System Identification and Control of Multiagent Systems Through Interactions

Jaskaran Singh Grover

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Prof. Katia Sycara (co-chair)
Prof. Changliu Liu (co-chair)
Prof. Zac Manchester
Dr. Mario Santillo

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

CMU-RI-TR-23-04

To my mother, my strength,
and my brother, my joy.

Abstract

This thesis investigates the problem of inferring the underlying dynamic model of individual agents of a multiagent system (MAS) and using these models to shape the MAS's behavior using robots extrinsic to the MAS. We investigate (a) how an observer can infer the latent task and inter-agent interaction constraints from the agents' motion and (b) how the observer can elicit a desired behavior out of the MAS by orchestrating its interactions with robots. The ability to learn individual dynamics models of an aggregated system has several applications such as learning local rules in biological swarms that give rise to emergent behavior and learning tactics of an adversarial multirobot team. Likewise, the ability to shape behavior using extrinsic robots can be used to defend against an adversarial team of robots and guide humans using robots as in social navigation.

The first part of this thesis focuses on the model learning problem. We model agents as integrators that solve a reactive optimization to calculate velocities for mediating between goal-directed motions and collision avoidance with other agents. We develop several estimators that allow an observer to infer this model's parameters and show that the learned parameters indeed rationalize the observed motions. Necessary identifiability conditions are derived that guarantee correct inference. Our proposed estimators include adaptive observers, Kalman filters and several inverse optimization algorithms that are robust to both measurement noise and model mismatch. To demonstrate this robustness, we evaluate these estimators on a pedestrian dataset and learn each pedestrian's desired velocity, aggressiveness coefficients and safety margins with walls, obstacles and other pedestrians.

The second part of this thesis focuses on eliciting a desired behavior out of the MAS by inducing interactions with robots. While the theory we develop is general, we consider the dog-sheep herding problem as a use case that requires controlling dog robots to repel sheep agents from a critical zone. We incorporate non-collocated feedback linearization in an optimization-based framework to compute the desired controls for the dogs. Both centralized and distributed implementations are developed to cater to the scalability, feasibility and budget-efficiency objectives. We validate the correctness of these controllers in multiple experiments on the CMU multirobot arena. We also develop a robust extension of these controllers, which we term control-barrier function based semidefinite programs (CBF-SDPs), that guarantee zone defense despite uncertainty in the sheep's dynamics. Finally, we conclude this thesis with an integration of the robust model learning algorithms with robust control algorithms followed by experimental validation on the multirobot arena.

Acknowledgements

I have been fortunate to have had a fruitful and wholesome PhD journey. This has been made possible by the confluence of good luck, good company, encouragement from my family, great mentorship and definitely my hard work. I want to take a moment to thank my companions in this journey who made this possible.

Firstly, I want to thank my wonderful advisors Changliu and Katia. Thank you for being my guiding compass in navigating what can be a rough sea. Katia gave me the freedom to explore many problems and chart my own PhD path. She gave me the opportunity and resources to construct a big multirobot arena in the lab that proved to be the most instrumental project to my PhD. She has been one of the most eclectic and passionate researchers I interacted with. I am inspired by her persistence in continuing to innovate and explore uncharted research domains.

Aside from being a model mentor, researcher and teacher, Changliu has been an understanding, caring and kind friend to me. She strives to develop a cordial and conducive lab culture. She gave me the freedom to explore problems, helped me identify connections of my work to practical use cases when I myself lacked faith and helped me with the nitty-gritty of proofs and derivations that always was the most gratifying part of my PhD. Her frequent feedback on my research, my papers and my thesis has made a better researcher. In fact, most of the technical topics I explored in my thesis are based on the lessons that I took away from her classes and from her own body of work that has been so foundational. Thank you for agreeing to take me on as your first PhD student. I am both lucky and proud to have had that opportunity.

I also want to thank my committee members, Mario and Zac. I am thankful to Mario for helping me identify connections of my work with many problems in self-driving cars and giving me technical feedback on my thesis and research. I am thankful to Zac for urging me to think about real world transferrability of my work. During my PhD, I had the chance to work at Microsoft Research. I want to thank Jayesh Gupta for being a great mentor over those months and for our daily stimulating conversations.

Before coming to Carnegie Mellon, I spent a brief amount of time studying controls and optimization at UCLA. While short, I learned an enormous amount from excellent teachers at UCLA. I want to thank Prof. Robert MCloskey for teaching linear dynamic systems in the most endearing way. It was my first graduate controls course and to this date remains my favorite. Many of the results in chapter 4 of this thesis are applications of topics I learned in his course. I want to thank Prof. Lieven Vandenberg for his amazing lectures and for writing the most prolific book on convex optimization that has been instrumental in almost all the chapters of my thesis. I want to thank Prof. Tetsuya Iwasaki for teaching robust control in the most endearing way. I looked forward to coming to your lectures because you made the course absolutely delightful. I want to thank my mentors at Intel Labs, Kumar Ranganathan and Venkat Natarajan for taking a chance on me. Such a full circle on things, half of my thesis has used different forms of Kalman filters that I first learned about while working at Intel. I want to thank you for putting me on the right track, for being rigorous mentors and giving me letters for graduate school. I want to thank my advisors in Pilani, Prof. Surekha Bhanot and Prof. Anu Gupta. It was their teaching and mentorship that instilled the confidence in me to pursue control theory for future research.

My life at CMU couldn't have been enjoyable without the support from my friends and labmates. I want to thank Shivam Vats, Shohin Mukherjee, Ram Natarajan, Tushar Kusnur, Yash Oza and Muhammad Suhail for being generous with your time, lending me your ears and your cozy couch. Thank you to Nick Blauch, Chris Song, Ashwin Khadke, Josh Jaekel, Ratnesh Madaan, Howard Coffin, Nico Zevallos, Humphrey Hu, Arun Srivatsan, Emmi Beuger, Jacky Liang, Navneet Sachdeva, Tab Lee, Aranyak Mishra, Andrea Bajcsy, Ada Taylor and Kiyn Chin. Thank you to my labmates Mike Cheng, Yisha, Andrew Jong, Seth Karten, Russel Schwartz, Brandon Rishi, Nishant Mohanty, Simon Steputtis, Jo Campbell, Dana Hughes, Wenhao Luo and Tianhao Wei. I want to thank my very dear friends away from Pittsburgh but ever so close to my heart: Himanshi Yadav, Pratik Kumar, Rick Goldstein and Shikha Arora.

I also want to thank all my doctors at UPMC, the medical staff at UHS and my therapist for their excellent care and concern. I have never felt being in safer hands. Thanks also to the staff at RI particularly Tim Angert, Suzanne Muth and Jean Harpley.

Last but not the least, I want to express my deep gratitude to my mother and brother, especially my mother for her endless support, her sacrifices, her love and for having my back every step of the way. Her discipline, perseverance, work ethic and commitment to always being grateful are beyond inspiring, and I hope I take a leaf from her book to incorporate those tenets in my life. There is no denying that without her steadfast confidence in me, I wouldn't have reached this milestone. Thank you mumma. And to my brother, I want to thank you for your infinite love, cheers and heart of gold that brighten every day of my life. Thank you mere sohne beeba, I love you.

Contents

List of Tables

List of Figures

I

Introduction, Review and Thesis Objectives

1 Introduction

1.1 Motivation

In the last decade, multi-robot systems (MRSs) have advanced from being researched in labs to being deployed in the real world for solving practical problems in domains such as warehousing [1–3], precision agriculture [4] and environment monitoring [5–8]. The redundancy offered by an aggregated system composed of individual self-sufficient units provides resilience to faults, efficient spatial coverage in an environment and parallelized and distributed acquisition of information. Thus naturally, several research efforts have led to the development of coordination and control algorithms that make multiple robots come together to solve team-level, global tasks using local interaction rules [9–11]. These algorithms share several desirable characteristics including (a) local (*i.e.* individual robots act on information locally available to them), (b) safe (*i.e.* result in collision-free motions amongst robots) and (c) emergent (*i.e.* global properties result from using local interaction rules) [12–15].

These characteristics can be treated as the insider's perspective *i.e.* design principles borne in mind by the control engineer when programming their *own* robots for a given task. Complementary to this is the outsider's perspective *i.e.* the perspective of an external agent watching a group carry out a task by executing motions consistent with these characteristics [16]. As swarms and MRSs become widespread, viewing their behavior from the vantage point of an external observer becomes equally important. For example, if the observer is adversarial, the control engineer must analyze how easily can the observer decrypt the group's task by tracking its members' motions [17]. This would guide the control engineer to design privacy preserving controllers for obfuscating group tactics [18]. On the other hand, if the group itself is adversarial, for example, by posing a threat to a high-value unit, then the observer must predict the group's motion and conscript external agents to defend the unit [19, 20]. Predicting the group's motion requires the observer to infer the underlying group dynamic model. Likewise, planning defensive strategies requires the observer to identify how group agents would interact with external non-group agents.

Understanding and influencing group behavior has applications beyond just the adversarial context. For example, shepherding behaviors, specifically, are one class of flocking behaviors in which one or more external agents (called shepherds) attempt to control the motion of another group of agents (called a flock) by exerting repulsive forces on them [21, 22]. A natural example is a dog guiding a flock of sheep and herding them to a goal position. A successful practical demonstration of robotic herding was achieved in the robot sheepdog project [23–25]. Here, an autonomous wheeled mobile robot (the external agent/shepherd) was used to gather a flock of ducks and maneuvered them to a specified goal position. The robot used a potential-field based model to capture the ducks' dynamics, their interactions amongst one another and their interaction with the robot.

1.2 Challenges

While successful, several existing methods for multiagent behavior inference and behavior shaping make strict assumptions such as homogeneity of group dynamics and a-priori knowledge of group dynamics. Moreover, they rely on heuristically guided control strategies for shaping behavior that lack theoretical guarantees. There are several challenges that are still unaddressed. We describe some of these issues below and build on them to layout the objectives of our work.

For the sake of the argument, we consider an example problem consisting of a group of N agents where each agent is performing goal-directed motion while maintaining a minimum safety distance with every other agent in the rest of the group. The observer is required to ensure that all agents stay away a from protected zone while en-route to their goals. To ensure this, the observer must reverse engineer the goals of the agents to see whether the possibility of breaching exists. If all agents' goals are far from the protected zone, the observer need not intervene. However, if not, the observer will have to program his robots to steer the agents away from the zone. This form of estimation and control is challenging because of the reasons identified below.

1. **Goal inference with unknown safety margins:** For the observer to infer the goal of any agent, the observer must disaggregate that agent's motion into a component arising from its motivation to reach its goal and another arising from its need to maintain a desired safety margin with the remaining agents. This is because the motion that the observer watches comes through the filters of goal-directed behavior and safety combined. However, much like the goal itself, the observer does not know the agent's latent safety margin. Because of this, the observer cannot tell the extent to which safety constraints manifest in the agent's motion. Therefore, before proceeding to inferring goals, the observer must learn these safety margins for each agent to perform this dynamics disaggregation.

2. **Observability/Identifiability:** Related to the challenge mentioned above is the issue of observability *i.e.* identifying when can the latent goal of an agent be inferred from its motion data. This is challenging because failure in identification may stem either from using a poorly tuned estimator or relying on motion data that is inherently deficient in

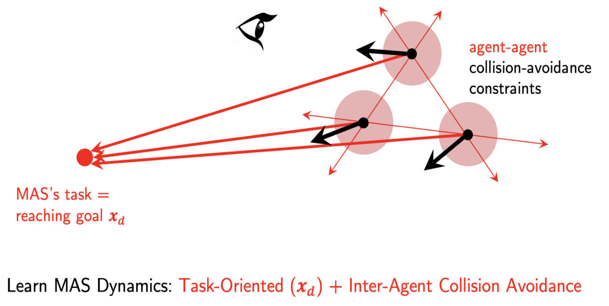
information about the goal. The latter situation can arise when an agent moves solely to ensure safety. Thus, the observer needs to develop estimator-agnostic identifiability criteria that give a yes or no answer to whether a recorded segment of trajectory data is useful for latent parameter extraction at all.

3. **Inference under noise and model-mismatch:** In any practical scenario, observed positions and velocities of the agents will be corrupted by noise. This could originate either on the observer's end possibly from bad localization or on the agents' end when performing state estimation. Moreover, agents may exhibit behavior that deviates from the model hypothesized by the observer. Any estimator that relies on perfect noiseless measurements and strict adherence to a template model would be too brittle and might converge to incorrect parameters. Thus, the observer needs to develop estimation algorithms that at the very least, are forgiving to small amounts of noise and model mismatch.

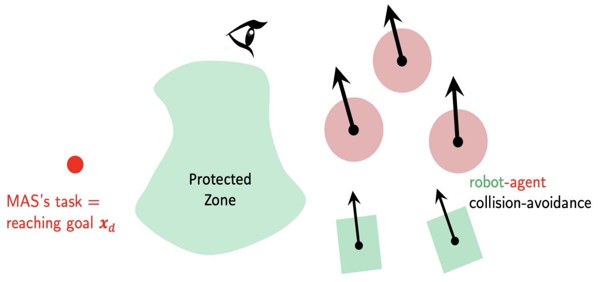
4. **Controllability:** The challenges mentioned above are related to the estimation aspect. Next, we identify hurdles posed by the problem of influencing group behavior using external robots. For this discussion, we assume that the observer a-priori knows the latent parameters of the group agents' underlying dynamic model. Since the external robots cannot directly command the actuators of the group agents, they must rely on their interaction dynamics (collision-avoidance constraints) with the group to influence the group's behavior. This results in a non-collocated control problem, because the actuators *i.e.* the observer's robots are not co-located on the plant *i.e.* the agents of the group [26]. Therefore, akin to the observability criteria, the observer must develop controllability criteria to determine when this problem is feasible. In particular, a formal theory is needed to capture the conflicts that can arise between the agents' self-motivated autonomous dynamics and the behavioral requirements expected of them by the observer.

5. **Scalability:** If there is one robot tasked with influencing a large group, the behavior shaping problem will likely be infeasible, since from the perspective of the robot, the system is severely underactuated. Technically, this will depend on the expected behavior, the group's nominal task and the interaction model between the group and the robot. To make this problem more tractable, the observer can solicit multiple robots. Thus, the observer must decide on how many robots would be sufficient to elicit a given behavior and how to divide labour among the robots so that the said behavior can be realized. In practice, there will be a trade-off between using multiple robots to favor feasibility of behavior shaping and using fewer robots to favor budget efficiency. Thus, the observer, must address these trade-offs.

6. **Control under model uncertainty:** While challenges **4** and **5** above assume that the group agents' model is known a-priori, in practice, the observer will not have access to the true latent parameters of the group's dynamic model. Thus, the observer must rely on nominal estimates of the parameters to synthesize the correct control inputs.


agent-agent
collision-avoidance
constraints
MAS's task =
reaching goal x_d
Learn MAS Dynamics: Task-Oriented (x_d) + Inter-Agent Collision Avoidance

(a) Learning parameters of the latent dynamics of the MAS (red agents)



(b) Controlling the MAS (red agents) through interactions with robots (green)

Figure 1.1: The observer (denoted by the eye) seeks to reverse engineer the agents' parameters (goal and inter-agent safety constraints). It uses this knowledge to prevent breach of the protected zone by orchestrating interactions of the MAS with robots.

However, since these nominal estimates can be far away from the true parameters, the resulting controls may not guarantee that the required behavior emerges out of the group agents. Thus, in addition to the nominal parameter estimates, the observer must incorporate uncertainties in their control synthesis approach. The challenge lies in developing a robust control framework, which, despite the uncertainty in latent parameters, will guarantee that the required behavior will emerge out of the group agents' motions.

1.3 Thesis Statement

Given these challenges, my thesis attempts to formally address the following questions

1. How can the observer disaggregate a group agent's motions into components resulting from its intended task-oriented behavior and its motivation to stay safe with respect to other agents?
2. How can the observer learn a group dynamic model that is robust to measurement noise, suboptimality of team members and a small amount of model mismatch?
3. How can the observer indirectly control the team to elicit a desired behavior by using agents external to the team (*i.e.* robots) and do so in a scalable way?
4. Lastly, how can the observer address question 3 above, when there is uncertainty in the dynamic model of the group agents?

The objectives of our work are (a) to develop algorithms for inferring a robust model of group dynamics by observing individual agents and (b) leveraging this behavioral model for eliciting a desired behavior in the group by orchestrating interactions of non-group members (i.e. robots) with those of the group.

1.4 Outline and Contributions:

An outline of this document is provided below.

- **Chapter** 2 covers related work in the field of system identification for distributed dynamic systems and control of distributed agents via external agents.
- **Chapter** 3 proposes a candidate dynamic model for modeling group behavior and justifies the choice of this model. This chapter outlines the necessary assumptions for both group behavior inference and group control from the perspective of an external observer agent.
- **Chapter** 4 develops identifiability conditions which are necessary for convergence of estimates of the parameters of the group dynamic model to its true parameters. This chapter assumes that the observer has access to perfect noiseless measurements of all agents' positions and velocities, and that the observer knows the safety constraints representing inter-agent interactions. Decentralized online parameter estimators are developed for inferring the task parameters of agents. The work in this chapter appeared in [27].
- **Chapter** 5 proposes an online region-based parameter estimator that learns correct bounds of the task parameters of the agents when the identifiability conditions developed in chapter 4 are violated. Like the last chapter, the assumption is that the observer has access to perfect measurements and knows the safety constraints. The work in this chapter appeared in [28].
- **Chapter** 6 relaxes the noiseless measurements assumption and develops three empirical risk minimization algorithms to learn task parameters when there is (a) noise in the observer's measurements, (b) suboptimality in the decisions taken by the agents and (c) a small amount of model mismatch. The algorithms developed here are based on ideas from inverse optimization literature. The work in this chapter appeared in [29].
- **Chapter** 7 further relaxes the a-priori known safety constraints assumption. Robust MIQP based algorithms are developed to learn parameters of the safety constraints in addition to the task parameters of individual agents. We demonstrate how these algorithms can be used to infer underlying dynamics of pedestrians in an indoor lab environment by using the dateset in [30]. The work in this chapter appeared in [31].

- **Chapter** 8 develops a preliminary approach to solve the one-agent behavior shaping problem using one robot. We describe how to use control barrier functions to elicit two types of shepherding behaviors from an agent using a robot. We identify the issues posed by modeling agents using optimization-based dynamics and take recourse to a simpler group dynamic model that is more amenable for being controlled through interactions with observer's robots.

- **Chapter** 9 develops optimization-based control algorithms to solve the M-agent, N-robot behavior shaping problem using the framework of control barrier functions. The chapter talks about trade-offs between feasibility and optimality *i.e.* using larger number of robots to guarantee feasibility of control synthesis v/s using fewer robots to favor budget efficiency. Accordingly, this chapter presents different algorithms for different preference metrics. In addition to the numerical simulations, the chapter also presents experimental validation of the behavior shaping algorithms on >10 Khepera robots. The work in this chapter appeared in [32] and [33].

- **Chapter** 10 extends the centralized optimization-based algorithms in chapter 9 by incorporating uncertainties in parameter estimates. Using the well-known s-procedure, we rewrite the CBF-QP controllers in the form of CBF-SDPs which guarantee feasibility despite finite-uncertainty in parameters of the agents' underlying dynamic model. Simulations and experimental results are also presented. The work in this chapter appeared in [34]

- **Chapter** 11 concludes the thesis with a summary of this work and outlook for future.

1.5 List of Publications

1. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Parameter Identification for Multirobot Systems Using Optimization-based Controllers.*" 2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS). IEEE, 2021. [Chapter 4].

2. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Feasible Region-Based System Identification Using Duality.*" 2021 European Control Conference (ECC). IEEE, 2021. [Chapter 5].

3. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*System Identification for Safe Controllers using Inverse Optimization.*" IFAC-PapersOnLine 54.20 (2021): 346-353. [Chapter 6].

4. **Jaskaran Grover**, Yiwei Lyu, Wenhao Luo, Changliu Liu, and Katia Sycara. "*Semantically-Aware Pedestrian Intent Prediction With Barrier Functions and Mixed-Integer Quadratic Programming* " 2022 IFAC-CPHS Conference, 2022 ICRA Worksop on Social Robot Navigation. [Chapter 7]

5. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Simultaneously Learning Safety Margins and Task Parameters in Multirobot Systems*". RSS Workshop on Behavior Inference in Multiagent Systems 2021. [Chapter 7]

6. **Jaskaran Grover**, Nishant Mohanty, Wenhao Luo, Changliu Liu, and Katia Sycara. "*Noncooperative Herding With Control Barrier Functions: Theory and Experiments*" 2022 IEEE 61st Conference on Decision and Control (CDC), 80-86. [Chapter 8 - 9].

7. **Jaskaran Grover**, Nishant Mohanty, Changliu Liu, and Katia Sycara. "*Distributed Multirobot Control for Non-Cooperative Herding*" Distributed Autonomous Robotic Systems, 2022. [Chapter 8 - 9].

8. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Control Barrier Functions-based Semi-Definite Programs (CBF-SDPs): Robust Safe Control For Dynamic Systems with Relative Degree Two Safety Indices*" arxiv preprint. [Chapter 10].

9. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*The Before, During and After of Multirobot Deadlock.*" International Journal of Robotics Research 2022.

10. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Why Does Symmetry Cause Deadlock?*" IFAC-PapersOnLine 53.2 (2020): 9746-9753.

11. **Jaskaran Grover**, Changliu Liu, and Katia Sycara. "*Deadlock Analysis and Resolution for Multi-Robot Systems.*" International Workshop on the Algorithmic Foundations of Robotics. Springer, Cham, 2020.

2 Related Work

2.1 Multiagent Behavior Inference

The problem of learning models for groups of interacting dynamic agents lies at the intersection of system identification and modeling of distributed dynamical systems. A vigorous body of work is emerging from the controls, robotics and machine learning communities focused on analyzing models of flocks, swarms, and similar distributed dynamic systems. In the realm of learning based approaches, [35] show how inverse reinforcement learning (IRL) can be extended to homogeneous large-scale problems. Their framework is able to produce meaningful local reward models that can replicate the observed global system dynamics in swarms. Similarly, [36, 37] use IRL to model the reward function of individual agents of biological swarms such as flocks of pigeons. They enforce priors by modeling the rewards with basis functions containing terms capturing cohesion, repulsions, following a leader etc. [38] use IRL to learn swarming behaviors observed in fish schools. They model all fish with an identical neural network to learn a common single policy to capture the emergent behavioral patterns. [39, 40] build on the learning-from-demonstration paradigm and develop a graph neural network architecture which can imitate an observed swarm and is able to extract rules governing group behavior. In the realm of control-theoretic approaches, some recent works have started to address the identifiability issue in reverse-engineering swarm dynamics. For example, [16, 17] propose empirical metrics of observability for learning parameters of flocking dynamics that were developed in [41].

These works focus on inferring group dynamics observed in swarms which tend to be homogeneous. Inference algorithms have also been developed for heterogeneous multiagent systems. [42] presents a model-based approach for understanding movement and interaction dynamics amongst cows. Their algorithms learn characteristic constants of attraction/repulsion forces that are unique for each cow. [43] train a deep network to learn goal-directed and collision avoidance behavior of robots in a multirobot system. Their motivation is to incorporate this learned model as a heuristic to speed up MILP-based multirobot motion planning. [44] shows how high-level behaviors can be inferred from an expert multirobot system and use them to train an untrained team of robots to execute a mission.

Several inference algorithms model the behavior of a multiagent system as resulting from a dynamic non-cooperative game [45–47]. The demonstrations are assumed to constitute an open-loop Nash equilibrium solution of the game. The behavior inference problem in this context reduces to inferring the unique long term cost-functions of each player subject to the dynamics of the plant they act on. [48] developed batch and online algorithms following single-agent inverse optimal control (IOC) [49, 50] for inferring cost functions of individual players in a game. [51, 52] develop robust algorithms for inferring player objectives from noisy demonstrations of player trajectories.

Several of these game theoretic approaches assume that there are no constraints on the plant state or on player's control inputs. The inference algorithms developed in these works operate on player trajectories that are Nash equilibria solutions for an unconstrained dynamic game. However, there is an issue with this approach. To illustrate, suppose a player is following a trajectory that is the open-loop Nash equilibrium solution, and suddenly an obstacle previously unaccounted for in the game, needs to be avoided. Since the player will try to locally adjust its plan to satisfy this constraint, the resulting adjusted trajectory would no longer represent the Nash solution. Thus, using this perturbed demonstration may not guarantee correct inference. This prevents the direct application of existing IOC based approaches for inferring long-term rewards when there are constraints on plant states. Recent work on reactive optimization-based controllers for multirobot systems has shown how a nominal plan can be minimally adjusted to satisfy safety constraints [53–55]. *This is the model that we adopt in our work for performing inference because this model captures both the long-term intentions of an agent and the local adjustments it makes to satisfy constraints such as safety with respect to the other agents.*

Our work develops algorithms to learn the long term intentions/task parameters of individual agents from observed demonstrations in the face of input constraints. To perform this inference, we first simplify our problem by assuming known constraints and learn the task parameters of all agents with this assumption. In chapter 4 we analyze situations in which the task parameters can be correctly inferred using the concept of persistency of excitation. We develop decentralized online estimators for inferring these parameters. Further improvements are presented in chapters 5-6. Subsequently, in chapter 7, we relax the known constraints assumption as well and develop batch algorithms to learn robust estimates of the parameters of costs and constraints simultaneously.

2.2 Multiagent Behavior Shaping

The problem of controlling the behavior of a group of agents using robots has not received as much attention as the behavior inference problem. There is a wide range of domains where controlling a multi-agent team via external robots finds practical applications. These include the use of robots to herd sheep [56, 57]; crowd control [58]; protecting aircraft from bird strikes [59, 60] and defense against adversarial swarms [19]. [21, 61] used the term shepherding behaviors to capture herding, patrolling and covering behaviors. They gave simulation results to show how this can be achieved using one or more shepherds. [62] gives a review of shepherding as a bio-inspired swarm-robotics guidance approach. Another term commonly used to describe these behaviors is noncooperative herding because the flock agents are neither adversarial nor fully cooperative with the shepherd [22, 63–67]. These works exploit the interaction dynamics between the flock and robots to shape the flock's behavior. While successful, one issue common in these approaches is that they fail to consider the self-motivated dynamics of the flock agents. As a result, the flock agents' motions are solely driven by repulsions from the robots. Secondly, the flock agents consider these repulsions perpetually regardless of how far the robots are located relative to the flock. Thirdly, not all of these approaches come with rigorous theoretical guarantees. In chapter 8, we show a formalism to address three issues simultaneously by modeling agents using optimization-based dynamics and using control barrier functions to pose the behavior shaping problem. To demonstrate our approach, we show simulation results that achieve successful herding and defending behaviors by using one robot against one agent. In subsequent chapters, we extend this to the multirobot-multiagent behavior shaping with both experimental and simulation results.

3 Background and Problem Formulation

In this chapter, we describe the representation of the dynamics for the multiagent system that we will adhere to, in the rest of the thesis. We formally pose the multiagent behavior inference problem and the multiagent behavior shaping problem bearing this representation of dynamics in mind. The outline of this chapter is as follows: section 3.1.1 provides the relevant definitions/notations of control barrier functions which we will use (a) for modeling collision avoidance constraints amongst agents and (b) for posing the group behavior shaping requirements. Section 3.2 gives a formal statement of the multiagent behavior inference problem. Finally, section 3.3 gives a formal statement of the multiagent behavior shaping problem using external robots.

3.1 Background

3.1.1 Control Barrier Functions

We recall the fundamentals of control barrier functions that are used to synthesize a safe controller for a dynamical system. We refer the reader to [55] for a comprehensive review on this subject. Consider a dynamical system with state $\boldsymbol{x} \in \mathbb{R}^n$ and control-affine dynamics

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}. \tag{3.1}$$

Here $\boldsymbol{u} \in \mathbb{R}^m$ is the control-input to the system[1]. We assume that functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are Lipschitz continuous. Unlike stability which involves steering the state of the system to a point, safety can be framed in the context of enforcing invariance of a set, *i.e.*, not leaving a certain predefined set. We denote this set as $\mathcal{C}$ and define it as the superlevel set of a

[1]Assuming that the system is control-affine is not restrictive, because any non-control affine system can be converted to a control affine form using dynamic extension.

continuously differentiable function $h : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}$

$$\begin{aligned} \mathcal{C} &\coloneqq \{\boldsymbol{x} \in \mathbb{R}^n | h(\boldsymbol{x}) \geq 0\} \\ \partial\mathcal{C} &\coloneqq \{\boldsymbol{x} \in \mathbb{R}^n | h(\boldsymbol{x}) = 0\} \\ \text{Int } \mathcal{C} &\coloneqq \{\boldsymbol{x} \in \mathbb{R}^n | h(\boldsymbol{x}) > 0\} \end{aligned} \tag{3.2}$$

We refer to $\mathcal{C}$ as the *safe set*. Let $\boldsymbol{u} = \boldsymbol{k}(\boldsymbol{x})$ be a Lipschitz continuous feedback controller such that the closed-loop dynamics of (3.1) are

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_{cl}(\boldsymbol{x}) \coloneqq \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{k}(\boldsymbol{x}) \tag{3.3}$$

Because $\boldsymbol{f}_{cl}(\boldsymbol{x})$ is locally Lipschitz, for any initial condition $\boldsymbol{x}_0 \in D$, there exists a maximum interval of existence $I(\boldsymbol{x}_0) = [0, \tau_{max})$ such that $\boldsymbol{x}(t)$ is the unique solution to (3.3) on $I(\boldsymbol{x}_0)$. This allows us to define safety,

Definition 1. *The set $\mathcal{C}$ is **forward invariant** if for every $\boldsymbol{x}_0 \in \mathcal{C}$, $\boldsymbol{x}(t) \in \mathcal{C}$ for $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and all $t \in I(\boldsymbol{x}_0)$. System (3.3) is **safe** with respect to $\mathcal{C}$ if $\mathcal{C}$ is **forward invariant**.*

Definition 2. *Let $\mathcal{C} \subset D \subset \mathbb{R}^n$ be the zero-level superset of a continuously differentiable $h : D \longrightarrow \mathbb{R}$. Then $h(\cdot)$ is a control barrier function (CBF) if there exists a class κ_∞ function $\alpha(\cdot)$ such that for the control system (3.1), the following holds*

$$\begin{aligned} \dot{h}(\boldsymbol{x}, \boldsymbol{u}) &\geq -\alpha(\boldsymbol{x}), \forall \boldsymbol{x} \in D \\ \longleftarrow \sup_{\boldsymbol{u} \in \mathbb{R}^m} [L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x})\boldsymbol{u}] &\geq -\alpha(\boldsymbol{x}), \forall \boldsymbol{x} \in D \end{aligned} \tag{3.4}$$

Given an $h(\cdot)$, we can characterize the set of all controls that render $\mathcal{C}$ safe per Def. 1:

$$\begin{aligned} K_{cbf}(\boldsymbol{x}) &\coloneqq \{\boldsymbol{u} \in \mathbb{R}^m | L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x})\boldsymbol{u} \geq -\alpha(\boldsymbol{x})\} \\ &\equiv \{\boldsymbol{u} \in \mathbb{R}^m | A(\boldsymbol{x})\boldsymbol{u} \leq b(\boldsymbol{x})\}, \end{aligned} \tag{3.5}$$

where $A(\boldsymbol{x}) \coloneqq -L_{\boldsymbol{g}} h(\boldsymbol{x})$ and $b(\boldsymbol{x}) \coloneqq \alpha(\boldsymbol{x}) + L_{\boldsymbol{f}} h(\boldsymbol{x})$. Finally, we come to the main result:

Theorem 1. *[55] Let $\mathcal{C}$ be defined as the superlevel set of a continuously differentiable function $h : \mathbb{R}^n \longrightarrow R$. If h is a CBF on D and $\frac{dh}{d\boldsymbol{x}} \neq \boldsymbol{0}$ $\forall \boldsymbol{x} \in \partial\mathcal{C}$, then any Lipschitz continuous controller $\boldsymbol{u} \in K_{cbf}(\boldsymbol{x})$ renders $\mathcal{C}$ forward invariant.*

For any given control objective for (3.1), we assume that a reference controller $\hat{\boldsymbol{u}}(\boldsymbol{x})$ exists. This may be generated by a high-level planner or by applying linear control synthesis techniques to a linearization of (3.1) at an operating point $\boldsymbol{x}^o$. This controller may not be safe with respect to our definition in (3.5). Thus, in order to synthesize a safe controller, a commonly used strategy is to project the reference controller $\hat{\boldsymbol{u}}(\boldsymbol{x})$ to the set $K_{cbf}(\boldsymbol{x})$ locally at every $\boldsymbol{x} \in D$. Since $K_{cbf}(\boldsymbol{x})$ in (3.5) defines a linear half-space in $\mathbb{R}^m$ when evaluated at a given $\boldsymbol{x} \in D$, the projection of $\hat{\boldsymbol{u}}(\boldsymbol{x})$ on $K_{cbf}(\boldsymbol{x})$ can be done by posing the following quadratic program

$$\begin{aligned} \boldsymbol{u}^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & L_{\boldsymbol{f}} h(\boldsymbol{x}) + L_{\boldsymbol{g}} h(\boldsymbol{x})\boldsymbol{u} \geq -\alpha(\boldsymbol{x}). \end{aligned} \tag{3.6}$$

This optimization problem is commonly referred to as the CBF-QP and results in a control that is minimally invasive to $\hat{\boldsymbol{u}}(\boldsymbol{x})$ and is safe by construction. In the next section, we describe how to model collision-free motions between any pair of agents using this optimization.

3.1.2 Collision-Free Multiagent System Dynamics

We describe the template representation of the dynamics of each agent in the multiagent system that we will use in this thesis. We model all agents as single integrators that are velocity-controlled, although the ensuing theory is applicable for double integrator agent dynamics as well. Suppose there are $M+1$ agents in our system. Denote the state of agent i by $\boldsymbol{x}_i \in \mathbb{R}^2 \; \forall i \in \{1, 2, \cdots, M+1\}$ which represents its position and denote its velocity as $\boldsymbol{u}_i \in \mathbb{R}^2$, which acts as the control input. The dynamics of this agent are

$$\dot{\boldsymbol{x}}_i = \boldsymbol{u}_i, \qquad \forall i \in \{1, 2, \cdots, M+1\}. \tag{3.7}$$

To keep the discussion simple, we focus on one agent *i.e.* the ego agent, so we will omit subscript i. Let the state of this agent be $\boldsymbol{x}$. This agent has a primary task to accomplish and we assume that it can accomplish this task using a reference control $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$. Assume the following representation of the reference control

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x}). \tag{3.8}$$

Here $\boldsymbol{\theta}$ are the parameters capturing the task of that agent and are in general different for each agent. $C(\boldsymbol{x}), d(\boldsymbol{x})$ are some task oriented basis functions. This representation is general enough to capture some elementary tasks relevant to this thesis. We give two examples here

1. *Reaching a goal position:* Suppose that the primary task of the agent is to reach a goal position $\boldsymbol{x}_d$ at an exponentially fast speed governed by a gain k_p. A candidate control that can accomplish this task is given by $\hat{\boldsymbol{u}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$. We rewrite this control in a form akin to (3.8) as follows

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = \underbrace{\begin{bmatrix} -\boldsymbol{x}_x & 1 & 0 \\ -\boldsymbol{x}_y & 0 & 1 \end{bmatrix}}_{C(\boldsymbol{x})} \underbrace{\begin{bmatrix} k_p \\ k_p \boldsymbol{x}_{d_x} \\ k_p \boldsymbol{x}_{d_y} \end{bmatrix}}_{\boldsymbol{\theta}} + \underbrace{\boldsymbol{0}}_{\boldsymbol{d}(\boldsymbol{x})}. \tag{3.9}$$

2. *Maintaining a constant velocity*: Suppose the task of the agent is to maintain a constant velocity $\boldsymbol{v}_d$, then we can select $\hat{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{v}_d$. This control expressed akin to (3.8) is

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = \underbrace{I}_{C(\boldsymbol{x})} \underbrace{\boldsymbol{v}_{\boldsymbol{d}}}_{\boldsymbol{\theta}} + \underbrace{\boldsymbol{0}}_{\boldsymbol{d}(\boldsymbol{x})} \tag{3.10}$$

In addition to performing its task, this agent is also driven by the need to stay collision free with respect to the other agents. We assume that in the multiagent system, all agents cooperate to achieve collision avoidance amongst one another while performing their respective tasks. Since the reference controller of the ego agent $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ is purely task-oriented and is not derived considering safety, it cannot be relied upon to ensure collision-avoidance of the ego agent with the rest. We derive constraints that must be satisfied by a candidate control to ensure collision free motion of the ego agent. Let the other agents be located at positions $\boldsymbol{x}_j^o \ \forall j \in \{1, 2, \cdots, M\}$. The ego agent and agent j are collision free if their positions $(\boldsymbol{x}, \boldsymbol{x}_j^o)$ satisfy $\|\Delta \boldsymbol{x}_j\|^2 \geq D_s^2$ where $\Delta \boldsymbol{x}_j \coloneqq \boldsymbol{x} - \boldsymbol{x}_j^o$ and D_s is a desired safety margin. Using this requirement, we pose a pairwise safety index $h : \mathbb{R}^2 \times \mathbb{R}^2 \longrightarrow \mathbb{R}$ and define it as

$$h_j(\boldsymbol{x}, \boldsymbol{x}_j^o) \coloneqq \left\|\boldsymbol{x} - \boldsymbol{x}_j^o\right\|^2 - {D_s}^2. \tag{3.11}$$

The ego agent is collision free with respect to all other agents iff $h_j(\boldsymbol{x}, \boldsymbol{x}_j) \geq 0 \ \forall j \in \{1, 2, \cdots, M\}$. We can treat these safety indices as CBFs. To ensure that all h_j continue to stay non-negative for all times, the agent enforces the following constraints

$$\frac{dh_j}{dt} \geq -\gamma h_j, \forall j \in \{1, 2, \cdots, M\}. \tag{3.12}$$

These constraints can be reformulated as constraints on the agent's velocity $\boldsymbol{u}$ following (3.5). These are given by $A\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)\boldsymbol{u} \leq \boldsymbol{b}_{\gamma, D_s}\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)$. Here $A\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big) \in \mathbb{R}^{M\times 2}$, $\boldsymbol{b}\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big) \in \mathbb{R}^M$ are defined such that the j^{th} row of A is $\boldsymbol{a}_j^T$ and the j^{th} entry of $\boldsymbol{b}_{\gamma, D_s}$ is b_j:

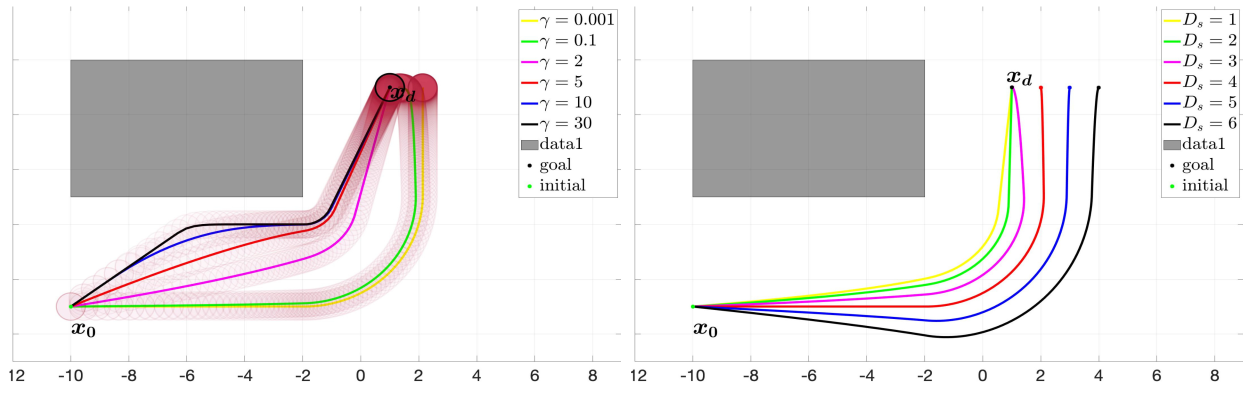
$$\boldsymbol{a}_j^T \coloneqq -\Delta \boldsymbol{x}_j^T = -(\boldsymbol{x} - \boldsymbol{x}_j^o)^T, \qquad b_j \coloneqq \frac{\gamma}{4}(\|\Delta \boldsymbol{x}_j\|^2 - {D_s}^2) \ \forall j \in \{1, 2, \ldots, M\} \tag{3.13}$$

To combine the collision avoidance requirement with the task completion objective, the agent solves a QP that computes a controller closest to its reference $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and satisfies M collision-avoidance constraints as follows

$$\begin{aligned} \boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)\boldsymbol{u} \leq \boldsymbol{b}_{\gamma, D_s}\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big). \end{aligned} \tag{3.14}$$

Going forward, we will assume that each agent in the multiagent system solves this QP at every time instant to determine its optimal control $\boldsymbol{u}^*$. Additionally, since agents do not come to an abrupt halt, we will also assume that a solution to this QP always exists *i.e.* the agents' velocities are always feasible. This control ensures collision avoidance while encouraging task completion. Aside from its position $\boldsymbol{x}$, the ego agent's control $\boldsymbol{u}^*$ depends on task parameters $\boldsymbol{\theta}$. This is implicitly encoded through the cost function of (3.14) (recall $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$). While all agents follow (3.14) to synthesize their control inputs, it is the task+constraint parameters *i.e.* $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \gamma, D_s)$ that distinguish one agent from another. To highlight the dependence of $\boldsymbol{u}^*$ on these parameters, we denote the control as $\boldsymbol{u}^*_{\boldsymbol{\Theta}}(\boldsymbol{x})$.

Fig. 3.1 shows example trajectories computed by this controller for an agent with one collision avoidance constraint relative to a rectangular obstacle. The reference control is $\hat{\boldsymbol{u}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$. To get an intuitive understanding of the parameter γ, we conducted several simulations of (3.14) with increasing values of γ keeping $D_s, \boldsymbol{x}_0, \boldsymbol{x}_d$ and k_p fixed. The resulting trajectories are shown in fig. 3.1(a). From this figure, it is evident that a large value of γ makes the trajectory less conservative and makes it follow the reference controller more closely while still maintaining D_s relative to the obstacle.



(a) Effect of varying γ (b) Effect of varying D_s

Figure 3.1: Sample trajectories produced by (3.14) with one rectangular obstacle as a constraint. These trajectories are computed for different values of γ and D_s. (a) For a fixed D_s, increasing γ makes the trajectory less conservative and makes it follow the reference controller more closely. (b) For a fixed γ, increasing D_s increases the distance from the obstacle as is expected.

3.1.3 Why Is This Representation Of Dynamics A Good Modeling Choice?

There are several reasons in favor of this representation of dynamics:

1. The nominal control $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ represents the agent's primary controller should there be no other agents in the system. This represents the self-motivated dynamics of that agent. Thus having an optimization model the dynamics automatically encourages the agent's intent to follow this controller as much as possible while also ensuring that safety is respected in the presence of other agents.

2. For a given state of the ego agent and other agents, some constraints will 'matter' while some will not (this notion will be formalized in the next chapter). This is equivalent

to saying that the agents associated with those constraints 'matter' to the ego agent while others don't. By matter, we mean whether or not they influence the ego agent's dynamics. Our prior work showed a one-to-one mapping between what it means for an agent to matter and the distance of that agent relative to the ego agent [68]. A direct implication of this result then, is that this model then automatically filters out agents far away from the ego agent that don't interfere with the ego agent's task, thus only neighboring agents end up influencing the ego's dynamics. Thus, this model captures the local interactions automatically.

3. Feasibility of constraints ensures strict collision avoidance, thus for demonstrations that never undergo collisions, it is reasonable to assume these demonstrations might be originating from constraints of the form in (3.14) on the agents' velocities.

3.2 Behavior Inference Problem Formulation

Having settled on the dynamic model, we now state the behavior inference problem for the multiagent system. In informal terms, an external observer wishes to understand the preferred task controller + safety constraints of each agent in the system by tracking its positions and velocities over some time. Before stating the inference problem, we state all assumptions on the observer's knowledge.

Assumption 1. *The observer knows that the ego agent's cost function is of the form* $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$

Assumption 2. *The observer knows the task functions* $C(\boldsymbol{x}), d(\boldsymbol{x})$ *of* $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$.

Assumption 3. *The observer knows the form of safety constraints* $A(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M)$ *and* $\boldsymbol{b}_{\gamma, D_s}(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M)$.

Problem 1 (**Multiagent Behavior Inference**). *The observer's problem is to infer parameters* $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \gamma, D_s)$ *for each agent by monitoring its position* $\boldsymbol{x}(t)$ *and the positions of other agents i.e.* $\boldsymbol{x}_j^o(t)\ \forall j \in \{1, 2, \cdots, M\}$ *over some finite-time.*

Since there are $M+1$ agents in the system, the observer will run $M+1$ parallel estimators to identify $\boldsymbol{\Theta}_i = (\boldsymbol{\theta}_i, \gamma_i, D_s^i)$ for $\forall i \in \{1, 2, \cdots, M\}$. While $\boldsymbol{\theta}$ represents information about the task-oriented part of the dynamics, (γ, D_s), on the other hand, contain information about safety margins and conservativeness *i.e.* they represent information about safety constraints/ the interaction part of the dynamics.

Let us point out why this inference is non-trivial. The dependence of $\boldsymbol{u}^*_{\boldsymbol{\Theta}}(\boldsymbol{x})$ on $\boldsymbol{\theta}$ comes through the filter of constraints and the objective function in (3.14). In a situation where $\boldsymbol{u}^*_{\boldsymbol{\Theta}}(\boldsymbol{x})$ is solely determined by constraints, $\boldsymbol{\theta}$ will not be inferable because it is the cost function that depends on $\boldsymbol{\theta}$ not the constraints. Naturally, the observer may not have means to deduce the extent of the dependence of $\boldsymbol{u}^*_{\boldsymbol{\Theta}}(\boldsymbol{x})$ on the constraints, and this is precisely what makes this problem challenging.

3.3 Behavior Shaping Problem Formulation

In the behavior shaping problem, the observer solicits N robots to elicit a desired behavior from the multiagent system under observation. Since this is an indirect control problem, the observer requires a model of interaction between the multiagent system and the robots to analyze how information flows from her robots to the agents and in turn modifies their dynamics. In this thesis, we interpret interaction as the collision-avoidance constraints between agents. Like the collision avoidance constraints that exist amongst group agents themselves, we assume that similar constraints model collision-avoidance of the multiagent system with the observer's robots. To distinguish between the agent of the group and the observer's robots, we will use subscript A and R respectively.

Suppose there M agents in the group collectively denoted as $\mathcal{A} \coloneqq \{1, 2, \cdots, M\}$. Let the position of agent i be $\boldsymbol{x}_{A_i} \in \mathbb{R}^2 \ \forall i \in \mathcal{A}$. Likewise, assume there are N robots collectively denoted as $\mathcal{R} \coloneqq \{1, 2, \cdots, N\}$. Let robot k be located at $\boldsymbol{x}_{R_k} \in \mathbb{R}^2 \ \forall k \in \mathcal{R}$. Consistent with our modeling paradigm so far, we choose to model the dynamics of the i^{th} agent A_i using a reactive one-step optimization

$$\begin{aligned}
\dot{\boldsymbol{x}}_{A_i} = \arg\min_{\boldsymbol{u}} \quad & \left\| \boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}_{A_i}}(\boldsymbol{x}_{A_i}) \right\|^2 \\
\text{subject to} \quad & A\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{A_j}\}_{j \in \mathcal{A} \backslash i}\Big)\boldsymbol{u} \leq \boldsymbol{b}_{\gamma_i, D_s^i}\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{A_j}\}_{j \in \mathcal{A} \backslash i}\Big) \text{ constraints with agents} \\
& A\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{R_k}\}_{k \in \mathcal{R}}\Big)\boldsymbol{u} \leq \boldsymbol{b}_{\tilde{\gamma}_i, \tilde{D}_s^i}\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{R_k}\}_{k \in \mathcal{R}}\Big) \text{ constraints with robots} \\
\coloneqq \quad & \boldsymbol{f}_i\Big(\{\boldsymbol{x}_{A_i}\}_{i \in \mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k \in \mathcal{R}}\Big).
\end{aligned} \tag{3.15}$$

We allow a different set of parameters $\tilde{\gamma}, \tilde{D}_s$ to distinguish the interaction amongst agents of the group from their interaction with the observer's robots. The cost function penalizes deviation from a reference task-based control $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}_{A_i}}(\boldsymbol{x}_{A_i})$, which is the control that A_i would follow should all its constraints be inactive. Thus, it represents the autonomous self-motivated dynamics of A_i. In totality, its the dynamics depend on parameters $(\boldsymbol{\theta}_{A_i}, \gamma_i, D_s^i, \tilde{\gamma}_i, \tilde{D}_s^i)$. We collectively denote all these parameters as $\boldsymbol{\Theta}_{A_i}$, and aggregate parameters of all agents as

$$\begin{aligned}
\boldsymbol{\Theta} &= (\boldsymbol{\Theta}_{A_1}, \cdots, \boldsymbol{\Theta}_{A_M}) \text{ where,} \\
\boldsymbol{\Theta}_{A_i} &\coloneqq (\boldsymbol{\theta}_{A_i}, \gamma_i, D_s^i, \tilde{\gamma}_i, \tilde{D}_s^i).
\end{aligned} \tag{3.16}$$

Further, we use superscript $\boldsymbol{\Theta}_{A_i}$ on $\boldsymbol{f}_i$ to emphasize its dependence on $\boldsymbol{\Theta}_{A_i}$. As for each robot, we assume it is velocity-constrolled with dynamics

$$\dot{\boldsymbol{x}}_{R_k} = \boldsymbol{u}_{R_k} \quad \forall k \in \mathcal{R}. \tag{3.17}$$

We can view (3.15) and (3.17) as a joint system, with its state as the combined states of all agents and robots, and the velocities of the robots as the control input to the system

$$\begin{pmatrix} \dot{\boldsymbol{x}}_{A_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{A_M} \\ \dot{\boldsymbol{x}}_{R_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{R_N} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1^{\boldsymbol{\Theta}_{A_1}}\left(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\right) \\ \vdots \\ \boldsymbol{f}_M^{\boldsymbol{\Theta}_{A_M}}\left(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\right) \\ \boldsymbol{u}_{R_1} \\ \vdots \\ \boldsymbol{u}_{R_N} \end{pmatrix}. \tag{3.18}$$

Given this joint model, we now pose the problem of eliciting a desired behavior from the agents. We denote the observer's behavioral requirement using a function $y_i : \mathbb{R}^2 \longrightarrow \mathbb{R}$ that corresponds to the behavioral requirement expected from agent i. Define a set $\mathcal{Y} \subset \mathbb{R}^{2M}$ as

$$\mathcal{Y} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^{2M} | y_i(\boldsymbol{x}_{A_i}) > 0 \ \forall i \in \mathcal{A}\} \tag{3.19}$$

Now depending on whether the observer knows the parameters of the agents or not, there can be two variants of the behavior shaping problem. We state both these variants below. Chapters 8 and 9 address the behavior shaping problem with known model. Chapter 10 develops a robust control technique to address the behavior shaping problem with an uncertain model.

Problem 2 (**Multiagent Behavior Shaping With Known Model**). *Assuming the true parameters $\boldsymbol{\Theta}$ are known to the observer, if the initial agent positions $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \in \mathcal{Y}$, the multiagent behavior shaping problem requires the observer to find robot controls $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \in \mathcal{Y} \ \forall t > 0$. However, if the positions of the agents $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \notin \mathcal{Y}$, then the observer's problem is to find $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \rightsquigarrow \mathcal{Y}$ in finite time.*

Problem 3 (**Multiagent Behavior Shaping With Uncertain Model**). *Assume that observer has estimates of the parameters given by $\hat{\boldsymbol{\Theta}}$ and an upper bound on estimation error $\left\|\hat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}\right\| \leq \eta$. If the initial agent positions $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \in \mathcal{Y}$, the multiagent behavior shaping problem requires the observer to find robot controls $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \in \mathcal{Y} \ \forall t > 0$. However, if the agent positions $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \notin \mathcal{Y}$, then the observer's problem is to find $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \rightsquigarrow \mathcal{Y}$ in finite time.*

II

Multiagent System Identification

4 Identifiability Criteria and Convergent Estimators

4.1 Introduction

In this chapter, we investigate the well-posedness of the multiagent behavior inference problem as stated in Def. 1 in chapter 3. For now, we assume that the constraint parameters γ, D_s are known to the observer and relax this later in chapter 7. We want to understand when are the positions of individual agents "rich enough" so that they suffice to reveal the parameters of their underlying tasks (*i.e.* $\boldsymbol{\theta}$) to the observer, given that the dynamics of agents are given by (3.14). Inference with such controllers is challenging because presence of an optimization in the closed-loop dynamics of an agent makes its positions/velocities depend *implicitly* on task parameters. On the other hand, it is an explicit relation that is practical for deriving identifiability criteria. We will describe how to derive such a relation, and develop necessary conditions which when satisfied would ensure correct inference of the task. The theory is for inferring an arbitrary task and specialized subsequently for the goal-reaching task, for which, the observer's problem will be to infer the goals and controller gains of each agent.

The outline is as follows: in section 4.2, we provide a brief background on parameter identification. Using the *persistency of excitation* [69] analysis, we derive a novel necessary condition for successful identification in lemma 1. In section 4.3, we recall the the multiagent behavior model (from (3.14)) and the inference problem for this system. The main contributions begin from section 4.4 where we derive the KKT conditions of the control-synthesis optimization. By focusing on the set of active interactions (*i.e.* active constraints) of an agent with the rest, we pose an equality-constrained optimization (EQP) which is the first step for deriving a relation between parameters and its dynamics. In section 4.5, we classify each agent's dynamics based on the number of constraints in this EQP, and linear independence relations amongst these constraints. Taking the SVD of these constraints allows us to derive the explicit relation between the parameters and dynamics of each agent that we wanted. Finally, using these relations in conjunction with the *persistency of excitation* requirement and the result derived in lemma 1, we provide the main necessary conditions for successful task identification of the multiagent system (theorems (2-6)). The message that these theorems convey is that as the number of agents that an ego agent interacts with increases, estimation of ego's parameters becomes difficult. This confirms our intuition, because with more interactions, the ego agent's motion (that the observer measures) is expended in satisfying collision avoidance constraints which it achieves by sacrificing task performance. We demonstrate this numerically in section 4.6, where we use an adaptive observer and a UKF to infer all agent's goals using their positions under various geometric settings. We conclude in 4.7 by summarizing and provide directions for future work.

