From Cooperation to Competition: Prediction and Planning in Constrained Multi-Agent Settings using Data-Driven and Model-Based Optimal Control Methods

By

Edward Liu Zhu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Summer 2024

From Cooperation to Competition: Prediction and Planning in Constrained Multi-Agent Settings using Data-Driven and Model-Based Optimal Control Methods

Abstract

From Cooperation to Competition: Prediction and Planning in Constrained Multi-Agent Settings using Data-Driven and Model-Based Optimal Control Methods

by

Edward Liu Zhu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

As robotic systems become more advanced and their applications more complex, it is insufficient to consider a robot's behavior in isolation. Robots are increasingly expected to operate in environments populated with other intelligent agents. In order to perform their tasks well while obeying constraints, these robotic systems must be endowed with the ability to predict the behavior of the other agents in the environment in addition to planning their own actions over a time horizon. This is especially important in scenarios where agents do not communicate with each other and may behave in an adversarial manner. In this thesis, we investigate methods for prediction and planning for multi-agent systems over a variety of reward structures and information structures. Namely, we examine approaches which tackle the problem in cooperative, non-cooperative, and competitive scenarios where agents engage in partial or no communication about their intentions and future plans. These approaches are formulated through a combination of model-based optimal control and data-driven learning techniques, where we use data in a principled manner to construct or augment the objective and constraint functions of optimal control problems. This allows us to incorporate the rich and expressive behavior stemming from learned models in a transparent manner. We place a particular emphasis on the problem of autonomous racing, which is highly illustrative of competitive multi-agent settings with no agent communication and where both prediction and planning are paramount to achieving good performance while maintaining safety in the presence of adversarial agents. The presented approaches are evaluated in simulation and hardware experiments of vehicle navigation and racing tasks.

To my mom Sharon 刘晓雯, my dad John 祝峥嵘, and my sister Emmeline 祝君怡.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisor Professor Francesco Borrelli for his incredible guidance in all things academic and beyond. It has been such a pleasure to work with you and your mentorship has been crucial towards my growth both in terms of technical skills as well as my own philosophy towards mentorship. During my time here, I have observed how you build a research group based on respect, integrity, and the pursuit of excellence, while still developing deep and rewarding personal connections with your students. I hope to emulate your care and dedication as I continue with my career. I would like to thank Professor Alice Agogino, Professor Mark Mueller, Professor Somayeh Sojoudi, and Professor Koushil Sreenath for serving on my qualifying exam committee and to Professor Negar Mehr and Professor Claire Tomlin for serving on my dissertation committee.

I would like to thank my fellow lab mates at the Model Predictive Control Lab who have been amazing sources of inspiration and sounding boards for both good and bad ideas. I am proud to be a part of this group of excellent people and I will miss you all.

To my friends at Berkeley and beyond, thank you so much for your support and love over the years. You can't know how important your presence has been to my time here when navigating both the highs and lows. In particular, I'd like thank Alex Cain for his S-tier tier-lists, jam sessions, and always welcoming me into his home with an amazing americano. To Matt Arnold for being my adventure buddy who was always (and hopefully will continue to be) game for backpacking and snowboarding trips. This is the year for the Jets, I know it. To Mitch Thomas, for teaching me everything I know about WWII aircraft. I promise that our next hike will really only be 6 miles. To Brian Cera for being my first friend at Berkeley and my lucky charm when it comes to free drinks and below market rate housing. To Tony Zheng for his wizardry in the kitchen and for being a great friend. Your progression with macarons has been truly inspirational. I promise I'll go to a rave with you at some point (maybe). To Charlott Vallon for our daily walks to move cars and for your endless support even during some tough times. George is the best cat in the whole world. To Kate Schweidel for nerding out with me over theater and musicals and for the days at Stinson Beach. To my housemates Michael Abbott, Cyndia Cao, and Sebastian Lee, living with you guys has been just an absolute treat. I hope we'll have more 4 hour Spirit Island and Spikeball sessions in the future. To my climbing buddies Tim Brüdigam and Markus Fossdal, I loved how guys pushed me to be better both as a climber and as a person. Last but certainly not least, to Aimee Goncalves, Matt Kryder, Mallory Nation, and Ryan O'Sullivan. I got so lucky meeting you guys during my first stint at Berkeley, excited to spend 30 minutes on each turn for our future Civ sessions.

Finally, I'd like to thank my family: my mom Sharon 刘晓雯, my dad John 祝峥嵘, my sister Emmeline 祝君怡, and my brother-in-law Jonathan Bean. You guys have been immutable constants in my life and I am so blessed to be able to count on your support and love. I don't think I'm exaggerating when saying that the work contained in this dissertation would have been impossible without you all. I love you guys.

Chapter 1

Introduction

1.1 Background

As robotic systems are introduced to the real-world, they become one of many intelligent agents co-inhabiting a multi-agent environment which can include other robots as well as humans. Such multi-agent systems are becoming ubiquitous through a broad spectrum of rapidly maturing applications in autonomous driving [153], entertainment [155], warehouse logistics [57], and last-mile delivery [62] to name a few. Due to the increasing need for robots to coexist and operate in these complex environments, the modeling and control of multi-agent systems remains an exciting and active field of research, with recent work exploring applications in collaborative object manipulation [26, 150, 99], tightly constrained robot navigation [124, 125], game theoretic prediction and planning methods [90, 95, 51], and competitive autonomous racing [149, 69, 17]. No matter the embodiment, be it on wheeled ground vehicles, legged robots, or quadrotors, etc., the intelligence of a robotic system operating in a multi-agent environment must invariably have the ability to reason about and predict the behavior of the other agents in the environment (*how will the other agents react to my actions?*) and determine a course of action towards completing its task (*how should I react to the other agents' behavior?*). This is due to the complex interactions which arise from the sharing of a common resource, e.g. workspace, energy, bandwidth, etc., which must be taken into account in order for an agent to successfully complete their assigned task in a performant manner while also satisfying constraints which can govern actuation limits and safety. Failures in solving the prediction and planning problem can lead to poor performance, commonly in the form of the freezing robot problem [136], or constraint violations, which can be catastrophic in safety critical scenarios. Of course, prediction and planning are only part of the autonomy stack for robotic systems operating in multi-agent environments and ultimate success also critically depends on effective sensing and perception for state estimation of the robot as well as the other agents in the environment, which are complex research problems in of themselves. In this thesis, we focus specifically on the prediction and planning problem and make the blanket assumption that at any time instant

t, we have access to the state of each of the $i \in \{1, \dots, M\}$ agents x_t^i.

Solution approaches to the prediction and planning problem for multi-agent systems can be broadly categorized into data-driven learning-based methods and model-based optimal control methods, which have both achieved impressive results. In particular, data-driven approaches have seen much recent success in reaching super-human performance in a variety of games. These include classic board games such as go, chess, and shogi [119], as well as video games such as car racing in Gran Turismo [149] and real-time strategy in StarCraft [140]. Such data-driven approaches have also seen some limited realizations on physical systems in the context of pursuit-evasion games with quadrupedal robots [12] and competitive drone racing against human pilots [69]. Data-driven learning-based approaches can lead to highly sophisticated models which solve the prediction and planning problem implicitly and in a fully coupled manner to produce rich and interactive behaviors which would otherwise be difficult to obtain using explicitly designed models. However, this typically comes at the cost of high data requirements and a loss of transparency in the learned model, which may be prohibitive to implementation on real-world systems with safety critical constraints. Because of these reasons, optimal control techniques which leverage explicit models for prediction and planning appear to still be the preferred approach for real-world applications such as car racing [18, 67], formation control [93, 6], and autonomous driving [128, 130, 80]. In general such approaches pose the following finite-horizon discrete time optimal control problem with horizon length N for the i-th agent at some time step t:

$$\begin{aligned}
\min_{\mathbf{x}^i, \mathbf{u}^i} \quad & \sum_{k=0}^{N-1} J_k(x_k^i, u_k^i, \hat{x}_k^{\neg i}, \hat{u}_k^{\neg i}) + J_N(x_N^i, \hat{x}_N^{\neg i}) && \text{(1.1a)}\\
\text{subject to} \quad & x_0^i = x_t^i, && \text{(1.1b)}\\
& x_{k+1}^i = f(x_k^i, u_k^i, \hat{x}_k^{\neg i}, \hat{u}_k^{\neg i}),\ k = 0, \dots, N-1, && \text{(1.1c)}\\
& g_k(x_k^i, u_k^i, \hat{x}_k^{\neg i}, \hat{u}_k^{\neg i}) \leq 0,\ k = 0, \dots, N-1, && \text{(1.1d)}\\
& g_N(x_N^i, \hat{x}_N^{\neg i}) \leq 0, && \text{(1.1e)}\\
& \hat{\mathbf{x}}^{\neg i}, \hat{\mathbf{u}}^{\neg i} = h(\mathbf{x}^i, \mathbf{u}^i), && \text{(1.1f)}
\end{aligned}$$

where $\mathbf{x}^i = \{x_0^i, x_1^i, \dots, x_N^i\}$ and $u^i = \{u_0^i, u_1^i, \dots, u_{N-1}^i\}$ are the state and input sequences for agent i, which are the decision variables for the optimal control problem in (1.1). J_k and J_N are the stage and terminal cost functions in (1.1a), which can importantly be a function of both the trajectory of agent i as well as the predicted state and input trajectories of the other agents in the environment, which are denoted as $\hat{\mathbf{x}}^{\neg i} = \{\hat{x}_0^{\neg i}, \hat{x}_1^{\neg i}, \dots, \hat{x}_N^{\neg i}\}$ and $\hat{\mathbf{u}}^{\neg i} = \{\hat{u}_0^{\neg i}, \hat{u}_1^{\neg i}, \dots, \hat{u}_{N-1}^{\neg i}\}$ respectively. (1.1b) specifies the initial condition for agent i and its system dynamics are described in (1.1c), where we note the dependence on $\hat{x}_k^{\neg i}$ and $\hat{u}_k^{\neg i}$, which can describe dynamic coupling between agents. The stage and terminal constraint functions in (1.1d) and (1.1e) are again a function of the trajectories of all the agents in the environment and can describe both private constraints such as actuation limits as well as coupling constraints such as collision avoidance. Finally, (1.1f) describes the prediction subproblem, which itself can involve a separate but coupled optimal control problem or be

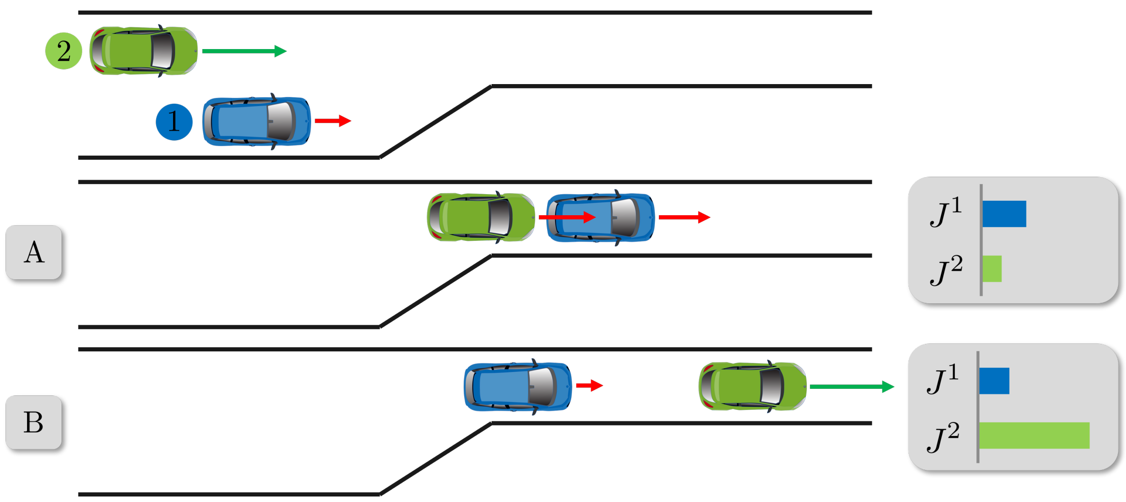


Figure 1.1: A bottleneck traversal scenario with two agents. (Top) The initial condition of the two agents. (Middle and Bottom) Two different outcomes of the scenario and the associated agent performance metrics J^1 and J^2, which in this case measure the amount of progress made by each agent along the road.

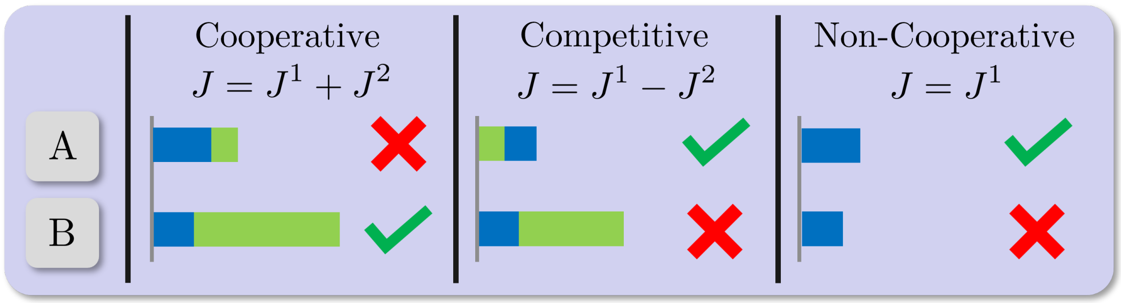


Figure 1.2: Illustration of reward structures from the scenario in Figure 1.1.

learned from data depending on the characteristics of the multi-agent environment which we will discuss momentarily. As in standard model predictive control (MPC) techniques, we may then apply the first element of solution input sequence to agent i and resolve at the next time step for online prediction and planning. So far, we have described the multi-agent prediction and planning problem at a high level and in general terms. Now, let us present the specific realizations of this problem which will be investigated in this thesis. In particular, we will view them through a combination of cooperative, non-cooperative, and competitive reward structures and full, partial, and no communication information structures.

Reward Structures

To understand how different reward structures impact both the optimal control problem in (1.1) as well as the behavior of the agents in the multi-agent system, let us consider the example scenario illustrated in Figure 1.1 where two cars are attempting to traverse a bottleneck which is only wide enough for one car to pass through at a time. We denote the blue and green cars as agent 1 and 2 respectively and initialize their states such that agent 2 starts behind agent 1 but with a higher velocity. For this initial condition, consider the two outcomes labeled as A and B in Figure 1.1. In outcome A, agent 1 speeds up and merges in front of agent 2, thereby forcing it to slow down and match the speed of agent 1. In outcome B, agent 1 slows down and merges behind agent 2. This allows agent 2 to preserve its speed through the bottleneck. Individual agent performance in this scenario is measured by the amount of progress made by each of the cars along the road and is visualized by the blue and green bars labeled J^1 and J^2. The aforementioned reward structures then correspond with different ways of describing the *overall* objective of the scenario from the perspective of each agent, i.e. how the objective functions J are constructed in (1.1a). In the case of cooperative agents, it is common to measure the performance of the scenario as a whole by summing together the objectives of the individual agents, i.e. $J = -(J^1 + J^2)$ (note the negative sign here is to conform to the minimization performed in (1.1)). Doing so, we observe a clear advantage for outcome B as shown in the first column of Figure 1.2. If we view the scenario from the perspective of agent 1, this means that in cooperative settings, it may be beneficial to sacrifice ones own performance as long as it benefits the performance of the group as a whole. In the competitive case, we can describe the performance of the scenario via the difference between the agent objectives, i.e. $J = J^2 - J^1$, which directly captures the conflicting goals of the two agents. Now if we were to again view the scenario from the perspective of agent 1, the scenario objective would indicate a preference for outcome A as shown in the second column of Figure 1.2. In this case, agent 1 is driven to behavior which not only maximizes their own performance, but minimizes or obstructs the performance of their opponent, i.e. agent 2. Finally, when agents are non-cooperative, it is straightforward to equate the agent objective with the scenario objective, i.e. $J = -J^1$ in the case of agent 1. In such scenarios, individual agents are disinterested in the performance of the other agents in the environment, yet their behavior can still be coupled via shared constraints, which necessitates planning over agent interactions.

