in a ROS2 node. The G&C module subscribes to navigation

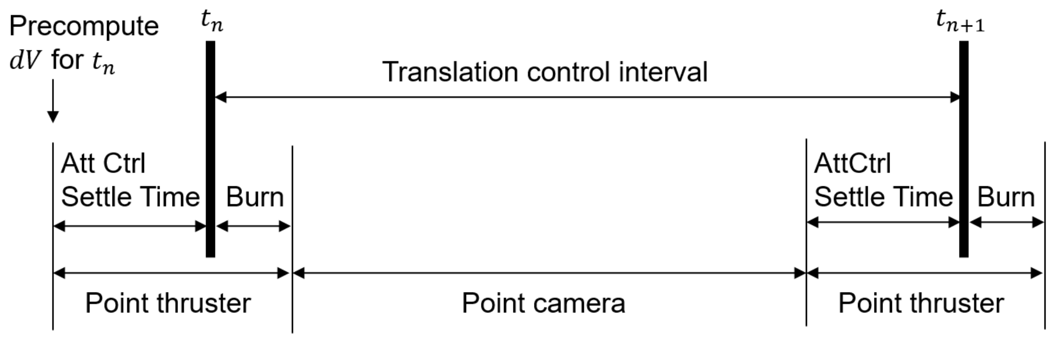


Figure 5.14: Attitude pointing schedule for alternating between delta-v maneuvers and visual tracking of the target.

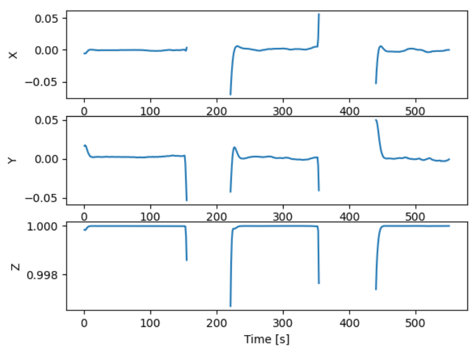
estimates and publishes the duration of thruster burns and the torque commands on each of the reaction wheels. These outgoing messages are sent to ROS-Basilisk which propagates dynamics based on these commands.

5.5 Simulation Results

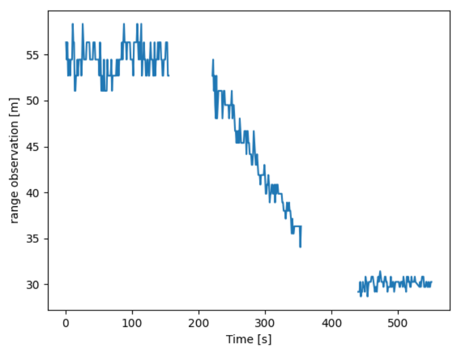
We now integrate all the components we discussed in previous sections—ROS-Basilisk, ROS-NeRF, and autonomous on-orbit inspection algorithms—in an on-orbit inspection mission scenario. In this simulation, ROS-Basilisk was responsible for modeling spacecraft dynamics and actuation hardware (i.e., thrusters and reaction wheels). The simulated ground truth states of spacecraft are sent to the sensor modules, such as ROS-NeRF, star tracker, and GPS sensors. The image and noisy GPS and star tracker measurements are sent to the navigation module which includes object detection and Extended Kalman Filtering. The navigation estimates of the target and observer spacecraft, are sent to the G&C module which calculates the necessary maneuvers for pointing and formation-keeping. The commands for the thruster and reaction wheels are sent from G&C modules back to the ROS-Basilisk module, which applies the actuation to spacecraft simulation, closing the loop. This loop is visualized in Fig. 5.11.

For this scenario, a servicing spacecraft was tasked to visually track an uncooperative target spacecraft and maintain a close formation 30 m distance. The two spacecraft were initially separated by 50 m. Spacecraft dynamics and attitude control ran at a high rate of 5 Hz. Image rendering, image detection, position EKF, and attitude guidance ran at 1Hz. The simulation was carried out on a desktop computer with Intel i7-8700 @ 3.20GHz as CPU and GTX 1070 (8GB memory) as GPU.

Bearing and range observations computed from the object detection algorithm are



(a) Bearing measurements (unit vector)



(b) Range measurements

Figure 5.15: Bearing and range measurements derived from the bounding box computed from the vision-based spacecraft detection algorithm.

shown in Fig. 5.15. There are measurements drop out seen between 50-220 seconds and 360-440 seconds; these are due to the intentional "look away" maneuvers where the observer is prioritizing pointing the thruster for the delta-v maneuvers instead of pointing the camera to the target. As soon as the observer points the camera back to the target (around 220 and 440 sec), the observations for the target spacecraft are re-acquired. The range observation indicates that the observer spacecraft begins to approach closer to the target after the first maneuver, and maintains approximately 30 m separation after the second maneuver.

Next, Fig. 5.16 shows the 3D position estimation errors for both the target and servicing spacecraft. The figures show that the positions of both the servicing and target spacecraft are accurately estimated within a few meters. The 3-sigma bounds for the target position estimate show that every 200 sec, the uncertainty of the estimate grows during the look-away maneuvers. As soon as the camera is pointed back to the target and the spacecraft is detected, the estimation uncertainty reduces back to a nominal value. Even though the servicing spacecraft loses sight of the target temporarily, the navigation module propagates the last known states with sufficient accuracy to visually acquire the target spacecraft.

Finally, Fig. 5.17 shows the relative position control tracking error with respect to the desired formation. Initially, the spacecraft is separated by approximately 50 m but the formation-keeping control reduces the error to 30 m.

In summary, the numerical simulation demonstrated an autonomous on-orbit inspection mission. Using dynamics from ROS-Basilisk and images generated by ROS-NeRF, we validated the autonomous algorithm can perform relative naviga-

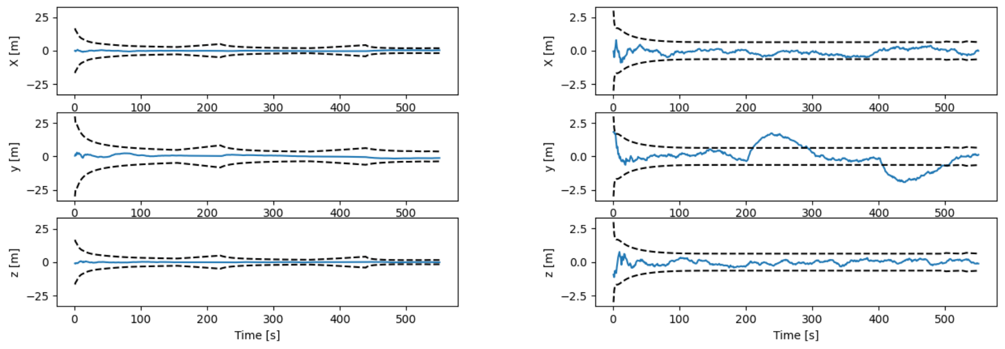


Figure 5.16: Position estimation error of target (left) and servicing (right) and 3-sigma bound.

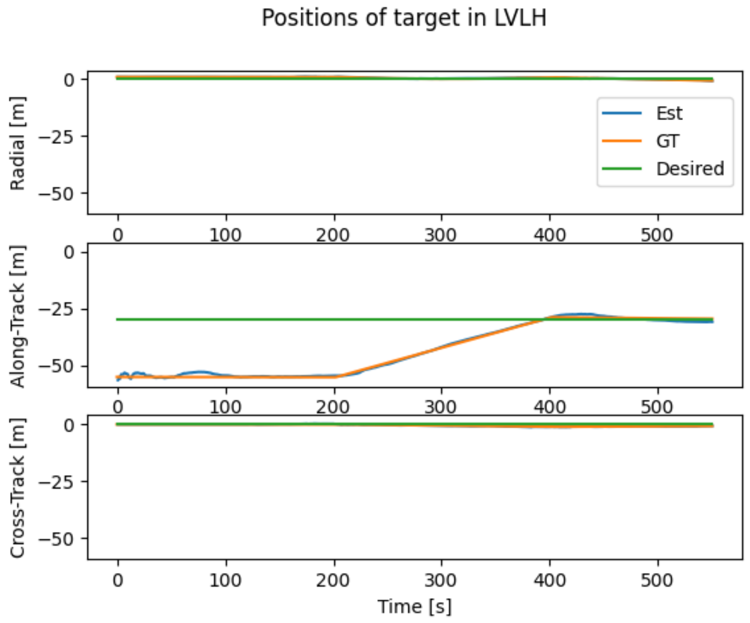


Figure 5.17: Relative position of the target with respect to the servicing spacecraft in LVLH frame. Blue is the estimated position, orange is the ground truth from the simulation and green is the desired formation.

tion and formation control while visually inspecting the target spacecraft.

5.6 Chapter Summary

In this chapter, we described two, new simulation tools for validating autonomy algorithms in space applications—ROS-Basilisk and ROS-NeRF. For the dynamics, we developed ROS-Basilisk, a ROS2 software interface that wraps Basilisk, an open-source astrodynamics simulation software. ROS-Basilisk allows flight software written in ROS2 to interact with high-fidelity spacecraft dynamics simulation in the loop. For the camera sensor, we integrated ROS-NeRF which uses the Neural Radiance Fields to rapidly render, in real-time, spacecraft images from the provided perspective and lighting conditions. We validated these simulation tools by considering an example mission scenario where an observer spacecraft uses vision-based navigating to maintain tight formation with respect to an uncooperative target spacecraft. We developed an autonomous on-orbit inspection algorithm pipeline, which included vision-based object detection, relative and absolute navigation, pointing, and formation-keeping controls. Finally, we integrated all of the ROS-Basilisk and ROS-NeRF simulation tools along with autonomous inspection algorithms in a single ROS2 simulation in a closed-loop fashion. We demonstrated that the simulation tools we developed can effectively validate the autonomous algorithms for on-orbit inspection.

Chapter 6

DISTRIBUTED FACTOR GRAPH OPTIMIZATION

This chapter contains material from the following publication:

[1] K. Matsuka and S.-J. Chung, "Localized and incremental probabilistic inference for large-scale networked dynamical systems," *IEEE Transactions on Robotics (Conditionally accepted for publication)*, 2023,

6.1 Introduction

While DPE in Chapter 3 solved the scalability with the number of agents in the swarm, it was not *the* optimal solution (i.e., the estimate did not coincide with the optimal solution of the centralized problem). In this chapter, we take a step back from the spacecraft-specific applications and address the limitations of existing estimation algorithms in a more general class of locally coupled dynamical systems. We develop novel algorithms that are both *scalable* and *optimal* for locally-coupled dynamical systems with an arbitrarily large number of agents. We achieve this by modeling the estimation problem as a distributed factor graph optimization (DFGO) and developing a novel algorithm to solve it in a scalable fashion.

To solve the DFGO, we first apply a type of Alternating Direction Method of Multipliers (ADMM) algorithms [43], [118], [119] called the Local Consensus ADMM (LC-ADMM). LC-ADMM extends the Separable Optimization Variable ADMM (SOVA) [41], [42] which is a distributed and localized algorithm for large-scale optimization. We provide new theoretical results for explicit convergence rates under certain convex assumptions as well as provide useful interpretations of the algorithm in terms of factor graphs.

LC-ADMM has multiple properties that are useful to DFGO. First, LC-ADMM naturally allows the use of various types of convex loss functions such as ℓ_1 and Huber loss for sparse outlier rejection. Second, LC-ADMM does not require computing any global topology information (e.g., graph coloring), making it suitable for ad-hoc or large networks. Third, LC-ADMM does not need to compute a summarized graph or exchange probability distribution parameters such as covariances [18]. Finally, *all* agents run their algorithms in parallel simultaneously rather than taking turns [3], thereby improving computational speed. While some of these properties can be seen

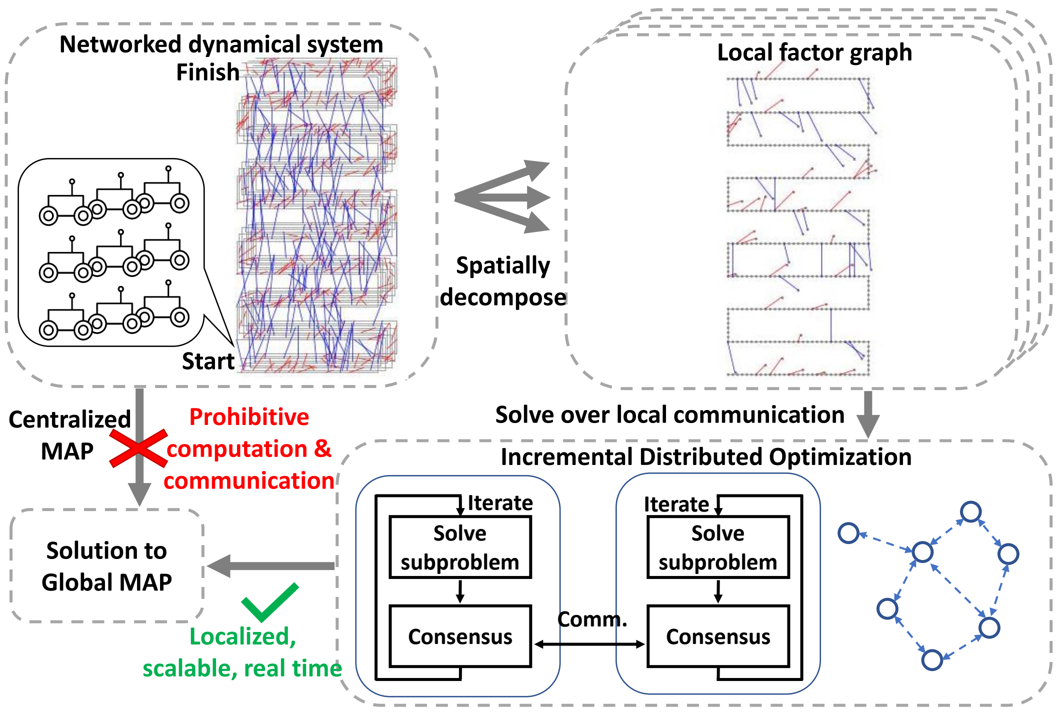

Networked dynamical system
Finish
Start
Spatially decompose
Local factor graph
Centralized MAP
Prohibitive computation & communication
Solve over local communication
Incremental Distributed Optimization
Iterate
Solve subproblem
Consensus
Comm.
Iterate
Solve subproblem
Consensus
Solution to Global MAP
Localized, scalable, real time

Figure 6.1: Our approach to solving the locally coupled, multi-agent factor graph optimization problem of large-scale networks. The LC-ADMM algorithm solves the distributed factor graph optimization problem in a localized fashion in batch. The iDFGO algorithm solves the real-time problem in an incremental fashion.

in previous approaches to DFGO [3], [25], [120], they have not simultaneously been applied in a single framework. By applying the ideas from the ADMM literature [41], [42], we provide a new approach that has these properties.

In addition to LC-ADMM, we also develop the Incremental DFGO (iDFGO) algorithm to solve the scalability of DFGO with respect to time horizon. The iDFGO algorithm builds upon LC-ADMM and leverages the tools we have for the single-agent incremental FGO (i.e., iSAM2 [23]). Simply applying iSAM2 to the local FGO update step of DFGO algorithms does not actually lead to a time-scalable algorithm. Instead, our iDFGO algorithm makes some unique extensions to iSAM2 such that each local FGO update step is computed in a way that is scalable in time. In numerical simulations, we simulate LC-ADMM/iDFGO on hundreds of agents and on multi-agent pose graph optimization problems to validate their scalability, convergence, optimality, and other properties.

The main concepts for our approach are summarized in Fig. 6.1. Given a networked dynamical system with a large number of agents, solving a centralized problem may be prohibitively computationally expensive. Our approach spatially decomposes the

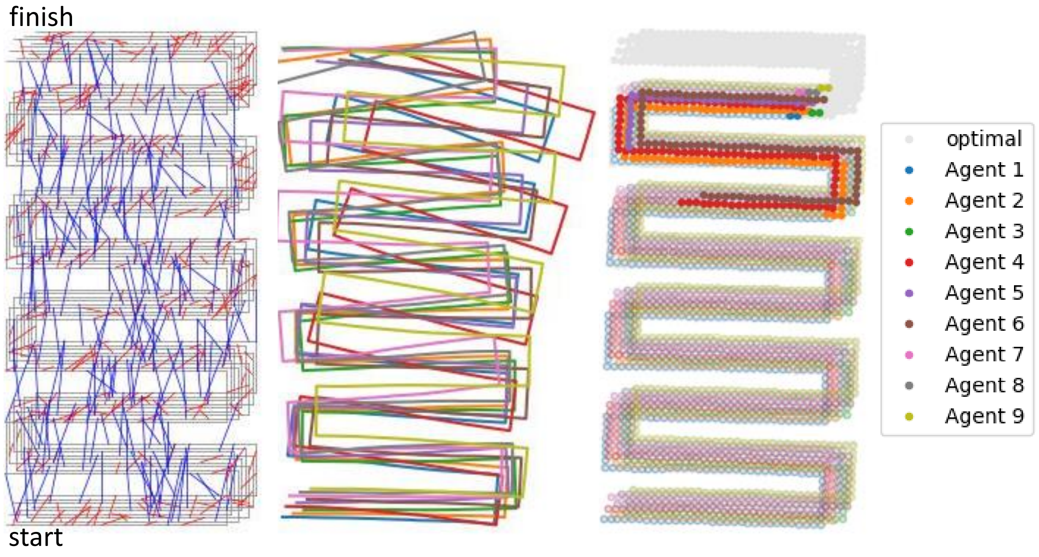


Figure 6.2: Multi-agent pose graph optimization solved via iDFGO. Left: Simulated trajectories of 9 robots traversing the environment in a formation. Noisy data contains odometry (grey), inter-agent observations at a given time (red), and self/inter-agent loop closure constraints (blue). Center: Individual estimates without any collaboration result in large errors. Right: Nine robots collaboratively estimate the trajectories using iDFGO (colored). The filled dots represent poses that are recomputed incrementally in this time frame and non-filled dots represent poses that were not modified. Updating only a subset of the trajectory facilitates scalability with respect to time.

problem into smaller, localized factor graphs. The agents iteratively (1) solve their local graphs in parallel and (2) exchange information, to arrive at the solution to the original centralized problem. This approach allows us to solve the problem in a localized and scalable fashion.

Summary of Contributions and Paper Organization

The rest of the paper is organized as follows. Section 6.2 considers the distributed factor graph optimization problem for one particular time horizon and derives the LC-ADMM algorithm. Section 6.3 establishes theoretical guarantees of convergence as well as some of the other mathematical properties of LC-ADMM. Section 6.4 extends LC-ADMM to real-time problems and develops the Incremental iDFGO algorithm. Section 6.5 gives the numerical validation of the algorithms in practical examples. Finally, Section 6.6 presents concluding remarks.

6.2 Problem Statement

Prior to defining the problem statement, we first introduce some of the notations that will be used throughout the paper. The norm notation $\|\cdot\|$ without any subscript denotes the ℓ_2-norm. $\|\cdot\|_F$ denotes the Frobenius norm. The weighted norm for some square matrix $Q \in \mathbb{R}^{n\times n}$ is defined as $\|x\|_Q = x^\top Qx$. The maximum singular value of the matrix $Q \in \mathbb{R}^{m\times n}$ is denoted as $\sigma_{\max}(Q)$ and the minimum non-zero singular value is denoted as $\tilde{\sigma}_{\min}(Q)$. The identity matrix of dimension $N \times N$ is denoted as I_N. The set of positive integers is denoted as $\mathbb{Z}_+$. Given a set $\mathcal{S}$, its index set is denoted by $\mathcal{I}(\mathcal{S})$, such that $\mathcal{S} = \{s_i \mid i \in \mathcal{I}(\mathcal{S})\}$. The cardinality of a set $\mathcal{S}$ is denoted as $|\mathcal{S}|$. A Cartesian product of sets is denoted as $\mathcal{S}_1 \times \mathcal{S}_2$ or $\prod_i \mathcal{S}_i$.