4.2 Observer based Parameter Identification

While there exist several parameter estimation algorithms that the observer can leverage (such as RLS [70], UKF [71]), we focus on adaptive observer-based methods borrowing ideas from [72, 73] because they provide conditions under which convergence to the true parameters is achieved. Consider a nonlinear parameter-affine system as follows

$$\dot{\boldsymbol{x}} = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x}), \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the measurable state, $\boldsymbol{\theta} \in \mathbb{R}^p$ is the unknown parameter and $G(\boldsymbol{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^{n\times p}$, $f(\boldsymbol{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ are known functions. In our context, $\boldsymbol{x}(t)$ will correspond to the position of the ego agent under observation and $\boldsymbol{\theta}$ denotes the task parameters of its controller that we wish to infer. We assume that the observer runs several parallel estimators synchronously, one for estimating the parameters of each agent, so the focus here is on the ego agent. The observer's problem is to design an estimation law $\dot{\hat{\boldsymbol{\theta}}} = \psi(\hat{\boldsymbol{\theta}}, \boldsymbol{x})$ that guarantees convergence of $\hat{\boldsymbol{\theta}} \longrightarrow \boldsymbol{\theta}$ by using $\boldsymbol{x}(t)$ over some $t \in [0, T]$ where T is large enough. Consider a state predictor defined analogously to (4.1)

$$\dot{\hat{\boldsymbol{x}}} = G(\boldsymbol{x})\boldsymbol{\theta}^0 + f(\boldsymbol{x}) + k_w(\boldsymbol{x} - \hat{\boldsymbol{x}}),\ \hat{\boldsymbol{x}}(0) = \boldsymbol{x}(\boldsymbol{0}), \tag{4.2}$$

where $\boldsymbol{\theta}^0 \in \mathbb{R}^p$ is a nominal initial estimate of $\boldsymbol{\theta}$ and $k_w > 0$. Define an auxiliary variable $\boldsymbol{\eta} \in \mathbb{R}^n$ as follows

$$\boldsymbol{\eta} = \boldsymbol{x} - \hat{\boldsymbol{x}} - W(\boldsymbol{\theta} - \boldsymbol{\theta}^0) \tag{4.3}$$

where $W \in \mathbb{R}^{n\times p}$ is generated according to

$$\dot{W} = -k_w W + G(\boldsymbol{x}),\ W(0) = 0. \tag{4.4}$$

Here W is a low-pass filtered version of $G(\boldsymbol{x})$. While $\boldsymbol{\eta}$ as defined in (4.3) is not measurable because it depends on $\boldsymbol{\theta}$ that is unknown, defining W as in (4.4) lets us generate $\boldsymbol{\eta}$ using

$$\dot{\boldsymbol{\eta}} = -k_w\boldsymbol{\eta},\ \boldsymbol{\eta}(0) = \boldsymbol{x}(0) - \hat{\boldsymbol{x}}(0). \tag{4.5}$$

Based on (4.1)-(4.5), let $Q \in \mathbb{R}^{p\times p}$ and $C \in \mathbb{R}^p$ be generated according to the following dynamics

$$\dot{Q} = W^T W,\ Q(0) = 0^{p\times p}, \tag{4.6}$$

$$\dot{C} = W^T(W\boldsymbol{\theta}^{\mathbf{0}} + \boldsymbol{x} - \hat{\boldsymbol{x}} - \boldsymbol{\eta}), C(0) = 0^{p\times 1}, \tag{4.7}$$

and let t_c be the time at which $Q(t_c) \succ 0$, then the following parameter update law

$$\dot{\hat{\boldsymbol{\theta}}} = \Gamma(C - Q\hat{\boldsymbol{\theta}}), \ \hat{\boldsymbol{\theta}}(0) = \boldsymbol{\theta}^0, \tag{4.8}$$

for $\Gamma \succ 0$ guarantees that $\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|$ is non-increasing for $0 \leq t \leq t_c$ and exponentially converges to 0 for $t > t_c$. Thus, as long as there exists t_c at which $Q(t_c) \succ 0$, convergence of the estimate $\hat{\boldsymbol{\theta}}$ to the true parameter $\boldsymbol{\theta}$ is guaranteed.

Definition 1 (Persistency of Excitation [74]). *A function $\Phi : \mathbb{R}^+ \to \mathbb{R}^n$ is persistently exciting (**PE**) if there exist constants $T, \epsilon > 0$ such that $\int_t^{t+T} \Phi(s)\Phi^T(s)ds \succcurlyeq \epsilon I \ \forall t \geq 0$*

Definition 2 (Interval Excitation [75]). *A function $\Phi : \mathbb{R}^+ \longrightarrow \mathbb{R}^n$ is said to be interval exciting (**IE**) if there exist constants $T_0, \epsilon > 0$ such that $\int_0^{T_0} \Phi(s)\Phi^T(s)ds \succcurlyeq \epsilon I$*

Remark. *The condition that $Q(t_c) \succ 0$ is equivalent to the **IE** condition presented in Def. (2) for $T_0 := t_c$. Indeed by defining $\Phi(t) := W^T(\boldsymbol{x}(t))$, we get that $W^T(\boldsymbol{x}(t))$ is **IE** iff $\int_0^{t_c} W^T(\boldsymbol{x}(s))W(\boldsymbol{x}(s))ds \succ 0 \iff Q(t_c) \succ 0$ since $Q(t) = \int_0^t W^T(\boldsymbol{x}(s))W(\boldsymbol{x}(s))ds$ using (4.6)*

Since $W(\boldsymbol{x}(t))$ is a low-pass filtered $G(\boldsymbol{x}(t))$ (4.4), $W^T(\boldsymbol{x}(t))$ is **IE** only when $G^T(\boldsymbol{x}(t))$ is **IE** [76]. Therefore, $G^T(\boldsymbol{x}(t))$ is **IE** implies existence of t_c such that $Q(t_c) \succ 0$ *i.e.*

$$\int_0^{t_c} G^T(\boldsymbol{x}(s))G(\boldsymbol{x}(s))ds \succcurlyeq \epsilon I \implies \exists t_c \mid Q(t_c) \succ 0. \tag{4.9}$$

Next, we derive a new necessary condition which is required for $G^T(\boldsymbol{x}(t))$ to be **IE**.

Lemma 1. *Let $\mathcal{N}(G(\boldsymbol{x})) \subset \mathbb{R}^p$ denote the null space of $G(\boldsymbol{x})$, then a necessary condition for $G^T(\boldsymbol{x}(t))$ to be **IE**, is that $\mathcal{N}(G(\boldsymbol{x}))$ must **not** be time-invariant.*

Proof. We prove this lemma by contradiction. Let $\boldsymbol{v}(\boldsymbol{x}(t)) \in \mathcal{N}(G(\boldsymbol{x}(t)))$ and assume that $\boldsymbol{v}(\boldsymbol{x}(t))$ is time-invariant *i.e.* $\boldsymbol{v}(\boldsymbol{x}(t)) \equiv \mathbf{v} \in \mathbb{R}^p$ for some constant non-zero vector $\mathbf{v}$. Let $T > 0$, then we have that

$$\begin{aligned} r &= \mathbf{v}^T \left(\int_0^T G^T(\boldsymbol{x}(s))G(\boldsymbol{x}(s))ds \right) \mathbf{v} \\ &= \int_0^T \left(\mathbf{v}^T G^T(\boldsymbol{x}(s))G(\boldsymbol{x}(s))\mathbf{v} \right) ds \\ &= 0 \ \forall t > 0. \end{aligned} \tag{4.10}$$

Since we assumed that $\mathbf{v} \neq \mathbf{0}$ and T was arbitrary,

$$\begin{aligned} r = 0 &\implies \int_0^T G^T(\boldsymbol{x}(s))G(\boldsymbol{x}(s))ds \not\succcurlyeq \epsilon I \ \forall t, \epsilon > 0 \\ &\implies G^T(\boldsymbol{x}(t)) \text{ is not } \mathbf{IE} . \end{aligned}$$

Since existence of such a $\mathbf{v}$ implies $G^T(\boldsymbol{x}(t))$ is not **IE**, therefore, $\nexists\, t_c$ for which $Q(t_c) \succ 0$. $\square$

Consequently, failure to obtain positive-definiteness of $Q(t)$ prevents unique identification of $\boldsymbol{\theta}$ using (4.8). What is the intuition for this result? Recall from (4.1) that the dynamics depend on the true parameter $\boldsymbol{\theta}$ affinely through $G(\boldsymbol{x})$. A time-invariant vector $\mathbf{v} \in \mathcal{N}(G(\boldsymbol{x}(t)))$ qualitatively represents a pathological parameter that does not influence the dynamics because $G(\boldsymbol{x}(t))\mathbf{v} = \mathbf{0}$ and by extension, also does not influence the measurements $\boldsymbol{x}(t)$. Said another way, suppose $\boldsymbol{\theta}$ is the true parameter of the system and let $\boldsymbol{\theta} + \alpha\mathbf{v}$ denote an arbitrary parameter for some $\alpha \in \mathbb{R}$. Then, the following calculation shows that either of these parameters result in the same observed dynamics $\dot{\boldsymbol{x}}(t)$ for any t for a given initial condition $\boldsymbol{x}(0)$, because $G(\boldsymbol{x}(t))\mathbf{v} = \mathbf{0}$:

$$\begin{aligned}\dot{\boldsymbol{x}} &= G(\boldsymbol{x})(\boldsymbol{\theta} + \alpha\mathbf{v}) + f(\boldsymbol{x}) \\ &= G(\boldsymbol{x})\boldsymbol{\theta} + \alpha G(\boldsymbol{x})\mathbf{v} + f(\boldsymbol{x}) \\ &= G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})\end{aligned}$$

As a result, the observed measurements $\boldsymbol{x}(t)$ would be identical for either choice of parameters (*i.e.* $\boldsymbol{\theta}$ or $\boldsymbol{\theta} + \alpha\mathbf{v}$). Therefore, unique identification of the true $\boldsymbol{\theta}$ solely based on these measurements is not possible. For our application involving parameter estimation for agents, the high-level task and the resulting dynamics of each agent govern the specific form of this condition.

4.3 Problem Formulation

We have $M+1$ agents in the system. The state of an ego agent is $\boldsymbol{x} \in \mathbb{R}^2$ which represents is position and the states of the other agents are $\boldsymbol{x}_j^o \ \forall j \in \{1, 2, \cdots, M\}$. The nominal task based control for the ego agent is

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x}) \tag{4.11}$$

and we take it as a given that this control guarantees completion of the ego agent's task. The closed-loop dynamics of this agent with the collision-avoidance constraints are

$$\begin{aligned}\dot{\boldsymbol{x}} = \boldsymbol{u}_{\boldsymbol{\theta}}^* = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)\boldsymbol{u} \le \boldsymbol{b}\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)\end{aligned} \tag{4.12}$$

where $A \in \mathbb{R}^{M\times 2}$, $\boldsymbol{b} \in \mathbb{R}^M$ are such that the j^{th} row of A and the j^{th} element of $\boldsymbol{b}$ are:

$$\begin{aligned}\boldsymbol{a}_j^T(\boldsymbol{x}, \boldsymbol{x}_j^o) &\coloneqq -\Delta\boldsymbol{x}_j^T = -(\boldsymbol{x} - \boldsymbol{x}_j^o)^T \\ b_j(\boldsymbol{x}, \boldsymbol{x}_j^o) &\coloneqq \frac{\gamma}{4}(\|\Delta\boldsymbol{x}_j\|^2 - D_s^2) \ \forall j \in \{1, 2, \ldots, M\}\end{aligned} \tag{4.13}$$

The problem for the observer is to monitor the positions of the ego agent $\boldsymbol{x}(t)$ and the other agents $\boldsymbol{x}_j^o(t) \ \forall j \in \{1, 2, \cdots, M\}$ and use these to infer $\boldsymbol{\theta}$. To estimate $\boldsymbol{\theta}$ using the algorithm in section 4.2 ((4.1)-(4.8)) and to compute the condition in lemma 1, the observer requires explicit dynamics of the form $\dot{\boldsymbol{x}} = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ in (4.1). That is, it must know $G(\boldsymbol{x})$ and $f(\boldsymbol{x})$. However, owing to the fact that $\dot{\boldsymbol{x}} = \boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x})$ is optimization-based, such explicit relations are not known. In the next section, we derive the KKT conditions of (4.12) which is the first step to derive these expressions.

4.4 Analysis using KKT conditions

To analyze the relation between the optimizer of (4.12) *i.e.* $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x})$ and parameters $\boldsymbol{\theta}$, we look at the KKT conditions of this QP. These are necessary and sufficient conditions satisfied by $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x})$. The Lagrangian for (4.12) is

$$L(\boldsymbol{u}, \boldsymbol{\mu}) = \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}\|_2^2 + \boldsymbol{\mu}^T(A\boldsymbol{u} - \boldsymbol{b}).$$

Let $(\boldsymbol{u}^*_{\boldsymbol{\theta}}, \boldsymbol{\mu}^*_{\boldsymbol{\theta}})$ be the optimal primal-dual solution to (4.12). The KKT conditions are [77]:

1. Stationarity: $\nabla_{\boldsymbol{u}} L(\boldsymbol{u}, \boldsymbol{\mu})|_{(\boldsymbol{u}^*_{\boldsymbol{\theta}}, \boldsymbol{\mu}^*_{\boldsymbol{\theta}})} = 0$,

$$\begin{aligned} \implies \boldsymbol{u}^*_{\boldsymbol{\theta}} &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}\sum_{j\in\{1,\cdots,M\}} \mu^*_{j\boldsymbol{\theta}} \boldsymbol{a}_j \\ &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A^T\boldsymbol{\mu}^*_{\boldsymbol{\theta}}. \end{aligned} \tag{4.14}$$

2. Primal Feasibility

$$A\boldsymbol{u}^*_{\boldsymbol{\theta}} \leq \boldsymbol{b} \iff \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} \leq b_j \ \forall j \in \{1, \cdots, M\}. \tag{4.15}$$

3. Dual Feasibility

$$\mu^*_{j\boldsymbol{\theta}} \geq 0 \ \forall j \in \{1, 2, \cdots, M\}. \tag{4.16}$$

4. Complementary Slackness

$$\mu^*_{j\boldsymbol{\theta}} \cdot (\boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} - b_j) = 0 \ \forall j \in \{1, 2, \cdots, M\}. \tag{4.17}$$

We define the set of active and inactive constraints as

$$\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}}) \coloneqq \{j \in \{1, 2, \cdots, M\} \mid \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_j\}, \tag{4.18}$$

$$\mathcal{IA}(\boldsymbol{u}^*_{\boldsymbol{\theta}}) \coloneqq \{j \in \{1, 2, \cdots, M\} \mid \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} < b_j\}. \tag{4.19}$$

The set of active constraints qualitatively represents those other agents that the ego agent "worries" about for collisions. From the perspective of the ego agent, we will simply refer to the "other agents" as *obstacles*. Let there be a total of K active constraints *i.e.* $\texttt{card}(\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})) = K$ where $K \in \{0, 1, \cdots, M\}$. Using (4.16) and (4.17), we deduce

$$\mu^*_{j\boldsymbol{\theta}} = 0 \ \forall j \in \mathcal{IA}(\boldsymbol{u}^*_{\boldsymbol{\theta}}). \tag{4.20}$$

Therefore, we can restrict the summation in (4.14) only to the set of active constraints *i.e.*

$$\begin{aligned}\boldsymbol{u}^*_{\boldsymbol{\theta}} &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}\sum_{j\in\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})} \mu^*_{j\boldsymbol{\theta}}\boldsymbol{a}_j \\ &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A^T_{ac}\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}. \end{aligned} \tag{4.21}$$

where $A_{ac}(\boldsymbol{x}) \in \mathbb{R}^{K\times 2}$ is the matrix formed using the rows of A that are indexed by the active set $\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$, and likewise $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} \coloneqq \{\mu^*_{j\boldsymbol{\theta}}\}_{j\in\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})}$. Similarly, let $\boldsymbol{b}_{ac}(\boldsymbol{x}) \in \mathbb{R}^K$ denote the vector formed from the elements of $\boldsymbol{b}$ indexed by $\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$. By deleting all inactive constraints and retaining only the active constraints, we can pose another QP that consists only of active constraints, whose solution is the same as that of (4.12). This equality-constrained program (EQP) is given by

$$\begin{aligned} \boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A_{ac}(\boldsymbol{x})\boldsymbol{u} = \boldsymbol{b}_{ac}(\boldsymbol{x}) \end{aligned} \tag{4.22}$$

Note that the system $A_{ac}(\boldsymbol{x})\boldsymbol{u} = \boldsymbol{b}_{ac}(\boldsymbol{x})$ is always consistent by construction because of (4.18), as long as a solution $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ to (4.12) exists. Now why do we care for this EQP? That is because it is easier to derive an expression $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}) = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ for (4.22) than the inequality constrained problem (4.12). The only question is how to estimate the active set $\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$ to determine $A_{ac}(\boldsymbol{x}), b_{ac}(\boldsymbol{x})$ for (4.22). This can be done as follows. Recall that the observer can measure both the position $\boldsymbol{x}(t)$ and velocity $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$ of the ego agent. Using these, the observer can determine $\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$ by comparing the residuals $|\boldsymbol{a}^T_j(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} - b_j(\boldsymbol{x})|$ against a small threshold $\epsilon > 0$ consistent with (4.18):

$$\mathcal{A}^{observer} \coloneqq \{j \in \{1, 2, \cdots, M\} \mid |\boldsymbol{a}^T_j(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} - b_j(\boldsymbol{x})| < \epsilon\}.$$

For a small enough threshold ϵ and with perfect noiseless measurements, it holds true that $\mathcal{A}^{observer} = \mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$ consistent with (4.18). This allows the observer to determine the active set. In the next section, we work with (4.22) to derive an explicit expression for control *i.e.* $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}) = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ for various combinations of $\texttt{card}(\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}})) = K$ and $\texttt{rank}(A_{ac}(\boldsymbol{x}))$.

4.5 SVD based Analysis

The aim of this section is to derive relations between $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ needed for identifying these parameters. We will show that the dependence of $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ on these parameters banks on $\texttt{rank}(A_{ac}(\boldsymbol{x}))$. Theorems 2, 3 and 5 *roughly* state that whenever there is none or one obstacle for the ego agent to actively avoid, the control $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ exhibits a well-defined dependence on t$\boldsymbol{\theta}$ making their inference using the estimation algorithm in section 4.2 *i.e.* equations (4.1)-(4.8) possible. On the other hand, theorem 6 states that whenever there are too many obstacles active, the agent is consumed by collision avoidance constraints, so $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ "gives up" on optimizing the objective. Therefore, $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ does not depend on $\boldsymbol{\theta}$ making its inference using (4.1)-(4.8) impossible.

4.5.1 Example Task

Suppose the agent's task-based control is to reach a goal position $\boldsymbol{x}_d$ at an exponential rate governed by gain k_p. The reference control for this task is $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$. Depending on what parameters the observer wishes to infer, functions $C(\boldsymbol{x}), \boldsymbol{d}(\boldsymbol{x})$ can be defined accordingly to ensure $-k_p(\boldsymbol{x} - \boldsymbol{x}_d) \equiv C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$.

1. If the gain k_p is known to the observer and the observer wants to estimate the goal *i.e.* $\boldsymbol{\theta} = \boldsymbol{x}_d$, then define $C(\boldsymbol{x}), \boldsymbol{d}(\boldsymbol{x})$ as follows:

$$\begin{aligned} C(\boldsymbol{x}) &\coloneqq k_p I \\ \boldsymbol{d}(\boldsymbol{x}) &\coloneqq -k_p \boldsymbol{x} \end{aligned} \tag{4.23}$$

2. If the goal $\boldsymbol{x}_d$ is known to the observer and the observer wants to estimate the gain *i.e.* $\boldsymbol{\theta} = k_p$, then define $C(\boldsymbol{x}), \boldsymbol{d}(\boldsymbol{x})$ as follows:

$$\begin{aligned} C(\boldsymbol{x}) &\coloneqq -(\boldsymbol{x} - \boldsymbol{x}_d) \\ \boldsymbol{d}(\boldsymbol{x}) &\coloneqq \boldsymbol{0} \end{aligned} \tag{4.24}$$

4.5.2 No active constraints

When no constraint is active, we have $\mu_{j\boldsymbol{\theta}} = 0 \ \forall j \in \{1, 2, \cdots, M\}$, so from (4.14) we get $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}) = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$. Intuitively this means that the agent does not worry about collisions with any obstacle, so it is free to use $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ itself. Defining $G(\boldsymbol{x}) \coloneqq C(\boldsymbol{x})$ and $\boldsymbol{f}(\boldsymbol{x}) \coloneqq \boldsymbol{d}(\boldsymbol{x})$, we get the following result:

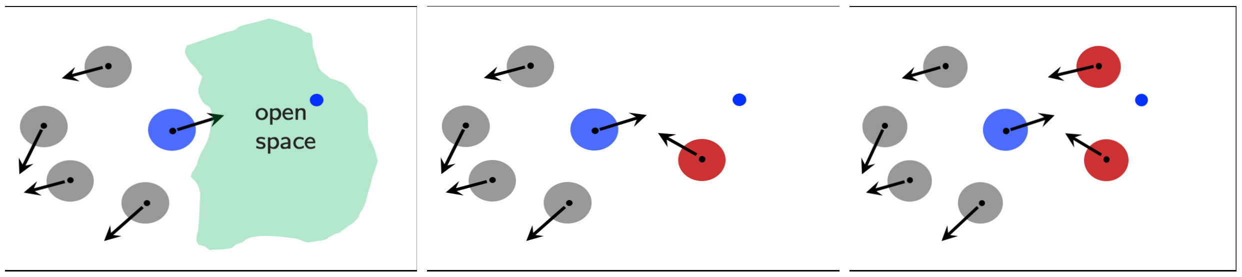
Theorem 2. *If* $\forall t \in [0, T]$*, no constraint is active, the observer can estimate* $\boldsymbol{\theta}$ *provided* $C(\boldsymbol{x})$ *is* **IE** over $[0, T]$.

Proof. The proof follows directly from (4.9) by choosing $G(\boldsymbol{x}) = C(\boldsymbol{x})$ for this case. □

We specialize this theorem to the case the agent's task is to reach a goal position $\boldsymbol{x}_d$ at an exponential rate governed by gain k_p using $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$. Fig 4.1(a) conveys this scenario intuitively.

Corollary 2.1. *If* $\forall t \in [0, T]$*, no constraint is active, then the observer can always estimate the goal assuming gain is known. Likewise, the observer can always estimate the gain assuming the goal is known, as long as the agent is not already at its goal.*

Proof. (a) If the observer wants to estimate the goal *i.e.* $\boldsymbol{\theta} = \boldsymbol{x}_d$, then for this case $C(\boldsymbol{x}) = k_p I$ from (4.23). For this case, $C(\boldsymbol{x})^T C(\boldsymbol{x}) = k_p^2 I \succ 0$ *i.e.* $C(\boldsymbol{x})$ is **IE** and $C(\boldsymbol{x})$ has no null space (lemma 1). Thus, goal estimation is always possible using theorem 2.



(a) No active constraint (b) One active constraint (c) Two active constraints

Figure 4.1: We try to intuitively convey the dependence of the dynamics on task parameters using a go-to-goal task as example. The ego robot is shown in blue, its goal as the blue dot and the rest of the robots are gray if they are perceived as inactive by the ego robot and red if active. In fig. 4.1(a), the ego robot can freely navigate to its goal since there is no other robot in its way. This scenario is information-rich for goal inference. In fig. 4.1(b), even though there is one robot in the way, the ego robot can use one degree of freedom to manifest goal directed motion. In fig. 4.1(c), since there are two robots in the way, the ego robot temporarily relinquishes goal directed behavior to repel these robots.

(b) If the observer wants to estimate the gain *i.e.* $\boldsymbol{\theta} = k_p$, then for this case $C(\boldsymbol{x}) := -(\boldsymbol{x} - \boldsymbol{x}_d)$ from (4.24). Gain estimation is only possible when $C^T(\boldsymbol{x}(t))C(\boldsymbol{x}(t)) = \|\boldsymbol{x}(t) - \boldsymbol{x}_d\|^2 \neq 0 \forall t \in [0, T]$ *i.e.* when the agent is not at its goal. This is expected because if the agent is already at its goal, then it will stay there forever, so there is no information about k_p in its position, hence the result. □

4.5.3 Exactly one active constraint

When one constraint is active, there is one obstacle that the ego agent "worries" about for collision. Since there are two degrees of freedom in the control, and one obstacle to avoid, the ego agent can avoid this obstacle and additionally minimize $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$ with the remaining degree of freedom. This causes $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ to exhibit a well-defined dependence on $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and by extension, on $\boldsymbol{\theta}$. This makes their inference using the estimation algorithm in section 4.2 *i.e.* equations (4.1)-(4.8) feasible.

Let $i \in \{1, 2, \cdots, M\}$ denote the index of the active constraint, meaning that it is the obstacle located at $\boldsymbol{x}^o_i$ that should be "actively" avoided. Thus, from (4.18), we have $A_{ac}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = \boldsymbol{a}^T_i(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_i(\boldsymbol{x})$ where $A_{ac}(\boldsymbol{x}) := \boldsymbol{a}^T_i(\boldsymbol{x})$ and $\boldsymbol{a}^T_i(\boldsymbol{x}), \boldsymbol{b}_i(\boldsymbol{x})$ are defined in (4.13). Since $\boldsymbol{a}^T_i(\boldsymbol{x}) \in \mathbb{R}^{1\times 2}$, from rank-nullity theorem it follows that $\boldsymbol{a}^T_i(\boldsymbol{x})$ has a non-trivial null space of dimension one. The null space gives one degree of freedom to the control to minimize $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$ while satisfying the constraint. We show this by computing the SVD

$\boldsymbol{a}_i^T(\boldsymbol{x}) = U(\boldsymbol{x})\Sigma(\boldsymbol{x})V^T(\boldsymbol{x})$:

$$\begin{aligned} U(\boldsymbol{x}) &\coloneqq 1 \\ \Sigma(\boldsymbol{x}) &\coloneqq \left[\Sigma_m(\boldsymbol{x}), 0\right] \text{ where } \Sigma_m(\boldsymbol{x}) = \|\boldsymbol{a}_i(\boldsymbol{x})\| \\ V(\boldsymbol{x}) &\coloneqq \left[V_1, V_2\right] \;\; V_1 = \frac{\boldsymbol{a}_i(\boldsymbol{x})}{\|\boldsymbol{a}_i(\boldsymbol{x})\|}, V_2 = R_{\frac{\pi}{2}} \frac{\boldsymbol{a}_i(\boldsymbol{x})}{\|\boldsymbol{a}_i(\boldsymbol{x})\|}. \end{aligned} \tag{4.25}$$

Since V forms a basis for $\mathbb{R}^2$, any $\boldsymbol{u}$ can be expressed as

$$\begin{aligned} \boldsymbol{u} &= \left[V_1, V_2\right] \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix}. \\ \implies \boldsymbol{a}_i^T \boldsymbol{u} - b_i &= U \left[\Sigma_m, 0\right] \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \left[V_1, V_2\right] \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} - b_i \\ &= U\Sigma_m \tilde{u}_1 + 0 \cdot \tilde{u}_2 - b_i = 0 \end{aligned} \tag{4.26}$$

Choosing $\tilde{u}_1 = \Sigma_m^{-1} U^T b_i$ and $\tilde{u}_2 = \psi \in \mathbb{R}$, we find that

$$\boldsymbol{u} = V_1 \Sigma_m^{-1} U^T b_i + V_2 \psi \tag{4.27}$$

satisfies $\boldsymbol{a}_i^T(\boldsymbol{x})\boldsymbol{u} = b_i(\boldsymbol{x}) \; \forall \psi \in \mathbb{R}$. Recall from the properties of SVD that V_2 forms a basis for $\mathcal{N}(\boldsymbol{a}_i^T(\boldsymbol{x}))$. We tune ψ to minimize $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}\|^2$ by solving the following unconstrained minimization problem

$$\begin{aligned} \psi^* &= \arg\min_{\psi} \quad \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}\|^2 \\ &= \arg\min_{\psi} \quad \left\|V_1 \Sigma_m^{-1} U^T b_i + V_2 \psi - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}\right\|^2, \end{aligned} \tag{4.28}$$

which gives $\psi^* = V_2^T \hat{\boldsymbol{u}}$. Substituting this in (4.27), gives

$$\boldsymbol{u}_{\boldsymbol{\theta}}^* = V_1 \Sigma_m^{-1} U^T b_i + V_2 V_2^T \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}. \tag{4.29}$$

Substituting $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$ gives

$$\begin{aligned} \boldsymbol{u}_{\boldsymbol{\theta}}^* &= V_1 \Sigma_m^{-1} U^T b_i + V_2 V_2^T (C\boldsymbol{\theta} + \boldsymbol{d}) \\ &= \underbrace{V_2 V_2^T C}_{G(\boldsymbol{x})} \boldsymbol{\theta} + \underbrace{V_1 \Sigma_m^{-1} U^T b_i + V_2 V_2^T \boldsymbol{d}}_{\boldsymbol{f}(\boldsymbol{x})} \\ &= G(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{f}(\boldsymbol{x}). \end{aligned} \tag{4.30}$$

This equation is the solution to (4.22) and by extension, to (4.12). Using this relation (4.30), we are ready to state the conditions under which inference of parameters using the algorithm ((4.1)-(4.8)) and lemma 1 in section 4.2 is possible.

Theorem 3. *If* $\forall t \in [0,T]$*, no constraint is active, the the observer can estimate* $\boldsymbol{\theta}$ *provided* $V_2(\boldsymbol{x})V_2^T(\boldsymbol{x})C(\boldsymbol{x})$ *is* **IE** over $[0,T]$

Proof. Follows from (4.9) by choosing $G(\boldsymbol{x}) = V_2(\boldsymbol{x})V_2^T(\boldsymbol{x})C(\boldsymbol{x})$ (from (4.30)) for this case. □

Next, we specialize this theorem to the case the agent's task is to reach a goal position $\boldsymbol{x}_d$ $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$. Fig 4.1(b) conveys this scenario intuitively.

Corollary 3.1. *If* $\forall t \in [0,T]$*, exactly one constraint is active, then the observer can estimate the goal (gain) assuming the gain (goal) is known, as long as the orientation of* $\frac{\boldsymbol{a}_i(\boldsymbol{x})}{\|\boldsymbol{a}_i(\boldsymbol{x})\|}$ *is not time-invariant and* $\boldsymbol{x}(t) \neq \boldsymbol{x}_d$ $\forall t \in [0,T]$

Proof. (a) If the observer wants to estimate the goal *i.e.* $\boldsymbol{\theta} = \boldsymbol{x}_d$ when k_p is known, then $C(\boldsymbol{x}) = k_pI, \boldsymbol{d}(\boldsymbol{x}) = -k_p\boldsymbol{x}$ from (4.23). Thus, using (4.30), $G(\boldsymbol{x}) := k_pV_2(\boldsymbol{x})V_2^T(\boldsymbol{x})$. Thus, per theorem 3, goal identification is possible when $G(\boldsymbol{x}(t))$ is **IE** *i.e.* $\int_0^T G^T(\boldsymbol{x}(t))G(\boldsymbol{x}(t))dt \succ 0$. The situation when positive-definiteness is not attained is when $\mathcal{N}(G(\boldsymbol{x}(t))$ is time-invariant which follows from Lemma 1. Note that $\mathcal{N}(G(\boldsymbol{x})) = \mathcal{N}(k_pV_2V_2^T) = V_1 = \frac{\boldsymbol{a}_i(\boldsymbol{x})}{\|\boldsymbol{a}_i(\boldsymbol{x})\|}$ which follows from the properties of SVD. Since V_1 is always a unit vector, it can only change through its orientation. If its orientation does not change over $[0,T]$, then V_1 is a time-invariant vector in $\mathcal{N}(G(\boldsymbol{x}))$ and hence goal estimation will not be possible, using Lemma 1. The video at `https://youtu.be/gh0WT_ErLxw` shows an example where invariance of the orientation of null-space results in failure to identify the goal.

(b) If the observer wants to estimate the gain *i.e.* $\boldsymbol{\theta} = k_p$, then $C(\boldsymbol{x}) = -(\boldsymbol{x}-\boldsymbol{x}_d), \boldsymbol{d}(\boldsymbol{x}) = \boldsymbol{0}$ from (4.24). Thus, $G(\boldsymbol{x}) := -V_2V_2^T(\boldsymbol{x} - \boldsymbol{x}_d)$ from (4.30). If $(\boldsymbol{x} - \boldsymbol{x}_d) \parallel V_1 \implies (\boldsymbol{x} - \boldsymbol{x}_d) \perp V_2$ then $G(\boldsymbol{x}) \equiv \boldsymbol{0}$. So if $(\boldsymbol{x}(t) - \boldsymbol{x}_d) \parallel V_1(\boldsymbol{x}(t))$ $\forall t \in [0,T]$ then $G(\boldsymbol{x}(t))$ will violate the **IE** condition needed for gain identification in theorem 3. □

4.5.4 More than one active constraint but linearly dependent

Now we consider the more general case in which there is more than one constraint active, but all of these are linearly dependent on one constraint among them. This means that effectively there is only one "representative constraint" or obstacle for the ego agent to worry about. Consequently, this case is similar to the case with just one active obstacle. We formally demonstrate this now. Let $i_1, i_2, \cdots, i_K \in \{1, \cdots, M\}$ be the indices of active constraints which satisfy

$$\begin{bmatrix} \boldsymbol{a}_{i_1}^T(\boldsymbol{x}) \\ \boldsymbol{a}_{i_2}^T(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{a}_{i_K}^T(\boldsymbol{x}) \end{bmatrix} \boldsymbol{u}_{\boldsymbol{\theta}}^* = \begin{bmatrix} b_{i_1}(\boldsymbol{x}) \\ b_{i_2}(\boldsymbol{x}) \\ \vdots \\ b_{i_K}(\boldsymbol{x}) \end{bmatrix} \quad \text{or}$$

$$A_{ac}(\boldsymbol{x})\boldsymbol{u}_{\boldsymbol{\theta}}^* = \boldsymbol{b}_{ac}(\boldsymbol{x}), \tag{4.31}$$

where $\boldsymbol{a}^T_{i_j}(\boldsymbol{x})$ and $b_{i_j}(\boldsymbol{x})$ are defined using (4.13). Since $\texttt{rank}(A_{ac}(\boldsymbol{x})) = 1$, WLOG we have $\boldsymbol{a}^T_{i_j}(\boldsymbol{x}) = \lambda_j \boldsymbol{a}^T_{i_1}(\boldsymbol{x})$ where $\lambda_j \in \mathbb{R}\ \forall j \in \{2, 3, \cdots, K\}$. Let's first see the geometric arrangements of the agent and the obstacles $i_1, i_2 \cdots, i_K$ when this case arises in practice.

Lemma 4. *The case with more than one constraint active and all linearly dependent can only arise in practice for $\lambda_j \in \{+1, -1\}$. $\lambda_j = +1$ means that obstacles indexed i_1 and i_j are coinciding and $b_{i_1}(\boldsymbol{x}) = b_{i_j}(\boldsymbol{x})$. $\lambda_j = -1$ means that the agent is located exactly in the middle of obstacles i_1 and i_j. Furthermore, even if there is just one j for which $\lambda_j = -1$, then $b_{i_j}(\boldsymbol{x}) = 0\ \forall j \in \{1, 2, \cdots, K\} \iff \boldsymbol{b}_{ac}(\boldsymbol{x}) = \boldsymbol{0}$.*

Proof. Since i_1, i_j are active constraints $\forall j \in \{2, 3, \cdots, K\}$

$$\boldsymbol{a}^T_{i_1}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_{i_1}(\boldsymbol{x}) \tag{4.32}$$
$$\boldsymbol{a}^T_{i_j}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_{i_j}(\boldsymbol{x})\ \forall j \in \{2, 3, \cdots, K\} \tag{4.33}$$

Substituting $\boldsymbol{a}^T_{i_1} = \lambda \boldsymbol{a}^T_{i_j}$ in (4.32), we get

$$\begin{aligned} &\lambda \boldsymbol{a}^T_{i_j}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_{i_1}(\boldsymbol{x}) \\ \implies &\lambda b_{i_j}(\boldsymbol{x}) = b_{i_1}(\boldsymbol{x})\ \forall j \in \{2, 3, \cdots, K\} \end{aligned} \tag{4.34}$$

Recalling that $\boldsymbol{b}_r(\boldsymbol{x}) \coloneqq \frac{\gamma}{2}(\|\boldsymbol{a}_r\|^2 - D_s^2)$ from (4.13), we get

$$\begin{aligned} &\lambda \frac{\gamma}{2}(\left\|\boldsymbol{a}_{i_j}\right\|^2 - D_s^2) = \frac{\gamma}{2}(\|\boldsymbol{a}_{i_1}\|^2 - D_s^2) \\ \implies &\lambda(\left\|\boldsymbol{a}_{i_j}\right\|^2 - D_s^2) = (\left\|\lambda \boldsymbol{a}_{i_j}\right\|^2 - D_s^2) \\ \implies &\lambda^2 \left\|\boldsymbol{a}_{i_j}\right\|^2 - \lambda(\left\|\boldsymbol{a}_{i_j}\right\|^2 - D_s^2) - D_s^2 = 0 \end{aligned} \tag{4.35}$$

This equation has two roots $\lambda = 1, -\frac{D_s^2}{\left\|\boldsymbol{a}_{i_j}\right\|^2}$.

1. $\lambda = 1 \implies b_{i_1}(\boldsymbol{x}) = b_{i_j}(\boldsymbol{x})$ and $\boldsymbol{a}^T_{i_1} = \boldsymbol{a}^T_{i_j}$ *i.e.* $\boldsymbol{x} - \boldsymbol{x}^o_{i_1} = \boldsymbol{x} - \boldsymbol{x}^o_{i_j}$ or $\boldsymbol{x}^o_{i_1} = \boldsymbol{x}^o_{i_j}$. This means obstacle i_j is coinciding with obstacle i_1. This is a trivial yet an expected result.

2. $\lambda = -\frac{D_s^2}{\left\|\boldsymbol{a}_{i_j}\right\|^2} < 0$ implies that $\boldsymbol{a}^T_{i_1}, \boldsymbol{a}^T_{i_j}$ are anti-parallel. However, when $\lambda < 0$, $b_{i_j}(\boldsymbol{x}) > 0 \implies b_{i_1}(\boldsymbol{x}) < 0$. Recalling the definition of $b_{i_j}(\boldsymbol{x})$, we know that $b_{i_j}(\boldsymbol{x}) > 0 \iff \left\|\boldsymbol{a}_{i_j}(\boldsymbol{x})\right\|^2 > D_s^2$ and therefore $b_{i_1}(\boldsymbol{x}) < 0 \iff \|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|^2 < D_s^2$. This means that if the agent is *strictly safe* with respect to obstacle i_j, then it is colliding with obstacle i_1. This means that the control at the previous time step $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}(t^-))$ caused this collision which is not possible. This conflict can only be resolved when we relax *strict safety* to $b_{i_j}(\boldsymbol{x}) = 0 \implies \left\|\boldsymbol{a}_{i_j}(\boldsymbol{x})\right\|^2 = D_s^2$ meaning that the agent and obstacle j are touching each other. This gives $b_{i_1}(\boldsymbol{x}) = 0$ implying that the agent and obstacle i_1 are also touching each other. In this case $\lambda = -\frac{D_s^2}{\left\|\boldsymbol{a}_{i_j}\right\|^2} = -1$ which implies $\boldsymbol{x} - \boldsymbol{x}^o_{i_1} = -(\boldsymbol{x} - \boldsymbol{x}^o_{i_j})$ or $\boldsymbol{x} = \frac{1}{2}(\boldsymbol{x}^o_{i_1} + \boldsymbol{x}^o_{i_j})$. Furthermore, even if there is one j for which $\lambda_j = 0$, then from (4.34), $b_{i_j} = 0 \forall j \{1, 2, \cdots, , K\} \implies \boldsymbol{b}_{ac} = \boldsymbol{0}$

□

Next, we derive an analytical expression for $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x})$ using (4.31). Note $A_{ac}(\boldsymbol{x}) = U(\boldsymbol{x})\Sigma(\boldsymbol{x})V^T(\boldsymbol{x})$ where

$$\begin{aligned} U &:= \left[U_1, U_2\right] \ , U_1 = \frac{1}{\sqrt{1+\Sigma_{j=2}^K \lambda_j^2}}\left[1, \lambda_2, \cdots \lambda_K\right]^T \\ \Sigma &:= \left[\begin{array}{c|c} \Sigma_r & 0^{1\times 1} \\ \hline 0^{K-1\times 1} & 0^{K-1\times 1}\end{array}\right] \ , \Sigma_r = \sqrt{1+\Sigma_{j=2}^K \lambda_j^2}\,\|\boldsymbol{a}_i(\boldsymbol{x})\| \\ V &:= \left[V_1, V_2\right], V_1 = \frac{\boldsymbol{a}_{i_1}(\boldsymbol{x})}{\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|}, V_2 = R_{\frac{\pi}{2}}\frac{\boldsymbol{a}_{i_1}(\boldsymbol{x})}{\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|}. \end{aligned} \tag{4.36}$$

Choosing $\boldsymbol{u} = V_1\tilde{u}_1 + V_2\tilde{u}_2$, from (4.31) we get

$$\begin{aligned} A_{ac}\boldsymbol{u} - \boldsymbol{b}_{ac} &= \left[U_1, U_2\right]\left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0\end{array}\right]\begin{bmatrix} V_1^T \\ V_2^T\end{bmatrix}\left[V_1, V_2\right]\begin{bmatrix}\tilde{u}_1 \\ \tilde{u}_2\end{bmatrix} - \boldsymbol{b}_{ac} \\ &= \left[U_1, U_2\right]\left(\begin{bmatrix}\Sigma_r\tilde{u}_1 \\ 0\end{bmatrix} - \begin{bmatrix} U_1^T\boldsymbol{b}_{ac} \\ U_2^T\boldsymbol{b}_{ac}\end{bmatrix}\right). \end{aligned} \tag{4.37}$$

Since $\|\|^2$ is unitary invariant, from (4.31)

$$\begin{aligned} \|A_{ac}\boldsymbol{u} - \boldsymbol{b}_{ac}\|^2 &= \left\|\begin{bmatrix}\Sigma_r\tilde{u}_1 \\ 0\end{bmatrix} - \begin{bmatrix} U_1^T\boldsymbol{b}_{ac} \\ U_2^T\boldsymbol{b}_{ac}\end{bmatrix}\right\|^2 \\ &= \left\|\Sigma_r\tilde{u}_1 - U_1^T\boldsymbol{b}_{ac}\right\|^2 + \left\|U_2^T\boldsymbol{b}_{ac}\right\|^2. \end{aligned} \tag{4.38}$$

The minimum norm is achieved for $\tilde{u}_1 = \Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac}$. Choosing $\tilde{u}_2 = \psi \in \mathbb{R}$, the "least-squares" solutions are

$$\boldsymbol{u} = V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2\psi. \tag{4.39}$$

Computing ψ by minimizing $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}\|^2$, we get $\psi^* = V_2^T\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ which gives

$$\boldsymbol{u}^*_{\boldsymbol{\theta}} = V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}. \tag{4.40}$$

Substituting $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$ in (4.40) gives

$$\begin{aligned} \boldsymbol{u}^*_{\boldsymbol{\theta}} &= V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T(C\boldsymbol{\theta} + \boldsymbol{d}) \\ &= \underbrace{V_2V_2^TC}_{G(\boldsymbol{x})}\,\boldsymbol{\theta} + \underbrace{V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T\boldsymbol{d}}_{\boldsymbol{f}(\boldsymbol{x})} \\ &= G(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{f}(\boldsymbol{x}). \end{aligned} \tag{4.41}$$

Suppose $\lambda_j = +1\ \forall j \in \{2, 3, \cdots, K\}$, then from (4.36), we have $U_1 = \frac{1}{\sqrt{K}}\mathbf{1}$, $\Sigma_r = \sqrt{K}\,\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|$. Moreover, from lemma 4, we have $b_{i_j}(\boldsymbol{x}) = b_{i_1}(\boldsymbol{x})\ \forall j \in \{2, 3, \cdots, K\}$, this means that, and

$\boldsymbol{b}_{ac}(\boldsymbol{x}) = \mathbf{1}b_{i_1}(\boldsymbol{x})$. Substituting this in (4.40), one can verify that we get the same expression for control as in (4.29). This is expected because $\lambda_j = 1\ \forall j \in \{2, 3, \cdots, K\}$ means that all obstacles are coinciding so the ego agent treats them all as one obstacle, hence the control is identical to one when there was just one active obstacle in 4.5.3. The slight difference between this case and 4.5.3 comes when there is a j for which $\lambda_j = -1$. From lemma 4, this happens when the agent is in the middle of i_1 and i_j and $\boldsymbol{b}_{ac} = \mathbf{0}$. Then it follows from (4.40) that

$$\boldsymbol{u}^*_{\boldsymbol{\theta}} = V_2 V_2^T \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}. \tag{4.42}$$

$V_2 V_2^T \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ is the projection of $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ along $V_2 = R_{\frac{\pi}{2}} \frac{\boldsymbol{a}_{i_1}(\boldsymbol{x})}{\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|}$. This is expected because when the agent is in the middle of the obstacles (lemma 4), the only feasible direction of motion is along the line that is perpendicular to the line segment connecting the obstacles *i.e.* along $R_{\frac{\pi}{2}} \frac{\boldsymbol{a}_{i_1}(\boldsymbol{x})}{\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|}$ because motion along any other direction will cause collisions. We are ready to state the conditions under which inference of $\boldsymbol{\theta}$ is possible using (4.41) as the expression for $\boldsymbol{u}^*_{\boldsymbol{\theta}}$.

Theorem 5. *If $\forall t \in [0, T]$, no constraint is active, then following (4.9), the observer can estimate $\boldsymbol{\theta}$ provided $V_2(\boldsymbol{x})V_2^T(\boldsymbol{x})C(\boldsymbol{x})$ is* **IE** over $[0, T]$.

Proof. Follows from (4.9) by choosing $G(\boldsymbol{x}) = V_2(\boldsymbol{x})V_2^T(\boldsymbol{x})C(\boldsymbol{x})$ (from (4.41)) for this case. □

Next, we specialize this theorem to the case where the agent's task is to reach a goal position $\boldsymbol{x}_d$ at an exponential rate governed by gain k_p using $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = -k_p(\boldsymbol{x} - \boldsymbol{x}_d)$.

Corollary 5.1. *If $\forall t \in [0, T]$, two or more constraints are active, all of which are linearly dependent on one among them, then the observer can estimate the goal (gain) assuming the gain (goal) is known, as long as the orientation of $\frac{\boldsymbol{a}_{i_1}(\boldsymbol{x})}{\|\boldsymbol{a}_{i_1}(\boldsymbol{x})\|}$ is not time-invariant and $\boldsymbol{x}(t) \neq \boldsymbol{x}_d\ \forall t \in [0, T]$*

Proof. The proof is similar to the proof of corollary 3.1 so it is skipped. □

4.5.5 Two or more linearly-independent active constraints

Now consider the case where *two* of K constraints are linearly independent, while the remaining constraints are linear combinations of these two. There are fewer degrees of freedom in control than the number of independent active obstacles to avoid, hence $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ is completely determined by these constraints and does not depend on $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$, and by extension, neither on $\boldsymbol{\theta}$. We formally demonstrate this claim as follows. Let $i_1, i_2, \cdots, i_K \in \{1, 2, \cdots, M\}$ be the indices of the K active constraints. These constraints satisfy (4.31) except that here

Table 4.1: Summarizing the control expressions for various cases. The red row shows the case which violates the persistency of excitation criterion.

Case	K	$\texttt{rank}(A_{ac})$	$\boldsymbol{u}^*(\boldsymbol{x})$	Relevant Equations
A (section 4.5.2)	None	-	$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$	-
B (section 4.5.3)	1	1	$V_1\Sigma_m^{-1}U^T b_i + V_2V_2^T\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$	(4.29)-(4.30)
C (section 4.5.4)	≥ 2	1	$V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$	(4.40)-(4.41)
D (section 4.5.5)	≥ 2	2	$A_{ac}^{\dagger}\boldsymbol{b}_{ac}$	(4.43)-(4.45)

$\texttt{rank}(A_{ac}(\boldsymbol{x})) = 2$. This problem is overdetermined but not ill-posed because by construction (4.31) is consistent. Its solution is

$$\begin{aligned} \boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \arg\min_{\boldsymbol{u}} \|A_{ac}(\boldsymbol{x})\boldsymbol{u} - \boldsymbol{b}_{ac}(\boldsymbol{x})\|_2^2 \\ &= A_{ac}^{\dagger}(\boldsymbol{x})\boldsymbol{b}_{ac}(\boldsymbol{x}) \end{aligned} \tag{4.43}$$

where $A_{ac}^{\dagger}(\boldsymbol{x})$ denotes the Moore-Penrose pseudo inverse defined as

$$A_{ac}^{\dagger}(\boldsymbol{x}) := (A_{ac}^T(\boldsymbol{x})A_{ac}(\boldsymbol{x}))^{-1}A_{ac}^T(\boldsymbol{x}). \tag{4.44}$$

When $K = 2$, $A_{ac}^{\dagger}(\boldsymbol{x}) \equiv A_{ac}^{-1}(\boldsymbol{x})$. Since neither $A_{ac}^{\dagger}(\boldsymbol{x})$ nor $\boldsymbol{b}_{ac}(\boldsymbol{x})$ depend on $\boldsymbol{\theta}$ (4.13), inference of $\boldsymbol{\theta}$ is **not** possible. This can be seen by rewriting (4.43) as

$$\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underbrace{O}_{G(\boldsymbol{x})}\boldsymbol{\theta} + \underbrace{A_{ac}^{\dagger}(\boldsymbol{x})\boldsymbol{b}_{ac}(\boldsymbol{x})}_{\boldsymbol{f}(\boldsymbol{x})} \tag{4.45}$$

Theorem 6. *If $\forall t \in [0, T]$, two or more than two constraints are active and two of these are linearly independent, then the observer **cannot** estimate $\boldsymbol{\theta}$.*

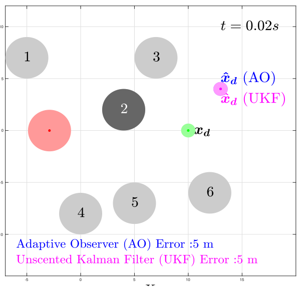
Proof. Since $G(\boldsymbol{x}) \equiv O\ \forall t \in [0, T]$ (4.45), it does not (4.9), hence the result. □

Next, we specialize this theorem to the case where the agent's task is to reach a goal position $\boldsymbol{x}_d$. Fig 4.1(c) conveys this scenario intuitively.

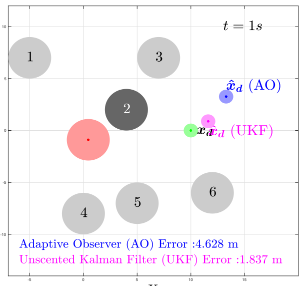
Corollary 6.1. *If $\forall t \in [0, T]$, two or more constraints are active and two of these are linearly independent, then the observer **cannot** estimate either the goal or the gain.*

Proof. Since $G \equiv O$, it does not matter whether the parameter to be inferred is the goal $\boldsymbol{x}_d$ or the gain k_p, either of these will not be inferrable due to theorem 6. □

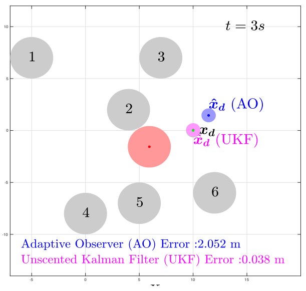
Table 4.1 summarizes the control expressions that we derived in sections 4.5.2 - 4.5.5


$t = 0.02s$
1
2
3
4
5
6
$\hat{x}_d$ (AO)
$\hat{x}_d$ (UKF)
x_d
Adaptive Observer (AO) Error :5 m
Unscented Kalman Filter (UKF) Error :5 m

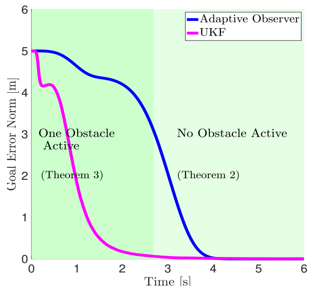
(a) $t = 0.02s$


$t = 1s$
1
2
3
4
5
6
$\hat{x}_d$ (AO)
$\hat{x}_d$ (UKF)
x_d
Adaptive Observer (AO) Error :4.628 m
Unscented Kalman Filter (UKF) Error :1.837 m

(b) $t = 1s$


$t = 3s$
1
2
3
4
5
6
$\hat{x}_d$ (AO)
$\hat{x}_d$ (UKF)
x_d
Adaptive Observer (AO) Error :2.052 m
Unscented Kalman Filter (UKF) Error :0.038 m

(c) $t = 3s$

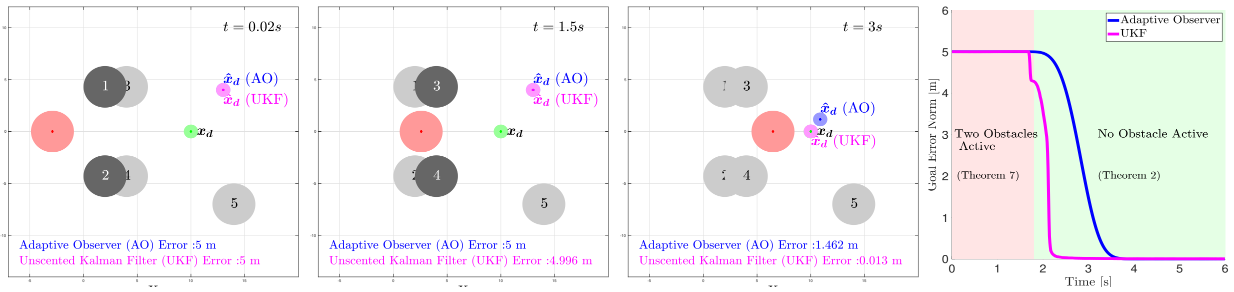


(d) Adaptive Observer and UKF

Figure 4.2: (a)-(c) A agent navigating to its goal (green) amongst static obstacles. Dark disc represents an active obstacle. Estimates of goal using AO and UKF are shown in blue and pink discs. (d) The norms of goal position errors for AO and UKF. Since atmost one obstacle is active, estimation errors converge to zero. Video at `https://youtu.be/jPIsdM08-II`

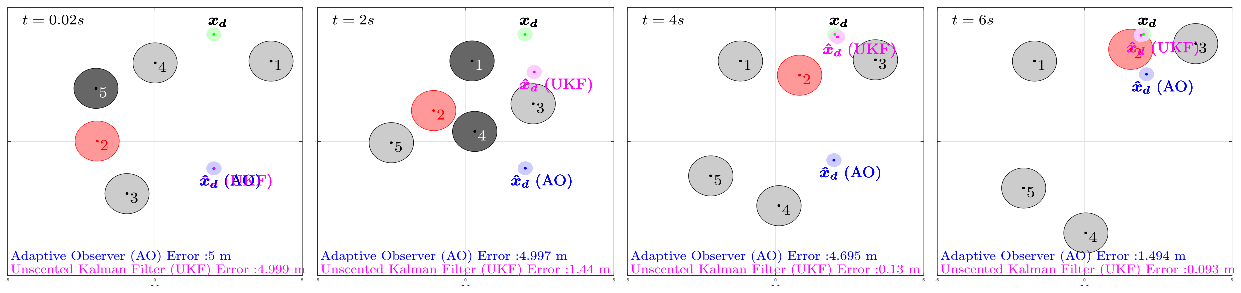
4.6 Simulation Results

In this section, we present results for estimation of goals for a multiagent system. We consider two estimators, (a) a UKF and (b) an adaptive observer (AO) (4.1) to (4.8). $G(\boldsymbol{x})$, $f(\boldsymbol{x})$ for AO are chosen per theorems 2-6 by checking the number of active obstacles at a given time. The AO converges only when the necessary conditions in these theorems are satisfied. As for UKF, while it doesn't require explicit dynamics, it doesn't provide any guarantees for convergence either. To substantiate this, we first show simulations for a single agent navigating towards its goal in an environment consisting of static obstacles. In Figs. 4.2(a)-4.2(c), an ego agent (red) is trying to reach its goal shown in green. As the agent moves to the right, obstacle two remains active until $t = 2.8s$. While there are six obstacles, only one of them is active in this duration, hence theorem 3 guarantees reduction in goal estimation error using AO. This is shown in the dark green left panel of Fig. 4.2(d). For $t > 2.8s$, no obstacle is active hence theorem 2 ensures that the goal estimation error converges to zero using AO as evident in Fig 4.2(d). A similar trend is obtained using UKF. In Figs. 4.3(a)-4.3(c), the same agent is trying to reach its goal shown in green. We have purposefully positioned the obstacles in such a way that as the agent moves, obstacle one and two are active until $t = 1.08s$, at which point, obstacle three and four become active, and stay so until $t = 1.8s$. Thus until $t = 1.8s$, two obstacles are always active. Hence, from theorem 6, agent dynamics do not depend on the goal location. As expected, the AO error does not decrease as is evident from the red panel of Fig. 4.3(d). *Interestingly, this is also true for UKF, which empirically shows that convergence of estimation error is agnostic to the choice of estimator*, which is because agent dynamics itself do not depend on the goal. Thereafter, no obstacle is active, hence the estimation errors converge to zero as is evident from the



(a) $t = 0.02s$ (b) $t = 1.5s$ (c) $t = 3s$ (d) Adaptive Observer and UKF

Figure 4.3: (a)-(c) Goal identification for a agent navigating amongst static obstacles. (d) The red patch represents the duration in which identification is not supposed to work, because two obstacles are active. After $t \sim 1.8s$, no obstacles are active, hence estimation errors begin to converge to zero. Video at `https://youtu.be/8k7Zegn2lvw`



(a) $t = 0.02s$ (b) $t = 2s$ (c) $t = 4s$ (d) $t = 6s$

Figure 4.4: (a)-(c) Goal estimation for a multiagent system. We highlight the ego agent (red) for legibility. AO and UKF estimates are shown in blue and pink discs respectively. Video at `https://youtu.be/i6RiyA_AtbU`

green panel in Fig. 4.3(d). Finally, we consider a multiagent system in Fig. 4.4 in which we run parallel estimators synchronously. To ensure that the snapshots are legible, we only highlight the ego agent (*i.e.* agent 2), while other agents are light grey or dark depending on whether they are active or inactive for the ego agent. In this simulation, there are times when one agent is active (Fig. 4.4(a)), two are active (Fig. 4.4(b)) and none are active (Fig. 4.4(c)). The estimation errors are shown in Fig. 4.5. The grey curves correspond to the non-ego agents and the blue (AO) and pink (UKF) curves are for the ego agent. Since all the curves converge to zero, the estimates of goals for all agents converge to their true goals.