Information Structures

Whereas the reward structure primarily impacts the construction of the objective function in (1.1a), the information structure governs the realization and complexity of the prediction module h in (1.1f). The three classes of information structure that we consider are illustrated in Figure 1.3 which show example communication graphs for a multi-agent system of three agents with a base station. In the case of full communication, each agent is connected to all other agents in the environment via bidirectional edges, thereby allowing for two-way communication between any pair of agents. Such information structures are rare in practice as the networking infrastructure and bandwidth required to maintain such a communication network scales poorly as the number of agents grows. However, they can lead to fairly trivial realizations of (1.1f) where each agent communicates their future plan of $\mathbf{x}^i$ and $\mathbf{u}^i$ in full to the other agents. The second case is that of partial communication, where the communication

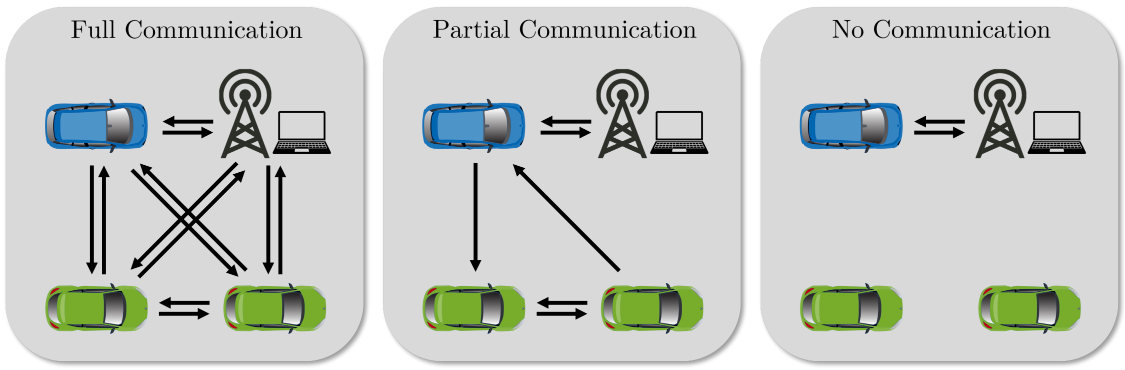


Figure 1.3: Illustration of information structures in multi-agent environments.

graph is directed and sparse. As an example, this can represent multi-agent systems with local coupling or a hub and spoke network where the agents are all connected to a central base station but do not directly communicate with each other. In these cases, prediction of the other agents typically requires joint solution of the planning problem over neighboring or coupled agents and the imposition of consensus constraints in h between coupled agents such that the plans obtained via solution of (1.1) by agent i align with those by agent j and vice versa. The last case is that of no communication, where no information is shared between agents. This case is commonly found in competitive scenarios where agents carefully guard information about their intentions and future plans to maintain a competitive advantage. In such cases, the prediction subproblem can be approached from a variety of directions such as machine learning and game theory, both of which will be explored in this thesis.

1.2 Thesis Outline and Contributions

This thesis is divided into two parts. The first part, which consists of Chapters 2, 3, and 4 consider the cooperative and non-cooperative settings where there are no adversarial intents between agents. The second part, which consists of Chapters 5, 8, 6, and 7 deal with competitive scenarios in which agents may behave in an adversarial manner towards each other and we focus on the specific application of car racing. The contributions of the individual chapters are highlighted below.

In Chapter 2, we consider the cooperative reward setting where agents are tasked with completing a task in an episodic or iterative manner. In particular, we propose a decentralized learning model predictive control scheme for nonlinear dynamically decoupled multi-agent systems where no communication is required during online execution of the task and only partial communication is required between each agent and a central coordinator between iterations of the task. Assuming that an initial feasible trajectory is provided, we guarantee that all agents will remain feasible and stable and that the performance of each agent is non-decreasing over iterations. This is accomplished by careful construction of a terminal set and cost-to-go function for the MPC problem as well as decomposing the shared workspace over agents using trajectory data from the previous iterations of task execution. As synthesis of all terminal components and constraints are performed between iterations, online execution of the task can then be done in a completely decentralized manner with no communication between agents.

In Chapter 3, we consider the same cooperative and iterative multi-agent setting, but specify the learning-based approach for systems with linear but possibly coupled dynamics in addition to coupling through constraints. By leveraging an ADMM based method for distributed optimization, the control scheme requires only partial communication between coupled agents in order to establish consensus over shared decision variables. We show that the global system is feasible and stable over iterations of task execution and that the performance of the system as a whole is non-decreasing and converges to the globally optimal trajectory of the centralized system.

In Chapter 4, we consider a non-cooperative multi-agent setting in a tightly constrained environment where there is no communication between agents. An example of such a scenario is a parking lot where simple but approximate geometric representations of a vehicle, such as circular collision buffers, can lead to deadlock or highly conservative stop-and-wait behavior in the presence of other moving vehicles. To reduce conservativeness in vehicle navigation, we propose a hierarchical planning framework which leverages learned high-level strategies to construct time-varying constraints which predict a reachable set to the goal at each time step of the navigation task. We incorporate these constraints in an MPC policy which also uses an exact representation of vehicle geometry to improve navigation performance in tight spaces.

In Chapter 5, we begin investigating competitive multi-agent scenarios with no inter-agent communication. A particularly illustrative instance of this class of multi-agent scenario is that of racing where each agent is not only trying to maximize their progress along a race track, but may also engage in adversarial behavior aimed at impeding the progress of their opponents, as was illustrated in outcome A of Figure 1.1. In such scenarios, the prediction subproblem becomes especially challenging, especially when using explicit model-based approaches, as knowledge of the objective, constraint, and dynamics functions of one's opponents is unlikely to be shared. We therefore propose a learning-based approach for solving the prediction subproblem which leverages trajectory data from prior races to train a Gaussian process regression model of an opponent's one-step closed-loop behavior. Importantly this model is conditioned on the behavior of an ego vehicle which allows for it to capture interactions between the agents. We leverage the predictions of this model as well as its associated uncertainties to construct a racing MPC policy which enabled an ego agent to win races against an adversarial opponent in hardware races.

Though explicit models of ones opponent are difficult to obtain in competitive scenarios, if available, they may be leveraged in a game-theoretic setting to obtain highly interactive solutions to the prediction and planning problem through the finding of game equilibrium. In Chapter 6, we present a novel iterative method for finding generalized Nash equilibria (GNE) of open-loop dynamic games with non-linear system dynamics, and non-convex constraint and objective functions. The approach, called DG-SQP, is based on the sequential quadratic programming method for non-linear programming and exhibits guaranteed local linear convergence to GNE. The novelty of our approach lies in an emphasis on globalization techniques which improve the numerical robustness of the solver as well as a dynamic game formulation which is tailored to racing tasks. We show that our approach out-performs the state-of-the-art in terms of success rate in convergence to GNE for various racing scenarios.

Though we are able to achieve high success rates when solving for GNE using DG-SQP from Chapter 6, one significant limitation which prevents its use for *real-time* prediction and planning is the low computational efficiency, which in part is caused by requiring the solution of a constrained quadratic program at each iteration of the solver. In order to incorporate game-theoretic equilibria in real-time racing, in Chapter 7, we propose a strategic learning approach which leverages the *reward outcomes* associated with a set of equilibria to learn a game-theoretic value function which is conditioned on opponent behavior. We incorporate this value function in a racing MPC policy as the terminal cost function in order to guide interaction and competition against an opponent in head-to-head hardware races. We show significant improvement in win-rate against the same MPC policy with a non-game-theoretic terminal cost function.

In Chapter 8, we present a discussion using simple but illustrative examples on the benefits and drawbacks of approaching the competitive racing problem using tools from game theory. In particular, we show that in the context of model-based methods, a game-theoretic formulation provides important advantages over a centralized optimal control approach in capturing the competitive nature of racing tasks which are commonly described via zero-sum objectives. However, we also show that by requiring assumptions about the opponent's models, the solutions to the prediction and planning problem can be fragile and sensitive to model mismatch. We conclude our discussion with a survey of approaches which seek to address this limitation and address how the work presented in Chapters 6 and 7 are complementary towards the larger effort of broadening the impact of game-theoretic methods in multi-agent robotic systems.

1.3 List of Publications

The results presented in this thesis have appeared in a number of publications by the author. In particular:

- Chapter 2 is based on:
  - **E. Zhu**, Y. Stürz, U. Rosolia, and F. Borrelli, "Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 6198-6203.
- Chapter 3 is based on:

  - Y. Stürz, **E. Zhu**, U. Rosolia, K. Johannson, and F. Borrelli, "Distributed learning model predictive control for linear systems". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 4366-4373.

- Chapter 4 is based on:

  - X. Shen, **E. Zhu**, Y. Stürz, U. Rosolia, and F. Borrelli, "Collision avoidance in tightly-constrained environments without coordination: a hierarchical control approach". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2674-2680.

- Chapter 5 is based on:

  - **E. Zhu**, F. Busch, J. Johnson, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 8186–8191.

- Chapter 6 is based on:

  - **E. Zhu** and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3211-3217.

Chapter 2

Decentralized Learning Model Predictive Control for Multi-Agent Non-linear Systems

2.1 Introduction

In this chapter, we study the problem of decentralized Model Predictive Control (MPC) for dynamically decoupled multi-agent systems under the minimum-time cost and coupled state constraints. Multi-agent systems typically exhibit inter-agent coupling, which can be expressed as constraints on the global system. MPC is a well-studied approach to the control of such constrained systems and can be applied in a global manner for centralized control of multi-agent systems with a small number of agents. However, as the number of agents increases, centralized approaches typically become intractable in practice due to limitations in computational power and communication capacities [100]. This gives rise to decentralized and distributed MPC schemes, which leverage the inherent parallelizable structure of multi-agent systems to reduce the required computational effort. Feasibility and stability of MPC are typically obtained using a terminal cost function and terminal constraints in the MPC design [94], which we refer to as terminal components. However, synthesis of these terminal components for the control of nonlinear multi-agent systems is in general challenging.

The primary advantage of decentralized MPC over its distributed counterpart is the lower communication demand. While only local information is used in the control of each agent in decentralized schemes, additional communication between agents is required to obtain a control action in distributed methods [15, 101, 43, 48, 35, 93, 89, 131].

In decentralized MPC, local controllers are synthesized for each agent, where feasibility and stability properties are attained via robustness of the controller against coupling to other agents. In prior work, conditions for the stability of decentralized MPC for nonlinear systems are investigated. In [4, 3], this is attained via terminal cost synthesis and an online supervisory scheme for modifying the decoupling structure to meet these conditions without destabilizing the system. The work in [71] achieves this by bounding the prediction error of neighboring agents' states. However, this method assumes the singleton terminal set of the origin, which limits the controller's domain of attraction. In addtion, neither method can deal with coupling state constraints between the agents.

Iterative approaches to the control of multi-agent systems have also been proposed. Of such methods, Sequential Convex Programming (SCP) [27] is similar to the approach proposed in this paper. SCP solves a non-convex optimization problem by successively forming convex approximations about previous solutions. In [10], SCP was used to generate collision-free trajectories for multiple quadcopters in a centralized manner. This was extended into a decentralized formulation in [33], which showed improvements in computational tractability. However, in these works, SCP is a *heuristic* method and may fail to find a feasible solution. In addition, SCP optimizes over the entire trajectory, which contrasts with the receding horizon approach taken in this work, and may be computationally challenging for long time horizons or fine time discretizations.

In this chapter, we propose a decentralized approach to trajectory optimization for non-linear multi-agent systems. By performing the same task over iterations, we collect data on the agents' closed-loop behavior to successively improve the construction of terminal components for the local controllers. In particular, as we improve an estimate of the terminal cost (an approximate value function) and expand the terminal constraint set (the domain of this function), we iteratively improve closed-loop performance of the multi-agent system, while maintaining feasibility and finite-time convergence guarantees.

The contribution of this chapter is twofold.

- We extend the method of Learning Model Predictive Control (LMPC) from [113] to the multi-agent case for dynamically decoupled agents under the minimum-time cost and coupled state constraints. In particular, we first propose a procedure for synthesizing the MPC terminal components using data from previous iterations of task execution. We then show that the resulting decentralized LMPC has the properties of persistent feasibility, finite-time closed-loop convergence to the goal state, and non-decreasing performance over iterations.

- We demonstrate the effectiveness of the decentralized method with a numerical example in the context of multi-vehicle collision avoidance, where we observe a significant reduction of computational effort compared to a centralized approach.

The results presented in this chapter have also appeared in:

- E. Zhu, Y. Stürz, U. Rosolia, and F. Borrelli, "Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 6198-6203.

2.2 Preliminaries

System Description

For a system of M agents, the set of indices $\{1, \ldots, M\}$ is denoted as $\mathcal{M}$. '$\preceq$' denotes an element-wise inequality. Consider the global nonlinear time-invariant discrete-time system composed of M agents

$$x_{t+1} = f(x_t, u_t), \tag{2.1}$$

where the global state and input vectors at sampling time $t \geq 0$ are formed by stacking those from each agent into a single column, i.e. $x_t = \mathrm{col}_{i\in\mathcal{M}}(x_{i,t}) = [x_{1,t}^\top, \ldots, x_{M,t}^\top]^\top \in \mathbb{R}^n$ and $u_t = \mathrm{col}_{i\in\mathcal{M}}(u_{i,t}) \in \mathbb{R}^m$, where $x_{i,t} \in \mathbb{R}^{n_i}$ and $u_{i,t} \in \mathbb{R}^{m_i}$. Each agent in the global system is subject to local state and input constraints,

$$x_{i,t} \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}, \quad u_{i,t} \in \mathcal{U}_i \subseteq \mathbb{R}^{m_i}, \quad \forall t \geq 0. \tag{2.2}$$

These local constraint sets are assumed to be closed, compact, and include the goal states $x_{i,F}$ in their relative interiors. The global system is additionally subject to coupling constraints on the system state,

$$g(x_t) \preceq 0, \quad \forall t \geq 0. \tag{2.3}$$

We assume that the agents are dynamically decoupled with continuous dynamics and locally stabilizable, which means that we can write (interchangeably with (2.1)), for all agents $i \in \mathcal{M}$, the local dynamics as

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}). \tag{2.4}$$

Control Objective

The objective is to design a controller which drives the system state to a goal state x_F by solving the following optimal control problem for the global system

$$\begin{aligned}
\min_{\mathbf{u}_T} \quad & \sum_{t=0}^{T} h(x_t, u_t) && \text{(2.5a)}\\
\text{s.t.} \quad & x_{t+1} = f(x_t, u_t), \quad \forall t \in \{0, \ldots, T-1\} && \text{(2.5b)}\\
& x_0 = x_S,\ x_T = x_F && \text{(2.5c)}\\
& x_t \in \mathcal{X},\ u_t \in \mathcal{U} \quad \forall t \in \{0, \ldots, T\} && \text{(2.5d)}\\
& g(x_t) \preceq 0, \quad \forall t \in \{0, \ldots, T\}, && \text{(2.5e)}
\end{aligned}$$

where the goal state $x_F = \mathrm{col}_{i\in\mathcal{M}}(x_{i,F})$ is assumed to be a feasible equilibrium state of (2.1). $x_S = \mathrm{col}_{i\in\mathcal{M}}(x_{i,S})$ is the initial condition of the system, and the state and input constraint sets in (2.5d) are the Cartesian products of the local constraint sets in (2.2). In this work, we are specifically interested in the minimum-time cost, which is defined as follows

$$h(x_t, u_t) = \mathbb{1}(x; x_F) = \begin{cases} 0 & \text{if } x = x_F \\ 1 & \text{otherwise.} \end{cases}$$

Learning Model Predictive Control

In this section, we briefly introduce and review the key concepts of LMPC [113, 114], which are extended upon in this work. LMPC was proposed as an iterative trajectory optimization method for single-agent nonlinear dynamical systems performing iterative tasks. In particular, the method provides a data-driven approach to terminal set and cost synthesis. We assume that an initial feasible input sequence $\mathbf{u}^0 = \{u_0^0, u_1^0, \ldots, u_{T^0-1}^0\}$ and closed-loop state trajectory $\mathbf{x}^0 = \{x_0^0, x_1^0, \ldots, x_{T^0}^0\}$ exists for (2.1) and is available at iteration $q = 0$. T^q denotes the time at which the closed-loop system reaches the terminal state, i.e. $x_{T^q}^q = x_F$. We note that for iterative tasks, the initial condition of the system is assumed to be the same over iterations, i.e. $x_0^q = x_S,\ \forall q \geq 0$.