A network of agents is modeled as an undirected graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ where $\mathcal{A}$ is the set of agents and $\mathcal{E} \subseteq \mathcal{A}\times\mathcal{A}$ is the set of edges. We say (i, j) is in the set of edges $\mathcal{E}$ *iff* the i-th and the j-th agents are connected. The set of neighbors for the i-th agent is given by $\mathcal{N}^i = \{j \in \mathcal{A} \mid (i, j) \in \mathcal{E}\}$ and the closed set of neighbors is defined as $\bar{\mathcal{N}}^i \triangleq \mathcal{N}^i \cup \{i\}$. We refer to $\mathcal{G}$ as the *physical graph*, in order to distinguish it from the other types of graphs used in this thesis.

Motivating Example

A class of problems that motivates the development of LC-ADMM and iDFGO is estimating the state of a network of robots or locally coupled dynamical systems as described in Section 2.2. For example, consider the multi-agent SLAM problem where a group of robots is tasked to collectively map an unknown environment with partial overlaps. Another example is the localization of swarms of agents, where the relative measurements locally couple the neighboring agents. In such scenarios, the challenge is to find an optimal solution in a distributed and scalable fashion.

Suppose the state of agent i at time τ is denoted as $r_{i,\tau}$, and agent i makes observations $y_{i,\tau} \in \mathbb{R}^{m_i}$. The dynamical system of a group of locally coupled robots is given by:

$$\begin{aligned} r_{i,\tau} =& a_{i,\tau}(r_{\bar{\mathcal{N}}^i,\tau-1}) + w_{i,\tau}, \quad \tau \in \mathbb{Z}_+, i \in \mathcal{A}, \\ y_{i,\tau} =& c_{i,\tau}(r_{\bar{\mathcal{N}}^i,\tau}) + v_{i,\tau}, \quad \tau \in \mathbb{Z}_+, i \in \mathcal{A}, \end{aligned} \tag{D1}$$

where $r_{\bar{\mathcal{N}}^i,\tau} \triangleq \{r_{j,\tau} \mid j \in \bar{\mathcal{N}}^i\}$ denotes the set of states at time τ in the closed neighborhood and the random noise $w_{i,\tau} \in \mathbb{R}^{p_i}$ and $v_{i,\tau} \in \mathbb{R}^{q_i}$ are assumed to be independent. Each agent's dynamics and measurement ($a_{i,\tau}$ and $c_{i,\tau}$, respectively) depend on its states as well as those of its neighbors. The system in (D1) models a variety of local interactions such as the downwash interaction of quadcopters [74] or relative range and bearing between robots.

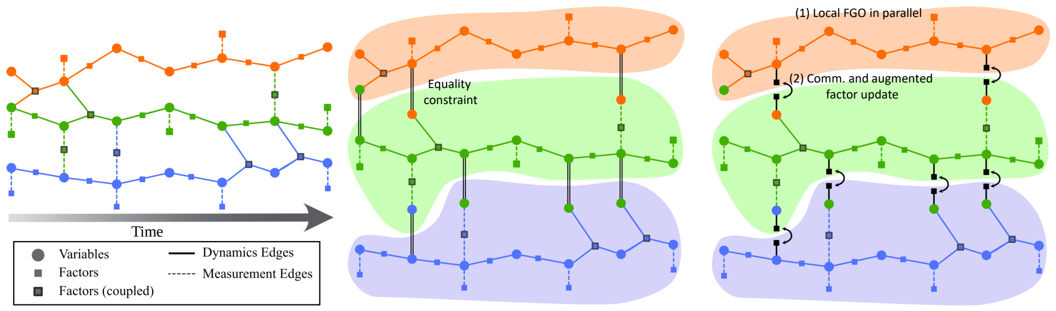


Figure 6.3: Visual representation of factor graphs for an estimation problem involving three robots: orange, green, and blue. Left: The locally-coupled multi-agent FGO in (P1). Center: local consensus optimization problem (P2) replaces local coupling with equality constraints (black double lines), which adds augmented factors. Right: LC-ADMM solves (P2) by iterating between: (1) all robots solve the local augmented FGO in parallel and (2) neighboring robots exchange information and update the pseudo measurements of the augmented factors.

The team of robots is tasked with collaboratively computing the Maximum A Posteriori (MAP) estimate given the set of all the observations available up to the current time t. In this example, the set of the state to be estimated is $\mathcal{X}_t = \{r_{i,\tau} \,|\, i \in \mathcal{A}, \tau = [0,t]\}$ and the set of measurements available is $\mathcal{Y}_t = \{y_{i,\tau} \,|\, i \in \mathcal{A}, \tau = [0,t]\}$. The MAP estimate at t is given by

$$\mathcal{X}_{t,*} \triangleq \arg\max_{\mathcal{X}_t} p(\mathcal{X}_t|\mathcal{Y}_t) = \arg\max_{\mathcal{X}_t} p(\mathcal{Y}_t|\mathcal{X}_t)p(\mathcal{X}_t), \tag{6.1}$$

where $p(\mathcal{X}_t)$ is the prior distribution and $p(\mathcal{Y}_t|\mathcal{X}_t)$ is the likelihood function. The prior distribution represents the probabilistic relations according to the dynamics of the system, while the likelihood function describes the observations made by the network. The remainder of Sections 6.2 and 6.3 focus on solving (6.1) for a fixed time horizon $\mathcal{T}(t)$ using LC-ADMM. For readability, the subscript t from $\mathcal{X}_t$ and $\mathcal{Y}_t$ are dropped in these sections.

Factor Graph Optimization of Multi-agent Systems

The optimization in (6.1) can be viewed a factor graph optimization problem. Let $\mathcal{X} = \{x_s\}$ be the variables to be estimated and $\mathcal{Y} = \cup_{i\in\mathcal{A}}\mathcal{Y}^i$ be the set of measurements where $\mathcal{Y}^i$ is the i-th agent's observations. The definitions of $\mathcal{X}$ and $\mathcal{Y}^i$ here can be more general than those we considered in example (D1).[1] Each element x_s is referred to as a *variable node* where the subscript s is used to index the variable nodes in $\mathcal{X}$ throughout this chapter.

[1]For example, $\mathcal{X}$ may include non-robot states such as landmarks or time-varying target states and $\mathcal{Y}$ may include loop closure observations.

We assume that the joint probability distribution in (6.1) can be factored as

$$p(\mathcal{X}, \mathcal{Y}) = \prod_{i \in \mathcal{A}} \prod_{\phi_f \in \mathcal{F}_i} \phi_f(\mathcal{X}^{\phi_f}),$$

where each ϕ_f is an independent probability distribution depending on a subset of variables $\mathcal{X}^{\phi_f} \subset \mathcal{X}$. The set $\mathcal{F}_i = \{\phi_f\}$ is the set of *local factors* corresponding to the i-th agent. It contains the probability distributions pertaining to the i-th agent and we assume $\mathcal{F}_i \cap \mathcal{F}_j = \varnothing$ for $\forall i, j \in \mathcal{A}$, $i \neq j$. The set of all the factors in the network is defined as $\mathcal{F} \triangleq \cup_{i \in \mathcal{A}} \mathcal{F}_i$. The *factor graph* modeling the distribution $p(\mathcal{X}, \mathcal{Y})$ is defined as the bipartite graph $\mathcal{G}_F = (\mathcal{X}, \mathcal{F}, \mathcal{E}_F)$ consisting of variable nodes $\mathcal{X}$, factor nodes $\mathcal{F}$, and edges $\mathcal{E}_F$. The graph $\mathcal{G}_F$ is said to have an edge $(\phi_f, x_s) \in \mathcal{E}_F$ *iff* $\phi_f \in \mathcal{F}$ depends on $x_s \in \mathcal{X}$.

An example factor graph for the locally coupled dynamical system in (D1) is shown on the left in Fig. 6.3. Note that factor graphs are often characterized by both spatial and temporal sparsity. As we reformulate the DFGO and derive LC-ADMM, we will refer to Fig. 6.3.

The local factors for the i-th agent, $\mathcal{F}_i$, depend on the *local subset of variable nodes*, which is defined as

$$\mathcal{X}^{(i)} \triangleq \left\{x_s \in \mathcal{X} \mid (\phi_f, x_s) \in \mathcal{E}_F \text{ for some } \phi_f \in \mathcal{F}_i\right\} \subseteq \mathcal{X}.$$

Using this definition, the MAP estimation problem in (6.1) can be re-written as

$$\min_{\mathcal{X}} \quad \sum_{i \in \mathcal{A}} f_i\left(\mathcal{X}^{(i)}\right) \tag{P1}$$

where each $f_i(\mathcal{X}^{(i)}) \triangleq -\sum_{f \in \mathcal{I}(\mathcal{F}_i)} \log\left(\phi_f(\mathcal{X}^{\phi_f})\right)$ is the objective. For large-scale estimation problems that involve many agents, we often have $|\mathcal{X}^{(i)}| \ll |\mathcal{X}|$, and $|\mathcal{X}^{(i)}|$ is independent of the network size. In other words, if the optimization problem in (P1) could be decoupled, the problem that each agent solves will be small. However, the objective in (P1) is still locally coupled because $\mathcal{X}^{(i)} \cap \mathcal{X}^{(j)}$ is not empty for $i \neq j$ in general.

The local coupling of the factor graph for (P1) can be visualized on the left diagram of Fig. 6.3. Variables are shown in circles and factors relating to variables are shown in squares. Local coupling is introduced by a subset of variables that are shared by the factors of multiple robots. If one was to solve (P1) jointly using standard FGO solvers (i.e., g2o[27], GTSAM[23]), it will not scale with respect to the number of robots in the network.

Local Consensus Reformulation

Instead of solving (P1) directly, we solve a closely related *local consensus optimization problem* (P2) that allows us to partially decouple (P1) which can be later solved by LC-ADMM. As we shall see, (P1) and (P2) are equivalent under certain conditions. To develop (P2), let us first denote the i-th agent's estimate for a variable node x_s as $x_s^{(i)}$ and define the set of local estimates[2] with respect to agent i as

$$\hat{\mathcal{X}}^{(i)} \triangleq \{x_s^{(i)} \mid s \in \mathcal{I}(\mathcal{X}^{(i)})\}. \tag{6.2}$$

Then, the local consensus optimization problem is written as

$$\begin{aligned} \min_{X,Z} \quad & \sum_{i\in\mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)}) && \text{(P2)} \\ \text{subject to} \quad & \begin{aligned} x_s^{(i)} - z_s^{(ij)} &= 0, \\ x_s^{(j)} - z_s^{(ij)} &= 0, \end{aligned} \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}. \end{aligned}$$

Each $z_s^{(ij)}$ is shared between agents i and j only and its role is to enforce the equality between $x_s^{(i)}$ and $x_s^{(j)}$. The set $\mathcal{I}^{ij} \triangleq \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \cap \mathcal{I}(\hat{\mathcal{X}}^{(j)})$ is the index set for all the variables shared between agents i and j. In the new problem, the decision variables are

$$\begin{aligned} X &\triangleq \cup_{i\in\mathcal{A}}\hat{\mathcal{X}}^{(i)} = \{x_s^{(i)} \mid \forall x_s^{(i)} \in \hat{\mathcal{X}}^{(i)}, \forall i \in \mathcal{A}\}, \\ Z &\triangleq \{z_s^{(ij)} \mid s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}\}. \end{aligned}$$

The visual representation of (P2) can be seen in the center diagram of Fig. 6.3. Compared to the factor graph representation of (P1) shown on the left, each agent now has copies of the shared variables. For each shared variable, the consistency between the multiple copies is enforced by the equality constraints as given in (P2).

Problems (P1) and (P2) are equivalent when a condition called variable connectivity [40] is satisfied. For each $s \in \mathcal{I}(\mathcal{X})$, we define the set of *co-dependent agents* as

$$\mathcal{A}_s \triangleq \{i \in \mathcal{A} \mid x_s^{(i)} \in \hat{\mathcal{X}}^{(i)}\}.$$

In other words, agent $i \in \mathcal{A}$ is part of $\mathcal{A}_s$ *iff* it has a local estimate for x_s. The induced subgraph of the physical graph $\mathcal{G}$ with respect to the variable node x_s is the undirected graph $\mathcal{G}_s \triangleq (\mathcal{A}_s, \mathcal{E}_s)$, where $\mathcal{E}_s \subseteq \mathcal{E}$ is given by

$$\mathcal{E}_s \triangleq \{(i,j) \in \mathcal{E} \mid i \in \mathcal{A}_s \text{ and } j \in \mathcal{A}_s\}.$$

[2]Later, we discuss how the definition of $\hat{\mathcal{X}}^{(i)}$ can be optionally modified to include additional variables. However, unless mentioned otherwise, we assume the definition of $\hat{\mathcal{X}}^{(i)}$ is as given in (6.2) (i.e., $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) = \mathcal{I}(\mathcal{X}^{(i)})$).

To establish the equivalence of (P1) and (P2), we make the following assumption on the connectivity of the network.

Assumption 1 (Variable Connectivity [40]). *For all $x_s \in \mathcal{X}$, the corresponding undirected induced subgraph $\mathcal{G}_s$ is connected.*

Later, we discuss the various implications of Assumption 1. Equipped with Assumption 1, we have the following lemma.

Lemma 1 (Equivalence of (P1) and (P2)). *Suppose Assumption 1 holds. Suppose $X_* = \cup_{i\in\mathcal{A}}\hat{\mathcal{X}}_*^{(i)}$, where $\hat{\mathcal{X}}_*^{(i)} = \{x_{s,*}^{(i)} \mid s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)})\}$, and $Z_* = \{z_{s,*}^{(ij)} \mid s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}\}$ are the feasible solutions of* (P2). *Then there exists a feasible solution $\mathcal{X}_* = \{x_{s,*} \mid s \in \mathcal{I}(\mathcal{X})\}$ to* (P1)*, such that*

$$
\begin{aligned}
x_{s,*} &= x_{s,*}^{(i)}, \quad \forall s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)}), i \in \mathcal{A},\\
x_{s,*} &= z_{s,*}^{(ij)}, \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}.
\end{aligned}
$$

Proof. Consider $x_s \in \mathcal{X}$ for which multiple agents are co-dependent agents ($|\mathcal{A}_s| \geq 2$). The feasible solution to (P2) satisfies the equality constraints in (P2), we have $x_{s,*}^{(i)} = x_{s,*}^{(j)} = z_{s,*}^{(ij)}$ for each $(i,j) \in \mathcal{E}$, $i,j \in \mathcal{A}_s$. We also have that the induced subgraph $\mathcal{G}_s = (\mathcal{A}_s, \mathcal{E}_s)$ is connected for variable x_s by Assumption 1, so all the local estimates for x_s have identical values. This holds true for an arbitrary $x_s \in \mathcal{X}$ with $|\mathcal{A}_s| \geq 2$. By renaming $x_s^{(i)} \to x_s$ for $\forall i \in \mathcal{A}_s$ for each $x_s \in \mathcal{X}$, we have that (P2) is equivalent to (P1). □

Lemma 1 shows that if we solve (P2), then each solution $\hat{\mathcal{X}}_*^{(i)}$ is a subset of the solution to the original centralized MAP estimation problem (P1). The next section discusses how to solve (P2) in a distributed fashion using LC-ADMM.

LC-ADMM Algorithm

We now introduce the LC-ADMM algorithm which is shown to converge to the global optimal solution (P2) in convex settings. We consider a generalized version of (P2) that additionally includes local affine equality constraints on each $\hat{\mathcal{X}}^{(i)}$.

$$
\begin{aligned}
\min_{X,Z} \quad & \sum_{i\in\mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)}) && \text{(P2a)}\\
\text{subject to} \quad & D_i\hat{\mathcal{X}}^{(i)} = E_i, \quad \forall i \in \mathcal{A}\\
& \left.\begin{aligned} x_s^{(i)} - z_s^{(ij)} &= 0,\\ x_s^{(j)} - z_s^{(ij)} &= 0, \end{aligned}\right\} \quad \forall s \in \mathcal{I}^{ij}, (i,j) \in \mathcal{E}
\end{aligned}
$$

This can be written more succinctly as an ADMM form [43]

$$\begin{aligned} \min_{X,Z} \quad & F(X) \\ \text{subject to} \quad & AX + BZ = C \end{aligned} \tag{P2b}$$

where $F(X) \triangleq \sum_{i \in \mathcal{A}} f_i(\hat{\mathcal{X}}^{(i)})$. Matrices A, B, and C are given by $A \triangleq [A_1; A_2; A_3]$, $B \triangleq [-I; -I; 0]$, and $C \triangleq [0; 0; C_3]$. The matrix A_1 (resp. A_2) is defined such that the equality constraint $A_1 X - Z = 0$ (resp. $A_2 X - Z = 0$) is a collection of consensus constraints where each block row corresponds to $x_s^{(i)} = z_s^{(ij)}$ (resp. $x_s^{(j)} = z_s^{(ij)}$) for some $s \in \mathcal{I}^{ij}$ and $(i, j) \in \mathcal{E}$. $A_3 X = C_3$ is defined such that the i-th block row corresponds to $D_i \mathcal{X}^{(i)} = E_i$ for agent i.