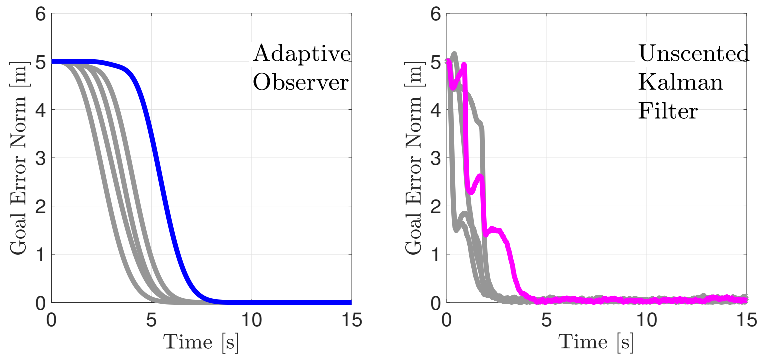


Figure 4.5: Estimation errors as a function of time for AO (left) and UKF (right). The highlighted curves are for the ego agent while the gray ones are for other agents.

4.7 Conclusions

In this chapter, we developed a mathematical framework for observer based task inference of a multiagent system by using ideas from system identification. Since these agents use optimization in the feedback loop, their dynamics depend implicitly on task parameters which makes the application of previously developed estimators as well as identifiability conditions non-trivial. We used duality theory to derive explicit relations to derive the identifiability conditions. The message that our theorems convey is that as the number of agents that an ego agent interacts with increases, estimation of ego's parameters becomes difficult because with more interactions, the ego agent's motion is expended in avoiding collisions which it achieves by sacrificing task performance. In the next chapter, we will explore how to learn bounds on parameters $\boldsymbol{\theta}$ when the identifiability conditions are violated.

5 Inferring Bounds on Task Parameters

5.1 Introduction

In the chapter 4, we derived necessary conditions based on *persistency of excitation* analysis for inferring the task parameters of each agent in a multiagent system. We demonstrated the utility of these conditions by trying to estimate the goals and gains of all agents using a UKF based parameter estimator and an adaptive observer [73], [72]. These algorithms are *pointwise estimators* in that they generate a single estimate of the parameter that converges to the true parameter when the system dynamics satisfy the persistency of excitation criterion [74]. However, as we saw in chapter 4, this condition may not necessarily be satisfied in a multiagent system. In particular, we showed that whenever an agent has active interactions with two or more other agents, this condition will get violated for that agent. Now since interactions among agents are inevitable, by this reasoning, pointwise parameter estimators would fail to identify the true parameter for that agent.

This takeaway from the previous chapter forms the motivation for this chapter. That is, while it may not be possible to get an exact estimate of the task parameter of an agent, we want to determine if we can estimate any bounds for it. To address this problem, we propose a *feasible-region based identification* algorithm that generates a bounded set which contains the true parameter. We design this algorithm such that the "measure" of this set (*i.e.* its size) is non-increasing with time. There are several advantages of our proposed approach over point-wise identification.

1. Firstly, this algorithm is anytime *i.e.* the set computed at a given time is comprised of parameters that are all valid candidates for explaining the agent measurements observed until that time.

2. Secondly, this algorithm does not have any gains to tune as in Kalman filters or adaptive observers [78] which require user's intuition for tuning.

3. Thirdly, this algorithm can also work symbiotically with point-wise estimation algorithms and can expedite their convergence by continually projecting their estimates to the feasible regions generated by this algorithm.

4. Finally, our empirical evidence suggests that even in cases where point-wise estimation is possible, our algorithm converges much faster than a point-wise estimator (a UKF used for comparison).

The outline of this chapter is as follows. In section 5.2, we give a brief review of point-wise parameter estimation following the development in [27, 79] and establish notation for our proposed *feasible region-based* identification approach. In section 5.3, we formalize the *feasible region-based* task identification problem. The main contributions begin from section 5.4. We use the KKT conditions of the control synthesis optimization (3.14) to pose an equality constrained optimization problem (EQP) formulated using the set of active constraints of the ego agent. In section 5.5, we derive explicit relations between agent dynamics, Lagrange multipliers and task parameters. Finally, using these relations along with the KKT conditions, we derive analytical descriptions of the sets where the task parameters must belong as we wanted. We demonstrate the power of this identification approach through numerical simulations in section 5.6. We consider geometric settings that span all combinations of number of active constraints and linear independence relations we theorized in section 5.5 and show through simulations how the bounds on task parameters computed by our approach converge much faster than a UKF. Finally, we conclude in section 5.7 and provide directions for future work.

5.2 Approaches for Parameter Identification

We describe two different approaches for parameter identification. These include (a) Point-wise identification and (b) Feasible-region based identification. The need for distinguishing between these two approaches is necessitated by our application *i.e.* task inference for multiagent systems. Approach (a) can often fail to converge owing to unavoidable interactions between agents. In such circumstances, approach (b) can provide reasonable bounds on task parameters.

5.2.1 Point-wise identification

The *point-wise identification* approach focuses on designing an online parameter update law which ensures that the estimate of the parameter converges to the true parameter of the underlying system. Several known estimation algorithms fall in this category, including RLS [70], UKF [71] and the ones based on adaptive observers [72, 73, 79]. Consider a nonlinear system:

$$\dot{\boldsymbol{x}} = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x}), \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the measurable state, $\boldsymbol{\theta} \in \mathbb{R}^p$ is the unknown parameter and $G(\boldsymbol{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^{n\times p}$, $f(\boldsymbol{x}) : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ are known functions. For example, in our context, $\boldsymbol{x}(t)$ will correspond to the position of the ego agent under observation and $\boldsymbol{\theta}$ denotes the task parameters of its controller that the observer wishes to infer. We can assume that the observer runs several parallel estimators, one for each agent, so the observer's focus is on the ego agent. Adhering to the *point-wise identification* paradigm, the observer's problem is to design an estimation law $\dot{\hat{\boldsymbol{\theta}}} = \psi(\hat{\boldsymbol{\theta}}, \boldsymbol{x})$ that guarantees convergence of $\hat{\boldsymbol{\theta}} \longrightarrow \boldsymbol{\theta}$ by using $\boldsymbol{x}(t)$. A necessary condition for this convergence is that $G(\boldsymbol{x})$ be persistently exciting [69].

5.2.2 Feasible region-based identification

In this approach, the aim is to compute a set where the parameter of the underlying system must belong rather than computing its exact value. The observer may use some measurements $\boldsymbol{y}(t)$ and compute a set $\boldsymbol{\Theta}(t) \subset \mathbb{R}^p$ such that

1. $\boldsymbol{\Theta}(t)$ is bounded.
2. $\boldsymbol{\theta} \in \boldsymbol{\Theta}(t)$ $\forall t$.
3. The measure of this set, denoted by $\mu(\boldsymbol{\Theta}(t))$ is non-increasing with time.

Ideally, if condition (2) is satisfied and condition (3) is replaced with a strict decrease in $\mu(\boldsymbol{\Theta}(t))$, it is guaranteed that $\texttt{lim}_{t\to\infty}\boldsymbol{\Theta}(t) = \boldsymbol{\theta}$. However, the relaxed condition (3) *i.e.* $\mu(\boldsymbol{\Theta}(t))$ is non-increasing is easier to ensure in practice and will be our focus. One way to define a set that satisfies these three conditions is as follows. Suppose at time t, we deduce that $\boldsymbol{\theta} \in \boldsymbol{\Omega}(t)$ where

$$\boldsymbol{\Omega}(t) \coloneqq \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | \boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}}) \prec \boldsymbol{0}\}, \tag{5.2}$$

for some function $\boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}})$ that depends on the observer's measurements $\boldsymbol{y}(t)$. We call $\boldsymbol{\Omega}(t)$ the *instantaneous feasible set.* Note that $\boldsymbol{\Omega}(t)$ need not be bounded, that would depend on how $\boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}})$ is defined. Further, suppose $\boldsymbol{\Theta}_0 \subset \mathbb{R}^p$ is a known time-invariant compact set in which $\boldsymbol{\theta}$ is known to belong a-priori. Then defining

$$\boldsymbol{\Theta}(t) \coloneqq \bigcap_{0\leq\tau\leq t} \boldsymbol{\Omega}(\tau) \cap \boldsymbol{\Theta}_0, \tag{5.3}$$

will ensure satisfaction of the three conditions proposed above. Indeed, let t_1 and t_2 be two time instants such that $t_1 < t_2$, then from (5.3) it is evident that $\boldsymbol{\Theta}(t_2) \subseteq \boldsymbol{\Theta}(t_1)$. This would in-turn imply that $\mu(\boldsymbol{\Theta}(t_2)) \leq \mu(\boldsymbol{\Theta}(t_1))$ or in other words, $\mu(\boldsymbol{\Theta}(t))$ non-increasing. Since $\boldsymbol{\Theta}(t)$ is derived from the set $\boldsymbol{\Omega}(t)$, it suffices to focus on constructing the *instantaneous feasible set* $\boldsymbol{\Omega}(t)$ *i.e.* deriving $\boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}})$.

In the rest of the chapter, our goal is to construct the *instantaneous feasible set* $\boldsymbol{\Omega}(t)$ for inferring parameters of each agent in the multiagent system. We will assume the same formalism as in chapter 3-chapter 4 to model the dynamics of each agent:

$$\begin{aligned} \dot{\boldsymbol{x}} = \boldsymbol{u}^*_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A\Big(\boldsymbol{x}, \{\boldsymbol{x}^o_j\}\Big)\boldsymbol{u} \leq \boldsymbol{b}\Big(\boldsymbol{x}, \{\boldsymbol{x}^o_j\}\Big) \end{aligned} \tag{5.4}$$

where $A \in \mathbb{R}^{M\times 2}$, $\boldsymbol{b} \in \mathbb{R}^M$. The nominal task based control for the ego agent is

$$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x}) \tag{5.5}$$

We will use these dynamics to illustrate the mechanics of constructing $\boldsymbol{\Omega}(t)$ for inferring $\boldsymbol{\theta}$.

5.3 Region-based Identification Problem

The observer's problem is to identify a region in the parameter space $\boldsymbol{\Theta}(t) \subset \mathbb{R}^p$ consistent with the three conditions of the *feasible-region based identification* approach proposed in Sec. 5.2. Following the analysis in section 5.2.2, it suffices to estimate the *instantaneous feasible set* $\boldsymbol{\Omega}(t)$ (5.2) as the 0-sublevel set of a function $\boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}})$. The measurements $\boldsymbol{y}(t)$ that the observer can use to compute this set include (a) the positions of the ego agent $\boldsymbol{x}(t)$ and (b) the positions of the other agents $\boldsymbol{x}^o_j(t)\ \forall j \in \{1, 2, \cdots, M\}$ *i.e.* $\boldsymbol{y}(t) = (\boldsymbol{x}(t), \boldsymbol{x}^o_1(t), \cdots, \boldsymbol{x}^o_M(t))$. The observer will run $M+1$ parallel identifiers to compute $\{\boldsymbol{\Omega}_i(t)\}_{i=1}^{M+1}$ for each agent in the team since each agent has its unique task parameter $\boldsymbol{\theta}_i$, in general. Focusing on the ego agent, the observer needs to derive functions $\boldsymbol{g}(\boldsymbol{y}(t), \hat{\boldsymbol{\theta}})$. For that, the observer must know an explicit relation between the dynamics of the ego agent and the parameters $\boldsymbol{\theta}$ of the form $\dot{\boldsymbol{x}} = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ similar to (5.1). That is, it must know $G(\boldsymbol{x})$ and $f(\boldsymbol{x})$. However, owing to the fact that $\dot{\boldsymbol{x}} = \boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x})$ is optimization-based (5.4), such explicit relations are not known. We derive these relations conditions using the KKT conditions of (5.4).

5.4 Analysis using KKT conditions

Since we covered the KKT conditions in chapter 4 (section 4.4), we state them here directly without deriving them.

Stationarity	$\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A^T\boldsymbol{\mu}^*_{\boldsymbol{\theta}} = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}\sum_{j\in\{1,\cdots,M\}} \mu^*_{j\boldsymbol{\theta}}\boldsymbol{a}_j$
Primal Feasibility	$A\boldsymbol{u}^*_{\boldsymbol{\theta}} \le \boldsymbol{b} \iff \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} \le b_j\ \forall j \in \{1, \cdots, M\}$
Dual Feasibility	$\mu^*_{j\boldsymbol{\theta}} \ge 0\ \forall j \in \{1, 2, \cdots, M\}$
Complementary Slackness	$\mu^*_{j\boldsymbol{\theta}} \cdot (\boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} - b_j) = 0\ \forall j \in \{1, 2, \cdots, M\}$

Recall the set of inactive and active constraints are:

$$\mathcal{IA}(\boldsymbol{u}^*_{\boldsymbol{\theta}}) \coloneqq \{j \in \{1, 2, \cdots, M\} \mid \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} < b_j\}, \tag{5.6}$$
$$\mathcal{A}(\boldsymbol{u}^*_{\boldsymbol{\theta}}) \coloneqq \{j \in \{1, 2, \cdots, M\} \mid \boldsymbol{a}_j^T\boldsymbol{u}^*_{\boldsymbol{\theta}} = b_j\}. \tag{5.7}$$

5.4.1 Using KKT conditions for deriving $\boldsymbol{\Omega}(t)$

Now we briefly describe how these conditions are useful for the computing the *instantaneous feasible set* $\boldsymbol{\Omega}(t)$ (5.2) where the parameter $\boldsymbol{\theta}$ belongs.

1. **Inactive constraints:** From the definition of inactive constraints (5.6), recall $\boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* < b_j \; \forall j \in \mathcal{IA}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)$. Thus, deriving an explicit expression for $\boldsymbol{u}_{\boldsymbol{\theta}}^* = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ will allow us to compute a set where $\boldsymbol{\theta}$ belongs.

2. **Non-negativity of Lagrange multipliers:** Likewise from dual feasibility, recall that $\mu_{j\boldsymbol{\theta}}^* \geq 0 \; \forall j \in \{1, 2, \cdots, M\}$. Since $\mu_{j\boldsymbol{\theta}}^*$ depends on $\boldsymbol{\theta}$, deriving an explicit expression for $\mu_{j\boldsymbol{\theta}}^*$ as a function of $\boldsymbol{\theta}$ will allow us to further prune the set where $\boldsymbol{\theta}$ belongs.

Thus, using these two criteria, we define $\boldsymbol{\Omega}(t)$ as

$$\boldsymbol{\Omega} := \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | \boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j < 0 \; \forall j \in \mathcal{IA}(\boldsymbol{u}_{\boldsymbol{\theta}}^*), \; \mu_{j\boldsymbol{\theta}}^* \geq 0 \; \forall j \in \{1, \cdots, M\}\}. \tag{5.8}$$

From this definition, we can see that we need explicit representation of $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ as a function of $\boldsymbol{\theta}$ and $\mu_{j\boldsymbol{\theta}}^*$ as a function of $\boldsymbol{\theta}$. While we calculated $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ in section 4.5, we still need to compute $\mu_{j\boldsymbol{\theta}}^*$. This is the subject of the next section.

5.4.2 Using KKT conditions for deriving $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ and $\mu_{j\boldsymbol{\theta}}^*$

The content of this section is similar to section 4.4, so we briefly recall the equations that are absolutely relevant. Let there be a total of K active constraints *i.e.* $\texttt{card}(\mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)) = K$ where $K \in \{0, 1, \cdots, M\}$. From dual feasibility and complimentary slackness, we have:

$$\mu_{j\boldsymbol{\theta}}^* = 0 \; \forall j \in \mathcal{IA}(\boldsymbol{u}_{\boldsymbol{\theta}}^*) \tag{5.9}$$

This results in a simplification of the stationarity condition:

$$\boldsymbol{u}_{\boldsymbol{\theta}}^* = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2} A_{ac}^T \boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac}. \tag{5.10}$$

where $A_{ac}(\boldsymbol{x}) \in \mathbb{R}^{K \times 2}$ is the matrix formed using the rows of A that are indexed by the active set $\mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)$, and likewise $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac} := \{\mu_{j\boldsymbol{\theta}}^*\}_{j \in \mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)}$. Similarly, let $\boldsymbol{b}_{ac}(\boldsymbol{x}) \in \mathbb{R}^K$ denote the vector formed from the elements of $\boldsymbol{b}$ indexed by $\mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)$. Likewise, we can define $A_{inac}(\boldsymbol{x})$ and $\boldsymbol{b}_{inac}(\boldsymbol{x})$ corresponding to the inactive set. By deleting all inactive constraints and retaining only the active constraints from (5.4), we arrive at equality-constrained QP:

$$\begin{aligned} \boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} & \quad \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} & \quad A_{ac}(\boldsymbol{x})\boldsymbol{u} = \boldsymbol{b}_{ac}(\boldsymbol{x}). \end{aligned} \tag{5.11}$$

This EQP is useful because it is easier to derive explicit expressions for both $\boldsymbol{u}_{\boldsymbol{\theta}}^*(\boldsymbol{x})$ and $\mu_{j\boldsymbol{\theta}}^*$ using (5.11) than (5.4). As we did in chapter 4, the observer can estimate the active set $\mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)$ by comparing residuals $|\boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j|$ against a threshold $\epsilon > 0$ consistent with (5.6):

$$\mathcal{A}^{obs.} := \{j \in \{1, \cdots, M\} \mid |\boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j| < \epsilon\}. \tag{5.12}$$

Table 5.1: Enumerating cases where either point-wise identification or region-based identification or both are possible

Case	K	$\texttt{rank}(A_{ac})$	**Point-wise**	**Region-based**
A	None	-	Possible	Possible
B	1	1	Possible	Possible
C	≥ 2	1	Possible	Possible
D	2	2	Not possible	Possible
E	≥ 3	2	Not possible	Possible

For a small threshold ϵ, it holds true that $\mathcal{A}^{obs.} = \mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*)$ consistent with (5.6). This allows the observer to determine the active set. Next, we work with (5.11) to derive an explicit expression for $\mu_{j\boldsymbol{\theta}}^*$ for various combinations of K and $\texttt{rank}(A_{ac}(\boldsymbol{x}))$. The expressions for $\boldsymbol{u}_{\boldsymbol{\theta}}^*(\boldsymbol{x}) = G(\boldsymbol{x})\boldsymbol{\theta} + f(\boldsymbol{x})$ were already derived in chapter 4 and summarized in Table 4.1.

5.5 Computing $\boldsymbol{\Omega}(t)$ using SVD of $A_{ac}(\boldsymbol{x})\boldsymbol{u} = \boldsymbol{b}_{ac}(\boldsymbol{x})$

The aim of this section is to derive expressions for the control $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ and Lagrange multipliers $\mu_{j\boldsymbol{\theta}}^*$ for computing the *instantaneous feasible set* $\boldsymbol{\Omega}(t)$ (5.8). We will use the EQP derived in (5.11) as a surrogate to the original problem (5.4). We consider different cases that can arise based on the number of active constraints (K) and linear independence relations among these constraints to analyze this EQP. Table 5.1 summarizes these cases along with whether point-wise identification and region-based identification is possible when either of these arises in practice. Of special significance are cases D and case E (last two rows of Table 5.1). These correspond to situations when the ego agent has active interactions with at-least two other agents. This table says that in these situations, point-wise estimation does not converge [27] and region-based estimation is the only way to infer information about the underlying parameters. Since in a multiagent system, multiple active interactions are inevitable, this result underscores the importance of region-based estimation for multiagent systems. Nevertheless, even in cases where point-wise estimation is possible, our empirical evidence suggests that region-based estimation converges much faster than point-wise estimation. Hence, our region-based estimation approach is beneficial either way. Next, we analyze these cases one by one.

5.5.1 No active constraints

When no constraint is active, we have $\mu_{j\boldsymbol{\theta}} = 0\ \forall j \in \{1, 2, \cdots, M\}$. This means that $\mathcal{A}(\boldsymbol{u}_{\boldsymbol{\theta}}^*) = \phi$ and $\mathcal{IA}(\boldsymbol{u}_{\boldsymbol{\theta}}^*) = \{1, 2, \cdots, M\}$. From (5.10) we get $\boldsymbol{u}_{\boldsymbol{\theta}}^* = \hat{\boldsymbol{u}}(\boldsymbol{x}) = C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$. From this expression, it is evident that the control $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ and the agent dynamics $\dot{\boldsymbol{x}}$, exhibit a well-defined dependence on $\boldsymbol{\theta}$. Therefore, it *is* possible to do point-wise identification of $\boldsymbol{\theta}$

using RLS, UKF or AO type of estimators (provided that $C(\boldsymbol{x})$ is persistently exciting [76]). However, since point-wise identification is not under the purview of this chapter, we focus on estimating a set where $\boldsymbol{\theta}$ belongs. Referring to 5.4.1, note that the condition for nonnegativity of Lagrange multipliers is satisfied trivially because $\mu_{j\boldsymbol{\theta}} = 0 \; \forall j \in \{1, 2, \cdots, M\}$, so they do not give any information about $\boldsymbol{\theta}$. However, the condition for inactive constraints gives

$$\begin{aligned} \boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j < 0 \; \forall j \in \{1, 2, \cdots, M\} \\ \implies \boldsymbol{a}_j^T (C\boldsymbol{\theta} + \boldsymbol{d}) - b_j < 0 \\ \implies (\boldsymbol{a}_j^T C)\boldsymbol{\theta} < b_j - \boldsymbol{a}_j^T \boldsymbol{d}, \end{aligned} \tag{5.13}$$

which represents a set of M halfspace constraints on $\boldsymbol{\theta}$. In (5.13), we have omitted the dependencies on $\boldsymbol{x}$ and $\boldsymbol{x}_j^o \; \forall j \in \{1, 2, \cdots, M\}$ to keep the notation light. Thus, the *instantaneous feasible set* $\boldsymbol{\Omega}$ is defined using (5.8) and (5.13) as follows

$$\boldsymbol{\Omega} \coloneqq \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | (AC)\hat{\boldsymbol{\theta}} < \boldsymbol{b} - A\boldsymbol{d}\}. \tag{5.14}$$

5.5.2 Exactly one active constraint

When exactly one constraint is active *i.e.* $K = 1$, there is one obstacle that the ego agent "worries" about for collision. Since there are two degrees of freedom in the control and one obstacle to avoid, the ego agent can avoid this obstacle and additionally minimize $\|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2$ with the remaining degree of freedom. This causes $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ to exhibit a well-defined dependence on $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and by extension, on $\boldsymbol{\theta}$. This allows for point-wise identification of $\boldsymbol{\theta}$.

Region-based identification is also possible and can be used to expedite the convergence of point-wise identification. To derive the feasible region where $\boldsymbol{\theta}$ belongs, we need the expression for control $\boldsymbol{u}_{\boldsymbol{\theta}}^*$ and the Lagrange multiplier corresponding to the active constraint. This expression was derived in chapter 4 (4.30) and we recall it here:

$$\begin{aligned} \boldsymbol{u}_{\boldsymbol{\theta}}^* &= V_1 \Sigma_m^{-1} U^T b_i + V_2 V_2^T (C\boldsymbol{\theta} + \boldsymbol{d}) \\ &= \underbrace{V_2 V_2^T C}_{G} \boldsymbol{\theta} + \underbrace{V_1 \Sigma_m^{-1} U^T b_i + V_2 V_2^T \boldsymbol{d}}_{\boldsymbol{f}} \\ &= G\boldsymbol{\theta} + \boldsymbol{f}. \end{aligned} \tag{5.15}$$

From inactive constraints, we get:

$$\begin{aligned} \boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j < 0 \; \forall j \in \mathcal{IA}(\boldsymbol{u}_{\boldsymbol{\theta}}^*) \\ \implies \boldsymbol{a}_j^T (G\boldsymbol{\theta} + \boldsymbol{f}) - b_j < 0 \\ \implies (\boldsymbol{a}_j^T G)\boldsymbol{\theta} < b_j - \boldsymbol{a}_j^T \boldsymbol{f} \\ \implies (A_{inac} G)\boldsymbol{\theta} \prec \boldsymbol{b}_{inac} - A_{inac}\boldsymbol{f}, \end{aligned} \tag{5.16}$$

where $A_{inac}, \boldsymbol{b}_{inac}$ are the rows of $A, \boldsymbol{b}$ indexed by the inactive constraints $\mathcal{IA}(\boldsymbol{u}^*_{\boldsymbol{\theta}})$. To get the Lagrange multiplier $\mu^*_{\boldsymbol{\theta}}$, we use (5.10) and (5.15),

$$\begin{aligned}
\boldsymbol{u}^*_{\boldsymbol{\theta}} &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A^T_{ac}\mu^*_{\boldsymbol{\theta}} \\
\implies \frac{1}{2}A^T_{ac}\mu^*_{i\boldsymbol{\theta}} &= \underbrace{(C - G)}_{\tilde{G}}\boldsymbol{\theta} + \underbrace{(\boldsymbol{d} - \boldsymbol{f})}_{\tilde{\boldsymbol{f}}} \\
&= \tilde{G}\boldsymbol{\theta} + \tilde{\boldsymbol{f}} \\
\implies \mu^*_{\boldsymbol{\theta}} &= 2\frac{A_{ac}(\tilde{G}\boldsymbol{\theta} + \tilde{\boldsymbol{f}})}{\|A^T_{ac}\|^2}.
\end{aligned} \tag{5.17}$$

From non-negativity of $\mu^*_{\boldsymbol{\theta}}$ we get

$$\mu^*_{\boldsymbol{\theta}} \geq 0 \iff -A_{ac}\tilde{G}\boldsymbol{\theta} \preceq A_{ac}\tilde{\boldsymbol{f}}. \tag{5.18}$$

Thus $\boldsymbol{\Omega}$ is defined using (5.16) and (5.18) as follows

$$\boldsymbol{\Omega} := \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | (A_{inac}G)\hat{\boldsymbol{\theta}} \prec \boldsymbol{b}_{inac} - A_{inac}\boldsymbol{f}, -A_{ac}\tilde{G}\hat{\boldsymbol{\theta}} \preceq A_{ac}\tilde{\boldsymbol{f}}\}. \tag{5.19}$$

5.5.3 More than one active constraint but linearly dependent

Let's consider the more general case in which there is more than one constraint active, but all these constraints are linearly dependent on one constraint among them. That is to say there is effectively only one "representative constraint". Consequently, this is similar to the case with just one active obstacle. We now write out expressions for $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ and $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}$. Let $i_1, i_2, \cdots, i_K \in \{1, \cdots, M\}$ be the indices of the active constraints which by definition (5.6) satisfy

$$\begin{gathered}
\begin{bmatrix} \boldsymbol{a}^T_{i_1}(\boldsymbol{x}) \\ \boldsymbol{a}^T_{i_2}(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{a}^T_{i_K}(\boldsymbol{x}) \end{bmatrix} \boldsymbol{u}^*_{\boldsymbol{\theta}} = \begin{bmatrix} b_{i_1}(\boldsymbol{x}) \\ b_{i_2}(\boldsymbol{x}) \\ \vdots \\ b_{i_K}(\boldsymbol{x}) \end{bmatrix} \quad \text{or} \\
A_{ac}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} = \boldsymbol{b}_{ac}(\boldsymbol{x}),
\end{gathered} \tag{5.20}$$

where $\boldsymbol{a}^T_{i_j}(\boldsymbol{x})$ and $b_{i_j}(\boldsymbol{x})$ are defined using (3.13). Let $A_{ac} = U\Sigma V^T$ where

$$\begin{aligned}
U &= \begin{bmatrix} U_1, U_2 \end{bmatrix} \text{ where, } U_1 \in \mathbb{R}^{K\times 1}, U_2 \in \mathbb{R}^{K\times K-1} \\
\Sigma &= \left[\begin{array}{c|c} \Sigma_r & 0^{1\times 1} \\ \hline 0^{K-1\times 1} & 0^{K-1\times 1} \end{array}\right] \\
V &= \begin{bmatrix} V_1, V_2 \end{bmatrix} \text{ where, } V_1, V_2 \in \mathbb{R}^2.
\end{aligned} \tag{5.21}$$

Then $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ is given by (recall from (4.41))

$$\begin{aligned}
\boldsymbol{u}^*_{\boldsymbol{\theta}} &= V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T(C\boldsymbol{\theta}+\boldsymbol{d}) \\
&= \underbrace{V_2V_2^TC}_{G}\,\boldsymbol{\theta} + \underbrace{V_1\Sigma_r^{-1}U_1^T\boldsymbol{b}_{ac} + V_2V_2^T\boldsymbol{d}}_{\boldsymbol{f}} \\
&= G\boldsymbol{\theta} + \boldsymbol{f}. \qquad (5.22)
\end{aligned}$$

Following (5.16), for the inactive constraints, we get:

$$(A_{inac}G)\boldsymbol{\theta} \prec \boldsymbol{b}_{inac} - A_{inac}\boldsymbol{f} \qquad (5.23)$$

To get the Lagrange multipliers $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}$, we use (5.10) and (5.22),

$$\begin{aligned}
\boldsymbol{u}^*_{\boldsymbol{\theta}} &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A_{ac}^T\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} \\
\implies \frac{1}{2}A_{ac}^T\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} &= (C-G)\boldsymbol{\theta} + (\boldsymbol{d}-\boldsymbol{f}) \\
&= \tilde{G}\boldsymbol{\theta} + \tilde{\boldsymbol{f}} \\
\implies A_{ac}^T\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} &= 2(\tilde{G}\boldsymbol{\theta} + \tilde{\boldsymbol{f}}). \qquad (5.24)
\end{aligned}$$

Here, given the fact that $A_{ac}^T \in \mathbb{R}^{2\times K}$ and $\texttt{rank}(A_{ac}^T) = 1$, the Lagrange multipliers are underdetermined. We use the SVD of $A_{ac}^T = \tilde{U}\tilde{\Sigma}\tilde{V}^T$ to derive $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}$. Here

$$\begin{aligned}
\tilde{U} &= \left[\tilde{U}_1, \tilde{U}_2\right] \text{ where } \tilde{U}_1, \tilde{U}_2 \in \mathbb{R}^2 \\
\tilde{\Sigma} &= \left[\begin{array}{c|c} \tilde{\Sigma}_r & 0^{1\times K-1} \\ \hline 0^{1\times 1} & 0^{1\times K-1} \end{array}\right] \\
\tilde{V} &= \left[\tilde{V}_1, \tilde{V}_2\right] \text{ where, } \tilde{V}_1 \in \mathbb{R}^{K\times 1}, \tilde{V}_2 \in \mathbb{R}^{K\times K-1}. \qquad (5.25)
\end{aligned}$$

Then using (5.24), we get

$$\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} = 2\tilde{V}_1\tilde{\Sigma}_r^{-1}\tilde{U}_1^T(\tilde{G}\boldsymbol{\theta} + \tilde{\boldsymbol{f}}) + \tilde{V}_2\boldsymbol{\eta}, \qquad (5.26)$$

where $\boldsymbol{\eta} \in \mathbb{R}^{K-1}$ are floating variables which can assume any value. (See [28] for a complete derivation). From non-negativity of $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}$ we get

$$\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} \succeq 0 \iff -2\tilde{V}_1\tilde{\Sigma}_r^{-1}\tilde{U}_1^T\tilde{G}\boldsymbol{\theta} - \tilde{V}_2\boldsymbol{\eta} \preceq 2\tilde{V}_1\tilde{\Sigma}_r^{-1}\tilde{U}_1^T\tilde{\boldsymbol{f}}. \qquad (5.27)$$

Thus, $\boldsymbol{\Omega}$ is defined using (5.23) and (5.27) as follows

$$\boldsymbol{\Omega} := \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | (A_{inac}G)\hat{\boldsymbol{\theta}} \prec \boldsymbol{b}_{inac} - A_{inac}\boldsymbol{f}, -2\tilde{V}_1\tilde{\Sigma}_r^{-1}\tilde{U}_1^T\tilde{G}\hat{\boldsymbol{\theta}} - \tilde{V}_2\boldsymbol{\eta} \preceq 2\tilde{V}_1\tilde{\Sigma}_r^{-1}\tilde{U}_1^T\tilde{\boldsymbol{f}} \text{ if } \exists\boldsymbol{\eta} \in \mathbb{R}^{K-1}\}$$

5.5.4 Two linearly independent active constraints

Consider the case where there are exactly *two* constraints that are active and linearly independent. In this case, there are as many degrees of freedom in control as the number of independent active obstacles to avoid. Consequently, $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ is completely determined by the active constraints and hence does not depend on $\hat{\boldsymbol{u}}$. We demonstrate this claim as follows. Let $i_1, i_2 \in \{1, 2, \cdots, M\}$ be the indices of the two active constraints. These constraints satisfy (5.20) except that here $A_{ac}(\boldsymbol{x}) \in \mathbb{R}^{2\times 2}$ and $\texttt{rank}(A_{ac}(\boldsymbol{x})) = 2$. This problem is well-posed, its solution is given by

$$\begin{aligned} A_{ac}(\boldsymbol{x})\boldsymbol{u}^*_{\boldsymbol{\theta}} &= \boldsymbol{b}_{ac}(\boldsymbol{x}) \\ \implies \boldsymbol{u}^*_{\boldsymbol{\theta}} &= A^{-1}_{ac}(\boldsymbol{x})\boldsymbol{b}_{ac}(\boldsymbol{x}). \end{aligned} \tag{5.28}$$

where the inverse exists because $\texttt{rank}(A_{ac}(\boldsymbol{x})) = 2$. Since neither A^{-1}_{ac} nor $\boldsymbol{b}_{ac}$ depend on $\boldsymbol{\theta}$ (3.13), $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ also does not depend on $\boldsymbol{\theta}$. Hence, point-wise identification is not possible. For feasible region-based identification, we cannot get information from inactive constraints because that is also contingent upon on $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ depending on $\boldsymbol{\theta}$ (section 5.4.1). Nevertheless, non-negativity of Lagrange multipliers is still useful. From (5.10) and (5.28), we have

$$\begin{aligned} \boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} &= 2A^{-T}_{ac}(C\boldsymbol{\theta} + \boldsymbol{d} - A^{-1}_{ac}\boldsymbol{b}_{ac}) \\ \implies \boldsymbol{\mu}^{ac} \succeq \boldsymbol{0} &\iff -A^{-T}_{ac}C\boldsymbol{\theta} \preceq A^{-T}_{ac}(\boldsymbol{d} - A^{-1}_{ac}\boldsymbol{b}_{ac}). \end{aligned}$$

Thus $\boldsymbol{\Omega}$ is defined as follows

$$\boldsymbol{\Omega} := \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | -A^{-T}_{ac}C\hat{\boldsymbol{\theta}} \preceq A^{-T}_{ac}(\boldsymbol{d} - A^{-1}_{ac}\boldsymbol{b}_{ac})\}. \tag{5.29}$$

5.5.5 More than two linearly-independent active constraints

Finally, let's consider the case where there are $K > 2$ constraints that are active and two of them are linearly independent. The constraints satisfy (5.20) except that here $\texttt{rank}(A_{ac}(\boldsymbol{x})) = 2$. This problem is well-posed albeit overdetermined, its solution is given by

$$\boldsymbol{u}^*_{\boldsymbol{\theta}} = A^{\dagger}_{ac}(\boldsymbol{x})\boldsymbol{b}_{ac}(\boldsymbol{x}). \tag{5.30}$$

where $A^{\dagger}_{ac}$ denotes the Moore-Penrose pseudoinverse which exists because $\texttt{rank}(A_{ac}(\boldsymbol{x})) = 2$. Since $\boldsymbol{u}^*_{\boldsymbol{\theta}}$ is independent of $\boldsymbol{\theta}$, point-wise identification is not possible. For feasible region-based identification, we cannot get information from inactive constraints. Nevertheless, non-negativity of Lagrange multipliers $\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}}$ is still useful. From (5.10) and (5.30),

$$\begin{aligned} A^{\dagger}_{ac}\boldsymbol{b}_{ac} &= \hat{\boldsymbol{u}}_{\boldsymbol{\theta}} - \frac{1}{2}A^{T}_{ac}\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} \\ \implies A^{T}_{ac}\boldsymbol{\mu}^{ac}_{\boldsymbol{\theta}} &= 2(C\boldsymbol{\theta} + \boldsymbol{d} - A^{\dagger}_{ac}\boldsymbol{b}_{ac}). \end{aligned} \tag{5.31}$$

The above linear system for determining $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac}$ is full rank but underdetermined. The SVD of A_{ac}^T is $A_{ac}^T = \tilde{U}\tilde{\Sigma}\tilde{V}^T$ where

$$\begin{aligned}
&\tilde{U} \in \mathbb{R}^{2\times 2}\\
&\tilde{\Sigma} = \left[\tilde{\Sigma}_m \,\middle|\, 0^{2\times K-2}\right] \text{ where, } \tilde{\Sigma}_m \in \mathbb{R}^{2\times 2}\\
&\tilde{V} = \left[\tilde{V}_1, \tilde{V}_2\right] \text{ where, } \tilde{V}_1 \in \mathbb{R}^{K\times 2}, \tilde{V}_2 \in \mathbb{R}^{K\times K-2}.
\end{aligned} \tag{5.32}$$

Then, we can derive the following expression for $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac}$ by solving (5.31)

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac} = 2\tilde{V}_1\tilde{\Sigma}_m^{-1}\tilde{U}^T(C\boldsymbol{\theta} + \boldsymbol{d} - A_{ac}^{\dagger}\boldsymbol{b}_{ac}) + \tilde{V}_2\boldsymbol{\eta}. \tag{5.33}$$

Here $\boldsymbol{\eta} \in \mathbb{R}^{K-2}$ are floating variables which can assume any value. (See [28] for a complete derivation). From non-negativity of $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac}$ we get

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{ac} \succeq 0 \iff -2\tilde{V}_1\tilde{\Sigma}_m^{-1}\tilde{U}^T C\boldsymbol{\theta} - \tilde{V}_2\boldsymbol{\eta} \preceq 2\tilde{V}_1\tilde{\Sigma}_m^{-1}\tilde{U}^T(\boldsymbol{d} - A_{ac}^{\dagger}\boldsymbol{b}_{ac}). \tag{5.34}$$

Thus $\boldsymbol{\Omega}$ is defined using (5.34) as follows

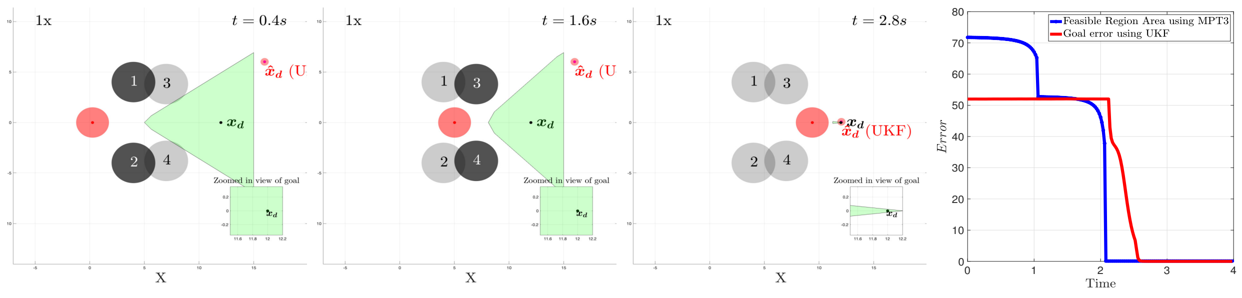
$$\boldsymbol{\Omega} := \{\hat{\boldsymbol{\theta}} \in \mathbb{R}^p | -2\tilde{V}_1\tilde{\Sigma}_m^{-1}\tilde{U}^T C\hat{\boldsymbol{\theta}} - \tilde{V}_2\boldsymbol{\eta} \preceq 2\tilde{V}_1\tilde{\Sigma}_m^{-1}\tilde{U}^T(\boldsymbol{d} - A_{ac}^{\dagger}\boldsymbol{b}_{ac}) \text{ if } \exists \boldsymbol{\eta} \in \mathbb{R}^{K-2}\}. \tag{5.35}$$

5.6 Simulation Results

In this section, we present simulations to demonstrate the effectiveness of our proposed region-based parameter identification. We consider a multiagent system in which the task of each agent is to reach a goal position while avoiding collisions with other agents. The observer's problem is to infer the goal of each agent using the its positions and velocities as measurements. We use a UKF as a baseline to show how our region-based estimator outperforms a UKF.

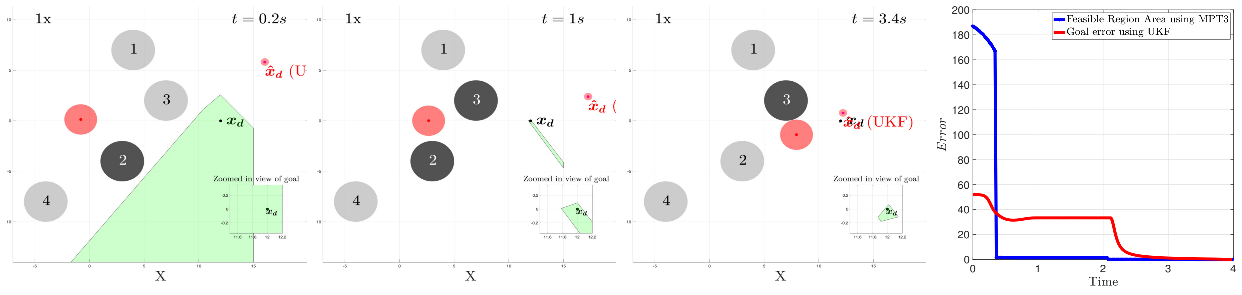
To infer agent goals, the observer must estimate the instantaneous feasible region $\boldsymbol{\Omega}(t)$ using which it will compute the *cumulative* feasible region $\boldsymbol{\Theta}(t)$. Recall from (5.3), that $\boldsymbol{\Theta}(t)$ is computed by taking intersections of all $\boldsymbol{\Omega}(t)$ over time, and then its intersection with a compact set $\boldsymbol{\Theta}_0$ where the $\boldsymbol{\theta}$ is known to belong. For the goal inference problem, the observer can compute the instantaneous set $\boldsymbol{\Omega}(t)$ using the results we derived in sections 5.5.1-5.5.5 by checking how many obstacles are active at a given time. As for $\boldsymbol{\Theta}_0$, we assume that the observer knows some reasonable upper and lower bounds on the location of the agent goals. We first show results for a single agent navigating in an environment consisting of static obstacles. Subsequently, we show results for multiagent inference as well.

In Figs. 5.1(a)-5.1(c), an ego agent (red) is trying to reach its goal $\boldsymbol{x}_d$. The green regions correspond to $\boldsymbol{\Theta}(t)$ computed using the instantaneous feasible regions $\boldsymbol{\Omega}(t)$. As the agent moves to the right, obstacles one and two remain active until $t = 1.1s$. After this, obstacles three and four stay active until $t = 2.1s$. Thus until $t = 2.1s$, since atleast two obstacles are active, point-wise parameter identification is not possible. As is evident from Figs. 5.1(a)-5.1(c), the UKF based estimator (red) does not get updated. However, the green regions



(a) $t = 0.40s$ (b) $t = 1.60s$ (c) $t = 2.80s$ (d) Error Comparison

Figure 5.1: (a)-(c) Goal identification for an agent navigating among static obstacles. Dark discs represent active obstacles. (d) Comparison of region-based identification (blue) with a UKF. Video at `https://youtu.be/jH3mxZhX2mA`

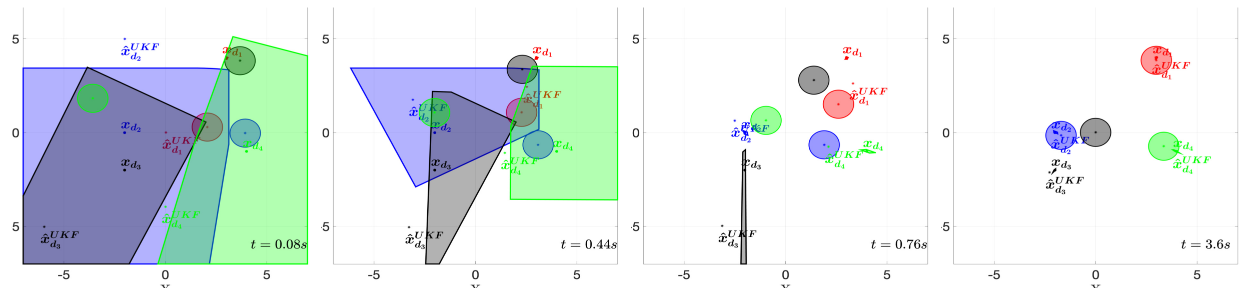


(a) $t = 0.20s$ (b) $t = 1.00s$ (c) $t = 3.40s$ (d) Error Comparison

Figure 5.2: (a)-(c) Goal identification for an agent navigating among static obstacles. Dark discs represent active obstacles. (d) Comparison of region-based identification (blue) with a UKF (red). Video at `https://youtu.be/VthIXiBvjfU`

computed from our approach continue to shrink. The insets in these figures show zoomed in views around the goal to empirically show that the feasible region always includes the goal. We measure $\mu(\boldsymbol{\Theta}(t))$ by computing its area using the MPT3 toolbox in MATLAB. Fig. 5.1(d) shows this area as a function of time and demonstrates the fast convergence as the agent moves, in comparison to the UKF error shown in red.

In Figs. 5.2(a)-5.2(c), we consider a different arrangement of obstacles. Here, until $t = 0.38s$, only one obstacle is active; then until $t = 2.08$s exactly two obstacles are active, then until $t = 3.68s$ exactly one is active, following which all obstacles are inactive. As the agent moves, the region-based estimator converges quickly to the true goal as is evident in the insets in Figs. 5.2(a)-5.2(c). Further, the errors shown in 5.2(d) illustrate how the UKF estimator takes some time to converge while the region based estimator converges much faster to the true goal. Furthermore, the UKF estimator does not respect the feasibility of



(a) $t = 0.08s$ (b) $t = 0.44s$ (c) $t = 0.76s$ (d) $t = 3.60s$

Figure 5.3: (a)-(d) Region-based goal identification for multiagent system. Video at https://youtu.be/r8W1CJ5FJo8

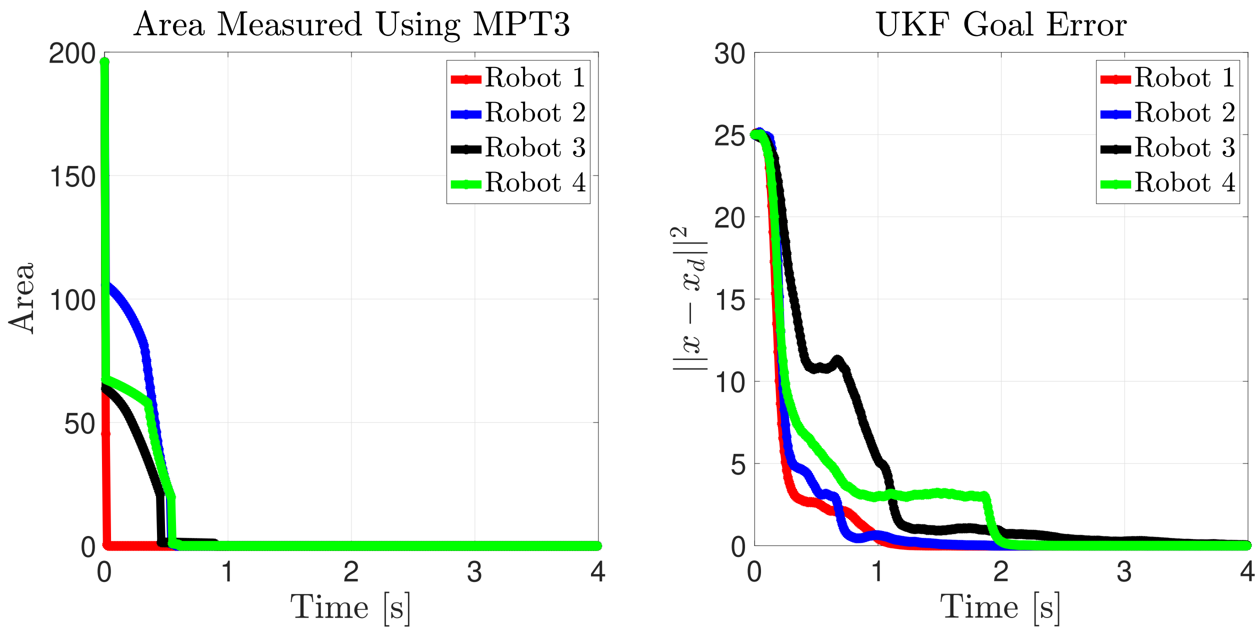


Figure 5.4: Estimation errors of our region-based estimator (left) and UKF (right). The areas shrink around the goal much faster than UKF errors converge to zero.

the goal, it frequently remains outside the feasible region. This is to be expected because the derivation for UKF estimator does not consider non-negativity of Lagrange multipliers.

Finally, we consider a multiagent system in Fig. 5.3 in which we run parallel region-based estimators synchronously. There are four agents shown in different colors, their goals are shown in the same colors as are their feasible regions $\boldsymbol{\Theta}_i(t)\ \forall i \in \{1, 2, 3, 4\}$. Fig. 5.4 compares the convergence of region based estimator (left) to the UKF estimator (right). As is evident, the region-based estimators converge to the true goal much faster than the UKF based estimators.

5.7 Conclusions

In this chapter, we proposed a region-based parameter identification method to compute bounds on parameters of tasks being performed by a multiagent system. This approach works even when exact estimation using point-wise estimators is likely to fail. We used all the KKT conditions to compute the instantaneous feasible set using which we computed contracting sets where the underlying parameters lie. To demonstrate the effectiveness of our method, we showed numerical simulations for a single agent moving amongst static obstacles as well as for a multiagent system. These scenarios show how our region based estimator outperforms a UKF in terms of speed of convergence.

6 Inference Under Noise

In chapter 4, we developed necessary conditions for correct inference of task-parameters of a multiagent system. We developed an RLS estimator that can infer these task parameters in a finite time. To handle situations where these conditions are violated, we developed a feasible-region based estimator in chapter 5. This estimator can infer bounds on the parameters instead of their exact values. While provably-correct, both these algorithms require the observer to perfectly guess the active set for each agent (*i.e.* the observer's estimated active-set must be the same as the agent's true active-set at every time). We defined the observer's guess as:

$$\mathcal{A}^{obs.} := \{j \in \{1, \cdots, M\} \mid |\boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j| < \epsilon\}. \tag{6.1}$$

It goes without saying that this is a brittle assumption, one that can fail in practice due to measurement noise. In particular, if the observer uses numerical differentiation to calculate velocities of agents from their positions, then the amplified noise in velocities can cause $|\boldsymbol{a}_j^T \boldsymbol{u}_{\boldsymbol{\theta}}^* - b_j|$ to exceed ϵ leading to missing out on an active-constraint. This is problematic because our task-parameter estimators in chapters 4 and 5 rely on a correct estimate of the active set for correct parameter inference.

Therefore, we need a practical algorithm that is tolerant to some noise in the positions/velocities of the agents and is not bound by the correct active-set estimate requirement. The algorithms we present in this chapter are developed with this focus in mind. We take recourse to the theory of inverse optimization (IO) to develop batch estimators that can not only handle measurement noise, but also account for a small amount of model mismatch (*i.e.* when the true dynamic model of an agent differs from the observer's hypothesis).

6.1 Introduction

Recent literature on inverse optimization (IO) has developed approaches for estimating cost/constraint parameters of latent parametric optimization problems [80]. While IO algorithms have been explored in finance [81] and operations research [82], they have not been explored as much in robotics. We consider three previously developed IO algorithms from [83], [81] and [84] and reformat these algorithms to perform task inference of individual agents in a multiagent system. Since we assumed that agents dynamics are optimization-based (3.14), these methods are directly applicable to our model. For these IO algorithms, the training data set needed to perform inference consists of pairs of exogenous signals to the agent's forward optimization problem, and the agent's decisions made in response to those signals [85]. In our context, we treat the positions of agents as the signals and the velocities computed by CBF-QPs (3.14) as decisions, to perform task inference.

The outline of this chapter is as follows. In section 6.2, we pose a mathematical formulation of the IO-based task inference problem. The main technical contributions start from section 6.3. We reformat previously developed IO algorithms for the task inference problem. In particular, we derive novel QP-based reformulations of the KKT-loss minimization algorithm of [84] and suboptimality-loss minimization algorithm of [81]. In section 6.4, we present numerical results for inference of controller gains and goal location of each agent in a multiagent system using the presented algorithms. We stress test our our previously developed identifiability conditions against these algorithms *i.e.* more than two active constraints result in incorrect inference and less than two result in success. In this chapter, we show that these results are valid for IO based algorithms as well. In section 6.5, we show experimental results of goal inference using these algorithms on Khepera-4 agents and demonstrate that these algorithms provide accurate estimates of goals even in the presence of perception noise. Finally, we summarize our work in section 6.6 and conclude with directions for future work.

6.2 IO-based Task Inference

We assume the same formalism as in chapters 3-5 for the dynamics of each agent:

$$\begin{aligned} \dot{\boldsymbol{x}} = \boldsymbol{u}^*_{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A\Big(\boldsymbol{x}, \{\boldsymbol{x}^o_j\}\Big)\boldsymbol{u} \le \boldsymbol{b}\Big(\boldsymbol{x}, \{\boldsymbol{x}^o_j\}\Big) \end{aligned} \tag{6.2}$$

where $A \in \mathbb{R}^{M\times 2}$, $\boldsymbol{b} \in \mathbb{R}^M$ given by (3.13). The inference approach we develop in this chapter can be easily extended to perform inference for multiple agents in parallel, so the focus is on the ego agent. The observer tracks this agent's position $\boldsymbol{x}(t)$, its velocity *i.e.* $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$ and additionally, tracks the positions of other agents *i.e.* $\{\boldsymbol{x}^o_j(t)\}_{j=1}^M$. The observer's problem is to infer the task parameter $\boldsymbol{\theta}$ based on the knowledge that the optimal control of the ego agent, $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$, is computed using (6.2) in response to the ego agent's position at $\boldsymbol{x}(t)$ and obstacles' positions at $\{\boldsymbol{x}^o_j(t)\}_{j=1}^M$ (the exogenous signals).

Most IO algorithms operate in the batch-setting. In the context of our problem, this implies that the observer will sample K signal-response pairs over some duration. By signal-response pairs, we refer to tuples of the form $\Bigg(\underbrace{\Big(\boldsymbol{x}(k), \{\boldsymbol{x}^o_j(k)\}_{j=1}^M\Big)}_{signal}, \underbrace{\boldsymbol{u}^*_{\boldsymbol{\theta}}(k)}_{response}\Bigg)$ $\forall k \in \{1, 2, \cdots, K\}$. The observer uses all of these K measurements in one step to compute $\boldsymbol{\theta}$.

6.3 IO-based algorithms for inference

We consider three prominent IO algorithms. Since different authors follow different notations, we have used their ideas but reformatted their notations and language to be compatible with our agent task inference problem.

