Learning Large-scale Multi-agent Control with Safety Certificates

by

Zengyi Qin

B.E., Tsinghua University (2020)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Department of Aeronautics and Astronautics
July 9, 2022

Certified by .....
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Learning Large-scale Multi-agent Control with Safety Certificates

by

Zengyi Qin

Submitted to the Department of Aeronautics and Astronautics
on July 9, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Multi-agent intelligence in autonomous systems has been fascinating roboticists for decades. The recent advances in machine learning has created unprecedented opportunities for achieving ultimate multi-agent intelligence and full autonomy in a data-driven way. However, a fundamental bottleneck of machine learning-based methods is their safety and reliability in controlling the autonomous system at large scale, due to the lack of formal safety guarantee. In addressing these challenges, we develop: (1) An machine learning-based large-scale multi-agent control framework with safety certificates, which simultaneously enjoys the versatility of machine learning and the assurance of safety. (2) A multi-agent trajectory tracking framework with convergence and safety guarantees. (3) A general method to learn safe controllers for black-box systems with unknown dynamics. Comprehensive experiments have shown that the proposed methods have notable performance in terms of safety rate, task completion rate, computational efficiency and large-scale scalability.

Thesis Supervisor: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

I would like to express my sincere gratitude to Professor Chuchu Fan and Professor Jonathan P. How. Chuchu guided me to focus on the crucial problems in safe autonomy, and gave me freedom to explore what truly fascinates me. Jonathan gave me crucial support in the last semester of my Master's study, and guided me through the transitional period to my doctoral program.

I am immensely fortunate to have the opportunities to work with a group of brilliant collaborators: Dawei Sun, Charles Dawson, Yue Meng, Yuxiao Chen, Kaiqing Zhang and Jingkai Chen. I am grateful for the countless brainstorming when building theoretical frameworks, the intense nights before deadlines, and the shared excitement when we succeeded. Without these extraordinary collaborators, it would be extremely hard for me to complete my research projects with absolute rigor, genuine novelty, notable quality and amazing visual impact.

I could never thank my family enough for their unwavering support and unconditional love. Thanks to their education, I had early exposure to automation, electronics and engineering, where I found my true passion. I would thank them for trusting me and letting me pursue what I truly believe in. I am proud of having them in my life!

Last but not least, I would like to acknowledge the funding support from the MathWorks Fellowship, the DARPA Assured Autonomy under contract FA8750-19-C-0089 and the Singapore Defense Science and Technology Agency.

Contents

Chapter 1

Introduction

Multi-agent control plays an important role in robotics. How to endow robots with human-level collective intelligence has been fascinating to roboticists for decades. While a multi-agent system should satisfy numerous properties, the *safety* is always of crucial importance. For example, we cannot tolerate frequent car crashes in an autonomous driving vehicle network. Neither can we imagine operating a fragile drone delivery system in real cities. As the number of agents increases, the multi-agent system becomes more and more complex, and ultimately, it could be extremely difficult, if not impossible, to predict failure or guarantee system safety. We argue that the safety of a multi-agent system can have a tremendous influence on its practicality and ability to make real-world impact. In this chapter, we provide an overview of the fundamental problems that we focus on, summarize our contribution and outline the whole thesis.

1.1 Large-scale Safe Multi-agent Control

Machine learning has created unprecedented opportunities for achieving full autonomy. Comparing to classical handcrafted controller design, machine learning-based methods are more widely applicable and versatile. For example, in order to control humanoid robots, classical methods [86] solve non-linear model predictive control (MPC) problems that are computationally expensive and easily trapped in numerical issues. Also, they require nearly complete knowledge of the dynamics of the system, which is unfortunately not always available. By contrast, reinforcement learning, as a branch of machine learning, shows remarkable results [50] and is applicable to a wide range of non-linear control problems, especially for problems that are difficult to model in an analytical way. Nevertheless, machine learning-based methods in autonomous systems can and do fail due to the lack of formal guarantees and limited generalization capability, which poses significant challenges for developing safety-critical autonomous systems, especially large-scale multi-agent systems, that are provably dependable. One of the emphasizes of this thesis is to develop reliable machine learning control methods that simultaneously possess the versatility of machine learning and the capability to provide safety guarantees.

Safety involves numerous aspects in multi-agent control. The most important safety requirement is that the agents should not collide with each other or with obstacles. In a single-agent system, the agent only needs to avoid obstacles, and there already exists a wide range of motion planning methods to deal with the single-agent case. Nevertheless, in a multi-agent system, the collision-avoidance is complicated by the interactions among the agents. The dimension of state space in the multi-agent system grows linearly with the

number of agents, but the volume and the number of possible states grows exponentially, which makes the analysis challenging. In addition to collision avoidance, the other safety requirements include speed and rotation limits, avoiding dangerous areas and so forth. The whole system is safe if and only if all agents are safe.

Recently, there has been increasing research interest in bridging model-based control and data-driven ML to address complex dynamical control problems [19, 12, 85, 20, 87, 79, 30], with the expectation of using such a combined approach to tackle the scalability issues and build large-scale multi-agent safe autonomous systems. We follow this line of works and study the problem of using ML to find robust, safe, and verifiable feedback control for large-scale multi-agent dynamical systems, which are known to be extremely challenging due to the lack of scalable and provably correct approaches. While many learning-based approaches for control systems, such as reinforcement learning (RL), have focused on how to train control strategies for complex and even unknown dynamical systems [51, 78, 77, 37], these methods do not normally come with formal guarantees on safety and robustness of the closed-loop systems. On the other side, although there is a rich literature on multi-agent trajectory planning and control with deterministic safety guarantees [13, 31, 57, 80], these centralized methods suffer from complexity issues, and very limited progress has been made on finding solutions that can be used on a very large number of agents.

1.1.1 Jointly Learning Controllers and Safety Certificates

Control certificates [12, 42, 20] can serve as proofs for the satisfaction of the desired properties of a system, under certain control policies. For example, Control Contraction Metrics (CCM) [54, 60] ensure the existence of feedback

controllers so that the controlled systems can be proved to converge to desired behaviors, and Control Barrier Functions (CBF) [6, 5, 10, 17, 19, 18] can supervise the synthesis of controllers such that the closed-loop (multi-agent) systems are guaranteed to always stay in certain invariant sets that encode safety requirements. We will exploit the combinatorial use of such control certificates to guide the synthesis of safe and robust controllers and provide strong theoretical guarantees.

One of the biggest challenges in using the above control theoretical approaches is the construction of certificates. It is extremely difficult to craft CCM and CBF by hand for complex nonlinear and nonholonomic systems. Other optimization-based approaches such as sum-of-squares (SoS) [82, 81, 6] cannot scale to large-dimension and large-scale systems. Therefore, there are very few existing methods that scale to systems with hundreds or thousands of agents. Recent advancements in neural networks (NN) stem a growing interest in learning-based control certificates [76, 83, 42, 9, 88, 73, 94, 12].

Our principle of building safe machine learning-based control method is to *jointly learn the controller and its safety certificate*. The parameterized controller and safety certificate are jointly optimized to satisfy a set of conditions. When such a controller is found, it is naturally endowed with safety guarantee provided by the jointly learned certificate.

1.1.2 Scaling Up to Large-scale Multi-agent Systems

Due to the *curse of dimensionality*, it is extremely difficult and computationally expensive for a *centralized* control strategy to scale up to an environment with an arbitrarily large number of agents. Instead, a decentralized controller design will be a fundamental throughout this thesis. In decentralized control, each agent has its own controller, and it is not necessary to have a centralized

controller handling all the agents simultaneously. Each agent takes its local observation as input and computes its own control command. Such a decentralized structure allows the computational cost to grow linearly, rather than exponentially, as the number of agents increases. A key challenge in decentralized control is how to ensure the global safety when each agent acts based on its local observation, and we will show how to achieve the global safety certificate by means of decentralized local safety certificates. We aim to achieve an improved balance of scalability and safety using data-driven approaches to learn decentralized control policies that are certified by the jointly learned control certificates and can theoretically be used on any number of agents.

1.1.3 Safe and Convergent Trajectory Tracking

The ability of trajectory tracking and goal reaching is crucial for multi-agent robotic systems to complete certain tasks. Convergence to the reference trajectories may cause conflicts among agents, particularly when the reference trajectories collide. It is common that a trajectory tracking algorithm cannot guarantee safety or resolve conflicts in multi-agent interaction. Similarly, the control algorithms that ensures multi-agent safety normally cannot guarantee the actual trajectory can converge to the reference trajectory. In this thesis, we will present a framework that jointly guarantees safety and the convergence to the reference trajectories.

1.1.4 Extension to Black-box Dynamics

Many dynamical systems in the real-world are black-box and lack accurate models. The most popular model-free approach to handle such black-box systems is safe reinforcement learning, which enforce safety and performance

by maximizing the expectation of the cumulative reward and constraining the expectation of the cost to be less or equal to a given threshold. The biggest disadvantage of safe RL methods is the lack of systematic or theoretically grounded way of designing cost function and reward functions, which heavily rely on empirical trials and errors. The lack of explainable safety guarantees and low sampling efficiency also make safe RL methods difficult to exhibit satisfactory performance.

The model-based control methods with safety guarantees are not directly applicable to model-free scenarios where the accurate model dynamics are not available. In addressing the challenge, we will present a novel framework that jointly learn the safety certificates and controllers for black-box systems.

1.2 Summary of Contributions

Safety and scalability are of fundamental importance in multi-agent controller synthesis. Machine learning brings new possibility to achieve the generality and versatility that are beyond the reach of classical control methods, but the safety and robustness issues must be addressed before being applied to real world. The objective of the thesis is to develop a large-scale multi-agent control method that jointly enjoys the capability of machine learning and the guarantees of safety certificates. The contributions of this thesis are three-fold:

1. Development of the joint-learning framework of multi-agent controllers and their safety certificates. The learned controllers are fully decentralized and is applicable to massive systems with an arbitrarily large number of agents. In addition, we also provide the theoretical analysis that proves the high probability guarantees to achieve global safety.

2. Development of the large-scale multi-agent trajectory tracking approach that jointly guarantees convergence to the reference trajectories and the safety of the multi-agent system.

3. Development of the black-box system control method that possesses the safety certificates similar to white-box systems. This greatly improves the practicality of the proposed joint-learning work, as the accurate model dynamics of many real-world systems are difficult to obtain.

1.3 Structure of Thesis

In Chapter 2, we introduce the mathematical and control-theoretical preliminaries. In Chapter 3, we develop the joint-learning framework for multi-agent controllers and their safety certificates. In Chapter 4, we develop the trajectory tracking method with convergence and safety guarantees. In Chapter 5, we make a non-trivial extension from white-box to black-box systems with unknown dynamics, and develop the machine learning-based safe control method. Finally, in Chapter 6, we conclude with a brief summary.

Chapter 2

Preliminaries

2.1 Mathematical Notations

A multi-agent system is defined as $\mathcal{M} = \langle \mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_N \rangle$, where each agent $\mathcal{A}_i = \langle \mathcal{X}_i, \mathcal{O}_i, \mathcal{U}_i, \mathbf{f}_i, \boldsymbol{\pi}_i \rangle$. $\mathcal{X}_i \in \mathbb{R}^n$ is the state space (e.g., position, velocity and orientation space). $\mathcal{O}_i$ is the observation space, which contains the state of surrounding agents and other obstacles. $\mathcal{U}_i \in \mathbb{R}^m$ is the control input space. Define the state-observation space as $\mathcal{S}_i = \mathcal{X}_i \times \mathcal{O}_i$. The dynamics function $\mathbf{f}_i : \mathcal{X}_i \times \mathcal{U}_i \mapsto \mathcal{X}_i$ specifies the evolution of state as:

$$\dot{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i), \tag{2.1}$$

where $\mathbf{u}_i$ is the control input. The controller $\boldsymbol{\pi}_i : \mathcal{S}_i \mapsto \mathcal{U}_i$ gives the control input $\mathbf{u}_i$ based on the current state and observation:

$$\mathbf{u}_i = \boldsymbol{\pi}_i(\mathbf{x}_i, \mathbf{o}_i). \tag{2.2}$$

We denote the indices from 1 to N: $\{1, \cdots, N\}$ as $[\![N]\!]$. For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, the notation $A \succ 0$ ($A \prec 0$) means A is positive (negative) definite. $A \succeq 0$ ($A \preceq 0$) means A is positive (negative) semi-definite. Also, $A \succ B$ means $A - B \succ 0$. For $A \in \mathbb{R}^{n\times n}$, we denote $A + A^T$ by $\widehat{A}$. For a matrix-valued function $M(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n\times n}$, its element-wise Lie derivative along a vector $\mathbf{v} \in \mathbb{R}^n$ is $\partial_{\mathbf{v}} M := \sum_i v^i \frac{\partial M}{\partial x^i}$, where x^i and v^i denote the i-th elements of the vectors. We denote the cross product of $\mathcal{X}_i, i \in [\![N]\!]$ as $\overset{N}{\underset{i=1}{\otimes}} \mathcal{X}_i$, and use $p \downarrow q$ to denote the projection from vector p to the vector space of q.

2.2 Safety for Multi-agent Systems

The safety of an individual agent can be determined by checking its state-observation pair $(\mathbf{x}_i, \mathbf{o}_i)$. We always assume that we are given a safety metric $d : \mathcal{S}_i \mapsto \mathbb{R}$, such that:

$$r(\mathbf{x}_i, \mathbf{u}_i) \geq 0 \iff \mathcal{A}_i \text{ is safe.} \tag{2.3}$$

And the global safety of the multi-agent system is achieved if and only if all agents are safe:

$$d(\mathbf{x}_i, \mathbf{u}_i) \geq 0, \forall i = 1, 2, \cdots, N \iff \mathcal{M} \text{ is safe.} \tag{2.4}$$

For an individual agent, its state-observation space $\mathcal{S}_i$ can be further specified into $\mathcal{S}_{i,s}, \mathcal{S}_{i,d}$ and $\mathcal{S}_{i,0}$. $\mathcal{S}_{i,s} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid d(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}$ is the safe set. $\mathcal{S}_{i,d} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid d(\mathbf{x}_i, \mathbf{o}_i) < 0\}$ is the dangerous set. $\mathcal{S}_{i,0}$ is the set of all possible initial conditions. It is assumed that all initial conditions are safe, namely, $\mathcal{S}_{i,0} \subset \mathcal{S}_{i,s}$.

Chapter 3

Joint-learning of Controllers and Safety Certificates

3.1 Introduction

We study the multi-agent safe control problem, where agents should avoid collisions to static obstacles and collisions with each other while reaching their goals. Our core idea is to learn the multi-agent control policy *jointly* with learning the control barrier functions (CBFs) as *safety certificates*. We propose a new joint-learning framework that can be implemented in a *decentralized* fashion, which can adapt to an arbitrarily large number of agents. Building upon this framework, we further improve the scalability by incorporating neural network architectures that are invariant to the quantity and permutation of neighboring agents. In addition, we propose a new spontaneous policy refinement method to further enforce the certificate condition during testing.

Experimental results are indeed promising. We study both 2D and 3D safe multi-agent control problems, each with several distinct environments and complex nonholonomic dynamics. Our joint-learning framework performs

exceptionally well: our control policies trained on scenarios with 8 agents can be used on up to 1024 agents while maintaining low collision rates, which has notably pushed the boundary of learning-based safe multi-agent control. Speaking of which, 1024 is not the limit of our approach, but rather due to the limited computational capability of our laptop used for the experiments. We also compare our approach with both leading learning-based methods [53, 55, 106] and traditional planning methods [27, 56]. Our approach outperforms all the other approaches in terms of both completing the tasks and maintaining safety.

3.2 Related Work

Learning-Based Safe Control via CBF. Barrier certificates [65] and CBF [99] is a well-known effective tool for guaranteeing the safety of nonlinear dynamic systems. However, the existing methods for constructing CBFs either rely on specific problem structures [16] or do not scale well [63]. Recently, there has been an increasing interest in learning-based and data-driven safe control via CBFs, which primarily consist of two categories: *learning CBFs from data* [9, 42, 76, 83], and *CBF-based approach for controlling unknown systems* [19, 88, 97, 98]. Our work is more pertinent to the former and is complementary to the latter, which usually assumes that the CBF is provided. None of these learning-enabled approaches, however, has addressed the multi-agent setting.

Multi-Agent Safety Certificates and Collision Avoidance. Restricted to holonomic systems, guaranteeing safety in multi-agent systems has been approached by limiting the velocities of the agents [4, 95]. Later, [10, 97] have proposed the framework of *multi-agent CBF* to generate collision-free con-

trollers, with either perfectly known system dynamics [10], or with worst-case uncertainty bounds [97]. Recently, [17] has proposed a *decentralized* controller synthesized approach under this CBF framework, which is scalable to an arbitrary number of agents. However, in [17] the CBF controller relies on online integration of the dynamics under the backup strategy, which can be computationally challenging for complex systems. Due to space limit, we omit other non-learning multi-agent control methods, but acknowledge their importance.

Safe Multi-Agent (Reinforcement) Learning (MARL). Safety concerns have drawn increasing attention in MARL, especially with the applications to safety-critical multi-agent systems [67, 79, 106]. Under the CBF framework, [18] considered the setting with *unknown* system dynamics, and proposed to design robust multi-agent CBFs based on the *learned* dynamics. This mirrors the second category mentioned above in single-agent learning-based safe control, which is perpendicular to our focus. RL approaches have also been applied for multi-agent collision avoidance [15, 26, 55, 105]. Nonetheless, no formal guarantees of safety were established in these works. One exception is [106], which proposed a multi-agent model predictive shielding algorithm that provably guarantees safety for any policy learned from MARL, which differs from our multi-agent CBF-based approach. More importantly, none of these MARL-based approaches scale to a massive number of, e.g., thousands of agents, as our approach does. The most scalable MARL platform, to the best of our knowledge, is [107], which may handle a comparable scale of agents as ours, but with *discrete* state-action spaces. This is in contrast to our continuous-space models that can model practical control systems such as robots and drones.

3.3 Decentralized Control Barrier Functions

CBF [6] is a powerful tool to guarantee system safety by means of forward invariance. In the multi-agent case, the CBF of an agent i is a function $h_i : \mathcal{X}_i \times \mathcal{O}_i \mapsto \mathbb{R}$, satisfying three properties:

$$\begin{aligned} &h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0, && \forall (\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,0} \\ &h_i(\mathbf{x}_i, \mathbf{o}_i) < 0, && \forall (\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,d} \\ &\dot{h}_i(\mathbf{x}_i, \mathbf{o}_i) + \alpha(h_i(\mathbf{x}_i, \mathbf{o}_i)) \geq 0, && \forall (\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,p}, \end{aligned} \tag{3.1}$$

where $\dot{h}_i$ can be written as:

$$\dot{h}_i(\mathbf{x}_i, \mathbf{o}_i) = \nabla_{\mathbf{x}} h_i \cdot \dot{\mathbf{x}}_i + \nabla_{\mathbf{o}} h_i \cdot \dot{\mathbf{o}}_i = \nabla_{\mathbf{x}} h_i \cdot \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i) + \nabla_{\mathbf{o}} h_i \cdot \dot{\mathbf{o}}_i, \tag{3.2}$$

and the set $\mathcal{S}_{i,p}$ is defined as:

$$\mathcal{S}_{i,p} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}, \tag{3.3}$$

and the function $\alpha : \mathbb{R} \mapsto \mathbb{R}$ is a class-$\mathcal{K}$ function that is strictly increasing and satisfies $\alpha(0) = 0$. Note that the third condition depends on the control input π_i. We refer to the three conditions in Equation (3.1) as *decentralized CBF conditions*. When the conditions are satisfied, the global safety can then be guaranteed as in Proposition 1:

Proposition 1 (Multi-Agent Safety with Decentralized CBF)
Given the multi-agent control policy $\mathbf{u}_i = \boldsymbol{\pi}_i(\mathbf{x}_i, \mathbf{o}_i)$ *and decentralized CBF* $h_i(\mathbf{x}_i, \mathbf{o}_i)$ *satisfying the conditions in* (3.1)*, if the initial condition of the multi-*

agent system satisfies:

$$(\mathbf{x}_i(t), \mathbf{o}_i(t)) \in \mathcal{S}_{i,p}, \quad 1 \leq i \leq N, \quad t = 0 \tag{3.4}$$

then we have:

$$(\mathbf{x}_i(t), \mathbf{o}_i(t)) \in \mathcal{S}_{i,p}, \quad 1 \leq i \leq N, \quad \forall t > 0. \tag{3.5}$$

Namely, if initially the agents are in $\mathcal{S}_{i,p}$, they will never leave $\mathcal{S}_{i,p}$, which is a forward invariant set. Since $\mathcal{S}_{i,p} \cap \mathcal{S}_{i,d} = \emptyset$, the agents will never enter the dangerous set $\mathcal{S}_{i,d}$. Thus, the multi-agent system is safe.