The augmented Lagrangian function for (P2b) is defined as

$$\begin{aligned} & L_\beta(X, Z, W) = \\ & F(X) + \langle W, AX + BZ - C \rangle + \frac{\beta}{2} \|AX + BZ - C\|^2 \end{aligned}$$

where W is the Lagrange multiplier and $\beta > 0$ is the scalar coefficient of the augmented terms[43]. Then, ADMM attempts to solve (P2b) using the following iterative algorithm:

$$\begin{aligned} X_{k+1} &= \arg\min_X L_\beta(X, Z_k, W_k), \\ Z_{k+1} &= \arg\min_Z L_\beta(X_{k+1}, Z, W_k), \\ W_{k+1} &= W_k + \beta(AX_{k+1} + BZ_{k+1} - C). \end{aligned} \tag{6.3}$$

The Lagrange multiplier W can be split into three blocks: $W = [Y; Y'; V]$. If $w_s^{(ij,i)}$ (resp. $w_s^{(ij,j)}$) is the Lagrange multiplier corresponding to $x_s^{(i)} = z_s^{(ij)}$ (resp. $x_s^{(j)} = z_s^{(ij)}$), then Y consists of all $w_s^{(ij,i)}$ (resp. Y' consists of all $w_s^{(ij,j)}$) for $\forall s \in \mathcal{I}^{ij}, \forall (i, j) \in \mathcal{E}$. Similarly, if $\mathcal{V}^{(i)}$ is the Lagrange multiplier corresponding to $D_i \hat{\mathcal{X}}^{(i)} = E_i$, V consists of $\mathcal{V}^{(i)}$ for $\forall i \in \mathcal{A}$. The iterations (6.3) can be rewritten in terms of their block components.

$$\begin{aligned} \hat{\mathcal{X}}_{k+1}^{(i)} = & \arg\min_{\hat{\mathcal{X}}^{(i)}} f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2} \|D_i \hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta} \mathcal{V}_k^{(i)}\|^2 \\ & + \sum_{j \in \mathcal{N}^i} \sum_{s \in \mathcal{I}^{ij}} \frac{\beta}{2} \|x_s^{(i)} - z_{s,k}^{(ij)} + \frac{1}{\beta} w_{s,k}^{(ij,i)}\|^2, \end{aligned} \tag{6.4}$$

$$z_{s,k+1}^{(ij)} = \frac{1}{2} \left(x_{s,k+1}^{(i)} + \frac{1}{\beta} w_{s,k}^{(ij,i)} + x_{s,k+1}^{(j)} + \frac{1}{\beta} w_{s,k}^{(ij,j)} \right), \tag{6.5}$$

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - z_{s,k+1}^{(ij)}), \tag{6.6}$$

$$\mathcal{V}_{k+1}^{(i)} = \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i). \tag{6.7}$$

Similar to other consensus ADMM algorithms such as [37], [43], it is easy to show that $z_{s,k}^{(ij)} = \bar{x}_{s,k}^{(ij)} = \frac{1}{2}(x_{s,k}^{(i)} + x_{s,k}^{(j)}), \forall k$ if we select the initial value of $w_s^{(ij,i)}$ and $w_s^{(ij,j)}$ to be $w_{s,0}^{(ij,i)} = -w_{s,0}^{(ij,j)}$[43]. Using this, the iterations (6.4)-(6.7) can be further simplified to the following form for implementation.

$$\begin{aligned}\hat{\mathcal{X}}_{k+1}^{(i)} =& \arg\min_{\hat{\mathcal{X}}^{(i)}} f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2}\|D_i\hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta}\mathcal{V}_k^{(i)}\|^2 \\ &+ \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - \bar{x}_{s,k}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2,\end{aligned} \tag{6.8}$$

$$w_{s,k+1}^{(ij,i)} = w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - \bar{x}_{s,k+1}^{(ij)}), \tag{6.9}$$

$$\mathcal{V}_{k+1}^{(i)} = \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i). \tag{6.10}$$

Remark. *These LC-ADMM iterations have a natural interpretation in terms of factor graphs. Recall that each term within $f_i(\hat{\mathcal{X}}^{(i)})$ corresponds to a factor in the local factor graph $\mathcal{G}_{Fi} = (\hat{\mathcal{X}}^{(i)}, \mathcal{F}_i, \mathcal{E}_{Fi})$. Similarly, one can view the other terms in* (6.8) *as the* augmented factors *that softly penalize the constraints. The terms such as $E_i - \frac{1}{\beta}\mathcal{V}_k^{(i)}$ and $\bar{x}_{s,k}^{(ij)} - \frac{1}{\beta}w_{s,k}^{(ij,i)}$ represent the* pseudo measurement *of the augmented factors, and the values of the pseudo measurements are modified by* (6.9)–(6.10) *at each k.*

With this perspective, the first part of LC-ADMM iteration (6.8) is referred to as the *local FGO update*. This update can be implemented as a standard factor graph optimization using single-agent solvers such as g2o or GTSAM. The second part of LC-ADMM iteration (6.9)-(6.10) is referred to as the *augmented factor update*. This step only involves information exchange and summation which can be implemented easily. We can visualize these LC-ADMM iterations in terms of the factor graph in the right diagram in Fig. 6.3.

One can execute the LC-ADMM iterations in a fully localized fashion as summarized in Algorithm 5. First, each agent independently solves the local FGO update in parallel as (6.8). Next, each agent broadcasts the shared variables $x_{s,k+1}^{(i)}$ to the neighbors who need that information. Once the neighbors' estimates are received, the average $\bar{x}_{s,k+1}^{(ij)}$ is computed and the augmented factor update (6.9) and (6.10) modify the pseudo measurements. The augmented factor update also takes place locally on each agent.

Algorithm 5: Localized Consensus ADMM

```
Result: $\hat{\mathcal{X}}_k^{(i)},\ i \in \mathcal{A}$
Each agent initializes $\hat{\mathcal{X}}_k^{(i)}$ and $\{w_{s,k}^{(ij,i)} \mid s \in \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \cap \mathcal{I}(\hat{\mathcal{X}}^{(j)}), (i,j) \in \mathcal{E}\}$ for
  $k = 0$;
while $k < K$ do
    Each $i \in \mathcal{A}$ solves its sub-problem: $\hat{\mathcal{X}}_{k+1}^{(i)} =$
        $$\begin{aligned}\underset{\hat{\mathcal{X}}^{(i)}}{\arg\min}\, & f_i(\hat{\mathcal{X}}^{(i)}) + \frac{\beta}{2}\|D_i\hat{\mathcal{X}}^{(i)} - E_i + \frac{1}{\beta}\mathcal{V}_k^{(i)}\|^2 \\ & + \sum_{j\in\mathcal{N}^i}\sum_{s\in\mathcal{I}^{ij}} \frac{\beta}{2}\|x_s^{(i)} - \bar{x}_{s,k}^{(ij)} + \frac{1}{\beta}w_{s,k}^{(ij,i)}\|^2;\end{aligned}$$
    For each neighboring pair $(i,j) \in \mathcal{E}$, the agents exchange $x_{s,k+1}^{(i)}$ and $x_{s,k+1}^{(j)}$
      for $\forall s \in \mathcal{I}^{ij}$;
    Each $i \in \mathcal{A}$ locally computes the average $\forall s \in \mathcal{I}^{ij}, \forall j \in \mathcal{N}^i$:
        $$\bar{x}_{s,k+1}^{(ij)} = \frac{1}{2}\left(x_{s,k+1}^{(i)} + x_{s,k+1}^{(j)}\right);$$
    Each $i \in \mathcal{A}$ updates its Lagrange multipliers:
        $$\begin{aligned} w_{s,k+1}^{(ij,i)} &= w_{s,k}^{(ij,i)} + \beta(x_{s,k+1}^{(i)} - \bar{x}_{s,k+1}^{(ij)}); \\ \mathcal{V}_{k+1}^{(i)} &= \mathcal{V}_k^{(i)} + \beta(D_i\hat{\mathcal{X}}_{k+1}^{(i)} - E_i)); \end{aligned}$$
    Iterate: $k \leftarrow k + 1$
end
```

Algorithmic Complexity of LC-ADMM

The LC-ADMM algorithm has several properties that make it suitable for a broad class of problems involving networked systems. First, the computational effort, communication bandwidth, and memory requirements of LC-ADMM have constant complexity with respect to the size of the network, so long as the $|\mathcal{X}^{(i)}| \sim O(1)$ assumption holds. Taking (D1) as an example, the communication bandwidth scales like $O(nT|\bar{\mathcal{N}}^i|)$, where $n \triangleq \dim(x_s)$ and $T \triangleq |\mathcal{T}(t)|$, and is independent of $|\mathcal{A}|$. Second, the agents do not need to share the probability distribution parameters (e.g., covariance matrix); therefore, the communication bandwidth scales only linearly with nT, not n^2T^2. This is an advantage compared to other distributed optimal estimation algorithms that require sharing the full probability distributions such as [9], [10].

Variable Connectivity and Augmenting Set $\hat{\mathcal{X}}^{(i)}$

In some instances, the variable connectivity assumption in Assumption 1 may not be satisfied (i.e., the agents that have the local estimate for some x_s are not connected). Consider the SLAM-like example in Fig. 6.4. The two landmarks (shown as black circles) at the top are shared between the orange and blue robots but are not directly visible to the green robot. Since $\mathcal{X}^{(i)}$ for the green robot does not include these two landmarks and there is no direct communication link between the orange and blue robots, the variable connectivity assumption in Assumption 1 does not hold for those two landmarks.

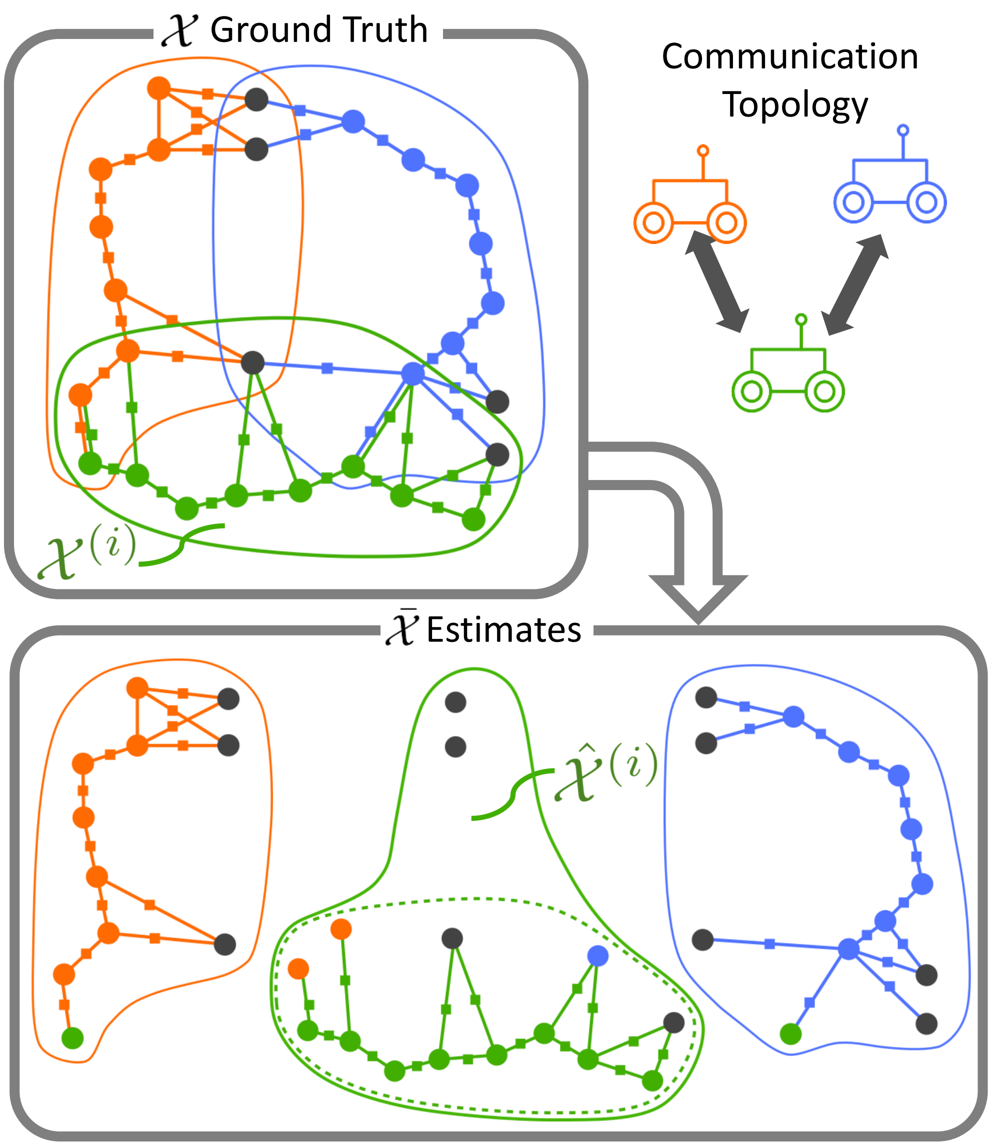
One way to address this type of scenario is to augment the local estimate set $\hat{\mathcal{X}}^{(i)}$ with additional variables. While minimizing the number of variables in $\hat{\mathcal{X}}^{(i)}$ is desirable, the only requirement on the membership of the set $\hat{\mathcal{X}}^{(i)}$ is that it satisfies $\mathcal{I}(\mathcal{X}^{(i)}) \subseteq \mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subseteq \mathcal{I}(\mathcal{X})$. In this example, the variable connectivity is satisfied by adding the two landmarks as part of $\hat{\mathcal{X}}^{(i)}$ for the green robot, as shown at the bottom of Fig. 6.4.

Another approach is to accept that Assumption 1 is not fully satisfied. The equality constraints for the multiple local estimates for a variable x_s are only enforced among the connected components of the induced subgraph $\mathcal{G}_s$. While (P1) and (P2) may not be strictly equivalent, the solutions for (P2) partially agree with the solution for (P1) and may still be acceptable depending on the application.

Lastly, one may consider augmenting $\hat{\mathcal{X}}^{(i)}$ for reasons other than ensuring the variable connectivity assumption holds. Consider an example where a distributed sensor network is tasked to track a common target. In this case, all the agents may include the local estimate of the target state in $\hat{\mathcal{X}}^{(i)}$, even if not all the agents make a direct observation of the target so that all agents have a copy of the target state estimate. Therefore choosing what variables to be included $\hat{\mathcal{X}}^{(i)}$ gives users additional choices of how to constrain the problem.

6.3 Mathematical Properties of LC-ADMM

This section details some theoretical guarantees on the convergence rate of LC-ADMM and some properties when applied to example systems. Before discussing our results, we note that it is well known that ADMM has (relatively slow) asymptotic convergence guarantee for a general class of convex problems without additional assumptions [43]. Since LC-ADMM is a type of ADMM, this result applies to LC-ADMM as well.



$\mathcal{X}$ Ground Truth
Communication Topology
$\mathcal{X}^{(i)}$
$\bar{\mathcal{X}}$ Estimates
$\hat{\mathcal{X}}^{(i)}$

Figure 6.4: Illustration of the definitions of $\mathcal{X}$, $\mathcal{X}^{(i)}$, $\hat{\mathcal{X}}^{(i)}$, and X. Colored circles represent the robot poses and the gray circles represent the landmarks. In this example, we have $\mathcal{I}(\mathcal{X}^{(i)}) = \mathcal{I}(\hat{\mathcal{X}}^{(i)})$ for i = "orange" while $\mathcal{I}(\mathcal{X}^{(i)}) \subset \mathcal{I}(\hat{\mathcal{X}}^{(i)})$ for i = "green.".

Theorem 3 (Asymptotic convergence of ADMM [43])**.** *Consider* (P2) *where the objective F is a closed, proper, convex function (not necessarily strongly convex) and the constraint $S^{(i)}$ is a convex set. Assume the unaugmented Lagrangian has a saddle point. Then we have*

- *Residual convergence:* $AX_k + BZ_k \longrightarrow 0$ *as* $k \longrightarrow \infty$,
- *Objective convergence:* $F(X_k) \longrightarrow F(X_*)$ *as* $k \longrightarrow \infty$,
- *Dual variable convergence:* $W_k \longrightarrow W_*$ *as* $k \longrightarrow \infty$.

Proof. The proof is given in [43]. □

In the next sections, we establish two stronger results under additional assumptions, as follows: (1) LC-ADMM converges at the rate of $o(1/k)$ when the problem has a unique solution (Theorem 4); and (2) LC-ADMM converges exponentially when the overall objective is strongly convex and has a Lipschitz continuous subgradient (Theorem 5). Theorem 5 can be viewed as a generalization of the previous result in the literature [37] to problems that are localized, non-differentiable, and have local equality constraints. Next, we apply Theorem 5 to a locally coupled dynamical system given in (D1) and show the relationship between observability and convergence. Finally, we discuss applying LC-ADMM to problems that involve inequality constraints and pose graph optimization problems.

First, we introduce some definitions needed for establishing the convergence results.

Definition 4 (Subdifferential)**.** *The subdifferential of F at X is a closed convex set defined as*

$$\partial F(X) \triangleq \bigcap_{X' \in domF} \{g \mid F(X') \geq F(x) + g^\top (X' - X)\}. \tag{6.11}$$

Definition 5 (Lipschitz Continuous Subdifferential)**.** *The subdifferential of a convex function F is* Lipschitz continuous *with parameter $L_F > 0$* iff *for all $X, X' \in domF$ and for all $g \in \partial F(X)$ and $g' \in \partial F(X')$, we have*

$$L_F \|X - X'\| \geq \|g - g'\|. \tag{6.12}$$

Definition 6 (Strong convexity)**.** *A convex function F is* strongly convex *with parameter $m_F > 0$ if for all $X, X' \in domF$ and $g \in \partial F(X)$ and $g' \in \partial F(X')$, we have*

$$m_F \|X - X'\|^2 \leq \langle X - X', g - g' \rangle. \tag{6.13}$$

Convergence Guarantees of LC-ADMM

To prove the convergence rates of LC-ADMM, we first rewrite the algorithm in a series of equivalent forms. Using the subgradient optimality condition, x-update (6.4) can be written as

$$\begin{aligned} &0 \in \partial_{X_{k+1}}(F(X_{k+1}) + \frac{\beta}{2}\|AX_{k+1} + BZ_k - C + \frac{1}{\beta}W_k\|^2) \\ \iff &-A^\top\left[W_k + \beta(AX_{k+1} + BZ_k - C)\right] \in \partial F(X_{k+1}). \end{aligned}$$

If we define $g_{k+1} \triangleq -A^\top\left[W_k + \beta(AX_{k+1} + BZ_k - C)\right]$, the ADMM iteration in (6.4)-(6.7) can be written as

$$g_{k+1} \in \partial F(X_{k+1}) \tag{6.14}$$

$$B^\top\left[W_k + \beta(AX_{k+1} + BZ_{k+1} - C)\right] = 0, \tag{6.15}$$

$$W_{k+1} - W_k - \beta(AX_{k+1} + BZ_{k+1} - C) = 0. \tag{6.16}$$

By left-multiplying (6.16) by $A^\top$ (resp. with $B^\top$) and subtracting from the definition of g_{k+1} (resp. from (6.15)), we have

$$g_{k+1} + A^\top\left[W_{k+1} + \beta B(Z_k - Z_{k+1})\right] = 0, \tag{6.17}$$

$$B^\top W_{k+1} = 0. \tag{6.18}$$

Since $W_k = [Y_k; Y'_k; V_k]$ and $B = [-I; -I; 0]$, (6.18) implies $Y_{k+1} = -Y'_{k+1}$. Recall $A \triangleq [A_1; A_2; A_3]$. If we define $\overline{M} \triangleq A_1 + A_2$ and $\underline{M} \triangleq A_1 - A_2$, then (6.17) can be written as

$$\begin{aligned} g_{k+1} + \underline{M}^\top Y_{k+1} + A_3^\top V_{k+1} \\ - \beta\overline{M}^\top(Z_k - Z_{k+1}) = 0. \end{aligned} \tag{6.19}$$

Equation (6.16) can be split into three blocks of equations by using $W_k = [Y_k; Y'_k; V_k]$ again. Adding and subtracting the first and second blocks of (6.16) from each other, we get

$$\frac{1}{2}\overline{M}X_{k+1} - Z_{k+1} = 0, \tag{6.20}$$

$$Y_{k+1} - Y_k - \frac{\beta}{2}\underline{M}X_{k+1} = 0, \tag{6.21}$$

$$V_{k+1} - V_k - \beta(A_3X_{k+1} - C_3) = 0. \tag{6.22}$$

Assume that there exist X_* and Z_* that are the unique solution[3] to (P2b). Since (P2b) is convex, closed, and proper, we also have that $(Y_k, V_k) \to (Y_*, V_*)$ as $k \to \infty$ and

[3]This is automatically satisfied if F is strongly convex as is the case for Theorem 5.