6.3.1 Predictability Loss Minimization

We know that the observer has access to state-control measurements of the ego agent. The observer can pose a parallel surrogate problem akin to the one being solved by the agent *i.e.* (6.2). In the surrogate problem, the observer treats the unknown parameter $\boldsymbol{\theta}$ as a tunable knob. The observer modulates this knob until the observer's *predicted* controls computed by the solving the surrogate problem in response to state $\boldsymbol{x}$, match with the controls measured from the agent's motion when the agent is also in state $\boldsymbol{x}$. To do this tuning, the observer poses the following problem:

$$\begin{gathered}\hat{\boldsymbol{\theta}}, \{\hat{\boldsymbol{u}}_k\}_{k=1}^{K} = \underset{\boldsymbol{\theta}, \{\boldsymbol{u}_k\}_{k=1}^{K}}{\arg\min} \; \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{u}_k - \boldsymbol{u}^*_{meas}(k)\|^2 \\ \text{such that } \boldsymbol{u}_k \text{ solves (6.2) } \forall k \in \{1, \cdots, K\}.\end{gathered} \tag{6.3}$$

In this problem, the observer is learning both the parameter $\boldsymbol{\theta} \in \mathbb{R}^p$ as well as predictions of the optimal control $\hat{\boldsymbol{u}}_k \in \mathbb{R}^2 \; \forall k \in \{1, 2, \cdots, K\}$. The cost function in (6.3) is the empirical average of the deviations of the predicted controls $\hat{\boldsymbol{u}}_k$ from the measured optimal controls $\boldsymbol{u}^*_{meas}(k)$. This is known as the *predictability loss* and was proposed in [83]. Naturally, it makes sense to minimize this loss only if the observer's predicted controls solve the forward problem (6.2) which is posed as a constraint in (6.3). Since (6.2) is in itself an optimization problem, problem (6.3) is a bi-level optimization which is known to be computationally difficult to solve. [83] proposed a duality based reformulation of a bi-level optimization to a single level problem. Applying their technique to our agent task inference, we replace the constraint in (6.3) with the optimality conditions of (6.2) *i.e.*: $\boldsymbol{u}_k$ solves (6.2) $\iff \exists\, \boldsymbol{\lambda}_k \in \mathbb{R}^M$ such that

1. $\|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}(k))\|^2 \leq h(\boldsymbol{\lambda}_k, \boldsymbol{x}(k), \boldsymbol{\theta})$
2. $\boldsymbol{\lambda}_k \geq \mathbf{0}$
3. $A(\boldsymbol{x}(k))\boldsymbol{u}_k \leq \boldsymbol{b}(\boldsymbol{x}(k))$ [1]

Here $\boldsymbol{\lambda}_k \in \mathbb{R}^M$ for each time instant k, are the M Lagrange multipliers corresponding to the M collision avoidance constraints of the ego agent in (6.2). $h(\boldsymbol{\lambda}_k, \boldsymbol{x}(k), \boldsymbol{\theta})$ is the Lagrange

[1]Technically, this should be $A\Big(\boldsymbol{x}(k), \{\boldsymbol{x}^o_j(k)\}_{j=1}^{M}\Big)\boldsymbol{u}_k \leq \boldsymbol{b}\Big(\boldsymbol{x}(k), \{\boldsymbol{x}^o_j(k)\}_{j=1}^{M}\Big)$ but instead we chose to write $A(\boldsymbol{x}(k))\boldsymbol{u}_k \leq \boldsymbol{b}(\boldsymbol{x}(k))$ to keep notation light.

dual function of (6.2) and is given by

$$h = \left\|\tilde{\boldsymbol{u}} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\right\|^2 + \boldsymbol{\lambda}_k^T(A(\boldsymbol{x}(k))\tilde{\boldsymbol{u}} - \boldsymbol{b}(\boldsymbol{x}(k)))$$
$$\text{where } \tilde{\boldsymbol{u}} = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}(k)) - \frac{1}{2}A^T(\boldsymbol{x}(k))\boldsymbol{\lambda}_k \tag{6.4}$$

Given these three conditions, (6.3) can be re-posed as follows

$$\begin{aligned}\hat{\boldsymbol{\theta}}, \{\hat{\boldsymbol{u}}_k\}_{k=1}^K, \{\hat{\boldsymbol{\lambda}}_k\}_{k=1}^K = \underset{\boldsymbol{\theta},\{\boldsymbol{u}_k\}_{k=1}^K,\{\boldsymbol{\lambda}_k\}_{k=1}^K}{\arg\min} \quad & \frac{1}{K}\sum_{k=1}^K \left\|\boldsymbol{u}_k - \boldsymbol{u}_{meas}^*(k)\right\|^2, \\ \text{subject to} \quad & \left\|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\right\|^2 \le h(\boldsymbol{u}_k, \boldsymbol{\lambda}_k, \boldsymbol{\theta}) \\ & \boldsymbol{\lambda}_k \ge \boldsymbol{0} \\ & A(\boldsymbol{x}(k))\boldsymbol{u}_k \le \boldsymbol{b}(\boldsymbol{x}(k)) \ \forall k \in \{1, \cdots, K\}\end{aligned} \tag{6.5}$$

Even though (6.5) is a single-level reformulation of (6.3), yet it is non-convex because of the first constraint in (6.5). Therefore, it can only be solved using generic nonlinear programming solvers which tend to be slow, especially when the number of measurements K is large. In Sec. 6.4, we present numerical results using (6.5).

6.3.2 KKT Loss Minimization

Another candidate loss that can be used to compute an estimate of risk is the KKT loss [84]. In our context, this loss quantifies the extent to which the observed optimal control violates the KKT conditions of the agent's optimization problem (6.2). Let's recall these conditions.

Stationarity	$\boldsymbol{u}^* = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{1}{2}A^T(\boldsymbol{x})\boldsymbol{\lambda}^*$
Primal Feasibility	$A(\boldsymbol{x})\boldsymbol{u}^* \le \boldsymbol{b}(\boldsymbol{x})$
Dual Feasibility	$\boldsymbol{\lambda}^* \ge \boldsymbol{0}$
Complementary Slackness	$\boldsymbol{\lambda}^* \odot \big(A(\boldsymbol{x})\boldsymbol{u}^* - \boldsymbol{b}(\boldsymbol{x})\big) = \boldsymbol{0}$

Using stationarity and complementary slackness, the KKT loss is defined as follows

$$\begin{aligned} l^{KKT} &= l^{stat.} + l^{comp.\ slack.} \text{ where,} \\ l^{stat.} &= \left\|\boldsymbol{u}^* - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) + \frac{1}{2}A^T(\boldsymbol{x})\boldsymbol{\lambda}^*\right\|^2 \\ l^{comp-slack.} &= \left\|\boldsymbol{\lambda}^* \odot \big(A(\boldsymbol{x})\boldsymbol{u}^* - \boldsymbol{b}(\boldsymbol{x})\big)\right\|^2 \end{aligned} \tag{6.6}$$

Using K observed signal-response pairs $\boldsymbol{\Omega}(k) = (\boldsymbol{x}(k), \boldsymbol{u}^*_{\boldsymbol{\theta}}(k))$, the observer poses an empirical risk minimization problem that queries for $\boldsymbol{\theta}$ and K Lagrange multipliers $\{\boldsymbol{\lambda}_k\}_{k=1}^K \in \mathbb{R}^M$ which minimize the total KKT loss:

$$\begin{aligned} \hat{\boldsymbol{\theta}}, \{\hat{\boldsymbol{\lambda}}_k\}_{k=1}^K = \underset{\boldsymbol{\theta}, \{\boldsymbol{\lambda}_k\}_{k=1}^K}{\arg\min} \quad & \sum_{k=1}^K l^{KKT}(\boldsymbol{\theta}, \boldsymbol{\lambda}_k, \boldsymbol{\Omega}(k)) \\ \text{subject to} \quad & \boldsymbol{\lambda}_k \geq \mathbf{0} \; \forall k \in \{1, \cdots, K\}. \end{aligned} \tag{6.7}$$

In this problem, the decision variables are the task parameter $\boldsymbol{\theta}$ and the Lagrange multipliers $\boldsymbol{\lambda}_k$. The objective function in (6.7) 'softens' the stationarity and complementary slackness conditions. The constraints in (6.7) capture dual feasibility. Given the complicated nature of the loss function in (6.6), the first instinct is to use a generic solver such as `fmincon` to perform inference. However, these solvers tend to be computationally slow, especially when the number of constraints and decision variables is large. We reformulate this as a QP for faster inference.

Reformulating (6.7) as a QP

Define a vector $\boldsymbol{\mu} = (\boldsymbol{\theta}^T, \boldsymbol{\lambda}_1^T, \cdots, \boldsymbol{\lambda}_K^T)^T \in \mathbb{R}^{p+MK}$ which is the decision variable of (6.7). Define matrices $E^{\boldsymbol{\theta}}$ and $E_k^{\boldsymbol{\lambda}}$ appropriately to extract $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}_k$ from $\boldsymbol{\mu}$ as follows:

$$\begin{aligned} \boldsymbol{\theta} &= E^{\boldsymbol{\theta}} \boldsymbol{\mu} \\ \boldsymbol{\lambda}_k &= E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \end{aligned} \tag{6.8}$$

We can already re-pose the constraints in (6.7) as follows:

$$\boldsymbol{\lambda}_k \geq \mathbf{0} \; \forall k \in \{1, \cdots, K\} \iff E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \geq \mathbf{0} \; \forall k \in \{1, \cdots, K\}, \tag{6.9}$$

which are convex by construction. Next, we reformulate the cost function of (6.7). Recall that $l^{KKT} = l^{stat.} + l^{comp.slack.}$. First we focus on $l_k^{stat.}$ from (6.6). We have

$$\begin{aligned} l_k^{stat.} &= \left\| \boldsymbol{u}^*(k) - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}(k)) + \frac{1}{2} A^T(\boldsymbol{x}(k)) \boldsymbol{\lambda}_k \right\|^2 \\ &= \left\| \underbrace{\boldsymbol{u}^*(k) - \boldsymbol{d}(\boldsymbol{x}(k))}_{\boldsymbol{r}_k} - C(\boldsymbol{x}(k))\boldsymbol{\theta} + \frac{1}{2} A^T(\boldsymbol{x}(k)) \boldsymbol{\lambda}_k \right\|^2 \\ &= \left\| \boldsymbol{r}_k - C(\boldsymbol{x}(k)) E^{\boldsymbol{\theta}} \boldsymbol{\mu} + \frac{1}{2} A^T(\boldsymbol{x}(k)) E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \right\|^2 \\ &= \left\| \boldsymbol{r}_k - \tilde{F}_k \boldsymbol{\mu} \right\|^2 \\ &= \boldsymbol{\mu}^T \tilde{F}_k^T \tilde{F}_k \boldsymbol{\mu} - 2\boldsymbol{r}_k^T \tilde{F}_k \boldsymbol{\mu} + \boldsymbol{r}_k^T \boldsymbol{r}_k \end{aligned} \tag{6.10}$$

In the equation above,

$$\tilde{F}_k = C(\boldsymbol{x}(k)) E^{\boldsymbol{\theta}} - \frac{1}{2} A^T(\boldsymbol{x}(k)) E_k^{\boldsymbol{\lambda}} \tag{6.11}$$

Similarly, using (6.6), we reformulate $l_k^{comp.\ slack.}$:

$$\begin{aligned} l_k^{comp.\ slack.} &= \left\| \boldsymbol{\lambda}_k \odot \underbrace{\Big(A(\boldsymbol{x}(k))\boldsymbol{u}^*(k) - \boldsymbol{b}(\boldsymbol{x}(k))\Big)}_{\boldsymbol{w}_k} \right\|^2 \\ &= \left\| E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \odot \boldsymbol{w}_k \right\|^2 \\ &= \boldsymbol{\mu}^T E_k^{\boldsymbol{\lambda}T} W_k E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \end{aligned} \tag{6.12}$$

where $W_k = \text{diag}\Big([\boldsymbol{w}_k^2(1), \boldsymbol{w}_k^2(2), \cdots, \boldsymbol{w}_k^2(M)]\Big)$. Adding (6.10) and (6.12) gives

$$\begin{aligned} l_k^{KKT} &= \boldsymbol{\mu}^T \tilde{F}_k^T \tilde{F}_k \boldsymbol{\mu} + \boldsymbol{\mu}^T E_k^{\boldsymbol{\lambda}T} W_k E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} - 2\boldsymbol{r}_k^T \tilde{F}_k \boldsymbol{\mu} + \boldsymbol{r}_k^T \boldsymbol{r}_k \\ &= \boldsymbol{\mu}^T \underbrace{\left(\tilde{F}_k^T \tilde{F}_k + E_k^{\boldsymbol{\lambda}T} W_k E_k^{\boldsymbol{\lambda}}\right)}_{Q_k} \boldsymbol{\mu} + \underbrace{\left(-2\boldsymbol{r}_k^T \tilde{F}_k\right)}_{\boldsymbol{v}_k^T} \boldsymbol{\mu} + \underbrace{\boldsymbol{r}_k^T \boldsymbol{r}_k}_{s_k} \\ &= \boldsymbol{\mu}^T Q_k \boldsymbol{\mu} + \boldsymbol{v}_k^T \boldsymbol{\mu} + s_k \end{aligned} \tag{6.13}$$

Thus, the total loss over all K measurements from the cost function of (6.7) is obtained by summing (6.13) as follows:

$$\begin{aligned} \sum_{k=1}^K l_k^{KKT} &= \sum_{k=1}^K \left(\boldsymbol{\mu}^T Q_k \boldsymbol{\mu} + \boldsymbol{v}_k^T \boldsymbol{\mu} + s_k\right) \\ &= \boldsymbol{\mu}^T \underbrace{\left(\sum_{k=1}^K Q_k\right)}_{Q} \boldsymbol{\mu} + \underbrace{\left(\sum_{k=1}^K \boldsymbol{v}_k^T\right)}_{\boldsymbol{v}^T} \boldsymbol{\mu} + \underbrace{\left(\sum_{k=1}^K s_k\right)}_{s} \\ &= \boldsymbol{\mu}^T Q \boldsymbol{\mu} + \boldsymbol{v}^T \boldsymbol{\mu} + s \end{aligned} \tag{6.14}$$

From (6.14), it is evident that the total KKT loss in (6.7) is indeed in quadratic in the decision variables $\boldsymbol{\mu}$. Hence, we can re-pose (6.7) using the reformulated cost in (6.14) and constraints in (6.9) as the following QP:

$$\begin{aligned} \hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T Q \boldsymbol{\mu} + \boldsymbol{v}^T \boldsymbol{\mu} \\ \text{subject to} \quad & E_k^{\boldsymbol{\lambda}} \boldsymbol{\mu} \geq \mathbf{0} \ \forall k \in \{1, \cdots, K\}. \end{aligned} \tag{6.15}$$

(6.15) is thus a QP-based reformulation of (6.7), and is amenable to faster solutions using existing QP-solvers.

6.3.3 Sub-optimality Minimization

[86] proposed data-driven techniques to infer unobservable parameters of models describing Nash equilibria in game theory. They combined ideas from inverse optimization with

variational inequalities to develop data-driven techniques for estimating the parameters of these models from observed equilibria. Following their approach, we show how to reformat (6.2) so that we can leverage their approach for inferring $\boldsymbol{\theta}$. Consider a general optimization problem

$$\begin{aligned} &\text{minimize } F_{\boldsymbol{\theta}}(\boldsymbol{\xi}) \\ &\text{subject to } \boldsymbol{\xi} \in \mathcal{X} \end{aligned} \tag{6.16}$$

Here $\boldsymbol{\theta}$ are parameters of the convex cost function F known to the agent solving (6.16) and $\mathcal{X} \subset \mathbb{R}^n$ is a convex set.

Assumption 1. *$\mathcal{X}$ can be represented as the intersection of a small number of conic inequalities in standard form, $\mathcal{X} = \{\boldsymbol{\xi} \in \mathbb{R}^n | G\boldsymbol{\xi} = \boldsymbol{h}, \boldsymbol{\xi} \geq \boldsymbol{0}\}$.*

Assumption 2. *$\mathcal{X}$ satisfies a Slater's condition.*

The following result from [86] characterizes necessary and sufficient conditions for $\hat{\boldsymbol{\xi}}$ to be an ϵ-optimal solution to (6.16):

Theorem 1. *[86] Assuming $\mathcal{X}$ satisfies 1-2, an observed decision $\hat{\boldsymbol{\xi}}$ is an ϵ-optimal solution to (6.16) if and only if $\exists$ $\boldsymbol{y}$ such that $G^T\boldsymbol{y} \leq \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\hat{\boldsymbol{\xi}}}$ and $\hat{\boldsymbol{\xi}}^T \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\hat{\boldsymbol{\xi}}} - \boldsymbol{h}^T\boldsymbol{y} \leq \epsilon$*

The inverse problem requires an observer to infer $\boldsymbol{\theta}$ based on the knowledge that the agent solves (6.16) using K samples of $\hat{\boldsymbol{\xi}}$ which are known to be ϵ-optimal solutions to (6.16). Since the observer does not know the suboptimality of a decision ϵ, the observer poses a suboptimality minimization problem querying for $\epsilon_k, \boldsymbol{y}_k, \boldsymbol{\theta}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{\theta}}, \{\hat{\epsilon}\}_{k=1}^K, \{\hat{\boldsymbol{y}}_k\}_{k=1}^K = & \underset{\boldsymbol{\theta}, \{\epsilon\}_{k=1}^K, \{\boldsymbol{y}_k\}_{k=1}^K}{\arg\min} \quad \sum_{k=1}^K \epsilon_k^2 \\ & \text{subject to} \qquad G^T\boldsymbol{y}_k \leq \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\hat{\boldsymbol{\xi}}_k} \\ & \qquad\qquad\qquad \hat{\boldsymbol{\xi}}_k^T \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\hat{\boldsymbol{\xi}}} - \boldsymbol{h}^T\boldsymbol{y}_k \leq \epsilon_k \ \forall k \in \{1, \cdots, K\} \end{aligned} \tag{6.17}$$

Notice that the cost function in (6.17) penalizes the suboptimality of observed solutions $\boldsymbol{\xi}_k$ whereas the constraints are necessary and sufficient conditions for observed decisions $\boldsymbol{\xi}_k$ to be ϵ_k-optimal solutions of (6.16) based on theorem 1.

Agent task inference using (6.17)

In the context of inferring task parameters of a agent, recall that our ego agent solves (6.2) where the task parameters are involved in the cost function. This optimization problem is a specific instance of the general problem in (6.16), therefore we need to reformat the ego-agent's optimization problem (6.2) to (6.16) to facilitate inference of $\boldsymbol{\theta}$ using (6.17). Recall that controls in (6.2) are required to satisfy safety constraints

$$A(\boldsymbol{x})\boldsymbol{u} \leq \boldsymbol{b}(\boldsymbol{x}). \tag{6.18}$$

On the other hand, the feasible set $\mathcal{X}$ in (6.16) is required to satisfy assumptions 1-2. To reformat (6.18) so that these assumptions are satisfied, define $\boldsymbol{u}_1 \geq \mathbf{0}$ and $\boldsymbol{u}_2 \geq \mathbf{0}$ such that $\boldsymbol{u} = \boldsymbol{u}_1 - \boldsymbol{u}_2$. One choice satisfying these requirements is $\boldsymbol{u}_1 = \boldsymbol{u} + |\boldsymbol{u}|, \boldsymbol{u}_2 = |\boldsymbol{u}|$. Define the following variables,

$$\boldsymbol{z} = (\boldsymbol{b}(\boldsymbol{x}) - A(\boldsymbol{x})\boldsymbol{u}) \in \mathbb{R}^M \tag{6.19}$$
$$\boldsymbol{\xi} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{z}) \tag{6.20}$$

Then, it is evident that

$$\begin{aligned} A(\boldsymbol{x})\boldsymbol{u} \leq \boldsymbol{b}(\boldsymbol{x}) &\iff A(\boldsymbol{x})(\boldsymbol{u}_1 - \boldsymbol{u}_2) \leq \boldsymbol{b}(\boldsymbol{x}) \\ &\iff A(\boldsymbol{x})(\boldsymbol{u}_1 - \boldsymbol{u}_2) + \boldsymbol{z} = \boldsymbol{b}(\boldsymbol{x}), \boldsymbol{z} \geq \mathbf{0} \end{aligned} \tag{6.21}$$

Define G and $\boldsymbol{h}$ required in assumption 1 as follows

$$\begin{aligned} G(\boldsymbol{x}) &:= \left[A(\boldsymbol{x}), -A(\boldsymbol{x}), I^M\right] \\ \boldsymbol{h}(\boldsymbol{x}) &:= \boldsymbol{b}(\boldsymbol{x}) \end{aligned} \tag{6.22}$$

where I^M is the $M \times M$ identity matrix. Then, it is easy to verify that $G(\boldsymbol{x})\boldsymbol{\xi} = \boldsymbol{h}(\boldsymbol{x})$. We have

$$\begin{aligned} &A(\boldsymbol{x})\boldsymbol{u} \leq \boldsymbol{b}(\boldsymbol{x}) \\ &\iff \boldsymbol{u}_1 \geq \mathbf{0}, \boldsymbol{u}_2 \geq \mathbf{0}, \boldsymbol{z} \geq \mathbf{0}, A(\boldsymbol{x})(\boldsymbol{u}_1 - \boldsymbol{u}_2) + \boldsymbol{z} = \boldsymbol{b}(\boldsymbol{x}) \\ &\iff \boldsymbol{\xi} \geq 0 \text{ and } G(\boldsymbol{x})\boldsymbol{\xi} = \boldsymbol{h}(\boldsymbol{x}) \end{aligned} \tag{6.23}$$

The cost function in (6.2) is

$$F_{\boldsymbol{\theta}}(\boldsymbol{u}) = \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \tag{6.24}$$

Define a matrix $E^{\boldsymbol{u}}$ appropriately so that $\boldsymbol{u} = E^{\boldsymbol{u}}\boldsymbol{\xi}$. The cost then becomes a function of $\boldsymbol{\xi}$:

$$\begin{aligned} F_{\boldsymbol{\theta}}(\boldsymbol{\xi}) &= \|E^{\boldsymbol{u}}\boldsymbol{\xi} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \implies \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi}) &= 2E^{\boldsymbol{u}T}(E^{\boldsymbol{u}}\boldsymbol{\xi} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})) \\ &= 2E^{\boldsymbol{u}T}(E^{\boldsymbol{u}}\boldsymbol{\xi} - C(\boldsymbol{x})\boldsymbol{\theta} - \boldsymbol{d}(\boldsymbol{x})) \end{aligned}$$

Thus, with (6.2) re-posed as (6.16), the signal-response pairs for inference are no longer $(\boldsymbol{x}(k), \boldsymbol{u}^*(k))$, rather they are $(\boldsymbol{x}(k), \boldsymbol{\xi}^*(k))$. To construct $\boldsymbol{\xi}^*(k)$ from $\boldsymbol{u}^*(k)$, define

$$\begin{aligned} \boldsymbol{u}_1(k) &:= \boldsymbol{u}^*(k) + |\boldsymbol{u}^*(k)| \\ \boldsymbol{u}_2(k) &:= |\boldsymbol{u}^*(k)| \\ \boldsymbol{z}(k) &:= \boldsymbol{b}(\boldsymbol{x}(k)) - A(\boldsymbol{x}(k))\boldsymbol{u}^*(k) \\ \implies \boldsymbol{\xi}^*(k) &:= (\boldsymbol{u}_1(k), \boldsymbol{u}_2(k), \boldsymbol{z}(k)). \end{aligned} \tag{6.25}$$

Note that $\boldsymbol{u}_1(k), \boldsymbol{u}_2(k) \geq \mathbf{0}$ because of the way we defined them. $|\boldsymbol{u}^*(k)|$ is the absolute value of $\boldsymbol{u}^*(k)$ taken element wise. $\boldsymbol{z}^k \geq \mathbf{0}$ because the measured controls $\boldsymbol{u}^*(k)$ must satisfy the safety constraints (6.18). Thus, $\boldsymbol{\xi}(k) \geq \mathbf{0}$ as is required by assumption 1. Assumption 2 is automatically satisfied by all QPs [77], and by extension, by (6.2). Once again, (6.17), in the form presented, can only be solved by general NLP solvers. In the next section, we show how to reformulate this as a QP.

Reformulating (6.17) as a QP

Define a vector $\boldsymbol{\mu} = (\boldsymbol{\theta}^T, \epsilon_1, \cdots, \epsilon_K, \boldsymbol{y}_1^T, \cdots, \boldsymbol{y}_K^T)^T$ which is the decision variable of (6.17). Define matrices $E^{\boldsymbol{\theta}}$, E_k^{ϵ} and $E_k^{\boldsymbol{y}}$ appropriately to extract $\boldsymbol{\theta}$, ϵ_k and $\boldsymbol{y}_k$ from $\boldsymbol{\mu}$ as follows:

$$\begin{aligned} \boldsymbol{\theta} &= E^{\boldsymbol{\theta}}\boldsymbol{\mu} \\ \boldsymbol{\epsilon}_k &= E_k^{\epsilon}\boldsymbol{\mu} \\ \boldsymbol{y}_k &= E_k^{\boldsymbol{y}}\boldsymbol{\mu} \end{aligned} \tag{6.26}$$

Based on these definitions, we can already re-pose the cost function in (6.17) as follows:

$$\sum_{k=1}^{K} \epsilon_k^2 = \sum_{k=1}^{K} \|E_k^{\epsilon}\boldsymbol{\mu}\|^2 = \boldsymbol{\mu}^T \underbrace{\left(\sum_{k=1}^{K} E_k^{\epsilon T} E_k^{\epsilon}\right)}_{Q} \boldsymbol{\mu} \tag{6.27}$$

The constraints can be re-posed as

$$\begin{aligned} & G^T\boldsymbol{y}_k \leq \nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\boldsymbol{\xi}^*(k)} \\ \iff\ & G^T(\boldsymbol{x}(k))E_k^{\boldsymbol{y}}\boldsymbol{\mu} \leq 2E^{\boldsymbol{u}T}(E^{\boldsymbol{u}}\boldsymbol{\xi}^*(k) - C(\boldsymbol{x}(k))E^{\boldsymbol{\theta}}\boldsymbol{\mu} - \boldsymbol{d}(\boldsymbol{x}(k)) \end{aligned}$$

After some rearrangement, this can be re-posed as an inequality in $\boldsymbol{\mu}$:

$$H_1\boldsymbol{\mu} \leq \boldsymbol{g}_1 \tag{6.28}$$

Here $H_1, \boldsymbol{g}_1$ are

$$\begin{aligned} H_1 &= G^T(\boldsymbol{x}(k))E_k^{\boldsymbol{y}} + 2E^{\boldsymbol{u}T}C(\boldsymbol{x}(k))E^{\boldsymbol{\theta}} \\ \boldsymbol{g}_1 &= 2E^{\boldsymbol{u}T}(E^{\boldsymbol{u}}\boldsymbol{\xi}^*(k) - \boldsymbol{d}(\boldsymbol{x}(k))) \end{aligned} \tag{6.29}$$

Similarly the second constraint in (6.17) can be re-posed as

$$\begin{aligned} & \boldsymbol{\xi}^{*T}\nabla_{\boldsymbol{\xi}} F_{\boldsymbol{\theta}}(\boldsymbol{\xi})|_{\boldsymbol{\xi}^*(k)} - \boldsymbol{h}^T\boldsymbol{y}_k \leq \epsilon_k \\ \iff\ & 2\boldsymbol{\xi}^{*T}(k)E^{\boldsymbol{u}T}\Big(E^{\boldsymbol{u}}\boldsymbol{\xi}^*(k) - C(\boldsymbol{x}(k))E^{\boldsymbol{\theta}}\boldsymbol{\mu} - \boldsymbol{d}(\boldsymbol{x}(k)\Big) - \boldsymbol{h}^T(\boldsymbol{x}(k))E_k^{\boldsymbol{y}}\boldsymbol{\mu} \leq E_k^{\epsilon}\boldsymbol{\mu} \end{aligned}$$

which after some rearrangement gives

$$H_2\boldsymbol{\mu} \leq \boldsymbol{g}_2 \tag{6.30}$$

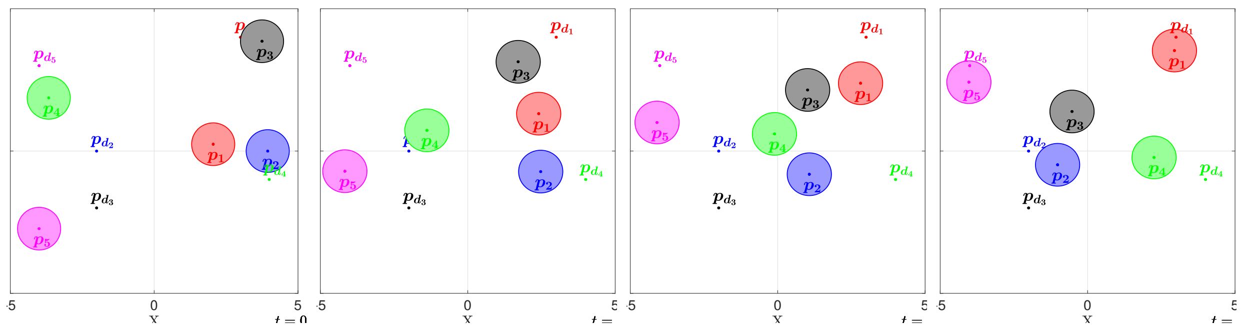
Here $H_2, \boldsymbol{g}_2$ are

$$\begin{aligned} H_2 &= -\Big(2\boldsymbol{\xi}^{*T}(k)E^{\boldsymbol{u}T}C(\boldsymbol{x}(k))E^{\boldsymbol{\theta}} + 2\boldsymbol{\xi}^{*T}(k)E^{\boldsymbol{u}T}\boldsymbol{h}^T(\boldsymbol{x}(k))E_k^{\boldsymbol{y}} + E_k^{\epsilon}\Big) \\ \boldsymbol{g}_2 &= -2\boldsymbol{\xi}^{*T}(k)E^{\boldsymbol{u}T}E^{\boldsymbol{u}}\boldsymbol{\xi}^* + 2\boldsymbol{\xi}^{*T}(k)E^{\boldsymbol{u}T}\boldsymbol{d}(\boldsymbol{x}(k)) \end{aligned} \tag{6.31}$$

Combining the quadratic cost in (6.27) with the affine constraints (6.28) and (6.30), we arrive at the following QP-based formulation:

$$
\begin{aligned}
\hat{\boldsymbol{\mu}} = \arg\min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T Q \boldsymbol{\mu} \\
\text{subject to} \quad & H_1 \boldsymbol{\mu} \leq \boldsymbol{g}_1 \ \forall k \in \{1, 2, \cdots, K\} \\
& H_2 \boldsymbol{\mu} \leq \boldsymbol{g}_2 \ \forall k \in \{1, 2, \cdots, K\}.
\end{aligned}
\tag{6.32}
$$

Thus, the convexity of this formulation allows for faster inference on batch-data by exploiting existing QP solvers.



(a) $t = 0.06s$ (b) $t = 0.6s$ (c) $t = 1.2s$ (d) $t = 2.4s$

Figure 6.1: Multiagent goal ientification.

6.4 Simulation Results

We provide numerical results for inference of task parameters in a multiagent system using (6.5), (6.15) and (6.32). We consider the task where each agent is trying to reach a goal position while avoiding collisions with every other agent. We will use these algorithms to estimate the desired goal $\boldsymbol{x}_d$ and proportional gain k_p for each agent by using their positions and velocities collected over some time. We also describe situations where these estimators will fail to identify these parameters, which occur when the measurements fail to satisfy certain "richness" criteria.

6.4.1 Gain and Goal inference in a multiagent setting

In Fig. 6.1, we have five agents located in a 5m × 5m area. Each agent has a unique color and is required to reach a goal position denoted with the same color while staying safe. The agents use $\boldsymbol{u}^{task}_{\boldsymbol{p}^*_d} = -k_p(\boldsymbol{p} - \boldsymbol{p}^*_d)$ as a nominal task-based controller in (6.2). The inference algorithm must compute estimates of $\hat{\boldsymbol{p}}_d$ and gain $\boldsymbol{k}^*_p$ for each agent. Table 6.1 shows the gain reconstruction errors for different algorithms. and Table 6.2 presents the goal-reconstruction errors. As is evident from the table, all three algorithms produce 0 error, meaning that they are able to correctly estimate both the correct goal and gain for each agent.

Table 6.1: Gain Estimation Errors $|k_p - k_p^*|$

agent ID	Predict. loss algorithm	KKT loss algorithm	Suboptimality loss algorithm
1	0.0000	0.0000	0.0000
2	0.0001	0.0000	0.0000
3	0.0002	0.0000	0.0000
4	0.0000	0.0000	0.0000
5	0.0000	0.0000	0.0000

Table 6.2: Goal Estimation Errors $\|\hat{\boldsymbol{p}}_d - \boldsymbol{p}_d^*\|$ in [m]

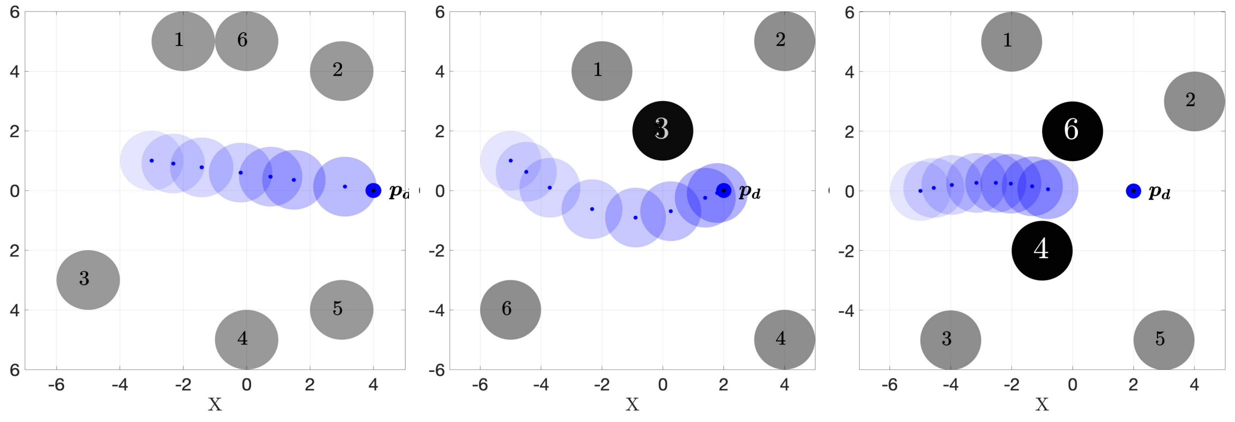
agent ID	Predict. loss algorithm	KKT loss algorithm	Suboptimality loss algorithm
1	0.0000	0.0000	0.0000
2	0.0005	0.0000	0.0000
3	0.0012	0.0000	0.0000
4	0.0002	0.0000	0.0000
5	0.0001	0.0000	0.0000

6.4.2 Inference when Identifiability Conditions are Violated

In chapter 4, we used persistency of excitation analysis to identify conditions where conventional estimators such as a Kalman filter would fail to infer task parameters $\boldsymbol{\theta}$ using position and velocity measurements of a agent. Given the new algorithms considered in the current work, we want to stress test these conditions against these new algorithms. To keep this chapter self-contained, we present these conditions only at a high-level. For an intuitive understanding, we present results for a single agent navigating amongst static obstacles. The inference problem is to determine the goal of this agent and its controller gain using its position and velocity measurements. The first condition, stated in Theorem 2 identifies a situation when inference will be successful.

Theorem 2. *[27] If $\forall t \in [0, T]$, no constraint is active, then the observer can always estimate the goal and gain using $\boldsymbol{x}(t), \boldsymbol{u}^*(\boldsymbol{x}(t))$ $\forall t \in [0, T]$ (unless the agent is not already at goal).*

By an active constraint, we refer to the constraint of (6.2) for which equality holds. Intuitively, this refers to an active interaction with an obstacle, because that obstacle is a potential risk of collision. This result says that if the agent does not have active interactions with any obstacles, then the position-velocity measurements of the agent will be so rich that



(a) No obstacle is active (b) One obstacle is active (c) Two obstacles are active

Figure 6.2: As the agent navigates to its goal, the success of inference depends on the number of obstacles that it actively interacts with. In Fig (c) there are two active interactions (dark), so per Theorem 3, goal/gain inference will fail.

Act. Obs.	**Pred. loss**	**KKT loss**	**Subopt. loss**	**UKF**	**RLS-AO**
0	0.0003 ± 0.001	0.000 ± 0.000	0.000 ± 0.000	0.024 ± 0.000	0.022 ± 0.002
1	0.031 ± 0.046	0.000 ± 0.000	0.000 ± 0.000	0.29± 0.307	0.021 ± 0.018
2	6.37 ± 2.448	5.885 ± 3.778	4.337 ± 2.236	5.657 ± 0.307	5.625 ± 0.018

Table 6.3: Goal inference errors when the number of active obstacles is 0 (top row), 1 (middle row) and 2 (bottom row).

goal/gain inference will always be successful. One situation when this occurs is when the obstacles are far enough from the agent such that the agent can freely use $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}$ as shown in Fig. 6.2(a)). We evaluate IO algorithms presented in this chapter and compare them with a UKF and an Adaptive observer (from [27]) to stress-test this theorem for inferring the location of the goal. Table 6.3 shows the numerical results for this situation. We show the mean and standard deviation of goal estimation errors averaged over ten trials. In each trial, we varied the final goal position of the agent and locations of obstacles while keeping the initial position of the agent identical. As is evident from the errors in the first row, goal inference is always successful for all estimators. Let us look at a situation where inference will fail.

Theorem 3. *[27] If $\forall t \in [0, T]$, two or more than two constraints are active, then the observer* ***cannot*** *estimate either the goal or the gain, using $\boldsymbol{x}(t), \boldsymbol{u}^*(\boldsymbol{x}(t))$ $\forall t \in [0, T]$.*

In this situation, the agent has active interactions with either two or more than two obstacles while navigating towards its goal. Consequently, this theorem says that the position-velocity measurements are not rich enough to facilitate correct inference of either the goal or the gain. Intuitively, this occurs because of the following. We know that the agent has two degrees of freedom in its control and when there are two or more obstacles to actively avoid, both of those degrees of freedom are exhausted in repelling the obstacles leaving no freedom dedicated for the task. Therefore, the motion that the observer measures does not have any explicit information about the task, which is why task inference will fail. Figure 6.2(c) shows an example simulation where the agent is moving towards its goal. There are several obstacles, two of these, shown in black, are the ones active during the agent's motion. Hence, for this simulation, the inference algorithm will not be able to deduce the goal or the gain. This is indeed evident from the large errors in the third row in Table 6.3.

In the middle of the spectrum is the situation where there is exactly one active obstacle to avoid. Figure 6.2(b) shows a simulation where the agent is navigating towards its goal, and obstacle 3 remains active during the agent's journey to the goal. Since the agent can use one degree of freedom to avoid that obstacle, and use the remaining degree of freedom to perform the task, its position-velocity measurements **will** have information about the task. Therefore, it **is** possible to infer either the goal or the gain. The second row in Table 6.3 shows the errors in estimating the goal. These errors are negligible, and comparable to those in the first row, demonstrating that goal inference is successful.

6.5 Experimental Results

We also performed physical experiments with three Khepera agents to analyze the effect of noise in hardware and perception sensing on the estimation of goals. We conducted experiments with three agents and performed a total of eight runs in which the initial positions of the agents, their gains and safety margins were varied, but their individual respective goal locations were kept same through the runs. Table 6.4 reports the errors of these individual runs for the three algorithms along with the mean and standard deviations. The videos of these experiments can be found at `https://bit.ly/3s7yovL`. From the numerical values of the errors, it is evident that all algorithms are able to obtain estimates of the goals that are within a maximum of $5cms$ of the true goals.

6.6 Conclusions

We considered the problem of inference of parameters of tasks being performed by agents in a multiagent system. In such a system, agents use optimization based controllers to mediate between task satisfaction and collision avoidance, thus the trajectories they take, reflect how a purely task-based motion is warped to ensure safety. This makes inference of task parameters non-trivial. We considered several IO algorithms to solve this problem in a batch setting and demonstrated how accurate estimates of underlying parameters can be reconstructed. Furthermore, we derived QP based reformulations of the KKT-loss minimization

Table 6.4: Goal Errors in Experiments $\|\hat{\boldsymbol{p}}_d - \boldsymbol{p}_d^*\|$ in [m]

Run Number	Predict. loss algorithm	KKT loss algorithm	Suboptimality loss algorithm
1	0.0166	0.0073	0.0566
2	0.0502	0.0185	0.1394
3	0.0171	0.0079	0.0133
4	0.0292	0.0218	0.0178
5	0.0047	0.0046	0.0332
6	0.0057	0.0058	0.0408
7	0.0043	0.0041	0.0951
8	0.0053	0.0050	0.0281
Mean	**0.0166**	**0.0094**	**0.0530**
Std.	**0.0151**	**0.0064**	**0.0406**

and suboptimality minimization algorithms. Finally, using our previously derived criteria for successful inference, we demonstrated that these IO algorithms may fail to identify the correct underlying task parameters whenever the ego agent interacts with two or more obstacles. In the next chapter, we extend this work to simultaneously learn parameters of the cost function as well as constraints of the ego-agent's forward problem.

7 Simultaneously Inferring Objectives and Constraints

In chapters 4 - 6, we developed algorithms for inferring the task-parameters $\boldsymbol{\theta}$ for each agent in the system. These parameters are associated with the nominal task-based control $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and manifest in the cost function of that agent's dynamics:

$$\begin{aligned} \boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & A\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big)\boldsymbol{u} \le \boldsymbol{b}_{\gamma, D_s}\Big(\boldsymbol{x}, \{\boldsymbol{x}_j^o\}_{j=1}^M\Big) \end{aligned} \tag{7.1}$$

In these chapters, we assumed that the observer knows the parameters of collision-avoidance constraints (γ, D_s) a-priori and uses them to infer $\boldsymbol{\theta}$. In particular, the algorithms in chapters 4, 5 required their values to compute the active-set. However, in practice, their values will also be unknown to the observer because these parameters encode individual agent preferences for ensuring collision free motions. Simultaneously inferring constraint parameters γ, D_s with task parameters $\boldsymbol{\theta}$ can be challenging because an agent's observed trajectory only partially manifests its long-term task; it also contains adjustments made by the agent to ensure collision avoidance with other agents and obstacles in the environment. Since an observer would have no means to determine the magnitude of these adjustments, it is difficult to isolate the task-oriented component from the observed motion.

Given this limitation of our previous algorithms, we ask how can an observer simultaneously infer both the cost-function parameters $\boldsymbol{\theta}$ and the constraint parameters γ, D_s by observing each agent. We build on chapter 6 to fill this gap. We develop two robust mixed-integer programming algorithms that infer the task and safety related parameters of this optimization problem from the positions and velocities of the agents. In addition to modeling inter-agent collision avoidance constraints as done in (7.1), we also model collision-avoidance constraints between the agents and obstacles/walls. Thus, our inference algorithms enable us to infer safety constraints with obstacles/walls in the environment. We validate these algorithms on synthetic datasets using parameter estimation errors, displacement errors and computation time as metrics. We further test these algorithms on a dataset of real human trajectories. We show that the learned parameters capture the true underlying pedestrian

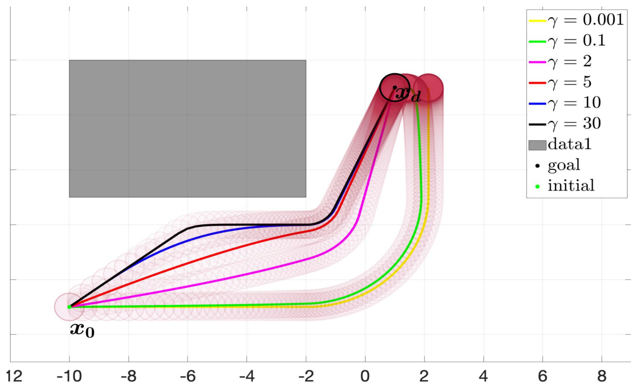


Figure 7.1: Sample trajectories produced by (7.2) with one rectangular obstacle as a constraint. Increasing γ makes the trajectory less conservative and makes it follow the reference controller more closely.

dynamics by rolling out the learned model and showing similarity between the ground truth trajectories and the reconstructed trajectories.

7.1 Application to inferring human intentions

As mentioned, we demonstrate the applicability of our proposed inference algorithms for learning intents (*i.e.* goals, safety margins) of multiple humans that interact among one another and with the walls/obstacles in a close-proximity environment. We identify a set of behavior and safety-related parameters for each human's dynamics model that describe their (i) long-term intention (such as goal, desired velocity etc.), and (ii) collision avoidance behavior around other agents or walls, (e.g. underlying safety margin, aggressiveness etc). Our chosen optimization-based dynamics modeling paradigm models cooperation among different agents to achieve collision-free behavior. Using these algorithms, a control engineer can the predict future behaviors of humans for generating safe robot motions.

We develop two mixed-integer quadratic programming (MIQP) based algorithms to learn the parameters of the CBF-QP based model describing each agent's behavior from its observed trajectories. This allows for individually learning the parameters of the task-oriented component and the collision avoidance-related component of the agent. We use stationarity error and prediction error as heuristics to learn parameters that best explain the observed measurements. Our proposed algorithms are decentralized, robust to model mismatch and do not require long training times (we show empirical results to support these claims). Additionally, by virtue of being model-based, the parameters we learn have an intuitive physics-based interpretation. Because of this feature, our inference approach enables reliable prediction that is generalizable to unseen scenarios, e.g. it describes how an agent will behave when entering a new scenario with a different set of nearby obstacles. Lastly, in addition to learning these parameters, our algorithm automatically learns which obstacles/agents in the environment influence the ego agent's dynamics from its velocity. Identification of these interest entities is what makes our algorithms integer-programs. We validate these algorithms on several simulated datasets and show small values of parameter reconstruction errors and small average and final displacement errors. Next, we evaluate them on a pedestrian dataset called THöR [30] which contains human motion trajectories recorded in a controlled indoor experiment at Örebro University. These trajectories exhibit social interactions that occur in populated spaces like offices, thus making them suitable for evaluating our algorithms. Our results show that the learned parameters capture pedestrian dynamics accurately, which we demonstrate by showing low values of average and final displacement errors.

7.2 Problem Formulation

To model the dynamics of the ego agent, we continue to follow the optimization-based model in (7.1). However, while this model only considers inter-agent safety constraints, we would also like to include safety constraints between the ego-agent and obstacles/walls in the environment. The new model is shown in (7.2)

$$\begin{aligned} \dot{\boldsymbol{x}} = \boldsymbol{u}^* = \quad & \arg\min_{\boldsymbol{u}} \|\boldsymbol{u} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2 \\ \text{subject to} \quad & \underbrace{\begin{bmatrix} A^A \\ A^O \end{bmatrix}}_{A} \boldsymbol{u} \le \underbrace{\begin{bmatrix} \boldsymbol{b}^A_{\gamma,D_s} \\ \boldsymbol{b}^O_{\gamma,D_s} \end{bmatrix}}_{\boldsymbol{b}_{\gamma,D_s}} \end{aligned} \tag{7.2}$$

Here $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) := C(\boldsymbol{x})\boldsymbol{\theta} + \boldsymbol{d}(\boldsymbol{x})$ like before. Moreover, A^A and $\boldsymbol{b}^A_{\gamma,D_s}$ model inter-agent safety constraints and so are the same as in (7.1). The new terms are A^O and $\boldsymbol{b}^O_{\gamma,D_s}$. For completeness, we recall the definitions of inter-agent safety constraints *i.e.* A^A and $\boldsymbol{b}^A_{\gamma,D_s}$, and present the derivation of agent-obstacle safety constraints *i.e.* A^O and $\boldsymbol{b}^O_{\gamma,D_s}$ next.

Modeling safety with other agents: We assume that all agents cooperate to achieve collision avoidance amongst one another while performing their respective tasks. Let the other agents be located at positions $\{\boldsymbol{x}^o_j\}$ $\forall j \in \{1, 2, \cdots, N_A\}$. The ego agent and agent j are collision-free iff their positions $(\boldsymbol{x}, \boldsymbol{x}^o_j)$ satisfy $\|\Delta \boldsymbol{x}_j\|^2 \ge D_s^2$ where $\Delta \boldsymbol{x}_j := \boldsymbol{x} - \boldsymbol{x}^o_j$ and D_s is a desired safety margin. In prior work [54], control barrier functions were used to derive linear constraints on agents' velocities $\boldsymbol{u}$ for ensuring inter-agent collision-free behavior. We use these constraints as is

$$A^A \boldsymbol{u} \le \boldsymbol{b}^A_{\gamma,D_s}, \tag{7.3}$$

where superscript A denotes that these constraints model safety with other agents. $A^A \in \mathbb{R}^{N_A \times 2}$, $\boldsymbol{b}^A \in \mathbb{R}^{N_A}$ are defined so that the j^{th} row of A^A and j^{th} entry of $\boldsymbol{b}^A_{\gamma,D_s}$ are

$$\boldsymbol{a}_j^T := -\Delta \boldsymbol{x}_j^T = -(\boldsymbol{x} - \boldsymbol{x}^o_j)^T \tag{7.4}$$

$$b_j := \gamma(\left\|\boldsymbol{x} - \boldsymbol{x}^o_j\right\|^2 - {D_s}^2) \ \forall j \in \{1, 2, \ldots, N_A\} \tag{7.5}$$

Here $\gamma > 0$ is a hyperparameter unique to each agent. It captures the unwillingness of an agent to compromise on task completion to ensure safety.

Modeling safety with walls/obstacles: Let there be N_O obstacles in the environment, which we assume are polytopic. We denote them as $\mathcal{P}_i := \{\boldsymbol{y} \in \mathbb{R}^2 | A_i^P \boldsymbol{y} \leq b_i^P\}$ $\forall i \in \{1, \cdots, N_O\}$. To model the agent's safety with the $\mathcal{P}_i$, we assume that the agent (located at $\boldsymbol{x}$) tries to stay D_s distance away from the point on $\mathcal{P}_i$ closest to $\boldsymbol{x}$. This point can be calculated by the following QP

$$\begin{aligned} \boldsymbol{y}_i^O = \arg\min_{\boldsymbol{y}} \quad & \|\boldsymbol{x} - \boldsymbol{y}\|^2 \\ \text{subject to} \quad & \boldsymbol{y} \in \mathcal{P}_i \iff A_i^P \boldsymbol{y} \leq b_i^P \end{aligned} \tag{7.6}$$

Akin to (7.3), the following constraints on the ego agent's velocity $\boldsymbol{u}$ account for its safety with each obstacle $\mathcal{P}_i$

$$A^O \boldsymbol{u} \leq \boldsymbol{b}^O_{\gamma, D_s}, \tag{7.7}$$

where superscript O denotes that these constraints model safety with obstacles. $A^O \in \mathbb{R}^{N_O \times 2}$, $\boldsymbol{b}^O \in \mathbb{R}^{N_O}$ are defined such that the i^{th} row of A^O and the i^{th} entry of $\boldsymbol{b}_{\gamma, D_s}$ are

$$\boldsymbol{a}_i^T := -(\boldsymbol{x} - \boldsymbol{y}_i^O)^T \tag{7.8}$$

$$b_i := \gamma(\left\|\boldsymbol{x} - \boldsymbol{y}_i^O\right\|^2 - {D_s}^2) \; \forall i \in \{1, 2, \ldots, N_O\}. \tag{7.9}$$

7.2.1 Inference Problem

The inference problem requires the observer to infer $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \gamma, D_s)$ simultaneously using this agent's positions $\boldsymbol{x}(t)$, its velocities $\boldsymbol{u}^*_{\boldsymbol{\theta}}(\boldsymbol{x}(t))$, the positions of all other agents $\{\boldsymbol{x}^o_j(t)\}_{j=1}^M$ and the positions of the walls and obstacles. The observer will use a batch of K signal-response pairs *i.e.* $\left(\underbrace{\left(\boldsymbol{x}(k), \{\boldsymbol{x}^o_j(k)\}_{j=1}^M\right)}_{signal}, \underbrace{\boldsymbol{u}^*_{\boldsymbol{\theta}}(k)}_{response}\right)$ $\forall k \in \{1, 2, \cdots, K\}$ to compute an estimate of $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \gamma, D_s)$. Like before, the focus is on inferring these for one agent, we can always employ parallelization for inferring these for all the agents together.

7.3 MIQP-based Robust Inference Algorithms

The general approach for inferring $(\boldsymbol{\theta}, \gamma, D_s)$ is to pose an empirical risk minimization algorithm that uses a reasonable heuristic as a loss. We propose two algorithms: the algorithm in 7.3.1 considers the prediction error as a heuristic while the algorithm in 7.3.2 considers a variant of the KKT loss proposed in [84] as a heuristic. Both these algorithms rely on the KKT conditions of (7.2). Thus, we state these conditions first before presenting these algorithms. Let $(\boldsymbol{u}^*, \boldsymbol{\lambda}^*)$ be the optimal primal-dual solution to (7.2). The KKT conditions are

1. Stationarity: $\boldsymbol{u}^*_{\boldsymbol{\theta}} = \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \frac{1}{2}A^T(\boldsymbol{x})\boldsymbol{\lambda}^*$
2. Primal Feasibility: $A(\boldsymbol{x})\boldsymbol{u}^* \leq \boldsymbol{b}_{\gamma,D_s}(\boldsymbol{x})$
3. Dual Feasibility: $\boldsymbol{\lambda}^* \geq \boldsymbol{0}$
4. Complementary Slackness: $\boldsymbol{\lambda}^* \odot \Big(A(\boldsymbol{x})\boldsymbol{u}^* - \boldsymbol{b}_{\gamma,D_s}(\boldsymbol{x})\Big) = \boldsymbol{0}$

Complementary slackness can be re-posed with an equivalent formulation by using the big-M approach [87]. This is done by augmenting the lower bounds $\boldsymbol{0} \leq \boldsymbol{b}_{\gamma,D_s}(\boldsymbol{x}) - A(\boldsymbol{x})\boldsymbol{u}^*$ and $\boldsymbol{0} \leq \boldsymbol{\lambda}^*$ with artificial upper bounds as follows:

$$\begin{aligned} \boldsymbol{0} \leq \boldsymbol{b}_{\gamma,D_s}(\boldsymbol{x}) - A(\boldsymbol{x})\boldsymbol{u}^* \quad & \leq M\boldsymbol{z} \\ \boldsymbol{0} \leq \boldsymbol{\lambda}^* \quad & \leq M(\boldsymbol{1} - \boldsymbol{z}) \end{aligned} \tag{7.10}$$

Here $\boldsymbol{z} \in \{0,1\}^{N_A+N_O}$ are Boolean variables and M is a large number chosen as a hyper-parameter. The Boolean variables $\boldsymbol{z}$ are also unknown and will be learned as part of the inference problem in the next section. Given these conditions, we are ready to develop the first inference algorithm.