Proof. Let us assume that $(\mathbf{x}_i, \mathbf{o}_i)$ is on the boundary of $\mathcal{S}_{i,p}$ and is trying to leave $\mathcal{S}_{i,p}$. Since $h_i(\mathbf{x}_i, \mathbf{o}_i) = 0$, based on the third condition in (3.1) and $\alpha(0) = 0$, we have:

$$\dot{h}_i(\mathbf{x}_i, \mathbf{o}_i) + \alpha(h_i(\mathbf{x}_i, \mathbf{o}_i)) = \dot{h}_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0, \tag{3.6}$$

which means $h_i(\mathbf{x}_i, \mathbf{o}_i)$ is non-decreasing on the boundary of $\mathcal{S}_{i,p}$. Thus, we have $h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0$ forever, and $(\mathbf{x}_i, \mathbf{o}_i)$ will never leave $\mathcal{S}_{i,p}$. □

3.4 Scalable Learning of Decentralized CBF

Based on the theory in Section 3.3, if we can jointly find the controller $\boldsymbol{\pi}_i$ and CBF h_i satisfying the conditions (3.1), then the multi-agent system is guaranteed to be safe. It is extremely challenging to handcraft $\boldsymbol{\pi}_i$ and h_i. When the model dynamics are non-linear, handcrafting $\boldsymbol{\pi}_i$ or h_i could become nearly impossible. Even if we are lucky enough to successfully handcraft a h_i for a complex system, we will have to repeat the process once the system is

changed. Therefore, we use machine learning techniques to automatically find $\boldsymbol{\pi}_i$ and h_i, which is shown to be a general and easy-to-use method for a large variety of linear and non-linear systems.

Following the theory in Section 3.3, we consider the practical learning of safe multi-agent control with neural barrier certificates, i.e., using neural networks for $\mathbf{u}_i$ and $\boldsymbol{\pi}_i$. We will present the formulation of loss functions in Section 3.4.1. Section 3.4.2 presents the neural network architecture of h_i and π_i, which are invariant to the quantity and permutation of neighboring agents. Section 3.4.3 demonstrates a spontaneous policy refinement method that enables the control policy to satisfy the decentralized CBF conditions as possible as it could during testing.

3.4.1 Loss Functions for the Joint-learning Framework

The main idea is to *jointly* learn the control policies and control barrier functions in multi-agent systems. During training, the CBFs regulate the control policies to satisfy the decentralized CBF conditions (3.1) so that the learned policies are safe. All agents are put into a single environment to generate state-observation samples, which forms datasets $\mathcal{D}_i = \{(\mathbf{x}_i, \mathbf{o}_i)_1, (\mathbf{x}_i, \mathbf{o}_i)_2, \cdots\}$, $1 \leq i \leq N$. The samples in the dataset are used to evaluate and minimize the loss empirical loss function $\mathcal{L}^c = \Sigma_i \mathcal{L}_i^c$, where $\mathcal{L}_i^c$ is the loss function for agent i formulated as:

$$\mathcal{L}_i^c(\boldsymbol{\theta}_i, \boldsymbol{\omega}_i) = \mathcal{L}_{i,0}^c(\boldsymbol{\theta}_i) + \mathcal{L}_{i,d}^c(\boldsymbol{\theta}_i) + \mathcal{L}_{i,p}^c(\boldsymbol{\theta}_i, \boldsymbol{\omega}_i) \tag{3.7}$$

The three terms are:

$$\begin{aligned}
\mathcal{L}^c_{i,0}(\boldsymbol{\theta}_i) &= \frac{1}{|\mathcal{S}_{i,0} \cap \mathcal{D}_i|} \sum_{(\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,0} \cap \mathcal{D}_i} \max\left(0, \gamma - h_i^{\boldsymbol{\theta}_i}(\mathbf{x}_i, \mathbf{o}_i)\right) \\
\mathcal{L}^c_{i,d}(\boldsymbol{\theta}_i) &= \frac{1}{|\mathcal{S}_{i,d} \cap \mathcal{D}_i|} \sum_{(\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,d} \cap \mathcal{D}_i} \max\left(0, \gamma + h_i^{\boldsymbol{\theta}_i}(\mathbf{x}_i, \mathbf{o}_i)\right) \\
\mathcal{L}^c_{i,p}(\boldsymbol{\theta}_i, \boldsymbol{\omega}_i) &= \frac{1}{|\mathcal{S}_{i,p} \cap \mathcal{D}_i|} \sum_{(\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,p} \cap \mathcal{D}_i} \max\left(0, \gamma - \dot{h}_i^{\boldsymbol{\theta}_i} - \alpha(h_i^{\boldsymbol{\theta}_i})\right),
\end{aligned} \tag{3.8}$$

where the time derivative $\dot{h}_i^{\boldsymbol{\theta}_i}$ in $\mathcal{L}^c_{i,p}(\boldsymbol{\theta}_i, \boldsymbol{\omega}_i)$ is written as:

$$\dot{h}_i^{\boldsymbol{\theta}_i} = \nabla_{\mathbf{x}} h_i^{\boldsymbol{\theta}_i} \cdot \mathbf{f}_i\left(\mathbf{x}_i, \boldsymbol{\pi}_i^{\boldsymbol{\omega}_i}(\mathbf{x}_i, \mathbf{o}_i)\right) + \nabla_{\mathbf{o}} h_i^{\boldsymbol{\theta}_i} \cdot \dot{\mathbf{o}}_i \tag{3.9}$$

The γ is a positive margin, which is $\gamma = 10^{-2}$ in implementation. $\boldsymbol{\theta}_i$ and $\boldsymbol{\omega}_i$ are neural network parameters. On the right side of Equation (3.7), the three items enforce the three CBF conditions respectively. Directly computing the third term could be challenging since we need to evaluate $\dot{\mathbf{o}}_i$, which is the time derivative of the observation. Instead, we approximate $\dot{h}(\mathbf{x}_i, \mathbf{o}_i)$ numerically:

$$\dot{h}(\mathbf{x}_i, \mathbf{o}_i) = \frac{1}{\Delta t}\Big(h(\mathbf{x}_i(t + \Delta t), \mathbf{o}_i(t + \Delta t)) - h(\mathbf{x}_i(t), \mathbf{o}_i(t))\Big) \tag{3.10}$$

For the class-$\mathcal{K}$ function $\alpha(\cdot)$, we simply choose a linear function $\alpha(h) = \lambda h$. Note that $\mathcal{L}^c$ mainly considers safety instead of goal reaching. To train a safe control policy $\boldsymbol{\pi}_i(\mathbf{x}_i, \mathbf{o}_i)$ that can drive the agent to the goal state, we also minimize the distance between $\mathbf{u}_i$ and $\mathbf{u}_i^g$, where $\mathbf{u}_i^g$ is the reference control input computed by classical controller $\boldsymbol{\pi}_i^g$ (e.g., LQR and PID controllers) to reach the goal. The goal reaching loss $\mathcal{L}^g = \Sigma_i \mathcal{L}_i^g$, where $\mathcal{L}_i^g$ is formulated as:

$$\mathcal{L}_i^g(\boldsymbol{\omega}_i) = \frac{1}{|\mathcal{S}_i \cap \mathcal{D}_i|} \sum_{(\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_i \cap \mathcal{D}_i} ||\boldsymbol{\pi}_i^{\boldsymbol{\omega}_i}(\mathbf{x}_i, \mathbf{o}_i) - \boldsymbol{\pi}_i^g(\mathbf{x}_i)||_2 \tag{3.11}$$

The final loss function $\mathcal{L} = \mathcal{L}^c + \eta\mathcal{L}^g$, where η is a balance weight that is set to 0.1 in our experiments.

Implementation Details In training, all agents are put into the random environment, which is not necessarily the same as the testing environment, to collect state-observation pairs $(\mathbf{x}_i, \mathbf{o}_i)$ under their current policies with probability 0.95 and random policies with probability 0.05. The collected $(\mathbf{x}_i, \mathbf{o}_i)$ are stored in dataset and in every step of policy update, 128 $(\mathbf{x}_i, \mathbf{o}_i)$ are randomly sampled from the temporary dataset to calculate the total loss $\mathcal{L}$. We minimize $\mathcal{L}$ by applying stochastic gradient descent with learning rate 10^{-3} and weight decay 10^{-6} to $\boldsymbol{\theta}_i$ and $\boldsymbol{\omega}_i$, which are the parameters of the CBF and control policies. Note that the gradients are computed by back-propagation rather than policy gradients because $\mathcal{L}$ is differentiable w.r.t. $\boldsymbol{\theta}_i$ and $\boldsymbol{\omega}_i$.

Iterative Data Collection and Training. It is important to note that we did not use a fixed set of state-observation pairs to train the decentralized CBF and controllers. Instead, we adopted an on-policy training strategy, where the training data are collected by running the current system. The collected state-observation pairs are stored in a temporary dataset that is used to calculate the loss terms and update the decentralized CBF and controllers via gradient descent. Then the updated controllers are used to run the system and re-generate new state-observation pairs as training data. The iterative data collection and training is performed until the loss converges. Such a training process is crucial for generalizing to testing scenarios.

3.4.2 Quantity-Permutation Invariant Observation Encoder

Recall that in Chapter 2, we define $\mathcal{O}_i$ as the observation space that contains the states of neighboring agents and other obstacles. This means the observation $\mathbf{o}_i$ has dynamic dimension and permutation, since the quantity and permutation of neighboring agents and obstacles can change with time. In order to scale to an arbitrary number of agents, there are two pivotal principles of designing the neural network architectures of $h_i(\mathbf{x}_i, \mathbf{o}_i)$ and $\boldsymbol{\pi}_i(\mathbf{x}_i, \mathbf{o}_i)$. First, the architecture should be able to dynamically adapt to the changing quantity of observed agents that affects the dimension of $\mathbf{o}_i$. Second, the architecture should be invariant to the permutation of observed agents, which should not affect the output of h_i or $\boldsymbol{\pi}_i$. All these challenges arise from encoding the local observation $\mathbf{o}_i$. Inspired by PointNet [66], we leverage the *max pooling* layer to build the quantity-permutation invariant observation encoder.

Let us start with a simple example with input observation $\mathbf{o}_i(t) \in \mathbb{R}^{n \times N_i(t)}$, where n is the dimension of state and $N_i(t)$ is the set of the neighboring agents at time t. n is fixed, while $N_i(t)$ can change from time to time. The permutation of the columns of $\mathbf{o}_i$ is also dynamic. Denote the weight matrix as $W \in \mathbb{R}^{p \times n}$ and the element-wise ReLU activation function as $\sigma(\cdot)$. Define the row-wise max pooling operation as $\mathrm{RowMax}(\cdot)$, which takes a matrix as input and outputs the maximum value of each row. Consider the following mapping $\rho : \mathbb{R}^{n \times N_i(t)} \mapsto \mathbb{R}^p$ formulated as

$$\rho(\mathbf{o}_i) = \mathrm{RowMax}(\sigma(W\mathbf{o}_i)), \tag{3.12}$$

where ρ maps a matrix $\mathbf{o}_i$ whose column has dynamic dimension and permutation to a fixed length feature vector $\rho(\mathbf{o}_i) \in \mathbb{R}^p$. The dimension of $\rho(\mathbf{o}_i)$

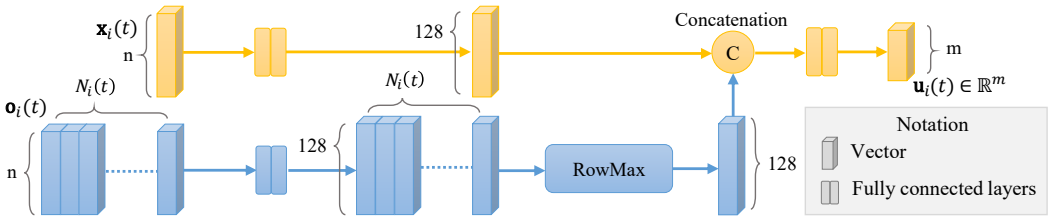


Figure 3-1: Neural network architecture of the control policy. The blue part indicates the quantity-permutation invariant observation encoder, which maps $\mathbf{o}_i(t) \in \mathbb{R}^{n \times N_i(t)}$ with time-varying dimension to a fixed length vector. The network takes the state $\mathbf{x}_i$ and local observation $\mathbf{o}_i$ as input to compute a control action $\mathbf{u}_i$. The neural network of the decentralized CBF h_i has a similar architecture, except that the output is a scalar.

remains the same even if the number of columns of $\mathbf{o}_i(t)$, which is $N_i(t)$, change over time. The network architecture of the control policy is shown in Figure 3-1, which uses the RowMax$(\cdot)$ operation. The network of the control barrier function is similar, except that the output is a scalar instead of a vector.

3.4.3 Spontaneous Online Policy Refinement

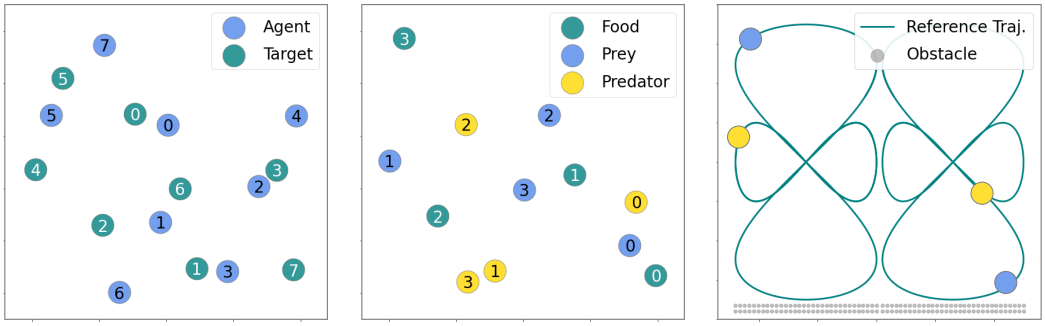
We propose a spontaneous online policy refinement approach that produces even safer control policies in testing than the neural network has actually learned during training. When the model dynamics or environment settings are too complex and exceed the capability of the control policy, the decentralized CBF conditions can be violated at some points along the trajectories. Thanks to the control barrier function jointly learned with the control policy, we are able to refine the control input $\mathbf{u}_i$ online by minimizing the violation of the decentralized CBF conditions. That is, the learned CBF can serve as a guidance on generating updated $\mathbf{u}_i$ in unseen scenarios to guarantee safety. This is also a standard technique used in (non-learning) CBF control where the CBF h_i is usually computed first using optimization methods like Sum-of-

Squares, then the control inputs $\mathbf{u}_i$ are computed online using h_i by solving quadratic programming problems [101, 7]. In the experiments, we also study the effects of such an online policy refinement step.

Given the state $\mathbf{x}_i$, local observation $\mathbf{o}_i$, and action $\mathbf{u}_i$ computed by the control policy, consider the scenario where the third CBF condition is violated, which means $\nabla_{\mathbf{x}} h_i \cdot \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i) + \nabla_{\mathbf{o}} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i) < 0$ when $h_i \geq 0$. Let $\mathbf{e}_i \in \mathbb{R}^m$ be an increment of the action $\mathbf{u}_i$. Define $\phi(\mathbf{e}_i) : \mathbb{R}^m \mapsto \mathbb{R}$ as

$$\phi(\mathbf{e}_i) = \max\Big(0, -\nabla_{\mathbf{x}} h_i \cdot \mathbf{f}_i(\mathbf{x}_i, \mathbf{u}_i + \mathbf{e}_i) - \nabla_{\mathbf{o}} h_i \cdot \dot{\mathbf{o}}_i - \alpha(h_i)\Big) + \mu ||\mathbf{e}_i||_2^2. \quad (3.13)$$

If the first term on the right side of Equation (3.13) is 0, then the third CBF condition is satisfied. We can enforce the satisfaction in every timestep of testing (after $\mathbf{u}_i$ is given by the neural network controller) by finding a $\mathbf{e}_i$ that minimizes $\phi(\mathbf{e}_i)$. μ is a regularization factor that punishes large $\mathbf{e}_i$. We set $\mu = 1$ in implementation and observed that in our experiment, a fixed μ is sufficient to make sure the $||\mathbf{u}_i + \mathbf{e}_i||_2$ do not exceed the constraint on control input bound. When evaluating on new scenarios and the constraints is violated, one can dynamically increase μ to strengthen the penalty. For every timestep during testing, we initialize $\mathbf{e}_i$ to zero and check the value of $\phi(\mathbf{e}_i)$. $\phi(\mathbf{e}_i) > 0$ indicates that the control policy is not good enough to satisfy the decentralized CBF conditions. Then we iteratively refine $\mathbf{e}_i$ by $\mathbf{e}_i = \mathbf{e}_i - \nabla_{\mathbf{e}} \phi(\mathbf{e}_i)$ until $\phi(\mathbf{e}_i) - \mu ||\mathbf{e}_i||_2^2 = 0$ or the maximum allowed iteration is exceeded. The final control input is $\mathbf{u}_i = \mathbf{u}_i + \mathbf{e}_i$. Such a refinement can flexibly refine the control input to satisfy the decentralized CBF conditions as much as possible.



(a) Navigation (b) Predator-Prey (c) Nested Rings

Figure 3-2: Illustrations of the 2D environments used in the experiments. The *Navigation* and *Predator-Prey* environments are adopted from the multi-agent particle environment [55]. The *Nested-Rings* environment is adopted from [75].

3.5 Experiments

Baseline Approaches. The baseline approaches we compare with include: MAMPS [106], PIC [53] and MADDPG [55]. For the drone tasks, we also compare with model-based planning method S2M2 [13]. A brief description of each method is as follows. MAMPS leverages the model dynamics to iteratively switch to safe control policies when the learned policies are unsafe. PIC proposes the permutation-invariant critic to enhance the performance of multi-agent RL. We incorporate the safety reward to its reward function and denote this safe version of PIC as PIC-Safe. The safety reward is -1 when the agent enters the dangerous set. MADDPG is a pioneering work on multi-agent RL, and MADDPG-Safe is obtained by adding the safety reward to the reward function that is similar to PIC-Safe. S2M2 is a state-of-the-art model-based multi-agent safe motion planner. When directly planning all agents fails, S2M2 evenly divides the agent group to smaller partitions for replanning until paths that are collision-free for each partition are found. The agents then follow the generated paths using PID or LQR controllers.

For each task, the environment model is the same for all the methods. The

exact model dynamics are visible to model-based methods including MAMPS, S2M2 and our methods, and invisible to the model-free MADDPG and PIC. Since the model-free methods do not have access to model dynamics but instead the simulators, they are more data-demanding. The number of state-observation pairs to train MADDPG and PIC is 10^3 times more than that of model-based learning methods to make sure they converge to their best performance. When training the RL-based methods, the control action computed by LQR for goal-reaching is also fed to the agent as one of the inputs to the actor network. So the RL agents can learn to use LQR as a reference for goal-reaching.

Evaluation Criteria. Since the primal focus of this chapter is the safety of multi-agent systems, we use the *safety rate* as a criteria when evaluating the methods. The safety rate is calculated as $\frac{1}{N}\Sigma_{i=1}^{N}\mathbb{E}_{t\in T}\left[\mathbb{I}((s_i(t), o_i(t)) \in \mathcal{X}_s)\right]$ where $\mathbb{I}(\cdot)$ is the indicator function that is 1 when its argument is true or 0 otherwise. The observation o_i contains the states of other agents within the observation radius, which is 10 times the safe distance. The safe distance is set to be the diagonal length of the bounding box of the agent. In addition to the safety rate, we also calculate the *average reward* that considers how good the task is accomplished. The agent is given a +10 reward if it reaches the goal and a -1 reward if it enters the dangerous set. Note that the agent might enter the dangerous set for many times before reaching the goal. The upper-bound of the total reward for an agent is +10, which is attained when the agent successfully reaches the goal and always stays in the safe set.