(Y_*, V_*) is a stationary point [43]; i.e., $\partial L_\beta(X_*, Z_*, Y_*, V_*) \ni 0$. Then (6.17), (6.20), (6.21), and (6.22) evaluated at this point is

$$g_* = -\underline{M}^\top Y_* + A_3^\top V_*, \quad g_* \in \partial F(X_*) \tag{6.23}$$

$$\frac{1}{2}\overline{M}X_* - Z_* = 0, \tag{6.24}$$

$$\frac{\beta}{2}\underline{M}X_* = 0, \tag{6.25}$$

$$A_3 X_* - C_3 = 0. \tag{6.26}$$

Subtracting (6.23)-(6.26) from (6.19)-(6.22), the ADMM iteration in terms of (X_k, Z_k, Y_k, V_k) can be written as:

$$\begin{aligned} &g_{k+1} - g_* + \underline{M}^\top (Y_{k+1} - Y_*) \\ &\quad + A_3^\top (V_{k+1} - V_*) - \beta\overline{M}^\top (Z_k - Z_{k+1}) = 0 \\ &\quad g_{k+1} \in \partial F(X_{k+1}), \quad g_* \in \partial F(X_*) \end{aligned} \tag{6.27}$$

$$\frac{1}{2}\overline{M}(X_{k+1} - X_*) - (Z_{k+1} - Z_*) = 0, \tag{6.28}$$

$$Y_{k+1} - Y_k - \frac{\beta}{2}\underline{M}(X_{k+1} - X_*) = 0. \tag{6.29}$$

$$V_{k+1} - V_k - \beta A_3 (X_{k+1} - X_*) = 0. \tag{6.30}$$

We use these equations to prove the convergence rate of LC-ADMM. For convenience, we define the subset of the variables as $U_k \triangleq [Z_k; Y_k; V_k]$. Two lemmas that establish the main theorem are the following.

Lemma 2 (Contractive sequence U_k)**.** *Suppose that F is convex, closed, and proper and that the sequence (X_k, Z_k, Y_k, V_k) is generated by the ADMM algorithm in* (6.3)*. If there exists a unique solution X_* and Z_* to* (P2b) *then $U_k = [Z_k; Y_k; V_k]$ satisfies*

$$\|U_k - U_{k+1}\|_G^2 \le \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2, \tag{6.31}$$

where the matrix $G \succ 0$ is given by

$$G \triangleq \begin{bmatrix} \beta I & 0 & 0 \\ 0 & \frac{1}{\beta} I & 0 \\ 0 & 0 & \frac{1}{2\beta} I \end{bmatrix}. \tag{6.32}$$

Additionally, if F is strongly convex, then we have

$$\begin{aligned} &m_F \|X_{k+1} - X_*\|^2 + \|U_k - U_{k+1}\|_G^2 \\ &\qquad \le \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2, \end{aligned} \tag{6.33}$$

where $m_F > 0$ was defined in Definition 6.

Proof of Lemma 2. The proof outline is similar to [37] with some notable differences: we generalize the result to the localized consensus case $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subset \mathcal{I}(\mathcal{X})$; we include local equality constraints; and we allow non-differentiable objective functions.

If F is convex, we have $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \geq 0$. Using (6.27)-(6.30), we may rewrite the left hand side as Substituting (6.27) to $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle$, we get

$$\begin{aligned}
&\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \\
&= \langle X_{k+1} - X_*, -\underline{M}^\top (Y_{k+1} - Y_*) - A_3^\top (V_{k+1} - V_*) \rangle \\
&\quad + \beta \langle X_{k+1} - X_*, \overline{M}^\top (Z_k - Z_{k+1}) \rangle. \\
&= \frac{2}{\beta} \langle Y_k - Y_{k+1}, Y_{k+1} - Y_* \rangle + \frac{1}{\beta} \langle V_k - V_{k+1}, V_{k+1} - V_* \rangle \\
&\quad + 2\beta \langle Z_{k+1} - Z_*, Z_k - Z_{k+1} \rangle \\
&= 2(U_{k+1} - U_*)^\top G (U_k - U_{k+1}) \\
&= \|U_k - U_*\|_G^2 - \|U_{k+1} - U_*\|_G^2 - \|U_k - U_{k+1}\|_G^2.
\end{aligned}$$

This proves (6.31). Because F is convex, we have $\langle X_{k+1} - X_*, g_{k+1} - g_* \rangle \geq 0$ and therefore (6.31) holds. When F is strongly convex, we automatically have X_* (and thus Z_*) are unique. Additionally, by Definition 6, we have

$$m_F \|X_{k+1} - X_*\|^2 \leq \langle X_{k+1} - X_*, g_{k+1} - g_* \rangle.$$

and therefore (6.33) holds. $\square$

Equation (6.31) in Lemma 2 shows that the sequence $\|U_k - U_*\|_G^2$ is monotonically non-increasing and converging because it is lower bounded by 0. This implies $\|U_{k+1} - U_k\|_G^2 \to 0$ as $k \to \infty$. Then, $U_k \to U_*$ follows the standard analysis of contraction methods.

We show the $o(1/k)$ convergence by making a summable, nonnegative, monotonic sequence argument. The proof for the following Lemma is readily available in other ADMM literature such as [36], [121], so it is omitted for brevity.

Lemma 3. *If a sequence $\{a_k\} \subseteq \mathbb{R}$ satisfies: (1) $a_k \geq 0$; (2) $\sum_{k=1}^{\infty} a_k < +\infty$; and (3) a_k is monotonically non-increasing, then we have $a_k = o(1/k)$.*

The next theorem establishes the $o(1/k)$ convergence for LC-ADMM.

Theorem 4 ($o(1/k)$ convergence of LC-ADMM)**.** *Assume that F is convex, closed, and proper, and that F has a unique solution. The sequence generated by* (6.3) *converges $(X_k, U_k) \longrightarrow (X_*, U_*)$ and its convergence rate is given by $\|U_k - U_{k+1}\|_G^2 = o(1/k)$.*

Proof. Using the first-order optimality conditions (6.19)-(6.22) for k and $k+1$, one can follow the same manipulations as in the proof for Lemma 2 to show that

$$\langle X_{k+1} - X_k, g_{k+1} - g_k \rangle = -2(U_{k+1} - U_k)^\top G (U_{k+1} - U_{k-1}).$$

Given F is convex, it follows that

$$\|U_k - U_{k-1}\|_G^2 - \|U_{k+1} - U_k\|_G^2 \geq \|U_{k+1} - U_{k-1}\|_G^2 \geq 0.$$

Therefore $\|U_{k+1} - U_k\|_G^2$ is monotonically non-increasing.
Next, we have that $\|U_{k+1} - U_k\|_G^2$ is summable by (6.31) of Lemma 2. Indeed, summing both the left- and right-hand sides of (6.31), we have

$$\sum_{k=0}^{\infty} \|U_k - U_{k+1}\|_G^2 \leq \|U_0 - U_*\|_G^2 < +\infty. \tag{6.34}$$

Because $\|U_k - U_{k+1}\|_G^2$ is non-negative, monotonically non-increasing, and summable, we have $\|U_k - U_{k+1}\|_G^2 = o(1/k)$ by Lemma 3. Because Z_* satisfies the equality constraint and F has a unique solution we also have $X_k \longrightarrow X_*$. □

Theorem 4 states that $U_k \longrightarrow U_*$ at the rate of $o(1/k)$ when the objective is convex and has a unique solution. Next, we show that LC-ADMM converges exponentially if we additionally assume that F is strongly convex and has Lipschitz continuous subdifferential. Before we present the result in Theorem 5, we show Lemma 4.

Lemma 4. *Suppose F is strongly convex and its subdifferential is Lipschitz continuous. For any $\mu > 1$, define a constant c such that*

$$c \triangleq \min \left\{ \frac{(\mu - 1)\tilde{\sigma}_{\min}^2(\underline{M})}{\mu \sigma_{\max}^2(\overline{M})}, \frac{m_F}{\frac{\beta}{4}\sigma_{\max}^2(\overline{M}) + \frac{\mu}{\beta} L_F^2 \tilde{\sigma}_{\min}^{-2}(\underline{M})} \right\} > 0 \tag{6.35}$$

where $L_F > 0$ and $m_F > 0$ are defined in Definitions 5 and 6, respectively. Then, we have

$$c\|U_{k+1} - U_*\|_G^2 \leq m_F \|X_{k+1} - X_*\|^2 + \|U_k - U_{k+1}\|_G^2. \tag{6.36}$$

Proof of Lemma 4. The proof is similar to [37] with a few notable differences: we generalize the result to the localized consensus case $\mathcal{I}(\hat{\mathcal{X}}^{(i)}) \subset \mathcal{I}(\mathcal{X})$; we include local equality constraints; and we allow non-differentiable objective functions.

We start by considering the following inequality which holds true for $\forall \mu > 0$ [37]:

$$\|a_1 + a_2\|^2 + (\mu - 1)\|a_1\|^2 \geq (1 - 1/\mu)\|a_2\|^2. \tag{6.37}$$

We select $\mu > 1$ so that all the terms are positive and it results in a meaningful bound. If we let $a_1 = g_{k+1} - g_*$, $a_2 = \underline{M}^\top (Y_{k+1} - Y_*) + A_3^\top (V_{k+1} - V_*)$, we have $a_1 + a_2 = \beta \overline{M}^\top (Z_k - Z_{k+1})$ by (6.27). The left-hand side of (6.37) is upper bounded by

$$\begin{aligned} &\|\beta \overline{M}(Z_{k+1} - Z_k)\|^2 + (\mu - 1)\|g_{k+1} - g_*\|^2 \\ &\leq \beta^2 \sigma_{\max}^2(\overline{M})\|Z_{k+1} - Z_k\|^2 + (\mu - 1)L_F^2\|X_{k+1} - X_*\|^2, \end{aligned} \tag{6.38}$$

where we used (6.12) from Lipschitz continuous subdifferential assumption. Next, we have that $Y_k \in \mathrm{Im}(\underline{M})$ and $V_k \in \mathrm{Im}(A_3)$, $\forall k > 0$ by (6.22) and (6.21). Therefore the right-hand side of (6.37) can be lower bound by

$$\begin{aligned} &\|\underline{M}^\top (Y_{k+1} - Y_*) + A_3^\top (V_{k+1} - V_*)\|^2 \\ &= \left\| \begin{bmatrix} \underline{M} \\ A_3 \end{bmatrix}^\top \begin{bmatrix} Y_{k+1} - Y_* \\ V_{k+1} - V_* \end{bmatrix} \right\|^2 \\ &\geq \tilde{\sigma}_{\min}^2(M) \left\| \begin{bmatrix} Y_{k+1} - Y_* \\ V_{k+1} - V_* \end{bmatrix} \right\|^2 \\ &= \tilde{\sigma}_{\min}^2(M)(\|Y_{k+1} - Y_*\|^2 + \|V_{k+1} - V_*\|^2) \end{aligned} \tag{6.39}$$

where $M \triangleq [\underline{M}; A_3]^\top$ and $\tilde{\sigma}_{\min}(M)$ is the minimum non-zero singular value of M. Therefore we have

$$\begin{aligned} &\beta^2 \sigma_{\max}^2(\overline{M})\|Z_{k+1} - Z_k\|^2 + (\mu - 1)L_F^2\|X_{k+1} - X_*\|^2 \\ &\geq \left(1 - \frac{1}{\mu}\right) \tilde{\sigma}_{\min}^2(\underline{M})(\|Y_{k+1} - Y_*\|^2 + \|V_{k+1} - V_*\|^2). \end{aligned}$$

This is further rearranged to

$$\begin{aligned} &\beta C_1 \|Z_{k+1} - Z_k\|^2 + \frac{C_2}{\beta}\|X_{k+1} - X_*\|^2 \\ &\geq \frac{1}{\beta}\|Y_{k+1} - Y_*\|^2 + \frac{1}{\beta}\|V_{k+1} - V_*\|^2 \end{aligned}$$

where

$$C_1 = \frac{\mu \sigma^2_{\max}(\overline{M})}{(\mu - 1)\tilde{\sigma}^2_{\min}(\underline{M})}, \quad C_2 = \frac{\mu L_F^2}{\tilde{\sigma}^2_{\min}(\underline{M})}.$$

Using $\|Z_{k+1} - Z_*\| \leq \frac{1}{2}\sigma_{\max}(\overline{M})\|X_{k+1} - X_*\|$, we get

$$\begin{aligned}
&\beta C_1 \|Z_{k+1} - Z_k\|^2 + \left(\frac{C_2}{\beta} + \frac{\beta}{4}\sigma^2_{\max}(\overline{M})\right)\|X_{k+1} - X_*\|^2 \\
&\geq \beta\|Z_{k+1} - Z_*\|^2 + \frac{1}{\beta}\|Y_{k+1} - Y_*\|^2 + \frac{1}{\beta}\|V_{k+1} - V_*\|^2.
\end{aligned}$$

Then, $c > 0$ as defined in (6.35) satisfies

$$\begin{aligned}
&\beta\|Z_{k+1} - Z_k\|^2 + m_F\|X_{k+1} - X_*\|^2 \\
&\geq c\beta\|Z_{k+1} - Z_*\|^2 + \frac{c}{\beta}\|Y_{k+1} - Y_*\|^2 + \frac{c}{\beta}\|V_{k+1} - V_*\|^2.
\end{aligned}$$

This further satisfies (6.36). □

The proof of Lemma 4 uses the optimality of the ADMM update equations and the additional assumptions introduced. By using Lemmas 2 and 4, it is straightforward to prove the exponential stability of U_*.

Theorem 5 (Exponential convergence of LC-ADMM)**.** *Consider the LC-ADMM that solves* (P2a)*. Assume F is convex, closed, and proper. If F is strongly convex and its subdifferential is Lipschitz continuous, $U_k \longrightarrow U_*$ and $X_k \longrightarrow X_*$ exponentially. Additionally, if Assumption 1 is satisfied, the obtained solutions $\hat{\mathcal{X}}_*^{(i)}$ for each $i \in \mathcal{A}$ are the subset of $\mathcal{X}_*$ which is the solution to* (P1).

Proof. Define $c > 0$ as (6.35) for any $\mu > 1$. After combining (6.33) in Lemma 2 and (6.36) in Lemma 4, we can write

$$\|U_{k+1} - U_*\|_G^2 \leq c_2\|U_k - U_*\|_G^2, \tag{6.40}$$

where $c_2 \triangleq \frac{1}{1+c}$. Since $0 < c_2 < 1$, U_k converges exponentially to U_*. Next, we observe from (6.33) that

$$\|X_{k+1} - X_*\|^2 \leq \frac{1}{m_F}\|U_k - U_*\|_G^2. \tag{6.41}$$

Since $\|U_k - U_*\|_G^2$ exponentially converges to zero, X_{k+1} exponentially converges to X_*. □

Theorem 5 is a generalization of the convergence rate of DC-ADMM in [37]. The result is extended to problems that may involve localized consensus, local affine equality constraints, and non-differentiable objective functions. DC-ADMM is a special case of LC-ADMM where $\hat{\mathcal{X}}^{(i)} = \{x_s^{(i)} \mid s \in \mathcal{I}(\mathcal{X})\}$. The exponential convergence of DC-ADMM assumes that all the agents in the network are connected. This connectivity assumption in DC-ADMM is replaced with the variable connectivity assumption (Assumption 1) in LC-ADMM which required that variable connectivity is satisfied per variable.

Finally, we remark on some properties of LC-ADMM. First, LC-ADMM converges to the optimal solution even when considering other convex loss functions such as the Huber norm. This is a contrast to other distributed algorithms such as [18] whose derivation explicitly assumes specific noise distributions. Moreover, in LC-ADMM, all the agents can update in parallel and the agents do not need global network topology information such as graph coloring. This is an especially desirable property for ad hoc networks. LC-ADMM preserves *privacy* in the sense that each agent only needs to share the coupled variables with the neighbors who need that information. Agents only need to reveal the augmented factor information to their neighbors.

Large-Scale Networked Dynamical System

We revisit the estimation problem of large-scale networked dynamical systems (D1) and consider the convergence guarantees that are further specialized for these systems. To facilitate the discussion pertaining to observability, let us assume that the noise distribution is given by independent, zero-mean Gaussian distributions and that a_t and c_t are twice differentiable and their gradients are Lipschitz continuous. Let $r_\tau \triangleq [r_{1,\tau}; \ldots, r_{|\mathcal{A}|,\tau}]$ be the state vector of the network at τ and $\mathcal{X} = [r_0; \ldots; r_t]$ be the trajectory of the network state up to time t. The linearized observability of (D1) at some reference trajectory X can be analyzed using the following linearized observability matrix:

$$O(\mathcal{X}) = \begin{bmatrix} \partial c_0(r_0) \\ \partial c_1(r_1)\partial a_1(r_0) \\ \vdots \\ \partial c_t(r_t)\partial a_t(r_{t-1}) \cdots \partial a_1(r_0) \end{bmatrix}. \tag{6.42}$$

For this example, we have the following corollary.

Corollary 1. *Suppose the dynamics model a_i and the measurement model c_i of* (D1) *are twice differentiable and their gradients are Lipschitz continuous. Also, suppose the noise models are given by $w_{i,t} \sim \mathcal{N}(0, W_i^{-1})$ and $v_i \sim \mathcal{N}(0, V_i^{-1})$ where $W_i, V_i \succ 0$. Suppose an optimal solution to* (P2a) *is given by $X_* = [r_{0*}; \dots; r_{T*}]$ and the matrix $O(\mathcal{X}*)$ has a full column rank for at $\mathcal{X}_*$. Then the objective $F(X)$ is locally strongly convex $\forall X \in \mathcal{B}_\rho(X_*)$ with some $\rho > 0$. Moreover, any trajectories starting within $\mathcal{B}_\rho(X_*)$ exponentially converge to X_*.*

Proof of Corollary 1. The objective function in (P1) is given by

$$\begin{aligned} F(X) =& \sum_{t=1}^{T} \|r_t - a_t(r_{t-1})\|_{W_t}^2 + \sum_{t=0}^{T} \|y_t - c_t(r_t)\|_{V_t}^2 \\ =& R^\top(X)\Sigma R(X), \end{aligned}$$

where $\Sigma = \operatorname{diag}(W_1, \dots, W_T, V_0, \dots, V_T)$ and $R(X) = [r_1 - a_1(r_0); \dots; r_T - a_T(r_{T-1}), y_0 - c_0(r_0), \dots, y_T - c_T(r_T)]$. Since Σ is a positive definite matrix, the Hessian of $F(X)$ is strictly positive definite at each $X \in \mathcal{B}_\rho(X_*)$ if and only if $\partial R(X)$ has a full column rank at X.

The residual Jacobian can be written as $\partial R(X)$ equals to

$$-\begin{bmatrix} \partial a_1(r_0) & I & 0 & \cdots & 0 \\ 0 & \partial a_2(r_1) & I & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \partial a_T(r_{T-1}) & I \\ \partial c_0(r_0) & 0 & \cdots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & & 0 & \partial c_T(r_T) \end{bmatrix}.$$

Applying block Gauss eliminations, one can write the first column as

$$[0; \dots; 0; \partial c_0; \partial c_1 \partial a_1; \dots; \partial c_T \partial a_T \cdots \partial a_1].$$

Since X_* is a feasible solution to (P2a), there exists a solution $\mathcal{X}_*$ for (P1) such that $x_{s,*}^{(i)} = x_{s,*}$ for $\forall s \in \mathcal{I}(\mathcal{X})$, $\forall i \in \mathcal{A}$ by Lemma 1. Since the system is observable at $\mathcal{X}_*$, i.e., $O(\mathcal{X}_*)$, the first column of $\partial R(X_*)$ after the Gauss elimination has a full column rank. One can see that other block columns of $\partial R(X_*)$ are also linearly independent by noting their identity block elements. Therefore $\partial R(X_*)$ has a full column rank and the Hessian of $F(X_*)$ is positive definite. Since a_t and c_t are twice differentiable, there exists a local neighborhood $\mathcal{B}_\rho(X_*)$ with sufficiently small $\rho > 0$ in which $F(X)$ is strongly convex for $\forall X \in \mathcal{B}_\rho(X_*)$ with the local coefficient of strong convexity $m_F > 0$ is given by:

$$m_F = \min_{X \in \mathcal{B}_\rho(X_*)} \sigma_{\min}(HF(X)), \tag{6.43}$$

where $\sigma_{\min}(\cdot)$ is the smallest eigenvalue of the matrix. Therefore Assumption 6 is satisfied locally.