7.3.1 Predictability Loss MIQP

The observer assumes that each agent uses (7.2) as the underlying model. Akin to this model, the observer poses a copy problem in which he treats $\boldsymbol{\theta}, \gamma, D_s$ as tunable knobs. These can be tuned until the *predicted* velocities computed by solving the copy problem match with the measured velocities. This can be done by solving:

$$\begin{aligned} \hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{D}_s, \{\hat{\boldsymbol{u}}_k\}_{k=1}^K = & \underset{\boldsymbol{\theta},\gamma,D_s,\{\boldsymbol{u}_k\}_{k=1}^K}{\arg\min} \sum_{k=1}^{K} \|\boldsymbol{u}_k - \boldsymbol{u}_k^{meas}\|^2 \\ \text{such that} \qquad & \boldsymbol{u}_k \text{ solves } (7.2) \;\; \forall k \in \{1,\cdots,K\} \end{aligned} \tag{7.11}$$

The cost function in (7.11) is the empirical sum of the deviations of the predicted controls $\boldsymbol{u}_k$ from the measured controls $\boldsymbol{u}_k^{meas}$. This is known as the *predictability loss* [83]. Naturally, it makes sense to minimize this loss only if the observer's predicted controls solve the forward problem (7.2) which is posed as a constraint in (7.11). Since (7.2) is in itself an optimization problem, (7.11) is a bi-level optimization problem, which is known to be computationally difficult to solve. We convert this to a single level problem by replacing the inner problem with its KKT conditions as follows:

$$
\begin{aligned}
&\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{D}_s, \{\hat{\boldsymbol{u}}_k\}_{k=1}^K, \{\hat{\boldsymbol{\lambda}}_k\}_{k=1}^K, \{\hat{\boldsymbol{z}}_k\}_{k=1}^K, \{\hat{\boldsymbol{\delta}}_k\}_{k=1}^K = \\
&\underset{\substack{\boldsymbol{\theta}, \gamma, D_s, \{\boldsymbol{u}_k\}_{k=1}^K, \\ \{\boldsymbol{\lambda}_k\}_{k=1}^K, \{\boldsymbol{z}_k\}_{k=1}^K, \{\boldsymbol{\delta}_k\}_{k=1}^K}}{\arg\min} \sum_{k=1}^K \|\boldsymbol{u}_k - \boldsymbol{u}_k^{meas}\|^2 + \rho \sum_{k=1}^K \|\boldsymbol{\delta}_k\|^2 \\
&\text{subject to} \\
&\quad \boldsymbol{0} \leq \boldsymbol{b}_{\gamma, D_s}(\boldsymbol{x}_k) - A(\boldsymbol{x}_k)\boldsymbol{u}_k \leq M\boldsymbol{z}_k \\
&\quad \boldsymbol{0} \leq \boldsymbol{\lambda}_k \leq M(\boldsymbol{1} - \boldsymbol{z}_k) \\
&\quad \{\boldsymbol{z}_k\}_{k=1}^K \in \{0,1\}^{N_A+N_O} \\
&\quad -\boldsymbol{\delta}_k \leq \boldsymbol{u}_k - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}_k) + \frac{1}{2}A^T(\boldsymbol{x}_k)\boldsymbol{\lambda}_k \leq \boldsymbol{\delta}_k \\
&\quad \boldsymbol{\theta}_L \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_U, \gamma_L \leq \gamma \leq \gamma_U, D_{sL} \leq D_s \leq D_{sU}
\end{aligned}
\tag{7.12}
$$

The cost function in (7.12) is quadratic. The first term in the cost is the aggregated prediction error and the second penalizes the magnitude of the slack variables. These variables account for how much the stationarity condition is violated. In case the observed measurements of the agent's velocity are noisy, this algorithm tries to find the controls that satisfy the KKT conditions as best as they can while simultaneously ensuring that the inferred/predicted velocities are as close to the measured velocities as possible. Thus including slack variables confers robustness to this algorithm. The constraints are linear in $\boldsymbol{\theta}, \gamma, \gamma D_s^2, \{\boldsymbol{u}_k, \boldsymbol{\lambda}, \boldsymbol{z}_k, \boldsymbol{\delta}_k\}_{k=1}^K$. Since $\{\boldsymbol{z}_k\}_{k=1}^K$ are restricted to be Boolean, the overall problem is a mixed-integer QP. We use Gurobi to solve this problem.

7.3.2 Stationarity Loss MIQP

Another heuristic that can be used to solve the inference problem is the stationarity loss. This loss quantifies the residual of the stationarity condition evaluated on observed positions and velocities:

$$
l_k^{Stat.} = \left\| \boldsymbol{u}_k^{meas} - \hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x}_k) + \frac{1}{2}A^T(\boldsymbol{x}_k)\boldsymbol{\lambda}_k \right\|^2 \tag{7.13}
$$

This residual is quadratic in both $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}_k$. Using K observed signal-response pairs, the observer poses an empirical risk minimization problem that queries for $\boldsymbol{\theta}, \gamma, D_s$, Lagrange multipliers $\{\boldsymbol{\lambda}_k\}_{k=1}^K$ and Boolean variables $\{\boldsymbol{z}_k\}_{k=1}^K$ which minimize the total stationarity loss evaluated on the observed measurements:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{D}_s, \{\hat{\boldsymbol{\lambda}}_k\}_{k=1}^K, \{\hat{\boldsymbol{z}}_k\}_{k=1}^K \;=\; & \underset{\substack{\boldsymbol{\theta},\gamma,D_s,\\ \{\boldsymbol{\lambda}_k\}_{k=1}^K,\{\boldsymbol{z}_k\}_{k=1}^K}}{\arg\min} \sum_{k=1}^{K} l_k^{Stat.} \\
\text{subject to} & \\
\mathbf{0} \quad & \leq \boldsymbol{b}_{\gamma,D_s}(\boldsymbol{x}_k) - A(\boldsymbol{x}_k)\boldsymbol{u}^* \leq M\boldsymbol{z}_k \\
\mathbf{0} \quad & \leq \boldsymbol{\lambda}_k \leq M(\mathbf{1}-\boldsymbol{z}_k) \\
\{\boldsymbol{z}_k\}_{k=1}^K \quad & \in \{0,1\}^{N_A+N_O} \\
\boldsymbol{\theta}_L \quad & \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_U, \gamma_L \leq \gamma \leq \gamma_U, D_{sL} \leq D_s \leq D_{sU}
\end{aligned}
\tag{7.14}
$$

The constraints in this problem capture all the KKT conditions in addition to bounds on $\boldsymbol{\theta}, \gamma, D_s$ from our prior knowledge. Since the cost is quadratic and all constraints are linear in $(\boldsymbol{\theta}, \gamma, \gamma D_s^2, \{\boldsymbol{u}_k\}_{k=1}^K, \{\boldsymbol{\lambda}_k\}_{k=1}^K, \{\boldsymbol{z}_k\}_{k=1}^K)$, the overall problem is a mixed-integer QP. We use Gurobi to solve this problem.

7.4 Results: Validation on Synthetic Datasets

Before testing these algorithms on the human trajectory dataset, we evaluate them on a simulated dataset. We generated several trajectories of a single agent using (7.2) by varying its initial position $\boldsymbol{x}(0)$ in each run. The nominal task for this agent is to follow a constant velocity $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}} = \boldsymbol{v}_d^*$. Additionally, the agent has to ensure safety margin of D_s^* with the obstacles with a conservativeness factor of γ^*. The observer's problem is to infer $\boldsymbol{v}_d, \gamma, D_s$ of this agent from the obtained trajectories using the predictability loss MIQP (7.11) and stationarity loss MIQP (7.14).

Testing Robustness: To test the robustness of these algorithms, we consider two scenarios: scenario (1) has no noise in the measured velocities and in scenario (2) we add zero mean Gaussian noise with 2 m/s standard deviation to the velocities ($\epsilon \sim \mathcal{N}(0, 2m/s)$). To assess repeatability, we conduct ten simulations with a randomly chosen initial position of the agent. The hyperparameter ρ for the predictability loss MIQP was chosen systematically by tuning performance on a validation dataset.

Warm Starting: Expecting that the computation time of predictability loss MIQP can be longer than stationarity loss MIQP due to more variables, we also compare the performance for predictability loss MIQP with and without a warm start using the result from stationary loss MIQP to see if there is any added benefit.

Error Metrics: The proposed algorithms are compared based on parameter reconstruction accuracy and computation time for parameter identification. In the human dataset, ground truth values of parameters are not available since the underlying model of humans

Table 7.1: Performance comparison on THoR environment with noiseless measurements

Metric\ Algorithm	$\|\hat{\boldsymbol{v}}_d - \boldsymbol{v}_d^*\|$ [m/s]	$\|\hat{\gamma} - \gamma^*\|$ [1/s]	$\|\hat{D}_s - D_s^*\|$ [m]	ADE [m]	FDE [m]	Time [s]
Stat. MIQP	0 ± 0	$49.71 \pm .29$	$.0238 \pm .028$	$.0012 \pm .0014$	$.044 \pm .0534$	$8.17 \pm .49$
Pred. MIQP	0 ± 0	$49.71 \pm .29$	$.0238 \pm .028$	$.0012 \pm .0014$	$.044 \pm .0539$	$1.63 \pm .067$
Pred. (warm start)	0 ± 0	$49.71 \pm .29$	$.0238 \pm .028$	$.0012 \pm .0014$	$.044 \pm .0539$	$1.61 \pm .056$

Table 7.2: Performance comparison on THoR environment with noisy measurements

Metric\ Algorithm	$\|\hat{\boldsymbol{v}}_d - \boldsymbol{v}_d^*\|$ [m/s]	$\|\hat{\gamma} - \gamma^*\|$ [1/s]	$\|\hat{D}_s - D_s^*\|$ [m]	ADE [m]	FDE [m]	Time [s]
Stat. MIQP	$.57 \pm 1.56$	$49.64 \pm .2$	$.0295 \pm .016$	$.048 \pm .105$	$.806 \pm 1.432$	$8.01 \pm .083$
Pred. MIQP	2.9 ± 2.73	$49.64 \pm .2$	$.0295 \pm .016$	$.373 \pm .372$	12.9 ± 12	$1.6 \pm .082$
Pred. (warm start)	2.9 ± 2.7	$49.64 \pm .2$	$.0295 \pm .016$	$.332 \pm .346$	9.67 ± 1.28	$1.59 \pm .05$

is not (7.2) necessarily. Therefore, we consider the reconstructed trajectories using inferred parameters for performance evaluation. To this end, comparisons are made based on the average displacement error (ADE) and final displacement error (FDE) of the trajectories generated using the inferred parameters relative to the ground truth trajectories.

Environment: We consider the environment modeled after the map in the THöR dataset. For this, we manually convert the walls and obstacles into polytopes. There are a total of 14 obstacles in this enivornment $i.e. N_O = 14$. This is shown in 7.2.

Analysis of Results: Table 7.1 shows the parameter reconstruction errors and ADE, FDE errors averaged over ten runs with noiseless demonstrations for the THöR environment. It is evident that when perfect measurements are available to the observer, all the three methods share the same performance in terms of inferred parameters accuracy and ADEs and FDEs. However, we notice that stationary loss MIQP takes much longer for computation. Warm start does reduce the computation time for predictability loss MIQP, but only marginally.
Table 7.2 shows these errors with noisy demonstrations. From these tables, it is evident that the stationarity loss MIQP exhibits a great amount of robustness to measurement noise. The ADEs and FDEs are much smaller for the stationarity loss MIQP compared to the other algorithms. The computation time for this algorithm is still very large compared to the other two. Additionally, we observe that warm start improves the performance of predictability loss MIQP with lower error in inferred parameters accuracy and as well as ADE and FDE. There is not much reduction in computation time. Overall, stationarity loss MIQP has the best performance and strongest tolerance to noise in the velocity measurements. In the next

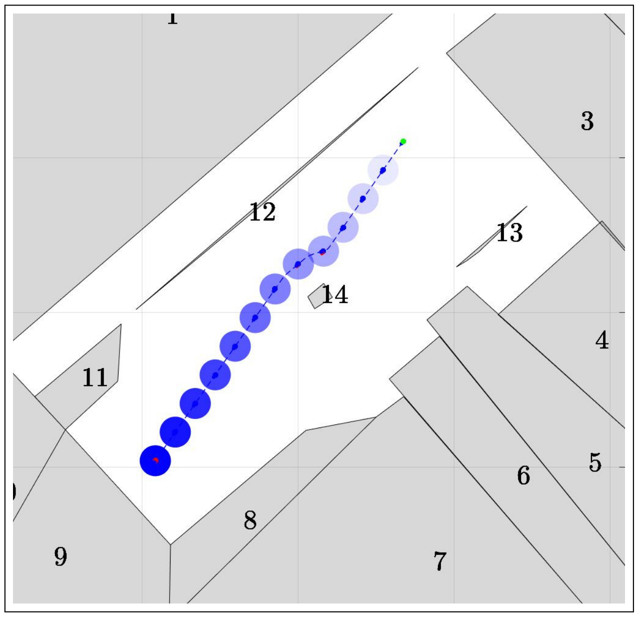


Figure 7.2: Environment used for generating the simulated trajectories using (7.2).

section, we show that the stationarity loss MIQP performs best on the human dataset as well.

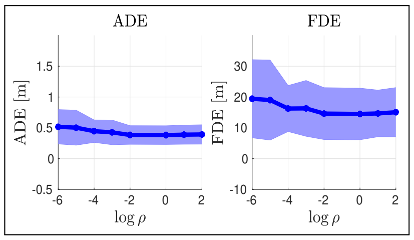
7.5 Results: Validation on Human Datasets

In this section, we validate the proposed methods on a human dataset THöR [30]. The trajectories in this dataset exhibit social interactions that occur in populated spaces like offices, malls etc., thus making them suitable for evaluating our algorithms. Our results show that the learned parameters capture pedestrian dynamics accurately, which we demonstrate by showing low ADE and FDE values. We assume that the nominal task of the humans is to move at a constant velocity. Because of this assumption, we handpicked some scenes in the dataset where the humans approximately moved at a constant velocity. Thus, instead of using the entire datase for parameter identification, we selected 15 handpicked scenarios with selected time intervals of human trajectories where the constant velocity assumption was valid for most of the human pedestrians.

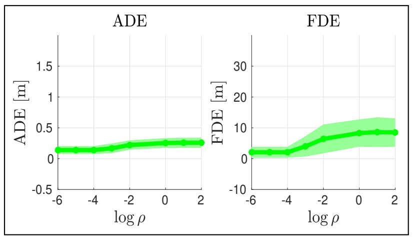
Parameter inference is conducted on each single pedestrian independently. The predictability loss MIQP (7.12) has a hyperparameter ρ which corresponds to the penalty on the slack variables. We chose this hyperparameter by doing a brute force search on all these scenarios. Since the humans do not necessarily follow (7.2) as the underlying model, the only way to check whether our inferred parameters rationalize the observed measurements is by rolling out (7.2) with the learned parameters and comparing the reconstructed trajectories with the ground truth trajectories. We use ADE and FDE as the metrics to do this comparison.

In Figure 7.3, we plot the mean and standard deviation of the ADE and FDE errors averaged over all 15 scenarios for the predictability loss MIQP without warm start (7.3(a)), predictability loss MIQP with warm start (7.3(b)) and the stationarity loss MIQP (7.3(c)). The y axis denotes these errors and the x axis corresponds the value of the slack penalty hyperparameter ρ (on $\log_{10}$ scale).

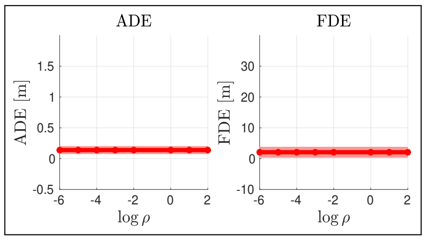
It is observed that predictability loss MIQP with warm start performs better then predictability loss MIQP without warm start, exhibiting both smaller mean and standard deviation in both ADE and FDE. Predictability loss MIQP with warm start performs exactly the same as the stationarity loss MIQP for $\rho < 10^{-4}$. However, the stationarity loss MIQP performs the best, exhibiting the smallest mean ADE and smallest mean FDE with small standard deviation as well.



(a) Pred. Loss MIQP (without warm start)



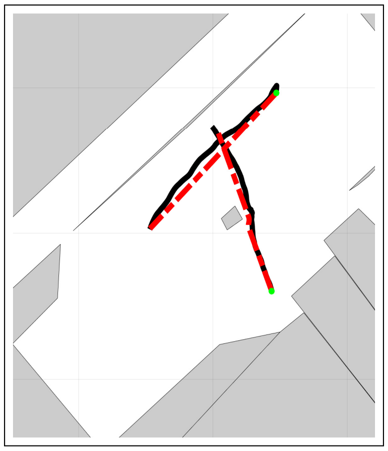
(b) Pred. Loss MIQP (with warm start)

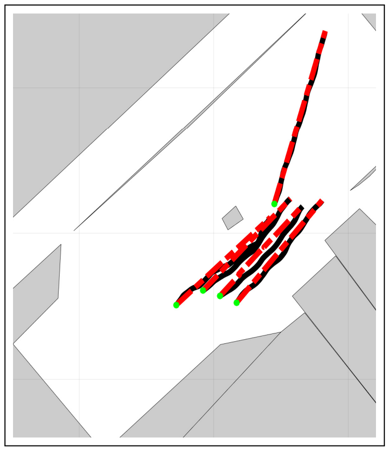


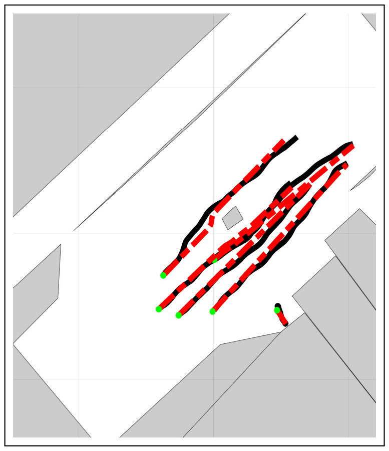
(c) Stat. Loss MIQP

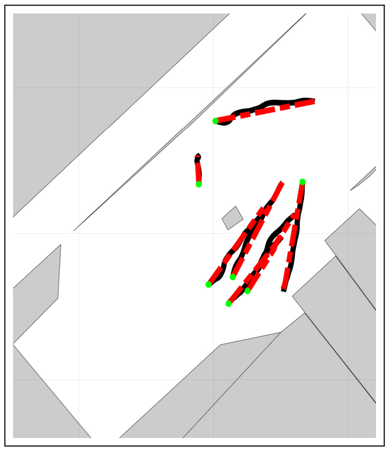
Figure 7.3: ADE and FDE Errors averaged over 15 scenarios sampled from the THöR dataset with varying number of pedestrians in each scenario. X axis represents the log of the slack penalty hyperparameter ρ in (7.12). Among all three algorithms, the stationarity loss MIQP (7.3(c)) exhibits the lowest mean ADE and mean FDE as well as a low standard deviation.

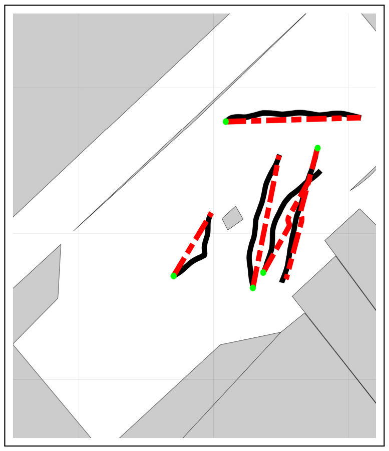
Thus, we select the stationarity loss MIQP to highlight some results on the human dataset. Figure 7.4 shows results from six scenarios sampled from our dataset. The black trajectories are the ground truth trajectories of the humans. The green dots indicate their starting positions. Using these trajectories, we learn the $(\hat{\boldsymbol{v}}_d, \hat{D}_s, \hat{\gamma})$ with (7.14) and then roll out (7.2) with the learned parameters. The obtained trajectories are plotted in red in 7.4. Since most of the red trajectories almost overlap with the given demonstrations, we conclude that the learned parameters do indeed rationalize the ground truth trajectories. In 7.4(e), there is one pedestrian whose motion changes abruptly exhibiting a turn and so technically this pedestrian does not satisfy the constant velocity assumption. Yet our algorithm learns the average behavior giving low FDE error for this pedestrian albeit high ADE. We can fix this problem by including some basis functions in $C(\boldsymbol{x}), \boldsymbol{d}(\boldsymbol{x})$ in the task-based control $\hat{\boldsymbol{u}}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and relaxing the constant velocity model. Making the nominal task more complex with richer basis functions will allow us to get additional accuracy in trajectory reconstruction.


(a) Scenario 1


(b) Scenario 2


(c) Scenario 3


(d) Scenario 4


(e) Scenario 5

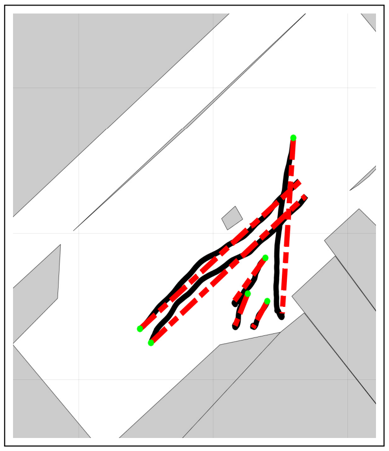

(f) Scenario 6

Figure 7.4: Validation on human dataset: Comparison between the ground truth trajectories (solid black lines) and reconstructed trajectories with learned parameters (dotted red lines) on six selected scenarios. Starting points are marked in green.

7.6 Discussion and Conclusions

We considered the problem of simultaneously inferring task and safety constraint parameters of individual agents of a multiagent system. We modeled the agents using control barrier functions and developed the predictability loss MIQP and the stationarity loss MIQP to solve the inference problem. We demonstrated that the reconstructed parameters rationalize the observed measurements in a simulated single-agent scenario and also a real human dataset. There are several directions that we would like to take in future work.

- **Richer basis functions** $C(\boldsymbol{x}), \boldsymbol{d}(\boldsymbol{x})$: Sticking to the constant velocity model was a first-order choice and this itself gave us accurate reconstructions. However, we hope that including richer functions that capture the spatial frequencies of motion can allow us to get more accuracy. The overall problem complexity will only increase marginally since as we showed the problem is still an MIQP. This will allow us to capture changes in human trajectory better.

- **Time-varying parameters:** In our framework, we assumed that the task and constraint parameters $(\boldsymbol{\theta}, \gamma, D_s)$ are constant. However, humans may become more conservative as they come around agents that exhibit uncertain motions. As a result, these parameters may be time-varying. In our future work, we will adjust our code to allow for time-varying parameters. The overall problem will still remain an MIQP, although it will have to learn many parameters.
- **Multimodal Learning:** Currently, our proposed method learns one set of parameters for a human. However, humans may exhibit a distribution of behaviors. In future work, we will explore how to adjust the inference framework so that we can reverse engineer a distribution of these parameters instead of learning one set of parameter values.

III

Multiagent Control Through Interactions

8 Single Agent's Control Through Interactions

Chapters 4 - 7 investigated the multiagent behavior inference problem. We developed robust algorithms for inferring parameters that encode information about each agent's task as well as its collision avoidance behavior with respect to other agents in the system. By and large, these algorithms address the first part of the thesis statement. The next part focuses on shaping the behavior of individual agents of a multiagent system by inducing interactions with robots under the control of an observer. This is the focus of this chapter and the next two chapters.

There is a wide range of domains where controlling multi-agent teams via external robots finds practical applications. These include the use of robots to herd sheep [56], [57]; crowd control [58]; protecting aircraft from bird strikes [59], [60] and defense against adversarial swarms [19]. We use the term *shepherding behaviors* to collectively represent (a) herding, (b) defending and (c) herding+defending [21] behaviors. In herding, the shepherd(s) steer a flock from one region to another and ensure that the flock stays there, while in defending, the shepherd(s) protect a designated region and keep the flock from entering it. In our context, the flock represents the multiagent system and the shepherd(s) represent the robots that are under the control of the observer. This is a difficult control problem because the shepherd does not have direct control over the actuators of the flock agents, rather must rely on the agents' interaction with the shepherd to regulate their behavior. This problem is also challenging because usually there are not as many shepherding robots as agents in the flock. Therefore, from the perspective of the shepherd, the control problem becomes very underactuated.

Several prior works have considered the problem of noncooperative shepherding using robots. Some of these include [22], [63], [64], [65], [66], [67]. They refer to the shepherding problem as noncooperative because the flock agents are not necessarily adversarial *i.e.* they do not work against the robots, but at the same time are not cooperative because the flock agents repel from the robots. These works exploit this repulsive interaction to develop feedback controllers for the robots to steer the flock agents to a designated region. While successful, one issue common among these works is that they fail to consider the self-motivated dynamics of the flock agents *i.e.* their nominal dynamics without any robots in the picture. As a result, the flock agents' motions are solely driven by repulsions from the robots. Secondly, it is assumed that the flock agents incorporate these repulsions perpetually regardless of how far the robots are located relative to the flock. These two assumptions enable the robots to capitalize on this *continual fear* and gives them more command over the flock than they would have if the flock has a mind of its own or if it ignores robots farther away. Our goal in this chapter is to relax both of these assumptions. Firstly, we want to make the robots move so that they make their presence acknowledged by the flock, and once this is done, we want to plan motions for the robots that elicit desired behaviors from the flock while being cognizant of the flock's nominal/natural dynamics. In this chapter and the next, we assume that the observer and their robots know the true parameters of the dynamic model of the group agents *i.e.* $\boldsymbol{\Theta}$ (3.16). We are solving the *Multiagent Behavior Shaping With Known Model* as defined in Def. 2. That is why these parameters will not be highlighted anywhere in this chapter. This assumption will be relaxed in chapter 10.

The outline of this chapter is as follows: in section 8.1, we describe our choice of the flock's dynamics and recall the multiagent behavior shaping problem from Def. 2 in chapter 3. In section 8.2, we describe our approach for shaping the behavior of a single agent using one robot. Section 8.3 gives numerical results for the herding and defending behaviors. Finally, we summarize in section 8.5 and give directions for extending our approach to the multirobot/multiagent behavior shaping case.

8.1 Problem Formulation

Recall the problem described in section 3.3: suppose there M agents in the group collectively denoted as $\mathcal{A} := \{1, 2, \cdots, M\}$. Let the position of agent i be $\boldsymbol{x}_{A_i} \in \mathbb{R}^2$. Likewise, assume there are N robots collectively denoted as $\mathcal{R} := \{1, 2, \cdots, N\}$. Let robot k be located at $\boldsymbol{x}_{R_k} \in \mathbb{R}^2$. Consistent with our modeling paradigm so far, we choose to model the dynamics of the agents using a reactive one-step optimization

$$\begin{aligned}
\dot{\boldsymbol{x}}_{A_i} = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{x}_{A_i})\|^2 \\
\text{subject to} \quad & A\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{A_j}\}_{j\in\mathcal{A}\setminus i}\Big)\boldsymbol{u} \le \boldsymbol{b}_{\gamma,D_s}\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{A_j}\}_{j\in\mathcal{A}\setminus i}\Big) \text{ constraints with agents} \\
& A\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big)\boldsymbol{u} \le \boldsymbol{b}_{\gamma,D_s}\Big(\boldsymbol{x}_{A_i}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big) \text{ constraints with robots} \\
:= \quad & \boldsymbol{f}_i\Big(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big)
\end{aligned} \tag{8.1}$$

The cost function penalizes deviation from a reference task-based control. In addition to the collision avoidance constraints with other agents in the flock, (8.1) considers constraints with the robots as well. These robots are controlled by the observer. The reference task-based control $\hat{\boldsymbol{u}}(\boldsymbol{x}_{A_i})$ is the control that the flock agent would follow should all its constraints be inactive. Thus, it represents the autonomous self-motivated dynamics of the flock agent. Additionally, if the constraint with robot k is inactive, this would imply that the flock agent has chosen to *ignore* robot k. This means that robot k cannot control the agent, it can only do so when its corresponding collision avoidance constraint is active. Thus robot k will have to take action to activate that constraint. In the next section, we will describe a formal strategy for robot k to achieve this. As before, each robot is assumed to be velocity-controlled with dynamics

$$\dot{\boldsymbol{x}}_{R_k} = \boldsymbol{u}_{R_k} \quad \forall k \in \mathcal{R} \tag{8.2}$$

We view (8.1) and (8.2) as a joint system, with its state defined by the joint states of all agents and robots and the velocities of the robots as the control input, *i.e.*,

$$\begin{pmatrix} \dot{\boldsymbol{x}}_{A_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{A_M} \\ \dot{\boldsymbol{x}}_{R_1} \\ \vdots \\ \dot{\boldsymbol{x}}_{R_N} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1\Big(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big) \\ \vdots \\ \boldsymbol{f}_M\Big(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big) \\ \boldsymbol{u}_{R_1} \\ \vdots \\ \boldsymbol{u}_{R_N} \end{pmatrix} \tag{8.3}$$

Given this joint model, we now pose the problem of eliciting a desired behavior from the agents. We denote the observer's behavioral requirement expected from agent i by a function $y_i : \mathbb{R}^2 \longrightarrow \mathbb{R}$. Define the set $\mathcal{Y} \subset \mathbb{R}^{2M}$ as

$$\mathcal{Y} \coloneqq \{(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}) \in \mathbb{R}^{2M} | y_i(\boldsymbol{x}_{A_i}) > 0 \ \forall i \in \mathcal{A}\} \tag{8.4}$$

Given this set, the multiagent behavior shaping problem is

Problem 1. *Provided the initial agent positions* $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \in \mathcal{Y}$*, find robot controls* $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ *such that* $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \in \mathcal{Y} \ \forall t > 0$*. However, if the positions of the agents* $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \notin \mathcal{Y}$ *, find* $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ *such that* $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \rightsquigarrow \mathcal{Y}$ *in finite time.*

We can express the herding and defending behaviors using the above terminology

1. **Herding to a home zone**: Suppose the desired region that the observer wants the agents to converge to is given by $\mathcal{B} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^2 | \, \|\boldsymbol{x} - \boldsymbol{x}_G\| < R_G\}$. Then, the observer can choose $y(\boldsymbol{x}) = R_G^2 - \|\boldsymbol{x} - \boldsymbol{x}_G\|^2$ so that $\mathcal{Y} \coloneqq \mathcal{B}$.

2. **Defending a protected zone**: Suppose the desired region that the observer wants the agents to stay away from is given by $\mathcal{P} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^2 | \, \|\boldsymbol{x} - \boldsymbol{x}_P\| < R_P\}$. Then, the observer can choose $y(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_P\|^2 - R_P^2$ so that $\mathcal{Y} \coloneqq \mathcal{P}^c$.

In the next section, we give a preliminary approach for the single agent shepherding problem.

8.2 Behavior shaping for one agent with one robot

We begin by considering the simple problem of shaping the behavior of one agent designated A using one robot designated R by commanding R's velocities $\boldsymbol{u}_R$. Since there are no other agents aside from A, the dynamics of A are

$$\begin{aligned} \dot{\boldsymbol{x}}_A = \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{x}_A)\|^2 \\ \text{subject to} \quad & \boldsymbol{a}^T(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u} \leq b(\boldsymbol{x}_A, \boldsymbol{x}_R) \quad \longleftarrow \mathbf{C}_R \end{aligned} \tag{8.5}$$

Here $\mathbf{C}_R$ represents the collision-avoidance constraint between the agent and the robot. Depending on whether it is active or inactive, these dynamics can be rewritten as

$$\dot{\boldsymbol{x}}_A = \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) = \begin{cases} \hat{\boldsymbol{u}}(\boldsymbol{x}_A) & \textit{if } \mathbf{C}_R \textit{ inactive} \\ \hat{\boldsymbol{u}}(\boldsymbol{x}_A) - \frac{1}{2}\mu \boldsymbol{a}(\boldsymbol{x}_A, \boldsymbol{x}_R) & \textit{if } \mathbf{C}_R \textit{ active.} \end{cases} \tag{8.6}$$

The observer's behavioral requirement is expressed using the function $y(\boldsymbol{x}_A)$ and its zero level superset $\mathcal{Y} := \{\boldsymbol{x}_A \in \mathbb{R}^2 | y(\boldsymbol{x}_A) > 0\}$. The behavior shaping problem for agent A requires the observer to find control inputs $\boldsymbol{u}_R$ such that $\boldsymbol{x}_A(t) \in \mathcal{Y}\ \forall t \geq 0$ if $\boldsymbol{x}_A(0) \in \mathcal{Y}$ or $\boldsymbol{x}_A(t) \rightsquigarrow \mathcal{Y}$ if $\boldsymbol{x}_A(0) \notin \mathcal{Y}$. Both these problems can be addressed using the theory of exponential control barrier functions. Informally, if we can find $\boldsymbol{u}_R$ such that

$$\ddot{y} + \alpha \dot{y} + \beta y \geq 0, \tag{8.7}$$

(where α and β are such that the roots of $s^2 + \alpha s + \beta = 0$ have strictly negative real parts), then we will have a guarantee that the behavior shaping requirement is met. For this, we treat $y(\boldsymbol{x}_A)$ as a control-barrier function to find the desired $\boldsymbol{u}_R$. Its derivatives are:

$$\begin{aligned} \dot{y}(\boldsymbol{x}_A, \boldsymbol{x}_R) &= \underbrace{\nabla_{\boldsymbol{x}_A} y^T(\boldsymbol{x}_A) \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R)}_{p(\boldsymbol{x}_A, \boldsymbol{x}_R)} \\ \ddot{y}(\boldsymbol{x}_A, \boldsymbol{x}_R) &= \nabla_{\boldsymbol{x}_A} \dot{y}^T \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) + \nabla_{\boldsymbol{x}_R} \dot{y}^T \boldsymbol{u}_R \\ &= \underbrace{\nabla_{\boldsymbol{x}_A} \Big(\nabla_{\boldsymbol{x}_A} y^T(\boldsymbol{x}_A) \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) \Big)^T \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R)}_{g(\boldsymbol{x}_A, \boldsymbol{x}_R)} + \underbrace{\nabla_{\boldsymbol{x}_R} \Big(\nabla_{\boldsymbol{x}_A} y^T(\boldsymbol{x}_A) \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) \Big)^T}_{G(\boldsymbol{x}_A, \boldsymbol{x}_R)} \boldsymbol{u}_R \\ &= g(\boldsymbol{x}_A, \boldsymbol{x}_R) + G(\boldsymbol{x}_A, \boldsymbol{x}_R) \boldsymbol{u}_R. \end{aligned} \tag{8.8}$$

Thus, substituting (8.8) in (8.7) gives

$$\begin{aligned} & \ddot{y} + \alpha \dot{y} + \beta y \geq 0 \\ \iff & g(\boldsymbol{x}_A, \boldsymbol{x}_R) + G(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R + \alpha p(\boldsymbol{x}_A, \boldsymbol{x}_R) + \beta y(\boldsymbol{x}_A) \geq 0 \\ \iff & \underbrace{-G(\boldsymbol{x}_A, \boldsymbol{x}_R)}_{A^1(\boldsymbol{x}_A, \boldsymbol{x}_R)} \boldsymbol{u}_R \leq \underbrace{g(\boldsymbol{x}_A, \boldsymbol{x}_R) + \alpha p(\boldsymbol{x}_A, \boldsymbol{x}_R) + \beta y(\boldsymbol{x}_A)}_{b^1_{\alpha,\beta}(\boldsymbol{x}_A, \boldsymbol{x}_R)} \\ \iff & A^1(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^1_{\alpha,\beta}(\boldsymbol{x}_A, \boldsymbol{x}_R). \end{aligned} \tag{8.9}$$

This gives us a linear constraint on $\boldsymbol{u}_R$. This constraint can become infeasible if $G(\boldsymbol{x}_A, \boldsymbol{x}_R) \equiv \mathbf{0}$. This can happen when

$$G \equiv \mathbf{0} \impliedby \nabla_{\boldsymbol{x}_R}\Big(\nabla_{\boldsymbol{x}_A} y^T(\boldsymbol{x}_A)\boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R)\Big)^T \equiv \mathbf{0} \impliedby \nabla_{\boldsymbol{x}_R}\boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) \equiv \mathbf{0} \impliedby \mathbf{C}_R \text{ inactive} \tag{8.10}$$

Thus, R needs to move such that $\mathbf{C}_R$ remains active if it is active at $t = 0$ or becomes active in a finite-time if it is inactive at $t = 0$. Intuitively, $\mathbf{C}_R$ being inactive implies that A does not have any interaction with R thereby preventing R from exerting any influence on A. To ensure activeness, we require that A's velocity in the absence of R violate A's safety constraint with R, when R is put back in. That is to say in R's absence, if the only choice for A's velocity *i.e.* $\hat{\mathbf{u}}(\boldsymbol{x}_A)$ becomes infeasible when R is put back, then $\mathbf{C}_R$ will become active, or in other words,

$$\boldsymbol{a}^T(\boldsymbol{x}_A, \boldsymbol{x}_R)\hat{\mathbf{u}}(\boldsymbol{x}_A) > b(\boldsymbol{x}_A, \boldsymbol{x}_R). \tag{8.11}$$

Treat $y(\boldsymbol{x}_A, \boldsymbol{x}_R) = \boldsymbol{a}^T(\boldsymbol{x}_A, \boldsymbol{x}_R)\hat{\mathbf{u}}(\boldsymbol{x}_A) - b(\boldsymbol{x}_A, \boldsymbol{x}_R)$ as yet another barrier function. To ensure activeness, we require $y(\boldsymbol{x}_A, \boldsymbol{x}_R) \geq 0$. Thus, treating this y as a barrier function, we can get a yet another linear constraint on $\boldsymbol{u}_R$ [88]

$$\begin{aligned}
&\dot{y} + \gamma \mathtt{sgn}(y(\boldsymbol{x}_A, \boldsymbol{x}_R))\sqrt{|y(\boldsymbol{x}_A, \boldsymbol{x}_R)|} \geq 0 \\
\iff &\nabla_{\boldsymbol{x}_A} y^T(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_R) + \nabla_{\boldsymbol{x}_R} y^T(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R + \gamma \mathtt{sgn}(y(\boldsymbol{x}_A, \boldsymbol{x}_R))\sqrt{|y(\boldsymbol{x}_A, \boldsymbol{x}_R)|} \geq 0 \\
\iff &A^2(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^2_{\gamma}(\boldsymbol{x}_A, \boldsymbol{x}_R),
\end{aligned} \tag{8.12}$$

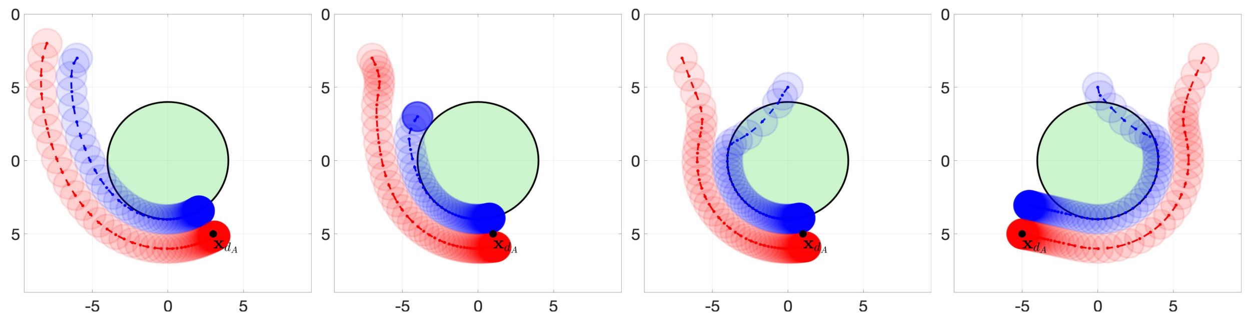
where $\gamma \geq 0$ is another hyperparameter to be chosen by the observer. Thus, $\boldsymbol{u}_R$ must satisfy both (8.12) and (8.9) to be able to shape A's behavior. In addition, the robot should stay collision-free with respect to the agent. This can be expressed using a yet another linear constraint $A^3(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^3_{\gamma_{collision}}(\boldsymbol{x}_A, \boldsymbol{x}_R)$ on the robot's velocity. The robot can use the following optimization to compute its optimal velocity $\boldsymbol{u}^*_R$ that satisfies all three constraints.

$$\begin{aligned}
\boldsymbol{u}^*_R = \arg\min_{\boldsymbol{u}_R} \quad & \|\boldsymbol{u}_R\|^2 \\
\text{subject to} \quad & A^1(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^1_{\alpha,\beta}(\boldsymbol{x}_A, \boldsymbol{x}_R) \\
& A^2(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^2_{\gamma}(\boldsymbol{x}_A, \boldsymbol{x}_R) \\
& A^3(\boldsymbol{x}_A, \boldsymbol{x}_R)\boldsymbol{u}_R \leq b^3_{\gamma_{collision}}(\boldsymbol{x}_A, \boldsymbol{x}_R).
\end{aligned} \tag{8.13}$$

This problem finds the min-norm velocities that satisfy the constraints in (8.12) and (8.9) in addition to the collision avoidance constraints. The cost function encourages robot R to move as little as possible unless either the constraint becomes inactive or breaching occurs or a collision occurs. In the next section, we present numerical results using this approach.

8.3 Results

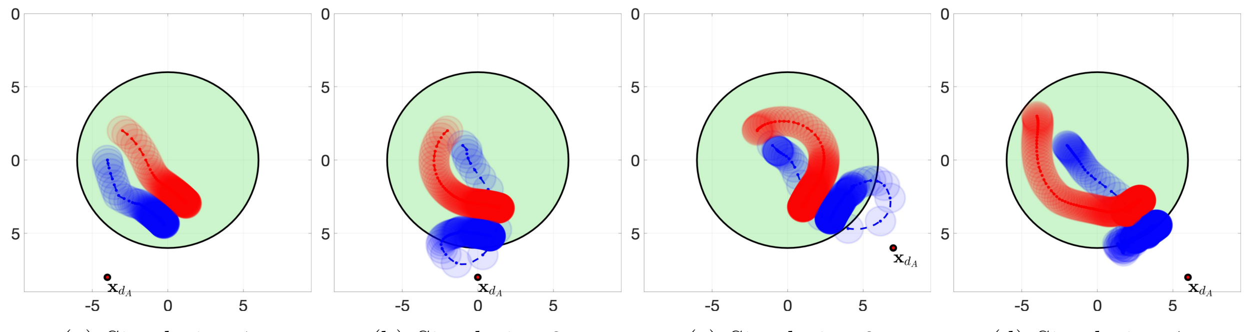
In this section, we present preliminary results that demonstrate the promise of our presented approach. We consider two shepherding behaviors: (a) defending a protected zone and (b) preventing escape from a desired zone. For defending a protected zone, we represent the zone using a circular disc with radius R_P and center at the origin. Thus, we use $y(\boldsymbol{x}_A) \coloneqq \|\boldsymbol{x}_A\|^2 - R_P^2$ as the barrier function in (8.8). The nominal task based control of the agent is $\hat{\boldsymbol{u}} = -k_p(\boldsymbol{x}_A - \boldsymbol{x}_{d_A})$ *i.e.* reaching a goal position $\boldsymbol{x}_{d_A}$. In our simulations, we cherrypick the agent's goal $\boldsymbol{x}_{d_A}$ and its initial position $\boldsymbol{x}_A(0)$ so that the nominal task would result in breaching of the protected zone. The robot's velocities are calculated using (8.13). The parameters α, β, γ are tuned using the guidelines in [55]. Figure 8.1 shows four simulation results for this behavior. In these simulations, we varied the initial position of the robot (blue), the agent (red) and the agent's goal. As can be noticed from the pictures, the robot is able to intercept the agent and prevent it from entering the protected zone.



(a) Simulation 1 (b) Simulation 2 (c) Simulation 3 (d) Simulation 4

Figure 8.1: Defending a protected zone: In these simulations, the robot is shown in blue and the agent is shown in red. The green disc represents the protected zone. The nominal task of the red agent is to go straight towards its goal $\mathbf{x}_A$. However, since this would result in infiltration of the protected zone, the robot intervenes using the optimization problem presented in (8.13).

Next, we consider the task of preventing the agent from escaping a home zone. Thus, we use $y(\boldsymbol{x}_A) \coloneqq R_P^2 - \|\boldsymbol{x}_A\|^2$ as the barrier function in (8.8). The nominal task based control of the agent is $\hat{\boldsymbol{u}} = -k_p(\boldsymbol{x}_A - \boldsymbol{x}_{d_A})$ *i.e.* reaching a goal position $\boldsymbol{x}_{d_A}$. Figure 8.2 shows four simulation results for this behavior. In these simulations, we varied the initial position of the robot (blue), the agent (red) and the agent's goal. As can be noticed from the pictures, the robot is able to intercept the agent and prevent it from escaping by creating a deadlock position. The steady state positions of the robot and the agent end up being collinear with the agent's goal in which deadlock occurs.



(a) Simulation 1 (b) Simulation 2 (c) Simulation 3 (d) Simulation 4

Figure 8.2: Preventing escape from the home zone: In these simulations, the green disc represents the region where the robot wants the agent to stay. The nominal task of the red agent is to go straight towards its goal $\mathbf{x}_A$. In all these simulations, the goal is outside the green region. However, since this would result in the agent escaping, the robot intervenes using the algorithm presented in section 8.2. Notice that the steady state configuration involves the robot deadlocking with the agent.

8.4 Towards multi-agent behavior shaping

In the previous section, we illustrated how one robot (R) can shape the behavior of one agent (A) using ideas from non-collocated feedback linearization and control barrier functions. While the presented formalism is sound for the single agent-single robot case, it is not straightforward to extend this to the case where there are multiple agents and one robot R. To see this, let us consider a scenario with one more agent, call it A'. In the presence of A' and R, the dynamics of A are

$$\begin{aligned} \dot{\boldsymbol{x}}_A = \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_{A'}, \boldsymbol{x}_R) = \arg\min_{\boldsymbol{u}} \quad & \|\boldsymbol{u} - \hat{\boldsymbol{u}}(\boldsymbol{x}_A)\|^2 \\ \text{subject to} \quad & \boldsymbol{a}^T(\Delta\boldsymbol{x}_{AR})\boldsymbol{u} \le b(\Delta\boldsymbol{x}_{AR}) \quad \longleftarrow \mathbf{C}_R \\ & \boldsymbol{a}^T(\Delta\boldsymbol{x}_{AA'})\boldsymbol{u} \le b(\Delta\boldsymbol{x}_{AA'}) \quad \longleftarrow \mathbf{C}_{A'}. \end{aligned} \tag{8.14}$$

The constraint on $\boldsymbol{u}_R$ to ensure $y(\boldsymbol{x}_A) \ge 0$ is similar to (8.9) except that now A^1, b^1 depend on $\boldsymbol{x}_{A'}$ in addition to $\boldsymbol{x}_A$ and $\boldsymbol{x}_R$

$$A^1(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'})\boldsymbol{u}_R \le b^1_{\alpha,\beta}(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'}). \tag{8.15}$$

To ensure feasibility of this constraint, $\mathbf{C}_R$ must be active. For this, like before, R must ensure that A's dynamics in R's absence result in collisions with R when R is put back in. However, this time we have A' as well, so due to A', A's dynamics in the absence of R depend on whether $\mathbf{C}_{A'}$ is active or not. This gives the following two possibilities, thus two requirements on $\boldsymbol{u}_R$:

$$\begin{aligned} \boldsymbol{a}^T(\Delta\boldsymbol{x}_{AR})\hat{\boldsymbol{u}}(\boldsymbol{x}_A) > b(\Delta\boldsymbol{x}_{AR}) &\implies A^2(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'})\boldsymbol{u}_R \le b^2_{\gamma}(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'}) \\ \boldsymbol{a}^T(\Delta\boldsymbol{x}_{AR})\left(\hat{\boldsymbol{u}}(\boldsymbol{x}_A) - \frac{1}{2}\mu_{A'}\boldsymbol{a}(\Delta\boldsymbol{x}_{AA'})\right) > b(\Delta\boldsymbol{x}_{AR}) &\implies A^3(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'})\boldsymbol{u}_R \le b^3_{\gamma}(\boldsymbol{x}_A, \boldsymbol{x}_R, \boldsymbol{x}_{A'}) \end{aligned} \tag{8.16}$$

The first is required when $\mathbf{C}_{A'}$ is inactive while the second is required with $\mathbf{C}_{A'}$ is active. Thus now $\boldsymbol{u}_R$ must satisfy both these constraints in addition to (8.15) to be able to shape A's behavior in the presence of A'. Note here that adding one additional agent A' resulted in two additional constraints on $\boldsymbol{u}_R$ (8.16). In general, adding M agents will result in 2^M additional constraints on $\boldsymbol{u}_R$. As a result of these many constraints, the search for $\boldsymbol{u}_R$ will be very conservative and will likely result in infeasibility. It goes without saying that aside from this potential infeasibility, this active/inactive constraint bookkeeping is also tedious. Thus, while this approach is in theory correct, this is not practically viable. To remedy these issues, we model the dynamics of agent A using the Reynolds-Boids model which captures the essential components of the model in (8.14). The dynamics of an agent A following this model can be posed as

$$\begin{aligned} \dot{\boldsymbol{x}}_A &= \boldsymbol{f}_A(\boldsymbol{x}_A, \boldsymbol{x}_{A'}, \boldsymbol{x}_R) \\ &= \underbrace{\hat{\boldsymbol{u}}(\boldsymbol{x}_A)}_{①} + \underbrace{k_A D_s^3 \frac{(\boldsymbol{x}_A - \boldsymbol{x}_{A'})}{\|\boldsymbol{x}_A - \boldsymbol{x}_{A'}\|^3}}_{②} + \underbrace{k_R \frac{\boldsymbol{x}_A - \boldsymbol{x}_R}{\|\boldsymbol{x}_A - \boldsymbol{x}_R\|^3}}_{③} - \underbrace{k_A(\boldsymbol{x}_A - \boldsymbol{x}_{A'})}_{④} \\ &= \hat{\boldsymbol{u}}(\boldsymbol{x}_A) - k_A\left(1 - \frac{D_s^3}{\|\boldsymbol{x}_A - \boldsymbol{x}_{A'}\|^3}\right)(\boldsymbol{x}_A - \boldsymbol{x}_{A'}) + k_R \frac{(\boldsymbol{x}_A - \boldsymbol{x}_R)}{\|\boldsymbol{x}_A - \boldsymbol{x}_R\|^3} \end{aligned} \tag{8.17}$$

We describe the terms appearing the dynamics individually.

- Term ① specifies the task-related velocity given by $\hat{\boldsymbol{u}}(\boldsymbol{x}_A)$. While the representation of dynamics in (8.14) require that $A's$ velocity $\dot{\boldsymbol{x}}_A$ merely minimize its deviation from $\hat{\boldsymbol{u}}(\boldsymbol{x}_A)$, the dynamics representation in (8.17) enforces that $A's$ velocity $\dot{\boldsymbol{x}}_A$ perpetually contain $\hat{\boldsymbol{u}}(\boldsymbol{x}_A)$.

- Term ② captures the repulsion of A from A'. This means that $A's$ velocity $\dot{\boldsymbol{x}}_A$ must always contain a repulsive term *i.e.* must always treat the collision avoidance constraint $\mathbf{C}_{A'}$ between A and A' as active. This is unlike the dynamics in (8.14) where this constraint may or may not be active. k_A is a proportionality constant and D_s is the safety distance.

- Term ③ captures the repulsion of A from R. Here k_R is the repulsion gain constant. This means that $A's$ velocity $\dot{\boldsymbol{x}}_A$ always acknowledges the presence of R. This is unlike the dynamics in (8.14) where repulsion from R is only incorporated when the collision-avoidance constraint $\mathbf{C}_R$ between A and R as active. Thus, the dynamics in

(8.17) treats the constraint with R as perpetually active. The benefit of this is that now R does not need to spend additional effort to activate the constraint $\mathbf{C}_R$ to be able to influence $A's$ behavior. Allowing for perpetual repulsions allows us to dispense with tedious bookkeeping problem we had to deal with while assuming the dynamics representation in (8.14).

- Term $\textcircled{4}$ captures cohesion of A towards A'. This ensures that both A and A' behave like a flock. This term is not captured in the dynamics in (8.14), but is a common characteristic of Reynolds-Boids models so we include it here.

While (8.17) considers one robot R and two agents A and A', we can generalize this representation to N robots $R_1, R_2, \cdots, R_N$ and M agents $A_1, A_2, \cdots, A_M$. To simplify notation, recall our two sets- $\mathcal{R} \coloneqq \{1, \cdots, N\}$ for robots, and $\mathcal{A} \coloneqq \{1, \cdots, M\}$ for the agents. With this generalization, the dynamics of agent i can be written as

$$\begin{aligned}\dot{\boldsymbol{x}}_{A_i} &= \hat{\boldsymbol{u}}(\boldsymbol{x}_{A_i}) - k_A \sum_{j\in\mathcal{A}\setminus i} \left(1 - \frac{D_s^3}{\left\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\right\|^3}\right)(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}) + k_R \sum_{k\in\mathcal{R}} \frac{(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k})}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\|^3} \\ &\coloneqq \boldsymbol{f}_i\Big(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big) \quad \forall i \in \mathcal{A}. \end{aligned} \tag{8.18}$$

In the next chapter, we develop scalable algorithms based on optimization, for solving the multiagent behavior shaping problem written in Def. 1 given the agent dynamics in (8.18).

8.5 Conclusions

In this chapter, we formulated the multiagent behavior shaping problem and derived a preliminary method to solve this problem. We showed that if the underlying dynamics of the agents are optimization-based, then keeping track of active and inactive constraints becomes tedious as the number of agents increases. Hence, we could only solve this problem for the one robot one agent case. Through simulations, we demonstrated our algorithm's success in finding the velocities for the robot to prevent breach of a zone as well as to prevent escape from the zone. To scale our proposed behavior shaping approach to multiple agents, we simplified the underlying representation of the dynamics. In the next chapter, we build on this representation of dynamics to solve the general problem.

9 Multi-agent Control Through Interactions

In this chapter, we present algorithms to solve the multiagent behavior shaping problem. To keep the discussion simple, we will focus on the defending behavior which requires preventing the breach of a high-value protected zone by a flock of agents using defending robots. The algorithms we propose in this chapter exploit the interaction dynamics between the flock agents and the robots to find robots' velocities that result in all agents getting repelled from the protected zone. These algorithms reactively solve convex optimization problems online that incorporate defending constraints to compute the desired velocities for all robots. Numerical and experimental results are presented to illustrate our algorithms.

The outline of this chapter is as follows. Section 9.1 recalls the dynamics of the agents, the robots and the definition of the behavior shaping problem pertaining to the defending protected zone behavior. Section 9.2 develops a centralized optimization-based controller that calculates the velocities of all robots simultaneously. Section 9.3 develops distributed implementations of the centralized scheme which come with optimality and feasibility guarantees. Section 9.4 provides numerical results showing the success of our approach for multiple robots v/s multiple flock agents. To test the repeatability of our algorithm, we conduct Monte Carlo simulations with increasing the number of robots and agents and demonstrate high-success rates. This provides empirical evidence of the scalability of our approach. Section 9.5 illustrates experimental results for both the distributed and centralized implementations. We wrap up this chapter in section 9.6 with conclusions and directions for future work.

9.1 Problem Formulation

Recall that we have M flock agents denoted by $\mathcal{A} \coloneqq \{1, 2, \cdots, M\}$ located at positions $\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}$ respectively. Likewise, we have N robots denoted by $\mathcal{R} \coloneqq \{1, 2, \cdots, N\}$ located at positions $\boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}$ respectively. We assume that the agents are exhibiting flocking dynamics *i.e.* moving towards a common goal while staying close enough to each other and getting repelled by the robots. These dynamics are

$$\begin{aligned}\dot{\boldsymbol{x}}_{A_i} &= -k_G(\boldsymbol{x}_{A_i} - \boldsymbol{x}_G) - k_A \sum_{j\in\mathcal{A}\setminus i}\left(1 - \frac{D_s^3}{\left\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\right\|^3}\right)(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}) + k_R \sum_{k\in\mathcal{R}} \frac{(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k})}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\|^3} \\ &\coloneqq \boldsymbol{f}_i\Big(\{\boldsymbol{x}_{A_i}\}_{i\in\mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k\in\mathcal{R}}\Big)\ \forall i \in \mathcal{A} \end{aligned} \tag{9.1}$$

Each robot is velocity-controlled with dynamics

$$\dot{\boldsymbol{x}}_{R_k} = \boldsymbol{u}_{R_k}\ \ \forall k \in \mathcal{R} \tag{9.2}$$

There is a protected zone given by

$$\mathcal{P} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^2 |\ \|\boldsymbol{x} - \boldsymbol{x}_P\| \le R_P\} \tag{9.3}$$

Given the dynamics of the flock agents as in (9.1) and the protected zone in (9.3), the agents may end up breaching it while en route to their goal. Therefore, the objective of the observer is to use the robots is to steer the flock agents away from this zone. The observer's behavioral requirements are defined by functions $y_i : \mathbb{R}^2 \longrightarrow \mathbb{R}$ given by

$$y_i(\boldsymbol{x}_{A_i}) = \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\|^2 - R_P^2. \tag{9.4}$$

Requiring $y_i(\boldsymbol{x}_{A_i}) > 0$ enforces that flock agent i be located outside $\mathcal{P}$. Since we would like all flock agents to be outside $\mathcal{P}$, we require $y_i(\boldsymbol{x}_{A_i}) > 0\ \forall i \in \mathcal{A}$. We define the zero-level superset of all these functions to characterize the allowable portion of the state-space where the states of the flock agents can belong. This is denoted as $\mathcal{Y} \subset \mathbb{R}^{2M}$ and defined as

$$\mathcal{Y} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^{2M} | y_i(\boldsymbol{x}_{A_i}) > 0\ \ \forall i \in \mathcal{A}\} \tag{9.5}$$

Given this set, the observer's problem can be posed formally as follows,

Problem 1. *Assuming $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \in \mathcal{Y}$, find robot controls $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \in \mathcal{Y}\ \forall t > 0$. If $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \notin \mathcal{Y}$, find $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \rightsquigarrow \mathcal{Y}$ in finite time.*

Our objective in this chapter is to come up with algorithms to solve this problem. Before presenting the algorithms, we state the assumptions on the observer's knowledge

Assumption 1. *The observer a-priori knows the agents' dynamics in (9.1) i.e. both the structure of the dynamics equations and the parameters $(\boldsymbol{x}_G, D_s, k_G, k_A, k_R)$.*

Assumption 2. *The observer can track the positions and velocities of all agents and robots.*

Assumption 1 is not stringent because if the dynamics are unknown, the robots can learn the dynamics online using multiagent system identification algorithms, some of which we have developed in our prior work [27, 28] and use certainty equivalence to design the controllers. We will demonstrate this in the next chapter. Given these assumptions, we are ready to develop the first algorithm to solve problem 1.