LMPC solves a Finite Horizon Optimal Control Problem (FHOCP), which approximates (2.5), in a receding horizon fashion. Given the initial condition x at time t of iteration q, the FHOCP is defined as

$$\begin{aligned} \min_{\mathbf{u}_{t,N}^q} \quad & \sum_{k=0}^{N-1} \mathbb{1}(x_{k|t}^q; x_F) + V^{q-1}(x_{N|t}^q) && (2.6a) \\ \text{s.t.} \quad & x_{k+1|t}^q = f(x_{k|t}^q, u_{k|t}^q),\ \forall k \in \mathbb{N}_{N-1} && (2.6b) \\ & x_{0|t}^q = x && (2.6c) \\ & x_{k|t}^q \in \mathcal{X},\ u_{k|t}^q \in \mathcal{U},\ \forall k \in \mathbb{N}_{N-1} && (2.6d) \\ & g(x_{k|t}^q) \preceq 0,\ \forall k \in \mathbb{N}_{N-1} && (2.6e) \\ & x_{N|t}^q \in \mathcal{SS}^{q-1}, && (2.6f) \end{aligned}$$

where $x_{k|t}^q$ and $u_{k|t}^q$ denote the decision variables of the predicted state and input at the sampling time $t + k$ of iteration q. The first input $u_{0|t}^q$ is then applied to the system (2.1).

The terminal set at iteration $q - 1$ in (2.6f), called a sampled safe set, is defined as

$$\mathcal{SS}^{q-1} = \left\{ \bigcup_{p \in \mathcal{Q}^{q-1}} \bigcup_{t=0}^{T^p} x_t^p \right\}, \tag{2.7}$$

where $\mathcal{Q}^{q-1} = \{p \in \{0, \ldots, q-1\} : x_{T^p}^p = x_F\}$ is the set of iteration indices up to iteration $q-1$ where the goal state x_F was successfully reached. The sampled safe set collects the closed-loop state trajectories from previous successful iterations, which implies that at iteration q,

$\forall x \in \mathcal{SS}^q$, there exists a known and feasible sequence of inputs, with $x_0 = x$, such that $f(x_t, u_t) \in \mathcal{SS}^q$, $\forall t \geq 0$. We note that by construction, $\mathcal{SS}^q$ is a control invariant set.

For the terminal cost function in (2.6a), an approximate value function is constructed which returns the minimum cost-to-go (over iterations 0 to $q-1$) from each state in the safe set. The cost-to-go from the state at time t along the closed-loop trajectory $\mathbf{x}^q$ with corresponding input sequence $\mathbf{u}^q$ is defined as $J^q_{T^q}(x^q_t, t) = \sum_{k=t}^{T^q} \mathbb{1}(x^q_k; x_F)$.

The approximate value function at iteration $q-1$ is then

$$V^{q-1}(x) = \begin{cases} \min_{(p,t)\in\mathcal{F}^{q-1}(x)} J^p_{T^q}(x,t) & \text{if } x \in \mathcal{SS}^{q-1} \\ +\infty & \text{otherwise} \end{cases} \tag{2.8}$$

where $\mathcal{F}^{q-1}(x) = \{(p,t) : x^p_t = x \text{ and } x^p_t \in \mathcal{SS}^{q-1}\}$ returns the set of iteration and time index pairs for the states in previous trajectories which are equal to x.

Using the properties of the constructed terminal components, the resulting scheme guarantees persistent feasibility of the FHOCP and finite-time closed-loop convergence of the closed-loop system. Non-decreasing performance can also be shown over iterations of task execution.

2.3 Problem Formulation

In this section, we extend the idea of LMPC from Sec. 2.2 to the multi-agent case. Specifically, our formulation leverages data from previous iterations of task execution to synthesize local controllers for each agent via decoupled FHOCPs. This allows for an entirely decentralized control scheme for the multi-agent system at each iteration. In the following, we describe the synthesis of the local terminal components, namely the time-varying sampled safe sets and approximate value function. We then present the resulting decoupled FHOCPs, which are solved in a receding horizon manner. Combining these elements, we arrive at a decentralized LMPC procedure for trajectory optimization of multi-agent systems.

Time-Varying Sampled Safe Sets and Global Constraint Decomposition

In this part, we address two challenges posed by a decentralized receding horizon approach. Namely 1) how global constraint satisfaction can be enforced through decentralized local constraint satisfaction, and 2) how feasibility over the entire task horizon can be maintained in a receding horizon implementation of (2.5), particularly when time-varying constraints are introduced.

To address 1), we introduce the following assumption. Notice that the original global time-invariant constraints are transformed into local time-varying constraints.

Assumption 1. At iteration q, there exists a time-varying local decomposition of the global constraints: $g^q_{i,t}(\cdot)$, $\forall t \geq 0$, $\forall i \in \mathcal{M}$, which can be constructed from feasible trajectories of the global system, At each time t, joint local constraint satisfaction is sufficient for global constraint satisfaction, i.e. $g^q_{i,t}(x_{i,t}) \preceq 0,\ \forall i \in \mathcal{M} \implies g(x_t) \preceq 0$.

Remark 1. Given a feasible trajectory $\mathbf{x}^q_i$, we may always obtain the decomposition $g^q_{i,t}(x_i) = \|x_i - x^q_{i,t}\|$ for each trajectory point in $\mathbf{x}^q_i$, which satisfies the condition in Assumption 1. In Sec. 2.5, we present a technique for constructing $g^q_{i,t}(\cdot)$ in a less conservative manner.

To address 2), constraint satisfaction of the FHOCPs over the entire task horizon is obtained by using control invariant sets. This was done in [113] via (2.7), which, recall, are control invariant sets by definition. We would like to achieve the same goal of using data to construct sets which grant the same properties to the decentralized scheme. However, in (2.5) (and due to the constraint decomposition in Assumption 1), we include time-varying constraints for which the classical definition of set invariance does not apply. For this reason, we do not collect all recorded states of the global system, which converge to x_F, into a single invariant set as in (2.7). Instead, we interpret the states of agent i at time t of previous successful iterations as a sampled subset of the $(\hat{T}^{q-1} - t)$-step reachable set to $x_{i,F}$, where $\hat{T}^q = \max_{p\in\{0,\dots,q\}}(T^p_i)$. As such, for each agent $i \in \mathcal{M}$ at iteration q, we propose to construct time-varying sampled safe sets $\mathcal{SS}^q_{i,t}$ using data from previous iterations of task execution, which are one-step reachable to $\mathcal{SS}^q_{i,t+1}$ (and therefore K-step reachable to $x_{i,F}$ for some $K \in \{1, \dots, \hat{T}^{q-1} - t\}$). Moreover, we require that the global coupling constraints are satisfied for all combinations of states in the constructed safe sets, i.e. $\forall t \geq 0$ and $\bar{x}_i \in \mathcal{SS}^q_{i,t}$, $g(\bar{x}) \preceq 0$, where $\bar{x} = \mathrm{col}_{i\in\mathcal{M}}(\bar{x}_i)$. The following assumption allows us to start with non-empty safe sets at the first iteration.

Assumption 2. At iteration $q = 0$, an initial feasible input and state sequence, which converges to the goal state in T^0_i steps, exists for each agent (2.4). Denote this input sequence and state trajectory for agent $i \in \mathcal{M}$ as $\mathbf{u}^0_i$ and $\mathbf{x}^0_i$.

Remark 2. This assumption is reasonable as one may provide initial feasible trajectories through demonstration or by employing a conservative controller.

Let $(\mathcal{D}_x, \mathcal{D}_u)$ be the dataset which collects the state trajectories $\mathbf{x}^0_i \dots, \mathbf{x}^q_i$ and input sequences $\mathbf{u}^0_i, \dots, \mathbf{u}^q_i$, and let $g^q_{i,t}(\cdot)$ be a constraint decomposition which satisfies Assumption 1. Then, initializing the recursive relationship with $\mathcal{SS}^q_{i,\hat{T}^{q-1}} = \{x_{i,F}\}$, we construct the time-varying sampled safe sets for each agent i at time t of iteration q as

$$\mathcal{SS}^q_{i,t} = \left\{ x^p_{i,k} \in \mathbf{x} :\ g^q_{i,t}(x^p_{i,k}) \preceq 0 \text{ and} f_i(x^p_{i,k}, u^p_{i,k}) \in \mathcal{SS}^q_{i,t+1} \right\}. \tag{2.9}$$

The time-varying sampled safe sets $\mathcal{SS}^q_{i,t}$ collect the state trajectory points from previous successful iterations for which there exists a trajectory to the goal state that satisfies the global constraints. We show in Sec. 2.4 that the sampled safe sets are a non-empty family of reachable sets to $x_{i,F}$.

Remark 3. For successful iterations, i.e. where $x^q_{i,T^q_i} = x_{i,F}$, it is straightforward to accommodate task iterations of different lengths when constructing $\mathcal{SS}^q_{i,t}$ for $t \in \{0, \dots, \max_{p\in\mathcal{I}}(T^p_i)\}$.

Since $x_{i,F}$ is an equilibrium point of system i, we may trivially apply the zero input to obtain $\mathcal{SS}^q_{i,t} = \{x_{i,F}\}$ for all $t > T^q_i$.

Remark 4. How the constraint decomposition is constructed depends on the specific problem at hand. In the multi-vehicle collision avoidance example shown in Sec. 2.5, we propose a procedure to construct hyperplanes which separate the position states of the sampled safe sets for pairs of agents.

Value Function Approximation

For an input sequence $\mathbf{u}^q_i$ and closed-loop state trajectory $\mathbf{x}^q_i$ of agent i at iteration q with length T^q_i, we define the cost-to-go from the state $x^q_{i,t}$ at time t along the closed-loop trajectory to be

$$J^q_{i,T^q_i}(x^q_{i,t}, t) = \sum_{k=t}^{T^q_i} \mathbb{1}(x^q_{i,k}, x_{i,F}) = T^q_i - t. \tag{2.10}$$

The iteration cost for agent i at iteration q can then be written as $J^q_{i,T^q_i}(x_{i,S}, 0)$, recalling that the system is initialized at the same state x_S for each iteration q. This leads to the definition of the approximate value function $V^q_i(\cdot, t)$ at time t over the sampled safe set $\mathcal{SS}^q_{i,t}$ as

$$V^q_i(x_i, t) = \begin{cases} \min_{(p,k)\in\mathcal{F}^q_{i,t}(x_i)} J^p_{i,T^p_i}(x_i, k), & \text{if } x_i \in \mathcal{SS}^q_{i,t} \\ +\infty, & \text{otherwise,} \end{cases} \tag{2.11}$$

where $\mathcal{F}^q_{i,t}(x_i)$ is defined in the same way as in (2.8). Thus, for a state x_i whose value is equal to some state in $\mathcal{SS}^q_{i,t}$, $V^q_i(x_i, t)$ returns the minimum cost-to-go over trajectories which pass through x_i in $\mathcal{SS}^q_{i,k}$ for $k \geq t$. Note that V^q_i is discontinuous as its domain is a finite set of points.

The Finite Horizon Optimal Control Problem

Synthesis of the terminal components is achieved using the function `synthesizeFHOCP` as summarized in Alg. 1, which acts as a global coordinator between iterations of task execution in our decentralized framework. In this algorithm, we introduce the design parameters $\bar{t}, \underline{t} \geq 0$ and $\underline{q} \in \{0, \dots, q\}$ to reduce the computational burden of safe set construction and to make satisfaction of one-step reachability from $\mathcal{SS}^q_{i,t}$ to $\mathcal{SS}^q_{i,t+1}$ trivial. In particular, we construct the set $\mathcal{I}$ which contains the iteration indices of the $\underline{q}$ most recent successful iterations and the sliding time range $\mathcal{T}_t = \{\max(t - \underline{t}, 0), \dots, t, \dots, t + \bar{t}\}$ and obtain the *candidate* safe sets for time t (line 6). After a constraint decomposition is constructed (line 8, see Rem. 4 and Sec. 2.5), the candidate safe sets are checked for constraint satisfaction (line 10). In the case where any constraint is violated, the design parameters are updated (lines 11-15) and new candidate safe sets are constructed (line 16). This procedure is repeated until safe sets satisfying (2.9) are found. We will show in Sec. 2.4 that the procedure described in Alg. 1

results in non-empty safe sets which satisfy the coupled constraints and are reachable to $x_{i,F}$. As such, after performing the task at iteration $q-1$, we obtain for agent i the decomposition of the global constraint $g_{i,t}^{q-1}(\cdot)$, the sampled safe sets $\mathcal{SS}_{i,t}^{q-1}$, and the approximate value function $V_i^{q-1}(\cdot, t)$, which are constructed using data from iterations 0 to $q-1$.

To obtain the control action for agent i at sampling time t of iteration q, we solve the following decoupled FHOCP with horizon length N and initial condition x_i.

$$\begin{aligned}
\mathcal{P}_{i,N}^q(x_i, t) = \min_{\mathbf{u}_{i,t}^q} \quad & \sum_{k=0}^{N-1} \mathbb{1}(x_{i,k|t}^q; x_{i,F}) + V_i^{q-1}(x_{i,N|t}^q, t+N) \\
\text{s.t.} \quad & x_{i,k+1|t}^q = f_i(x_{i,k|t}^q, u_{i,k|t}^q), && (2.12a) \\
& x_{i,0|t}^q = x_i && (2.12b) \\
& x_{i,k|t}^q \in \mathcal{X}_i, \; u_{i,k|t}^q \in \mathcal{U}_i, && (2.12c) \\
& g_{i,t+k}^{q-1}(x_{i,k|t}^q) \preceq 0, && (2.12d) \\
& x_{i,N|t}^q \in \mathcal{SS}_{i,t+N}^{q-1} && (2.12e) \\
& \forall k \in \{0, \dots, N-1\}
\end{aligned}$$

where (2.12a) and (2.12b) represent the system dynamics and initial condition, respectively. The local state and input constraints are given in (2.12c). (2.12d) enforces satisfaction of the decomposed constraint for each agent, which is sufficient for global constraint satisfaction. Finally, (2.12e) ensures that the terminal state is a member of the time-varying sampled safe set $\mathcal{SS}_{i,t+N}^{q-1}$. We denote the locally optimal value of the FHOCP cost in (2.12) as $J_{i,N}^{*,q}(x_i, t)$.