The objective $F(X)$ is also Lipschitz continuous subgradient, given that $a_t(r_{t-1})$ and $c_t(r_t)$ are Lipschitz continuous. This satisfies Assumption 5. Finally, we have that LC-ADMM is locally exponentially stable at X^* by Theorem 5. □

An important implication of Corollary 1 is that LC-ADMM only requires that the network as a whole is observable. Each agent, when considered in isolation from the other agents, need not be observable for LC-ADMM to work. We validate this point later in numerical simulation experiments. Additionally, if (D1) is a linear, time-varying (LTV) system with Gaussian noise, the exponential convergence is guaranteed globally.

Corollary 2 (Convergence for LTV systems)**.** *In addition to the assumptions in Corollary 1, further assume that the dynamical system in* (D1) *is LTV. Then, LC-ADMM is globally exponentially convergent to the unique solution. Moreover, each sub-problem in* (6.8) *simplifies to an unconstrained, quadratic program, for which the analytical solution is given by the matrix multiplication of form* $\hat{X}^{(i)}_{k+1} = (H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}b_k$. *The matrix* $(H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}$ *is only computed once at the beginning of LC-ADMM on agent i, and* b_k *is an affine function of* Z_k *and* W_k.

Proof. The proof is similar to that of Corollary 1. The observability implies that $F(X)$ is strongly convexity and $F(X)$ is Lipschitz continuous subgradient. By Theorem 5, we have that LC-ADMM globally converges exponentially to the global minimum. □

Corollary 2 shows that each computation of LC-ADMM is quite inexpensive for linear systems once $(H^{(i)\top}H^{(i)})^{-1}H^{(i)\top}$ is computed for each agent i.

LC-ADMM for Problems with Weaker Assumptions

We consider the application of LC-ADMM with weaker assumptions. First, consider a case where the original problem (P2) additionally involves local inequality constraints $\hat{\mathcal{X}}^{(i)} \in S^{(i)}$, where $S^{(i)}$ is a convex set. In this case, one can modify the objective function (P2) to also include the indicator function

$$f_i(\hat{\mathcal{X}}^{(i)}) \leftarrow f_i(\hat{\mathcal{X}}^{(i)}) + \mathbb{I}_{S^{(i)}}(\hat{\mathcal{X}}^{(i)}).$$

Then, the LC-ADMM iterations in (6.8) and (6.9) can be written as

$$\begin{aligned}\hat{\mathcal{X}}^{(i)}_{k+1} = &\operatorname*{arg\,min}_{\hat{\mathcal{X}}^{(i)} \in S^{(i)}} f_i(\hat{\mathcal{X}}^{(i)}) \\ &+ \sum_{j \in \mathcal{N}^i} \sum_{s \in \mathcal{I}^{ij}} \frac{\beta}{2} \| x^{(i)}_s - \bar{x}^{(ij)}_{s,k+1} + \frac{1}{\beta} w^{(ij,i)}_{s,k} \|^2,\end{aligned} \tag{6.44}$$

$$w^{(ij,i)}_{s,k+1} = w^{(ij,i)}_{s,k} + \beta(x^{(i)}_{s,k+1} - \bar{x}^{(ij)}_{s,k+1}). \tag{6.45}$$

Even though the additional inequality constraints can model a broader class of problems, the exponential convergence proof for Theorem 5 would not work anymore, so it was not included in the problem definition (P2a).

Next, we consider non-convex optimization problems such as PGO for which Theorems 4 and 5 do not hold. While prior works in the literature showed that ADMM converges for a certain class of non-convex problems [36], (P2) does not satisfy all the assumptions of [36]. Therefore, it remains a future work to determine whether LC-ADMM converges for a certain class of non-convex problems. Instead, we demonstrate that LC-ADMM can still be derived for non-convex problems and empirically show that LC-ADMM converges in some simulation examples.

We consider the problems involving manifold-constrained variables such as PGO. Suppose a pose state in d-dimensional space is given by $T = [R, t] \in \mathrm{SE}(d)$ where $R \in \mathrm{SO}(d)$ is the rotation component and $t \in \mathbb{R}^d$ is the translation component of T. If $T_p, T_q \in \mathcal{X}$ are the variables involving some factor $\phi \in \mathcal{F}$ with noisy relative pose observation $\tilde{T}_{pq}$, we can write the local consensus reformulation of the PGO problem as

$$\begin{aligned}\min_{X,Z} \quad & \sum_{i \in \mathcal{A}} \sum_{\phi \in \mathcal{F}_i} L(T^{(i)}_p, T^{(i)}_q, \tilde{T}_{pq}) \\ \text{subject to} \quad & \left.\begin{aligned} T^{(i)}_s = T^{(ij)}_s, \\ T^{(j)}_s = T^{(ij)}_s, \end{aligned}\right. \quad \forall s \in \mathcal{I}(\hat{\mathcal{X}}^{ij}), (i,j) \in \mathcal{E}, \\ & T^{(i)}_s \in SE(d), \quad \forall s \in \mathcal{I}(\mathcal{X}^{(i)}), i \in \mathcal{A}\end{aligned} \tag{6.46}$$

where $L(T_p, T_q, \tilde{T}_{pq})$ represents some loss function on a relative pose. Like we did in (6.44), one can simply incorporate the manifold constraint in the local FGO update equation. The manifold constraint $T_s \in \mathrm{SE}(d)$ introduces the non-convex constraints.

In practice, there are some heuristic strategies to mitigate the risks of being stuck at a local minimum. First, the local FGO update step of LC-ADMM does not preclude the use of optimization techniques that guarantee convergence to the global minimum for PGO (i.e., SE-Sync [2]). Therefore, loop-closure constraints within the agent itself are remedied. Second, we develop iDFGO, an incremental version of LC-ADMM, in Section 6.4 where agents build the map incrementally. This incremental approach, combined with the ability to resolve the loop closures with self, makes the algorithm more robust against local minima. We also demonstrate empirically in simulations that LC-ADMM and iDFGO converge quickly to the optimal solution.

6.4 Incremental DFGO

In the previous sections, we applied LC-ADMM to the multi-agent MAP problem (6.1) over some fixed time horizon. If we take the system in (D1) as an example, LC-ADMM estimates $\mathcal{X}_t = \{r_{i,\tau} \,|\, i \in \mathcal{A}, \tau \in \mathcal{T}(t)\}$ where $\mathcal{T}(t) \triangleq [0, t]$ is the time horizon at t. In real-time applications, however, the distributed optimization problem needs to be recomputed at each time step t as new factors are added to the graph. As the length of the time horizon increases, the size of the problem in LC-ADMM also grows.

To address this challenge, we further extend LC-ADMM and derive the Incremental DFGO (iDFGO) algorithm that is scalable with respect to the increasing time horizon. The main idea is to combine a single-agent incremental probabilistic inference algorithm (i.e., iSAM2 [23]) with the local FGO update step of LC-ADMM such that only a recent subset of the factor graph needs to be recomputed. It turns out that simply applying an incremental algorithm to solve the local FGO problem does not result in an algorithm that is scalable with respect to time. To derive a time-scalable algorithm, we exploit the temporal sparsity of the real-time factor graph and the convergence of the augmented factors of LC-ADMM. The computation and communication bandwidth of the overall algorithm is both scalable in size of the network and in time.

For this section, we also additionally assume Gaussian distributions as they do in the derivation of iSAM2 [18], [23]. For problems with non-Gaussian distributions, one may consider using the moving time horizon approach where the width of the time horizon is kept constant.

Fluid Augmented Factor Update in Bayes Tree

One of the most prominent works for incremental probabilistic inference for a single robot is iSAM2 [23] which reformulates factor graphs as Bayes trees. The primary benefit of reformulating the original factor graphs as Bayes trees is that it permits an incremental update of the FGO solution. In the Bayes tree, the joint probability of the original factor graph is refactored as the product of conditional probabilities $\prod p(\delta_F|\delta_S)$ that is modeled as a chordal tree [23]. Each node is a conditional probability $p(\delta_F|\delta_S)$ for a clique of the original factor graph and one can obtain this set of conditional probabilities by eliminating one variable at a time. Suppose the factor f_{joint} depends on δ_F and δ_S where δ_S is the separator variables and δ_F is the frontal variables.

$$f_{\text{joint}}(\delta_F, \delta_S) \propto \exp\left(-\frac{1}{2}\|A_F\delta_F + A_S\delta_S - b\|^2\right) \tag{6.47}$$

The separator variables are those shared with its parent node and they are eliminated from the node [23]. The probability distribution can be written as $f_{\text{joint}}(\delta_F, \delta_S) = P(\delta_F|\delta_S)f_{new}(\delta_S)$ where

$$P(\delta_F|\delta_S) \propto \exp\left(-\frac{1}{2}\|\delta_F + R\delta_S - d\|^2\right), \tag{6.48}$$

$$f_{\text{new}}(\delta_S) \propto \exp\left(-\frac{1}{2}\|A'\delta_S - b'\|^2\right). \tag{6.49}$$

Here we have $R = A_F^{\dagger}A_S$, $d = A_F^{\dagger}b$, $A' = A_S - A_F R$, and $b' = b - A_F d$ and $A_F^{\dagger} = (A_F^{\top}A_F)^{-1}A_F^{\top}$ is the pseudo-inverse of A_F. Finally, after the elimination, we have parent factor $f_{\text{new}}(\delta_S)$ and child factor $P(\delta_F|\delta_S)$. This elimination process is repeatedly applied to the top portion of the Bayes tree until only one parent factor remains as the root of the tree.

In iSAM2, the A' and b' terms for the parent are recomputed only when the linearization point changes sufficiently. Therefore, so long as the linearization point does not change too much, only the root of the tree needs to be modified in an incremental fashion when new factors are added. One of the assumptions that iSAM2 has is that the measurement values of the nonlinear factors (i.e., b term) are fixed in their lifetime. While this assumption holds true in the nominal single-agent FGO scenario, in which each measurement is observed once and does not change, we need to revisit the assumption for iDFGO.

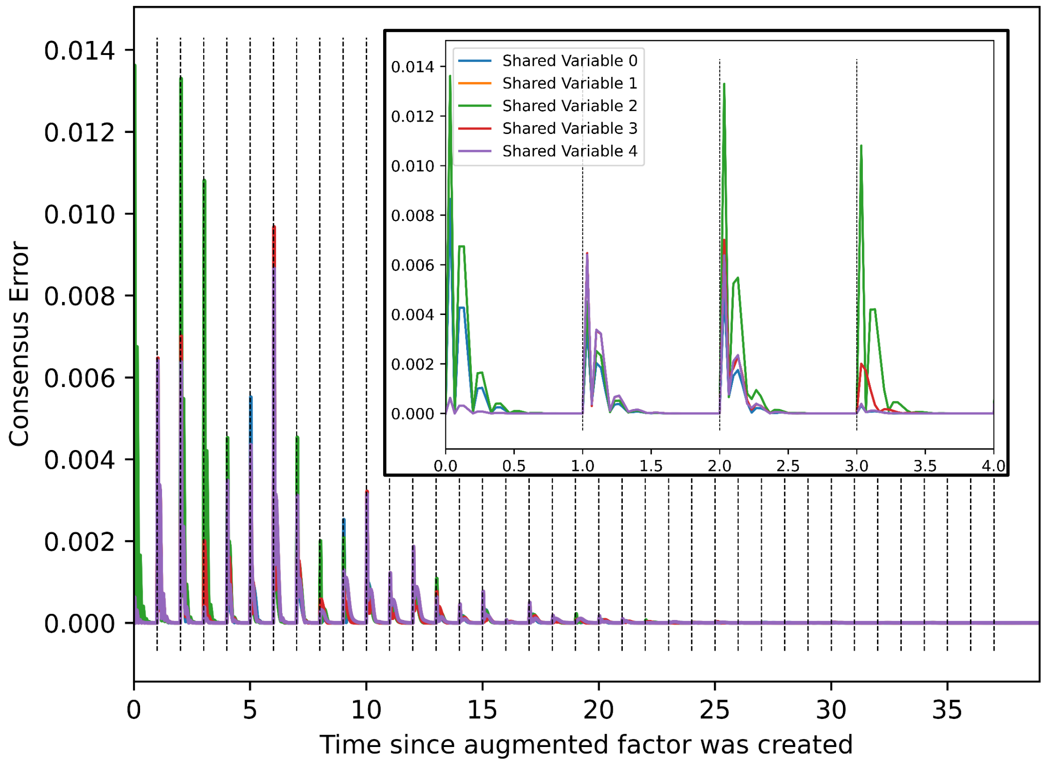


Figure 6.5: Consensus error $\|x_s^{(i)} - x_s^{(j)}\|$ for 5 sample variables converges over time in the real-time electrical grid example. New factors are added per time step (indicated by black dotted lines) and $K = 30$ LC-ADMM iterations are applied in batch at each time step. The consensus errors converge to zero over time.

Since iSAM2 scales well with respect to time, we extend it to distributed estimation problems. One naive attempt would be to simply solve the local FGO update step of LC-ADMM using iSAM2; however, this does not immediately result in a scalable algorithm. Because each augmented factor modifies its pseudo measurement at each LC-ADMM iteration, the assumption that the measurement term is fixed does not hold anymore. To compute the correct optimal MAP solution using the Bayes tree, every time b changes in a clique, the value of b' term in its parent node must be modified. This recursively affects the ancestors of the node, all the way up to the root of the tree. If there is an augmented factor near the bottom of the tree and it modifies its pseudo measurement, a large number of variables are affected in each update. Therefore, simply applying iSAM2 to the update step of LC-ADMM does not result in an algorithm that scales well with time.

We exploit the temporal structure of the real-time problem to address this issue. To this end, we make an important observation on the convergence of consensus errors. Recall that in LC-ADMM, the role of the augmented factors is to enforce

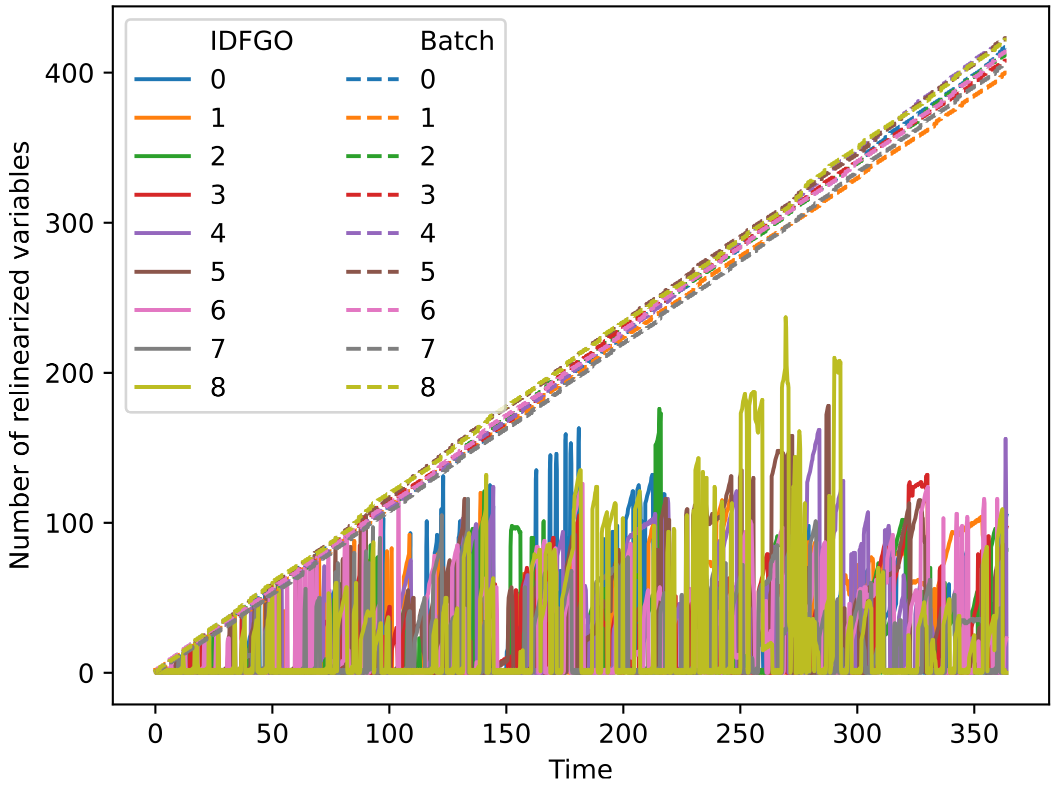


Figure 6.6: Number of variables updated at each time step of iDFGO for the multi-agent SLAM example. As the size of the factor graph grows, the computational complexity of iDFGO remains bounded.

the consistency between a pair of variable estimates $x_s^{(i)}$ and $x_s^{(j)}$ for a variable x_s and agents i and j. In real-time applications, the effects of newer factors on older variables $x_s^{(i)}$ and $x_s^{(j)}$ diminish in comparison to those of augmented factors and the two estimates converge over time. Figure 6.5 shows some sample trajectories of the consensus error $\|x_s^{(i)} - x_s^{(j)}\|$ as a function of time from an electrical power grid example. At each time step, new factors are added to the graph and $K = 30$ LC-ADMM iterations are applied in a batch. Even though new factors are added to the graph, the error between each pair of estimates converges to zero around $t = 30$ after the augmented factor was created in this example.

We exploit this convergence of consensus error and propose the *fluid augmented factor update* where each augmented factor pair terminates its update once the consensus is achieved to some threshold. The fluid augmented factor update is summarized in Algorithm 6. If older augmented factors that reached consensus do not modify their pseudo measurements, the Bayes tree update only needs to recompute the more recent subset of the augmented factors. Figure 6.6 shows the number of variables recomputed in a multi-agent PGO example described later in Section 6.5. Even though the number of factors in the graph grows over time, the number of affected variables remains bounded in iDFGO. In addition to reducing the computational effort, fluid augmented factor update also improves the communication bandwidth. While the pair of augmented factors are converged, the neighboring robots do not need to continuously exchange the pseudo measurements for those factors. Therefore fluid augmented factor update improves computation and communication simultaneously.

For the augmented factors that have not converged yet, the change in the value of pseudo measurements must be reflected recursively up to the root of the Bayes tree. We implement this similarly to the *fluid relinearization* step of iSAM2. We mark all the affected cliques and their ancestors in the tree and redo their variable elimination. Compared to the single-agent iSAM2 case, the number of affected variables during redoing the top of the Bayes tree is typically larger because some augmented factors from the past have not yet converged. However, because these factors are involved in the cliques near the top of the Bayes tree, the number of variables involved in the update step is still reduced compared to optimizing the whole factor graph in a batch.