9.2 Centralized Controller

In this section, we present the first algorithm to solve problem 1. Given the protected zone as defined in (9.3), we first pose the requirement for defending against one agent, say agent i located at $\boldsymbol{x}_{A_i}$. Subsequently, we will generalize this to the rest of the agents in the herd. For this agent, treat $y_i(\cdot)$ (9.4) as a safety index. By our choice, $y_i \geq 0 \implies \boldsymbol{x}_{A_i} \in \mathcal{P}^c$ *i.e.* y_i is non-negative whenever agent i is outside $\mathcal{P}$. Thus, assuming that at $t = 0$, $y_i(\boldsymbol{x}_{A_i}(0)) \geq 0$, we require $y_i(\boldsymbol{x}_{A_i}(t)) \geq 0 \ \forall t \geq 0$. Treating $y_i(\cdot)$ as a control barrier function [55], this can be achieved if the derivative of $y_i(\cdot)$ satisfies the following constraint,

$$\begin{aligned}
&\dot{y}_i(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}) + p_1 y_i(\boldsymbol{x}_{A_i}) \geq 0 \\
\implies &2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \dot{\boldsymbol{x}}_{A_i} + p_1 y_i(\boldsymbol{x}_{A_i}) \geq 0 \\
\implies &2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i + p_1 y_i(\boldsymbol{x}_{A_i}) \geq 0. \qquad (9.6)
\end{aligned}$$

Defining $\boldsymbol{x} = (\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N})$, we rewrite this as

$$2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i(\boldsymbol{x}) + p_1 y_i(\boldsymbol{x}_{A_i}) \geq 0. \qquad (9.7)$$

Here p_1 is a design parameter that we choose to ensure that

$$p_1 > 0 \quad \text{and} \quad p_1 > -\frac{\dot{y}_i(\boldsymbol{x}(0))}{y_i(\boldsymbol{x}(0))}. \qquad (9.8)$$

The first condition on p_1 requires that the pole is real and negative. The second depends on the initial positions $\boldsymbol{x}(0)$ of the flock agents and the robots. Now while (9.7) depends on the positions of the agent and robots, it is the velocities of the robots *i.e.* $\boldsymbol{u}_R^{all} = (\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ that are directly controllable not their positions. Since these velocities do not show up in (9.7), we define another function $v_i(\cdot) : \mathbb{R}^{2(M+N)} \longrightarrow \mathbb{R}$

$$v_i(\boldsymbol{x}) = \dot{y}_i(\boldsymbol{x}) + p_1 y_i(\boldsymbol{x}_{A_i}). \qquad (9.9)$$

Like before, in order to ensure $v_i \geq 0$ is maintained, its derivative needs to satisfy

$$\dot{v}_i(\boldsymbol{x}) + p_2 v_i(\boldsymbol{x}) \geq 0. \qquad (9.10)$$

Here p_2 is another design parameter which we choose p_2 to ensure that the following is satisfied

$$p_2 > 0 \quad \text{and} \quad p_2 > -\frac{\ddot{y}_i(\boldsymbol{x}(0)) + p_1 \dot{y}_i(\boldsymbol{x}(0))}{\dot{y}_i(\boldsymbol{x}(0)) + p_1 y_i(\boldsymbol{x}_{A_i}(0))} \qquad (9.11)$$

Using (9.9) in (9.10), we get,

$$\begin{aligned}
&\ddot{y}_i(\boldsymbol{x}) + \underbrace{(p_1 + p_2)}_{\alpha} \dot{y}_i(\boldsymbol{x}) + \underbrace{p_1 p_2}_{\beta} y_i(\boldsymbol{x}_{A_i}) \geq 0 \\
\implies &\ddot{y}_i(\boldsymbol{x}) + \alpha \dot{y}_i(\boldsymbol{x}) + \beta y_i(\boldsymbol{x}_{A_i}) \geq 0. \qquad (9.12)
\end{aligned}$$

Here α and β are hyperparameters and are chosen by ensuring that p_1 and p_2 satisfy the requirements in (9.8) and (9.11). The derivatives of $y_i(\cdot)$ required in (9.12) are

$$\begin{aligned}
\dot{y}_i(\boldsymbol{x}) &= 2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \dot{\boldsymbol{x}}_{A_i} \\
&= 2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}) \\
\ddot{y}_i(\boldsymbol{x}) &= 2\dot{\boldsymbol{x}}_{A_i}^T \dot{\boldsymbol{x}}_{A_i} + 2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \left(\sum_{j=1}^{M} \mathbb{J}_{ji}^{A} \dot{\boldsymbol{x}}_{A_j} + \sum_{k=1}^{N} \mathbb{J}_{ki}^{R} \boldsymbol{u}_{R_k} \right) \\
&= 2\boldsymbol{f}_i^T \boldsymbol{f}_i + 2(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \left(\sum_{j=1}^{M} \mathbb{J}_{ji}^{A} \boldsymbol{f}_j + \sum_{k=1}^{N} \mathbb{J}_{ki}^{R} \boldsymbol{u}_{R_k} \right),
\end{aligned} \tag{9.13}$$

where $\mathbb{J}_{ji}^{A}$ and $\mathbb{J}_{ki}^{R}$ are

$$\begin{aligned}
\mathbb{J}_{ji}^{A} &:= \nabla_{\boldsymbol{x}_{A_j}} \boldsymbol{f}_i(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}) \\
\mathbb{J}_{ki}^{R} &:= \nabla_{\boldsymbol{x}_{R_k}} \boldsymbol{f}_i(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N})
\end{aligned}$$

Note here that $\ddot{y}_i(\boldsymbol{x})$ contains the velocities of robots as we wanted. Substituting (9.13) in (9.12), we get a linear constraint on the velocities of all the robots to ensure that the i^{th} agent stays outside $\mathcal{P}$

$$A_i^H \boldsymbol{u}_R^{all} \leq b_i^H, \tag{9.14}$$

where,

$$\begin{aligned}
A_i^H &:= (\boldsymbol{x}_P - \boldsymbol{x}_{A_i})^T \begin{bmatrix} \mathbb{J}_{1i}^{R} & \mathbb{J}_{2i}^{R} & \ldots\ldots & \mathbb{J}_{Ni}^{R} \end{bmatrix} \\
b_i^H &:= \boldsymbol{f}_i^T \boldsymbol{f}_i + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \sum_{j=1}^{M} \mathbb{J}_{ji}^{A} \boldsymbol{f}_j + \alpha(\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i + \beta \frac{y_i}{2}.
\end{aligned} \tag{9.15}$$

While this constraint ensures that only agent i is repelled from $\mathcal{P}$, the observer requires this for all M flock agents, not just agent i. This can be done by augmenting constraints for all agents in the flock as follows

$$\begin{bmatrix} A_1^H \\ \vdots \\ A_M^H \end{bmatrix} \boldsymbol{u}_R^{all} \leq \begin{bmatrix} b_1^H \\ \vdots \\ b_M^H \end{bmatrix} \implies \boldsymbol{\mathcal{A}}^H \boldsymbol{u}_R^{all} \leq \boldsymbol{b}^H \tag{9.16}$$

Here $\boldsymbol{\mathcal{A}}^H \in \mathbb{R}^{M \times 2N}$ and $\boldsymbol{b}^H \in \mathbb{R}^M$. Thus, there are M constraints, one for each agent in the flock. Given these constraints on the robots' velocities, we pose the following QP that searches for the min-norm velocities that satisfies these constraints

$$\begin{aligned}
\boldsymbol{u}_R^{*all} &= \arg\min_{\boldsymbol{u}_R^{all}} \left\| \boldsymbol{u}_R^{all} \right\|^2 \\
\text{subject to} \quad & \boldsymbol{\mathcal{A}}^H \boldsymbol{u}_R^{all} \leq \boldsymbol{b}^H
\end{aligned} \tag{9.17}$$

The cost function in (9.17) is $\left\|\boldsymbol{u}_R^{all}\right\|^2 = \sum_{i=1}^N \|\boldsymbol{u}_{R_i}\|^2$. Choosing this cost function encourages the optimizer to find velocities that consume the least total effort aggregated over all robots. Thus, the robots will move only if not moving might result in violation of the constraints. Furthermore, since (9.17) computes the velocities of all robots together, it is a centralized approach by construction. Now while in the above derivation, we considered preventing the agents from breaching only one protected zone $\mathcal{P}$, we can just as easily protect an additional zone $\mathcal{P}'$ by formulating similar constraints $\mathcal{A}_2^H \boldsymbol{u}_R^{all} \leq \boldsymbol{b}_2^H$ on the robots' velocities corresponding to $\mathcal{P}'$. By augmenting (9.17) with new constraints for $\mathcal{P}'$, we will be able to defend both zones from all agents simultaneously. This is a benefit offered by our constraint based framework. An experimental validation of this is shown in Fig. 9.5. The following theorem proves that for the one robot v/s one agent case, (9.17) is always feasible:

Theorem 1. *If there is one robot and one agent, then* (9.17) *always has a solution.*

Proof. See appendix 9.7.1. □

The defending constraints we posed in (9.16) do not guarantee that the robots won't collide with the flock agents. Even though the agents' dynamics have repulsions from the robots, the robot velocities computed using (9.17) can result in aggressive behavior. Thus, we augment the defending constraints with additional constraints to ensure collision-free behavior. Following the approach in [54], we define a pairwise safety index $b^{ik}(\cdot) : \mathbb{R}^2 \times \mathbb{R}^{2N} \longrightarrow \mathbb{R}$ as:

$$\begin{aligned} b^{ik}(\boldsymbol{x}_{A_i}, \boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}) &= \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\|^2 - R_A^2 \\ &= \left\|\boldsymbol{x}_{A_i} - C_k \boldsymbol{x}_R^{all}\right\|^2 - R_A^2. \end{aligned}$$

$b^{ik}(\cdot) \geq 0$ iff robot k is atleast R_A distance away from agent i. Here C_k is a matrix defined appropriately to extract the position of the k^{th} robot from $\boldsymbol{x}_R^{all}$. If $b^{ik}(\boldsymbol{x}_{A_i}(0), \boldsymbol{x}_R^{all}(0)) \geq 0$ $\forall k \in \mathcal{R}$, we would like to ensure that $b^{ik}(\boldsymbol{x}_{A_i}(t), \boldsymbol{x}_D^{all}(t)) \geq 0$ $\forall t \geq 0$ and $\forall k \in \mathcal{R}$. This can be achieved by requiring that

$$\dot{b}^{ik}(\boldsymbol{x}) + \gamma b^{ik}(\boldsymbol{x}) \geq 0 \quad \forall k \in \mathcal{R}, \tag{9.18}$$

where $\gamma > 0$. This gives us a total of N linear constraints on the velocity of all robots for avoiding collisions with the i^{th} agent:

$$A_i^C \boldsymbol{u}_R^{all} \leq b_i^C, \tag{9.19}$$

where,

$$A_i^C = \begin{bmatrix} (\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_1})^T C_1 \\ \vdots \\ (\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_N})^T C_N \end{bmatrix} \text{ and } b_i^C = \begin{bmatrix} \frac{\gamma}{2} b^{i1} + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_1})^T \boldsymbol{f}_i \\ \vdots \\ \frac{\gamma}{2} b^{iN} + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_N})^T \boldsymbol{f}_i \end{bmatrix}. \tag{9.20}$$

To ensure all collision avoidance with all M agents[1], we augment constraints (9.19) for all the herd as follows

$$\begin{bmatrix} A_1^C \\ \vdots \\ A_M^C \end{bmatrix} \boldsymbol{u}_R^{all} \leq \begin{bmatrix} b_1^C \\ \vdots \\ b_M^C \end{bmatrix} \implies \mathcal{A}^C \boldsymbol{u}_R^{all} \leq \boldsymbol{b}^C. \tag{9.21}$$

Next, we combine the defending constraints in (9.16) with the collision avoidance constraints in (9.21) and incorporate them in the following QP to find the optimal velocities of the robots

$$\begin{aligned} \boldsymbol{u}_R^{*all} = \underset{\boldsymbol{u}_R^{all}}{\arg\min} & \left\| \boldsymbol{u}_R^{all} \right\|^2 \\ \text{subject to} \quad & \mathcal{A}^H \boldsymbol{u}_R^{all} \leq \boldsymbol{b}^H \\ & \mathcal{A}^C \boldsymbol{u}_R^{all} \leq \boldsymbol{b}^C. \end{aligned} \tag{9.22}$$

Here $\boldsymbol{u}_R^{*all}$ are the optimal velocities for all the robots to ensure both defending and collision avoidance simultaneously. The cost function penalizes the total speed of the robots, thus encouraging them to minimize their movement. This QP is a centralized optimization problem, in that, we are calculating the velocities of all robots together. From our observations, we noticed that centralization results in automatic allocation of the robot for defense against the agent closest to it, so the total movement aggregated over all the robots is minimized. Moreover, we also observed that a small amount of underactuation *i.e.* fewer robots than flock agents was enough for zone defense in most circumstances. However, this was not repeatable; breach occurred whenever (9.17) became infeasible. Thus, while budget-efficient, the centralized algorithm lacks the feasibility guarantee whenever $N < M$. In the next section, we develop an algorithm that can provide a guarantee on zone defense and is also distributed in its implementation. We also develop another distributed algorithm that attains the same budget efficiency as promised by the centralized algorithm.

9.3 Distributed Controllers

In the previous section, we developed a centralized algorithm to solve the behavior shaping problem (9.17). We could only provide a proof of feasibility of the centralized algorithm when the number of agents and robots are both one. In an attempt to generalize that result, in this section, we develop a distributed algorithm that guarantees that problem (1) is solvable when the number of robots is equal to the number of flock agents *i.e.* $M = N$. To achieve this feasibility, we allocate each flock agent to a unique robot and pose a constraint on that robot's velocity to herd its allocated agent away from the protected zone. This approach allows to guarantee the existence of a solution to problem (9.17) while coming at the expense of necessitating as many robots as flock agents.

[1]inter-robot collision avoidance constraints can also be added following a similar procedure.

While this is a strong result in itself, in many experiments and simulations, we observed that often times, fewer robots than agents were sufficient to repel all agents away from $\mathcal{P}$ *i.e.* underactuation did not necessarily result in zone breach. This observation led us to develop a second distributed algorithm. In this algorithm, we develop an iterative approach that asymptotically attains the same velocities as computed by the centralized algorithm, thereby attaining the same total optimality (measured in terms of the total movement the robots exhibit) as the centralized approach and obviating the need to have equal numbers of robots and flock agents. We build on the dual-decomposition algorithms proposed in [89, 90] for developing this distributed algorithm. Both of our proposed distributed algorithms are compositional in nature *i.e.*, we can protect multiple zones by including more constraints, as shown in figure 9.1(c). While the first distributed algorithm favors feasibility, *i.e.* guaranteeing the existence of solutions to (9.17), the other favors optimality *i.e* inherits the optimality (*i.e.* budget efficiency) from the centralized approach. The starting point of both these algorithms is the the centralized algorithm in (9.17) which we recall here.

$$\begin{aligned} \boldsymbol{u}_R^{*all} &= \arg\min_{\boldsymbol{u}_R^{all}} \left\| \boldsymbol{u}_R^{all} \right\|^2 \\ &\text{subject to} \quad A_i^H \boldsymbol{u}_R^{all} \leq \boldsymbol{b}_i^H \ \forall i \in \{1, 2, \cdots, M\} \text{ where,} \end{aligned} \tag{9.23}$$

$$\begin{aligned} A_i^H &:= (\boldsymbol{x}_P - \boldsymbol{x}_{A_i})^T \begin{bmatrix} \mathbb{J}_{1i}^R & \mathbb{J}_{2i}^R & \ldots.. & \mathbb{J}_{Ni}^R \end{bmatrix} \\ b_i^H &:= \boldsymbol{f}_i^T \boldsymbol{f}_i + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \sum_{j=1}^{M} \mathbb{J}_{ji}^A \boldsymbol{f}_j + \alpha (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i + \beta \frac{y_i}{2} \end{aligned}$$

9.3.1 Approach 1: Allocating one agent to one robot

In this approach, we assume that we have an equal number of robots and flock agents *i.e* $M = N$. By exploiting this equality, we assign a unique agent A_i for $i \in \{1, \cdots, N\}$ to a unique robot R_k for $k \in \{1, \cdots, N\}$ and make R_k responsible for herding A_i away from $\mathcal{P}$. In other words, R_k computes a velocity $\boldsymbol{u}_{R_k}$ that repels A_i from $\mathcal{P}$, thereby ensuring that $\boldsymbol{x}_{A_i}(t) \notin \mathcal{P} \ \forall t \geq 0$. The premise is that owing to the equality, each agent will end up being herded by a unique robot, therefore, no agent will breach the protected zone[2]. Now while this strategy necessitates having an equal number of robots and agent, the benefit of this approach stems from the feasibility guarantee (that we prove shortly), which the centralized approach lacks. Simple algebraic manipulation of constraint in (9.23) yields a constraint on the velocity of R_k as follows

$$A_{ik}^H \boldsymbol{u}_{R_k} \leq b_{ik}^H, \quad \text{where} \tag{9.24}$$

[2]Note that although A_i is assigned to R_k, the position of the remaining robots $\{1, \cdots, N\} \backslash k$ and the remaining agent $\{1, \cdots, N\} \backslash i$ do influence R_k's constraint parameters (A_i^H, b_i^H), and in turn, its computed velocity $\boldsymbol{u}_{R_k}^*$.

$$A_{ik}^H \coloneqq (\boldsymbol{x}_P - \boldsymbol{x}_{A_i})^T \mathbb{J}_{ki}^R$$
$$b_{ik}^H \coloneqq \boldsymbol{f}_i^T \boldsymbol{f}_i + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \sum_{j \in \mathcal{A}} \mathbb{J}_{ji}^A \boldsymbol{f}_j + \alpha (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i + \beta \frac{y_i}{2} - (\boldsymbol{x}_P - \boldsymbol{x}_{A_i})^T \sum_{l \in \mathcal{R} \backslash k} \mathbb{J}_{li}^R \boldsymbol{u}_{R_l}$$

Here $A_{ik}^H \in \mathbb{R}^{1\times 2}$ and $b_{ik}^H \in \mathbb{R}$. The term $\boldsymbol{u}_{R_l}$ in the expression of b_{ik}^H is computed by using numerical differentiation of the positions $\boldsymbol{x}_{R_l}$. We pose a QP to obtain the min-norm velocity for R_k as follows

$$\begin{aligned} \boldsymbol{u}_{R_k}^* &= \underset{\boldsymbol{u}_{R_k}}{\arg\min} \|\boldsymbol{u}_{R_k}\|^2 \\ \text{subject to} \quad & A_{ik}^H \boldsymbol{u}_{R_k} \le b_{ik}^H. \end{aligned} \tag{9.25}$$

This optimization problem calculates only the velocity for robot R_k *i.e.* $\boldsymbol{u}_{R_k}$ subject to constraints only on $\boldsymbol{u}_{R_k}$. Furthermore, the cost function also minimizes the effort only for R_k not the other robots. That is why (9.25) can be implemented locally on robot R_k, making it a distributed algorithm. It is by no means guaranteed that the $R_k's$ velocity computed by the centralized approach in (9.17) will be the same as the one computed by (9.25). We will remedy this issue in the next distributed algorithm. That being said, the velocity obtained from (9.25), if feasible, guarantees that the protected zone $\mathcal{P}$ will not be breached by A_i. Since each robot in $\mathcal{R}$ is in-charge of herding exactly one agent in $\mathcal{A}$, the feasibility of (9.25) $\forall k \in \mathcal{R}$ would ensure that no agent breaches $\mathcal{P}$. Thus, all that remains to show is that (9.25) is feasible. Before showing that, we state some assumptions.

Assumption 3. *We make the following assumptions on the distances between pairs of agents:*

1. *There exists a lower bound and upper bound on the distance between any pair of agents, i.e,* $L_S \le \left\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\right\| \le M_S$, $\forall i, j \in \{1, \cdots, N\}$ *and* $i \ne j$.
2. *There exists a lower bound on the distance between every agent and robot, i.e.,* $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\| \ge L_D$ $\forall i \in \{1, \cdots, N\}$ *and* $k \in \{1, \cdots, N\}$.
3. *There exists a upper bound on the distance between each agent and its goal i.e.,* $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_G\| \le M_G$ *and between the agent and the center of the protected zone i.e.,* $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \le M_P$.

Theorem 2. *In a scenario with 'N' robots and 'N' agents, with each agent assigned a unique robot, the herding constraint* (9.24) *for a given robot is always feasible, provided assumptions 3 are met.*

Proof. See appendix (section 9.7.2). □

9.3.2 Approach 2: Iterative reformulation of centralized controller

The distributed formulation proposed in (9.25) comes with a feasibility guarantee ensuring that all agents will be herded away from $\mathcal{P}$. While vital, this comes at the cost of requiring as many robots as the number of flock agents. This is because, in a way, this equality ensures that controlling the agents from the perspective of robots is not an underactuated problem. Be that as it may, in our simulations and experiments involving the centralized approach with an equal number of robots and agents, we frequently observed that not all robots needed to move to repel the agents away from $\mathcal{P}$ *i.e.,* equality may have been an overkill. Thus, in terms of budget efficiency, at least empirically, the centralized approach outweighs the distributed approach.

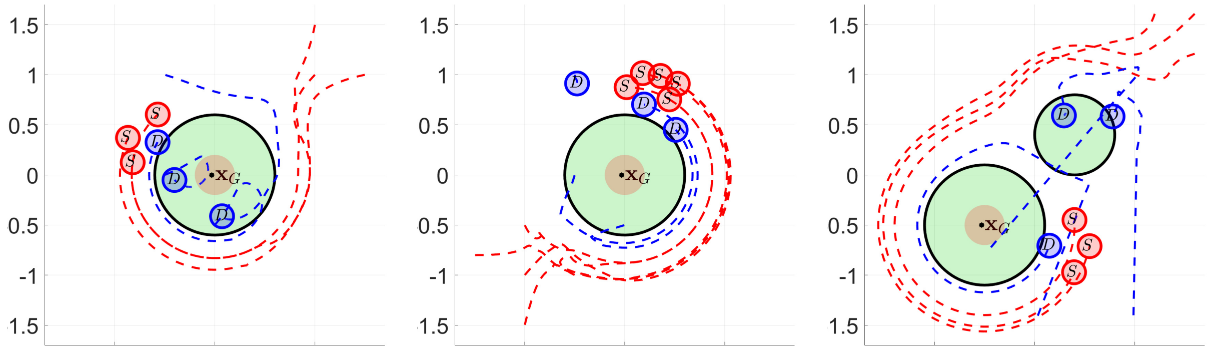
This raises the question, can we convert the centralized algorithm of (9.23) into a distributed version that inherits the budget efficiency (optimality) promised by (9.23)? Indeed, we found out that [89, 90] propose algorithms to convert constrained-coupled convex optimization problems (such as (9.23)) into distributed counterparts. They combine techniques called dual decomposition and proximal minimization and develop iterative distributed schemes which consist of local optimization problems. The solutions to these optimization problems asymptotically converge to the solution of centralized optimization under mild convexity assumptions and connectivity properties of the communication network. In our case, this network refers to the communication between robots. Below, we present the distributed dual sub-gradient method of [89, 90] adapted to the costs and constraints of (9.23). This algorithm calculates an estimate of robot R_k's velocity $\hat{\boldsymbol{u}}_{R_k}$ which, given large enough iterations T_{max}, matches with the k^{th} velocity component in the optimal velocities $\boldsymbol{u}_R^{*all}$ returned by (9.23). $A_k \in \mathbb{R}^{M\times 2}$ refers to those columns of A^H that correspond to $\boldsymbol{u}_{R_k}$ in $\boldsymbol{u}_R^{all}$.

Algorithm 1 Distributed Dual Subgradient for (9.23) (based on sec. 3.4.2 in [90])

```
**Initialize Lagrange Multiplier**: $\boldsymbol{\mu}_k^0 = \mathbf{0} \in \mathbb{R}^M$
**Evolution**: $t = 1, 2, \cdots, T_{max}$
    **Gather Multipliers** $\boldsymbol{\mu}_l^t$ from $R_l$ $\forall l \in \{1, \cdots, N\}\backslash k$
    **Average Multipliers**: $\boldsymbol{v}_k^{t+1} = \frac{1}{N}\sum_{l\in\{1,\cdots,N\}\backslash k}\boldsymbol{\mu}_l^t$
    **Local Solution**: $\boldsymbol{u}_{R_k}^{t+1} = \arg\min_{\boldsymbol{u}} \|\boldsymbol{u}\|^2 + (\boldsymbol{v}_k^{t+1})^T(A_k\boldsymbol{u} - \frac{1}{N}\boldsymbol{b}^H) = -\frac{1}{2}A_k^T\boldsymbol{v}_k^{t+1}$
    **Update Multiplier**: $\boldsymbol{\mu}_k^{t+1} = \Big[\boldsymbol{v}_k^{t+1} + \gamma_t\Big(A_k\boldsymbol{u}_{R_k}^{t+1} - \frac{1}{N}\boldsymbol{b}\Big)\Big]_+$
**Return Average**: $\hat{\boldsymbol{u}}_{R_k} = (1/T_{max})\sum_{t=1}^{T_{max}}\boldsymbol{u}_{R_k}^t$
```

9.4 Simulation Results

In this section, we show results of our centralized and distributed approaches by testing them on different scenarios with varying numbers of flock agents and robots, and varying initial positions. We represent the protected zone with a circular disc with radius R_p centered at the origin *i.e.* $\boldsymbol{x}_P = \mathbf{0}$. We purposefully choose the agents' goal to be the center of the protected zone *i.e.* $\boldsymbol{x}_G = \boldsymbol{x}_P$. This is done so that the agents are motivated to breach the protected zone should the robots not interfere. The initial positions $\boldsymbol{x}_{A_i}(0)$ of all agents are chosen such that they are all close to each other. This is done to ensure that the agents have enough time to stabilize as a flock before interacting with the robots. The initial positions $\boldsymbol{x}_R^{all}(0)$ of the robots are chosen randomly within the area of operation. The agents' velocities are calculated using (9.1). The values of the gains in the agent dynamics were taken as $k_G = 1$, $k_A = 0.3$ and $k_R = 0.08$.



(a) Three robots v/s three agents. (b) Three robots v/s five agents. (c) Three robots v/s three agents.

Figure 9.1: **Centralized Controller** for preventing the breaching of the protected zone. In these simulations, the robots are shown in blue and the flock agents are shown in red. The green disc represents the protected zone. The nominal task of the flock agents is to go straight towards goal $\boldsymbol{x}_G$. However, since this would result in infiltration of the protected zone, the robots intervene using the control algorithm presented in (9.22). In 9.1(c), we defend two protected zones from three agents.

9.4.1 Simulations for the Centralized Controller

The velocities of the defending robot were obtained using (9.22). The hyperparameters α, β, γ are tuned satisfy the conditions given in (9.8) and (9.11). Figure 9.1 shows three simulation results. In these simulations, we varied the initial positions of the flock agents (red), the robots (blue), the number of flock agents and the number of robots. It can be noticed from the figure that in all three scenarios, the robots are able to successfully intercept all agents and prevent them from entering the protected zone while also avoiding collisions with the agents.

We further study the performance of the proposed control strategy by using Monte Carlo simulations by varying the initial configurations and the number of flock agents M and the number of robots N. We vary these from one to ten, and for a given pair of (M, N), we run the simulation for a hundred times with random initializations of $\boldsymbol{x}_0 = (\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0), \boldsymbol{x}_{R_1}(0), \cdots, \boldsymbol{x}_{R_N}(0))$ in every run. Table 9.1 reports these results. Each entry of this table reports the percentage success rate *i.e.* in how many cases the agents got diverted away from the protected zone. As can be seen, almost all entries are 100, which proves the success of our algorithm. The failure cases correspond to scenarios when during the transition from the initial configuration to the final configuration, the QP becomes infeasible in certain cases and hence leads to breaching of the protected zone. When the QP becomes infeasible, we assign the robots to have zero velocity.

Further, we considered the impact of including collision avoidance constraints. The metrics with these constriants incorporated are reported in Table 9.2. Because it is possible that these collision avoidance constraints conflict with the defending constraints, the QP may become infeasible. Thus, we do not observe as good successes in this case compared to when there are no collision avoidance constraints. Lastly, we demonstrate one additional

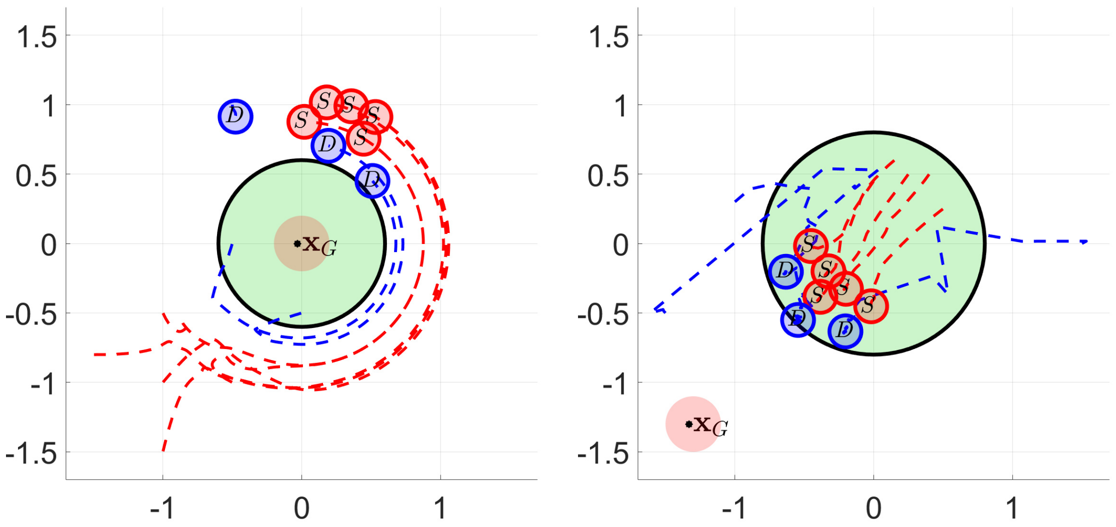
Table 9.1: Performance of the proposed strategy with varying number of flock agents and robots. Here, we did not consider collision avoidance constraints.

$M\backslash N$	**2**	**4**	**6**	**8**	**10**
2	100	100	100	100	100
4	100	100	100	100	100
6	100	98	100	100	100
8	100	98	100	100	98
10	100	98	98	100	96

Table 9.2: Performance of the proposed strategy with varying number of flock agents and robots. Here we considered collision avoidance constraints in the dynamics of the robots.

$M\backslash N$	**2**	**4**	**6**	**8**	**10**
2	72	99	99	100	100
4	62	74	90	97	100
6	28	83	99	99	100
8	63	82	100	100	100
10	70	79	90	91	94

benefit of our approach. Since our code relies on using automatic differentiation and symbolic computation tools for calculating all the gradients, we can easily change the behavioral requirements expected out of the agents. For example, instead of preventing them from breaching a protected zone (Fig. 9.2(a)), we can prevent them from escaping a home zone (Fig. 9.2(b)). In the next section, we present the results of our distributed optimization algorithms.



(a) Preventing breaching of protected zone (b) Preventing escape from protected zone

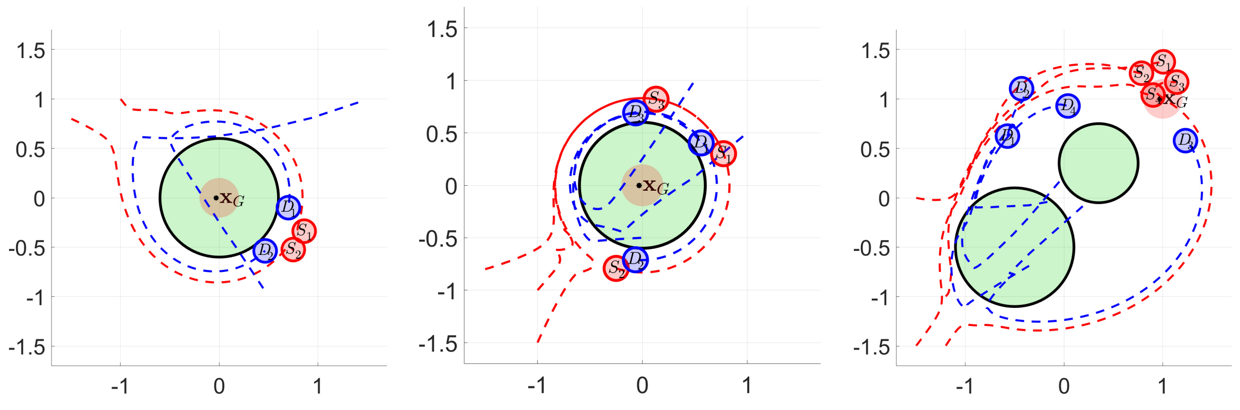
Figure 9.2: **Centralized Controller:** Demonstration of our results showing (a) how to prevent agent (red) from breaching a protect zone (green) and (b) preventing agent (red) from escaping the protected zone using robots (blue).

9.4.2 Simulations for the Distributed Controllers

In this section, we demonstrate the effectiveness of the distributed controller developed in (9.25). This algorithm requires equal numbers of flock agents and robots. Figures 9.3(a) and 9.3(b) shows two examples involving a) two robots vs. two flock agents and b) three robots vs. three flock agents. To demonstrate the compositionality of our approach, we consider two protected zones in figure 9.3(c) where we have four robots defending both zones from four flock agents. In all these simulations, none of the flock agents breach any zone, thus demonstrating the correctness of our approach. In the interest of space, we skip the simulation results for the algorithm in 9.3.2 but do provide experimental results.

9.5 Experimental Results

Finally, we tested our algorithm in robots in the multirobot test arena in our lab. It consists of a 14ft $\times$ 7ft platform, several Khepera IV robots and additionally eight Vicon cameras for motion tracking. All control inputs are computed on a desktop and conveyed to the robots over WiFi. While we developed our algorithms assuming that the dynamics of



(a) Two robots v. two flock agents. (b) Three robots v. three agents (c) Four robots v. four flock agents

Figure 9.3: **Distributed Controller**: Preventing the breaching of the protected zone using our proposed distributed controller in (9.25). Here robots are shown in blue and flock agents in red. The green disc represents the protected zone. The nominal task of the flock agents is to go straight towards goal $\boldsymbol{x}_G$. However, since this would result in infiltration of the protected zone, the dog intervenes using the distributed control algorithm. In Fig. 9.3(c), we defend two protected zones from four flock agents.

all agents are single-integrator based, the robots have unicycle dynamics given by

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}v\cos\theta\\ v\sin\theta\\ \omega\end{pmatrix} \tag{9.26}$$

Thus, we do a minor adjustment to map the inputs computed from our algorithms to the angular speed and forward translational speed of these robots. This is done by considering a point at a distance d on the x_b axis of the body frame of the robot:

$$\begin{aligned}\boldsymbol{x} &= \begin{pmatrix}x + d\cos\theta\\ y + d\sin\theta\end{pmatrix}\\ \implies \dot{\boldsymbol{x}} &= \underbrace{\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & d\end{pmatrix}}_{M}\begin{pmatrix}v\\ \omega\end{pmatrix} = \tilde{\boldsymbol{u}}\\ \implies \begin{pmatrix}v\\ \omega\end{pmatrix} &= M^{-1}\tilde{\boldsymbol{u}} \end{aligned} \tag{9.27}$$

9.5.1 Robot Experiments with the centralized controller

For flock agent robots, $\tilde{\boldsymbol{u}}$ is obtained from (9.1) while for the defending robots, $\tilde{\boldsymbol{u}}$ is obtained from (9.17). In Fig. 9.4, we have one agent (in red box) and one defending robot (in blue box). The protected zone is highlighted in green and the goal of the sheep is the black dot. We use (9.17) to compute the velocity of the robot and convert it to angular and translational speeds using (9.27). As can be noted from the snapshots, the defending robot is able successfully defend the zone from the agent. Finally, in Fig. 9.5 we demonstrate that our approach can deal with multiple protected zones simultaneously. We purposefully kept the goal of the agent in the left most protected zone so the agent would be incentivized to breach both zones. Yet still, our algorithm is able to find velocities for robots to defend both the zones from both agents. Figure 9.6 shows a case with 2 robots and 4 flock agents. The

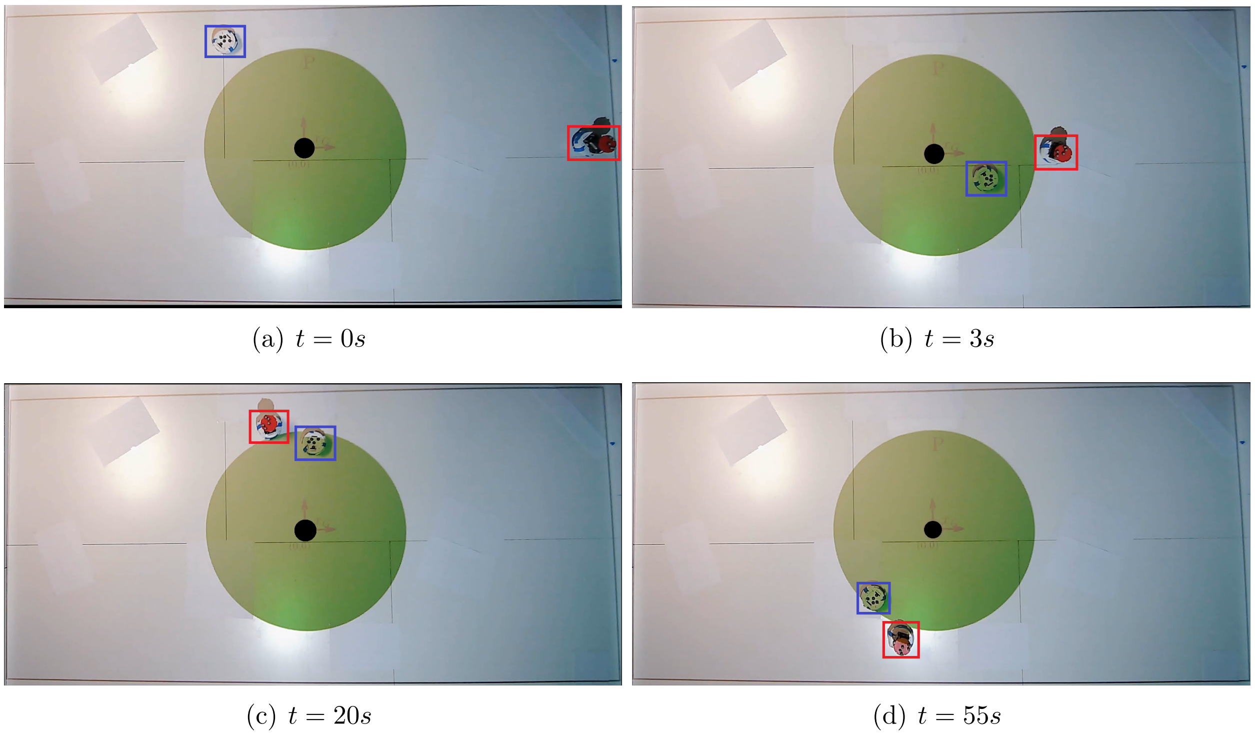
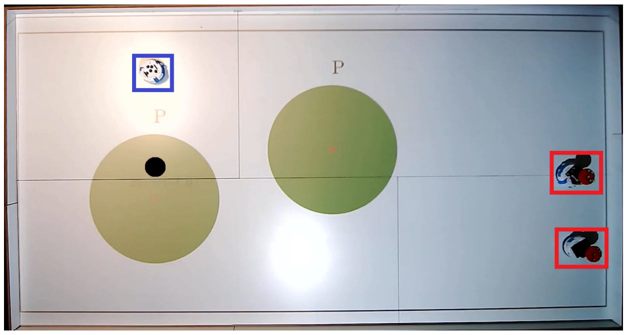
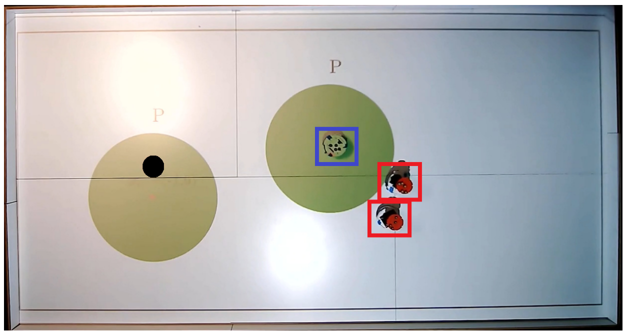
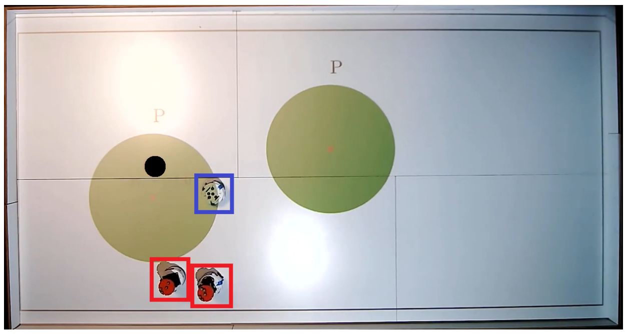

(a) $t = 0s$ (b) $t = 3s$ (c) $t = 20s$ (d) $t = 55s$

Figure 9.4: **Experiments for Centralized Control:** One defending robot preventing one agent from the breaching of the protected zone. The defending robot is highlighted in blue and the agent in red. The goal position x_G is at the center of the protected zone and given as a black solid circle. The nominal task of the agent is to go straight towards its goal $\boldsymbol{x}_G$. However, since this would result in infiltration of the protected zone, the robot intervenes using the control algorithm presented in (9.17). Video at `https://tinyurl.com/2p9fjeft`.

robots have a green tail, and the flock agents have an orange tail. The tails are pointing in the opposite direction of the robot's heading angle. Another example is shown in figure 9.7 where 3 robots successfully prevent breaching against 5 flock agents.


(a) $t = 0s$


(b) $t = 7s$


(c) $t = 10s$

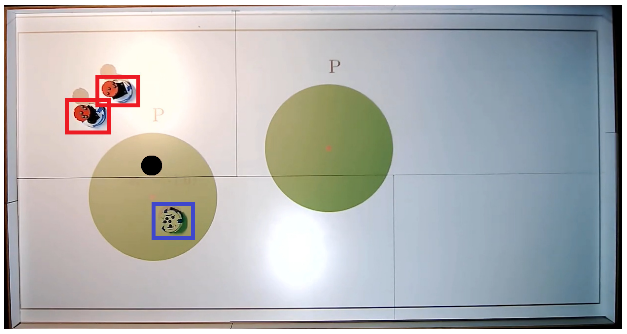
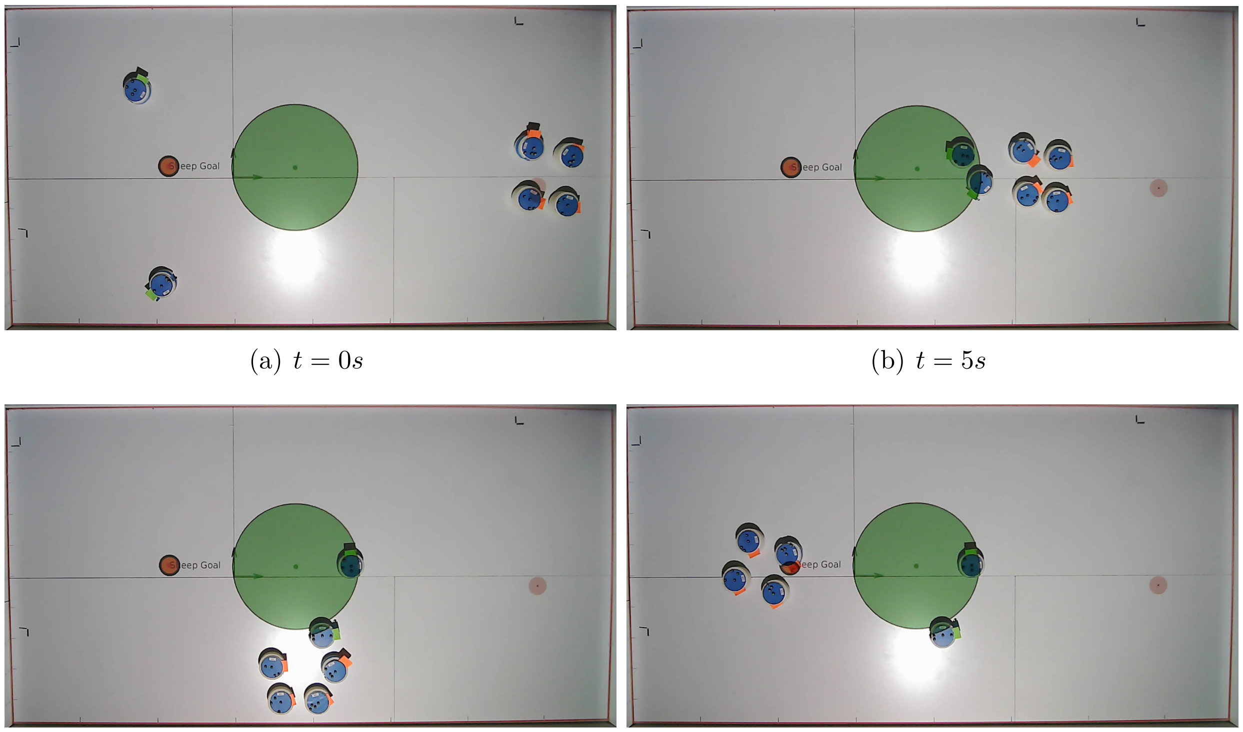

(d) $t = 14s$

Figure 9.5: **Experiments for Centralized Control:** One defending robot preventing two agent from the breaching of two protected zones. The goal lies in the left most protected zone. Video at `https://tinyurl.com/ycuyhwe6`.

9.5.2 Robot experiments with distributed controllers

Following that, multiple experiments were conducted using the distributed algorithm presented in section 9.3.1, which requires equal number of robots and flock agents. Figure 9.8 shows 4 robots against 4 flock agents. Here we take two protected zones and show that the robots can protect both of them. This highlights the compositional nature of our algorithm. We conducted experiments with 5 robots and 5 flock agents, as shown in Figure 9.9. Here we can see some robots did not move as fewer robots were enough to repel all flock agents from the protected zone. Finally, we test our distributed algorithm presented in section 9.3.2. Figure 9.10 shows a case where 2 robots prevent the breaching of protected zone against three robots. This highlights that our distributed approach can handle underactuated scenarios. Figure 9.11 and figure 9.6 can be compared to see both centralized and distributed algorithm handling a similar scenario of 2 robots against 4 flock agents.



(a) $t = 0s$

(b) $t = 5s$

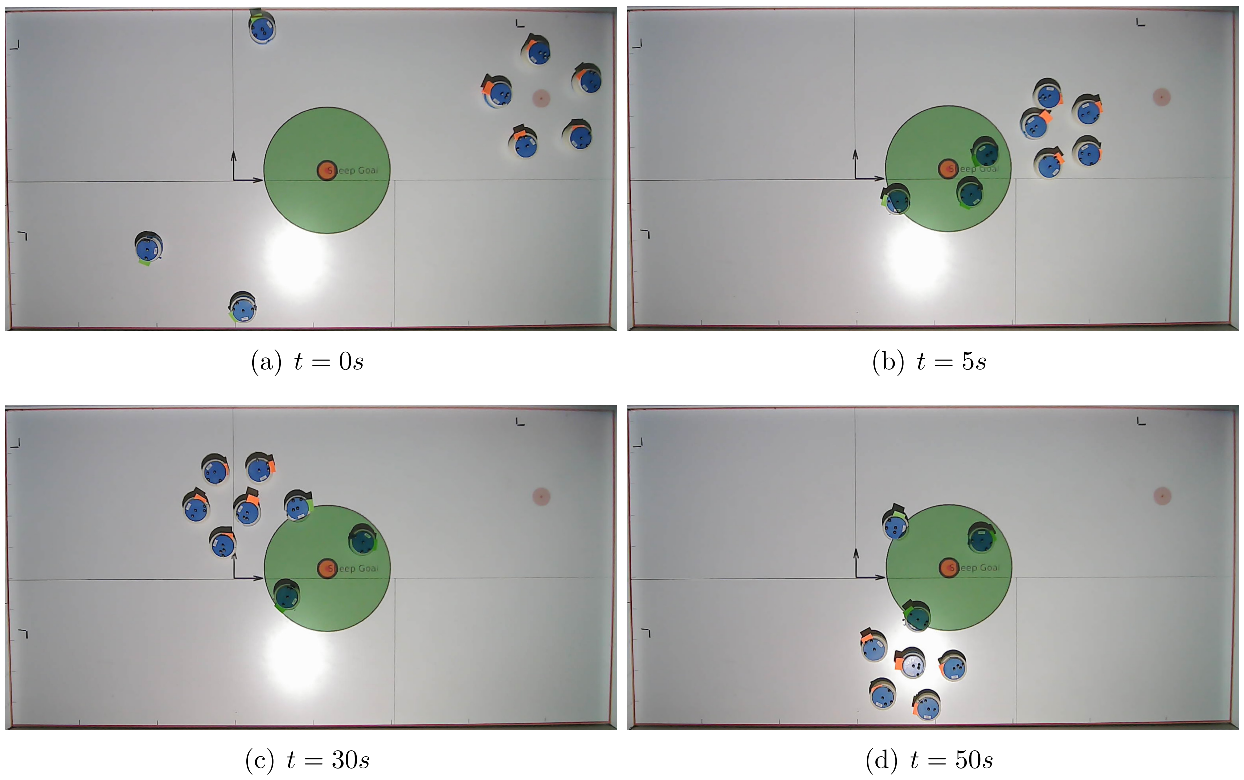
(c) $t = 12s$

(d) $t = 30s$

Figure 9.6: **Experiments for Centralized Control:** Two robots defending the protected zone from four flock agents using centralized control algorithm (9.17). Video at `https://bit.ly/3OTAnOu`.

9.6 Conclusions

In this chapter, we developed optimization-based control algorithms for a group of robots to prevent a flock of agents from breaching a protected zone. We developed both centralized and distributed algorithms. Our constraint based framework allowed us to include multuple protected zones. Through empirical simulations, we concluded that the centralized algorithm allows for defense against as many as 10 agents using as few as 6 robots. Next, we built on top of this algorithm to come up with two distributed implementations. For the first distributed algorithm, we kept feasibility as the main criterion and provided proof of feasibility of the controller when the number of flock agents and robots are equal. We developed another distributed algorithm that iteratively computes a solution that agrees with the solution returned by the centralized problem without requiring equal number of robots and flock agents. We experimentally validated all algorithms on the multirobot arena in our lab. In future work, we aim to analyze the feasibility of these algorithms in the presence of actuator limit constraints and evaluate their resilience under sitations where the true model of the flock agents differs significantly from the model assumed by the robots.



(a) $t = 0s$ (b) $t = 5s$ (c) $t = 30s$ (d) $t = 50s$

Figure 9.7: **Experiment for Centralized Control:** Three robots (green-tailed robots) defending a protected zone from five flock agents (orange-tailed robots) using centralized control (9.17). Video at https://youtu.be/2_Xuxnd9jZw.

9.7 Appendix

9.7.1 Proof of feasibility of the centralized controller

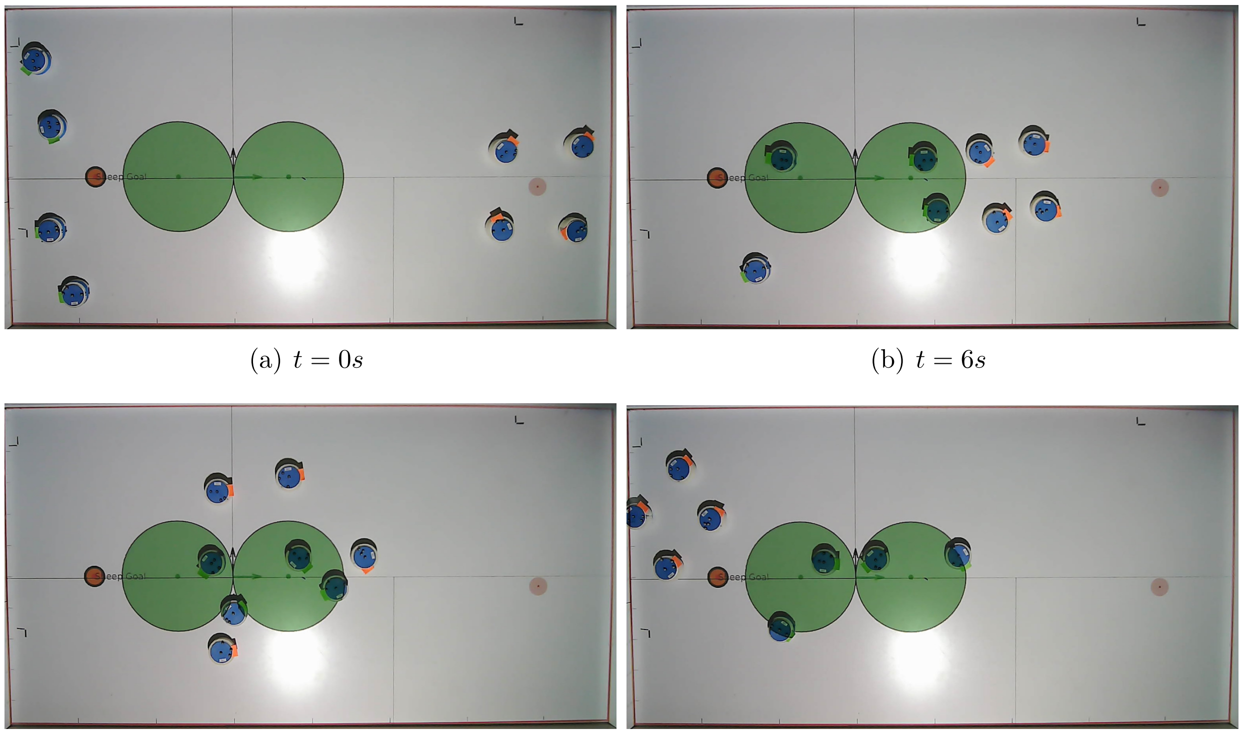
Theorem 1. *If there is one robot and one agent, then* (9.17) *always has a solution.*

Proof. Let the position of the robot be $\boldsymbol{x}_R$ and that of the agent be $\boldsymbol{x}_A$. The agent dynamics can be simplified to

$$\dot{\boldsymbol{x}}_A = \boldsymbol{f}(\boldsymbol{x}_A, \boldsymbol{x}_R) = k_G\left(\boldsymbol{x}_G - \boldsymbol{x}_A\right) + k_R \frac{\boldsymbol{x}_A - \boldsymbol{x}_R}{\|\boldsymbol{x}_A - \boldsymbol{x}_R\|^3} \tag{9.28}$$

The only case when (9.17) does not have a solution is when the defending constraint $A^H \boldsymbol{u}_R \leq b^H$ is infeasible. This can occur

- either when $A^H = \mathbf{0}$ and $b^H < 0$ (possibility 1)
- or when $b^H = -\infty$ (possibility 2).