Let $\mathbf{u}_{i,t}^{*,q}(x_i) = \{u_{i,0|t}^{*,q}(x_i), \dots, u_{i,N-1|t}^{*,q}(x_i)\}$ denote the input sequence which minimizes (2.12) for initial state x_i at sampling time t, and $\mathbf{x}_{i,t}^{*,q} = \{x_{i,0|t}^{*,q}, \dots, x_{i,N|t}^{*,q}\}$ be the corresponding state trajectory beginning at $x_{i,0|t}^{*,q} = x_i$. In typical *receding horizon* fashion, for each agent $i \in \mathcal{M}$, the first element of $\mathbf{u}_{i,t}^{*,q}(x_i)$ is applied to system (2.4), which defines the state feedback policy

$$u_{i,t}^q = \kappa_i^q(x_i, t) \doteq u_{i,0|t}^{*,q}(x_i), \tag{2.13}$$

with $x_i = x_{i,t}^q$. This results in the closed-loop state trajectory $\mathbf{x}_i^q = \{x_{i,0}^q, x_{i,1}^q, \dots, x_{i,t}^q, \dots\}$ and input sequence $\mathbf{u}_i^q = \{u_{i,0}^q, u_{i,1}^q, \dots, u_{i,t}^q, \dots\}$ for agent i at iteration q. We will show that the closed-loop state trajectory under (2.13) converges to the goal state $x_{i,F}$ in finite-time.

Remark 5. Due to the construction of the sampled safe sets as collections of discrete trajectory points, $\mathcal{P}_{i,N}^q(x_i, t)$ is a mixed integer nonlinear program, which can be computationally intensive to solve. However, the minimum-time formulation can be exploited for computational efficiency, e.g. by parallelization and branch and bound methods. Certain nonlinear systems may also admit a convex relaxation of the sampled safe set and approximate value function while maintaining performance guarantees [114].

Algorithm 1: `synthesizeFHOCP`

```
Input: $\mathcal{D}_x$, $\mathcal{D}_u$, $\mathbf{t}$, $\theta = \{\underline{q}, \underline{t}, \bar{t}\}$
1  $\check{q} \leftarrow \underline{q}$;
2  $\mathcal{I} \leftarrow \{\max(q - \underline{q}), \dots, q\}$;
3  for $t \in \{0, \dots, \max_{p \in \mathcal{I}}(T_i^p)\}$ do
4      for $i \in \mathcal{M}$ do
5          $\mathcal{T}_t \leftarrow \{\max(t - \underline{t}, 0), \dots, t, \dots, t + \bar{t}\}$;
6          $\mathcal{SS}_{i,t}^q \leftarrow \bigcup_{p \in \mathcal{I}} \{x_{i,k}^p \in \mathcal{D}_x : k \in \mathcal{T}_t\}$;
7      end
8      $\{g_{i,t}^q(\cdot)\}_{i \in \mathcal{M}} \leftarrow$ obtain decomposition according to Assumption 1;
9  end
10 while any constraint in $g_{i,t}^q(x)$ is violated for any
     $x \in \mathcal{SS}_{i,t}^q,\ i \in \mathcal{M},\ t \in \{0, \dots, \max_{p \in \mathcal{I}}(T_i^p)\}$ do
11     $\underline{q} \leftarrow \underline{q} - 1$;
12     if $\underline{q} = 0$ then
13         $\bar{t} \leftarrow \max(\bar{t} - 1, 0)$, $\underline{t} \leftarrow \max(\underline{t} - 1, 0)$;
14         $\underline{q} \leftarrow \check{q}$;
15     end
16     Repeat lines 2-9 with updated $\underline{t}$, $\bar{t}$ and $\underline{q}$;
17 end
18 $V_i^q(\cdot, t) \leftarrow$ compute as in (2.10) and (2.11) $\forall x_i^q \in \mathcal{SS}_{i,t}^q$;
Output: $\{\mathcal{SS}_{i,t}^q, V_i^q(\cdot, t), g_{i,t}^q(\cdot)\}$
```

Decentralized Learning Model Predictive Control

The resulting iterative LMPC scheme for the multi-agent system is described in Alg. 2, where the for loop beginning at line 3 corresponds to the iterations of the decentralized LMPC, the for loop over agents from lines 4 to 14 may be executed in an entirely decentralized manner with no communication between agents, and the while loop from lines 6 to 11 iterates over time steps. Note that we assume that there is no mismatch between the model used in (2.12a) and the system on which the policy is applied (line 8). The inputs to Alg. 2 are $Z \geq 1$: the number of LMPC iterations, $\mathbf{x}^0$ and $\mathbf{u}^0$: the initial feasible state and input sequences, $\{T_i^0\}_{i \in \mathcal{M}}$: the lengths of the initial trajectories, and θ which contains the design parameters for `synthesizeFHOCP`. It is implicitly assumed that each iteration is successful. We will show that this is true in Sec. 2.4.

Algorithm 2: Decentralized Learning Model Predictive Control

```
   Input: $Z$, $\mathbf{x}^0$, $\mathbf{u}^0$, $\{T_i^0\}_{i\in\mathcal{M}}$, $\theta$
1  $\mathcal{D}_x \leftarrow \{\mathbf{x}^0\}$, $\mathcal{D}_u \leftarrow \{\mathbf{u}^0\}$, $\mathbf{t} \leftarrow \{T_i^0\}_{i\in\mathcal{M}}$;
2  $\{\mathcal{SS}_{i,t}^0, V_i^0(\cdot,t), g_{i,t}^0(\cdot)\} \leftarrow$ synthesizeFHOCP($\mathcal{D}_x, \mathcal{D}_u, \mathbf{t}, \theta$);
3  for $q \in \{1,\dots,Z\}$ do
4      for $i \in \mathcal{M}$ in parallel do
5          $t \leftarrow 0$, $x_{i,0}^q \leftarrow x_{i,S}$, $\mathbf{x}_i^q \leftarrow \{x_{i,S}\}$, $\mathbf{u}_i^q \leftarrow \emptyset$;
6          while $x_{i,t}^q \neq x_{i,F}$ do
7              $\kappa_i^q(\cdot,t) \leftarrow$ solve $\mathcal{P}_{i,N}^q(x_{i,t}^q, t)$;
8              $x_{i,t+1}^q \leftarrow$ apply $u_{i,t}^q = \kappa_i^q(x_{i,t}^q, t)$ and measure state;
9              $\mathbf{x}_i^q \leftarrow \mathbf{x}_i^q \cup \{x_{i,t+1}^q\}$, $\mathbf{u}_i^q \leftarrow \mathbf{u}_i^q \cup \{u_{i,t}^q\}$;
10             $t \leftarrow t+1$;
11         end
12         $\mathcal{D}_x \leftarrow \mathcal{D}_x \cup \{\mathbf{x}_i^q\}$, $\mathcal{D}_u \leftarrow \mathcal{D}_u \cup \{\mathbf{u}_i^q\}$;
13         $T_i^q \leftarrow t$
14     end
15     $\mathbf{t} \leftarrow \mathbf{t} \cup \{T_i^q\}_{i\in\mathcal{M}}$;
16     $\{\mathcal{SS}_{i,t}^q, V_i^q(\cdot,t), g_{i,t}^q(\cdot)\} \leftarrow$ synthesizeFHOCP($\mathbf{x}, \mathbf{u}, \mathbf{t}, \theta$);
17 end
```

2.4 Properties of the Decentralized LMPC

In this section, we show that for each iteration q, the sampled safe sets as constructed in (2.9) have the property of reachability to $x_{i,F}$, which is sufficient for persistent feasibility of the decentralized LMPC over the entire task horizon for all iterations q. Furthermore, we show that the closed-loop system converges to the goal state in finite time, and that task performance is non-decreasing over iterations.

Proposition 1. *Let Assumption 2 hold, then for agent i at iteration q, the sampled safe sets as constructed in (2.9) using Alg. 1 and the dataset $(\mathbf{x}, \mathbf{u})$ are reachable to $x_{i,F}$ and satisfy the decomposed constraints $g_{i,t}^q(x) \succeq 0$, $\forall x \in \mathcal{SS}_{i,t}^q$.*

Proof. We first assume that $\underline{t}$, $\bar{t}$, and $\mathcal{I}$ are chosen according to Alg. 1 such that the constructed sampled safe sets satisfy the decomposed constraints for all sample times t. Now, at time t and $t+1$, we obtain the time ranges $\mathcal{T}_t = \{\max(t-\underline{t},0),\dots,t,\dots,t+\bar{t}\}$ and $\mathcal{T}_{t+1} = \{\max(t-\underline{t}+1,0),\dots,t+1,\dots,t+\bar{t}+1\}$, respectively. For the case when $0 \le t \le \underline{t}-1$, i.e. both lower limits evaluate to zero, the sampled safe sets at time t and $t+1$ contain $\{x_{i,0}^p,\dots,x_{i,t+\bar{t}}^p\}$ and $\{x_{i,0}^p,\dots,x_{i,t+\bar{t}+1}^p\}$, respectively, for all $p \in \mathcal{I}$. Recall that $\mathcal{I}$ contains indices corresponding to successful iterations, i.e. $x_{i,T_i^p}^p = x_{i,F}$, $\forall p \in \mathcal{I}$. Moreover, by Assumption 2, $\mathcal{I}$ must be non-empty. By the fact that the feasible input sequence $\{u_{i,0}^p,\dots,u_{i,t+\bar{t}}^p\}$ exists in the dataset $\mathbf{u}$, such that $x_{i,k+1}^p = f_i(x_{i,k}^p, u_{i,k}^p)$, $\forall k \in \{0,\dots,t+\bar{t}\}$, we see that the property of reachability to $x_{i,F}$ holds. For the case when $t \geq \underline{t}$, the sampled safe sets at time t and $t+1$ contain $\{x^p_{i,t-\underline{t}}, \dots, x^p_{i,t+\bar{t}}\}$ and $\{x^p_{i,t-\underline{t}+1}, \dots, x^p_{i,t+\bar{t}+1}\}$. By the same argument as before, we see that reachability holds. We conclude that the sampled safe sets for agent i at iteration q are reachable to $x_{i,F}$ and satisfy the decomposed constraints for all $t \geq 0$. □

Proposition 2. *Let Assumption 2 hold and assume that $\mathcal{P}^q_{i,N}(x^q_{i,t}, t)$ defined in* (2.12) *is feasible for agent i at time t and iteration q for some $x^q_{i,t} \in \mathcal{X}_i$. If $x_{i,F} \in \mathcal{SS}^{q-1}_{i,t+N}$, then the system* (2.4) *in closed-loop with* (2.12) *and* (2.13) *converges in at most $t + \bar{T}$ steps to $x_{i,F}$, where $\bar{T} = N + T^{p^*}_i - k^*$ and $(k^*, p^*) = \arg\min_{k,p} T^p_i$ subject to the constraint $x^{*,q}_{i,N|t} = x^p_{i,k}$.*

Proof. By assumption, $\mathcal{P}^q_{i,N}(x^q_{i,t}, t)$ is feasible at time t, which implies that there exists a sequence of feasible inputs $\bar{\mathbf{u}} = \{\mathbf{u}^{*,q}_{i,t}(x^q_{i,t}), u^{p^*}_{i,k^*}, \dots, u^{p^*}_{i,T^{p^*}_i - 1}\}$, which drives agent i to the goal state $x_{i,F}$ in $\bar{T}$ steps. Therefore,

$$J^{*,q}_{i,N}(x^q_{i,t}, t) = \bar{T} - 1. \tag{2.14}$$

Since $\mathcal{SS}^{q-1}_{i,t+N} \ni x_{i,F}$, it follows from standard MPC arguments and Prop. 1 that $\mathcal{P}^q_{i,N}(\cdot, k)$ is feasible for time steps $k > t$ and

$$J^{*,q}_{i,N}(x^q_{i,t}, t) \geq \sum_{l=t}^{k} \mathbb{1}(x^q_{i,l}; x_{i,F}) + J^{*,q}_{i,N}(x^q_{i,k+1}, k+1), \tag{2.15}$$

where $x^q_{i,k+1} = f_i(x^q_{i,k}, \bar{\mathbf{u}}_{k-t})$. Assume that $x^q_{i,k} \neq x_{i,F}$ for $k = \{t, \dots, t + \bar{T} - 1\}$. Then at time step $k = t + \bar{T} - 1$, using (2.14) and (2.15) we have

$$\begin{aligned} J^{*,q}_{i,N}(x^q_{i,k+1}, k+1) &\leq J^{*,q}_{i,N}(x^q_{i,t}, t) - \sum_{l=t}^{k} \mathbb{1}(x^q_{i,l}; x_{i,F}) \\ &= (\bar{T} - 1) - (\bar{T} - 1) = 0 \end{aligned}$$

which implies that $x^q_{i,k+1} = x^q_{i,t+\bar{T}} = x_{i,F}$. □

Theorem 1. *Consider the system in* (2.1) *in closed loop with the decentralized LMPC* (2.12) *and* (2.13)*. Let Assumption 1 and 2 hold. Then at each iteration q for every agent $i \in \mathcal{M}$:*

1. *The decentralized LMPC* (2.12) *is feasible for all time steps $t \geq 0$.*
2. *The system* (2.4) *in closed-loop with* (2.12) *and* (2.13) *converges to the equilibrium point $x_{i,F}$ in finite time.*
3. *The sampled safe sets $\mathcal{SS}^q_{i,t}$ are non-empty for all sample times t.*

Proof. At iteration $q = 1$, by Assumption 1 and 2, the sampled safe sets $\mathcal{SS}^0_{i,t}$ are non-empty for all $t \in \mathbb{N}$ and (2.12) is feasible at $t = 0$.

From Proposition 1, we have that at iteration q, for each agent i, the terminal constraint sets $\mathcal{SS}^{q-1}_{i,t}$ are reachable to $x_{i,F}$ and satisfy the decomposed constraints for all $t \in \{0, \ldots, \max_{p\in\mathcal{I}}(T^p_i)\}$. Additionally at time $t = 0$, (2.12) is feasible and therefore the proof for 1) follows from standard MPC arguments [25].

Again, leveraging the reachability property of $\mathcal{SS}^{q-1}_{i,t}$, $\exists k \in \{0, \ldots, \max_{p\in\mathcal{I}}(T^p_i)\}$ such that $x_{i,F} \in \mathcal{SS}^{q-1}_{i,k}$. Since from 1) we have that (2.12) is feasible for all $t \geq 0$, 2) follows immediately after applying Proposition 2.

From 1) and 2), task execution at iteration q is successful, i.e. $x^q_{i,T^q_i} = x_{i,F}$. Therefore, by Asssumption 1, we may construct $\mathcal{SS}^q_{i,t} \ni x^q_{i,t}$ using Alg. 1, which shows 3). Repeating the argument for each iteration q concludes the proof. □

Corollary 1. *Consider the system (2.1) in closed loop with the decentralized LMPC (2.12) and (2.13). Let Assumption 2 hold. Then for every agent $i \in \mathcal{M}$, the task completion time does not increase with the iteration index q, i.e. $T^q_i \leq T^{q-1}_i$.*

Proof. Let $x^q_{i,t}$ and $u^q_{i,t}$ be the elements from the closed-loop state trajectory and corresponding input sequence at iteration q. From (2.10), we have that the iteration cost at iteration $q-1$ can be written as

$$\begin{aligned} J^{q-1}_{i,T^{q-1}_i}(x_{i,S}, 0) &= \sum_{t=0}^{N-1} \mathbb{1}(x^{q-1}_{i,t}; x_{i,F}) + \sum_{t=N}^{T^{q-1}_i} \mathbb{1}(x^{q-1}_{i,t}; x_{i,F}) \\ &\geq \sum_{t=0}^{N-1} \mathbb{1}(x^{q-1}_{i,t}; x_{i,F}) + V^{q-1}_i(x^{q-1}_{i,N}, N) \\ &\geq J^{*,q}_{i,N}(x_{i,S}, 0). \end{aligned} \tag{2.16}$$

Next, using (2.15), we have that at $t = 0$

$$J^{*,q}_{i,N}(x^q_{i,0}, 0) \geq \sum_{l=0}^{T^q_i} \mathbb{1}(x^q_{i,l}; x_{i,F}) = J^q_{i,T^q_i}(x_{i,S}, 0). \tag{2.17}$$

From (2.16) and (2.17), and using the fact that for agent i, $x^q_{i,0} = x_{i,S}$, we arrive at the final bounds

$$J^q_{i,T^q_i}(x_{i,S}, 0) \leq J^{*,q}_{i,N}(x_{i,S}, 0) \leq J^{q-1}_{i,T^{q-1}_i}(x_{i,S}, 0),$$

which implies that $T^q_i \leq T^{q-1}_i$. □

2.5 Multi-Vehicle Collision Avoidance

In this section, we present a numerical example of decentralized LMPC in the context of multi-vehicle collision avoidance. We compare the results from the decentralized LMPC with those from a centralized approach. The control objective is for $M = 3$ vehicles to reach the goal equilibrium points ζ_F^i from their respective initial states ζ_S^i in minimum time (the code is available online[1,2]).