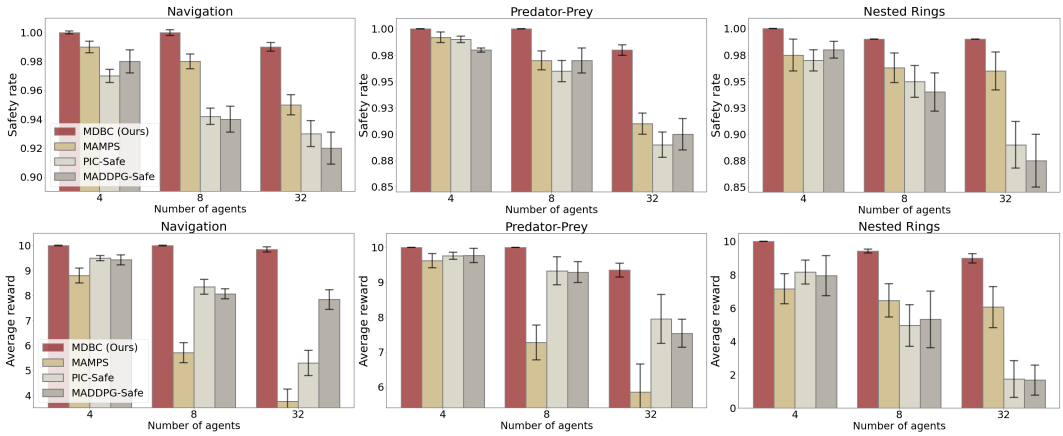


Figure 3-3: Safety rate and reward in the 2D tasks. Results are taken after each method converged and are averaged over 10 independent trials.

3.5.1 Experiments on Ground Robots

We consider three tasks illustrated in Figure 3-2. In the Navigation task, each agent starts from a random location and aims to reach a random goal. In the Predator-Prey task, the preys aim to gather the food while avoid being caught by the predators chasing the preys. We only consider the safety of preys but not predators. In the Nested-Rings task, the agents aim to follow the reference trajectories while avoid collision. In order for the RL-based agents to follow the rings trajectory, we also give the agents a negative reward proportional to the distance to the nearest point on the rings. When adding more agents to an environment, we will also enlarge the area of the environment to ensure the overall density of agents remains similar.

Figure 3-3 demonstrates that when the number of agents grows (e.g., 32 agents), our approach can still maintain a high safety rate and average reward, while other methods have much worse performance. However, our method still cannot guarantee that the agents are 100% safe. The failure is mainly because we cannot make sure the decentralized CBF conditions are satisfied for every

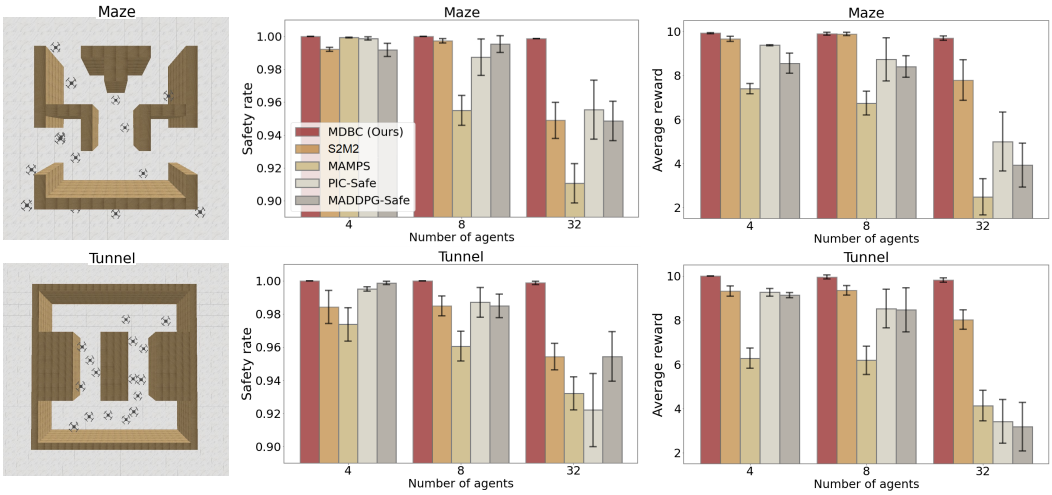


Figure 3-4: Environments and results of 3D tasks. In Maze and Tunnel, the initial and target locations of each drone are randomly chosen. The drones start from the initial locations and aim to reach the targets without collision. The results are taken after each method converged and are averaged over 10 independent trials.

state-observation pair in testing, even if they are satisfied on all training samples due to the generalization error. We also show the generalization capability of our method with up to 1024 in the appendix and also visualization results in the supplementary materials.

3.5.2 Experiments on Drones

We experiment with 3D drones whose dynamics are even more complex. Figure 3-4 and 3-5 demonstrate the environments and the results of each approach. Similar to the results of ground robots, when there are numerous agents (e.g., 32 agents), our method can still maintain a high reward and safety rate, while other methods have worse performance. Figure 3-6 shows the generalization capability of our method across different environments and number of agents. For each experiment, we train 8 agents during training, but test with up to

Figure 3-5: Illustration of the Maze environment with 1024 drones.

1024 agents. The extra agents are added by copying the neural network parameters of the trained 8 agents. Results show that our method has remarkable generalization capability to diverse scenarios. Another related work [17] can also handle the safe multi-drone control problem via CBF, but their CBF is handcrafted and based on quadratic programming to solve the u_i. Their paper only reported the results on two agents, and for 32 agents it would take more than 70 hours for a single run of evaluation (7000 steps and 36 seconds per step). By contrast, our method only takes $\sim$ 200s for a single run of evaluation with 32 agents, showing a significant advantage in computational efficiency.

3.6 Summary

We have presented a novel approach of learning safe multi-agent control via jointly learning the decentralized control barrier functions as safety certificates.

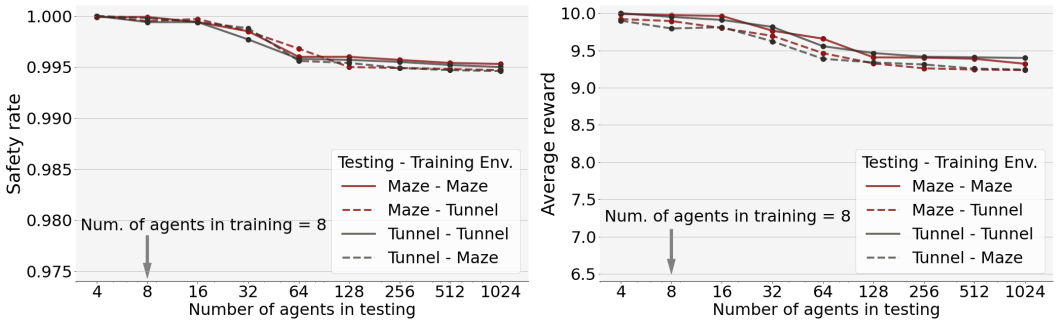


Figure 3-6: Generalization capability of our method in the 3D tasks. Our method can be trained with 8 agents in one environment and generalize to 1024 agents in another environment in testing.

We provide the theoretical generalization bound, as well as the effective techniques to realize the learning framework in practice. Experiments show that our method significantly outperforms previous methods by being able to scale to an arbitrary number of agents, and demonstrates remarkable generalization capabilities to unseen and complex multi-agent environments.

Chapter 4

Convergent Trajectory Tracking with Contraction Metrics

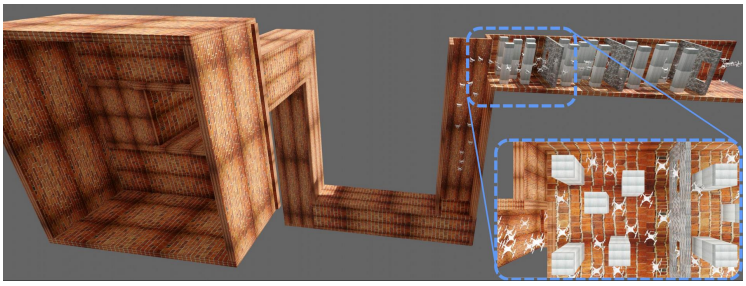
4.1 Introduction

Both safety and goal-reaching are crucial in multi-agent control design. While the previous chapter mainly considers safety, we will incorporate goal-reaching in this chapter. In terms of goal-reaching, the control contraction metric (CCM) [60] is an established certificate to guarantee tracking arbitrary reference trajectories with guaranteed bound on tracking error, where the reference trajectories lead to the desired goals. We combine the CBF and CCM in a unified framework, which can produce safe and robust controllers for multi-agent dynamical systems to perform reach-avoid tasks and can scale to a very large number of agents.

Our framework consists of three major components: a single-agent planner, a tracking controller certified by a learned CCM, and a learned decentralized CBF. First, the single-agent planner plans for each agent a reference trajectory connecting it to its next destination. Then, the controllers learned with

(a) Auto cargo transportation for port containers

(b) Drone delivery in a city



(c) Drone fleet control in an obstacle-rich tunnel

Figure 4-1: Three simulation environments considered in this chapter.

CCM can guarantee that each agent can track the planned reference to reach its destination under disturbances, and the control input is further refined by the learned decentralized CBF to make sure that the agents will not collide with each other and other obstacles. We prove that such a certified controller can guarantee safety and goal-reaching for multi-agent reach-avoid tasks (Theorem 1).

We propose a novel learning framework that *jointly* learns multi-agent control policies and control certificates (i.e., CCM and CBF) from data, which can be implemented in a *decentralized* fashion and hence scalable to an arbitrary number of agents. We present the loss functions for the joint-learning framework, as well as the neural network architecture that enables the agents to handle complex multi-agent environments when the local topology constantly changes. The decentralized CBF can be seen as a contract among agents and allows agents to learn a mutual agreement with each other on how to avoid collisions. Once such a controller is achieved through the joint-learning framework, it can be applied to an arbitrary number of agents in scenarios that are different from the training scenarios, which resolves the fundamental scalability issue in multi-agent control. We also present several techniques that make the learning framework more effective and scalable for practical nonlinear and nonholonomic multi-agent systems.

We conduct experiments in the simulation environments illustrated in Figure 4-1, which include three multi-agent control problems with complex nonholonomic dynamics: 1) controlling multiple forklifts to transport cargo for port containers, 2) controlling multiple drones to deliver packages in a city, and 3) controlling a drone fleet to fly through an obstacle-rich tunnel. The agents are required to reach a sequence of goal locations and avoid collision with each other. Comparing to state-of-the-art multi-agent planning [13] and

safe reinforcement learning [106, 53, 55] methods, the experimental results of our method are truly promising. Our approach can handle more than 1000 agents under limited computational resources and can avoid 96%-100% collision among agents compared to the baseline. It is worth noting that when the number of agents exponentially increases, our approach exhibits almost no performance drop, while the performance of the compared methods drops drastically. For example, in the cargo transportation task, as the number of agents increases from 4 to 256, the avoided collision using method [13] drops from 100% to 22%. On the contrary, using our method, the number only decreases from 100% to 99.7%. Although maintaining a high safety rate, our method is not conservative and only takes the agents 3%-17% more travel time than the baseline method to reach the goal. Furthermore, our method demonstrates promising generalization capability. Trained with 8 agents, the learned controller and CBF can generalize to testing scenarios with 1024 agents, even in a different environment. 1024 is not the limit of our approach and even more agents can be handled given sufficient computational resource, since the computation for each agent is parallel and fully decentralized.

Some parts of this chapter have been demonstrated in [85] and [69]. In [85] we presented the jointly learning framework for a tracking controller and a CCM, and in [69] we presented the jointly learning framework for a safe controller and a decentralized CBF. However, those controllers were developed separately and only used for simple tracking or collision-avoidance purposes under relatively simple experimental settings. In this chapter, we study the composition of these two control certificates to finish complex reach-avoid tasks where each agent has a (possibly infinite) sequence of goals to visit, and they need to avoid collisions with each other at run time. We develop completely new experiments and simulation environments that better capture the practi-

cal use of our framework. Last but not least, we provide a more comprehensive literature review on controller synthesis for trajectory tracking and multi-agent safety.

The rest of the chapter is organized as follows. In Section 4.3, we will define the multi-agent reach-avoid problem that we study, and introduce the basics of CCM and multi-agent decentralized CBF. In Section 4.4, we will provide an overview of the proposed framework, explain how the CCM and decentralized CBF are combined in the multi-agent setting, and give a theoretical analysis of our method. In Section 4.5, we will elaborate on the joint-learning of the controllers with their CCM and CBF certificates. We will derive the loss functions used in training, and present a novel neural network architecture that handles time-varying quantity and permutation of neighboring agents. In Section 4.6, we will conduct comprehensive experiments in the three simulation environments illustrated in Figure 4-1.

4.2 Related Work

Controller Synthesis with Bounded Tracking Error. The idea of synthesizing a controller and bounding the tracking error meanwhile through precomputation has gained increasing popularity. Control Lyapunov functions can be used for this purpose [70]. Extensions to richer temporal specifications in the presence of noise has also been proposed [41]. Learning-based CLF has also been studied in some recent works [20, 44, 73]. However, in order to use CLF for synthesizing controller to track reference trajectories, the original dynamics have to be converted into error dynamics depending on the reference trajectory, which limits the diversity of available reference trajectories. Similarly, funnel library methods have been studied in [58, 59], where tracking

controllers for a fixed set of reference trajectories and the corresponding tracking error are computed offline. Tube Model Predicative Control [47, 62, 29] is another class of related techniques, where a tracking controller is computed such that the actual trajectory remains in a tube centered at the planned MPC nominal trajectory in the presence of bounded disturbances. As for non-linear systems, linearization and Lipschitz-continuity-based reachability analysis are used [46, 103, 61], but this is too conservative for motion planning in limited free space.

Control Contraction Metrics for Trajectory Tracking. Contraction analysis [54] is another series of methods for analyzing the incremental stability of systems. Recently, it has been extended to controlled systems in [60] and thus enabled tracking control synthesis for arbitrary reference trajectories with guaranteed bound on tracking error. The most challenging part of contraction analysis is the search for a valid contraction metric which entails solving Linear Matrix Inequalities (LMIs). In [8], the authors proposed to solve this feasibility problem with Sum-of-Squares (SoS) programming. In [81], the authors extended the SoS-based method to controlled systems, i.e., search for a *control* contraction metric (CCM) instead of an ordinary one, and proposed a more general method for synthesizing control given a valid CCM. However, in order to apply SoS-based methods, the dynamics of the system have to be represented by polynomials or can be approximated by polynomials. Furthermore, the method proposed in [81] relies on an assumption on the structure which encodes the controllability of the system.

4.3 Preliminaries and Problem Statement

We first formalize the multi-agent safe-control problem, then introduce control contraction metrics and decentralized control barrier functions as certificates for contraction and safety respectively.

4.3.1 Multi-agent Safe Control for Reach-avoid Problems

We assume each agent follows the control-affine dynamics:

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t)) + B_i(\mathbf{x}_i(t))\mathbf{u}_i(t) + \mathbf{d}_i(t), \tag{4.1}$$

where $\mathbf{f}_i : \mathcal{X}_i \to \mathbb{R}^n$ and $B_i : \mathcal{X}_i \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous, and $\mathbf{d}_i(t)$ is the disturbance belonging to a compact disturbance set $\mathcal{D}_i \subseteq \mathbb{R}^n$. The dynamics in 4.1 is slightly different from Chapter 2.1 as we add the disturbance and assume the system is control-affine.

Definition 1 (Multi-agent Reach-avoid Problem)
Consider a multi-agent system modeled as in Chapter 2.1, where each agent can obtain a local observation of its surroundings $\mathbf{o}_i(t)$ at any time. The multi-agent reach-avoid problem is to find the control input $\mathbf{u}_i$ for each agent i such that (i) Each agent i can sequentially visit the goal states $\{\mathbf{p}_i^1, \mathbf{p}_i^2, \cdots\}$. That is, there exist $0 < t_1 < t_2 < \cdots$ such that

$$\|\mathbf{x}_i(t_j) - \mathbf{p}_i^j\|_2 \leq \eta,\ j = 1, 2, \cdots; \tag{4.2}$$

and (ii) The distance from agent i to its nearest obstacle or other agent is greater than a safety threshold:

$$r(\mathbf{x}_i(t), \mathbf{o}_i(t)) \geq \kappa,\ \forall t > 0, \tag{4.3}$$

and each agent i also satisfy $g_i(\mathbf{x}_i(t)) \geq 0$, $\forall t > 0$, where η, κ are positive constants and $g_i : \mathcal{X}_i \to \mathbb{R}$ is a function that defines additional safety constraints.

The above definition for multi-agent reach-avoid problem can describe many practical robotics applications, including all three use cases in Figure 4-1. The problem of finding the sequential goals $\{\mathbf{p}_i^1, \mathbf{p}_i^2, \cdots\}$ for each agent i can be solved using path planners and is out of the scope of this chapter.

A straightforward solution to the multi-agent reach-avoid problem is to use the multi-agent trajectory planning algorithms [13, 31, 57, 80], which output collision-free trajectories connecting agents to the goal locations. However, there are two facts that make these algorithms impractical for large-scale multi-agent safe control in our settings. First, those centralized planning methods cannot handle a large and potentially changing number of agents with complex model dynamics due to their high computational complexity. Second, the disturbances could cause tracking errors between the actual trajectories and the planned ones, therefore in practice even perfectly planned trajectories could lead to collisions. Therefore, we choose to combine simple single-agent planning with decentralized safe control to mitigate the scalability issue while keeping agents safe. The single-agent planner only needs to plan a reference path connecting each agent to their goals while avoiding some static obstacles, and any off-the-shelf single-agent planner can be used to do so. Our primary focus is to design controllers that allow each agent to track its own reference trajectory and avoid collisions with each other at run time. These controllers will be found through control theoretical approaches based on contraction metrics and barrier functions.

4.3.2 Control Contraction Metrics

Contraction theory [54] analyzes the incremental stability of a system by considering the evolution of the distance between any pairs of trajectories. Let us first consider a time-invariant autonomous system of the form $\dot{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{x}_i(t))$. For a trajectory $\mathbf{x}_i(t)$ of the system, assume that $\mathbf{x}_i(t) + \delta_{\mathbf{x}_i}(t)$ is also a valid trajectory, where $\delta_{\mathbf{x}_i}$ is an infinitesimal displacement. Then, the evolution of $\delta_{\mathbf{x}_i}$ is dictated by a linear time varying (LTV) system: $\dot{\delta}_{\mathbf{x}_i} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}(\mathbf{x}_i)\delta_{\mathbf{x}_i}$, which is called the linearization of the original system along the trajectory $\mathbf{x}_i(t)$. Thus, the dynamics of the squared distance $\delta_{\mathbf{x}_i}^\intercal \delta_{\mathbf{x}_i}$ is given by $\frac{d}{dt}(\delta_{\mathbf{x}_i}^\intercal \delta_{\mathbf{x}_i}) = 2\delta_{\mathbf{x}_i}^\intercal \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}\delta_{\mathbf{x}_i} = \delta_{\mathbf{x}_i}^\intercal \widehat{\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}}\delta_{\mathbf{x}_i}$. Recall that $\widehat{\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} + \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}^\intercal$. If $\widehat{\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}}$ is uniformly negative definite, i.e., there exists a constant $\lambda > 0$ such that for all $\mathbf{x}_i$, $\widehat{\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}} \preceq -\lambda \mathbf{I}$, then $\delta_{\mathbf{x}_i}^\intercal \delta_{\mathbf{x}_i}$ converges to zero exponentially at rate λ. Hence, all trajectories of this system will converge to a common trajectory [54]. Such a system is referred to be *contracting*.