(a) $t = 0s$ (b) $t = 6s$

(c) $t = 12s$ (d) $t = 20s$

Figure 9.8: **Experiment for the distributed algorithm in section 9.3.1 :** Four robots (green-tailed robots) defending two protected zone from four flock agents (orange-tailed robots). The goal position x_G (red disc) is in extreme left that would encourage flock agents to breach both zones. However, our proposed algorithm moves the robots so that none of the zones get breached. Video at `https://bit.ly/3yo9ziC`.

For this case A^H is:

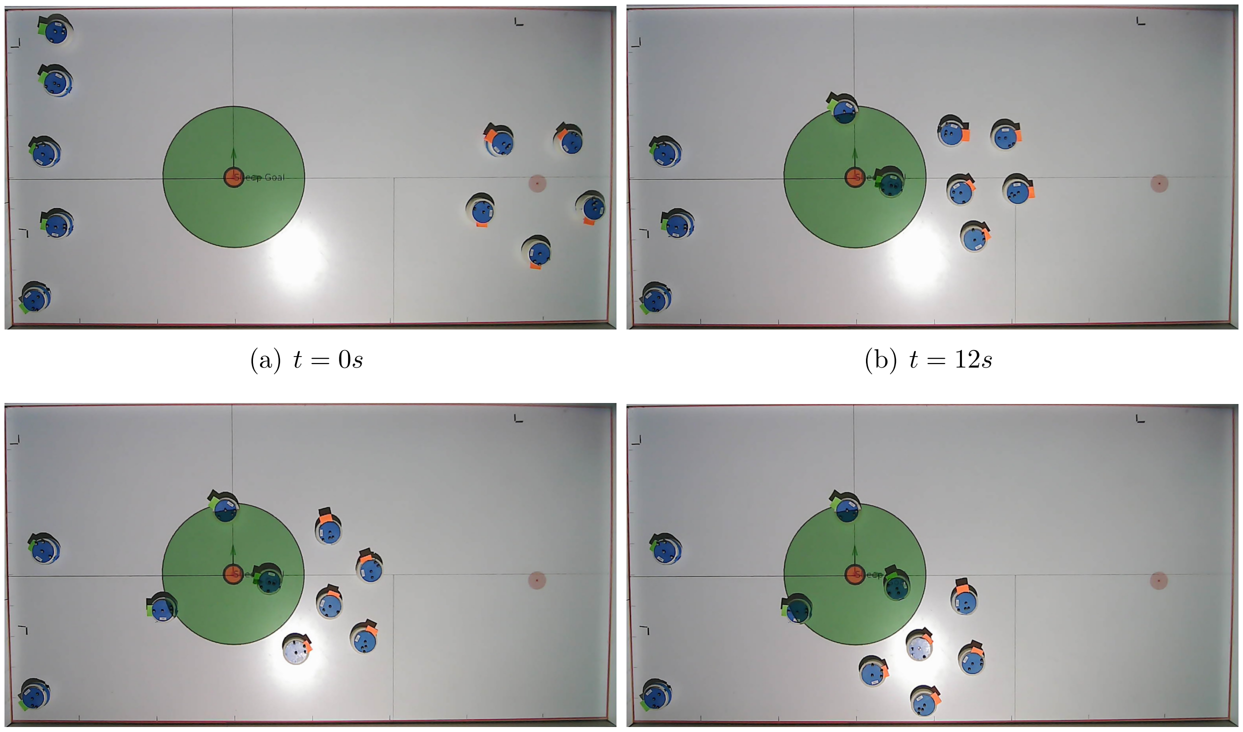
$$A^H = (\boldsymbol{x}_P - \boldsymbol{x}_A)^T \mathbb{J}_{11}^R \tag{9.29}$$

Thus, if $\mathbb{J}_{11}^R$ is non-singular, $(\boldsymbol{x}_P - \boldsymbol{x}_A)^T \mathbb{J}_{11}^R \neq \mathbf{0}$. From our calculations, we find that the determinant of $\mathbb{J}_{11}^R$ is

$$det(\mathbb{J}_{11}^R) = \frac{-2k_R^2}{\|\boldsymbol{x}_R - \boldsymbol{x}_A\|^3} \tag{9.30}$$

As long as the distance between the robot and the agent is finite, $det(\mathbb{J}_{11}^R)$ is always non zero. Thus, there exists no null space for the jacobian matrix $\mathbb{J}_{11}^R$. This implies $A^H \neq \mathbf{0}$ $\forall \boldsymbol{x}_A \in \mathbb{R}^n, \boldsymbol{x}_R \in \mathbb{R}^2$. This rules out possibility 1 for infeasibility. For possibility 2, we need to examine when does $b^H \longrightarrow -\infty$. The expression for b^H is:

$$b^H = \boldsymbol{f}^T \boldsymbol{f} + (\boldsymbol{x}_A - \boldsymbol{x}_P)^T \mathbb{J}_{11}^A \boldsymbol{f} + \alpha(\boldsymbol{x}_A - \boldsymbol{x}_P)^T \boldsymbol{f} + \beta \frac{y}{2}$$



(a) $t = 0s$ (b) $t = 12s$

(c) $t = 25s$ (d) $t = 40s$

Figure 9.9: **Experiment for the distributed algorithm in section 9.3.1 :** Five robots (green-tailed robots) defending the protected zone from five flock agents (orange-tailed robots). The flock agents's goal (red disc) is in the center of the protected zone. Eventually, in this scenario a deadlock occurs where all flock agents come to a stop outside the protected zone. Video at `https://bit.ly/3o51Cu1`.

We want to find the worst case lower bound of b^H. Here $\boldsymbol{f}^T\boldsymbol{f} \geq 0$ always. We assume that at the current time step, the agent is outside the $\mathcal{P}$, this ensures $\beta\frac{y}{2} \geq 0$.

Assumption 4. *Assume* $\|\boldsymbol{x}_A - \boldsymbol{x}_G\| \leq M_1, \|\boldsymbol{x}_A - \boldsymbol{x}_P\| \leq M_2$ *and* $\|\boldsymbol{x}_A - \boldsymbol{x}_R\| \geq M_3 \; \forall t.$

With these assumptions, we can lower bound b^H as follows:

$$\begin{aligned} b^H &\geq (\boldsymbol{x}_A - \boldsymbol{x}_P)^T \mathbb{J}_{11}^A \boldsymbol{f} + \alpha(\boldsymbol{x}_A - \boldsymbol{x}_P)^T \boldsymbol{f} \\ &\geq -(\sigma_{max}(\mathbb{J}_{11}) + \alpha) \|\boldsymbol{f}\| \, \|\boldsymbol{x}_A - \boldsymbol{x}_P\| \\ &\geq -(\sigma_F(\mathbb{J}_{11}) + \alpha) \|\boldsymbol{f}\| \, \|\boldsymbol{x}_A - \boldsymbol{x}_P\| \end{aligned} \tag{9.31}$$

Here $\|\boldsymbol{f}\| \leq k_G \|\boldsymbol{x}_A - \boldsymbol{x}_G\| + \frac{k_R}{\|\boldsymbol{x}_A - \boldsymbol{x}_R\|^2}$ using triangle inequality on (9.28). This gives $\|\boldsymbol{x}_A - \boldsymbol{x}_P\| \|\boldsymbol{f}\| \leq k_G M_1 M_2 + \frac{k_R M_2}{M_3^2}$. We can show that $\sigma_F(\mathbb{J}_{11}) \leq \lambda_M :=$

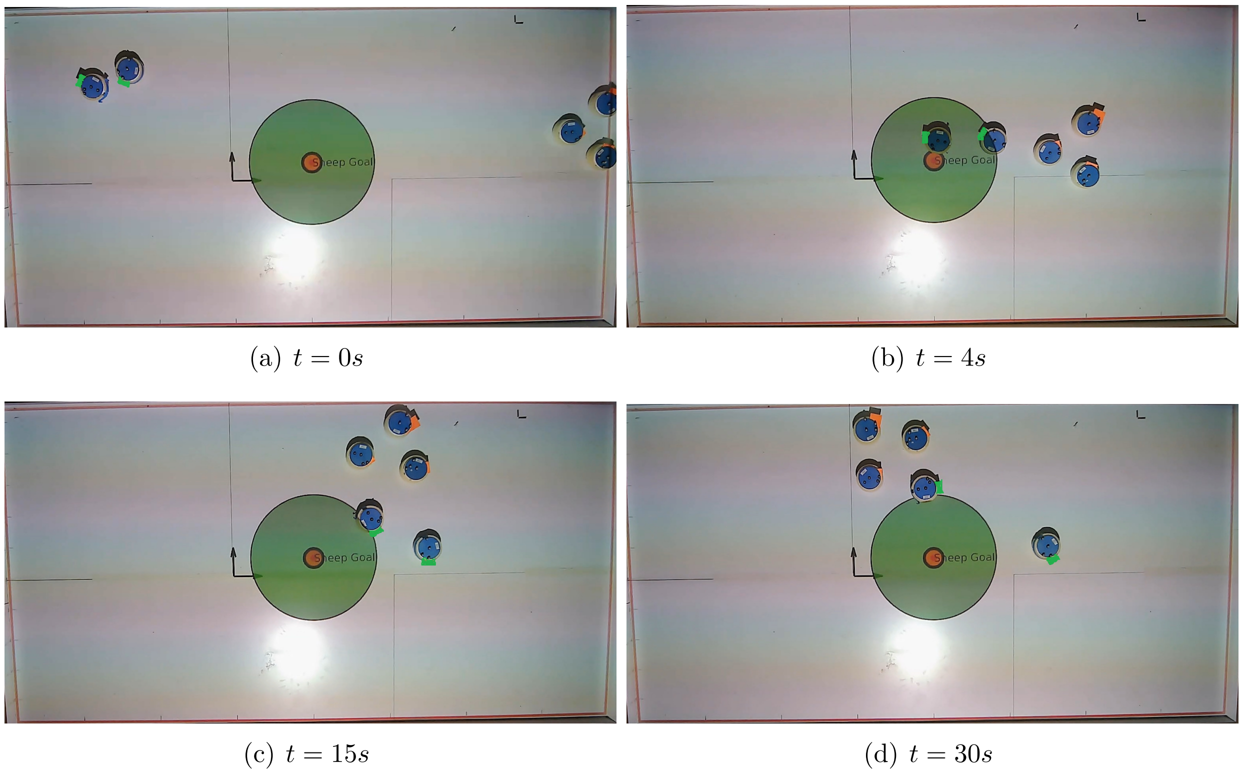


Figure 9.10: **Experiment for distributed algorithm in section 9.3.2 :** Two robots (green-tailed robots) defending the protected zone from three flock agents (orange-tailed robots). The goal position x_G (red disc) is at the center of the zone. Video at https://youtu.be/IbCjkR1ye0c.

$\sqrt{2k_G^2 + 5\frac{k_R^2}{M_3^6} + \frac{2k_Gk_R}{M_3^3}}$. Thus, using this, we obtain the following lower bound for b^H

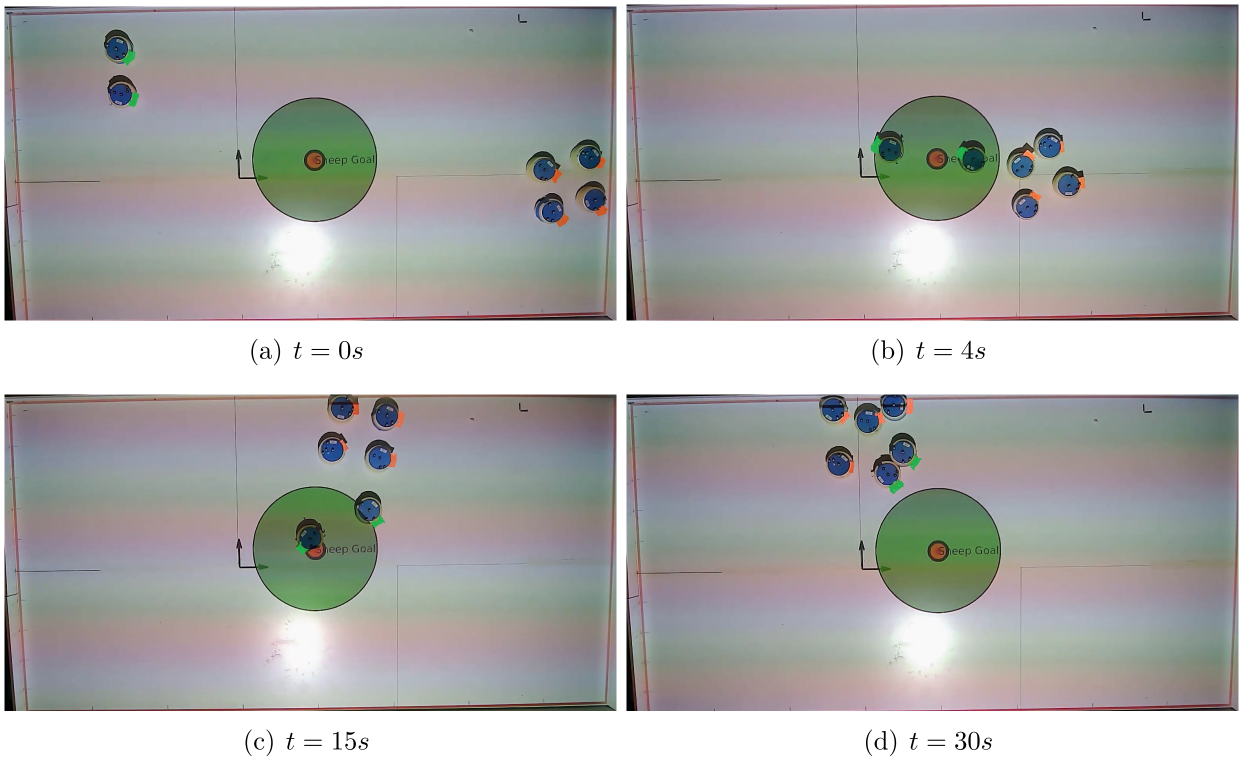
$$b^H \geq -(\lambda_M + \alpha)\left(k_G M_1 M_2 + \frac{k_R M_2}{M_3^2}\right) \tag{9.32}$$

This shows that b^H is lower bounded and thus does not reach $-\infty$. Hence possibility 2 is also ruled out. Thus, (9.17) is always feasible. □

9.7.2 Proof of feasibility of the distributed controller

Theorem 2. *In a scenario with 'N' robots and 'N' agent, with each robot assigned a unique agent, the herding constraint (9.24) for a given robot is always feasible, provided assumptions 3 are met.*

Proof. Our strategy to guarantee feasibility of constraint (9.24) relies on ruling out situations in which it is infeasible. (9.24) can become infeasible



(a) $t = 0s$ (b) $t = 4s$ (c) $t = 15s$ (d) $t = 30s$

Figure 9.11: **Experiment for distributed algorithm in section 9.3.2) :** Two robots (green-tailed robots) defending the protected zone from four flock agents (orange-tailed robots). This case is similar to the one shown in fig. 9.6. Video at `https://youtu.be/51FoHZWFYC4`.

- either when $A_{ik}^H = \mathbf{0}$ and $b_{ik}^H < 0$ (possibility 1)
- or when $b_{ik}^H = -\infty$ (possibility 2).

To determine when possibility 1 may occur, we calculate the determinant of $\mathbb{J}_{ki}^R$ as

$$det(\mathbb{J}_{ki}^R) = \frac{-2k_R^2}{\|\boldsymbol{x}_{R_k} - \boldsymbol{x}_{A_i}\|^3}.$$

$det(\mathbb{J}_{ki}^R)$ is non-zero as long as the distance between robot R_k and agent A_i is finite. Therefore, $\mathbb{J}_{ki}^R$ will have no null space, implying that $A_{ik}^H \neq 0 \ \forall \boldsymbol{x}_{A_i} \in \mathbb{R}^2, \boldsymbol{x}_{R_k} \in \mathbb{R}^2$. This rules out possibility 1 for infeasibility. To rule out possibility 2, we need to check for condition when $b_{ik}^H \longrightarrow -\infty$. Given b_{ik}^H in (9.14), we find its worst case lower bound. Here $\boldsymbol{f}_i^T \boldsymbol{f}_i \geq 0$ and as we assume that at the current time step, the agent is outside $\mathcal{P}$, this ensures $\beta \frac{y_i}{2} \geq 0$. By removing these terms, the lower bound of b_{ik}^H can be given as

$$b_{ik}^H \geq (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \sum_{j=1}^{M \setminus i} \mathbb{J}_{ji}^A \boldsymbol{f}_j + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \mathbb{J}_{ii}^A \boldsymbol{f}_i + \alpha (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \boldsymbol{f}_i + (\boldsymbol{x}_{A_i} - \boldsymbol{x}_P)^T \sum_{l=1}^{N \setminus k} \mathbb{J}_{li}^R \boldsymbol{u}_{R_l}$$

Using the triangle and Cauchy-Schwarz inequalities, we get

$$
\begin{aligned}
b_{ik}^H \geq & \sum_{j=1}^{M\backslash i} \left(-\sigma_{max}\left(\mathbb{J}_{ji}^A\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_j\|\right) - \sigma_{max}\left(\mathbb{J}_{ii}^A\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_i\| - \alpha \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_i\| \\
& + \sum_{l=1}^{N\backslash k} \left(-\sigma_{max}\left(\mathbb{J}_{li}^R\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{u}_{R_l}\|\right)
\end{aligned}
$$

where σ_{max} is the largest singular value of a matrix. Further, using the fact that the largest singular value of a matrix (σ_{max}) is upper bounded by its Frobenius norm (σ_F), we obtain

$$
\begin{aligned}
b_{ik}^H \geq & \sum_{j=1}^{M\backslash i} \left(-\sigma_F\left(\mathbb{J}_{ji}^A\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_j\|\right) - \sigma_F\left(\mathbb{J}_{ii}^A\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_i\| - \alpha \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{f}_i\| \\
& + \sum_{l=1}^{N\backslash k} \left(-\sigma_F\left(\mathbb{J}_{li}^R\right) \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\| \|\boldsymbol{u}_{R_l}\|\right)
\end{aligned}
$$

Now to compute this lower bound we make use of assumption 3. We use the dynamics in (9.1) to compute $\mathbb{J}_{ii}^A$ and obtain the upper bound on $\sigma_F\left(\mathbb{J}_{ii}^A\right)$ and use the bounds on distances from assumption 3 to get following upper bound:

$$
\begin{aligned}
\sigma_F\left(\mathbb{J}_{ii}^A\right) &\leq \sum_{j=1}^{N\backslash i} k_A \left(\sqrt{2} + \frac{\sqrt{5}R^3}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\|^3}\right) + \sqrt{2}k_G + \sum_{l=1}^{N} \frac{\sqrt{5}k_R}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_l}\|^3} && (9.33) \\
&\leq (N-1)\left(\sqrt{2}k_A + \frac{\sqrt{5}k_A R^3}{L_S^3}\right) + \sqrt{2}k_G + N\left(\frac{\sqrt{5}k_R}{L_D^3}\right) := \lambda_M && (9.34)
\end{aligned}
$$

We omit the proof of this computation in the interest of space. Similarly, using the dynamics in (9.1), we compute an expression for $\mathbb{J}_{ji}^A$ and obtain an upper bound on $\sigma_F\left(\mathbb{J}_{ji}^A\right)$ as follows:

$$
\sigma_F\left(\mathbb{J}_{ji}^A\right) \leq \sqrt{2}k_A + \frac{\sqrt{5}k_A R^3}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\|^3} \leq \sqrt{2}k_A + \frac{\sqrt{5}k_A R^3}{L_S^3} := \lambda_S
$$

Likewise, an upper bound of $\sigma_F\left(\mathbb{J}_{li}^R\right)$, is given by

$$
\sigma_F\left(\mathbb{J}_{li}^R\right) \leq \frac{\sqrt{5}k_R}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_l}\|^3} \leq \frac{\sqrt{5}k_R}{L_D^3} := \lambda_D
$$

Lastly, we use obtain an upper bound on the dynamics of each agent $\boldsymbol{f}_i$ as:

$$
\|\boldsymbol{f}_i\| \leq \sum_{j \in \mathcal{S}\backslash i} k_A \left(\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\| + \frac{R^3}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\|^2}\right) + k_G\|\boldsymbol{x}_G - \boldsymbol{x}_{A_i}\| + \sum_{l \in \mathcal{D}} k_R \frac{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_l}\|}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_l}\|^3}
$$

Now we need to compute the maximum possible value of the RHS to get the upper bound of the agent dynamics. The first term has a local minima at $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\| = (2)^{1/3}R$. Therefore the maximum value can occur at either the lower bound or upper bound of $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\|$. Thus the maximum value of the first term can be given as $F_{max} := \max(k_A L_S + k_A \frac{R^3}{L_S^2}, k_A M_S + k_A \frac{R^3}{M_S^2})$. Second term is maximum when $\|\boldsymbol{x}_G - \boldsymbol{x}_{A_i}\| = M_G$. The last term is maximum when distance of the agent to the robots are minimum, $\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\| = L_D$. Using these the upper bound on the agent dynamics is computed as:

$$\|\boldsymbol{f}_i\| \leq (n-1)F_{max} + k_G M_G + n k_R \left(\frac{1}{L_D^2}\right)$$

Assuming that the velocity of the robots have an upper bound, and by taking the upper bound on the dynamics of all the agent to be equal, the lower bound on b_{ik}^H from is (taking $\gamma = -(\alpha + \lambda_M + (n-1)\lambda_S)M_p$)

$$b_i^H \geqslant \gamma \left\{(n-1)F_{max} + k_G M_G + \frac{n k_R}{L_D^2}\right\} - (n-1)\lambda_D M_P \|\boldsymbol{u}_D\|_{\max}$$

This shows that b_i^H has a finite lower bound, thus ruling out possibility 2. Thus, the herding constraint (9.14) for a one robot to repel one agent from the protected zone is always feasible. Since each agent in $\mathcal{S}$ is allocated to one unique robot in $\mathcal{D}$, extension of this feasibility result to all agent ensures that none of them will breach the protected zone. $\square$

10 Robust Control Through Interactions

10.1 Introduction

In the last two chapters, we considered the known-model behavior shaping problem as we defined in Def. 2. In solving this problem, we made the assumption that the latent parameters in the dynamics of the flock agents were known a-priori to the dog robots and they used these parameters to synthesize their velocities that guaranteed that the protected zone will not get breached (the constraint terms A_i^H and b_i^H in (9.23) explicitly depend on the parameters of the agent dynamics). In this chapter, we relax this assumption and consider the uncertain model behavior shaping problem as stated in Def. 3. The observer's dog robots instead have estimates of parameters and an upper bound on the estimation error. Using these two metrics, the dog robots must synthesize velocities to meet the no breach requirement. We describe how this uncertain model behavior shaping problem can be cast as a convex optimization problem that can be solved in real time.

The outline of this chapter is as follows. In section 10.2 we recall the dynamic model of the sheep flock, the uncertain model behavior shaping problem and the known-model centralized control algorithm *i.e.* (9.23). In section 10.3, we build on this controller and modify the constraints so that they can accommodate the nominal estimates of flock's dynamic parameters and the uncertainties in these parameters. We show how incorporating uncertainty results in a convex semi-infinite program solving which is computationally intractable. Using two independent philosophies, one based on weak duality and another based on the s-procedure, we convert this problem to a tractable finite-dimensional convex optimization, to wit, a semidefinite program (SDP), which is roughly nearly as easy to solve as a modestly sized QP. Thus, we show that the problem of computing safe controls with uncertainty can be posed as a tractable convex problem and can be solved online. To validate this strategy, we show simulations as well as conduct experiments where the dog robots are required to defend the protected zone with uncertainty in the goal of the flock agents. These results are presented in section 10.4. Finally, we conclude in section 10.6 with outlook for future work.

10.2 Problem Formulation

Recall that we have M flock agents denoted by $\mathcal{A} \coloneqq \{1, 2, \cdots, M\}$ located at positions $\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}$ and N robots $\mathcal{R} \coloneqq \{1, 2, \cdots, N\}$ located at positions $\boldsymbol{x}_{R_1}, \cdots, \boldsymbol{x}_{R_N}$ respectively. The dynamics of the agent i in the flock are

$$\dot{\boldsymbol{x}}_{A_i} = -k_G(\boldsymbol{x}_{A_i} - \boldsymbol{x}_G) - k_A \sum_{j \in \mathcal{A} \setminus i} \left(1 - \frac{{D_s}^3}{\left\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}\right\|^3}\right)(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{A_j}) + k_R \sum_{k \in \mathcal{R}} \frac{(\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k})}{\|\boldsymbol{x}_{A_i} - \boldsymbol{x}_{R_k}\|^3} \tag{10.1}$$

The parameters indicated in red are unknown to the observer. We define the aggregated set of parameters as

$$\boldsymbol{\theta} \coloneqq (k_G, k_G\boldsymbol{x}_G, k_A, k_A D_s^3, k_R) \tag{10.2}$$

With representation of parameters, the dynamics of A_i in (10.1) can be rewritten as

$$\dot{\boldsymbol{x}}_{A_i} = G_i\Big(\{\boldsymbol{x}_{A_i}\}_{i \in \mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k \in \mathcal{R}}\Big)\boldsymbol{\theta} + \boldsymbol{f}_i\Big(\{\boldsymbol{x}_{A_i}\}_{i \in \mathcal{A}}, \{\boldsymbol{x}_{R_k}\}_{k \in \mathcal{R}}\Big) \tag{10.3}$$

Here G_i and $\boldsymbol{f}_i$ are matrices of basis functions that depend on the positions of all agents and the positions of all robots. Each robot's dynamics are

$$\dot{\boldsymbol{x}}_{R_k} = \boldsymbol{u}_{R_k} \;\; \forall k \in \mathcal{R} \tag{10.4}$$

The multiagent behavior shaping problem with uncertainty is recalled from problem 3,

Problem 1 (Multiagent Behavior Shaping With Uncertain Model). *Assume that observer has estimates of the parameters given by $\hat{\boldsymbol{\theta}}$ and an upper bound on estimation error $\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\| \leq \eta$. If the initial agent positions $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \in \mathcal{Y}$, find robot controls $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \in \mathcal{Y}\ \forall t > 0$. However, if the agent positions $(\boldsymbol{x}_{A_1}(0), \cdots, \boldsymbol{x}_{A_M}(0)) \notin \mathcal{Y}$, find $(\boldsymbol{u}_{R_1}, \cdots, \boldsymbol{u}_{R_N})$ such that $(\boldsymbol{x}_{A_1}(t), \cdots, \boldsymbol{x}_{A_M}(t)) \rightsquigarrow \mathcal{Y}$ in finite time.*

For the task of defending the protected zone $\mathcal{P} \coloneqq \{\boldsymbol{x} \in \mathbb{R}^2 | \, \|\boldsymbol{x} - \boldsymbol{x}_P\| \leq R_p\}$, we chose to define the set $\mathcal{Y}$ as

$$\mathcal{Y} \coloneqq \{(\boldsymbol{x}_{A_1}, \cdots, \boldsymbol{x}_{A_M}) \in \mathbb{R}^{2M} | \, \|\boldsymbol{x}_{A_i} - \boldsymbol{x}_P\|^2 - R_p^2 \geq 0 \; \forall \, i \in \mathcal{A}\} \tag{10.5}$$

In chapter 9, we derived the following constraints on the velocities of dog robots to ensure that the i^{th} agent stays outside $\mathcal{P}$

$$A_i^H(\boldsymbol{\theta})\boldsymbol{u}_R^{all} \leq b_i^H(\boldsymbol{\theta}). \tag{10.6}$$

We explicitly highlight $\boldsymbol{\theta}$ to emphasize the dependence of A_i^H and b_i^H on $\boldsymbol{\theta}$. In fact, using the expressions for A_i^H and b_i^H derived in (9.15), we can show that A_i^H exhibits an affine dependence on $\boldsymbol{\theta}$ while b_i^H exhibits a quadratic dependence on $\boldsymbol{\theta}$. To defend against all agents in $\mathcal{A}$, we augmented constraints for all agents in $\mathcal{A}$ as follows

$$A_i^H(\boldsymbol{\theta})\boldsymbol{u}_R^{all} \leq b_i^H(\boldsymbol{\theta}) \,\forall\, i \in \mathcal{A} \tag{10.7}$$

Finally, we incorporated these constraints in a min-norm QP problem to find the optimal velocities of the robots

$$\begin{aligned} \boldsymbol{u}_R^{*all} &= \underset{\boldsymbol{u}_R^{all}}{\arg\min} \left\| \boldsymbol{u}_R^{all} \right\|^2 \\ \text{subject to} \quad & A_i^H(\boldsymbol{\theta})\boldsymbol{u}_R^{all} \leq b_i^H(\boldsymbol{\theta}) \,\forall\, i \in \mathcal{A} \end{aligned} \tag{10.8}$$

Since the parameters $\boldsymbol{\theta}$ are now unknown, (10.8) cannot be solved in the form as stated. Instead of the true parameters, we have an estimate $\hat{\boldsymbol{\theta}}$ and an upper bound on estimation error η *i.e.* $\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|_2 \leq \eta$. Denoting the closed ball centered at $\hat{\boldsymbol{\theta}}$ with radius η as $B_\eta(\hat{\boldsymbol{\theta}})$, we can rewrite $\left\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|_2 \leq \eta$ as $\boldsymbol{\theta} \in B_\eta(\hat{\boldsymbol{\theta}})$. Thus, if we can solve (10.8) for all $\boldsymbol{\theta}^\dagger \in B_\eta(\hat{\boldsymbol{\theta}})$, we are guaranteed that the returned velocities will ensure defense of the protected zone because by our assumption, the true parameters $\boldsymbol{\theta} \in B_\eta(\hat{\boldsymbol{\theta}})$. Thus, the robust version of (10.8) becomes

$$\begin{aligned} \boldsymbol{u}_R^{*all} &= \underset{\boldsymbol{u}_R^{all}}{\arg\min} \left\| \boldsymbol{u}_R^{all} \right\|^2 \\ \text{subject to} \quad & A_i^H(\boldsymbol{\theta}^\dagger)\boldsymbol{u}_R^{all} \leq b_i^H(\boldsymbol{\theta}^\dagger) \quad \forall\, \boldsymbol{\theta}^\dagger \in B_\eta(\hat{\boldsymbol{\theta}}) \text{ and } \forall\, i \in \mathcal{A} \end{aligned} \tag{10.9}$$

In the form as stated, (10.9) is a semi-infinite program because the inner constraint requirement *i.e.* $\boldsymbol{\theta}^\dagger \in B_\eta(\hat{\boldsymbol{\theta}})$ is infinite dimensional. Hence, the total number of constraints in (10.9) becomes $M \times \infty = \infty$. Hence, as stated, this cannot be solved in a computationally tractable way. In the next section, we demonstrate how we can exploit the structure of $A_i^H(\boldsymbol{\theta}^\dagger)$ and $b_i^H(\boldsymbol{\theta}^\dagger)$ to convert this to a tractable finite-dimensional convex optimization problem.

10.3 Robust Control Formulation

There are two steps we follow to convert (10.9) to a tractable convex problem. Step 1 is constraint normalization while step 2 invokes duality theory to rewrite the normalized constraint as a finite-dimensional LMI constraint. We can obtain the same LMI as obtained in step 2 using s-procedure. This approach is also presented. Finally, in step 3, we combine all LMI constraints to pose the robust tractable version of (9.23).

10.3.1 Step 1: constraint normalization

Here, we exploit our knowledge that A_i^H and b_i^H are respectively affine and quadratic in $\boldsymbol{\theta}^\dagger$. Thus, we can write them explicitly as

$$\begin{aligned} A_i^H &= \boldsymbol{\theta}^{\dagger T}\tilde{C}_i + \tilde{\boldsymbol{d}}_i^T \\ b_i^H &= \boldsymbol{\theta}^{\dagger T}\tilde{H}_i\boldsymbol{\theta}^\dagger + \tilde{\boldsymbol{f}}_i^T\boldsymbol{\theta}^\dagger + \tilde{g}_i. \end{aligned} \tag{10.10}$$

Here $\tilde{H}_i$ is the hessian of b_i^H with respect to $\boldsymbol{\theta}^\dagger$, $\tilde{\boldsymbol{f}}_i$ is its jacobian with respect to $\boldsymbol{\theta}^\dagger$ and $\tilde{g}_i$ is residual term that does not depend on $\boldsymbol{\theta}^\dagger$. Similarly, $\tilde{C}_i$ is the jacobian of A_i^H with respect to $\boldsymbol{\theta}^\dagger$ and $\tilde{\boldsymbol{d}}_i$ is the residual term in A_i^H that does not depend on $\boldsymbol{\theta}^\dagger$. These terms can be computed by doing symbolic differentiation of (9.15) with respect to $\boldsymbol{\theta}^\dagger$. Substituting $A_i^H(\boldsymbol{\theta}^\dagger)$ and $b_i^H(\boldsymbol{\theta}^\dagger)$ from (10.10) in the constraint $A_i^H(\boldsymbol{\theta}^\dagger)\boldsymbol{u}_R^{all} \leq b_i^H(\boldsymbol{\theta}^\dagger)\ \forall\ \boldsymbol{\theta}^\dagger \in B_\eta(\hat{\boldsymbol{\theta}})$ from (10.9), we get,

$$\left(\boldsymbol{\theta}^{\dagger T}\tilde{C}_i + \tilde{\boldsymbol{d}}_i^T\right)\boldsymbol{u}_R^{all} \leq \boldsymbol{\theta}^{\dagger T}\tilde{H}_i\boldsymbol{\theta}^\dagger + \tilde{\boldsymbol{f}}_i^T\boldsymbol{\theta}^\dagger + \tilde{g}_i \quad \text{subject to } \left\|\boldsymbol{\theta}^\dagger - \hat{\boldsymbol{\theta}}\right\|_2 \leq \eta. \tag{10.11}$$

We normalize this constraint with a variable $\boldsymbol{z}$ by substituting $\boldsymbol{\theta}^\dagger$ as follows

$$\boldsymbol{\theta}^\dagger = \hat{\boldsymbol{\theta}} + \eta\boldsymbol{z}, \tag{10.12}$$

which gives,

$$\begin{aligned}\left(\boldsymbol{z}^T C_i + \boldsymbol{d}_i^T\right)\boldsymbol{u}_R^{all} \leq \boldsymbol{z}^T H_i\boldsymbol{z} + \boldsymbol{f}_i^T\boldsymbol{z} + g_i \quad &\text{subject to } \|\boldsymbol{z}\|_2 \leq 1, \text{ where,}\\ C_i &:= \eta\tilde{C}_i\\ \boldsymbol{d}_i^T &:= \hat{\boldsymbol{\theta}}^T\tilde{C}_i + \tilde{\boldsymbol{d}}_i\\ H_i &:= \eta^2\tilde{H}_i\\ \boldsymbol{f}_i &:= \eta\tilde{\boldsymbol{f}}_i + 2\eta\tilde{H}_i\hat{\boldsymbol{\theta}}\\ g_i &:= \hat{\boldsymbol{\theta}}^T\tilde{H}_i\hat{\boldsymbol{\theta}} + \tilde{\boldsymbol{f}}_i^T\hat{\boldsymbol{\theta}} + \tilde{g}_i.\end{aligned} \tag{10.13}$$

With this substitution, problem (10.9) can be rewritten as

$$\begin{aligned}\boldsymbol{u}_R^{*all} = \underset{\boldsymbol{u}_R^{all}}{\arg\min}\quad & \left\|\boldsymbol{u}_R^{all}\right\|^2\\ \text{subject to}\quad & \left(\boldsymbol{z}^T C_i + \boldsymbol{d}_i^T\right)\boldsymbol{u}_R^{all} \leq \boldsymbol{z}^T H_i\boldsymbol{z} + \boldsymbol{f}_i^T\boldsymbol{z} + g_i \text{ subject to } \|\boldsymbol{z}\|_2 \leq 1\ \forall\ i \in \mathcal{A}\end{aligned} \tag{10.14}$$

In the next section, we show how to pose this optimization problem as a semi-definite program (SDP). To keep the notation light, we will omit the subscript i and simply augment additional constraints in the final problem section 10.3.4, one for defense against each agent in the flock.

10.3.2 Step 2: SDP formulation using duality theory

Consider the following QP

$$\begin{aligned}\boldsymbol{u}_R^{*all} = \underset{\boldsymbol{u}_R^{all}}{\arg\min}\quad & \left\|\boldsymbol{u}_R^{all}\right\|^2\\ \text{subject to}\quad & \left(\boldsymbol{z}^T C + \boldsymbol{d}^T\right)\boldsymbol{u}_R^{all} \leq \boldsymbol{z}^T H\boldsymbol{z} + \boldsymbol{f}^T\boldsymbol{z} + g \text{ subject to } \|\boldsymbol{z}\|_2 \leq 1.\end{aligned} \tag{10.15}$$

In this QP, the constraint effectively consists of an infinite number of inequality constraints corresponding to each $\boldsymbol{z}$ in the unit ball. These infinite constraints make the overall problem a convex semi-infinite optimization problem. One approach to solve this problem is by minimizing the objective while requiring the satisfaction of the most difficult constraint among the infinite constraints. This conservative problem can be posed as follows

$$\begin{aligned} \boldsymbol{u}_R^{*all} = \underset{\boldsymbol{u}_R^{all}}{\arg\min} \quad & \left\|\boldsymbol{u}_R^{all}\right\|^2 \\ \text{subject to} \quad & 0 \leq \inf_{\|\boldsymbol{z}\|_2 \leq 1} \underbrace{\boldsymbol{z}^T H \boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T \boldsymbol{z} + g - \boldsymbol{d}^T \boldsymbol{u}_R^{all}}_{f_0(\boldsymbol{z})} \end{aligned} \tag{10.16}$$

By requiring the infimum of $f_0(\boldsymbol{z})$ over the unit ball to be non-negative, we have converted an infinite number of constraints into a single constraint. This infimization results in another optimization problem within the problem of finding $\boldsymbol{u}_R^{*all}$. Let's consider this inner problem

$$\begin{aligned} \text{inf.} \quad & \boldsymbol{z}^T H \boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T \boldsymbol{z} + g - \boldsymbol{d}^T \boldsymbol{u}_R^{all} \\ \text{subject to} \quad & \|\boldsymbol{z}\|_2 \leq 1. \end{aligned} \tag{10.17}$$

Note that (10.17) is equivalent to the following (possibly non-convex) QCQP

$$\begin{aligned} \text{inf.} \quad & \boldsymbol{z}^T H \boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T \boldsymbol{z} + g - \boldsymbol{d}^T \boldsymbol{u}_R^{all} \\ \text{subject to} \quad & \boldsymbol{z}^T \boldsymbol{z} \leq 1. \end{aligned} \tag{10.18}$$

Let $\boldsymbol{z}^*(\boldsymbol{u}_R^{all})$ be the optimizer of (10.18) (assume it exists). We require $\boldsymbol{z}^*(\boldsymbol{u}_R^{all})$ to satisfy $f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all})) \geq 0$ per the robustness requirement in the constraint of (10.16). Thus, if we can find any lower bound for $f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all}))$ and constrain that lower bound to be non-negative, then we will ensure $f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all})) \geq 0$ as required by the robustness constraint. We use weak duality to obtain a lower bound for $f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all}))$. The Lagrangian for (10.18) is

$$\begin{aligned} \mathcal{L}(\boldsymbol{z}, \lambda) &:= \boldsymbol{z}^T H \boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T \boldsymbol{z} + g - \boldsymbol{d}^T \boldsymbol{u}_R^{all} + \lambda(\boldsymbol{z}^T \boldsymbol{z} - 1) \\ &= \boldsymbol{z}^T (H + \lambda I) \boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T \boldsymbol{z} + g - \boldsymbol{d}^T \boldsymbol{u}_R^{all} - \lambda. \end{aligned} \tag{10.19}$$

Define the Lagrange dual function $g(\lambda)$ as

$$g(\lambda) := \inf_{\boldsymbol{z} \in \mathbb{R}^p} \mathcal{L}(\boldsymbol{z}, \lambda). \tag{10.20}$$

If $\lambda \geq 0$, then from the lower bound property[1], we have that $g(\lambda) \leq f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all}))$ *i.e.* if $\lambda \geq 0$, then $g(\lambda)$ is a candidate lower bound for $f_0(\boldsymbol{z}^*(\boldsymbol{u}_R^{all}))$. Thus, the robustness constraint in (10.16) can be posed by requiring $g(\lambda) \geq 0$. To impose this constraint, we

[1] https://web.stanford.edu/class/ee364a/lectures/duality.pdf

derive an expression for $g(\lambda)$ in (10.20) by using the Lagrangian defined in (10.19).

$$g(\lambda) := \inf_{\boldsymbol{z}\in\mathbb{R}^p} \boldsymbol{z}^T(H+\lambda I)\boldsymbol{z} + (\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T\boldsymbol{z} + g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda$$
$$= \begin{cases} -\infty & \text{if } H+\lambda I \not\succcurlyeq 0 \\ -\infty & \text{if } H+\lambda I \succcurlyeq 0 \text{ and } \boldsymbol{f} - C\boldsymbol{u}_R^{all} \notin \mathcal{R}(H+\lambda I) \\ -\frac{1}{4}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T(H+\lambda I)^\dagger(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) + g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda & \text{otherwise} \end{cases} \tag{10.21}$$

Thus, to ensure that $g(\lambda) \geq 0$ when $\lambda \geq 0$, we require

1. $-\frac{1}{4}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T(H+\lambda I)^\dagger(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) + g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \geq 0$
2. $H + \lambda I \succcurlyeq 0$
3. $\boldsymbol{f} - C\boldsymbol{u}_R^{all} \in \mathcal{R}(H+\lambda I) \iff \Big(I - (H+\lambda I)(H+\lambda I)^\dagger\Big)(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) = \boldsymbol{0}$.

From the Schur Complement theorem[2], these three conditions are equivalent to

$$\begin{bmatrix} H+\lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \tag{10.22}$$

This is a linear matrix inequality type constraint. In conjunction with $\lambda \geq 0$ and (10.22), problem (10.16) becomes

$$\begin{aligned} \boldsymbol{u}_R^{*all}, \lambda^* = \arg\min_{\boldsymbol{u}_R^{all}, \lambda} & \left\|\boldsymbol{u}_R^{all}\right\|^2 \\ \text{subject to} & \begin{bmatrix} H+\lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \\ & \lambda \geq 0. \end{aligned} \tag{10.23}$$

Using the epigraph trick and Schur complement theorem, we can write (10.23) as

$$\begin{aligned} \boldsymbol{u}_R^{*all}, \lambda^*, t^* = \arg\min_{\boldsymbol{u}_R^{all}, \lambda, t} & \; t \\ \text{subject to} & \begin{bmatrix} H+\lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \\ & \lambda \geq 0 \\ & \begin{bmatrix} I & \boldsymbol{u}_R^{all} \\ \boldsymbol{u}_R^{allT} & t \end{bmatrix} \succcurlyeq 0. \end{aligned} \tag{10.24}$$

[2] https://chrisyeh96.github.io/2021/05/19/schur-complement.html

Problem (10.24) is a semi-definite program (SDP) which produces a conservative solution to problem (10.15). Compared to (10.15), the constraints in this SDP are finite-dimensional and the overall problem is still convex. Hence, we can solve this problem using off-the-shelf convex optimization packages. Next, we show that we can arrive at the same SDP from problem (10.15) following an alternative approach by invoking the s-procedure.

10.3.3 Step 2: SDP formulation using s-procedure

Let us recall problem (10.15) again

$$\begin{aligned} \boldsymbol{u}_R^{*all} = \arg\min_{\boldsymbol{u}_R^{all}} \quad & \left\|\boldsymbol{u}_R^{all}\right\|^2 \\ \text{subject to} \quad & \left(\boldsymbol{z}^T C + \boldsymbol{d}^T\right)\boldsymbol{u}_R^{all} \le \boldsymbol{z}^T H \boldsymbol{z} + \boldsymbol{f}^T\boldsymbol{z} + g \quad \text{subject to } \|\boldsymbol{z}\|_2 \le 1 \end{aligned} \tag{10.25}$$

We write the $\|\boldsymbol{z}\|_2 \le 1$ constraint as follows:

$$\begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix}^T \begin{bmatrix} -I & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix} \ge 0 \iff \tilde{\boldsymbol{z}}^T P \tilde{\boldsymbol{z}} \ge 0, \tag{10.26}$$

where we have defined

$$\begin{aligned} \tilde{\boldsymbol{z}} &:= (\boldsymbol{z}^T, 1)^T \\ P &:= \begin{bmatrix} -I & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{10.27}$$

Similarly, we write $\left(\boldsymbol{z}^T C + \boldsymbol{d}^T\right)\boldsymbol{u}_R^{all} \le \boldsymbol{z}^T H \boldsymbol{z} + \boldsymbol{f}^T\boldsymbol{z} + g$ as

$$\begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix}^T \begin{bmatrix} H & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} \end{bmatrix} \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix} \ge 0 \iff \tilde{\boldsymbol{z}}^T Q \tilde{\boldsymbol{z}} \ge 0, \tag{10.28}$$

where we have defined

$$Q := \begin{bmatrix} H & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} \end{bmatrix} \tag{10.29}$$

Thus we want $\tilde{\boldsymbol{z}}^T P \tilde{\boldsymbol{z}} \ge 0 \implies \tilde{\boldsymbol{z}}^T Q \tilde{\boldsymbol{z}} \ge 0$. Recall the statement of s-lemma[3].

Lemma 1. *Let M and N be two symmetric matrices such that there exists a $\boldsymbol{u}^0$ satisfying $(\boldsymbol{u}^0)^T M \boldsymbol{u}^0 > 0$. Then the implication $\boldsymbol{u}^T M \boldsymbol{u} \ge 0 \implies \boldsymbol{u}^T N \boldsymbol{u}$ holds true if and only if there exists $\lambda \ge 0$ such that $N \succcurlyeq \lambda M$.*

[3] https://web.stanford.edu/class/msande314/lecture15.pdf

Thus, by using $M \coloneqq P$ and $N \coloneqq Q$ in the s-lemma, $\tilde{\boldsymbol{z}}^T P \tilde{\boldsymbol{z}} \geq 0 \implies \tilde{\boldsymbol{z}}^T Q \tilde{\boldsymbol{z}} \geq 0$ occurs when $\exists \lambda \geq 0$ for which $Q \succcurlyeq \lambda P$ *i.e.*,

$$\begin{bmatrix} H & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} \end{bmatrix} \succcurlyeq \lambda \begin{bmatrix} -I & 0 \\ 0 & 1 \end{bmatrix} \iff \begin{bmatrix} H + \lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \tag{10.30}$$

Thus, problem (10.15) becomes

$$\begin{aligned} \boldsymbol{u}_R^{*all}, \lambda^* = \underset{\boldsymbol{u}_R^{all}, \lambda}{\arg\min} & \left\|\boldsymbol{u}_R^{all}\right\|^2 \\ \text{subject to} & \begin{bmatrix} H + \lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \\ & \lambda \geq 0. \end{aligned} \tag{10.31}$$

Using the epigraph trick and Schur complement theorem, we can write this as

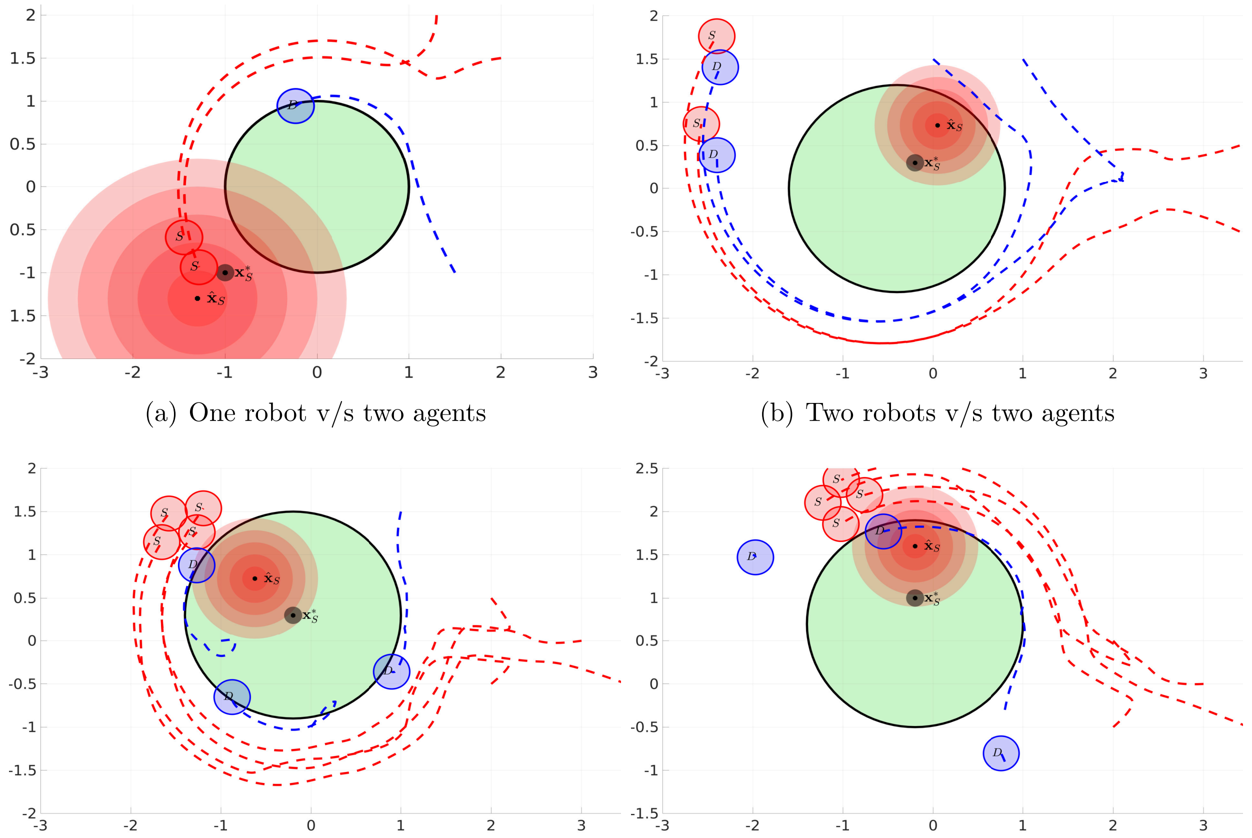
$$\begin{aligned} \boldsymbol{u}_R^{*all}, \lambda^*, t^* = \underset{\boldsymbol{u}_R^{all}, \lambda, t}{\arg\min} & \quad t \\ \text{subject to} & \begin{bmatrix} H + \lambda I & \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f} - C\boldsymbol{u}_R^{all})^T & g - \boldsymbol{d}^T\boldsymbol{u}_R^{all} - \lambda \end{bmatrix} \succcurlyeq 0 \\ & \lambda \geq 0 \\ & \begin{bmatrix} I & \boldsymbol{u}_R^{all} \\ \boldsymbol{u}_R^{allT} & t \end{bmatrix} \succcurlyeq 0. \end{aligned} \tag{10.32}$$

Note that this SDP is identical to (10.24).

10.3.4 Final Robust SDP

While deriving the LMI constraint in (10.22) from the original problem (10.14), we omitted the index i, thereby making the implicit assumption that we are finding velocities for all robots for defense against just one agent. To extend this to accommodating constraints for all agents in the flock, we simply augment additional constraints in (10.32) corresponding to each agent in $\mathcal{A}$. This gives the final robust version of (9.23), *i.e.*,

$$\begin{aligned} \boldsymbol{u}_R^{*all}, \{\lambda_i^*\}_{i\in\mathcal{A}}, t^* = \underset{\boldsymbol{u}_R^{all}, \{\lambda_i\}_{i\in\mathcal{A}}, t}{\arg\min} & \quad t \\ \text{subject to} & \begin{bmatrix} H_i + \lambda_i I & \frac{1}{2}(\boldsymbol{f}_i - C_i\boldsymbol{u}_R^{all}) \\ \frac{1}{2}(\boldsymbol{f}_i - C_i\boldsymbol{u}_R^{all})^T & g_i - \boldsymbol{d}_i^T\boldsymbol{u}_R^{all} - \lambda_i \end{bmatrix} \succcurlyeq 0 \;\; \forall \; i \in \mathcal{A} \\ & \lambda_i \geq 0 \;\; \forall \; i \in \mathcal{A} \\ & \begin{bmatrix} I & \boldsymbol{u}_R^{all} \\ \boldsymbol{u}_R^{allT} & t \end{bmatrix} \succcurlyeq 0 \end{aligned} \tag{10.33}$$



(a) One robot v/s two agents

(b) Two robots v/s two agents

(c) Three robots v/s four agents

(d) Three robots v/s four agents

Figure 10.1: Demonstration of our results showing how to prevent agents (red) from breaching the protect zone (green) in the presence of uncertainty in the goal of the agents. The true goal is highlighted in black, the estimate of the goal is highlighted with a black dot and the red discs around it represent the uncertainty.

10.4 Results

We validate our proposed robust centralized controller on several test cases. To show repeatability, we vary the numbers of flock agents and robots, the initial positions of these agents and the magnitude of uncertainty in the location of the goal of the flock agents. The robots do not know the true goal, they just know an estimate of the location of the flock's goal and an upper bound on the estimation error. The initial positions $\boldsymbol{x}_{A_i}(0)$ of all agents are chosen such that they are all close to each other. This is done to ensure that the agents have enough time to stabilize as a flock before interacting with the robots. The initial positions $\boldsymbol{x}_R^{all}(0)$ of the robots are chosen randomly within the area of operation.

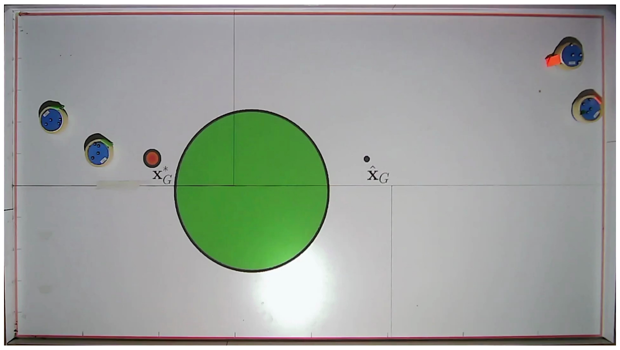
10.4.1 Simulations

Figure 10.1 shows four simulation results. The robots' velocities are calculated using (10.33). The values of the parameters are $k_G = 0.2, k_A = 0.7, k_R = 0.1, D_s = 0.4, R_p = 1.2, \alpha = 5, \beta = 6$. The true goal of the flock is $\boldsymbol{x}_S$ and highlighted in black whereas the estimate is $\hat{\boldsymbol{x}}_S$ and the outermost red disc around it represents the magnitude of uncertainty. It can be noticed from the figure that in all four scenarios, the robots are able to successfully intercept all agents and prevent them from entering the protected zone despite uncertainty in the goal of the flock agents.

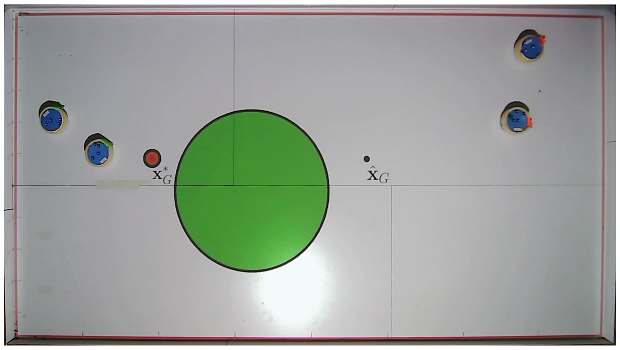
10.4.2 Experiments

Next, we conducted experiments in the multirobot arena. These experiments are similar to the ones we conducted in chapter 9 except that now the dog robots do not know the true goals of the sheep. To demonstrate the superiority of our proposed robust controller, we show how the results of a certainty equivalent centralized controller that does not incorporate estimation error, rather it treats an estimate of the goal as the true goal. Figure 10.2 shows these results. Here there are two dog robots trying to defend the protected zone against two sheep agents. Both the true goal and the estimate of the goal are shown. The dog robots do not know the true goal, so they treat the estimate as the true goal and use the controller developed in (10.8). As can be seen in the snapshots, the dog robots fail to act quickly because of which a breach occurs.