Agent Model

We model the vehicles using the kinematic bicycle model, which is discretized using forward Euler integration with a time step of $dt = 0.1$s as in [73].

$$\zeta_{i,t+1} = \zeta_{i,t} + dt \begin{bmatrix} v_{i,t}\cos(\psi_{i,t} + \beta_{i,t}) \\ v_{i,t}\sin(\psi_{i,t} + \beta_{i,t}) \\ v_{i,t}\sin(\beta_{i,t})/l_r \\ a_{i,t} \end{bmatrix} = f_i(\zeta_{i,t}, u_{i,t}),$$

where $\beta_{i,t} = \arctan(l_r \tan(\delta_{i,t})/(l_f + l_r))$.

The state and input variables are $\zeta_{i,t} = [x_{i,t}, y_{i,t}, \psi_{i,t}, v_{i,t}]^\top \in \mathbb{R}^4$ (with units m, m, rad, m/s) and $u_{i,t} = [\delta_{i,t}, a_{i,t}]^\top \in \mathbb{R}^2$ (with units rad, m/s^2) respectively. The vehicles are coupled via collision avoidance constraints where we define a circular collision buffer about the geometric center of the vehicle with radius r_i and require that for all sample times,

$$\|\zeta_{i,t}(1{:}2) - \zeta_{j,t}(1{:}2)\|_2 \geq r_i + r_j,\ \forall i, j \in \mathcal{M},\ i \neq j, \tag{2.18}$$

Decoupled FHOCP Formulation

The decoupled FHOCP for agent i at time t and iteration q given initial condition ζ_i is formulated as in (2.12). For the local state and input constraints in (2.12c), in addition to the box constraints $|\zeta_{i,k|t}^q| \preceq [10, 10, \infty, 10]^\top$ and $|u_{i,k|t}^q| \preceq [0.5, 3]^\top$, $\forall k \in \mathbb{N}_{N-1}$, we also impose constraints on the control rate via $|u_{i,0|t}^q - u_{i,t-1}^q| \preceq dt \cdot [0.7, 7]^\top$ and $|u_{i,k+1|t}^q - u_{i,k|t}^q| \preceq dt \cdot [0.7, 7]^\top$, $\forall k \in \mathbb{N}_{N-2}$. Here, $|\cdot|$ represents the element-wise absolute value.

In (2.12d), we decompose the global constraints into time-varying hyperplane constraints on the position states of each vehicle, i.e. $g_{i,t+k}^{q-1}(\zeta_{i,k|t}^q) = H_{i,t+k}^{q-1}\zeta_{i,k|t}^q + h_{i,t+k}^{q-1} \preceq 0$, with $H_{i,t+k}^{q-1} \in \mathbb{R}^{M-1\times 4}$ and $h_{i,t+k}^{q-1} \in \mathbb{R}^{M-1}$ for all $k \in \mathbb{N}_{N-1}$. This implementation of line 8 of Alg. 1 is achieved by solving a hard margin support vector machine (SVM) for all agent pairs $(i, j) \in \mathcal{M},\ i \neq j$, where maximum margin separating hyperplanes can be found for time t if the position states in the candidate sampled safe sets $\mathcal{SS}_{i,t}^{q-1}$ and $\mathcal{SS}_{j,t}^{q-1}$ are linearly separable. If all $\binom{M}{2}$ pairwise SVM problems are feasible for all time t and the distance between the

[1]https://github.com/zhu-edward/multi-agent-LMPC

[2]A video of the results may be found at https://youtu.be/cB9zckRm5j8

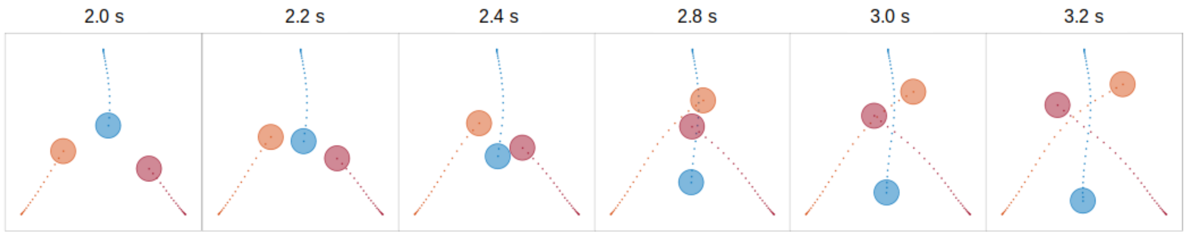


Figure 2.1: Snapshots of the decentralized LMPC trajectory at convergence. Circles represent the collision buffers centered at the position of agent 1 (blue), 2 (orange), and 3 (red).

Table 2.1: Parameter Values

$\zeta_{1,S}$	$(0, 5, -\pi/2, 0)$	$\zeta_{1,F}$	$(0, -5, -\pi/2, 0)$
$\zeta_{2,S}$	$(-5, -5, \pi/4, 0)$	$\zeta_{2,F}$	$(5, 5, \pi/4, 0)$
$\zeta_{3,S}$	$(5, -5, 3\pi/4, 0)$	$\zeta_{3,F}$	$(-5, 5, 3\pi/4, 0)$
l_f, l_r	0.5m	r	0.75m
N	20	Z	20
ϵ	1e-4	$\underline{q}$	2
$\bar{t}$	175	$\underline{t}$	0

hyperplanes is no less than $2r$ in all cases, then we construct the constraint decomposition for agent i at time t by stacking the solution H_{ij} and h_i from all SVM problems involving agent i into the matrix $H_{i,t}^{q-1}$ and vector $h_{i,t}^{q-1}$ respectively. Otherwise, following Alg. 1, we shrink the sets $\mathcal{T}_t$ and $\mathcal{I}$ and retry the SVM problem.

We compute the initial feasible state trajectory and input sequence $(\boldsymbol{\zeta}_i^0, \mathbf{u}_i^0)$ using a linear time-varying MPC controller. At each time step, the decoupled FHOCPs are solved using IPOPT [141] by constructing a set of $|\mathcal{SS}_{i,t+N}^{q-1}|$ problems where (2.12e) is formulated as equality constraints for each element in the safe set. Corollary 1 is useful here as it allows us to prune the safe sets using an upper bound on the iteration cost, which can be computed for each point in the safe sets without solving the FHOCP. Specifically, at each time t, we have the cost-to-come for the current state $x_{i,t}$, we know that the sum of the stage costs over the optimization horizon is upper bounded by the horizon length N, and the cost-to-go from each point in the safe set is known. If for any safe set point, the sum of these three quantities is greater than T_i^{q-1}, we may discard that point. This helps to manage the computational burden induced by the discrete safe sets. The parameters used for this experiment are shown in Table 2.1.

Results and Discussion

As seen in Table 2.2, the decentralized LMPC converges to a steady state solution where the optimal cost of each iteration is non-increasing. We additionally implemented a centralized LMPC, with the same parameters and solver, for the global system subject to the original

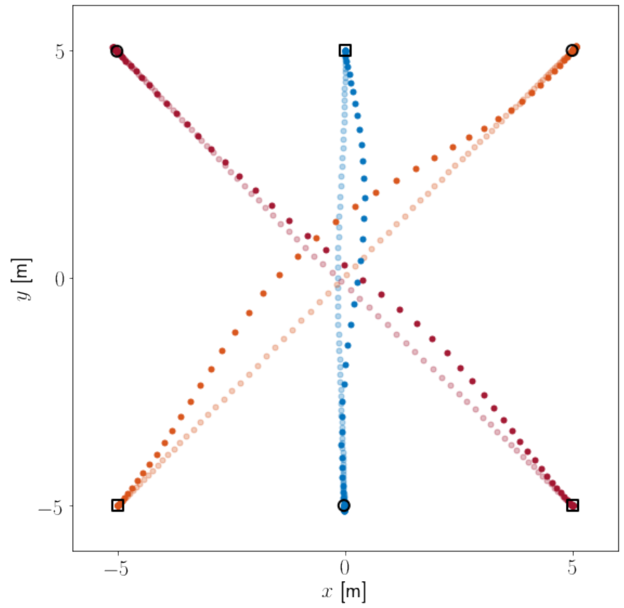


Figure 2.2: Initial (light) vs. steady state (dark) position trajectory (top). The starting and goal states are denoted as the square and circle markers respectively.

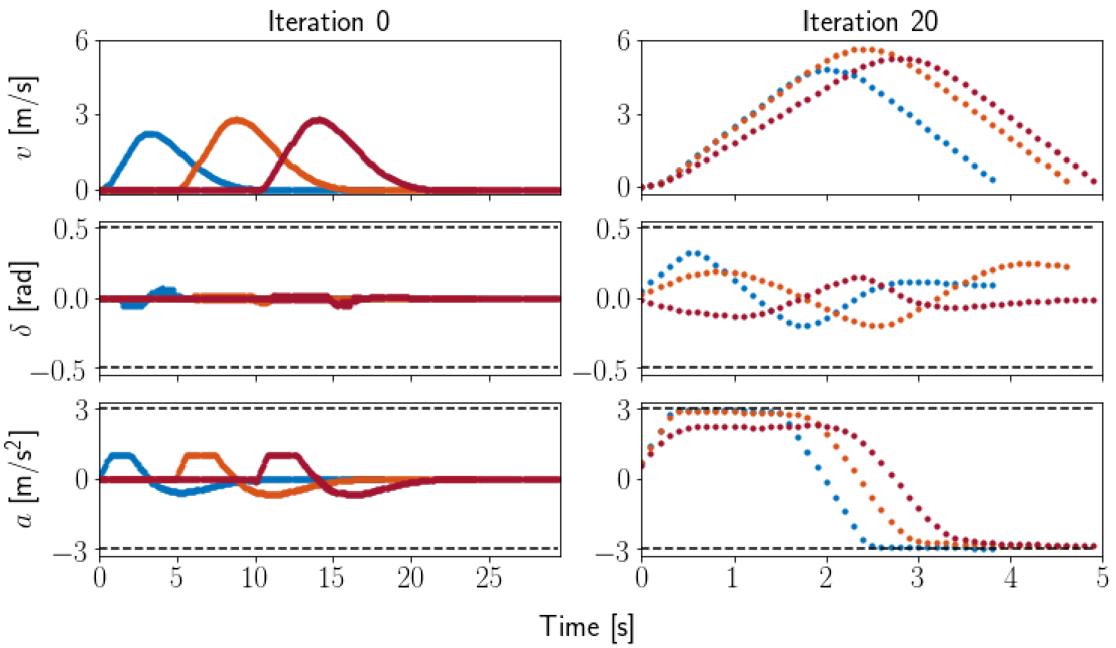


Figure 2.3: Velocity profile and input sequence (bottom) for agent 1 (blue), 2 (orange), and 3 (red) at the first and last iterations. The black dashed lines correspond to the input box constraints.

constraint (2.18). This approach achieved a steady state cost of 48, which is only about a 4% difference in cost with respect to the decentralized case. We also compare the computation time for solving a single FTOCP in the decentralized and centralized cases. This is summa-

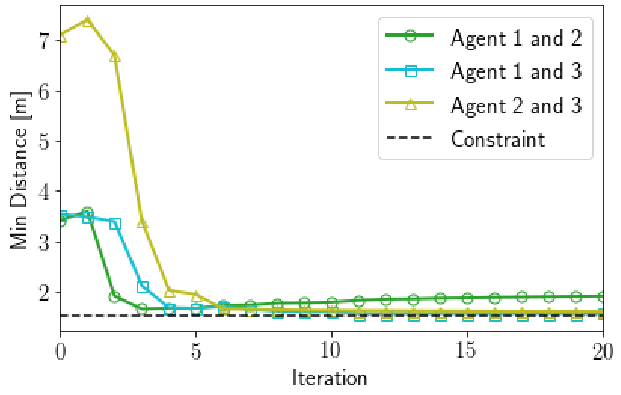


Figure 2.4: Minimum distance between agents at each iteration.

Table 2.2: Optimal Cost of the Decentralized LMPC at each Iteration

Iteration	Iteration Cost	Iteration	Iteration Cost
$q = 0$	296	$q = 5$	51
$q = 1$	122	$q = 6$	50
$q = 2$	79	$q = 7$	50
$q = 3$	55	$q = 8$	50
$q = 4$	51		

Table 2.3: FTOCP Solve Time

	Max Time [s]	Min Time [s]	Avg. Time [s]
Decentralized	10.2	1.97	3.35
Centralized	48.3	15.8	20.5

rized in Table 2.3. For the former, we record the maximum solve time over agents at each sampling time. For the latter, we record the solve time of the centralized FHOCP at each sampling time. We obtain that over all iterations, computation time of the decentralized case is lower by a factor of 4.6x to 24x.

In Figures 2.2 and 2.3, we compare the initial feasible trajectory to the steady state trajectory at convergence. In the initial feasible trajectory, the agents' movements are intentionally staggered in time to guarantee safety. This can be clearly seen in the velocity profile at iteration 0 in Figure 2.3. At convergence, all three agents begin moving simultaneously and steer to avoid collisions around the intersection point at the origin. We notice that in the steady state input sequence, the acceleration input either saturates or is close to saturating the imposed constraint and resembles a bang-bang controller [86] which switches between acceleration and deceleration at the midpoint of the trajectory.

In Figure 2.1, we look closely at the steady state trajectory about the intersection point

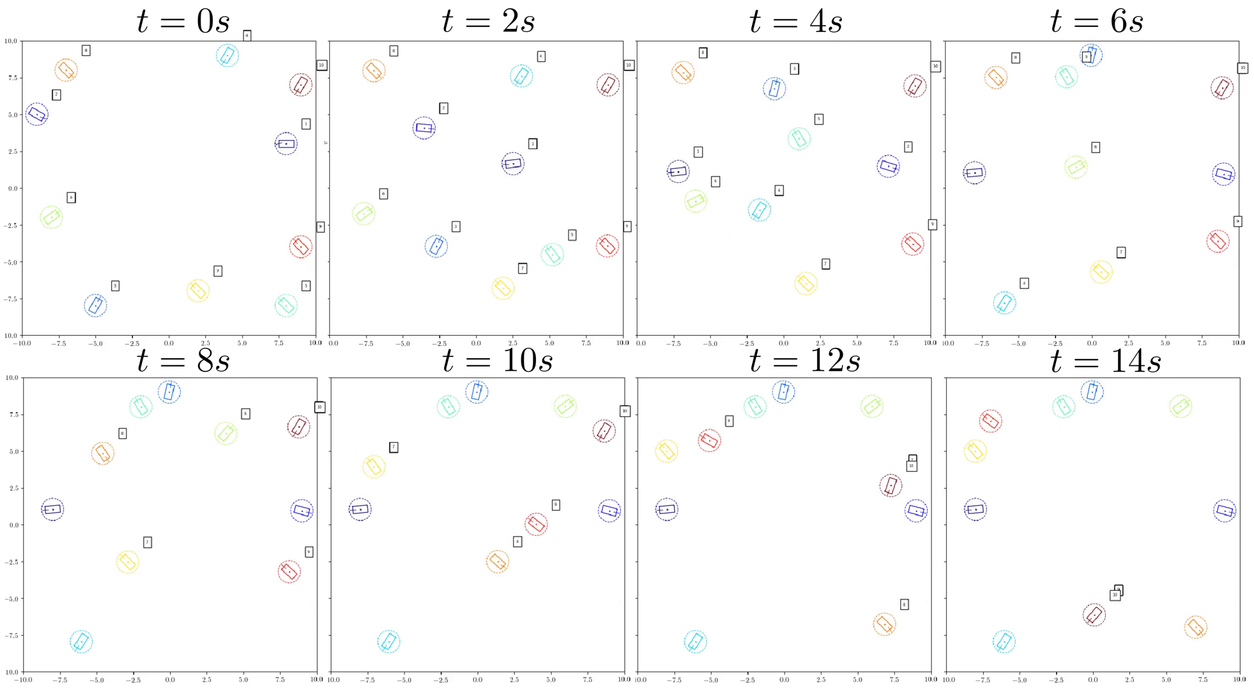


Figure 2.5: Initial feasible trajectories for the 10-agent example.

and see that the collision avoidance constraints are satisfied and are almost active for agents 1 and 3 at 2.4s and agents 2 and 3 at 2.8s. This is clearly reflected in Figure 2.4 which plots the minimum pairwise distance between the three agents over iterations of decentralized LMPC.