Algorithm 6: Fluid augmented factor update for agent i, neighbor j, variable x_s, time t, and iteration k

```
Result: Updated $\{w^{(ij,i)}_{s,k,t}\}$, $\{\bar{x}^{(ij)}_{s,k,t}\}$ and marked variable set $M'$
for each $s \in \mathcal{I}^{ij}$ and $j \in \mathcal{N}^i$ do
    if not converged then
        $\bar{x}^{(ij)}_{s,k,t} = \frac{1}{2}\left(x^{(i)}_{s,k,t} + x^{(j)}_{s,k,t}\right)$ ;
        $w^{(ij,i)}_{s,k,t} = w^{(ij,i)}_{s,k-1,t} + \frac{1}{\beta}\left(x^{(i)}_{s,k,t} - \bar{x}^{(ij)}_{s,k,t}\right)$ ;
        Update the pseudo measurement to $x^{(i)}_{s,k,t} - \frac{1}{\beta} w^{(ij,i)}_{s,k,t}$;
        Add variable $x^{(i)}_s$ to the set $M'$;
    end
    else
        $\bar{x}^{(ij)}_{s,k,t} = \bar{x}^{(ij)}_{s,k-1,t}$;
        $w^{(ij,i)}_{s,k,t} = w^{(ij,i)}_{s,k-1,t}$;
    end
end
```

Variable Ordering

An important aspect to consider in developing the Bayes tree is the order of variable elimination. The iSAM2 algorithm uses the constrained column approximate minimum degree (constrained COLAMD) algorithm to compute the order of elimination such that it reduces the sparsity fill-ins in the presence of loop closures. This heuristic algorithm works well in improving computational efficiency for single-agent mapping problems in the presence of loop closures. However, it actually leads to a large number of recomputed variables when used in conjunction with the augmented factor updates of iDFGO. Instead, we choose a temporal ordering for variable elimination, where the newer variables are placed closer to the root of the tree. This approach preserves the temporal structure in the Bayes tree and so, the fluid augmented factor update only affects the top part of the tree. Fig. 6.7 shows a set of example Bayes trees obtained by this variable ordering scheme in a multi-agent PGO example where the highlighted nodes are the cliques that are affected during the particular update.

iDFGO Algorithm

The overall algorithm for iDFGO is summarized in Alg. 7. The local FGO update part of iDFGO can be seen as an extension of iSAM2, and some of its subroutines are reused. First, at each time step, new factors and variables are added to the local factors. Second, the fluid augmented factor update step modifies some of the pseudo measurement of augmented factors as described in Alg. 6. All the variables related to the updated augmented factors are marked and stored in a set M in this step. Next, if there are new augmented factors at any k, they are also added to the factor graph and the involved variables are also added to the set M. The fluid relinearization step of iSAM2 (Alg. 5 in [23]) may relinearize some of the nonlinear factors and returns its own set of marked keys. These marked keys are also added to the set M. Afterward, we redo the top of the Bayes tree in a similar way as Alg. 6 in [23] with the exception of two modifications. The first modification is that we use the modified marked key M which contains the variables affected by both fluid relinearization and fluid augmented factor update. We use this set to mark all the affected cliques in the Bayes tree and their ancestors. The second modification is the variable elimination order. After redoing the Bayes tree, neighboring agents locally exchange the new estimate. The iDFGO iterates this for K steps per each time step t. To promote an efficient implementation, the iDFGO algorithm is implemented in C++ using iSAM2 in the GTSAM library. We refer readers to [23] for details of the

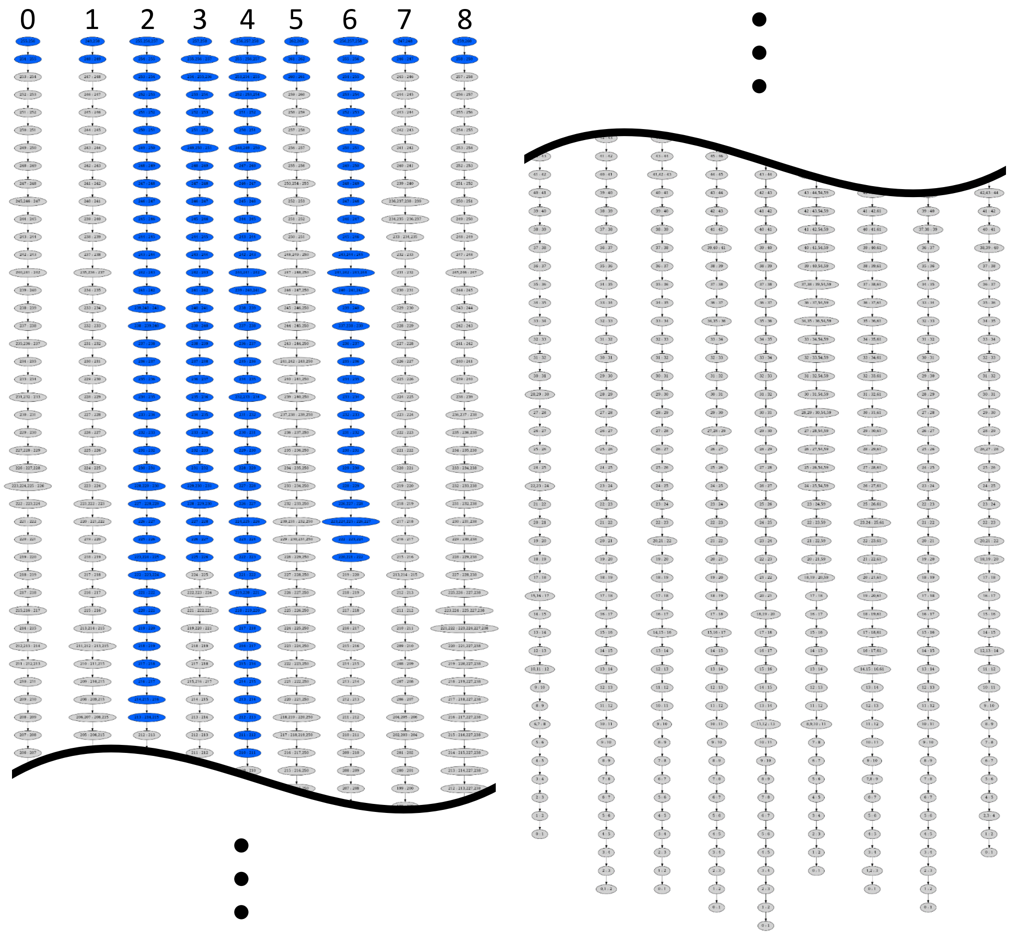


Figure 6.7: The top of the Bayes trees (left) and the bottom (right) for the 9 agents running iDFGO for real-time PGO example. The blue nodes show the variables whose nodes are modified by the incremental update.

iSAM2 algorithm.

Scalability with Time

We discuss the complexity of the iDFGO algorithm with respect to the size of the trajectory. First, for an estimation of a dynamical system (e.g., system considered in (D1)), only the recent subset of nodes in the Bayes tree is updated instead of solving the optimization problem over the whole trajectory in a batch (Fig. 6.7). The number of variables involved in the update has the complexity of $O(1)$ with respect to time.

In the presence of loop closures, the computational effort of the iDFGO algorithm adapts as necessary. Suppose a newly added loop closure constraint affects variables from τ' time steps ago in the past. The variables directly involved in the loop closure (up to $t-\tau'$) as well as variables in its proximity are affected by the loop closure. If the

Algorithm 7: Incremental DFGO

```
Result: $\hat{\mathcal{X}}^{(i)}_{t,k}$, $i \in \mathcal{A}$
Each agent $i \in \mathcal{A}$ locally initializes $\hat{\mathcal{X}}^{(i)}_{0,K} = \emptyset$ and $t = 0$;
while true do
    Add any new factors;
    Initialize any new variables and add them to $\hat{\mathcal{X}}^{(i)}_{t,0}$;
    for k=[0,K-1] do
        Fluid augmented factor update (Alg. 6) yields marked variables $M$;
        Create any new augmented factors and add their variables to set $M$;
        Apply fluid relinearization (Alg. 5 in [23]). The marked variables from
          Alg 5 are also added to $M$;
        Taking $M$ as input, redo the top of Bayes tree (Alg. 6 of [23]) that is
          modified to eliminate variables in a temporal ordering;
        Solve for delta $\Delta$ (Alg. 7 in [23]);
        The current estimate is given by $\hat{\mathcal{X}}^{(i)}_{t,k+1} = \hat{\mathcal{X}}^{(i)}_{t,k} \oplus \Delta$;
        Exchange shared variable estimates with neighbors;
    end
    $t \rightarrow t + 1$;
end
```

change in the estimate is sufficiently large, the variables may have a large consensus error compared to the neighbors and may be required to apply the augmented factor update. The estimates of the distributed agents will converge again sometime after the loop closure is added, but the augmented factor continues to update until the consensus error converges sufficiently. In terms of algorithm complexity, suppose there is an upper bound $T' > 0$ to the time lag on loop closure (e.g., $\tau' \leq T'$ for all loop closure). Then, the most general bound on the number of variables involved in each iDFGO update scales like $o(T'^3)$ and it is constant complexity with respect to the length of the whole trajectory.

6.5 Numerical Evaluation

We evaluate various aspects of LC-ADMM and iDFGO on multiple networked-system models using numerical simulations. First, we validate scalability, optimality, and convergence. Second, we demonstrate the application of LC-ADMM to a distributed outlier rejection problem. Finally, we compare the performance of LC-ADMM and iDFGO against other PGO algorithms. We use the Levenberg-Marquardt algorithm to solve each agent's individual FGO update step of LC-ADMM, and we implement this using the GTSAM library in C++. The iDFGO

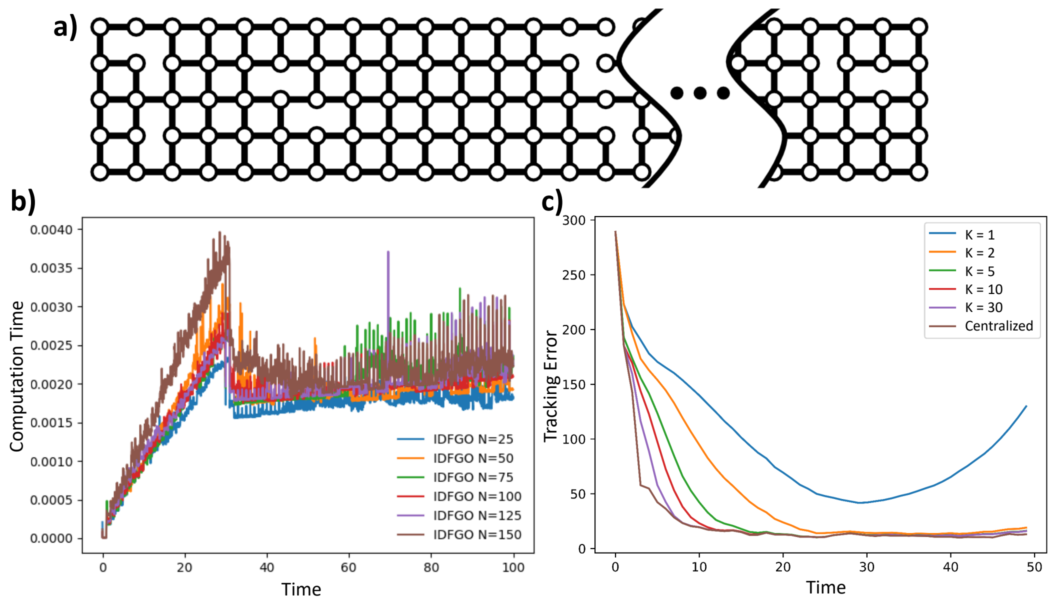


Figure 6.8: Electrical power grid example. Dynamics is given by (6.50). a) The network size N is varied by changing the number of columns in the grid. b) The average computation time per node per iteration as a function of time. Each color shows one simulation with a different network size. As the network size increases, the local computational effort remains the same. c) The optimality gap of iDFGO at steady-state vs. the number of LC-ADMM iterations K.

algorithm is implemented using a modified version of iSAM2 [23] as described in Algorithm 7. The ground truth simulation and iDFGO are run on a single computer and the inter-agent communication is simulated by providing each agent with only its neighbors' information. All the timing results are obtained on a laptop with Intel 2.5 GHz i9-11900H processor.

Scalability, Convergence, and Optimality

Consider an example of a large-scale networked dynamical system (D1) given as a network of electrical power grids [20] involving hundreds of agents. Each plant is tasked to monitor its own state given the local measurement and the information exchanged with the neighboring plants. The dynamics and measurement models for each agent $i \in \mathcal{A}$ are given by

$$r_{i,t+1} = \sum_{j \in \bar{\mathcal{N}}^i} A_{ij} r_{j,t} + w_{i,t}, \quad y_{i,t} = C_i r_{i,t} + v_{i,t}, \tag{6.50}$$

$$A_{ii} = \begin{bmatrix} 1 & \Delta t \\ -\frac{k_i}{m_i}\Delta t & 1 - \frac{d_i}{m_i}\Delta t \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & 0 \\ \frac{k_{ij}}{m_i}\Delta t & 0 \end{bmatrix},$$

$$C_i = [\ 1 \quad 0\ ].$$

The dynamics are locally coupled by A_{ij} terms, and the noise terms $w_{i,t}$ and $v_{i,t}$ are sampled from their respective i.i.d., zero-mean Gaussian distributions. The system parameters k_{ij}, d_i, m_i^{-1} and Δt are set to 0.2, and $k_i = \sum_{(i,j)\in\mathcal{E}} k_{ij}$. The grid network has R rows and C columns with $N = R \times C$ agents in total, as shown in Fig. 6.8(a). The static network topology is defined such that each pair of adjacent agents have a 70% chance of having an edge between them. Because this system is linear, time-invariant, and observable, the convergence of LC-ADMM is globally exponential by Theorem 5 and Corollary 2.

We study the convergence of the consensus errors of LC-ADMM in Fig. 6.5. To isolate the effect of variable-wise consensus, for Fig. 6.5 and this paragraph only, we apply LC-ADMM in batch at each time step instead of iDFGO. Five samples of agent state $r_{i,\tau}$ are randomly selected, and the consensus error between $r_{i,\tau,k}^{(i)}$ and $r_{i,\tau,k}^{(j)}$ for a pair of neighbors i and j is tracked over each consensus iteration $k = [0, \ldots, K-1]$ and each time step since τ (i.e., $t = [\tau, \tau+1, \tau+2, \ldots]$). Each time step is indicated by the black vertical lines in Fig. 6.5 while there are $K = 30$ consensus iterations in between each time step. The new factors are added to the local factor graph at every time step. The figure shows that consensus error converges over k between each time update, but the error increases each time the new factors are added to the graph. This is because the local estimate (i.e., $r_{i,\tau,k}^{(i)}$) changes based on the new information obtained from the augmented factor. Figure 6.5 also shows that the magnitude of the error jumps at each time step decays over time. This is because $t - \tau$ is increasing and the effects of new factors on variable $r_{i,\tau}$ diminish over time. For all the variables sampled, neighbors reach a steady agreement (i.e., $r_{i,\tau,k}^{(i)} = r_{i,\tau,k}^{(j)}$) within 30 time steps since τ.

Next, we verify the scalability of iDFGO in network size and in time. The size of the grid network is varied from N=25 to N=150 agents as shown in Fig. 6.8(a). For each simulation, $K = 30$ iterations are applied per time step. Figure 6.8(b) shows the network-averaged computation time per LC-ADMM update as a function of time in a real-time scenario. The sharp drop in computation time near $t \approx 30$ is due to the internal mechanism within the Bayes tree update of the iSAM2 where it switches from calling a batch update to an incremental update. This switch happens because the ratio between the number of variables affected and the total number of variables in the problem goes under a threshold value. After $t \approx 30$ the computation time remains mostly constant even as the network size increases due to the locality property of LC-ADMM. We observe the average computation time are similar among various network sizes. These plots confirm the scalability of iDFGO in both the size of the network as well as time.

Finally, we study the trade-off between the convergence error and communication bandwidth for iDFGO. We run iDFGO for various numbers of ADMM iterations per time step K where larger K means higher bandwidth. Suppose a ground truth state of agent i time step t is denoted as $r_{i,t}$ and the real-time estimate by agent i is denoted as $r^{(i)}_{i,\tau,K}$ where $\tau = t$ and K is the total number of consensus iteration per time step. Then, the tracking error is defined as $\sum_{i\in\mathcal{A}} \|r^{(i)}_{i,\tau,K} - r_{i,t}\|$. For comparison, we also compute the centralized optimal estimation solution using a batch optimization method at each time horizon. This solution serves as the best possible error that the iDFGO algorithm could get. The results are shown in Fig. 6.8(c). The figure shows the iDFGO tracks the true trajectory $K = 2, 5, 10, 30$ while the algorithm diverges for $K = 1$ because not enough consensus is applied at each time step. The more frequent the communication is, the faster the algorithm converges to the centralized optimal solution. Over time, however, the iDFGO approximates the centralized optimal solution even with a relatively small number of communication per time step.

Distributed Outlier Rejection on Locally-Coupled System

Next, we demonstrate that non-Gaussian noise models can be used with LC-ADMM to solve distributed outlier rejection problems. Continuing with the power grid example, suppose now that the observations model in (6.50) is further corrupted by some sparse outlier noise

$$y_{i,t} = C_i r_{i,t} + v_{i,t} + \psi_{i,t}$$

where $\psi_{i,t}$ is a sparse but large noise. This type of outlier is particularly relevant for large-scale networks that are low-cost but have a higher risk of failure and makes accurate estimation more challenging. Unfortunately, algorithms that explicitly assume Gaussian distribution perform poorly under spurious outliers.

Our approach is to alternatively use convex loss functions that correspond to probability distributions with fatter tails. This modification can be incorporated naturally in iDFGO since the derivation of LC-ADMM does not assume a specific type of noise model. One example of an alternative loss function that models a fatter tail is

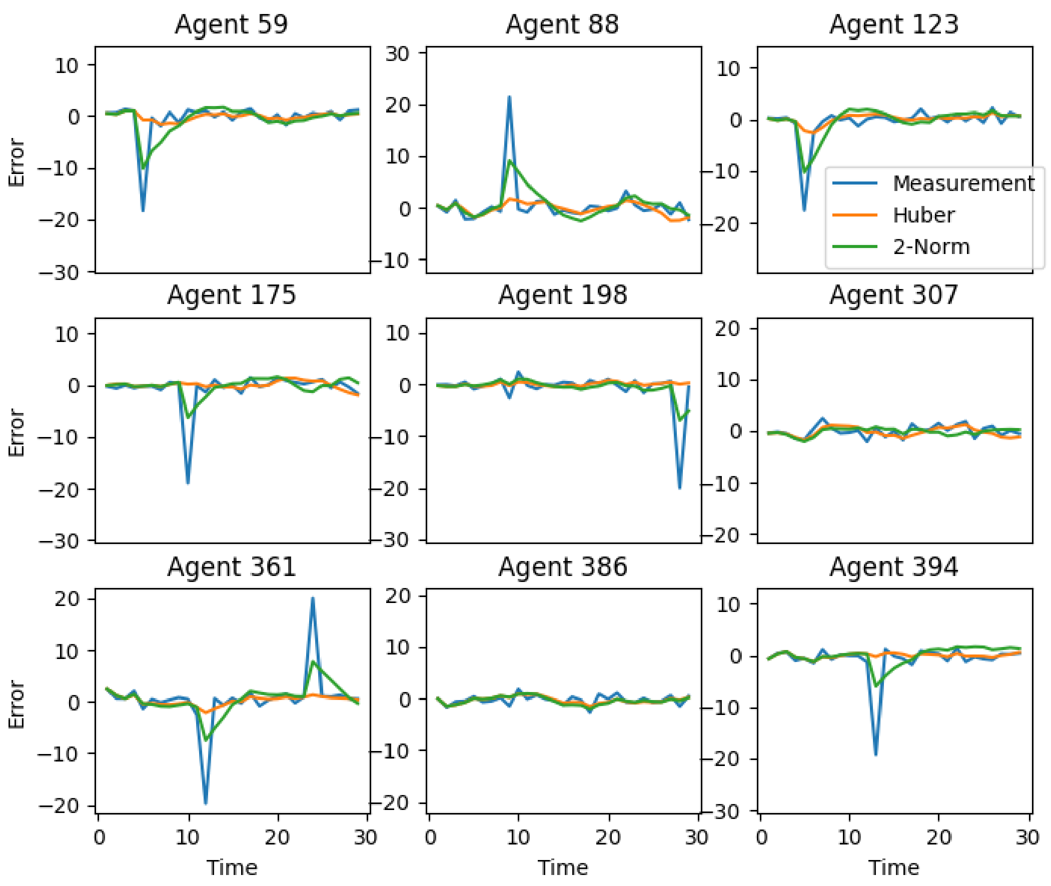


Figure 6.9: Distributed outlier rejection in an power grid example using $\alpha = 1.35$ [1], $\beta = 1$, and K=10. The raw observation error (blue), robust estimate (yellow), and quadratic estimate (green) are shown for nine randomly selected agents out of 400. Errors are defined with respect to the ground truth from the simulation. The robust algorithm successfully rejects the spurious noises that otherwise affect the quadratic estimate.

the Huber loss function [1], which is defined as

$$L_\alpha(x) \triangleq \begin{cases} x^2 & \text{for } |x| \le \alpha, \\ \alpha(2|x| - \alpha), & \text{otherwise} \end{cases} \tag{6.51}$$

for some $\alpha > 0$. The individual agent's objective using the Huber loss function can be written as

$$f_i(\hat{\mathcal{X}}^{(i)}) = \sum_t L_\alpha(\|r_{i,t} - A_i r_{\bar{\mathcal{N}}^i,t-1}\|_{W_i^t}) + L_\alpha(\|y_{i,t} - C_i r_{\bar{\mathcal{N}}^i,t}\|_{V_i^t}).$$

The resulting sub-problem can be written in Quadratic Program (QP) form for which efficient algorithms exist. Because the Huber loss is convex, $o(1/k)$ convergence by Theorem 4 holds.