The above analysis can be generalized by introducing a *contraction metric* $M_i : \mathbb{R}^n \mapsto \mathbb{R}^{n\times n}$, which is a smooth function and satisfies $M_i(\mathbf{x}_i) \succ 0$ for all $\mathbf{x}_i$. Then, $\delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}$ can be interpreted as a Riemannian squared length. Since $M_i(\mathbf{x}_i) \succ 0$ for all $\mathbf{x}_i$, if $\delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}$ converges to 0 exponentially, then $\delta_{\mathbf{x}_i}^\intercal \delta_{\mathbf{x}_i}$ also converges to 0 exponentially, and the system is contracting. The converse is also true. As shown in [54], if a system is contracting, then there exists a contraction metric $M_i(\mathbf{x}_i)$ and a constant $\lambda > 0$ such that $\frac{d}{dt}(\delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}) < -\lambda \delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}$ for all $\mathbf{x}_i$ and $\delta_{\mathbf{x}_i}$.

Contraction theory can be further extended to control-affine systems. Consider the unperturbed version of Equation (4.1) (i.e., with $\mathbf{d}_i \equiv 0$). The dynamics of $\delta_{\mathbf{x}_i}$ is then given by $\dot{\delta}_{\mathbf{x}_i} = A_i(\mathbf{x}_i, \mathbf{u}_i)\delta_{\mathbf{x}_i} + B_i(\mathbf{x}_i)\delta_{\mathbf{u}_i}$, where $A_i(\mathbf{x}_i, \mathbf{u}_i) := \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} + \sum_{j=1}^m \mathbf{u}_i^j \frac{\partial \mathbf{b}_i^j}{\partial \mathbf{x}_i}$, $\mathbf{b}_i^j$ is the j-th column of B_i, $\mathbf{u}_i^j$ is the j-th element of $\mathbf{u}_i$, and $\delta_{\mathbf{u}_i}$ is an infinitesimal difference of $\mathbf{u}_i$. We then consider the reference track-

ing problem. A trajectory $\mathbf{x}_i^* : \mathbb{R}^{\geq 0} \mapsto \mathcal{X}_i$ is called a reference if there exists $\mathbf{u}_i^* : \mathbb{R}^{\geq 0} \mapsto \mathcal{U}_i$ such that $\mathbf{x}_i^*$ and $\mathbf{u}_i^*$ solve the differential equation in (4.1) with $\mathbf{d}_i \equiv 0$. We aim to find a tracking controller that is certified to be able to track any reference $\mathbf{x}_i^*(t)$ with the corresponding input $\mathbf{u}_i^*(t)$. A fundamental theorem in the CCM theory states that if there exists a metric $M_i(\mathbf{x}_i)$ such that the following conditions hold for all $\mathbf{x}_i \in \mathcal{X}_i$ and some $\lambda > 0$,

$$B_{i\perp}^\intercal \left(-\partial_{\mathbf{f}_i} W_i(\mathbf{x}_i) + \widehat{\frac{\partial \mathbf{f}_i(\mathbf{x}_i)}{\partial \mathbf{x}_i} W_i(\mathbf{x}_i)} + 2\lambda W_i(\mathbf{x}_i) \right) B_{i\perp} \preceq 0, \tag{4.4}$$

$$B_{i\perp}^\intercal \left(\partial_{b_i} W_i(\mathbf{x}_i) - \widehat{\frac{\partial b_i^j(\mathbf{x}_i)}{\partial \mathbf{x}_i} W_i(\mathbf{x}_i)} \right) B_{i\perp} = 0, \ j = 1, \ldots, m, \tag{4.5}$$

where $B_{i\perp}(\mathbf{x}_i)$ is an annihilator matrix of $B_i(\mathbf{x}_i)$ satisfying $B_{i\perp}^\intercal B_i = 0$, and $W_i(\mathbf{x}_i) = M_i(\mathbf{x}_i)^{-1}$ is the dual metric, then there exists a tracking controller $\mathbf{u}_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*)$ such that the closed-loop system controlled by $\mathbf{u}_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*)$ is contracting with rate λ under metric $M_i(\mathbf{x}_i)$ [60]. This also means $\mathbf{x}_i(t)$ converges to $\mathbf{x}_i^*(t)$ exponentially fast. If we can find such a controller for each agent, then we can ensure that each agent converges to its own desired nominal trajectory $\mathbf{x}_i^*(t)$. However, note that the above conditions (4.4) and (4.5) are controller-independent, i.e., $\mathbf{u}_i$ does not appear in the conditions. Although these conditions imply the existence of such a controller, computation of it is still hard and computationally expensive as shown in [81]. Thus, we proposed to learn such a controller directly. To this end, controller-dependent conditions are needed. Plugging the controller into the system in Equation (4.1) with $\mathbf{d}_i \equiv 0$, the dynamics of the generalized squared length of the virtual displacement $\delta_{\mathbf{x}_i}$ under metric $M_i(\mathbf{x}_i) = W_i(\mathbf{x}_i)^{-1}$ is given by

$$\frac{d}{dt}(\delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}) = \delta_{\mathbf{x}_i}^\intercal (\dot{M}_i + \widehat{M_i(A_i + B_i K_i)})\delta_{\mathbf{x}_i}, \tag{4.6}$$

where $K_i = \frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_i}$, and $\dot{M}_i = \sum_{j=1}^{n} \frac{\partial M_i}{\partial \mathbf{u}_i^j} \dot{\mathbf{u}}_i^j$. Here, $\mathbf{x}_i^j$ denotes the j^{th} element of the state vector of agent i. Now, we propose the following condition: there exists $\lambda > 0$ such that for all $\mathbf{x}_i \in \mathcal{X}_i$, $\mathbf{x}_i^* \in \mathcal{X}_i$ and $\mathbf{u}_i^* \in \mathcal{U}_i$,

$$\dot{M}_i + \widehat{M_i(A_i + B_i K_i)} + 2\lambda M_i \preceq 0. \tag{4.7}$$

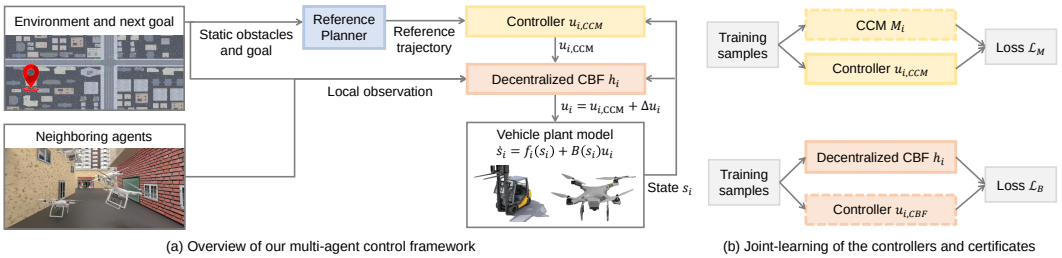
Now assume that the above condition is satisfied. Then, given an arbitrary reference $\mathbf{x}_i^*(t)$, the closed-loop system is contracting. Thus, starting from any initial state, the trajectory converges to a "common" trajectory. Furthermore, assume that the controller satisfies that if $\mathbf{x}_i = \mathbf{x}_i^*$, then $\mathbf{u}_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*) = \mathbf{u}_i^*$. Then the reference $\mathbf{x}_i^*(t)$ itself is a valid trajectory of the closed-loop system, and thus the "common" trajectory is indeed $\mathbf{x}_i^*(t)$. In other words, all the trajectories converge to the reference. This is formalized in the following proposition.

Proposition 2

If condition (4.7) *holds and two positive scalars* $\overline{m} \geq \underline{m} > 0$ *satisfy* $\underline{m}\mathbf{I} \preceq M_i(\mathbf{x}_i) \preceq \overline{m}\mathbf{I}$ *for all* $\mathbf{x}_i$*, and in* (4.1) *the disturbance is bounded as* $\|\mathbf{d}_i(t)\|_2 \leq \epsilon$ *for all* t*, then for an arbitrary reference* $\mathbf{x}_i^* : \mathbb{R}^{\geq 0} \mapsto \mathcal{S}_i$ *and an arbitrary initial condition* $\mathbf{x}_i(0)$*, the difference between the reference and the actual trajectory is bounded as:*

$$\|\mathbf{x}_i(t) - \mathbf{x}_i^*(t)\|_2 \leq \sqrt{\frac{\overline{m}}{\underline{m}}}\, e^{-\lambda t} \|\mathbf{x}_i(0) - \mathbf{x}_i^*(0)\|_2 + \sqrt{\frac{\overline{m}}{\underline{m}}} \cdot \frac{\epsilon}{\lambda}(1 - e^{-\lambda t}). \tag{4.8}$$

Proof. Since $\forall \mathbf{x}_i$, $M_i(\mathbf{x}_i) \succ 0$, there exists $\Theta_i(\mathbf{x}_i)$ such that $M_i(\mathbf{x}_i) = \Theta_i(\mathbf{x}_i)^\intercal \Theta_i(\mathbf{x}_i)$ for all $\mathbf{x}_i$. Now, let us consider the smallest path integral between $\mathbf{x}_i(t)$ and $\mathbf{x}_i^*(t)$ w.r.t. metric $M_i(\mathbf{x}_i)$ and denote it by $R(t) = \int_{\mathbf{x}_i(t)}^{\mathbf{x}_i^*(t)} \sqrt{\delta_{\mathbf{x}_i}^\intercal M_i(\mathbf{x}_i) \delta_{\mathbf{x}_i}} = \int_{\mathbf{x}_i(t)}^{\mathbf{x}_i^*(t)} \|\Theta_i(\mathbf{x}_i)\delta_{\mathbf{x}_i}\|_2$. Then differentiating $R(t)$ w.r.t. t yields $\dot{R} \leq -\lambda R +$



(a) Overview of our multi-agent control framework

(b) Joint-learning of the controllers and certificates

Figure 4-2: Method overview. (a) The proposed multi-agent control framework consists of three components: the reference planner (blue) that gives each agent i its reference trajectory (not necessarily collision-free), the tracking controller $\mathbf{u}_{i,CCM}$ (orange) certified by CCM and the decentralized CBF (red), which computes the final control input $\Delta\mathbf{u}_i$ for each agent i such that the whole multi-agent system is collision-free and each agent can track its reference trajectory with a bounded error. The control is decentralized and we only show the pipeline for agent i. (b) The controllers and corresponding certificates are represented by neural networks and are jointly learned during training.

$\|\Theta_i \mathbf{d}_i\|_2 \leq -\lambda R + \epsilon\sqrt{\overline{m}}$. (see [54], pp. 11 (vii)) By comparison lemma, $R(t)$ is bounded as $R(t) \leq R(0)e^{-\lambda t} + \frac{\epsilon}{\lambda}\sqrt{\overline{m}}(1 - e^{-\lambda t})$. Since Θ_i is uniformly bounded as $\Theta_i(x) \succeq \sqrt{\underline{m}}\mathbf{I}$, we have $R(t) \geq \sqrt{\underline{m}}\|\mathbf{x}_i(t) - \mathbf{x}_i^*(t)\|_2$, which follows from the fact that the smallest path integral w.r.t. a constant metric is the path integral along the straight line. Similarly we have that $R(0) \leq \sqrt{\overline{m}}\|\mathbf{x}_i(0) - \mathbf{x}_i^*(0)\|_2$. Combining these inequalities yields the proposition. □

4.4 Proposed Control Framework

In this section, we present the entire learning-based control framework and the theoretical guarantees it can provide, assuming we have a certified tracking controller and a decentralized CBF. In Section 4.5 we will introduce techniques on learning those controllers and certificates from data. An overview of our method is provided in Figure 4-2. Since the controllers are decentralized and each agent executes its own controller individually, we only demonstrate the

control pipeline for a single agent i. The pipeline structure is the same for all the agents, even if their dynamics are different.

There are three components in our framework: the single-agent reference planner, the controller certified by CCM for tracking the reference trajectory, and the decentralized CBF to refine the tracking controller and ensure safety of the multi-agent system. As the first component, the reference trajectory planner takes the static obstacles and the goal as input and outputs the reference trajectory for each individual agent. At this stage, we do not require the reference trajectories to avoid collision with other agents, and the planners are decentralized for each agent. Therefore, any single-agent trajectory planning algorithms can be used as a reference planner. In our implementation, we use the FACTEST tool [27], which can output piecewise feasible reference trajectories in a very short time (usually within seconds), but other methods such as RRT [48] are also applicable. The single-agent planners are not our focus in this chapter and therefore we omit the detailed discussion. The key feature of the first component is that we do not require a multi-agent trajectory planner that considers the collision avoidance among agents. As we discussed in Section 4.3.1, multi-agent planners suffer from high computational complexity and are extremely difficult to scale up to a large number of agents.

As the second component, we propose a learning-based tracking controller $\mathbf{u}_{i,CCM}$ such that the closed-loop system has a corresponding contraction metric M_i which guarantees that each agent can track any feasible reference trajectory with an exponential convergence rate. Existing control algorithms based on CCM [81, 82, 94] usually first synthesize the CCM, and then compute the control input separately by computing geodesics of the CCM. However, geodesics cannot be computed exactly, and approximation algorithms are very time consuming. Moreover, those methods also rely on some assumptions

about the structure of the dynamics. In Section 4.5, we will present a learning-based method that can jointly synthesize the CCM and the tracking controller for a general class of control-affine systems. Such a neural controller can be evaluated in real time and still enjoys the strong convergence guarantees.

As the third component, we propose to adjust the tracking controller $\mathbf{u}_{i,CCM}$ so it can avoid collisions in the multi-agent system and satisfy other safety requirements. This can be achieved using the decentralized CBF h_i as introduced in (3.1). The final control policy $\mathbf{u}_i$ should always make sure the closed-loop system satisfy the CBF conditions. Previous works using decentralized CBF typically need to handcraft the CBF [16, 17], which can be extremely difficult for general nonlinear and nonholonomic dynamics. In Section 4.5 we present a learning-based method to directly learn the decentralized CBF in a data-driven way. Note in this case, while h_i is obtained jointly with a learned safe controller $\mathbf{u}_{i,CBF}$, we do not use $\mathbf{u}_{i,CBF}$ in testing but rather to ensure that the set of controllers that satisfy the CBF conditions is not empty. Since the CBF learning framework is independent from the CCM learning framework, the jointly learned neural $\mathbf{u}_{i,CBF}$ may not be able to tracking the reference trajectory. Therefore, we only need h_i to compute the final control input $\mathbf{u}_i$, which ensures that collision among agents can be avoided at run time, and each agent is able to track its reference trajectory with a bounded error. Note that our control framework is *decentralized*, which means each agent has its own controller. In order to scale up to a large number of agents, we cannot synthesis a centralized controller that simultaneously computes the control input for all agents. The input and output space of a centralized controller will grow exponentially with the number of agents and can eventually make the problem intractable.

The combination of the second and third components is critical to keeping

the multi-agent system safe and enabling each agent to reach its goal by tracking its reference trajectory. Since the reference trajectories of different agents may collide, the decentralized CBFs will take effect and enable the agents to avoid each other by slightly deviating from the reference trajectories. We will prove that using our composed CCM and CBF framework, the agents' deviations from the reference trajectory are bounded under some assumptions on the bounds of the controllers and certificates. In the following, we first present how the second and third components are combined mathematically. Given the tracking controller $\mathbf{u}_{i,CCM}$ satisfying (4.7) and the decentralized CBF h_i satisfying (3.1), we seek to compute a $\Delta\mathbf{u}_i$ such that $\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i \in \mathcal{U}_i$ satisfies the third condition in (3.1). For an agent model in (4.1), we assume that $\nabla_{\mathbf{x}_i} h_i \cdot \mathbf{d}_i$ is bounded as $\|\nabla_{\mathbf{x}_i} h_i \cdot \mathbf{d}_i\|_2 \leq \epsilon_h$. We consider the following optimization problem for finding $\Delta\mathbf{u}_i$:

$$\begin{aligned} &\min_{\Delta\mathbf{u}_i} \|B_i \Delta\mathbf{u}_i\|_2^2 \\ &\text{s.t.} \quad \nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)(\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i)) \; + \\ &\qquad \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i(t) + \alpha\left(h_i\right) \geq \epsilon_h, \quad \forall (\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,h} \end{aligned} \tag{4.9}$$

where $\mathcal{S}_{i,h} := \{(\mathbf{x}_i, \mathbf{o}_i) \mid h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}$. The desired $\Delta\mathbf{u}_i$ can be found online by solving (4.9) using Quadratic Programming (QP). Theorem 1 shows that if $\|B_i \Delta\mathbf{u}_i\|_2$ is bounded, then the current trajectory of agent i will exponentially converge to its reference trajectory with a bounded tracking error, and the multi-agent system is safe under the control input $\mathbf{u}_i = \mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i$.

Theorem 1

For each agent i in the multi-agent system, given $\mathbf{u}_{i,CCM}, M_i$ satisfying (4.7), h_i satisfying (3.1), $\Delta\mathbf{u}_i$ the solution to (4.9), $\overline{m} \geq \underline{m} > 0$ satisfying $\underline{m}\mathbf{I} \preceq M_i(\mathbf{x}_i) \preceq \overline{m}\mathbf{I}$ for all $\mathbf{x}_i$, $\|B_i(\mathbf{x}_i(t))\Delta\mathbf{u}_i(t)\|_2 \leq \epsilon_u$ and $\|\mathbf{d}_i(t)\|_2 \leq \epsilon_d$ hold for all

$t > 0$. *Then under the control input* $\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i$, *the distance between the current trajectory* $\mathbf{x}_i(t)$ *and the reference trajectory* $\mathbf{x}_i^*(t)$ *is bounded as:*

$$\|\mathbf{x}_i(t) - \mathbf{x}_i^*(t)\|_2 \leq \sqrt{\frac{\overline{m}}{\underline{m}}}\, e^{-\lambda t}\|\mathbf{x}_i(0) - \mathbf{x}_i^*(0)\|_2 + \sqrt{\frac{\overline{m}}{\underline{m}}} \cdot \frac{\epsilon_u + \epsilon_d}{\lambda}(1 - e^{-\lambda t}). \quad (4.10)$$

Also, if for $\forall i \in [\![N]\!]$, $(\mathbf{x}_i(0), \mathbf{o}_i(0)) \in \{(\mathbf{x}_i, \mathbf{o}_i) \mid h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}$, *then:*

$$r(\mathbf{x}_i(t), \mathbf{o}_i(t)) \geq \kappa,\ g_i(\mathbf{x}_i(t)) \geq 0,\ \forall i = [\![N]\!],\ t > 0, \quad (4.11)$$

which means the minimum distance from an agent to its neighbouring agents and obstacles is greater than the safety threshold κ, *and the additional safety constraints specified by* $g_i(\mathbf{x}_i)$ *are satisfied. Thus the multi-agent system is safe.*

Proof. The model dynamics under the adjusted control input $\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i$ is $\dot{s}_i = \mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)\mathbf{u}_{i,CCM} + (B_i(\mathbf{x}_i)\Delta\mathbf{u}_i + \mathbf{d}_i(t))$. Note that $\|B_i(\mathbf{x}_i)\Delta\mathbf{u}_i + \mathbf{d}_i(t)\|_2 \leq \|B_i(\mathbf{x}_i)\Delta\mathbf{u}_i\|_2 + \|\mathbf{d}_i(t)\|_2 = \epsilon_u + \epsilon_d$. Based on the result of Proposition 2 and replacing ϵ with $\epsilon_u + \epsilon_d$, we can derive the inequality (4.10). Since $\Delta\mathbf{u}_i$ satisfies the inequality constraint in (4.9), we have $\nabla_{\mathbf{x}_i} h_i \cdot \dot{\mathbf{x}}_i + \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i) = \nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)(\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i) + \mathbf{d}_i(t)) + \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i) \geq \nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)(\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i)) - \|\nabla_{\mathbf{x}_i} h_i \cdot \mathbf{d}_i\|_2 + \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i) \geq \nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)(\mathbf{u}_{i,CCM} + \Delta\mathbf{u}_i)) - \epsilon_h + \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i) \geq 0$. Thus the decentralized CBF conditions in (3.1) are satisfied. By Proposition 1, the multi-agent system is safe. If the initial condition $(\mathbf{x}_i(0), \mathbf{o}_i(0)) \in \mathcal{S}_{i,0} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}$, $(\mathbf{x}_i, \mathbf{o}_i)$ will never enter the dangerous set $\mathcal{S}_{i,d}$, which is equivalent to (4.11). □

Note that $t = 0$ in Theorem 1 does not necessarily mean the initial time of the whole simulation. It can be seen as any time point of the simulation and Theorem 1 guarantees the safety and goal-reaching (with bounded error) starting from that time point.