Finally, in Figures 2.5 and 2.6, we show the approach applied to a system with ten agents and see that we were able to achieve an over 50% reduction in task completion time for the overall system, going from 15.5 seconds for the initial feasible trajectories shown in Figure 2.5 to 7 seconds for the converged trajectories in Figure 2.6.

2.6 Chapter Summary

In this chapter, we presented a decentralized LMPC framework for dynamically decoupled multi-agent systems performing iterative tasks. In particular, we proposed a procedure for decomposing global constraints and synthesizing terminal sets and terminal cost functions for the FHOCP using data from previous iterations of task execution. We showed that the resulting decentralized LMPC has the properties of persistent feasibility, finite-time convergence to the goal state, and non-decreasing performance over iterations.

In the multi-vehicle collision avoidance example, due to the parallelization opportunities afforded by the decentralized implementation, we observe a significant improvement in computation time compared to a centralized approach with only a 4% increase in cost. In fact,

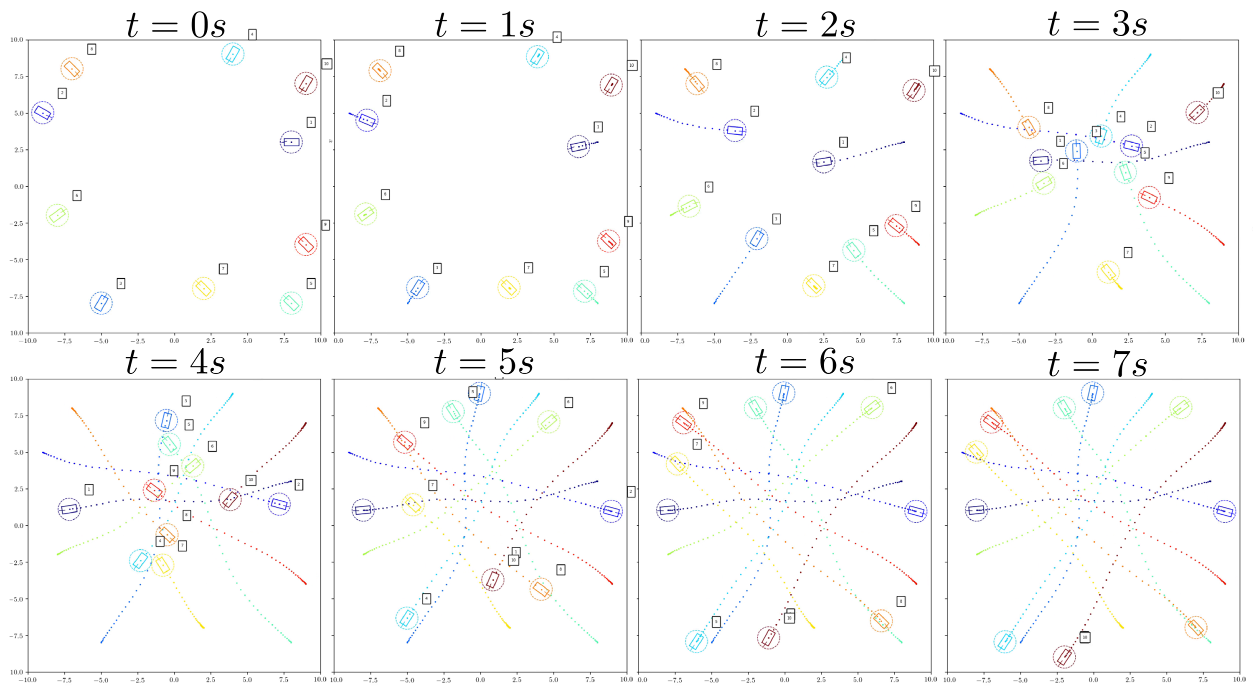


Figure 2.6: Trajectories at convergence for the 10-agent example.

the steady state solution from the decentralized approach saw saturation of the coupling collision avoidance constraint.

Moving forward, we would like to relax the assumption of perfect model knowledge and investigate approaches which leverage techniques in robust and stochastic optimal control to guarantee performance in the presence of model mismatch. In particular, an interesting direction is to extend the the work from [9, 8] to the multi-agent case, which uses tools from statistical identification to learn rich models of the system from data while maintaining guarantees on system safety and robustness.

We will next look at how a similar method can be developed for the case when the agents exhibit coupling through their dynamics functions in addition to their constraints. This will require communication during the iterations of the task, but we will show how this can be limited to only local communication between agents where this coupling exists.

Chapter 3

Distributed Learning Model Predictive Control for Multi-Agent Linear Systems

3.1 Introduction

Complex systems composed of multiple subsystems are present in many control applications. The large scale and spatial distribution of these systems often make the control by a centralized unit intractable due to limitations in computation and communication. Research has therefore focused on proposing design schemes for local controllers which compute control actions for the individual subsystems based on only local information in decentralized schemes, and on communicated information from neighboring subsystems in distributed control schemes. One line of research has focused on exploiting the interconnection structure of the system in order to design interconnected controllers based on a convex reformulation involving linear matrix inequalities in a scalable way [133]. If constraints need to be accounted for, distributed model predictive control (DMPC) techniques can be employed. They can mainly be categorized into non-cooperative, such as tube-based [109], and cooperative schemes [31, 55, 35, 37, 1]. The latter often involve distributed optimization techniques [16] where the subsystems communicate local information and agree on a solution, thus solving the optimization problem cooperatively.

The main challenge in DMPC schemes is to enable distributed computation by decomposing the optimization problem into subproblems for the individual subsystems. Most of the DMPC approaches in the literature therefore impose the distributed structure of the system on the terminal set and cost function of the MPC problem [109, 55, 35, 46, 137, 92]. In particular, they first design a structured terminal controller and cost based on Lyapunov stability and then design structured positive invariant sets under this terminal controller, satisfying the constraints. Two aspects in these schemes can lead to conservatism: (1) Imposing structure on the terminal controllers and terminal sets, and (2) computing positive invariant sets for one specific choice of terminal controller which is fixed in the design phase, lead to a possibly small inner approximation of the maximal control invariant set. In order to mitigate the conservatism introduced by the imposed structure, some works have proposed to adapt the terminal sets based on the states of the subsystems in operation [137, 92, 35]. In [37, 1], terminal sets also computed online within the MPC problem.

Recent research has focused on learning-based and data-driven DMPC schemes, [5, 162, 98, 63, 59]. In [112], a data-driven MPC scheme, Learning MPC (LMPC), was introduced, where previously seen data are exploited in order to construct the terminal components of the MPC problem. In [115] this framework was extended to uncertain systems, and it was shown how the LMPC scheme can be used to iteratively enlarge the domain of the policy.

In this chapter, we propose a distributed LMPC (DLMPC) scheme. The contributions of this chapter are the following:

- We present a novel DLMPC scheme for linear systems able to handle coupled dynamics, coupled state constraints and coupled cost functions. The main improvement w.r.t. existing DMPC approaches is fully distributed computations without imposing any structure on the terminal cost function or constraint set. This is achieved by exploiting previously seen local data by the individual subsystems in order to build local data driven terminal sets and terminal cost functions. A consensus on specific parameters in the construction of the local costs and constraints is achieved by distributed optimization which guarantees that the local terminal sets are a control invariant set and the sum of the local terminal cost functions is a Lyapunov function for the global system. This can considerably reduce conservatism w.r.t. DMPC schemes that rely on finding a positive invariant terminal set under a fixed structured stabilizing terminal controller.

- For iterative control tasks, given a first feasible trajectory, the proposed scheme provides recursive feasibility and asymptotic stability. Furthermore, we prove that the proposed DLMPC has a non-increasing control performance over iterations and, under mild conditions, converges to the global centralized optimal solution.

- For non-iterative control tasks, or if an initial feasible trajectory is difficult to obtain, we further present an algorithm that by iteratively performing the proposed DLMPC scheme with changing starting conditions leads to an enlargement of the domain of the DLMPC policy. This can be used to safely explore the state space and to generate the required data in a sample efficient and distributed way.

The results presented in this chapter have also appeared in:

- Y. Stürz, E. Zhu, U. Rosolia, K. Johannson, and F. Borrelli, "Distributed learning model predictive control for linear systems". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. 2020, pp. 4366-4373.

Notation: Let $\mathbb{R}$ denote the set of real numbers. $\mathbb{N}$ and $\mathbb{N}_+$ denote the set of non-negative and positive natural numbers. We denote the transpose of a vector $v \in \mathbb{R}^n$ as $v^\top$, and its Euclidean norm as $\|v\|$. The matrix $M = \text{diag}(M_1, ..., M_m)$ is the block-diagonal matrix with submatrices M_i on its diagonal. The symbol $\succcurlyeq$ is used to indicate elementwise inequality. The identitiy matrix of dimension n is denoted as I_n and the vector of all ones is denoted as $\mathbf{1}$.

3.2 Problem Formulation

In this section, we present the model of the distributed systems considered in this paper, and then state the control problem formulation.

Dynamically Coupled Constrained Linear Systems

We consider the discrete-time linear time-invariant system with dynamics given by

$$x_{t+1} = Ax_t + Bu_t, \tag{3.1}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the system state and input at time $t \in \mathbb{N}$. The system matrices A and B are assumed to be known. The system states and inputs are subject to linear constraints

$$x_t \in \mathcal{X}, \qquad u_t \in \mathcal{U}, \tag{3.2}$$

which are formulated as

$$Gx_t \leq g, \qquad Lu_t \leq l, \tag{3.3}$$

with G, L, g, and l given matrices and vectors, respectively.

We consider systems in (3.1) which have a structure that admits a decomposition into subsystems $\mathcal{N} = \{1, ..., M\}$ which may be coupled in their state dynamics. The state of the ith subsystem is $x_{i,t} \in \mathbb{R}^{n_i}$, and we assume that the ith input $u_{i,t} \in \mathbb{R}^{m_i}$ affects only the i-th state. Thus, the system states and inputs are partitioned as

$$x_t = \begin{bmatrix} x_{1,t}^\top & \dots & x_{M,t}^\top \end{bmatrix}^\top, \quad u_t = \begin{bmatrix} u_{1,t}^\top & \dots & u_{M,t}^\top \end{bmatrix}^\top. \tag{3.4}$$

For each subsystem $i \in \mathcal{N}$, we define the set of neighboring subsystems $\mathcal{N}_i \subseteq \mathcal{N}$ which contains all those subsystems that are coupled to subsystem i over the dynamics, constraints or cost. We define the state vector $x_{\mathcal{N}_i,t} \in \mathbb{R}^{n_{\mathcal{N}_i}}$ containing the local states of subsystem i and its neighboring subsystems in $\mathcal{N}_i$, which can be expressed as $x_{\mathcal{N}_i,t} = X_{\mathcal{N}_i} x_t$, with $X_{\mathcal{N}_i}$ being a projection matrix, i.e., a binary matrix $X_{\mathcal{N}_i} \in \{0,1\}^{n_{\mathcal{N}_i} \times n}$. Similarly, the projection matrices $X_i \in \{0,1\}^{n_i \times n}$ and $U_i \in \{0,1\}^{m_i \times m}$, are defined such that $x_{i,t} = X_i x_t$ and $u_{i,t} = U_i u_t$. The dynamics of subsystem i is then reformulated as

$$x_{i,t+1} = A_{\mathcal{N}_i} x_{\mathcal{N}_i,t} + B_i u_{i,t}, \quad \forall i \in N, \tag{3.5}$$

with

$$A_{\mathcal{N}_i} = X_i A X_{\mathcal{N}_i}^\top, \quad B_i = X_i B U_i^\top. \tag{3.6}$$

The local state and input constraints are defined as

$$\begin{aligned} x_{\mathcal{N}_i,t} &\in \mathcal{X}_{\mathcal{N}_i} = \{x_{\mathcal{N}_i,t} \in \mathbb{R}^{n_{\mathcal{N}_i}} : G_{\mathcal{N}_i} x_{\mathcal{N}_i} \leq g_{\mathcal{N}_i}\}, \\ u_{i,t} &\in \mathcal{U}_i = \{u_{i,t} \in \mathbb{R}^{m_i} : L_i u_i \leq l_i\}, \end{aligned} \tag{3.7}$$

with $L_i = U_i L U_i^\top$ and $l_i = U_i l$, and $G_{\mathcal{N}_i} = X_{\mathcal{N}_i} G X_{\mathcal{N}_i}^\top$ and $g_{\mathcal{N}_i} = X_{\mathcal{N}_i} g$.

Control Problem Formulation

Let us consider system (3.1). We are given an iterative task, where the trajectories of the subsystems start at the same initial states at each iteration. We will discuss the case of non-iterative tasks in Section 12. In the following, we denote the iteration by a superscript q and the initial state at iteration q by

$$x_{i,0}^q = x_{i,S}, \quad \forall i \in \mathcal{N}, \tag{3.8}$$

where the overall initial state $x_0^q = x_S$ is defined as a stacked vector similar to (3.4).

The goal is to solve the following infinite horizon optimal control problem (IHOCP) at each iteration

$$\begin{aligned} J_{0\to\infty}^*(x_S) = \min_{u_0,u_1,\ldots} \quad & \sum_{t=0}^{\infty} h(x_t, u_t) \\ \text{s.t.} \quad & x_{t+1} = Ax_t + Bu_t, \ \forall t \geq 0 \\ & x_t \in \mathcal{X}, \ \forall t \geq 0 \\ & u_t \in \mathcal{U}, \ \forall t \geq 0 \\ & x_0 = x_S. \end{aligned} \tag{3.9}$$

In the following, we consider problems that involve decomposable stage costs in (3.9), i.e., where $h(x_t, u_t)$ is given as a sum of local stage costs $h_i(x_{\mathcal{N}_i,t}, u_{i,t})$ as

$$h(x_t, u_t) = \sum_{i=1}^{M} h_i(x_{\mathcal{N}_i,t}, u_{i,t}). \tag{3.10}$$

We assume that the local stage costs $h_i(\cdot,\cdot)$ are continuous, jointly convex and satisfy

$$\begin{cases} h_i(x_{\mathcal{N}_i,F}, 0) = 0, \\ h_i(x_{\mathcal{N}_i,t}^q, u_{i,t}^q) \geq 0, \quad \forall\ x_{\mathcal{N}_i,t}^q \in \mathbb{R}^{n_{\mathcal{N}_i}} \backslash \{x_{\mathcal{N}_i,F}\}, \ \forall\ u_{i,t}^q \in \mathbb{R}^{m_i} \backslash \{0\}, \end{cases} \tag{3.11}$$

where the final state x_F is a feasible equilibrium for system (3.1) under no input, i.e., $Ax_F = x_F$.