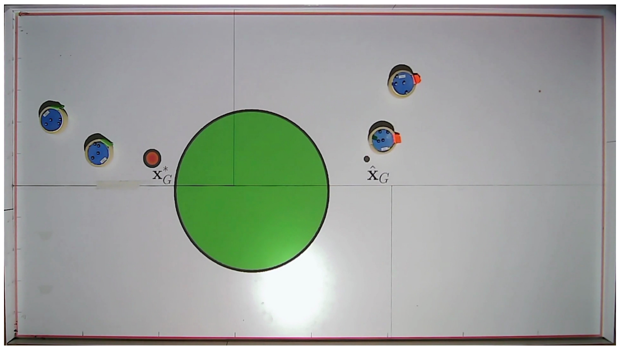
Next, we show how our proposed robust controller that explicitly incorporates an upper bound on estimation error in addition to the estimate of the goal prevents the breach from occurring. Figure 10.3 shows these results. Here the yellow disc represents all possible candidates of the goal and the size of this disc represents the uncertainty. As can be seen from the snapshots, the dog robots are able to defend the protected zone by acting quickly. Figure 10.5 shows the case of one robot defending against two flock agents. As can be seen in the snapshots, none of the flock agents breach the protected zone.



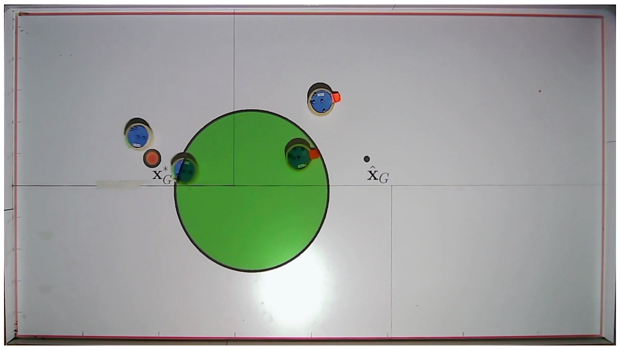
(a) $t = 0.0s$



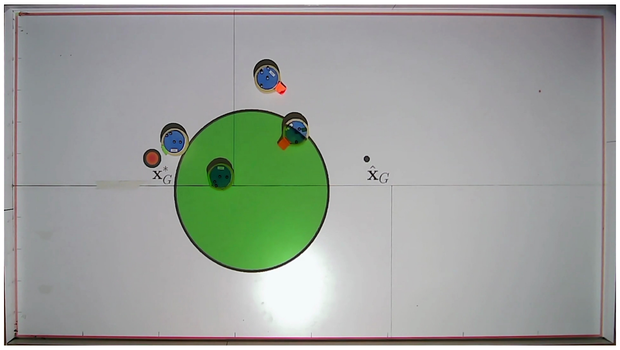
(b) $t = 2.0s$



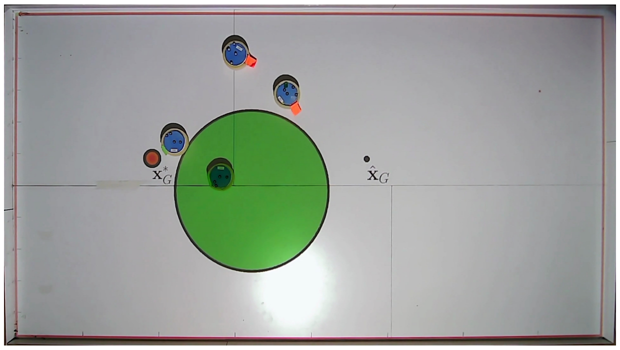
(c) $t = 10.0s$



(d) $t = 50.0s$

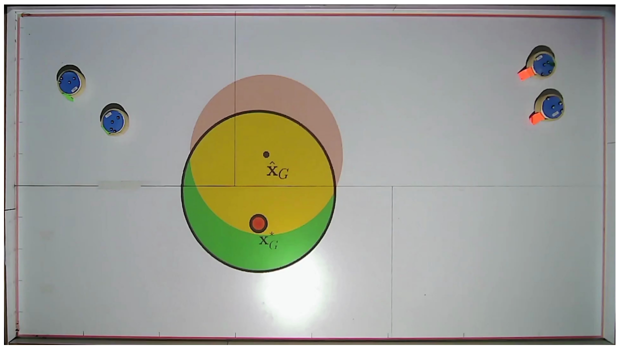


(e) $t = 50.0s$

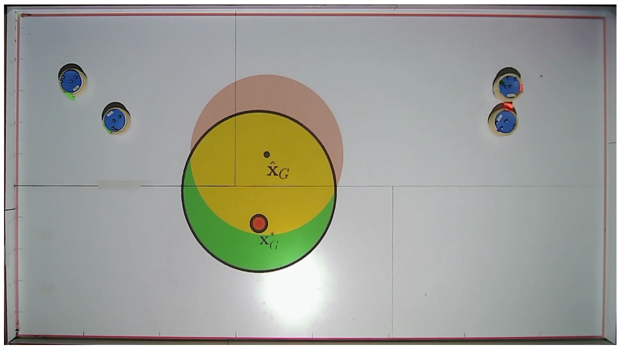


(f) $t = 50.0s$

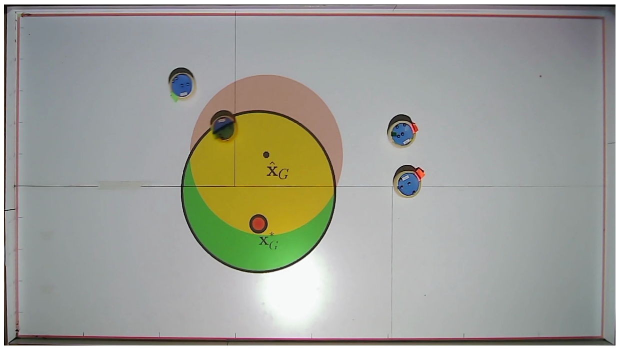
Figure 10.2: In these images, we can see two sheep agents going towards their goal. The dog robots only have an estimate of the goal. The dog robots use the (10.8) with the estimate. Because of this incorrect assumption, the sheep robots are able to breach the protected zone as the dog robots act too late.



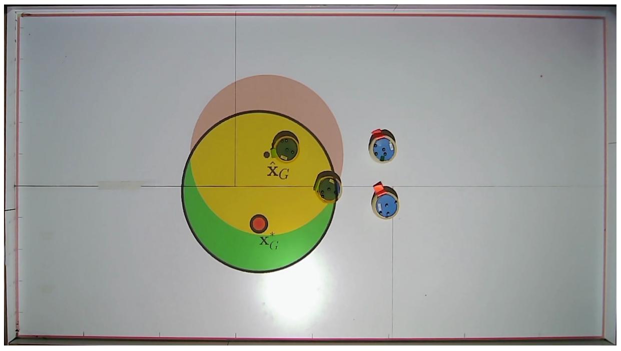
(a) $t = 0.0s$



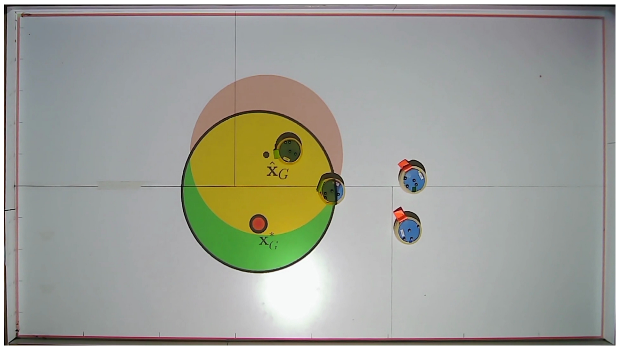
(b) $t = 2.0s$



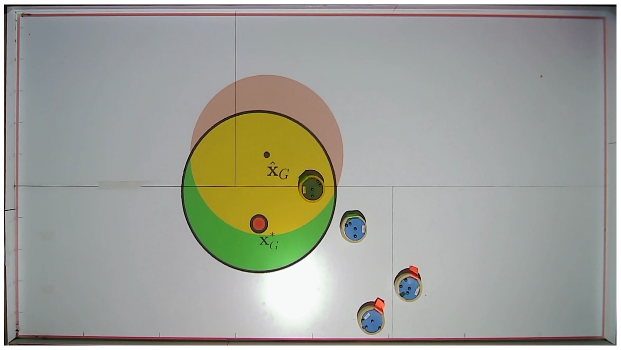
(c) $t = 10.0s$



(d) $t = 50.0s$

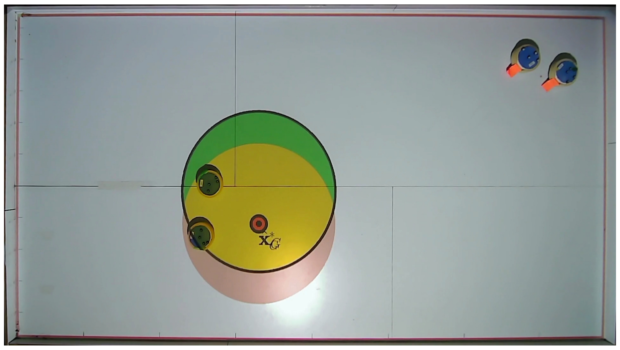


(e) $t = 50.0s$

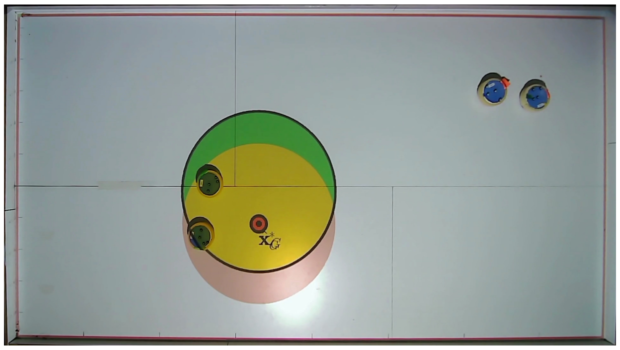


(f) $t = 50.0s$

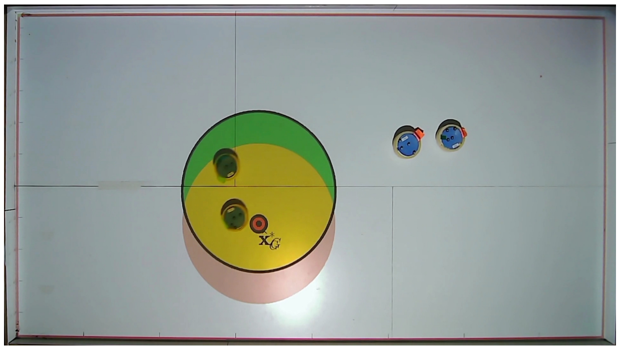
Figure 10.3: In these images, we can see two sheep agents going towards their goal. The dog robots have both an estimate of the goal and an upper bound on error shown with the yellow disc. As the sheep move towards the goal, the dog robots are able to defend the protected zone despite uncertainty in the goal of the sheep.



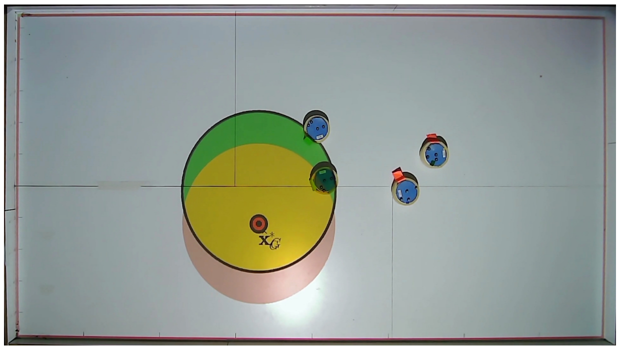
(a) $t = 0.0s$



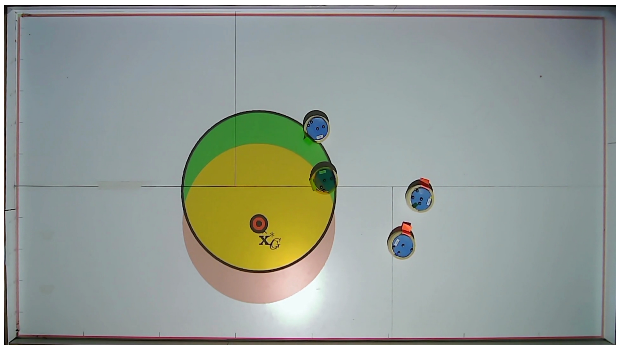
(b) $t = 2.0s$



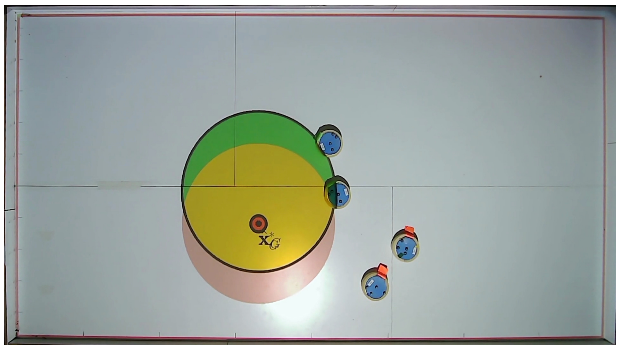
(c) $t = 10.0s$



(d) $t = 50.0s$

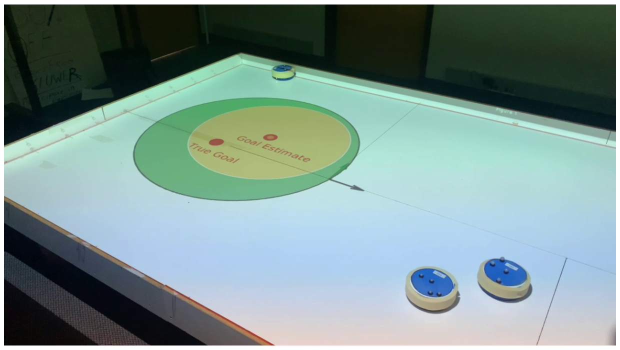


(e) $t = 50.0s$

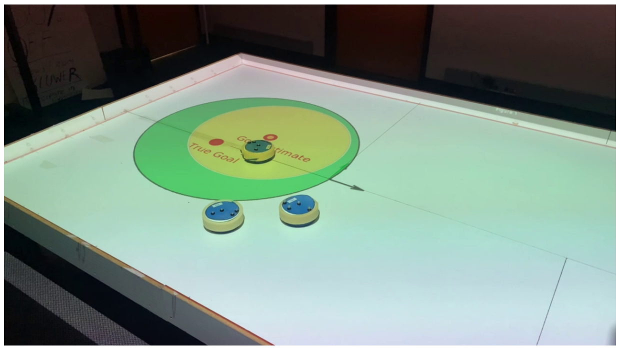


(f) $t = 50.0s$

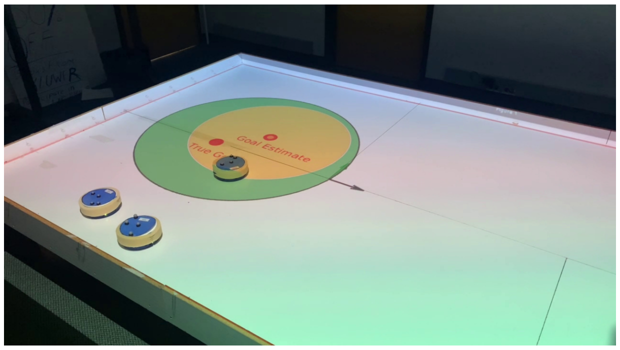
Figure 10.4: In these images, we can see two sheep agents going towards their goal. The dog robots have both an estimate of the goal and an upper bound on error shown with the yellow disc. That the estimate of the goal is the same as the true goal is unbeknownst to the dog robots. Thus, they use the robust controller which ensures successful defense.



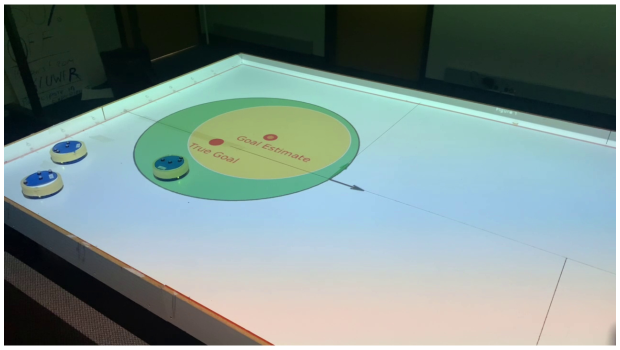
(a) $t = 0.0s$



(b) $t = 2.0s$



(c) $t = 10.0s$



(d) $t = 50.0s$

Figure 10.5: In these images, we can see two sheep agents going towards their goal. The dog robot only has an estimate of the goal and an upper bound on error shown with the red disc. As the sheep move towards the goal, the dog robot is able to defend the protected zone despite uncertainty in the goal of the sheep.

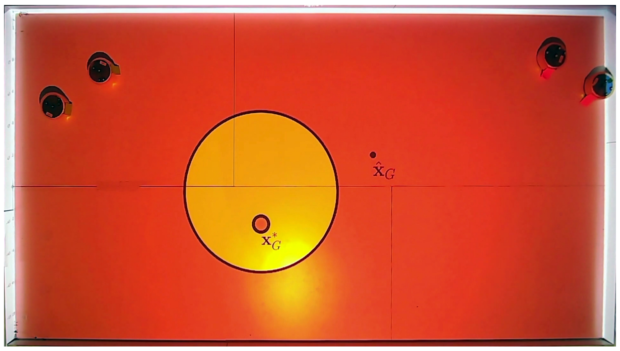
10.5 Online Parameter Adaptation And Control

In our proposed robust control framework, we assumed that the control engineer controlling the robots has an a-priori known nominal estimate of the sheep agents' goal and an a-priori known upper bound on the error in that estimate. The question then must be asked, can we learn a nominal estimate of the goal and its associated error online based on the observed motions of the sheep agents and the structure of the sheep agents' underlying dynamic model. We addressed this model-based online parameter estimation question in chapters 4-6 of this thesis. To wit, several parameter estimation algorithms were developed to learn the desired goals of individual agents in a multiagent system.

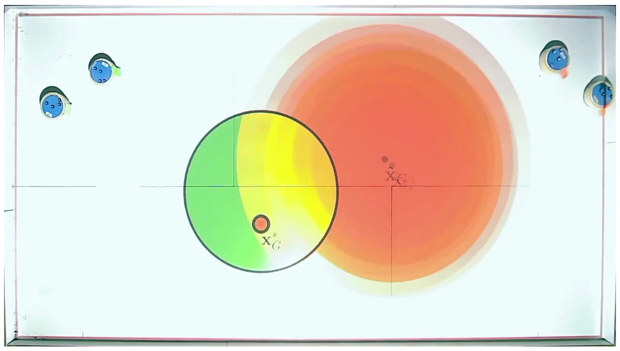
Thus, we can now integrate the real-time estimates of goal and associated errors from our parameter learning algorithms with the developed robust controller to develop an adaptive robust controller. This is beneficial because adaptation of parameters to observed measurements reduces conservatism in the robust controller especially if the initial user specified uncertainty is high. To demonstrate this, we conducted several experiments in which we used a Kalman filter to learn the true goal of all the sheep and used the 15σ uncertainty as the upper bound on the estimation error required by the robust controller. Thus, instead of providing the constant uncertainty to the algorithm in (10.33), we provide it with time-varying uncertainty. This is shown in Fig. 10.6 and 10.7 for the two dogs v/s two sheep case. The red discs represent various uncertainty levels. As can be seen in consecutive snapshots, the size of the red discs is decreasing which denotes decreasing uncertainty in the goal. As can be noticed in the snapshots, the sheep agents never breach the protected zone, thus demonstrating the repeatability and robustness of our proposed control algorithm. We tried the same approach with a handcrafted estimator that provably monotonically converges to the true goal and its uncertainty monotonically converges to zero. This is shown in Fig 10.8 for the one dog v/s two sheep case. Figs. 10.9 and 10.10 demonstrate the same for the two dogs v/s two sheep cases.

10.6 Conclusions

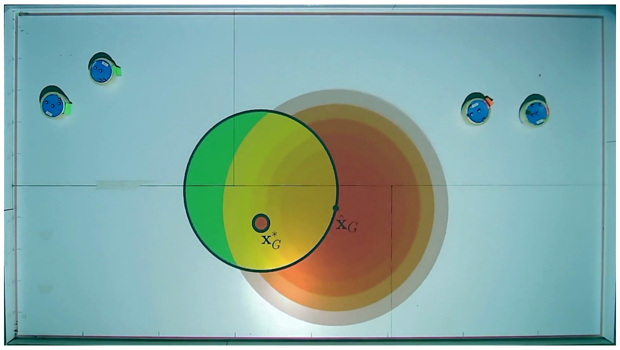
In this chapter, we extended our results in chapter 9 by endowing robustness to the centralized controllers. Previously, we assumed that the observer's robots know the true latent parameters of the flock agents a-priori. In this chapter, we relaxed this assumption and provided the robots with only an estimate of the parameters and an upper bound on the estimation error. Using these estimates, we first developed a convex semi-infinite program that can generate correct velocities for the robots for each instantiation of the parameters within the uncertainty. Since this is computationally intractable, we used duality theory to



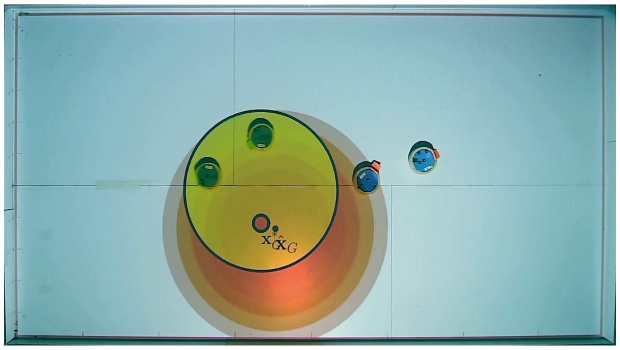
(a) $t = 0.0s$



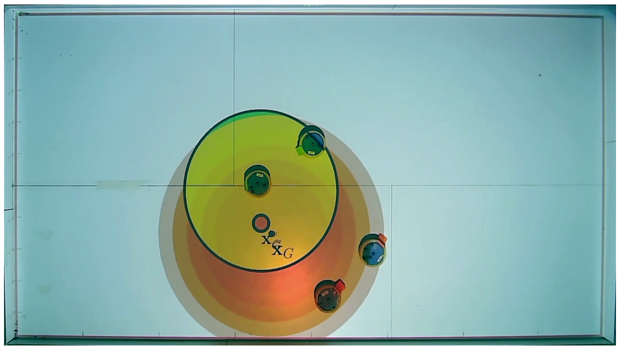
(b) $t = 2.0s$



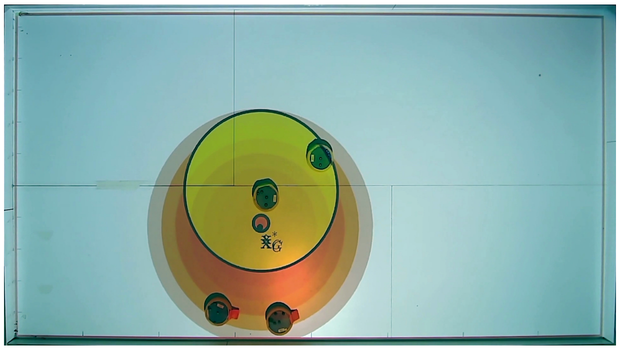
(c) $t = 10.0s$



(d) $t = 50.0s$



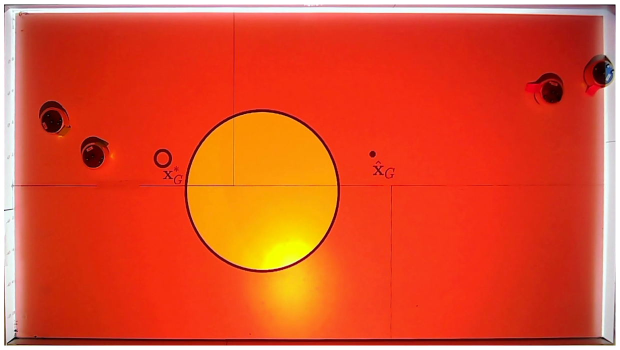
(e) $t = 70.0s$



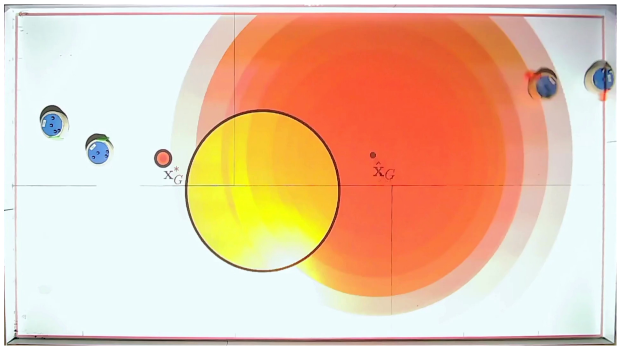
(f) $t = 90.0s$

Figure 10.6: In these images, we can see two dog robots defending the protected zone from two sheep agents. Here an initial nominal estimate of the goal and the uncertainty is updated as a function of time. The outermost red disc represents the 15σ uncertainty which is used as η in the robust controller. Thus, they use the robust controller which ensures successful defense.

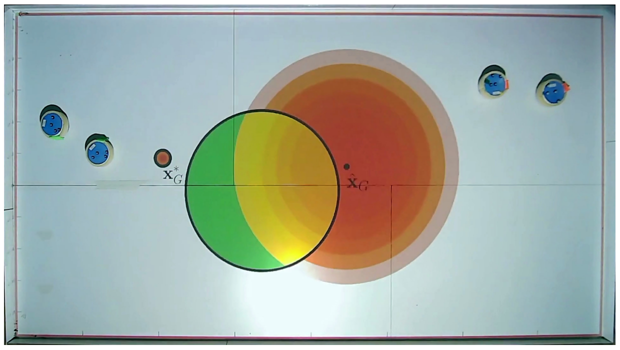
re-pose this optimization problem as a semi-definite program which can be solved reactively online. We showed simulation results as well as experimental results demonstrating that our proposed robust control strategy indeed defends the protected zone from flock agents.



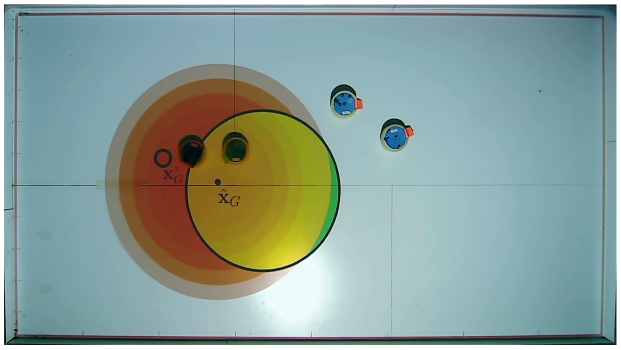
(a) $t = 0.0s$



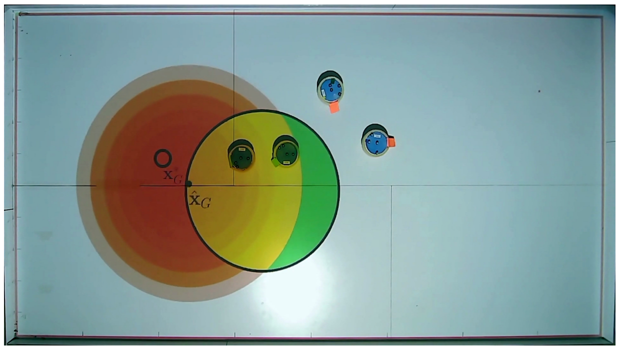
(b) $t = 2.0s$



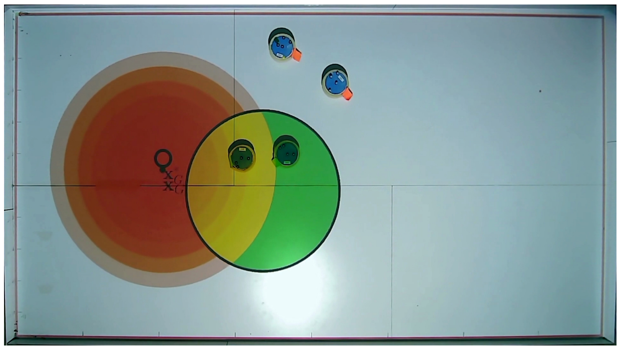
(c) $t = 10.0s$



(d) $t = 50.0s$



(e) $t = 70.0s$



(f) $t = 90.0s$

Figure 10.7: In these images, we can see two dog robots defending the protected zone from two sheep agents. Here an initial nominal estimate of the goal and the uncertainty is updated as a function of time. The outermost red disc represents the 15σ uncertainty which is used as η in the robust controller. Thus, they use the robust controller which ensures successful defense.

Additionally, we also augmented this with our system identification algorithms to allow the robust controller to adapt its uncertainty to the measurements of the agents' motions as observed in real-time.

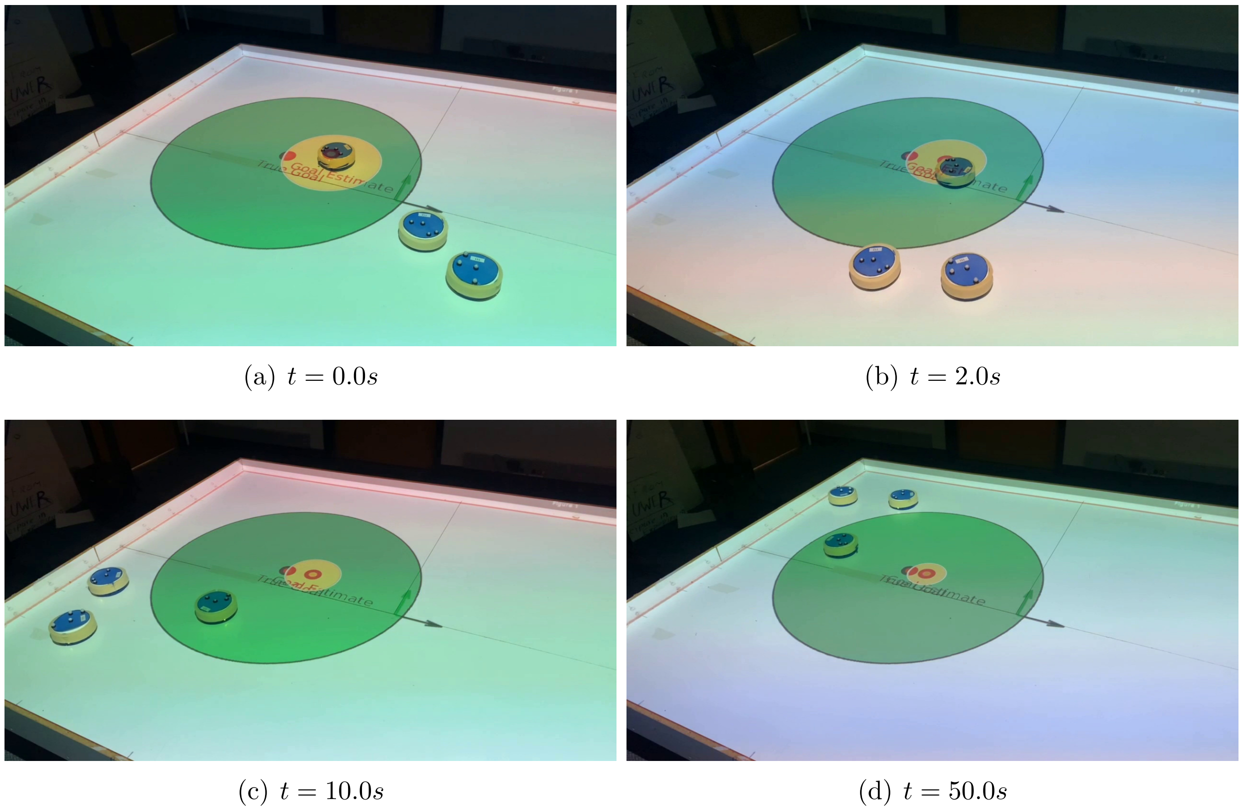
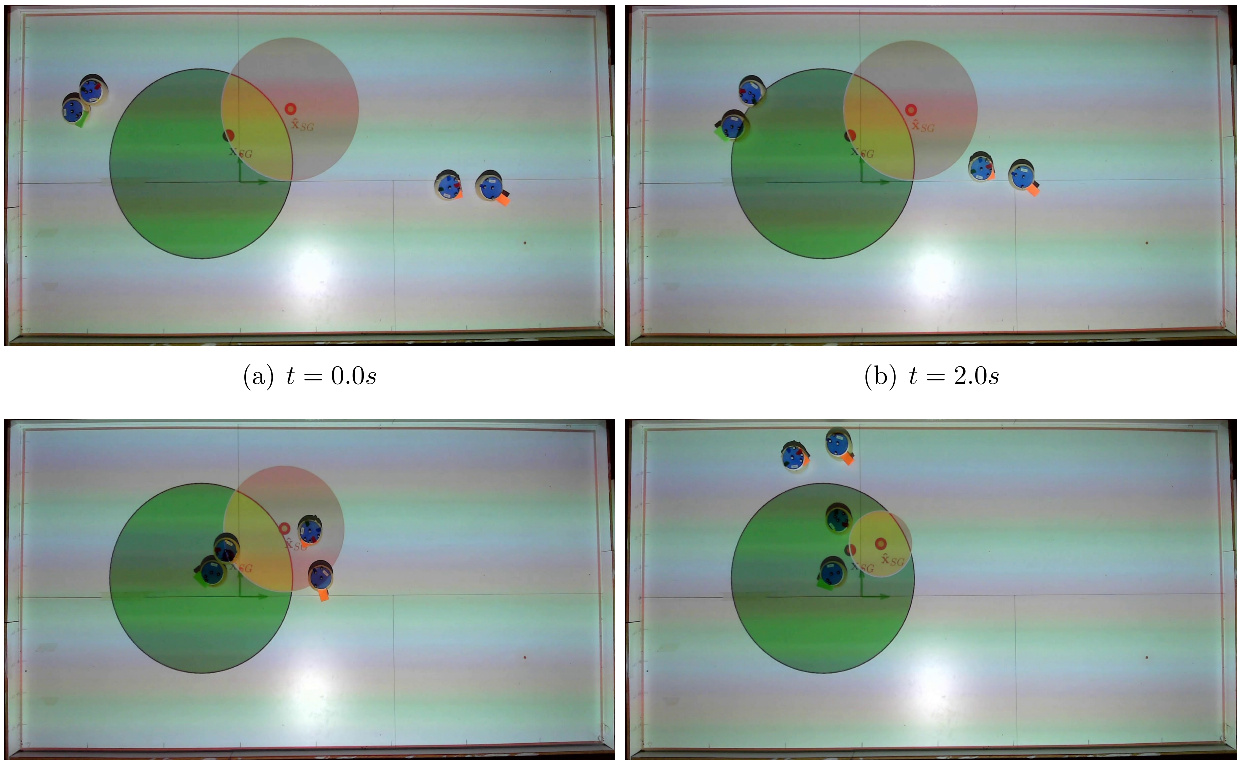

(a) $t = 0.0s$ (b) $t = 2.0s$ (c) $t = 10.0s$ (d) $t = 50.0s$

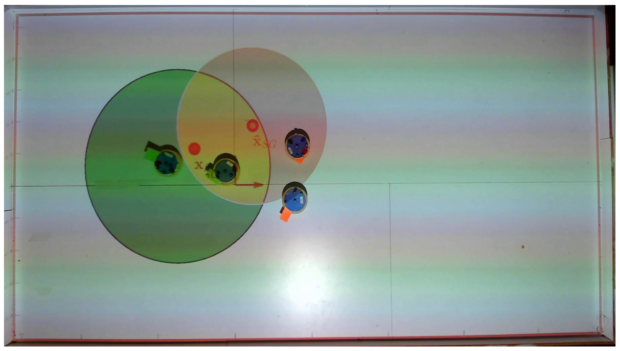
Figure 10.8: In these images, we can see two sheep agents (orange tails) going towards their goal. The dog robots (green tails) do not know the true goal, they only have an estimate of the goal and an upper bound on error shown with the red disc. The magnitude of uncertainty is decreasing with time because the dog robots are internally using an identification algorithm to learn the goal of the sheep. As the sheep move towards the goal, the dog robots are able to defend the protected zone despite uncertainty in the goal of the sheep.



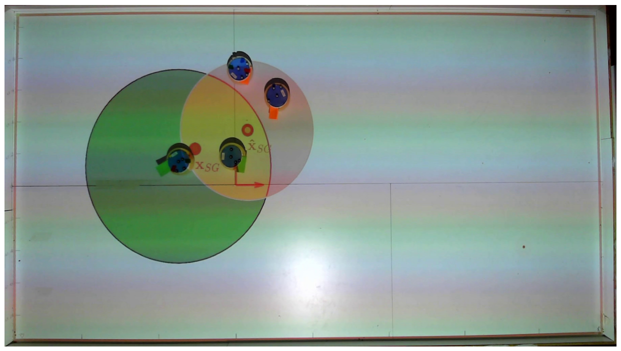
(a) $t = 0.0s$ (b) $t = 2.0s$

(c) $t = 10.0s$ (d) $t = 50.0s$

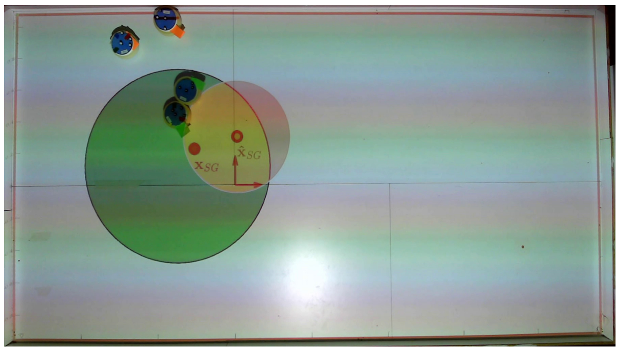
Figure 10.9: In these images, we can see two sheep agents (orange tails) going towards their goal. The dog robots (green tails) do not know the true goal, they only have an estimate of the goal and an upper bound on error shown with the red disc. As the sheep move towards the goal, the dog robots are able to defend the protected zone despite uncertainty in the goal of the sheep. This uncertainty is decreasing with time because the dog robots use an online identification algorithm that is learning this goal.



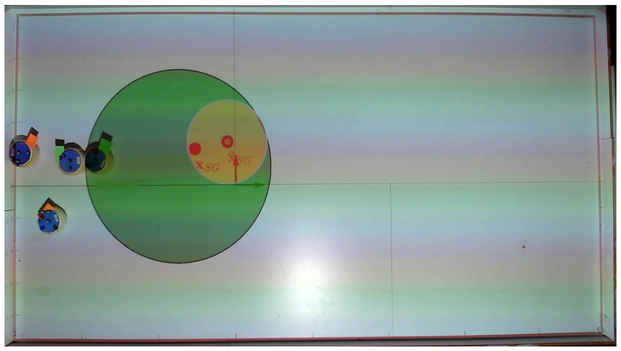
(a) $t = 0.0s$



(b) $t = 2.0s$



(c) $t = 10.0s$



(d) $t = 50.0s$

Figure 10.10: In these images, we can see two sheep agents (orange tails) going towards their goal. The dog robots (green tails) do not know the true goal, they only have an estimate of the goal and an upper bound on error shown with the red disc. As the sheep move towards the goal, the dog robots are able to defend the protected zone despite uncertainty in the goal of the sheep. This uncertainty is decreasing with time because the dog robots use an online identification algorithm that is learning this goal.

IV

Thesis Conclusions

11 Conclusions and Outlook

We began this thesis with the motivating problem of defending a critical zone from adversarial/non-cooperative agents by using robots controlled by an observer. This problem requires the observer to (a) infer the underlying intent of all agents of the group and (b) incorporate this for planning motions of their robots. Part 1 of this thesis focused on the intent learning problem whereas part 2 of this focused on the control problem.

In part 1, we developed scalable algorithms for inferring a robust model of group dynamics by monitoring individual agents from their positions and velocities as measured by the observer. Towards that end, we developed identifiability conditions using the persistency of excitation criterion which tell when the task-inference problem is feasible. We developed online parameter estimators that can learn the task parameters such as goals of agents and their desired velocities. We developed another identifier that infers bounds on these parameters by identifying the set of active interactions of each agent under observation. While these estimators relied on perfect noiseless measurements and a-priori known safety margins, we relaxed these assumptions by taking recourse to inverse optimization. We developed robust inference algorithms that used KKT-loss and predictability loss as heuristics and posed mixed-integer quadratic programs to learn both task parameters and safety margins together. These estimators can account for noise in the measurements, suboptimality of agents and small amount of model mistmatch. To demonstrate the versatility of these estimators, we showed how we can use them to infer intents of humans in navigating in an indoor lab setting.

In part 2 of this thesis, we developed several control algorithms for robots to evoke a desired behavior out of the agents by orchestrating interactions of the robots with the agents. We used ideas from non-collocated partial feedback linearization and control barrier functions to pose this behavior shaping problem. We developed optimization-based control algorithms for a group of robots to prevent a flock of agents from breaching a protected zone by making the robots expend as little energy as possible. We provided both centralized and distributed implementations of our algorithms. The constraint based framework allowed us to include multiple protected zones. While the centralized implementation provided high budget efficiency and allowed for greater underactuation, it lacked the feasibility guarantee. The distributed algorithm, on the other hand, provided us with a feasibility guarantee yet required equal number of robots as flock agents. The proposed centralized and distributed algorithms required a-priori knowledge of the latent parameters of the group agents. In chapter 10, we relaxed this assumption and provided only estimates of these parameters and bounds on estimation error. We developed robust extension of the centralized algorithm by taking recourse to duality theory and s-procedure. Its correctness was validated by performing both experiments and simulations.

Finally, we integrated our online parameter learning algorithms and the proposed robust control algorithm to allow for simultaneous learning and control of the robots to prevent breach. Future extensions of this work will consider learning time-varying parameters and robust control techniques for time-varying systems. Additionally, future work should focus on developing estimation algorithms that allow for multimodal learning *i.e.* they produce a family of parameter estimates rather than just one, and similarly, future work should consider developing multi-modal robust control algorithms that can take multi-modal representations of agent intents for planning motions of the robots. Lastly, future work should consider experiments in which the flock agents are controlled by the humans to mimic adversarial agents and develop robust control algorithms that can learn time varying human intents and defend the zone reactively.

Bibliography

[1] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012.

[2] R. D'Andrea and G. E. Dullerud, "Distributed control design for spatially interconnected systems," *IEEE Transactions on automatic control*, vol. 48, no. 9, pp. 1478–1495, 2003.

[3] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy," in *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 209–215, IEEE, 2013.

[4] W. Kazmi, M. Bisgaard, F. Garcia-Ruiz, K. D. Hansen, and A. la Cour-Harbo, "Adaptive surveying and early treatment of crops with a team of autonomous vehicles," in *Proceedings of the 5th European Conference on Mobile Robots ECMR 2011*, pp. 253–258, 2011.

[5] M. Ouimet and J. Cortés, "Collective estimation of ocean nonlinear internal waves using robotic underwater drifters," *IEEE Access*, vol. 1, pp. 418–427, 2013.

[6] A. Dhariwal, G. S. Sukhatme, and A. A. Requicha, "Bacterium-inspired robots for environmental monitoring," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 2, pp. 1436–1443, IEEE, 2004.

[7] J. Cortés and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE Journal of Control, Measurement, and System Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[8] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous robots and agents*, pp. 9–16, Springer, 2007.

[9] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[10] J. Lin, A. S. Morse, and B. D. Anderson, "The multi-agent rendezvous problem-the asynchronous case," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 2, pp. 1926–1931, IEEE, 2004.

[11] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control*, vol. 27. Springer, 2008.

[12] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, 1987.

[13] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Transactions on Automatic control*, vol. 52, no. 5, pp. 863–868, 2007.

[14] 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.

[15] P. Ogren, M. Egerstedt, and X. Hu, "A control lyapunov function approach to multi-agent coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 847–851, 2002.

[16] Q. Gong, W. Kang, C. Walton, I. Kaminer, and H. Park, "Partial observability analysis of an adversarial swarm model," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 2, pp. 250–261, 2020.

[17] H. Park, Q. Gong, W. Kang, C. Walton, and I. Kaminer, "Observability analysis of an adversarial swarm's cooperation strategy," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pp. 992–997, IEEE, 2018.

[18] L. Li, A. Bayuelo, L. Bobadilla, T. Alam, and D. A. Shell, "Coordinated multi-robot planning while preserving individual privacy," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2188–2194, IEEE, 2019.

[19] C. Walton, I. Kaminer, Q. Gong, A. H. Clark, and T. Tsatsanifos, "Defense against adversarial swarms with parameter uncertainty," *Sensors*, vol. 22, no. 13, p. 4773, 2022.

[20] T. Tsatsanifos, A. H. Clark, C. Walton, I. Kaminer, and Q. Gong, "Modeling and control of large-scale adversarial swarm engagements," *arXiv:2108.02311*, 2021.

[21] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4, pp. 4159–4164, IEEE, 2004.

[22] A. Pierson and M. Schwager, "Controlling noncooperative herds with robotic herders," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 517–525, 2017.

[23] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, “Robot control of animal flocks,” in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell*, pp. 277–282, IEEE, 1998.

[24] R. Vaughan, N. Sumpter, A. Frost, and S. Cameron, “Robot sheepdog project achieves automatic flock control,” in *Proc. Fifth International Conference on the Simulation of Adaptive Behaviour*, vol. 489, p. 493, 1998.

[25] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, “Experiments in automatic flock control,” *Robotics and autonomous systems*, vol. 31, no. 1-2, pp. 109–117, 2000.

[26] M. W. Spong, “Partial feedback linearization of underactuated mechanical systems,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’94)*, vol. 1, pp. 314–321, IEEE, 1994.

[27] J. Grover, C. Liu, and K. Sycara, “Parameter identification for multirobot systems using optimization-based controllers,” in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pp. 173–180, IEEE, 2021.

[28] J. Grover, C. Liu, and K. Sycara, “Feasible region-based system identification using duality,” in *2021 European Control Conference (ECC)*, pp. 255–262, 2021.

[29] J. Grover, C. Liu, and K. Sycara, “System identification for safe controllers using inverse optimization,” *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 346–353, 2021.

[30] A. Rudenko, T. P. Kucner, C. S. Swaminathan, R. T. Chadalavada, K. O. Arras, and A. J. Lilienthal, “Thör: Human-robot navigation data collection and accurate motion trajectories dataset,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 676–682, 2020.

[31] J. Grover, Y. Lyu, W. Luo, C. Liu, J. Dolan, and K. Sycara, “Semantically-aware pedestrian intent prediction with barrier functions and mixed-integer quadratic programming,” *IFAC-PapersOnLine*, vol. 55, no. 41, pp. 167–174, 2022.

[32] J. Grover, N. Mohanty, C. Liu, W. Luo, and K. Sycara, “Noncooperative herding with control barrier functions: Theory and experiments,” in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 80–86, IEEE, 2022.

[33] N. Mohanty, J. Grover, C. Liu, and K. Sycara, “Distributed multirobot control for non-cooperative herding,” *To appear in the proceedings of Distributed Autonomous Robotic Systems (DARS) 2022*, 2022.

[34] J. S. Grover, C. Liu, and K. Sycara, “Control barrier functions-based semi-definite programs (cbf-sdps): Robust safe control for dynamic systems with relative degree two safety indices,” *arXiv:2208.12252*, 2022.

[35] A. Šošić, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koeppl, "Inverse reinforcement learning in swarm systems," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, (Richland, SC), p. 1413–1421, International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[36] X. Yu, W. Wu, P. Feng, and Y. Tian, "Swarm inverse reinforcement learning for biological systems," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 274–279, IEEE, 2021.

[37] R. Pinsler, M. Maag, O. Arenz, and G. Neumann, "Inverse reinforcement learning of bird flocking behavior," in *ICRA Swarms Workshop*, 2018.

[38] T. Costa, A. Laan, F. J. Heras, and G. G. De Polavieja, "Automated discovery of local rules for desired collective-level behavior through reinforcement learning," *Frontiers in Physics*, vol. 8, p. 200, 2020.

[39] S. Zhou, M. J. Phielipp, J. Sefair, S. I. Walker, and H. B. Amor, "Swarmnet: Towards imitation learning of multi-robot behavior with graph neural networks," -, -.

[40] S. Zhou, M. J. Phielipp, J. A. Sefair, S. I. Walker, and H. B. Amor, "Clone swarms: Learning to predict and control multi-robot systems by imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4092–4099, IEEE, 2019.

[41] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, vol. 3, pp. 2968–2973, IEEE, 2001.

[42] M. Schwager, C. Detweiler, I. Vasilescu, D. M. Anderson, and D. Rus, "Data-driven identification of group dynamics for motion prediction and control," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 305–324, 2008.

[43] M. Srinivasan, A. Chakrabarty, R. Quirynen, N. Yoshikawa, T. Mariyama, and S. Di Cairano, "Fast multi-robot motion planning via imitation learning of mixed-integer programs," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 598–604, 2021.

[44] P. Pierpaoli, H. Ravichandar, N. Waytowich, A. Li, D. Asher, and M. Egerstedt, "Inferring and learning multi-robot policies by observing an expert," *arXiv:1909.07887*, 2019.

[45] S. Rothfuß, J. Inga, F. Köpf, M. Flad, and S. Hohmann, "Inverse optimal control for identification in non-cooperative differential games," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14909–14915, 2017.

[46] J. Inga, A. Creutz, and S. Hohmann, "Online inverse linear-quadratic differential games applied to human behavior identification in shared control," in *2021 European Control Conference (ECC)*, pp. 353–360, IEEE, 2021.

[47] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Lucidgames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5485–5492, 2021.

[48] T. L. Molloy, J. J. Ford, and T. Perez, "Online inverse optimal control on infinite horizons," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1663–1668, IEEE, 2018.

[49] M. Johnson, N. Aghasadeghi, and T. Bretl, "Inverse optimal control for deterministic continuous-time nonlinear systems," in *52nd IEEE Conference on Decision and Control*, pp. 2906–2913, IEEE, 2013.

[50] P. Englert, N. A. Vien, and M. Toussaint, "Inverse kkt: Learning cost functions of manipulation tasks from demonstrations," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[51] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. J. Tomlin, and C. Stachniss, "Inferring objectives in continuous dynamic games from noise-corrupted partial state observations," *arXiv:2106.03611*, 2021.

[52] L. Peters, D. Fridovich-Keil, V. Rubies-Royo, C. J. Tomlin, and C. Stachniss, "Cost inference in smooth dynamic games from noise-corrupted partial state observations," *arXiv:2106.03611*, 2021.

[53] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, IEEE, 2008.

[54] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[55] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[56] B. Bat-Erdene and O.-E. Mandakh, "Shepherding algorithm of multi-mobile robot system," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, pp. 358–361, IEEE, 2017.

[57] N. Sumpter, A. J. Bulpitt, R. T. Vaughan, R. D. Tillett, and R. D. Boyle, "Learning models of animal behaviour for a robotic sheepdog.," in *MVA*, pp. 577–580, 1998.

[58] J.-M. Lien and E. Pratt, "Interactive planning for shepherd motion.," in *AAAI Spring Symposium: Agents that Learn from Human Teachers*, pp. 95–102, 2009.

[59] S. Gade, A. A. Paranjape, and S.-J. Chung, "Herding a flock of birds approaching an airport using an unmanned aerial vehicle," in *AIAA guidance, navigation, and control conference*, p. 1540, 2015.

[60] S. Gade, A. A. Paranjape, and S.-J. Chung, "Robotic herding using wavefront algorithm: Performance and stability," in *AIAA Guidance, Navigation, and Control Conference*, p. 1378, 2016.

[61] J.-M. Lien, S. Rodriguez, J.-P. Malric, and N. M. Amato, "Shepherding behaviors with multiple shepherds," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 3402–3407, IEEE, 2005.

[62] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. A. Abbass, "A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 523–537, 2020.

[63] A. Pierson and M. Schwager, "Bio-inspired non-cooperative multi-robot herding," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1843–1849, IEEE, 2015.

[64] R. A. Licitra, Z. I. Bell, E. A. Doucette, and W. E. Dixon, "Single agent indirect herding of multiple targets: A switched adaptive control approach," *IEEE Control Systems Letters*, vol. 2, no. 1, pp. 127–132, 2017.

[65] R. A. Licitra, Z. D. Hutcheson, E. A. Doucette, and W. E. Dixon, "Single agent herding of n-agents: A switched systems approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14374–14379, 2017.

[66] E. Sebastián and E. Montijano, "Multi-robot implicit control of herds," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1601–1607, IEEE, 2021.

[67] M. Bacon and N. Olgac, "Swarm herding using a region holding sliding mode controller," *Journal of Vibration and Control*, vol. 18, no. 7, pp. 1056–1066, 2012.

[68] J. S. Grover, C. Liu, and K. Sycara, "Deadlock analysis and resolution for multi-robot systems," in *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pp. 294–312, Springer, 2021.

[69] P. A. Ioannou and J. Sun, *Robust adaptive control.* Courier Corporation, 2012.

[70] T. F. Edgar, "Recursive least squares parameter estimation for linear steady state and dynamic models," *Department of Chemical Engineering University of Texas, Austin*, 2010.

[71] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[72] V. Adetola and M. Guay, "Performance improvement in adaptive control of linearly parameterized nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2182–2186, 2010.

[73] J. Na, M. N. Mahyuddin, G. Herrmann, X. Ren, and P. Barber, "Robust adaptive finite-time parameter estimation and control for robotic systems," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 16, pp. 3045–3071, 2015.

[74] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[75] C. Yang, Y. Jiang, W. He, J. Na, Z. Li, and B. Xu, "Adaptive parameter estimation and control design for robot manipulators with finite-time convergence," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8112–8123, 2018.

[76] K. S. Narendra and A. M. Annaswamy, *Stable adaptive systems.* Courier Corporation, 2012.

[77] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge university press, 2004.

[78] M. Hartman, N. Bauer, and A. R. Teel, "Robust finite-time parameter estimation using a hybrid systems framework," *IEEE transactions on automatic control*, vol. 57, no. 11, pp. 2956–2962, 2012.

[79] V. Adetola and M. Guay, "Finite-time parameter estimation in adaptive control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, pp. 807–811, 2008.

[80] R. K. Ahuja and J. B. Orlin, "Inverse optimization," *Operations Research*, vol. 49, no. 5, pp. 771–783, 2001.

[81] D. Bertsimas, V. Gupta, and I. C. Paschalidis, "Inverse optimization: A new perspective on the black-litterman model," *Operations research*, vol. 60, no. 6, pp. 1389–1403, 2012.

[82] S. Carr and W. Lovejoy, "The inverse newsvendor problem: Choosing an optimal demand portfolio for capacitated resources," *Management Science*, vol. 46, no. 7, pp. 912–927, 2000.

[83] A. Aswani, Z.-J. Shen, and A. Siddiq, "Inverse optimization with noisy data," *Operations Research*, vol. 66, no. 3, pp. 870–892, 2018.

[84] A. Keshavarz, Y. Wang, and S. Boyd, "Imputing a convex objective function," in *2011 IEEE international symposium on intelligent control*, pp. 613–619, IEEE, 2011.

[85] P. M. Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn, “Data-driven inverse optimization with imperfect information,” *Mathematical Programming*, vol. 167, no. 1, pp. 191–234, 2018.

[86] D. Bertsimas, V. Gupta, and I. C. Paschalidis, “Data-driven estimation in equilibrium using inverse optimization,” *Mathematical Programming*, vol. 153, no. 2, pp. 595–633, 2015.

[87] C. Dong, Y. Chen, and B. Zeng, “Generalized inverse optimization through online learning,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, (Red Hook, NY, USA), p. 86–95, Curran Associates Inc., 2018.

[88] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, “Formally correct composition of coordinated behaviors using control barrier certificates,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3723–3729, IEEE, 2018.

[89] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, “Dual decomposition for multi-agent distributed optimization with coupling constraints,” *Automatica*, vol. 84, pp. 149–158, 2017.

[90] G. Notarstefano, I. Notarnicola, A. Camisa, *et al.*, “Distributed optimization for smart cyber-physical networks,” *Foundations and Trends® in Systems and Control*, vol. 7, no. 3, pp. 253–383, 2019.