4.5 Learning Large-scale Multi-agent Certified Control with Neural CCM and CBF

One of the main contributions of this work is a co-learning framework that jointly synthesizes the CCM M_i, the tracking controller $\mathbf{u}_{i,CCM}$, the decentralized CBF h_i, and the safe controller $\mathbf{u}_{i,CBF}$. In this section, we will elaborate on the formulation of the learning problem and the loss functions used for the training. We will also present a novel neural network architecture that can encode the local observation with time-varying dimension and permutation, which is crucial for our method to handle the complex multi-agent environment where neighboring topology constantly change.

4.5.1 Co-learning the CCM and Tracking Controller

Both the CCM and tracking controller are represented by neural networks, and the same type of agents share the same network parameters. Denote the reference state trajectory and control input of agent i as $\mathbf{x}_i^*(t)$ and $\mathbf{u}_i^*(t)$. The tracking controller $\mathbf{u}_{i,CCM}(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^{\mathbf{u}})$ is modeled using a neural network with parameters $\boldsymbol{\theta}_i^{\mathbf{u}}$, and the dual metric $W_i(\mathbf{x}_i; \boldsymbol{\theta}_i^w)$ is also a neural network with parameters $\boldsymbol{\theta}_i^w$. By design, the controller satisfies that $\mathbf{x}_i = \mathbf{x}_i^* \Rightarrow \mathbf{u}_{i,CCM}(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^{\mathbf{u}}) = \mathbf{u}_i^*$, and $W_i(\mathbf{x}_i; \boldsymbol{\theta}_i^w)$ is a symmetric matrix satisfying $W_i(\mathbf{x}_i; \boldsymbol{\theta}_i^w) \succeq \underline{w}\mathbf{I}$ for all $\mathbf{x}_i$ and all $\boldsymbol{\theta}_i^w$, where $\underline{w}$ is a hyper-parameter.

Denote the left side of (4.7) as $G_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^{\mathbf{u}})$. Then the objective of jointly learning the CCM and the tracking controller can be formally stated as:

$$\begin{aligned} &\text{For all } i, \text{ find } \boldsymbol{\theta}_i^w \text{ and } \boldsymbol{\theta}_i^{\mathbf{u}} \\ &\text{s.t.} \quad G_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^{\mathbf{u}}) \preceq 0, \quad \forall \mathbf{x}_i, \mathbf{x}_i^* \text{ and } \mathbf{u}_i^* \end{aligned} \tag{4.12}$$

Based on the objective (4.12), we are able to design loss functions to optimize the neural network parameters $\boldsymbol{\theta}_i^w$ and $\boldsymbol{\theta}_i^{\mathbf{u}}$. Let $\mathcal{V}_i$ denote the uniform distribution on $\mathcal{S}_i \times \mathcal{S}_i \times \mathcal{U}_i$. The *contraction loss* is defined as

$$\mathcal{L}_u(\boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^{\mathbf{u}}) = \mathop{\mathbb{E}}_{(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*) \sim \mathcal{V}_i} L_{PD}(-G_i(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^{\mathbf{u}})), \tag{4.13}$$

where $L_{PD}(\cdot) \geq 0$ is for penalizing non-positive definiteness and satisfies that $L_{PD}(A) = 0$ iff $A \succeq 0$. In practice, L_{PD} is implemented as follows. Given a matrix $A \in \mathbb{R}^{n \times n}$, we randomly sample K points $\{p_i \in \mathbb{R}^n \mid ||p_i||_2 = 1\}_{i=1}^K$. Then, the loss value is calculated as $L_{PD}(A) = \frac{1}{K} \sum_{i=1}^K \min\{0, -p_i^\intercal A p_i\}$. Obviously, $\mathcal{L}_u(\boldsymbol{\theta}_i^w, \boldsymbol{\theta}_i^{\mathbf{u}}) = 0$ implies that $u(\mathbf{x}_i, \mathbf{x}_i^*, \mathbf{u}_i^*; \boldsymbol{\theta}_i^{\mathbf{u}})$ and $W_i(\mathbf{x}_i; \boldsymbol{\theta}_i^w)$ satisfy inequality (4.7) exactly. Theoretically, the loss term $\mathcal{L}_u$ suffices to learn the CCM and the tracking controller. However, in practice, minimizing $\mathcal{L}_u$ alone leads to a very slow convergence in the training process. Therefore, we introduce a few auxiliary loss terms as follows.

First, inspired by the CCM theory, we add two auxiliary loss terms for the dual metric. As shown in Section 4.3, conditions (4.4) and (4.5) are sufficient for a metric to be a valid CCM. Intuitively, imposing these constraints on the dual metric $W_i(\mathbf{x}_i; \boldsymbol{\theta}_i^w)$ provides more guidance for finding a valid metric. Denoting the left side of (4.4) and (4.5) by $C_1(\mathbf{x}_i; \boldsymbol{\theta}_i^w)$ and $\{C_2^j(\mathbf{x}_i; \boldsymbol{\theta}_i^w)\}_{j=1}^m$,

the following loss functions are used

$$\mathcal{L}_{w1}(\boldsymbol{\theta}_i^w) = \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{x}_i^*,\mathbf{u}_i^*)\sim\mathcal{V}_i} L_{PD}(-C_1(\mathbf{x}_i;\boldsymbol{\theta}_i^w)), \tag{4.14}$$

$$\mathcal{L}_{w2}(\boldsymbol{\theta}_i^w) = \sum_{j=1}^{m} \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{x}_i^*,\mathbf{u}_i^*)\sim\mathcal{V}_i} \|C_2^j(\mathbf{x}_i;\boldsymbol{\theta}_i^w)\|_F, \tag{4.15}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Furthermore, as shown in Proposition 2, the overshoot of the tracking error is affected by the condition number of the metric. Since the smallest eigenvalue of the dual metric is lower bounded by $\underline{w}$ by design, penalizing the condition number is equivalent to penalizing the largest eigenvalue, and thus the following loss function is used

$$\mathcal{L}_c(\boldsymbol{\theta}_i^w) = \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{x}_i^*,\mathbf{u}_i^*)\sim\mathcal{V}_i} L_{PD}(\overline{w}\mathbf{I} - W_i(\mathbf{x}_i;\boldsymbol{\theta}_i^w)), \tag{4.16}$$

where $\overline{w}$ is a hyper-parameter.

In training, the following loss function is minimized to find a CCM and the corresponding controller:

$$\mathcal{L}_M(\boldsymbol{\theta}_i^{\mathbf{u}},\boldsymbol{\theta}_i^w) = \mathcal{L}_u(\boldsymbol{\theta}_i^w,\boldsymbol{\theta}_i^{\mathbf{u}}) + \mathcal{L}_c(\boldsymbol{\theta}_i^w) + \mathcal{L}_{w1}(\boldsymbol{\theta}_i^w) + \mathcal{L}_{w2}(\boldsymbol{\theta}_i^w).$$

Note that for each type of agents, only one CCM with the corresponding tracking controller needs to be trained.

4.5.2 Co-learning the Decentralized CBF and Safe Controller

In order to learn h_i, we can jointly learn a safe controller $\mathbf{u}_{i,CBF}$ such that the decentralized CBF conditions in (3.1) are satisfied. Although $\mathbf{u}_{i,CBF}$ is not directly used by our control pipeline in Figure 4-2 (a), it is still required in training because the third inequality in (3.1) depends on both the CBF and the controller. Both $\mathbf{u}_{i,CBF}$ and h_i are represented as neural networks with parameters $\boldsymbol{\theta}_i^\pi$ and $\boldsymbol{\theta}_i^h$ respectively. Let $T \subset \mathbb{R}_+$ be the time interval and $\tau_i = \{\mathbf{x}_i(t), \mathbf{o}_i(t)\}_{t\in T}$ be a trajectory of state and observation of agent i. Let $\mathcal{T}_i$ be the set of all possible trajectories of agent i. Define the function y_i as:

$$
\begin{aligned}
y_i(\tau_i, \boldsymbol{\theta}_i^h, \boldsymbol{\theta}_i^\pi) := \\
&\min \Big\{ \inf_{\mathcal{S}_{i,0}\cap\tau_i} h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h), \inf_{\mathcal{S}_{i,d}\cap\tau_i} -h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h), \\
&\inf_{\mathcal{S}_{i,h}\cap\tau_i} (\nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)\mathbf{u}_{i,CBF}(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^\pi)) + \\
&\qquad \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i + \alpha(h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h)) - \epsilon_h) \Big\}. \quad (4.17)
\end{aligned}
$$

The set $\mathcal{S}_{i,h} := \{(\mathbf{x}_i, \mathbf{o}_i) \mid h_i(\mathbf{x}_i, \mathbf{o}_i) \geq 0\}$. Note that the third term on the RHS of (4.17) depends on both the control policy and the CBF. It is clear that if we can find the $\boldsymbol{\theta}_i^h$ and $\boldsymbol{\theta}_i^\pi$ such that $y_i(\tau_i, \boldsymbol{\theta}_i^h, \boldsymbol{\theta}_i^\pi) > 0$ for $\forall \tau_i \in \mathcal{T}_i$ and $\forall i$, the conditions in (3.1) are satisfied. Therefore, the objective of learning multi-agent decentralized CBF is formulated as:

$$
\begin{aligned}
&\text{For all } i, \text{ find } \boldsymbol{\theta}_i^h \text{ and } \boldsymbol{\theta}_i^\pi \\
&\text{s.t.} \quad y_i(\tau_i, \boldsymbol{\theta}_i^h, \boldsymbol{\theta}_i^\pi) > 0, \quad \forall \tau_i \in \mathcal{T}_i.
\end{aligned} \quad (4.18)
$$

The decentralized CBF is required to satisfy three conditions specified in

(4.18), corresponding to the three terms in (4.17). The first condition states that for $(\mathbf{x}_i, \mathbf{o}_i)$ in the initial set $\mathcal{S}_{i,0}$, the CBF h_i is non-negative, from which we derive the loss function:

$$\mathcal{L}_{h1}(\boldsymbol{\theta}_i^h) = \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{o}_i)\in\mathcal{S}_{i,0}} \max\left(0, -h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h)\right). \tag{4.19}$$

The second condition requires that for $(\mathbf{x}_i, \mathbf{o}_i)$ in the dangerous set $\mathcal{S}_{i,d}$, h_i is negative, which gives rise to the loss function:

$$\mathcal{L}_{h2}(\boldsymbol{\theta}_i^h) = \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{o}_i)\in\mathcal{S}_{i,d}} \max\left(0, h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h)\right). \tag{4.20}$$

The third condition depends on both the controller and the control barrier function, requiring that $\dot{h}_i + \alpha(h_i) \geq 0$ for $(\mathbf{x}_i, \mathbf{o}_i) \in \mathcal{S}_{i,h}$, from which we can derive the loss function:

$$\begin{aligned}\mathcal{L}_{h3}(\boldsymbol{\theta}_i^h, \boldsymbol{\theta}_i^\pi) = \mathop{\mathbb{E}}_{(\mathbf{x}_i,\mathbf{o}_i)\in\mathcal{S}_{i,h}} \max\big(0, \epsilon_h - \nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + \\ B_i(\mathbf{x}_i)\mathbf{u}_{i,CBF}(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^\pi)) - \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i - \alpha(h_i(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^h))\big).\end{aligned}$$

For the class-$\mathcal{K}$ function $\alpha(\cdot)$, we simply choose a linear function $\alpha(h) = \lambda h$. Directly computing $\mathcal{L}_{h3}$ is difficult because we need to evaluate $\dot{o}_i$, which is the time derivative of the observation. We approximate $\nabla_{\mathbf{x}_i} h_i \cdot (\mathbf{f}_i(\mathbf{x}_i) + B_i(\mathbf{x}_i)\mathbf{u}_{i,CBF}(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^\pi)) + \nabla_{\mathbf{o}_i} h_i \cdot \dot{\mathbf{o}}_i$ by

$$\frac{1}{\Delta t}(h_i(\mathbf{x}_i(t + \Delta t), \mathbf{o}_i(t + \Delta t); \boldsymbol{\theta}_i^h) - h_i(\mathbf{x}_i(t), \mathbf{o}_i(t); \boldsymbol{\theta}_i^h)).$$

Note that $\mathcal{L}_{h1}, \mathcal{L}_{h2}$ and $\mathcal{L}_{h3}$ mainly consider safety instead of goal reaching. To train a safe control policy $\mathbf{u}_{i,CBF}(\mathbf{x}_i, \mathbf{o}_i; \boldsymbol{\theta}_i^\pi)$ that can drive the agent to the goal state, we also minimize the distance between $\mathbf{u}_{i,CBF}$ and $\mathbf{u}_{i,CCM}$. The goal

reaching loss $\mathcal{L}_g(\boldsymbol{\theta}_i^\pi) = \sum_{(\mathbf{x}_i,\mathbf{o}_i)\in\mathcal{S}} ||\mathbf{u}_{i,CBF}(\mathbf{x}_i,\mathbf{o}_i;\boldsymbol{\theta}_i^\pi) - \mathbf{u}_{i,CCM}||_2$. The final loss function is formulated as:

$$\mathcal{L}_B(\boldsymbol{\theta}_i^h,\boldsymbol{\theta}_i^\pi) = \mathcal{L}_{h1}(\boldsymbol{\theta}_i^h) + \mathcal{L}_{h2}(\boldsymbol{\theta}_i^h) + \mathcal{L}_{h3}(\boldsymbol{\theta}_i^h,\boldsymbol{\theta}_i^\pi) + \mathcal{L}_g(\boldsymbol{\theta}_i^\pi).$$

It is important to note that we did not use a *fixed* set of state-observation pairs to train the decentralized CBF and controllers. Instead, we adopted an on-policy training strategy, where the training data are collected by running the system with the current control policy. The collected state-observation pairs are stored in temporary dataset that is used to calculate the loss terms and update the decentralized CBF and controllers via gradient descent. Then the updated controllers are used to run the system and re-generate new state-observation pairs as training data. The iterative data collection and training is performed until the loss converges.

If the agents are homogeneous, for every agent i, their $h_i(\mathbf{x}_i,\mathbf{o}_i;\boldsymbol{\theta}_i^h)$ share the same neural network parameters $\boldsymbol{\theta}_i^h$. The same is true for $\mathbf{u}_{i,CBF}(\mathbf{x}_i,\mathbf{o}_i;\boldsymbol{\theta}_i^\pi)$. When the neural network parameters are found, adding new agents to the environment is straightforward. We only need to copy the $\mathbf{u}_{i,CCM}$ and h_i to the new agents. If the agents are heterogeneous, the neural network parameters for agents of the same type are shared.

4.6 Experiments

In this section, we will conduct comprehensive experiments on three simulation environments illustrated in Figure 4-1 with multi-agent reach-avoid tasks. We will evaluate the scalability to a large number of agents, the performance in terms of collision-avoidance and goal-reaching, as well as the time spent by

all agents to reach all their goals. A simple illustrative visualization of the decentralized CBF will be given to help interpret what the neural network has learned.

4.6.1 Task Environment Description

Auto cargo transportation for port containers In our first case study, we consider the simulated port automation task illustrated in Figure 4-1 (a), where multiple autonomous forklifts are controlled to move between port containers to transport cargo. We require each forklift to sequentially visit 4 goal locations and avoid collision with other forklifts and containers. The containers are of size $10m \times 5m \times 5m$ and are evenly placed $5m$ apart. The goal locations are randomly selected from the freespace of the seaport. We use the nonholonomic vehicle model described in Section 2 of [74]. The safety threshold $\kappa = 1m$, the safe set $\mathcal{S}_{i,s} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) \geq 1) \wedge (g_i(\mathbf{x}_i) \geq 0)\}$ and the dangerous set $\mathcal{S}_d = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) < 1) \vee (g_i(\mathbf{x}_i) < 0)\}$. $g_i(\mathbf{x}_i)$ encodes the maximum allowed velocity $5m/s$ of the forklifts as an additional safety constraint.

Drone deliveries in the city In our second case study, we consider the task of package delivery in a simulated city using drones, as shown in Figure 4-1 (b). Each drone is required to sequentially visit 4 goals to pick or place packages and avoid collision with other drones and static obstacles. The initial locations and goals are selected randomly from the freespace of the environment. We use the quadcopter model from Section VII of [38]. The safety threshold is set to be $\kappa = 1.6m$. The safe set $\mathcal{S}_{i,s} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) \geq 1.6) \wedge (g_i(\mathbf{x}_i) \geq 0)\}$ and the dangerous set $\mathcal{S}_d = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) < 1.6) \vee (g_i(\mathbf{x}_i) < 0)\}$. The maximum allowed velocity of the drones is set to be $20m/s$ and is encoded by

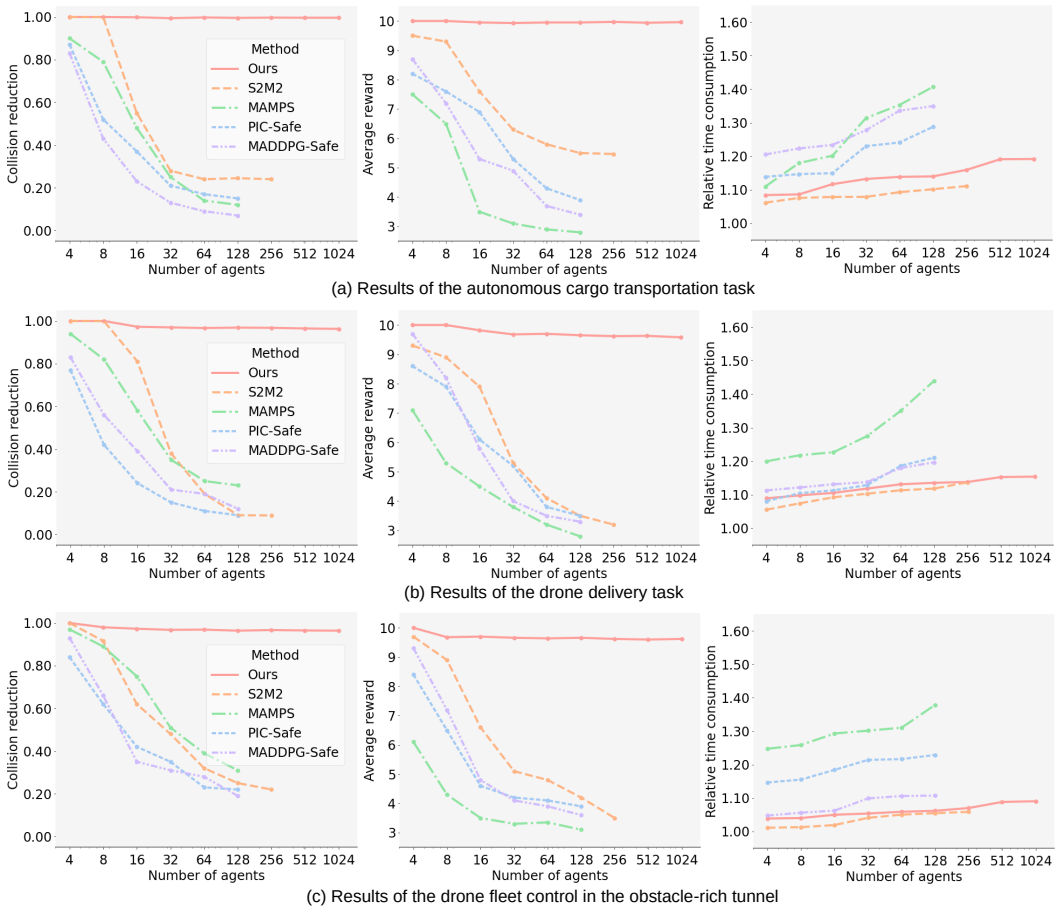
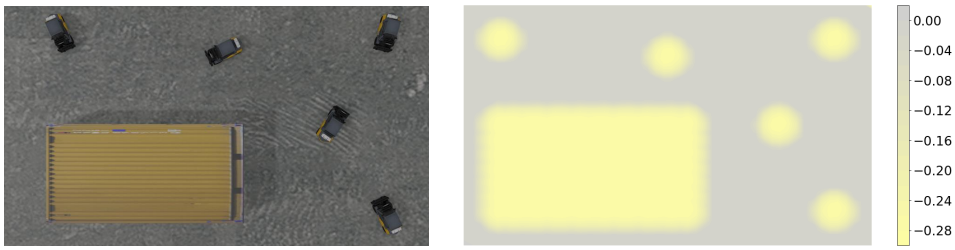


Figure 4-3: Results of the three case studies. (a) Autonomous cargo transportation for port containers. (b) Drone delivery in the city. (c) Drone fleet in an obstacle-rich tunnel. From left to right: collision reduction, average reward and relative time consumption. The results are averaged over 10 independent trials. The maximum number of agents for S2M2, MAMPS, PIC-Safe and MADDPG-Safe are limited by their memory and computational cost.