The simulation considers 400 agents in a connected 20×20 grid. If the nominal observation noise has standard deviation σ, the sparse outlier noise is simulated by setting $\psi_i^t = \pm 20\sigma$ with 2.5% probability of occurrence, and otherwise $\psi_i^t = 0$. Even though the chance of having each outlier is small, there are several outliers somewhere in the network at each time step. We compare the results of applying LC-ADMM with two different loss functions: a 2-norm loss function (baseline) and a Huber loss function (robust estimation). Since the iDFGO assumes Gaussian noise (like iSAM2 does), we apply LC-ADMM over a sliding time horizon of width $T = 30$ for this simulation example only.

Figure 6.9 shows the measurement errors with respect to the first state variable as well as the estimation errors for the baseline and robust estimation algorithms. The figure shows the tracking errors for 9 randomly selected agents out of 400. The baseline algorithm is significantly impacted by the outlier-corrupted signal, while the robust estimation algorithm rejects the outlier well, maintaining low estimation error. To the best of the authors' knowledge, this is the first demonstration of the distributed and localized estimation algorithm that can explicitly reject sparse outliers in a unified framework.

Multi-Agent PGO via LC-ADMM

Next, we apply LC-ADMM and iDFGO to the multi-agent PGO problems (6.46). This section evaluates LC-ADMM on a batch multi-agent PGO while the next section evaluates iDFGO on a real-time problem. We note that LC-ADMM does not show a theoretical guarantee for convergence in non-convex problems such as PGO. However, instead, we empirically show that LC-ADMM locally converges to the centralized optimal solution in these examples. The batch PGO problem was validated on numerical simulations using a benchmark SLAM data set Manhattan 3500 [122], split into 5 segments to emulate multi-agent scenarios.

We compare the estimates computed by LC-ADMM with two state-of-the-art PGO algorithms SE-Sync [2] and DC2-PGO [3]. SE-Sync is a centralized algorithm that is known to converge globally under mild conditions. We solve the centralized PGO problem using SE-Sync and refer to this solution as the centralized optimal solution. The DC2-PGO algorithm is a certifiably correct distributed PGO algorithm (for batch problems). DC2-PGO algorithm is implemented for comparison against LC-ADMM. We use the accelerated version of DC2-PGO and the rank is initialized as

$r = 2$. Note that DC2-PGO optimizes a loss function defined as $L(T_q, T_p, \tilde{T}_{pq}) =$

$$\kappa_{pq} \|R_q - R_p \tilde{R}_{pq}\|_F^2 + \tau_{pq} \|t_p - t_q - R_p \tilde{t}_{pq}\|^2 \tag{6.52}$$

for some $\kappa_{pq} > 0$, $\tau_{pq} > 0$, while the loss function for LC-ADMM is defined as $L(T_q, T_p, \tilde{T}_{pq}) =$

$$\kappa_{pq} \| \log{(R_q^{-1} R_p \tilde{R}_{pq})}^{\wedge} \|^2 + \tau_{pq} \|t_p - t_q - R_p \tilde{t}_{pq}\|^2. \tag{6.53}$$

We denote $\log : \mathrm{SE}(d) \to \mathfrak{se}(d)$ as the logarithm map and $(\cdot)^{\wedge} : \mathbb{R}^{d \times d} \to \mathbb{R}^d$ is the inverse of the skew-symmetric matrix operator.

Once both estimates are computed using their respective loss functions, we use (6.52) to quantify the estimation error for evaluation purposes only. Both algorithms are initialized with the optimal solution to the single-agent PGO using the observations available only to itself (computed using a certifiably correct single-agent PGO algorithm [2]).

First, the colored trajectories in Fig. 6.10 show the LC-ADMM estimate by the five agents after $k = 10$ iterations. Only after a small number of iterations, LC-ADMM converges to the centralized optimal solution (grey, computed by SE-Sync). The figure also qualitatively confirms that LC-ADMM is not stuck at some local minima. Next, Fig. 6.11 shows the convergence rates of LC-ADMM and DC2-PGO[4]. Each agent only contributes its own poses and not a copy of the other agents' poses. The cost is defined as the centralized PGO objective function with (6.52) as the loss function and evaluated at $\tilde{\mathcal{X}}$. The optimal cost f^* is evaluated at the globally optimal solution to the centralized PGO computed by [2] and the normalized cost is defined as f/f^*. The normalized cost for both algorithms is shown as a function of iterations in Fig. 6.11. While LC-ADMM does not have any guarantees to converge to the correct global optimal solution for a general non-convex problem, Fig. 6.11 shows that LC-ADMM converged to the optimal solution in this specific example. LC-ADMM also converged faster than DC2-PGO even though both algorithms start from the same initial trajectories. This result empirically suggests the applicability of LC-ADMM when provided with an initial guess that is good enough to avoid the local minimum.

[4]In a general multi-agent PGO, the outer loop of DC2-PGO iterates on possible ranks of the low-rank Semi-Definite Programming. In this example, DC2-PGO only iterates this outer loop once before convergence. Therefore, the convergence of DC2-PGO in Fig. 6.11 equivalently represents the convergence of its inner loop, the Riemannian Block Coordinated Descent algorithm (RBCD)[3].

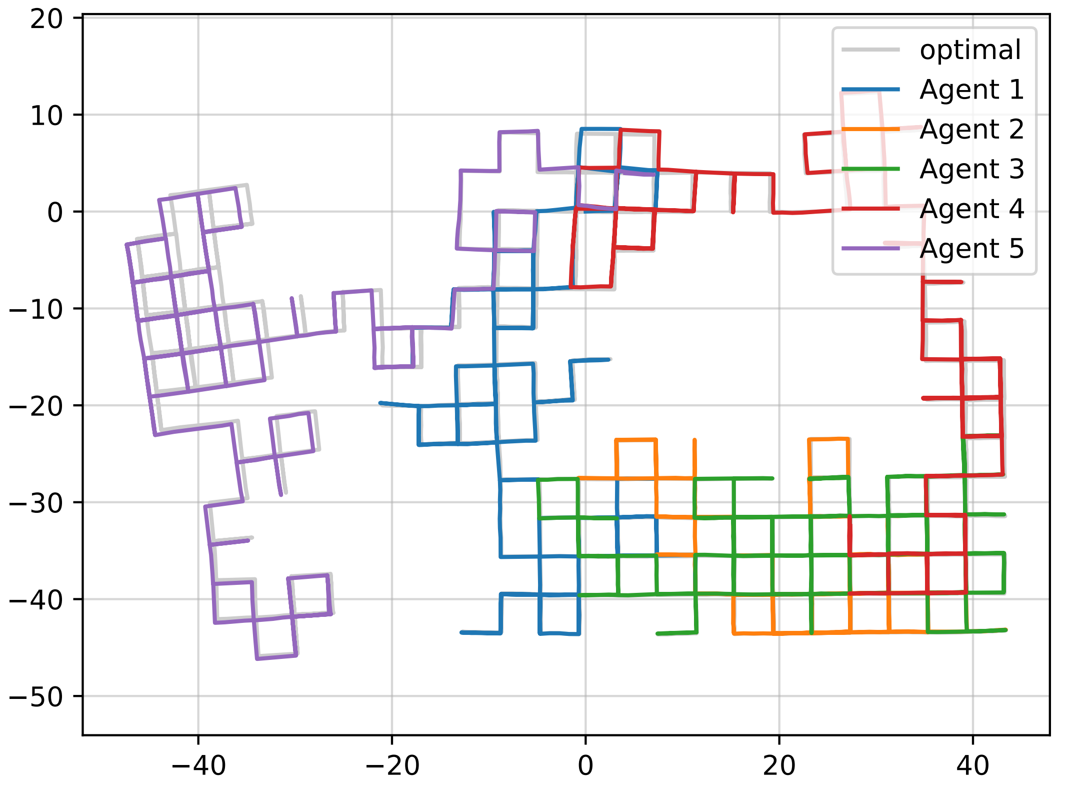


Figure 6.10: Five agents cooperatively solve the distributed PGO problem. The pose estimates are shown for LC-ADMM only after $k = 10$ iterations (colored) and for the optimal solution (gray) computed by a centralized method (SE-Sync [2]).

Multi-Agent PGO via iDFGO

Next, we evaluate iDFGO in a real-time multi-agent PGO example. In this example, 9 robots traverse an environment in a formation. The trajectories of the 9 agents are shown on the left in Fig. 6.2. The task is to maintain both a good global map as well as good relative estimates. The dataset contains noisy observations for (1) odometry, (2) relative pose between neighbor robots at a given time (shown in red), and (3) loop closures with some nearby poses (shown in blue). Without any inter-agent cooperation, the estimate that each agent computes will have a large drift as well as a large relative pose estimation error (shown in the center of Fig. 6.2). We set the number of consensus iterations applied at each time step to be $K = 5$.

A snapshot of the trajectory estimates computed by the iDFGO algorithm at time $t = 320, k = 0$ is shown on the right in Figure 6.2. Each color corresponds to a different agent's trajectory estimate and the gray plot shows the centralized optimal solution computed SE-Sync [2]. It shows that the trajectories estimated by iDFGO

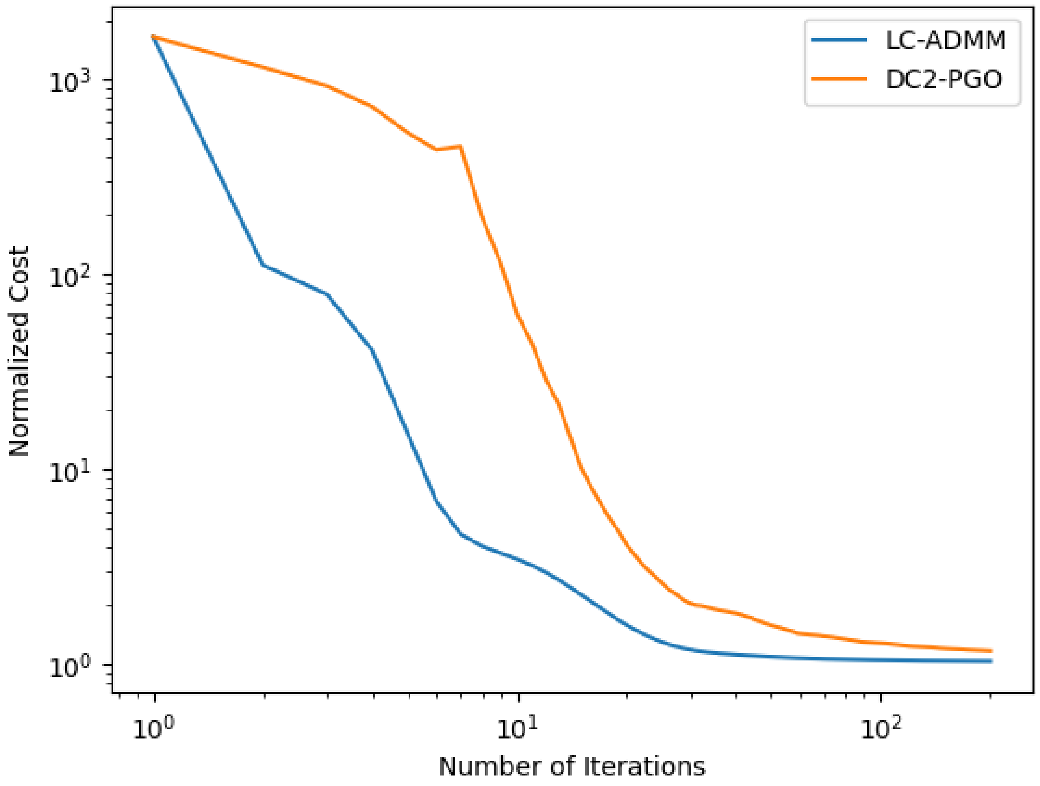


Figure 6.11: Normalized cost vs. number of iterations for LC-ADMM and DC2-PGO [3] on M3500 dataset.

match the optimal solution. The darker color in each agent's trajectory denotes variables that were affected by the iDFGO update at step, while the lighter color denotes unaffected variables. The figure shows that iDFGO successfully updates the recent subset of the factor graph in an incremental fashion. Some of the agents have a larger number of affected variables than others. This is explained by the loop closure constraints on those agents affecting more variables from the past.

More details of the number of affected variables over the trajectory are shown in Fig. 6.6. The solid lines show the time history of the number of affected variables at the given time while the dotted line shows the total number of variables updated if the whole trajectory of each agent is estimated in a batch. While the problem size increases for the batch case time increases, the size remains bounded for the iDFGO algorithm.

This simulation demonstrated that the iDFGO algorithm solves the multi-agent PGO for a team of robots traversing an environment in a formation. The estimate matches the centralized optimal solution well. The algorithm is efficient as it involves a small number of consensus iterations per time step and the factor graphs are updated in an incremental fashion, leading to a more time-scalable algorithm.

6.6 Chapter Summary

This chapter presented new algorithms to solve certain classes of Distributed Factor Graph Optimization (DFGO) problems. The Maximum A Posteriori estimation problem was formulated as a spatially-decomposable factor graph optimization problem by exploiting the sparsity structure of locally-coupled systems. First, showed the Localized Consensus ADMM (LC-ADMM) algorithm has various advantages when solving DFGO in a batch over a fixed time horizon. LC-ADMM scales well with the number of agents, incorporates robust convex loss objectives, solves problems fully in parallel, and does not need information about global graph topology. We have two new theoretical convergence results on LC-ADMM: (1) it converges at $o(1/k)$ rate if the objective function is convex and has a unique solution, and (2) it converges exponentially if the objective is strongly convex and its subdifferential is Lipschitz continuous. Next, using LC-ADMM as a backbone, we derived the Incremental Distributed Factor Graph Optimization algorithm (iDFGO) for real-time problems. The iDFGO algorithm integrates LC-ADMM with an extended version of iSAM2 to incrementally recompute the local factor graph optimization problem. The iDFGO is scalable both in the number of agents and in time.

The performance of LC-ADMM and iDFGO was evaluated in numerical simulations with examples from the electrical power grid and multi-agent pose graph optimization. The simulation results verified scalability properties and illustrated the trade-offs between optimality and communication. The multi-agent pose graph optimization examples showed that the algorithm converged to the correct optimal solution despite problems being non-convex, indicating the possible applicability to a broad class of problems.

In the overall road map of this thesis, LC-ADMM and iDFGO address the shortcomings identified in DPE (Chapter 3) and MSEPS (Chapter 4). DPE was $O(1)$ scalable with respect to the size of the network but there was a gap when compared to the centralized optimal solution. MSEPS used Decentralized Extended Information Filter which approximates the centralized optimal solution; however, it was not scalable with respect to a number of landmarks, as they were not localized. On the other hand, LC-ADMM addresses both optimality and scalability simultaneously for large-scale problems. In comparison to DPE, iDFGO is a preferred choice when estimation accuracy is a high priority in the system design process. While both DPE and iDFGO are scalable in memory, computation, and communication exchange, iDFGO requires more overhead on each of these resources due to the complexity of the optimization-based algorithm. Therefore, it is possible that DPE is preferred over iDFGO in some scenarios; e.g., when the system is extremely resource-limited and DPE provides sufficient estimation accuracy. However, because the computational effort needed for iDFGO is comparable to single-agent SLAM algorithms which have been shown to run on small, low-cost edge computing devices available today (e.g., NVIDIA Jetson TX2), iDFGO is suitable in many multi-agent robotic platforms, including spacecraft swarms.

Chapter 7

CONCLUSION

In Chapter 3, we presented the Decentralized Pose Estimation (DPE) algorithm that solves the swarm localization problem for formation flying spacecraft. The DPE considers *ad hoc* relative sensing and communication networks to determine a set of observable spacecraft and shares these spacecraft's measurements to jointly estimate their poses with respect to the LVLH frame at each time step. As a part of the DPE, the Swarm Reference Frame Estimation (SRFE) algorithm applies the information consensus filter to estimate the common LVLH frame in a decentralized fashion. The DPE is a local, decentralized algorithm that has a constant complexity with respect to the swarm size. Numerical simulations verify that the estimation errors of the DPE are improved compared to those for no cooperation cases and that the computation time remains constant as the swarm size increases. An experimental result using the air-bearing spacecraft simulators demonstrates good DPE performance using vision-based relative pose measurements with *ad hoc* networks.

In Chapter 4, we presented the cooperative vision-based pose estimation algorithm called Multi-spacecraft Simultaneous Estimation of Pose and Shape (MSEPS). MSEPS is posed as a distributed sensor network paradigm. Using inter-spacecraft communication, MSEPS tightly integrates the vision-based feature tracking problem with a distributed estimation framework. We provided an overview of the multi-spacecraft algorithm architecture that consists of the computer vision pipeline, communication, and back-end filtering. We proposed the extended decentralized information filtering that approximately solves the minimum variance estimate of the global information but is implemented in a distributed fashion, and proposed an improvement in computation that exploits the special structure of the MSEPS problem. We validated the distributed algorithm and some of the algorithm pipelines using the simulation.

In Chapter 5, we developed new high-fidelity simulation tools to aid the development and testing of robotics algorithms for on-orbit servicing. First, we developed ROS-Basilisk, a software that enables the astrodynamics simulation software Basilisk to run in real-time alongside ROS2. ROS-Basilisk software translates messages between Basilisk and ROS2 such that it allows autonomy algorithms implemented in ROS2 to interact with high-fidelity spacecraft dynamics simulation of Basilisk in a closed-loop fashion. Second, we present a camera simulation module named ROS-NeRF which uses the Neural Radiance Fields to rapidly render spacecraft images in real time. Because the neural network only needs to learn the specific 3D scene (i.e., target spacecraft), this approach can rapidly generate realistic images much faster than high-fidelity rendering methods such as Blender. We evaluated the performance of ROS-NeRF by applying basic computer vision tasks of relative navigation such as object detection and keypoint matching to the NeRF-rendered images. Finally, we integrated these simulation tools along with an example on-orbit inspection algorithm that involves spacecraft object detection, formation control, and attitude control to demonstrate a closed-loop simulation in ROS2.