Remark 1. While the local stage costs $h_i(x_{\mathcal{N}_i,t}, u_{i,t})$ can account for coupling between the subsystems, this formulation includes the special case of completely separable cost functions with local stage costs given as $h_i(x_{i,t}, u_{i,t})$.

Remark 2. A specific choice of the stage cost $h(x_t, u_t)$ can be the quadratic function

$$h(x_t, u_t) = x_t^\top Q x_t + u_t^\top R u_t,$$

with positive semi-definite and positive definite weighting matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$, respectively. In this case, the local stage costs are given by

$$h_i(x_{\mathcal{N}_i,t}, u_{i,t}) = x_{\mathcal{N}_i,t}^\top Q_{\mathcal{N}_i} x_{\mathcal{N}_i,t} + u_{i,t}^\top R_i u_{i,t},$$

with $Q_{\mathcal{N}_i}$ and R_i such that the global weighting matrices Q and R are given by $Q = \sum_{i\in\mathcal{N}} X_{\mathcal{N}_i}^\top Q_{\mathcal{N}_i} X_{\mathcal{N}_i}$ and $R = \sum_{i\in\mathcal{N}} U_i^\top R_i U_i$. A completely separable quadratic stage cost is then defined as

$$h_i(x_{i,t}, u_{i,t}) = x_{i,t}^\top Q_i x_{i,t} + u_{i,t}^\top R_i u_{i,t}, \tag{3.12}$$

with Q_i such that $Q = \sum_{i\in\mathcal{N}} X_i^\top Q_i X_i$ and R_i as before.

3.3 Background on LMPC

We review the LMPC problem formulation for the global system in (3.1) from [113]. For this, we define the vectors that collect all inputs applied to system (3.1) and its resulting states for all time steps t of iteration q as

$$\begin{aligned} \mathbf{u}^q &= [u_0^{q\top}, u_1^{q\top}, ..., u_t^{q\top}, ...]^\top, \\ \mathbf{x}^q &= [x_0^{q\top}, x_1^{q\top}, ..., x_t^{q\top}, ...]^\top. \end{aligned} \tag{3.13}$$

Convex Safe Set

In order to guarantee stability of MPC laws, an N-step controllable set to a control invariant set can be used. Computing such a set is usually numerically challenging or even intractable for nonlinear systems or large scale distributed systems. To alleviate this problem, we will as [112] exploit previously seen trajectories that successfully completed the iterative task. Since they represent a subset of the maximal stabilizable set, the sampled safe set $\mathcal{SS}^q$ is defined over the realized trajectories of the system from previous iterations

$$\mathcal{SS}^q = \left\{ \bigcup_{l\in T^q} \bigcup_{t=0}^{\infty} x_t^l, \right\}, \tag{3.14}$$

where T^q collects all iteration indices from previous successful iterations, i.e., which were feasible and converged to x_F, defined as

$$T^q = \left\{ l \in [0, q] : \lim_{t\to\infty} x_t^l = x_F \right\}. \tag{3.15}$$

Because of the convexity of the constraints $\mathcal{X}$ and $\mathcal{U}$, any convex combination of the elements in the safe set $\mathcal{SS}^q$ is again a control invariant set for system (3.1), i.e., for any element in the convex safe set

$$\mathcal{CS}^q = \mathrm{conv}(\mathcal{SS}^q) = \left\{ \sum_{l \in T^q} \sum_{t=0}^{\infty} \alpha_t^l x_t^l : \ \alpha_t^l \geq 0, \ \sum_{l \in T^q} \sum_{t=0}^{\infty} \alpha_t^l = 1, \ x_t^l \in \mathcal{SS}^q \right\}, \tag{3.16}$$

there exists a sequence of control inputs that steers the system (3.1) to x_F [24]. If all previous successful trajectories are taken into account, then it holds that the sets are growing over the iterations, i.e., $T^{q-1} \subseteq T^q$ and therefore

$$\mathcal{CS}^{q-1} \subseteq \mathcal{CS}^q. \tag{3.17}$$

Terminal Cost

For the qth realized trajectory $\mathbf{x}^q$ and associated input sequence $\mathbf{u}^q$ in (3.13), the cost-to-go from time t onwards is given by

$$J_{t \to \infty}^q(x_t^q) = \sum_{k=t}^{\infty} h(x_k^q, u_k^q). \tag{3.18}$$

The performance of the qth trajectory is defined as the cost from time $t = 0$, i.e.,

$$J_{0 \to \infty}^q(x_0^q) = \sum_{t=0}^{\infty} h(x_t^q, u_t^q). \tag{3.19}$$

The barycentric function [66] is used as the terminal cost in the LMPC for linear systems in [112]. It is defined as

$$\begin{aligned} V^{q,*}(x) = \min_{\boldsymbol{\alpha}^q} \quad & \sum_{l=0}^{q} \sum_{t=0}^{\infty} \alpha_t^l J_{t \to \infty}^l(x_t^l) \\ \text{s.t.} \quad & \sum_{l=0}^{q} \sum_{t=0}^{\infty} \alpha_t^l = 1 \\ & \sum_{l=0}^{q} \sum_{t=0}^{\infty} \alpha_t^l x_t^l = x, \\ & \alpha_t^l \geq 0, \quad \forall t \in \mathbb{N}, \end{aligned} \tag{3.20}$$

with x_t^l being the realized state at time t of the lth iteration, and where $\boldsymbol{\alpha}^q$ comprises all $\alpha_t^l, \forall l \in \{0, ..., q\}, \forall t \in \mathbb{N}$. The function $V^{q,*}$ thus assigns to every point in the convex safe set the corresponding convex combination of minimum costs-to-go along the previous trajectories in the safe set.

Remark 3. In practical applications, the iterations will have a finite time duration. For simplicity, we adopt the infinite time formulation in this paper.

An LMPC [112] for a centralized linear system then solves at each time step t the following finite horizon optimal control problem (FHOCP),

$$\begin{aligned} J_{t\to t+N}^{\mathrm{LMPC},q}(x_t^q) = \min_{\mathbf{x}_{t,N},\mathbf{u}_{t,N-1},\boldsymbol{\alpha}^{q-1}} \quad & \left[\sum_{k=t}^{t+N-1} h(x_{k|t},u_{k|t}) + V^{q-1,*}(x_{k+N|t})\right] \\ \text{s.t.} \quad & x_{k+1|t} = Ax_{k|t} + Bu_{k|t}, \\ & x_{k|t} \in \mathcal{X}, \\ & u_{k|t} \in \mathcal{U},\ k = t,...,t+N-1 \\ & x_{t|t} = x_t^q, \\ & x_{t+N|t} \in \mathcal{CS}^{q-1}, \end{aligned} \tag{3.21}$$

with

$$\mathbf{x}_{t,N} = [x_{t|t}^\top,...,x_{t+N|t}^\top]^\top,\ \mathbf{u}_{t,N-1} = [u_{t|t}^\top,...,u_{t+N-1|t}^\top]^\top. \tag{3.22}$$

Let us denote the optimal solution to (3.21) by

$$\mathbf{x}_{t,N}^* = [x_{t|t}^{*\top},...,x_{t+N|t}^{*\top}]^\top,\ \mathbf{u}_{t,N-1}^* = [u_{t|t}^{*\top},...,u_{t+N-1|t}^{*\top}]^\top. \tag{3.23}$$

At time t, the first input is applied to the system, i.e., $u_t^q = u_{t|t}^{*q}$, and the problem (3.21) is solved again for the next time step in a receding horizon fashion.

Under the assumption that at iteration $q = 1$ the convex safe set is non-empty, i.e., $\mathcal{CS}^{q-1} = \mathcal{CS}^0 \neq \emptyset$, recursive and iterative feasibility, asymptotic stability and non-decreasing performance over the iterations are proved in [113].

3.4 DLMPC

In the following, we present the problem formulation of DLMPC, which extends the LMPC approach to distributed systems. Let us consider the coupled constrained linear distributed system from (3.1). We define the vectors that collect all inputs applied to subsystem i in (3.5) and its resulting states for all time steps t of iteration q as

$$\mathbf{u}_i^q = [u_{i,0}^{q\top}, u_{i,1}^{q\top},...,u_{i,t}^{q\top},...]^\top,\ \mathbf{x}_i^q = [x_{i,0}^{q\top}, x_{i,1}^{q\top},...,x_{i,t}^{q\top},...]^\top. \tag{3.24}$$

We further define the local sampled safe sets for subsystems $i \in \mathcal{N}$ over the realized trajectories of the subsystem from all successful previous iterations up to q as

$$\mathcal{SS}_i^q = \left\{\bigcup_{l\in T^q}\bigcup_{t=0}^{\infty} x_{i,t}^l,\right\}, \tag{3.25}$$

with T^q as defined before for (3.14). Moreover, we note that we can decompose the safe set from (3.16) into the following local convex safe sets

$$\mathcal{CS}_i^q = \left\{ \sum_{l \in T^q} \sum_{t=0}^{\infty} \alpha_{i,t}^l x_{i,t}^l : \alpha_{i,t}^l \geq 0,\ \sum_{l \in T^q} \sum_{t=0}^{\infty} \alpha_{i,t}^l = 1,\ x_{i,t}^l \in \mathcal{SS}_i^q \right\}, \tag{3.26}$$

where the coefficients $\alpha_{i,t}^l, \forall l \in \{0, ..., q-1\}, \forall t \geq 0$ will be optimized over in problems (3.28) and (3.30).

We note the following relation of the convex safe set $\mathcal{CS}^q$ in (3.16) and the local convex safe sets $\mathcal{CS}_i^q$ in (3.26), which will be important for the decomposition of the problem in (3.21):

$$x = [x_1^\top, ..., x_M^\top]^\top \in \mathcal{CS}^q \iff x_i \in \mathcal{CS}_i^q, \quad \boldsymbol{\alpha}_i^q = \boldsymbol{\alpha}_j^q, \forall i \neq j, i, j \in \mathcal{N}, \tag{3.27}$$

with $\boldsymbol{\alpha}_i^q$ comprising $\alpha_{i,t}^l, \forall l \in \{0, ..., q-1\}, \forall t \geq 0$ in (3.26).

Based on the assumption before that the global system is decomposable into M coupled subsystems, the global LMPC problem in (3.21) can equivalently be decomposed into the following subproblems

$$\begin{aligned} J_{i,t \to t+N}^{\text{LMPC},q}(x_{i,t}^q) = \min_{\substack{\mathbf{x}_{\mathcal{N}_i,t,N}, \\ \mathbf{u}_{i,t,N-1}, \\ \boldsymbol{\alpha}_i^{q-1}}} & \left[\sum_{k=t}^{t+N-1} h_i(x_{\mathcal{N}_i,k|t}, u_{i,k|t}) + V_i^{q-1,*}(x_{i,t+N|t}) \right] \\ \text{s.t.} \quad & x_{i,k+1|t} = A_{\mathcal{N}_i} x_{\mathcal{N}_i,k|t} + B_i u_{i,k|t}, \\ & x_{\mathcal{N}_i,k|t} \in \mathcal{X}_{\mathcal{N}_i}, \\ & u_{i,k|t} \in \mathcal{U}_i,\ k = t, ..., t+N-1 \\ & x_{\mathcal{N}_i,t|t} = x_{\mathcal{N}_i,t}, \\ & x_{i,t+N|t} \in \mathcal{CS}_i^q, \end{aligned} \tag{3.28}$$

with

$$\mathbf{x}_{\mathcal{N}_i,t,N} = [x_{\mathcal{N}_i,t|t}^\top, ..., x_{\mathcal{N}_i,t+N-1|t}^\top]^\top,\ \mathbf{u}_{i,t,N-1} = [u_{i,t|t}^\top, ..., u_{i,t+N-1|t}^\top]^\top, \tag{3.29}$$

with $\boldsymbol{\alpha}_i^{q-1}$ comprising $\alpha_{i,t}^l, \forall l \in \{0, ..., q-1\}, \forall t \in \mathbb{N}$, and with $V_i^{q-1,*}(x_{i,k+N|t})$ being defined as in (3.20), but with $J_{t \to \infty}^l(x_t^l)$ replaced by

$$J_{i,t \to \infty}^l(x_{i,t}^l) = \sum_{k=t}^{\infty} h_i(x_{\mathcal{N}_i,k}^l, u_{i,k}^l).$$

In order to guarantee that the decomposed problem in (3.28) is an exact reformulation of the global problem in (3.21), i.e., to guarantee that they have the same solutions, the following consensus constraints need to be introduced

$$\begin{aligned} \boldsymbol{\alpha}_i^{q-1} &= \boldsymbol{\alpha}_j^{q-1}, & \forall i, j \in \mathcal{N}, i \neq j, \\ \mathbf{x}_{\mathcal{N}_i,t,N} &= X_{\mathcal{N}_i} \mathbf{x}_{t,N}, & \forall i \in \mathcal{N},\ t \geq 0, \end{aligned} \tag{3.30}$$

with $\mathbf{x}_{t,N}$ the planned state trajectory of the global system as defined in (3.22). The consensus constraint in the first line of (3.30) ensures the condition in (3.27), and the one in the second line ensures that overlapping parts of state variables from neighboring subsystems in $\mathbf{x}_{\mathcal{N}_i,t,N}$, i.e., variables of different subsystems that have the same physical meaning, are the same. The local FHOCPs in (3.28) are solved in a receding horizon fashion, i.e., the first local inputs $u^q_{i,t} = u^{*q}_{i,t|t}$ are applied to the subsystems at time t. The next section presents a distributed solution method to solve the subproblems (3.28).

3.5 Distributed Synthesis for DLMPC

In this section, we present a distributed solution method for the local decomposed subproblems in (3.28) coupled over the consensus constraints in (3.30). Various distributed optimization algorithms can be employed [16]. We propose a distributed solution scheme based on the alternating direction method of multipliers (ADMM) because of its fast convergence in practice [46, 28]. A consensus algorithm involving a central coordinator [28] could be implemented, which requires communication to every subsystem and therefore might not be tractable in practice. We propose a scheme, where only nearest-neighbor communication and no global coordination is required. A similar scheme has been presented before for distributed controller synthesis of large-scale systems in [132].