(a) Top view of the forklifts and container (b) Visualization of the CBF heatmap

Figure 4-4: Visualization of the CBF heatmap. Given the scenario in (a), the heatmap in (b) shows the value of CBF h_i of a new agent at each location as if the agent were assigned to be in the scene. Negative values indicate the dangerous set $\mathcal{S}_{i,d}$.

$g_i(\mathbf{x}_i)$.

Drone fleet control in an obstacle-rich tunnel The third case study is to control multiple drones to fly through an obstacle-rich tunnel environment without colliding with other drones or static obstacles, as is illustrated in Figure 4-1 (c). The obstacles are randomly placed in the last section of the tunnel. Starting from the left entrance, the drones are required to fly through the tunnel and exit from the right. We use the quadcopter model from Section VII of [38]. Similar to the drone delivery task, the safety threshold is set to be $\kappa = 1.6m$. The safe set $\mathcal{S}_{i,s} = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) \geq 1.6) \wedge (g_i(\mathbf{x}_i) \geq 0)\}$ and the dangerous set $\mathcal{S}_d = \{(\mathbf{x}_i, \mathbf{o}_i) \mid (r(\mathbf{x}_i, \mathbf{o}_i) < 1.6) \vee (g_i(\mathbf{x}_i) < 0)\}$. $g_i(\mathbf{x}_i)$ encodes the maximum allowed velocity $10m/s$.

4.6.2 Baseline Approaches

The baseline approaches we compare with include: MAMPS [106], PIC [53], MADDPG [55] and S2M2 [13]. A brief description of each method is as follows. MAMPS leverages the model dynamics to iteratively switch to safe control policies when the learned policies are unsafe. PIC proposes the permutation-

invariant critic to enhance the performance of multi-agent RL. We incorporate the safety reward to its reward function and denote this safe version of PIC as PIC-Safe. The safety reward is -1 when the agent enters the dangerous set. MADDPG is a pioneering work on multi-agent RL, and MADDPG-Safe is obtained by adding the safety reward to the reward function that is similar to PIC-Safe. S2M2 is a state-of-the-art model-based multi-agent safe motion planner combining FACTEST [27] and priority-based search [56]. As S2M2 is incomplete, when directly planning all agents fails, S2M2 evenly divides the agent group to smaller partitions for replanning until paths that are collision-free within each partition are found. The agents then follow the generated paths using LQR controllers. Notice that in the partition case, collision-avoidance is not guaranteed across partitions.

4.6.3 Evaluation Criteria

Since the primal focus of this chapter is the safety of multi-agent systems, we use **collision reduction** as a criteria when evaluating the methods. We first define the collision rate β of the multi-agent system as:

$$\beta = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{t\in T}\left[\mathbb{I}((\mathbf{x}_i(t),\mathbf{o}_i(t))\in\mathcal{S}_{i,d})\right], \tag{4.21}$$

where $\mathbb{I}(\cdot)$ is the indicator function that is 1 when its argument is true or 0 otherwise. The observation $\mathbf{o}_i$ contains the states of other agents within the observation (or sensing) radius, which is 10 times the safe distance. The safe distance is set to be the diagonal length of the bounding box of the agent. β measures the average frequency of entering the dangerous set $\mathcal{S}_d$. Note that β not only counts the collisions but also the time of violations of other safe requirements encoded in $g_i(\cdot)$, such as speed limits. However, in practice we

notice that collisions are the dominant unsafe behaviors, so we call the unsafe behaviors rate the “collision rate".

Given method P and a baseline method Q, the relative collision reduction of method P w.r.t. method Q is calculated as:

$$\gamma_{PQ} = \frac{\beta_Q - \beta_P}{\beta_Q} \in (-\infty, 1]. \tag{4.22}$$

If $\gamma_{PQ} = 1$, then $\beta_P = 0$, which means method P can avoid all the collision. If $\gamma_{PQ} = 0$, then method P does not have any improvement over the baseline method Q. γ_{PQ} can even be negative, which means method P exhibits a higher collision rate than method Q. In all experiments, the baseline method Q is set to be the LQR controller for each agent that directly tracks the reference trajectories, which do not consider collision-avoidance among agents.

In addition to the collision reduction, we also calculate the **average reward** that evaluates how good the task is accomplished in terms of goal reaching and safety. The agent is given a +10 reward if it reaches a goal and a -1 reward if it enters the dangerous set. Note that the agent might enter the dangerous set for many times before reaching the goal. If an agent has multiple goals to reach, the reward will be divided by the number of goals when we calculate the average reward. The reward is further averaged over all agents. The upper bound of the reward is 10, which can be obtained when the agent successfully reach its goals without any collision.

In order to evaluate the efficiency of different control methods, we measure the average time for all agents to reach the goals and define the **relative time consumption** metric. For a method P and baseline method Q, denote their time consumption as t_P and t_Q. Then the relative time consumption of P w.r.t. Q is t_P/t_Q. We use the same baseline method Q as we compute

the collision reduction. This measures how much delay was cause for agents to avoid collisions, comparing to each of them driving their goals without avoiding each other.

4.6.4 Implementation

The proposed framework is implemented using Python [96] and Tensorflow [1] on a computer with Intel Core-i7 processors and 8GB memory. In training, the tracking controller $\mathbf{u}_{i,CCM}$ and CCM M_i are learned for the forklift and drone models respectively using uniform random samples from the state and input space. The decentralized CBF h_i is trained with 8 homogeneous agents, and the training data is collected online as described in Section 4.5.2. To examine the generalization capability of our method, the h_i for drones are trained in a simple maze environment in [69] and has not seen the city or tunnel environments that it will be tested on. In testing, each agent is equipped with a copy of the learned CBF h_i and the tracking controller $\mathbf{u}_{i,CCM}$, then computes $\mathbf{u}_i$ in a decentralized fashion. For MAMPS, PIC-Safe and MADDPG-Safe, the environment and the number of agents are the same in training and testing. S2M2 is not learning-based and is directly used in testing without training.

In testing, we start from 4 agents and gradually increase to 1024 agents. The maximum number of agents for MAMPS, PIC-Safe and MADDPG-Safe is limited to 128 because their memory and computational requirements cannot be satisfied when training with 256 agents. The S2M2 method can simultaneously handle at most 8 agents in our scenario, so we partition the agents into small groups with size 8 and apply S2M2 to each group. Although S2M2 can avoid collision for agents in the same group, the collision among agents in different groups cannot be avoided. We will see that the performance of S2M2 drops drastically when there are more than 8 agents.

The exact model dynamics are visible to model-based methods including MAMPS, S2M2, and our methods, and invisible to the model-free MADDPG-Safe and PIC-Safe. Since the model-free methods do not have access to model dynamics but instead the simulators, they are more data-demanding. The number of state-observation pairs to train MADDPG-Safe and PIC-Safe is 10^3 times more than that of model-based learning methods to make sure that they converge to their best performance. Moreover, in training the RL-based methods MADDPG-Safe and PIC-Safe, the control action computed by LQR is also fed to the agent as one of the inputs to the actor-network. So the RL agents can also learn to use LQR as a reference controller for goal-reaching.

4.6.5 Experimental Results

The collision reduction, average reward and relative time consumption for the three case studies are shown in Figure 4-3. It is shown that our method is able to handle a very large number of agents and maintain a 0.96-1.0 collision reduction compared to the baseline, which means more than 96% collision among agents can be avoided. As the number of agents increases exponentially, there is only a slight performance drop for our method, while the compared methods exhibit drastic performance decline. Take the cargo transportation task as an example. When the number of agents grows from 4 to 256, the collision reduction of S2M2 decreases from 1.0 to 0.22, while using our method the number only decreases from 1.0 to 0.997. Similarly, in the drone delivery task with 128 agents, the collision reduction is 0.97 for our method and 0.12 for MADDPG-Safe, which demonstrates that our method can bring notable improvement in safety, even with a large number of agents. While being safe, our method is not conservative and does not sacrifice much in terms of time efficiency. Based on the relative time consumption in Figure 4-3, our goal-

reaching time only increases 3%-17% compared to the baseline, depending on the number of agents and the type of task. In comparison, MAMPS is more conservative and requires 10%-43% more time than the baseline, because it tries to stop the agents if they are predicted to collide.

The results in Figure 4-3 also demonstrate the generalization capability of our method to unseen scenarios. As we mentioned in Section 4.6.4, the tracking controller and CCM are trained with uniform random samples drawn from the state and input space without seeing the testing environment. Also, the decentralized CBF is trained with 8 agents but tested in different environments with much more agents. Our method is able to generalize to these unseen scenarios with notable performance. Although we tested with at most 1024 agents in our experiments, 1024 is not the limit of our approach. Given sufficient computational resource, more agents can be handled because the proposed control framework is decentralized and the computation for each agent is parallel. Adding more agents will not delay the computation of the control input.

A visualization of the learned CBF is shown in Figure 4-4 (b), given the scenario in Figure 4-4 (a) with 1 container and 5 agents. The heatmap is obtained by assigning a new agent to the scene and recording the output value of its CBF at each location. The velocities of all the agents are set to 0. It is shown that the CBF is negative in the vicinity of existing agents and obstacles, which means these locations are dangerous and can cause collision. At safe locations, the CBF is non-negative.

4.7 Summary

In this chapter, we presented a learning-based safe control framework for large-scale multi-agent reach-avoid control problems. We introduced a data-driven approach to jointly synthesize multi-agent decentralized control policy and their certificates for safety (using control barrier functions) and goal-reaching (using control contraction metrics). The whole framework is fully decentralized and can be used to control an arbitrarily large number of agents. We used three case studies to show that our method can generalize to scenarios with more than one thousand agents, and can be applied to an arbitrarily large number agents given sufficient computational resource and freespace in the environments. Comparing with state-of-the-art multi-agent control approaches, our method demonstrates notable improvement in terms of safety and goal-reaching performance.

The learned controllers and certificates are not perfect, and collision among agents can still occur occasionally. This is because minimizing the empirical loss functions on training samples cannot guarantee that the CCM and decentralized CBF conditions are fully satisfied over the entire space and on all the testing samples. To further validate the controller and certificates, we can use neural network analysis tools, such as reachability analysis [93, 100, 25, 40] and model checking of neural networks [65, 35, 104, 12, 2] to find counter-examples and improve the training process. This is be left as our future work.

Chapter 5

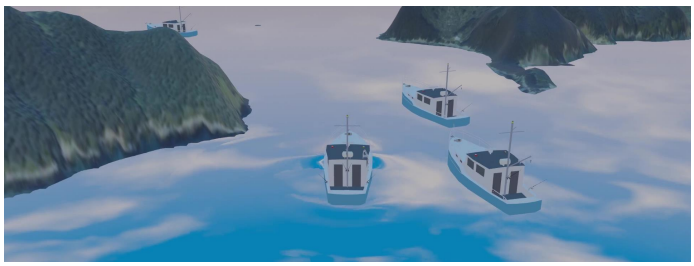
Learning Safe Control for Black-box Dynamical Systems

5.1 Introduction

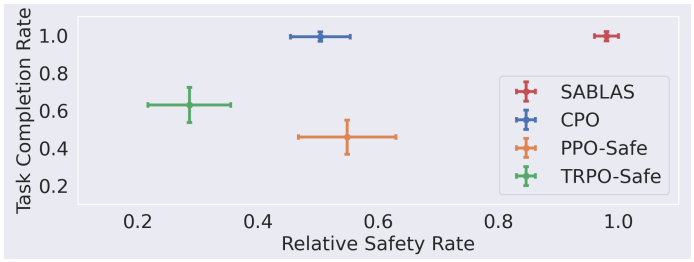
Control certificates based on barrier functions have been a powerful tool to generate probably safe control policies for dynamical systems. However, existing methods based on barrier certificates are normally for white-box systems with differentiable dynamics, which makes them inapplicable to many practical applications where the system is a black-box and cannot be accurately modeled. On the other side, model-free reinforcement learning (RL) methods for black-box systems suffer from lack of safety guarantees and low sampling efficiency.

In this chapter, instead of stressing about the trade-off between the strong guarantees from control certificates and the practicability of model-free methods, we propose SABLAS to achieve both. SABLAS is a general-purpose approach to learning **sa**fe control for **bla**ck-box dynamical **s**ystems. SABLAS enjoys the guarantees provided by the safety certificate from CBF theory with-

(a) CityEnv



(b) ValleyEnv



(c) Comparison with existing methods

Figure 5-1: (a) and (b) are two task environments we consider. In both CityEnv and ValleyEnv, the controller should control the ego vehicle to avoid collision with NPC vehicles. There are 1024 NPC drones in the city and 32 NPC ships in the valley. Each vehicle is assigned a random initial and goal location. (c) shows the average result for the two tasks, where the errorbars represent two times the standard deviation. SABLAS exhibits significant improvement in relative safety rate and task completion rate.

out requiring for an accurate model for the dynamics. Instead, SABLAS only needs a nominal dynamics function that can be obtained through regressions over simulation data. There is no need to model the error between the nominal dynamics and the real dynamics since SABLAS essentially re-design the loss function in a novel way to back-propagate gradient to the controller even when the black-box dynamical system is non-differentiable. The resulting CBF (and

the corresponding safety certificate) holds directly on the original black-box system if the training process converges. The proposed algorithm is easy-to-implement and follows almost the same procedure of learning CBF for white-box systems with minimal modification. SABLAS fundamentally solves the problem that control certificates cannot be learned directly on black-box systems, and opens up the next chapter use of CBF theory on synthesizing safe controllers for black-box dynamical systems.

Experimental results demonstrate the superior advantages of SABLAS over leading learning-based safe control methods for black-box systems including CPO [3], PPO-Safe [78] and TRPO-Safe [77, 89]. We evaluate SABLAS on two challenging tasks in simulation: drone control in a city and ship control in a valley (as shown in Figure 5-1). The dynamics of the drone and ship are assumed unknown. In both tasks, the controlled agent should avoid collision with uncontrolled agents and other obstacles, and reach its goal before the testing episode ends. We also examine the generalization capability of SABLAS on testing scenarios that are not seen in training. Figure 5-1 shows that SABLAS can reach a near 1.0 relative safety rate and task completion rate while using only 1/10 of the training data compared to existing safe RL methods, demonstrating a significant improvement. We also study the effect of model error (between the nominal model and actual dynamics) on the performance of the learned policy. It is shown that SABLAS is tolerant to large model errors while keeping a high safety rate. A detailed description of the results is presented in the experiment section. Video results can be found at supplementary materials.

To summarize the strength of SABLAS:

1. SABLAS can jointly find a safety certificate (i.e. CBF) and the corresponding control policy on black-box dynamics;

2. Unlike RL-based methods that need tedious trial-and-error on designing the rewards, SABLAS provides a systematic way of learning certified control policy, without parameters (other than the standard hyper-parameters in NN training) that need fine-tuning.

3. Empirical results that SABLAS can achieve a nearly perfect performance in terms of guaranteeing safety and goal-reaching, using much less samples than state-of-the-art safe RL methods.

5.2 Relate Work

There is a rich literature on controlling black-box systems and safe RL. Due to the space limit, we only discuss a few directly related and commonly used techniques on black-box system control. We will also skip the literature review for the large body of works on model-based safe control and trajectory planning as the research problems we are solving are very different.

Controller Synthesis for Black-box Systems. Proportional–integral–derivative (PID) controller is widely used in controlling black-box systems. The advantage of PID controller is that it does not rely on a model of the system and only requires the measurement of the state. A drawback of PID controller is that it does not guarantee safety or stability, and the system may overshoot or oscillate about the control setpoint. If the underlying black-box system is linear time-invariant, existing work [14] has presented a polynomial-time control algorithm without relying on any knowledge of the environment. For non-linear black-box systems, the dynamics model can be approximated using system identification and controlled using model-based approaches [32], and PID can be used to handle the error part. The concept of practical

relative degree [49] is also proposed to enhance the control performance on systems with heavy uncertainties. Recent advance in reinforcement learning [36, 52, 78, 77] also gives us insight into treating the system as a pure black-box and estimating the gradient for the black-box functions in order to optimize the control variables. However, in safety-critical systems, these black-box control methods still lack formal safety guarantee or certificate.

Simulation-guided controller synthesis methods can also generate control certificates for black-box systems, and sometimes those certificates can indicate how policies should be constructed [43, 90, 71]. However, most of these techniques use polynomial templates for the certificates, which limits their use on high-dimensional and complex systems. Another line of work studies the use of [23, 24] data-driven reachability analysis, jointly with receding-horizon control to constructed optimal control policies. These methods rely on side information about the black-box systems (e.g. lipschitz constants of the dynamics, monotonicity of the states, decoupling in the states' dynamics) to do the reachability, which is not needed in our method.

Safe Reinforcement Learning. Safe RL [3, 68, 84, 89, 102] extends RL by adding constraints on the expectation of certain cost functions, which encode safety requirements or resource limits. CPO [3] derived a policy improvement step that increases the reward while satisfying the safety constraint. DCRL [68] imposes constraint on state density functions rather than cost value functions, and shows that density constraint has better expressiveness over cost value function-based constraints. RCPO [89] weights the cost using Lagrangian multipliers and add it to the reward. FISAR [84] uses a meta-optimizer to achieve forward invariance in the safe set. A disadvantage of safe RL is that it does not provide safety guarantee, or their safety guarantee cannot be realized in

practice. The problem of sampling efficiency and sparsity of cost also increase the difficulty to synthesize safe controller through RL.

Safety Certificate and Control Barrier Functions. Mathematical certificates can serve as proofs that the desired property of the system is satisfied under the corresponding planning [91, 92] and control components. Such certificate is able to guide the controller synthesis for dynamical systems in order to ensure safety. For example, Control Lyapunov Function [22, 34, 39] ensures the existence of a controller so that the system converges to desired behavior. Control Barrier Function [5, 6, 10, 17, 18, 19, 21, 69] ensures the existence of a controller that keep the system inside a safe invariant set. However, the existing controller synthesis with safety certificate heavily relies on a white-box model of the system dynamics. For black-box systems or system with large model uncertainty, these methods are not applicable. While recent work [11] proposes to learn the model uncertainty effect on the CBF conditions, it still assumes that a handcrafted CBF is given, which is not always available for complex non-linear dynamical systems. Our approach represents a substantial improvement over the existing CBF-based safe controller synthesis strategy. The proposed SABLAS framework simultaneously enjoys the safety certificate from the CBF theory and the effectiveness on black-box dynamical systems.

5.3 Learning CBF on black-box systems

In this section, we first elaborate on why it is difficult to learn the safe controller and its CBF for black-box dynamical systems. Then we will propose an important and easy-to-implement re-formulation of the optimization objective, which makes learning safe controller in black-box systems as easy as in

white-box systems.

For clarity, we drop the subscript i and provide the analysis for a single agent. As the controllers and CBFs are fully decentralized, the analysis can be easily copied and extended to all agents.