In Chapter 6, we presented new algorithms to solve certain classes of Distributed Factor Graph Optimization (DFGO) problems. The Maximum A Posteriori estimation problem was formulated as a spatially-decomposable factor graph optimization problem by exploiting the sparsity structure of locally-coupled systems. First, showed the Localized Consensus ADMM (LC-ADMM) algorithm has various advantages when solving DFGO in a batch over a fixed time horizon. LC-ADMM scales well with the number of agents, incorporates robust convex loss objectives, solves problems fully in parallel, and does not need information about global graph topology. We have two new theoretical convergence results on LC-ADMM: (1) it converges at $o(1/k)$ rate if the objective function is convex and has a unique solution, and (2) it converges exponentially if the objective is strongly convex and its subdifferential is Lipschitz continuous. Next, using LC-ADMM as a backbone, we derived the Incremental Distributed Factor Graph Optimization algorithm (iDFGO) for real-time problems. The iDFGO algorithm integrates LC-ADMM with an extended version of iSAM2 to incrementally recompute the local factor graph optimization problem. The iDFGO is scalable both in the number of agents and in time.

Finally, the performance of LC-ADMM and iDFGO was evaluated in numerical simulations with examples from the electrical power grid and multi-agent pose graph optimization. The simulation results verified scalability properties and illustrated the trade-offs between optimality and communication. The multi-agent pose graph optimization examples showed that the algorithm converged to the correct optimal solution despite problems being non-convex, indicating the possible applicability to a broad class of problems.

BIBLIOGRAPHY

[1] P. J. Huber, *Robust statistics*. Hoboken, New Jersey, USA: John Wiley & Sons, 2004, vol. 523.

[2] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, “SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group,” *International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 95–125, 2019.

[3] Y. Tian, K. Khosoussi, D. M. Rosen, and J. P. How, “Distributed certifiably correct pose-graph optimization,” *IEEE Transactions on Robotics*, 2021.

[4] F. Y. Hadaegh, S.-J. Chung, and H. M. Manohara, “On development of 100-gram-class spacecraft for swarm applications,” *IEEE Systems Journal*, vol. 10, no. 2, pp. 673–684, 2016.

[5] O. Brown, P. Eremenko, and P. Collopy, “Value-centric design methodologies for fractionated spacecraft: Progress summary from phase i of the darpa system f6 program,” in *AIAA Space 2009 Conference & Exposition*, 2009, p. 6540.

[6] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, “Distributed Kalman filtering based on consensus strategies,” *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 622–633, 2008.

[7] R. Olfati-Saber, “Kalman-consensus filter: Optimality, stability, and performance,” in *48h IEEE Conference on Decision and Control*, 2009, pp. 7036–7042.

[8] S. Park and N. C. Martins, “Design of distributed lti observers for state omniscience,” *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 561–576, 2016.

[9] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, “Information weighted consensus filters and their application in distributed camera networks,” *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3112–3125, 2013.

[10] S. Bandyopadhyay and S.-J. Chung, “Distributed bayesian filtering using logarithmic opinion pool for dynamic sensor networks,” *Automatica*, vol. 97, pp. 7–17, 2018.

[11] U. A. Khan and J. M. Moura, “Distributing the Kalman filter for large-scale systems,” *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4919–4935, 2008.

[12] G. Han, H. Xu, T. Q. Duong, J. Jiang, and T. Hara, “Localization algorithms of wireless sensor networks: A survey,” *Telecommunication Systems*, vol. 52, no. 4, pp. 2419–2436, 2013.

[13] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, “Collaborative sensor network localization: Algorithms and practical issues,” *Proc. the IEEE*, vol. 106, no. 6, pp. 1089–1114, 2018.

[14] V. Kekatos and G. B. Giannakis, “Distributed robust power system state estimation,” *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1617–1626, 2012.

[15] F. Bullo, J. Cortés, and S. Martínez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton, New Jersey, USA: Princeton University Press, 2009.

[16] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, “GLAS: global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4249–4256, 2020.

[17] K. Matsuka, A. O. Feldman, E. S. Lupu, S.-J. Chung, and F. Y. Hadaegh, “Decentralized formation pose estimation for spacecraft swarms,” *Advances in Space Research*, vol. 67, no. 11, pp. 3527–3545, 2021. DOI: 10.1016/j.asr.2020.06.016,

[18] A. Cunningham, V. Indelman, and F. Dellaert, “DDF-SAM 2.0: Consistent distributed smoothing and mapping,” in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5220–5227.

[19] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni, “System level synthesis,” *Annual Reviews in Control*, vol. 47, pp. 364–393, 2019.

[20] Y.-S. Wang, S. You, and N. Matni, “Localized distributed Kalman filters for large-scale systems,” *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 52–57, 2015, 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems NecSys 2015, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.10.306.

[21] C. Cadena, L. Carlone, H. Carrillo, *et al.*, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[22] S. Thrun and Y. Liu, “Multi-robot SLAM with sparse extended information filers,” in *Robotics Research. The Eleventh International Symposium*, Springer, 2005, pp. 254–266.

[23] 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.

[24] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous localization and mapping via square root information smoothing,” *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[25] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert, “Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models,” *International Journal of Robotics Research*, vol. 36, no. 12, pp. 1286–1311, 2017.

[26] F. Dellaert, M. Kaess, *et al.*, “Factor graphs for robot perception,” *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[28] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[29] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[30] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[31] J. C. Duchi, A. Agarwal, and M. J. Wainwright, “Dual averaging for distributed optimization: Convergence analysis and network scaling,” *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2011.

[32] A. Nedić and A. Olshevsky, “Distributed optimization over time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.

[33] I. Necoara, V. Nedelcu, and I. Dumitrache, “Parallel and distributed optimization methods for estimation and control in networks,” *Journal of Process Control*, vol. 21, no. 5, pp. 756–766, 2011.

[34] M. Zhu and S. Martínez, “On distributed convex optimization under inequality and equality constraints,” *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2011.

[35] B. Gharesifard and J. Cortés, “Distributed continuous-time convex optimization on weight-balanced digraphs,” *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 781–786, 2013.

[36] Y. Wang, W. Yin, and J. Zeng, “Global convergence of ADMM in nonconvex nonsmooth optimization,” *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[37] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, “On the linear convergence of the ADMM in decentralized consensus optimization,” *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[38] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization—part i: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[39] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 3495–3501.

[40] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2014.

[41] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proc. 2020 American Control Conference*, 2020, pp. 3619–3626.

[42] O. Shorinwa and M. Schwager, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in *International Symposium on Multi-Robot and Multi-Agent Systems*, 2021, pp. 56–65.

[43] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Delft, Netherlands: Now Publishers Inc, 2011.

[44] A. Ogilvie, J. Allport, M. Hannah, and J. Lymer, "Autonomous satellite servicing using the orbital express demonstration manipulator system," in *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'08)*, 2008, pp. 25–29.

[45] M. Pyrak and J. Anderson, "Performance of northrop grumman's mission extension vehicle (mev) rpo imagers at geo," in *Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea and Space Vehicles and Infrastructure 2022*, SPIE, vol. 12115, 2022, pp. 64–82.

[46] P. Bodin, R. Larsson, F. Nilsson, C. Chasset, R. Noteborn, and M. Nylund, "Prisma: An in-orbit test bed for guidance, navigation, and control experiments," *Journal of Spacecraft and Rockets*, vol. 46, no. 3, pp. 615–623, 2009.

[47] J. W. Gangestad, C. C. Venturini, D. A. Hinkley, and G. Kinum, "A sat-to-sat inspection demonstration with the aerocube-10 1.5 u cubesats," in *35th Annual Small Satellite Conference*, 2021.

[48] C. Blackerby, A. Okamoto, S. Iizuka, *et al.*, "The elsa-d end-of-life debris removal mission: Preparing for launch," in *Proceedings of the International Astronautical Congress, IAC*, vol. 8, 2019.

[49] M. Lavalle, I. Seker, J. Ragan, *et al.*, “Distributed aperture radar tomographic sensors (darts) to map surface topography and vegetation structure,” in *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, IEEE, 2021, pp. 1090–1093.

[50] A. Battaglia, P. Kollias, R. Dhillon, *et al.*, “Spaceborne cloud and precipitation radars: Status, challenges, and ways forward,” *Reviews of Geophysics*, vol. 58, no. 3, e2019RG000686, 2020.

[51] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, “A survey of spacecraft formation flying guidance and control (part i): Guidance,” in *Proceedings of the American Controls Conference*, Denver, Colorado, Jun. 2003, pp. 1733–1739.

[52] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, “A survey of spacecraft formation flying guidance and control. part ii: Control,” in *Prof. 2004 American control conference*, Ieee, vol. 4, 2004, pp. 2976–2985.

[53] M. Romano, D. A. Friedman, and T. J. Shay, “Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target,” *Journal of Spacecraft and Rockets*, vol. 44, no. 1, pp. 164–173, 2007.

[54] B. E. Tweddle and A. Saenz-Otero, “Relative computer vision-based navigation for small inspection spacecraft,” *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 5, pp. 969–978, 2015.

[55] G. Zhang, P. Vela, P. Tsiotras, and D.-M. Cho, “Efficient closed-loop detection and pose estimation for vision-only relative localization in space with a cooperative target,” in *AIAA SPACE 2014 Conference and Exposition*, 2014, p. 4262.

[56] K. Black, S. Shankar, D. Fonseka, J. Deutsch, A. Dhir, and M. R. Akella, “Real-time, flight-ready, non-cooperative spacecraft pose estimation using monocular imagery,” *arXiv preprint arXiv:2101.09553*, 2021.

[57] S. Sharma and S. D’Amico, “Neural network-based pose estimation for noncooperative spacecraft rendezvous,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, pp. 4638–4658, 2020.

[58] B. E. Tweddle, A. Saenz-Otero, J. J. Leonard, and D. W. Miller, “Factor graph modeling of rigid-body dynamics for localization, mapping, and parameter estimation of a spinning object in space,” *Journal of Field Robotics*, vol. 32, no. 6, pp. 897–933, 2015.

[59] T. P. Setterfield, D. W. Miller, J. J. Leonard, and A. Saenz-Otero, “Mapping and determining the center of mass of a rotating object using a moving observer,” *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 83–103, 2018.

[60] T. P. Setterfield, D. W. Miller, A. Saenz-Otero, E. Frazzoli, and J. J. Leonard, "Inertial properties estimation of a passive on-orbit object using polhode analysis," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 10, pp. 2214–2231, 2018.

[61] S. Augenstein and S. M. Rock, "Improved frame-to-frame pose tracking during vision-only SLAM/SFM with a tumbling target," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3131–3138.

[62] V. Capuano, S. R. Alimo, A. Q. Ho, and S.-J. Chung, "Robust features extraction for on-board monocular-based spacecraft pose acquisition," in *AIAA Scitech 2019 Forum*, 2019, p. 2005.

[63] A. Harvard, V. Capuano, E. Y. Shao, and S.-J. Chung, "Spacecraft pose estimation from monocular images using neural network based keypoints and visibility maps," in *AIAA Scitech 2020 Forum*, 2020, p. 1874.

[64] M. Dor, T. Driver, K. Getzandanner, and P. Tsiotras, "AstroSLAM: Autonomous monocular navigation in the vicinity of a celestial small body–theory and experiments," *arXiv preprint arXiv:2212.00350*, 2022.

[65] D. Fourie, B. E. Tweddle, S. Ulrich, and A. Saenz-Otero, "Flight results of vision-based navigation for autonomous spacecraft inspection of unknown objects," *Journal of Spacecraft and Rockets*, vol. 51, no. 6, pp. 2016–2026, 2014.

[66] L. Fluckiger and B. Coltin, "Astrobee robot software: Enabling mobile autonomy on the iss," Tech. Rep., 2019.

[67] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh, "Swarm-keeping strategies for spacecraft under j2 and atmospheric drag perturbations," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 5, pp. 1492–1506, 2012.

[68] S. Bandyopadhyay, R. Foust, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Review of formation flying and constellation missions using nanosatellites," *Journal of Spacecraft and Rockets*, vol. 53, no. 3, pp. 567–578, 2016.

[69] S.-J. Chung and F. Y. Hadaegh, "Swarms of femtosats for synthetic aperture applications," in *Proc. of the Fourth International Conference on Spacecraft Formation Flying Missions & Technologies*, St-Hubert, Quebec, May 2011.

[70] W. Cash, "Detection of earth-like planets around nearby stars using a petal-shaped occulter," *Nature*, vol. 442, no. 7098, pp. 51–53, 2006.

[71] E. Gdoutos, C. Leclerc, F. Royer, *et al.*, "A lightweight tile structure integrating photovoltaic conversion and rf power transfer for space solar power applications," in *2018 AIAA Spacecraft Structures Conference*, 2018, p. 2202.

[72] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[73] J. Badger, D. Gooding, K. Ensley, K. Hambuchen, and A. Thackston, "Ros in space: A case study on robonaut 2," in *Robot Operating System (ROS)*, Springer, 2016, pp. 343–373.

[74] G. Shi, W. Hönig, X. Shi, Y. Yue, and S.-J. Chung, "Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions," *IEEE Transactions on Robotics*, 2021.

[75] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Information weighted consensus," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, IEEE, 2012, pp. 2732–2737.

[76] B. Açıkmeşe, M. Mandić, and J. L. Speyer, "Decentralized observers with consensus filters for distributed discrete-time linear systems," *Automatica*, vol. 50, no. 4, pp. 1037–1052, 2014.

[77] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[78] Y. K. Nakka, R. C. Foust, E. S. Lupu, *et al.*, "Six degree-of-freedom spacecraft dynamics simulator for formation control research," in *2018 AAS/AIAA Astrodynamics Specialist Conference*, American Institute of Aeronautics and Astronautics, 2018, pp. 1–20.

[79] R. Foust, E. S. Lupu, Y. K. Nakka, S.-J. Chung, and F. Y. Hadaegh, "Ultrasoft electromagnetic docking with applications to in-orbit assembly," in *69th International Astronautical Congress (IAC), Bremen, Germany*, 2018.

[80] H. Schaub and J. L. Junkins, "Analytical mechanics of space systems," in J. A. Schetz, Ed. Reston, Virginia: American Institute of Aeronautics and Astronautics, 2005, ch. Spacecraft Formation Flying.

[81] K. Yamanaka and F. Ankersen, "New state transition matrix for relative motion on an arbitrary elliptical orbit," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 1, pp. 60–66, 2002.

[82] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive survey and assessment of spacecraft relative motion dynamics models," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 8, pp. 1837–1859, 2017.

[83] F. L. Markley and J. L. Crassidis, *Fundamentals of spacecraft attitude determination and control*. New York: Springer, 2014, vol. 33.

[84] F. L. Markley, "Attitude error representations for Kalman filtering," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[85] R. Hermann and A. Krener, "Nonlinear controllability and observability," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 728–740, 1977.

[86] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[87] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014, ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2014.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[88] R. Opromolla, G. Fasano, G. Rufino, and M. Grassi, "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations," *Progress in Aerospace Sciences*, vol. 93, pp. 53–72, 2017.

[89] J. M. Kelsey, J. Byrne, M. Cosgrove, S. Seereeram, and R. K. Mehra, "Vision-based relative pose estimation for autonomous rendezvous and docking," in *2006 IEEE Aerospace Conference*, IEEE, 2006, pp. 1–20.

[90] V. Capuano, K. Kim, A. Harvard, and S.-J. Chung, "Monocular-based pose determination of uncooperative space objects," *Acta Astronautica*, vol. 166, pp. 493–506, 2020.

[91] L. P. Cassinis, R. Fonod, and E. Gill, "Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft," *Progress in Aerospace Sciences*, vol. 110, no. 100548, Oct. 2019.

[92] V. Capuano, A. Harvard, Y. Lin, and S.-J. Chung, "DGNSS-vision integration for robust and accurate relative spacecraft navigation," in *Proceedings of the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), Miami, Florida*, 2019, pp. 2923–2939.

[93] H. Li and R. Hartley, "Five-point motion estimation made easy," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 1, 2006, pp. 630–633. DOI: 10.1109/ICPR.2006.579.

[94] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[95] S. Lee, V. Capuano, A. Harvard, and S.-J. Chung, "Fast uncertainty estimation for deep learning based optical flow," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[96] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[97] V. Capuano, C. Botteron, J. Leclère, J. Tian, Y. Wang, and P.-A. Farine, “Feasibility study of GNSS as navigation system to reach the moon,” *Acta Astronautica*, vol. 116, pp. 186–201, 2015.

[98] P. W. Kenneally, S. Piggott, and H. Schaub, “Basilisk: A flexible, scalable and modular astrodynamics simulation framework,” *Journal of Aerospace Information Systems*, vol. 17, no. 9, pp. 496–507, 2020.

[99] K. Matsuka, C. Ohenzuwa, and S.-J. Chung, “Rapid synthetic image generation using neural radiance fields for vision-based formation flying spacecraft,” in *2023 33rd AAS/AIAA Space Flight Mechanics Meeting*, AIAA, 2023,

[100] H. Benninghoff, F. Rems, E.-A. Risse, and C. Mietner, “European proximity operations simulator 2.0 (epos)-a robotic-based rendezvous and docking simulator,” *Journal of Large-Scale Research Facilities*, 2017.

[101] S. Van Overeem and H. Schaub, “Small satellite formation flying application using the basilisk astrodynamics software architecture,” in *International Workshop on Satellite Constellations and Formation Flying, University of Strathclyde, Glasgow, Scotland*, 2019, pp. 19–88.

[102] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[103] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *arXiv preprint arXiv:2201.05989*, 2022.

[104] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, “Plenoctrees for real-time rendering of neural radiance fields,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[105] C. Reiser, S. Peng, Y. Liao, and A. Geiger, “Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345.

[106] H. Turki, D. Ramanan, and M. Satyanarayanan, “Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 922–12 931.

[107] M. Tancik, V. Casser, X. Yan, *et al.*, “Block-nerf: Scalable large scene neural view synthesis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[108] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[109] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[110] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, "Inerf: Inverting neural radiance fields for pose estimation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2021, pp. 1323–1330.

[111] B. O. Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: `http://www.blender.org`.

[112] G. Jocher, A. Chaurasia, A. Stoken, *et al.*, *ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations*, version v6.2, Aug. 2022. DOI: `10.5281/zenodo.7002879`. [Online]. Available: `https://doi.org/10.5281/zenodo.7002879`.

[113] B. E. Tweddle, "Computer vision-based localization and mapping of an unknown, uncooperative and spinning target for spacecraft proximity operations," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[114] M. Dor and P. Tsiotras, "ORB-SLAM applied to spacecraft non-cooperative rendezvous," in *2018 Space Flight Mechanics Meeting*, 2018, p. 1963.

[115] V. Capuano, K. Kim, J. Hu, A. Harvard, and S.-J. Chung, "Monocular-based pose determination of uncooperative known and unknown space objects," in *Proc. 69th International Astronautical Congress (IAC)*, 2018.

[116] K. Matsuka, A. Santamaria-Navarro, V. Capuano, A. Harvard, A. Rahmani, and S.-J. Chung, "Collaborative pose estimation of an unknown target using multiple spacecraft," in *2021 IEEE Aerospace Conference*, IEEE, 2021, pp. 1–11. DOI: `10.1109/AERO50100.2021.9438352`,

[117] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[118] R. Zhang and J. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *International Conference on Machine Learning*, 2014, pp. 1701–1709.

[119] K. Huang and N. D. Sidiropoulos, "Consensus-ADMM for general quadratically constrained quadratic programming," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5297–5310, 2016.

[120] T. Fan and T. Murphey, “Majorization minimization methods for distributed pose graph optimization with convergence guarantees,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 5058–5065.

[121] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, “Parallel multi-block ADMM with o(1/k) convergence,” *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[122] E. Olson, J. Leonard, and S. Teller, “Fast iterative alignment of pose graphs with poor initial estimates,” in *2006 IEEE International Conference on Robotics and Automation*, 2006, pp. 2262–2269.