5.3.1 Challenges in Black-box Dynamical Systems

Among the three terms of the CBF loss function 3.7, the third term is the only one that is differentiable w.r.t. the controller $\mathbf{u}$. We denote this term by $\mathcal{L}_p$:

$$\mathcal{L}_p = \frac{1}{|\mathcal{S}_p \cap \mathcal{D}|} \sum_{(\mathbf{x},\mathbf{o}) \in \mathcal{S}_p \cap \mathcal{D}} \max\Big(0, \gamma - \dot{h}(\mathbf{x}, \mathbf{o}) - \alpha\big(h(\mathbf{x}, \mathbf{o})\big)\Big). \tag{5.1}$$

Since $\mathcal{L}_p$ is differentiable w.r.t. control input $\mathbf{u}$, when we are minimizing $\mathcal{L}_p$ by gradient descent, the gradient will back-propagate to the parameters of the controller $\boldsymbol{\pi}$. This is how the controller is trained in white-box dynamical systems. Nevertheless, the main challenge of training safe controller with its CBF for black-box dynamical systems is that the gradient can no longer be back-propagated to $\boldsymbol{\pi}$ when $\mathbf{f}$ is unknown. Therefore, a safe controller cannot be trained by minimizing the loss functions in (3.7).

Given state samples $(\mathbf{x}(t), \mathbf{o}(t))$ and $(\mathbf{x}(t + \Delta t), \mathbf{o}(t + \Delta t))$ from the black-box system where Δt is a sufficiently small time interval, we can approximate $\dot{h}$ and compute $\mathcal{L}_p$ as:

$$\begin{aligned} \dot{h}_1(\mathbf{x}, \mathbf{o}) &= \frac{1}{\Delta t} h(\mathbf{x}(t + \Delta t), \mathbf{o}(t + \Delta t)) - h(\mathbf{x}(t), \mathbf{o}(t)) \\ \mathcal{L}_{p1} &= \frac{1}{|\mathcal{S}_p \cap \mathcal{D}|} \sum_{(\mathbf{x},\mathbf{o}) \in \mathcal{S}_p \cap \mathcal{D}} \max\Big(0, -\dot{h}_1(\mathbf{x}, \mathbf{o}) - \alpha(h(\mathbf{x}, \mathbf{o}))\Big). \end{aligned} \tag{5.2}$$

$\mathcal{L}_{p1}$ does give the exact value of $\mathcal{L}_p$, but its backward gradient flow to the controller is cut off by the black-box system that is non-differentiable. (5.2)

can only be used to train CBF h but not the safe controller $\boldsymbol{\pi}$. Even worse, the h obtained by minimizing (5.2) does not guarantee that a corresponding safe controller $\boldsymbol{\pi}$ exists. If we have a differential expression of dynamics $\mathbf{f}$ and replace $h(\mathbf{x}(t+\Delta t), \mathbf{o}(t+\Delta t))$ with:

$$h\big(\mathbf{x}(t) + \mathbf{f}\big(\mathbf{x}(t), \mathbf{u}(t)\big)\Delta t,\ \mathbf{o}(t) + \dot{\mathbf{o}}\Delta t\big), \tag{5.3}$$

then the gradient flow can successfully reach $\mathbf{u}$ and update the controller parameter. However, this is not immediately possible because $\mathbf{f}$ is unknown by the black-box assumption.

A possible way to back-propagate gradient to $\boldsymbol{\pi}$ is to use a differentiable nominal model $\mathbf{f}_{nom}$. There are many methods to obtain $\mathbf{f}_{nom}$, such as fitting a neural network using sampled data from the real black-box system. We do not require $\mathbf{f}_{nom}$ to perfectly match the real dynamics $\mathbf{f}$, because there will always exist an error between them. With $\mathbf{f}_{nom}$, we can approximate $\mathcal{L}_p$ as:

$$\begin{aligned}
\dot{h}_2(\mathbf{x}, \mathbf{o}) &= \frac{1}{\Delta t}\Big(h\big(\mathbf{x}(t) + \mathbf{f}_{nom}\big(\mathbf{x}(t), \mathbf{u}(t)\big)\Delta t,\ \mathbf{o}(t) + \dot{\mathbf{o}}\Delta t\big) - h(\mathbf{x}(t), \mathbf{o}(t))\Big) \\
\mathcal{L}_{p2} &= \frac{1}{|\mathcal{S}_p \cap \mathcal{D}|} \sum_{(\mathbf{x},\mathbf{o}) \in \mathcal{S}_p \cap \mathcal{D}} \max\Big(0, -\dot{h}_2(\mathbf{x}, \mathbf{o}) - \alpha(h(\mathbf{x}, \mathbf{o}))\Big),
\end{aligned} \tag{5.4}$$

which is differentiable w.r.t. $\boldsymbol{\pi}$ because both h and $\mathbf{f}_{nom}$ are differentiable. The gradient of $\mathcal{L}_{p2}$ can be back-propagated to the controller to update its parameters. However, whatever way we get $\mathbf{f}_{nom}$, there still exists an error between the real dynamics $\mathbf{f}$ and $\mathbf{f}_{nom}$, which means $\dot{h}_2$ is not a good approximation of $\dot{h}$ and $\mathcal{L}_{p2}$ is not the true value of $\mathcal{L}_p$. Using $\mathcal{L}_{p2}$, it is not guaranteed that the third CBF condition will be satisfied.

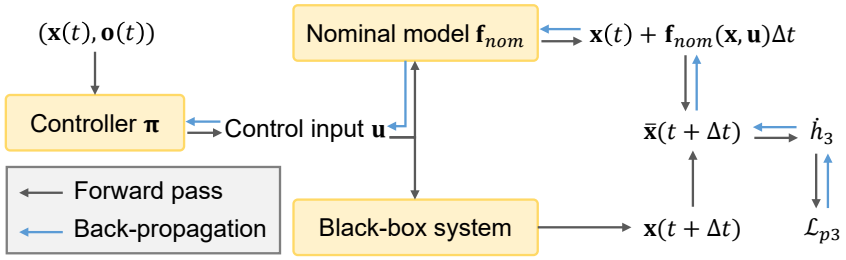


Figure 5-2: Computational graph of $\mathcal{L}_{p3}$. The blue lines show how the gradient from $\mathcal{L}_{p3}$ is back-propagated to the controller $\boldsymbol{\pi}$ despite that the black-box system itself is non-differentiable.

5.3.2 Rewire the Gradient Flow

We present a novel re-formulation of $\mathcal{L}_p$ that makes learning safe controller with CBF for black-box dynamical systems as easy as for white-box systems. Our formulation possesses two features: it enables the gradient to back-propagate to the controller in training, and offers an error-free approximation of $\dot{h}$.

Given state-observation samples $(\mathbf{x}(t), \mathbf{o}(t))$ and $(\mathbf{x}(t+\Delta t), \mathbf{o}(t+\Delta t))$ from the trajectories of the real black-box dynamical system, where Δt is a sufficiently small time interval, we define $\mathbf{x}_{nom}(t + \Delta t)$ as :

$$\mathbf{x}_{nom}(t + \Delta t) = \mathbf{x}(t) + \mathbf{f}_{nom}(\mathbf{x}(t), \boldsymbol{\pi}(\mathbf{x}(t), \mathbf{o}(t))\Delta t,$$

then construct $\bar{\mathbf{x}}(t + \Delta t)$ as:

$$\bar{\mathbf{x}}(t + \Delta t) = \mathbf{x}_{nom}(t + \Delta t) + g(\mathbf{x}(t + \Delta t) - \mathbf{x}_{nom}(t + \Delta t)),$$

where $g(x) = x$ is an identity function but *without gradient*. We need to pretend that $g(x)$ is a constant and in back-propagation, the gradient on $g(x)$ cannot propagate to its argument x. In PyTorch [64], there is an off-the-shelf implementation of $g(x)$ as $g(x) = x$.detach(), which cuts off the gradient from

g to x in back-propagation. Then we approximate $\mathcal{L}_p$ using:

$$\begin{aligned}\dot{h}_3(\mathbf{x},\mathbf{o}) &= \frac{1}{\Delta t}h(\bar{\mathbf{x}}(t+\Delta t),\mathbf{o}(t+\Delta t)) - h(\mathbf{x}(t),\mathbf{o}(t)) \\ \mathcal{L}_{p3} &= \frac{1}{|\mathcal{S}_p \cap \mathcal{D}|}\sum_{(\mathbf{x},\mathbf{o})\in\mathcal{S}_p\cap\mathcal{D}} \max\left(0, -\dot{h}_3(\mathbf{x},\mathbf{o}) - \alpha(h(\mathbf{x},\mathbf{o}))\right).\end{aligned} \tag{5.5}$$

Theorem 2

$\nabla_{\boldsymbol{\omega}}\mathcal{L}_{p3}$ exists and $\lim_{\Delta t\to 0}\mathcal{L}_{p3} = \mathcal{L}_p$. Namely, $\mathcal{L}_{p3}$ is differentiable w.r.t. the controller parameter, and $\mathcal{L}_{p3}$ is an error-free approximation of $\mathcal{L}_p$ as $\Delta t \to 0$.

Proof. Since $\mathbf{f}_{nom}$ is differentiable, $\mathbf{x}_{nom}$ and $\bar{\mathbf{x}}(t+\Delta t)$ are differentiable w.r.t. $\boldsymbol{\pi}$, $\dot{h}_3$ and $\mathcal{L}_{p3}$ are also differentiable w.r.t. $\boldsymbol{\pi}$ and its parameter $\boldsymbol{\omega}$. Thus, $\nabla_{\boldsymbol{\omega}}\mathcal{L}_{p3}$ exists. Furthermore, since $\bar{\mathbf{x}}(t+\Delta t) = \mathbf{x}(t+\Delta t)$, $\dot{h}_3$ is an error-free approximation of the real $\dot{h}$ when $\Delta t \to 0$. Thus $\mathcal{L}_{p3}$ is also an error-free approximation of $\mathcal{L}_p$ when $\Delta t \to 0$. □

Note that Theorem 2 reveals the reason why the proposed SABLAS method can jointly learn the CBF and the safe controller for black-box dynamical system. **First**, since $\nabla_{\boldsymbol{\omega}}\mathcal{L}_{p3}$ exists, the gradient from $\mathcal{L}_{p3}$ can be back-propagated to the controller parameters to learn a safe controller. On the contrary, $\dot{h}_1$ is not differentiable w.r.t. $\boldsymbol{\pi}$ so $\mathcal{L}_{p1}$ cannot be used to train the controller. **Second**, since $\mathcal{L}_{p3}$ is a good approximation of $\mathcal{L}_p$, minimizing $\mathcal{L}_{p3}$ contributes to the minimization of $\mathcal{L}_p$ and the satisfaction of the third CBF condition. On the contrary, $\dot{h}_2$ is an inaccurate approximation of $\dot{h}$ as we elaborated in Section 5.3.1. The construction of $\mathcal{L}_{p3}$ incorporates the advantages of $\mathcal{L}_{p1}$ and $\mathcal{L}_{p2}$, and avoids the disadvantages of $\mathcal{L}_{p1}$ and $\mathcal{L}_{p2}$ at the same time. The computational graph of $\mathcal{L}_{p3}$ is illustrated in Figure 5-2, which shows the forward pass and the backward gradient propagation from $\mathcal{L}_{p3}$ to controller $\boldsymbol{\pi}$.

Algorithm 1 summarizes the learning process of the safe controller and

Algorithm 1 Learning Safe Controller with CBF for Black-box Systems

Input: The set of initial condition $\mathcal{S}_0$, the dangerous set $\mathcal{S}_d$, controller $\boldsymbol{\pi}$ and CBF h with randomly initialized parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$, nominal dynamics $\mathbf{f}_{nom}$, loss function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\omega})$, the number of episodes K in data collection, the number of training iterations N.
for $i = 1, 2, \cdots, N$ **do**
    Initialize dataset $\mathcal{D} \leftarrow \emptyset$
    for $j = 1, 2, \cdots, K$ **do**
        $(\mathbf{x}(0), \mathbf{o}(0)) \leftarrow \textsc{Sample}(\mathcal{S}_0)$
        $\mathcal{D}_j \leftarrow \textsc{Run}(\boldsymbol{\pi}, \mathbf{x}(0), \mathbf{o}(0))$, $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$
    end for
    $\boldsymbol{\theta}, \boldsymbol{\omega} \leftarrow \textsc{Update}(\boldsymbol{\theta}, \boldsymbol{\omega}, \mathcal{D}, \mathcal{L})$
end for
return CBF parameter $\boldsymbol{\omega}$ and controller parameter $\boldsymbol{\theta}$

the corresponding safety certificate (CBF) for black-box dynamical systems. The RUN function runs the black-box system using the controller $\boldsymbol{\pi}$ under initial condition $\mathbf{x}(0)$, and returns the trajectory data. The UPDATE function updates parameters $\boldsymbol{\theta}, \boldsymbol{\omega}$ by minimizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \mathcal{L}^c(\boldsymbol{\theta}, \boldsymbol{\omega}) + \eta \mathcal{L}^g(\boldsymbol{\omega})$ using the state samples in $\mathcal{D}$ via gradient descent. The definition of $\mathcal{L}^c(\boldsymbol{\theta}, \boldsymbol{\omega})$ follows Equation (3.7), with $\mathcal{L}_p$ replaced by $\mathcal{L}_{p3}$ in Equation (5.5). $\mathcal{L}^g(\boldsymbol{\omega})$ is the goal-reaching loss following Equation (3.11).

Remark 1. One may argue that the gradient received by $\boldsymbol{\pi}$ via minimizing $\mathcal{L}_{p3}$ is not exactly the gradient it should receive if we had a perfect differentiable model of the black-box system. Despite this, minimizing $\mathcal{L}_{p3}$ directly contributes to the satisfaction of the third CBF condition. A safe controller and its CBF can be found to keep the system within the safe set.

Remark 2. Although the current formulation of $\mathcal{L}_{p3}$ leads to promising performance in simulation as we will show in experiments, $\mathcal{L}_{p3}$ requires further consideration in hardware experiments. Directly using the future state

$\mathbf{x}(t + \Delta t)$ to calculate the time derivative of h or $\mathbf{x}$ is not always desirable because noise will possibly dominate the numerical differentiation. When the noise dominates the time derivative of $\mathbf{x}$ or h, the training will have convergence issues. But a moderate noise is actually beneficial to training, because our optimization objective makes the CBF conditions hold even under noise disturbance, which increases the robustness of the trained CBF and controller. On physical robots where noise dominates the numerical differentiation, one can incorporate filtering techniques to mitigate the noise.

5.4 Experiments

The primary objective of our experiment is to examine the effectiveness of the proposed method in terms of safety and goal-reaching when controlling black-box dynamical systems. We will conduct comprehensive experiments on two simulation environments illustrated in Figure 5-1 (a) and (b), and compare with state-of-the-art learning-based control methods for black-box systems.

5.4.1 Task Environment Description

Drone control in a city (CityEnv) In our first case study, we consider the package delivery task in a city using drones, as is illustrated in Figure 5-1 (a). There is one controlled drone and 1024 non-player character (NPC) drones that are not controlled by our controller. In each simulation episode, each drone is assigned a sequence of randomly selected goals to visit. The aim of our controller is to make sure the controlled drone reach its goals while avoiding collision with NPC drones at any time. A reference trajectory $\mathbf{x}_{ref}(t), t \in [0, T]$ will be given, which sequentially connects the goals and avoid collision with buildings. The reference trajectory can be generated by

any off-the-shelf single-agent path planning algorithm. We use FACTEST [28] in our implementation, and other options such as RRT [48] are also suitable. The reference path planner does not need to consider the dynamic obstacles, such as the moving NPCs in our experiment. A nominal controller $\boldsymbol{\pi}_{norm}$ will also be given, which outputs control commands that drive the drone to follow the reference trajectory. However, $\boldsymbol{\pi}_{norm}$ is purely for goal-reaching and does not consider safety. The CityEnv has two modes: with **static** NPCs and **moving** NPCs. If the NPCs are static, they will constantly stay at their initial locations. If the NPCs are moving, they will follow pre-planned trajectories to sequentially visit their goals. The drone model is with state space $[x, y, z, v_x, v_y, v_z, \theta_x, \theta_y]$, where θ_x and θ_y are row and pitch angles. The control inputs are the angular acceleration of θ_x, θ_y and the vertical thrust. The underlying model dynamics is from [69] and assumed unknown to the controller and CBF in our experiment.

Ship control in a valley (ValleyEnv) In our second case study, we consider task of controlling a ship in valley illustrated in Figure 5-1 (b). There are one controlled ship and 32 NPC ships. The number of NPCs in ValleyEnv is less than CityEnv because ships are large in size and inertia, and hard to maneuver in dense traffic. Also, different from the 3D CityEnv, ValleyEnv is in 2D, which means the agents have fewer degrees of freedom to avoid collision. The initial location and goal location of each ship is randomly initialized at the beginning of each episode. The aim of our controller is to ensure the controlled ship reach its goal and avoid collision with NPC ships. Similar to CityEnv, a reference trajectory and nominal controller will be provided. There are also two modes in ValleyEnv, including the static and moving NPC mode, as is in CityEnv. The ship model is with state space $[x, y, \theta, u, v, \omega]$, where θ

is the heading angle, u, v are speed in the ship body coordinates, and ω is the angular velocity of the heading angle. The ship model is from Section 4.2 of [33] and is unknown to the controller and CBF.

Evaluation Criteria. Three evaluation criteria are considered. **Relative safety rate** measures the improvement of safety comparing to a nominal controller that only targets at goal-reaching but not safety. To formally define the relative safety rate, we first consider the absolute safety rate α:

$$\alpha = \frac{1}{T}\int_0^T \mathbb{I}((\mathbf{x}(t), \mathbf{o}(t)) \notin \mathcal{S}_d)\ dt, \tag{5.6}$$

which measures the proportion of time that the system stays outside the dangerous set. Given two control policies $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ with absolute safety rate α_1 and α_2, the relative safety rate of $\boldsymbol{\pi}_1$ w.r.t. $\boldsymbol{\pi}_2$ is defined as:

$$\beta_{12} = \frac{\alpha_1 - \alpha_2}{1 - \alpha_2} \in (-\infty, 1]. \tag{5.7}$$

If $\beta_{12} = 0$, then control policy $\boldsymbol{\pi}_1$ does not have any improvement over $\boldsymbol{\pi}_2$ in terms of safety. If $\beta_{12} = 1$, then $\boldsymbol{\pi}_1$ completely guarantees safety of the system in $t \in [0, T]$. In our experiment, $\boldsymbol{\pi}_1$ is the controller to be evaluated, and $\boldsymbol{\pi}_2$ is the nominal controller $\boldsymbol{\pi}_{norm}$ that only accounts for goal-reaching without considering safety. **Task completion rate** is defined as the success rate of reaching the goal state before timeout. **Tracking error** is the average deviation of the system's state trajectory comparing to a pre-planned reference trajectory $\mathbf{x}_{ref}(t), t \in [0, T]$ as:

$$\gamma = \frac{1}{T}\int_0^T ||\mathbf{x}(t) - \mathbf{x}_{ref}(t)||_2^2\ dt \tag{5.8}$$

Note that we do not assume $\mathbf{x}_{ref}$ always stay outside the dangerous set.

Baseline Approaches. In terms of safe control for black-box systems, the most recent state-of-the-art approaches are safe reinforcement learning (safe RL) algorithms. We choose three safe RL algorithms for comparison: **CPO** [3] is a general-purpose policy optimization algorithm for black-box systems that maximizes the expected reward while satisfying the safety constraints. **PPO-Safe** is a combination of PPO [78] and RCPO [89]. It uses PPO to maximize the expected cumulative reward while leveraging the Lagrangian multiplier update rule in RCPO to enforce the safety constraint. **TRPO-Safe** is a combination of TRPO [77] and RCPO [89]. The expected reward is maximized via TRPO and the safety constraints are imposed using the Lagrangian multiplier in RCPO.

Implementation and Training. Both the controller $\boldsymbol{\pi}$ and CBF h are multi-layer perceptrons (MLP) with architecture adopted from Section 4.2 of [69]. $\boldsymbol{\pi}$ and h not only take the state of the controlled agent as input, but also the states of 8 nearest NPCs that the controlled agent can observe. In Algorithm 1, we choose $K = 1, N = 2000$. The total number of state samples collected during training is 10^6. In UPDATE of the algorithm, we use the Adam [45] optimizer with learning rate 10^{-4} and batch size 1024. The gradient descent runs for 100 iterations in UPDATE. The nominal model dynamics are fitted from trajectory data in simulation. We used 10^4 state samples to fit a linear approximation of the drone dynamics, and 10^5 samples to fit a non-linear 3-layer MLP as the ship dynamics.

In training the safe RL methods, the reward in every step is the negative distance between the system's current state and the goal state, and the cost is

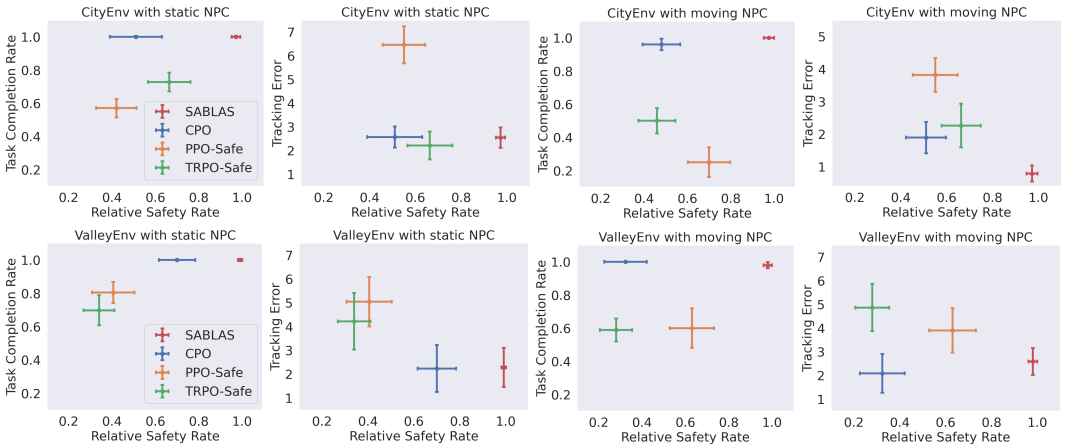


Figure 5-3: Quantitative results of relative safety rate, task completion rate and tracking error. The center points are mean and error bars represent standard deviations. Results are collected over 50 random runs on each method. The first row is for drone control in CityEnv and the second row is for ship control in ValleyEnv. Our method reaches a high task completion rate and relative rate at the same time, while keeping a low tracking error.

1 if the system is within the dangerous set $\mathcal{S}_d$ and 0 otherwise. The threshold for expected cost is set to 0, which means we wish the system never enter the dangerous set (never reach a state with a positive cost). During training, the agent runs the system for 10^7 timesteps in total and performs 2000 policy updates. In each policy update, 100 iterations of gradient descent are performed. The implementation of the safe RL methods is based on [72].

All the methods are trained with static NPCs and tested on both static and moving NPCs. We believe this can make the testing more challenging and examine the generalization capability of the tested methods in different scenarios. All the agents are assigned random initial and goal locations in every simulation episode, which prevents the learned controller from overfitting a single configuration.

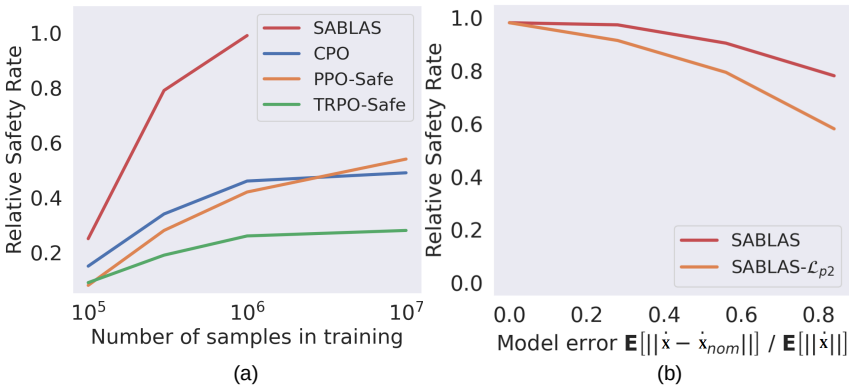


Figure 5-4: (a) Comparison of sampling efficiency. With only 1/10 samples, SABLAS achieves a nearly perfect relative safety rate. (b) Influence of model error on safety performance. SABLAS-$\mathcal{L}_{p2}$ uses Equation 5.4 instead of the proposed Equation 5.5 as loss function. SABLAS is tolerant to large error between nominal and real models while keeping a high safety rate.

5.4.2 Experimental Results

Safety and goal-reaching performance Results are shown in Figure 5-3. Among the compared methods, our method is the only one that can reach a high task completion rate and relative safety rate at the same time. For other methods such as TRPO-Safe, when the controlled drone or ship is about to hit the NPCs, the learned controller tend to brake and decelerate. Thus, the agent is less likely to reach its goal when a simulation episode ends. The task completion rate and safety rate are opposite to each other for CPO, PPO-Safe and TRPO-Safe. On the contrary, the controller obtained by our method can maneuver smoothly among NPCs without severe deceleration. This enables the controlled agent to reach the goal location on time. Our method can also keep a relatively low tracking error, which means the difference between actual trajectories and reference trajectories is small.

Generalization capability to unseen scenarios As is stated in **Implementation and Training**, the NPCs are static in training and can be either static or moving in testing. Figure 5-3 also demonstrates that our method has promising generalization capability across different training and testing scenarios.

Sampling efficiency In Figure 5-4 (a), we show the safety performance under different sizes of the training set. The results are averaged over the drone and ship control tasks. SABLAS only needs around 1/10 of the samples required by the compared methods to achieve a nearly perfect relative safety rate. Note that SABLAS requires an extra 10^4 to 10^5 samples to fit the nominal dynamics, but this does not change the fact that the total number of samples needed by SABLAS is much fewer than the baselines.

Effect of model error We investigate the influence of the model error $||\dot{\mathbf{x}} - \dot{\mathbf{x}}_{nom}||$ between the real dynamics and the nominal dynamics on the safety performance. We change the modeling error of the drone model and test the learning controller on CityEnv with static NPCs. We also perform an ablation study where we use $\mathcal{L}_{p2}$ in Equation 5.4 instead of $\mathcal{L}_{p3}$ in Equation 5.5 as loss function. The red curve in Figure 5-4 (b) show that SABLAS is tolerant to large model errors while exhibiting a promising safety rate. In our previous experiments, the model error $e = \mathbf{E}[||\dot{\mathbf{x}} - \dot{\mathbf{x}}_{nom}||]/\mathbf{E}[||\dot{\mathbf{x}}||]$ is always less than 0.2. We did not encounter any difficulty fitting a nominal model with empirical error $e \leq 0.2$. The orange curve in Figure 5-4 (b) shows that if we use $\mathcal{L}_{p2}$ in Equation 5.4, the trained controller will have a worse performance in terms of safety rate. This is because $\mathcal{L}_{p2}$ only uses the nominal dynamics to calculate the loss, without leveraging the real black-box dynamics.

Limitation The main limitation of the proposed approach is that it cannot guarantee the satisfaction of the CBF conditions in the entire state space. Even if we minimize the loss functions to 0 during training, the CBF conditions may still be occasionally violated during testing. After all, the training samples are finite and cannot cover the continuous state space. If the testing distribution and training distribution are the same, one can leverage the Rademacher complexity to give an error bound that the CBF conditions are violated, as is in Appendix B of [69]. But if the testing distribution is different from training, it is still unclear to derive the generalization error of the CBF conditions. To train CBF and controller that provably satisfy the CBF conditions, one can also use verification tools to find the counterexamples in the state space that violates the CBF conditions and add those counterexamples to the training set [22, 12]. The process is finished when no more counterexample can be found. However, the time complexity of the verification makes it not applicable for large and expressive neural networks. Also, the error between the nominal and real dynamics will have a negative impact on the safety performance. These limitations are left for future work.

5.5 Summary

We presented SABLAS, a general-purpose safe controller learning approach for black-box systems, which is supported by the theoretical guarantees from control barrier function theory, and at the same time is strengthened using a novel learning structure so it can directly learn the policies and barrier certificates for black-box dynamical systems. Simulation results show that SABLAS indeed provides a systematic way of learning safe control policies with a great improvement over safe RL methods. For future works, we plan to study SABLAS on multi-agent systems, especially with adversarial players.

Chapter 6

Summary of Thesis

Multi-agent control is a necessity in achieving full robot autonomy and accomplishing complex tasks that are beyond the capability of each individual. This thesis presents three original approaches to the key problems in multi-agent control. First, for multi-agent reach-avoid problems, we proposed a decentralized control barrier function-based framework, which simultaneously enjoys the expressiveness of machine learning and the safety guarantee of classical control theory. Second, for the multi-agent trajectory problems, we propose to combine neural contraction metrics with control barrier functions, which jointly guarantees tracking convergence and safety. Third, to handle black-box dynamical systems, we proposed a machine learning-based control framework that guarantees safety even when the system dynamics are unknown. Extensive experiments have shown the superior performance of the proposed methods against previous state-of-the-arts, and the potential to be applied in real-world settings.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[4] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed Autonomous Robotic Systems*, pages 203–216. Springer, 2013.

[5] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[6] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6271–6278. IEEE, 2014.

[7] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[8] Erin M Aylward, Pablo A Parrilo, and Jean-Jacques E Slotine. Stability and robustness analysis of nonlinear systems via contraction metrics and sos programming. *Automatica*, 44(8):2163–2170, 2008.

[9] Nicholas M Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques E Slotine, and Vikas Sindhwani. Learning stability certificates from data. *arXiv preprint arXiv:2008.05952*, 2020.

[10] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-Papers-OnLine*, 48(27):68–73, 2015.

[11] Fernando Castañeda, Jason J Choi, Bike Zhang, Claire J Tomlin, and Koushil Sreenath. Pointwise feasibility of gaussian process-based safety-critical control under model uncertainty. *Computing Research Repository (CoRR)*, abs/2106.07108, 2021.

[12] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, pages 3245–3254, 2019.

[13] Jingkai Chen, Jiaoyang Li, Chuchu Fan, and Brian C. Williams. Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, 2021.

[14] Xinyi Chen and Elad Hazan. Black-box control for linear dynamical systems. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 1114–1143. PMLR, 15–19 Aug 2021.

[15] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[16] Yuxiao Chen, Huei Peng, and Jessy Grizzle. Obstacle avoidance for low-speed autonomous vehicles with barrier function. *IEEE Transactions on Control Systems Technology*, 26(1):194–206, 2017.

[17] Yuxiao Chen, Andrew Singletary, and Aaron D Ames. Guaranteed obstacle avoidance for multi-robot operations with limited actuation: a control barrier function approach. *IEEE Control Systems Letters*, 5(1):127–132, 2020.

[18] Richard Cheng, Mohammad Javad Khojasteh, Aaron D Ames, and Joel W Burdick. Safe multi-agent interaction through robust control barrier functions with learned uncertainties. *arXiv preprint arXiv:2004.05273*, 2020.

[19] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[20] Jason Choi, Fernando Castañeda, Claire J Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*, 2020.

[21] Jason J Choi, Donggun Lee, Koushil Sreenath, Claire J Tomlin, and Sylvia L Herbert. Robust control barrier-value functions for safety-critical control. *Computing Research Repository (CoRR)*, abs/2104.02808, 2021.

[22] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *Robotics Science and Systems (RSS)*, 2021.

[23] Franck Djeumou, Abraham P Vinod, Eric Goubault, Sylvie Putot, and Ufuk Topcu. On-the-fly control of unknown smooth systems from limited data. In *2021 American Control Conference (ACC)*, pages 3656–3663. IEEE, 2021.

[24] Franck Djeumou, Aditya Zutshi, and Ufuk Topcu. On-the-fly, data-driven reachability analysis and control of unknown systems: an f-16 aircraft case study. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–2, 2021.

[25] Souradeep Dutta, Xin Chen, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Sherlock-a tool for verification of neural network feedback systems: demo abstract. In *Proceedings of the 22nd ACM International*

Conference on Hybrid Systems: Computation and Control, pages 262–263, 2019.

[26] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[27] Chuchu Fan, Kristina Miller, and Sayan Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification*, pages 629–652, Cham, 2020. Springer International Publishing.

[28] Chuchu Fan, Kristina Miller, and Sayan Mitra. Fast and guaranteed safe controller synthesis for nonlinear vehicle models. In Shuvendu K. Lahiri and Chao Wang, editors, *Computer Aided Verification*, pages 629–652, Cham, 2020. Springer International Publishing.

[29] Marcello Farina and Riccardo Scattolini. Tube-based robust sampled-data mpc for linear continuous-time systems. *Automatica*, 48(7):1473–1476, 2012.

[30] Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476, 2018.

[31] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Tenth Annual Symposium on Combinatorial Search*, 2017.

[32] Michel Fliess, Cédric Join, and Hebertt Sira-Ramirez. Complex continuous nonlinear systems: their black box identification and their control. *IFAC Proceedings Volumes*, 39(1):416–421, 2006.

[33] Thor I. Fossen. A survey on nonlinear ship control: from theory to practice. *IFAC Proceedings Volumes*, 33(21):1–16, 2000. 5th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2000), Aalborg, Denmark, 23-25 August 2000.

[34] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[35] Jie Fu and Ufuk Topcu. Probably approximately correct mdp learning and control with temporal logic constraints. *arXiv preprint arXiv:1404.7073*, 2014.

[36] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *6th International Conference on Learning Representations (ICLR)*, 2018.

[37] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[38] Sylvia L. Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F. Fisac, and Claire J. Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522, 2017.

[39] Alberto Isidori, ED Sontag, and M Thoma. *Nonlinear control systems*, volume 3. Springer, 1995.

[40] Radoslav Ivanov, James Weimer, Rajeev Alur, George J Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.

[41] Susmit Jha, Sunny Raj, Sumit Kumar Jha, and Natarajan Shankar. Duality-based nested controller synthesis from stl specifications for stochastic linear systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 235–251. Springer, 2018.

[42] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

[43] James Kapinski, Jyotirmoy V Deshmukh, Sriram Sankaranarayanan, and Nikos Aréchiga. Simulation-guided lyapunov analysis for hybrid

dynamical systems. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 133–142, 2014.

[44] S Mohammad Khansari-Zadeh and Aude Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 2014.

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (ICLR)*, 2015.

[46] Markus Kögel and Rolf Findeisen. Discrete-time robust model predictive control for continuous-time nonlinear systems. In *2015 American Control Conference (ACC)*, pages 924–930. IEEE, 2015.

[47] Wilbur Langson, Ioannis Chryssochoos, SV Raković, and David Q Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[48] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5:293–308, 2001.

[49] Arie Levant. Practical relative degree in black-box control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 7101–7106. IEEE, 2012.

[50] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations*, 2016.

[51] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR 2016 : International Conference on Learning Representations 2016*, 2016.

[52] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[53] Iou-Jen Liu, Raymond A Yeh, and Alexander G Schwing. Pic: permutation invariant critic for multi-agent deep reinforcement learning. In *Conference on Robot Learning*, pages 590–602, 2020.

[54] Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[55] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[56] Hang Ma, Daniel Harabor, Peter. J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. *AAAI 2019 : Thirty-Third AAAI Conference on Artificial Intelligence*, 33(1):7643–7650, 2019.

[57] Hang Ma, Sven Koenig, Nora Ayanian, Liron Cohen, Wolfgang Hönig, TK Kumar, Tansel Uras, Hong Xu, Craig Tovey, and Guni Sharon. Overview: Generalizations of multi-agent path finding to real-world scenarios. *arXiv preprint arXiv:1702.05515*, 2017.

[58] Anirudha Majumdar and Russ Tedrake. Robust online motion planning with regions of finite time invariance. In *Algorithmic foundations of robotics X*, pages 543–558. Springer, 2013.

[59] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36:947–982, 2017.

[60] Ian R Manchester and Jean-Jacques E Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 2017.

[61] David Q Mayne, Erric C Kerrigan, EJ Van Wyk, and Paola Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[62] David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[63] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[65] Stephen Prajna, Ali Jadbabaie, and George J Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[66] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[67] Han Qie, Dianxi Shi, Tianlong Shen, Xinhai Xu, Yuan Li, and Liujing Wang. Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning. *IEEE Access*, 7:146264–146272, 2019.

[68] Zengyi Qin, Yuxiao Chen, and Chuchu Fan. Density constrained reinforcement learning. In *International Conference on Machine Learning*, pages 8682–8692. PMLR, 2021.

[69] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *International Conference on Learning Representations*, 2021.

[70] Hadi Ravanbakhsh and Sriram Sankaranarayanan. Robust controller synthesis of switched systems using counterexample guided framework. In *2016 international conference on embedded software (EMSOFT)*, pages 1–10. IEEE, 2016.

[71] Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

[72] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *Deep Reinforcement Learning Workshop, Conference on Neural Information Processing Systems (NeurIPS)*, 7, 2019.

[73] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. *arXiv preprint arXiv:2004.03315*, 2020.

[74] Erick J Rodríguez-Seda, Chinpei Tang, Mark W Spong, and Dušan M Stipanović. Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing. *The International Journal of Robotics Research*, 33(12):1569–1592, 2014.

[75] Erick J. Rodríguez-Seda, Chinpei Tang, Mark W. Spong, and Dušan M. Stipanović. Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing. *The International Journal of Robotics Research*, 33(12):1569–1592, 2014.

[76] Matteo Saveriano and Dongheui Lee. Learning barrier functions for constrained motion planning with dynamical systems. *arXiv preprint arXiv:2003.11500*, 2020.

[77] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[78] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[79] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[80] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[81] Sumeet Singh, Benoit Landry, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust feedback motion planning via contraction theory. *The International Journal of Robotics Research*, 2019.

[82] Sumeet Singh, Spencer M Richards, Vikas Sindhwani, Jean-Jacques E Slotine, and Marco Pavone. Learning stabilizable nonlinear dynamics with contraction-based regularization. *arXiv:1907.13122*, 2019.

[83] Mohit Srinivasan, Amogh Dabholkar, Samuel Coogan, and Patricio Vela. Synthesis of control barrier functions using a supervised machine learning approach. *arXiv preprint arXiv:2003.04950*, 2020.

[84] Chuangchuang Sun, Dong-Ki Kim, and Jonathan P How. Fisar: Forward invariant safe reinforcement learning with a deep neural network-based optimizer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10617–10624. IEEE, 2021.

[85] Dawei Sun, Susmit Jha, and Chuchu Fan. Learning certified control using contraction metric. In *Conference on Robot Learning*, 2020.

[86] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[87] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions. *arXiv preprint arXiv:1912.10099*, 2019.

[88] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*, pages 708–717, 2020.

[89] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *7th International Conference on Learning Representations (ICLR)*, 2019.

[90] Ufuk Topcu, Andrew Packard, and Peter Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008.

[91] Jesus Tordesillas and Jonathan P How. Mader: Trajectory planner in multiagent and dynamic environments. *IEEE Transactions on Robotics*, 2021.

[92] Jesus Tordesillas, Brett T Lopez, and Jonathan P How. Faster: Fast and safe trajectory planner for flights in unknown environments. In *2019*

IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 1934–1940. IEEE, 2019.

[93] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T Johnson. Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. *arXiv preprint arXiv:2004.05519*, 2020.

[94] Hiroyasu Tsukamoto and Soon-Jo Chung. Neural contraction metrics for robust estimation and control: A convex optimization approach. *IEEE Control Systems Letters*, 5(1):211–216, 2020.

[95] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935. IEEE, 2008.

[96] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[97] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

[98] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[99] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.

[100] Weiming Xiang and Taylor T Johnson. Reachability analysis and safety verification for neural network control systems. *arXiv preprint arXiv:1805.09944*, 2018.

[101] Xiangru Xu, Jessy W Grizzle, Paulo Tabuada, and Aaron D Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 15(3):1216–1229, 2017.

[102] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *8th International Conference on Learning Representations (ICLR)*, 2020.

[103] Shuyou Yu, Christoph Maier, Hong Chen, and Frank Allgöwer. Tube mpc scheme based on robust control invariant set with application to lipschitz nonlinear systems. *Systems & Control Letters*, 62(2):194–200, 2013.

[104] Mojtaba Zarei, Yu Wang, and Miroslav Pajic. Statistical verification of learning-based cyber-physical systems. In *Proceedings of the 23nd ACM International Conference on Hybrid Systems: Computation and Control*, 2020.

[105] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*, pages 5872–5881, 2018.

[106] Wenbo Zhang and Osbert Bastani. Mamps: Safe multi-agent reinforcement learning via model predictive shielding. *arXiv preprint arXiv:1910.12639*, 2019.

[107] Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *arXiv preprint arXiv:1712.00600*, 2017.