Distributed Synthesis for DLMPC

Let us define the local variable vector of subsystem i as

$$s_i = [\mathbf{x}^\top_{\mathcal{N}_i,t,N},\ \boldsymbol{\alpha}_i^{q-1\top},\ \mathbf{u}^\top_{i,t,N-1}]^\top, \tag{3.31}$$

and the projection matrices E_{ij}, which project s_i onto those variables over which a consensus needs to be achieved between subsystem i and its neighboring subsystems $j \in \mathcal{N}_i$, i.e., $E_{ij}s_i = E_{ji}s_j$ is the consensus constraint from (3.30) for subsystem i with j. The decomposed problem in (3.28) and (3.30) can now be formulated as the following M subproblems for all $i \in \mathcal{N}$

$$\begin{aligned} \min_{s_i}\ & J^q_{i,t\to t+N}(s_i) + g_i(s_i) \\ \text{s.t. } & E_{ij}s_i = E_{ji}s_j, \quad \forall j \in \mathcal{N}_i. \end{aligned} \tag{3.32}$$

with $J^q_{i,t\to t+N}(s_i)$ being the cost function in (3.28), and $g_i(s_i)$ being the indicator function for the constraints in (3.28), i.e.,

$$g_i(s_i) = \begin{cases} 0 & \text{if } s_i \text{ satisfies the constraints in (3.28)}, \\ +\infty & \text{otherwise.} \end{cases}$$

Algorithm 3: Distributed computation of local input $u^q_{i,t} = u^{*q}_{i,t|t}$ for subsystem i at time t of iteration q

```
   Input: Iteration $q$, time $t$, $\rho > 0$, set of neighboring subsystems $\mathcal{N}_i$, current
          subsystems states $x^q_{i,t}$, initial values $s_i^{(0)}$, $\forall i \in \mathcal{N}$
1  for $i \in \mathcal{N}$ do
2      Initialization: Set $\kappa \leftarrow 0$, $\lambda_i^{(0)} \leftarrow 0$;
3      while not converged do
4          Communicate $E_{ij}\, s_i^{(\kappa)}$ to neighboring nodes $j \in \mathcal{N}_i$;
5          $\lambda_i^{(\kappa+1)} \leftarrow \lambda_i^{(\kappa)} + \rho \sum_{j\in\mathcal{N}_i}(T_{ij}\, s_i^{(\kappa)} - T_{ji}\, s_j^{(\kappa)})$;
6          $s_i^{(\kappa+1)} \leftarrow$
             $\underset{s_i}{\operatorname{argmin}}\left\{ J^q_{i,t\to t+N}(s_i) + g_i(s_i) + s_i^\top \lambda_i^{(\kappa+1)} + \rho \sum_{j\in\mathcal{N}_i} \|T_{ij}\, s_i - \frac{T_{ij}\, s_i^{(\kappa)} + T_{ji}\, s_j^{(\kappa)}}{2}\|^2 \right\}$;
7      end
8      $\kappa \leftarrow \kappa + 1$;
9      Set $s_i^* = s_i^\kappa$;
10     Return $u^{*q}_{i,t|t}$ from $s_i^*$;
11 end
   Output: local inputs $u^q_{i,t}$,  $\forall i \in \mathcal{N}$
```

In order to derive the ADMM steps, we formulate the augmented Lagrangian, which allows a decomposition into the following sum of local terms

$$\mathcal{L}_\rho = \sum_{i\in\mathcal{N}} \mathcal{L}_{\rho,i}, \tag{3.33}$$

with

$$\mathcal{L}_{\rho,i} = J^q_{i,t\to t+N}(s_i) + g_i(s_i) + \sum_{j\in\mathcal{N}_i} \left(\lambda_{ij}^\top (E_{ij}s_i - E_{ji}s_j) + \frac{\rho}{2}\|E_{ij}s_i - E_{ji}s_j\|_2^2 \right). \tag{3.34}$$

The modified ADMM update steps are summarized in Algorithm 3. The derivation can be found in [132]. The update steps require communication only between neighboring subsystems, i.e., subsystems that are coupled through their dynamics, constraints, or costs.

The DLMPC for iterative tasks, with distributed solution of the subproblems by Algorithm 3, is given in Algorithm 4.

Properties of the DLMPC

Next, we present our main result on the properties of Algorithm 4. We make the following assumptions.

Algorithm 4: DLMPC

```
Input: Initial states $x_{i,S}$, target states $x_{i,F}$, sets of neighboring subsystems $\mathcal{N}_i$,
       initial successful feasible trajectories $\mathbf{x}^0_{i,t}$, $\mathbf{u}^0_{i,t}$ with $\mathcal{CS}^0_i$ and
       $J^0_{i,t\to\infty}(x_{i,t}), \forall x_{i,t} \in \mathbf{x}^0_{i,t}$, $\forall i \in \mathcal{N}$, $q_{\max}$
1  for iteration $q = 1$ to $q_{\max}$ do
2      for $i \in \mathcal{N}$ do
3          while $x_{i,t} \neq x_{i,F}$ do
4              Solve local problem (3.32) via Algorithm 3;
5              Apply local input $u^q_{i,t} = u^{*q}_{i,t|t}$;
6              Obtain local state $x^q_{i,t}$;
7          end
8          Update $\mathcal{CS}^q_i$ by adding $\mathbf{x}^*_{i,t}$;
9          Compute and save $J^q_{i,t\to\infty}(x^q_{i,t}), \forall x^q_{i,t} \in \mathbf{x}^*_{i,t}$;
10     end
11     $q = q + 1$;
12 end
Output: Closed-loop trajectories $\mathbf{x}^*_{i,t}$, $\mathbf{u}^*_{i,t}$, $\forall i \in \mathcal{N}$
```

Assumption 1. We have access to feasible trajectories $\mathbf{x}^q_i$ at iteration $q = 0$ converging to $x_{i,F}$ for all subsystems $i \in \mathcal{N}$, and therefore the convex safe sets at iteration $q = 1$, $\mathcal{CS}^{q-1}_i = \mathcal{CS}^0_i$, are non-empty.

Assumption 2. We assume that the local cost functions $J^q_{i,t\to t+N}(\cdot) + g_i(\cdot)$ in (3.32) are closed, proper and convex for all subsystems $i \in \mathcal{N}$, and that the unaugmented Lagrangian

$$\mathcal{L}_i = J^q_{i,t\to t+N}(s_i) + g_i(s_i) + \sum_{j\in\mathcal{N}_i} \lambda^\top_{ij} \left(E_{ij}s_i - E_{ji}s_j\right)$$

has a saddle point, and that the ADMM update steps in Algortihm 3 are feasible.

In addition to the classical MPC properties, namely, persistent feasibility in each iteration, and asymptotic stability of the equilibria $x_{i,F}$, the following properties hold for the DLMPC in Algorithm 4.

Theorem 1. Consider system (3.1), with distributed structure (3.5) and (3.7). Let Assumptions 1 and 2 hold. Then, the DLMPC in Algorithm 4 has the following properties:

1. The DLMPC is feasible for all $t \geq 0$ and at every iteration $q \geq 1$. The equilibrium points $x_{i,F}$ are asymptotically stable for the closed-loop coupled subsystems under the DLMPC law.

2. The iteration cost $J^q_{0\to\infty}(x_S)$ of the closed-loop system does not increase with the iteration index q, i.e., $J^{q+1}_{0\to\infty} \leq J^q_{0\to\infty}$.

3. If the closed-loop system under the DLMPC converges to the steady-state inputs $\mathbf{u}_i^\infty = \lim_{q\to\infty} \mathbf{u}_i^q$ and the related steady-state trajectories $\mathbf{x}_i^\infty = \lim_{q\to\infty} \mathbf{x}_i^q$, for all subsystems $i \in \mathcal{N}$, and the conditions from [112, Theorem 3] are satisfied, then, $\mathbf{u}_i^\infty$ and $\mathbf{x}_i^\infty$ are global optimal solutions for the IHOCP (3.9).

Proof. The properties of Theorem 1 have been proven in [112] and [113] for a single system. It therefore suffices to show that the proposed decomposed problem solved in Algorithm 4 is an exact reformulation of the global centralized problem and that the distributed solution method in Algorithm 3 converges to the global optimal solution.

It can easily be seen that the local subproblems in (3.28) together with the consensus constraints in (3.30), and their reformulation into the subproblems in (3.32) are exact reformulations of the global problem in (3.21). This follows from the decomposability of the cost function in (3.10) and the structure of the system in (3.5) and (3.7), together with the definitions of $\mathcal{CS}_i^q$ and $V_i^{q,*}$ in (3.26) and (3.28) with the consensus constraints in (3.30). For linear system dynamics and convex constraints, the problems (both the global and the local ones) are convex and therefore admit a global optimal solution.

The proposed distributed solution method in Algorithm 3 is equivalent to the update steps of consensus ADMM in [28]. This equivalence has been shown in the derivation of the steps of Algorithm A.1 in [13]. Under Assumption 2, the residuals $E_{ij}s_i - E_{ji}s_j,\ \forall j \in \mathcal{N}_i,\ \forall i \in \mathcal{N}$ in Algorithm 3 asymptotically converge to zero and the cost $\sum_{i\in\mathcal{N}}(J_{i,t\to t+N}^q(\cdot) + g_i(\cdot))$ from (3.32) asymptotically converges to the global optimal solution. This is true in each time step and therefore $J_{0\to\infty}^q(x_S) = \sum_{i\in\mathcal{N}} J_{i,0\to\infty}^q(x_{i,S})$ converges to the global optimal solution. With the previous results, this is equivalent to the global optimal solution of the global centralized problem (3.21).

Therefore, the proofs in [112] and [113] for a single system can be applied to the global centralized problem and thus the properties in Theorem 1 hold. ■

Note that the properties in Theorem 1 hold for the global system in (3.1), i.e., for the ensemble of all coupled subsystems. In particular, property 2) guarantees a decrease in the iteration cost $J_{0\to\infty}^q(x_S) = \sum_{i\in\mathcal{N}} J_{i,0\to\infty}^q(x_{i,S})$ of the sum of costs of all subsystems over iterations, rather than a decrease in the iteration costs $J_{i,0\to\infty}^q(x_{i,S})$ of the individual subsystems. Similarly, the optimal cost function $J_{t\to t+N}^{\mathrm{LMPC}}(\cdot) = \sum_{i\in\mathcal{N}} J_{i,t\to t+N}^{\mathrm{LMPC}}(\cdot)$, is a Lyapunov function for the equilibrium point x_F of the closed loop system (3.1) rather than the individual cost functions $J_{i,t\to t+N}^{\mathrm{LMPC}}(\cdot)$ for the individual subsystems. Furthermore, Algorithm 3 enables a distributed implementation of the global terminal constraint set $\mathcal{CS}^{q-1}$ on which no distributed structure is imposed. The approach presented in this paper therefore captures the couplings between the subsystems and thus reduces conservatism w.r.t. other approaches of distributed MPC in the literature which impose structure on the terminal cost or terminal constraint sets.

The size of the decomposed local FHOCPs in (3.28) are of the size of the individual subsystems and are independent of the number of subsystems. Since only nearest-neighbor communication is required in Algorithm 3, also the solution method scales well with the number of subsystems. The number of data points for the construction of the convex safe sets in (3.26) grows in each iteration with adding the most recent closed loop trajectories to the safe sets in (3.25). In order to reduce the required computational effort, the set of data points can be truncated, i.e., not all previously seen data points need to be included in the safe sets in (3.25). For example only the most recent trajectories, or only the previous trajectory can be chosen to be included.

Safe and Efficient Data Generation and Domain Enlargement of the DLMPC Policy

In order to use Algorithm 4 for a (possibly iterative) task, data from at least one set of successful feasible trajectories of the subsystems are required to construct the local terminal sets and cost functions, which guarantee the properties in Theorem 1. While successful feasible trajectories might be easy to obtain for distributed systems in some applications, such as by locally or manually controlling multiple loosely coupled subsystems in a non-optimal way, in other applications, such as for tightly coupled subsystems with safety-critical constraints, these data might be difficult to generate. We therefore propose in the following a distributed algorithm which allows the safe and efficient generation of the data required for the computation of the terminal sets and costs in Algorithm 4. We present this data generation method for a control task from given initial states $x_{i,0}^{\text{des}}$ to the target states $x_{i,F}$ of the subsystems. Let us define the following FHOCP, which is similar to the one in (3.28) except for a different cost function, and with the initial states $x_{i,0}$ being optimization variables

$$\begin{aligned} \min_{\substack{\mathbf{x}_{\mathcal{N}_i,t,N},\\ \mathbf{u}_{i,t,N-1},\\ \boldsymbol{\alpha}_i^{q-1}}} \quad & \|x_{i,0}^q - x_{i,0}^{\text{des}}\|_2^2 \\ \text{s.t.} \quad & x_{i,k+1|t} = A_{\mathcal{N}_i} x_{\mathcal{N}_i,k|t} + B_i u_{i,k|t}, \;\; k = 0, ..., N-1 \\ & x_{\mathcal{N}_i,k|t} \in \mathcal{X}_{\mathcal{N}_i}, \;\; k = t, ..., t+N-1 \\ & u_{k|t}^i \in \mathcal{U}_i, \;\; k = t, ..., t+N-1, \\ & x_{i,t+N|t} \in \mathcal{CS}_i^{q-1}, \end{aligned} \tag{3.35}$$

with $\mathbf{x}_{\mathcal{N}_i,t,N}$, $\mathbf{u}_{i,t,N-1}$, $\boldsymbol{\alpha}_i^{q-1}$ as in (3.29), and where the consensus constraints in (3.30) have to hold. Note that no initial successful feasible trajectories $\mathbf{x}_{i,t}^0$ need to be available. Instead, we use only the target states $x_{i,F}$ as initial feasible trajectories and therefore define $\mathcal{CS}_i^0 = x_{i,F}$.

Iteratively solving (3.35) and computing the DLMPC closed-loop trajectories by Algorithm 4 enlarges the domain of the DLMPC policy and converges to feasible trajectories starting at $x_{i,0} = x_{i,0}^{\text{des}}$ and ending in $x_{i,0} = x_{i,F}$, which can used as the input to Algorithm 4. These steps are summarized in the following Algorithm 5

Remark 4. Algorithm 5 can also be used to compute a larger domain of the DLMPC policy. If no specific initial states $x_{i,0}^{\text{des}}$ are given, instead of the cost function $\|x_{i,0}^q - x_{i,0}^{\text{des}}\|_2^2$ in (3.35), a different cost function can be used, for example to compute the initial states $x_{i,0}$ for all

Algorithm 5: Efficient and safe distributed data generation for the DLMPC policy

```
   Input: Terminal states $x_{i,F}$, sets of neighboring systems $\mathcal{N}_i$, desired initial states
          $x_{i,0}^{\text{des}}$, $\forall i \in \mathcal{N}$, $r_{\max}$
1  Initialize: Set $\mathcal{CS}_i^0 = x_{i,F}$;
2  Set iteration count $r = 1$;
3  for $i \in \mathcal{N}$ do
4      while $\|x_{i,0}^r - x_{i,0}^{\text{des}}\|_2^2 \leq \epsilon$ and $r \leq r_{\max}$ do
5          Solve (3.35) to obtain $x_{i,0}^r$;
6          Compute $\mathbf{x}_{i,t}^*$, $\mathbf{u}_{i,t}^*$ via Algorithm 4 with inputs: $x_{i,S} = x_{i,0}^r$, $\mathcal{CS}_i^0 = \mathcal{CS}_i^{r-1}$,
            $J_{i,t\to\infty}^0 = J_{i,t\to\infty}^r$, $q_{\max} = 1$, until line 7 of Algorithm 4, then break;
7          Update $\mathcal{CS}_i^r$ by $\mathbf{x}_{i,t}^*$;
8          Compute and save $J_{i,t\to\infty}^r(x_{i,t}), \forall x_{i,t} \in \mathbf{x}_{i,t}^*$;
9          $r = r + 1$;
10     end
11     Set $\mathcal{CS}_i = \mathcal{CS}_i^r$;
12 end
   Output: $\mathcal{CS}_i$ containing successful feasible trajectories from $x_{i,0}^{\text{des}}$ to $x_{i,F}$,   $\forall i \in \mathcal{N}$
```

subsystems $i \in \mathcal{N}$ as the points furthest in the direction of interest at the borders of the convex safe sets $\mathcal{CS}_i^q$.

3.6 Numerical Experiments

In this section, we present numerical examples to demonstrate the methods of the DLMPC scheme in Algorithm 4 and the data generation in Algorithm 5.

We consider a system of three dynamically coupled subsystems with coupled state constraints. The subsystems have two states each, i.e., $x_i = [x_{i1},\ x_{i2}]^\top$, $\forall i \in \mathcal{N}$. The overall system state is given by $x = [x_1^\top,\ x_2^\top,\ x_3^\top]^\top \in \mathbb{R}^6$, and the input vector by $u = [u_1,\ u_2,\ u_3]^\top \in \mathbb{R}^3$. The system matrices of the global system are given by

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 \\ 0 & A_{22} & A_{23} \\ A_{31} & 0 & A_{33} \end{bmatrix}, \quad B = \operatorname{diag}(B_{11}, B_{22}, B_{33}), \tag{3.36}$$

with

$$\begin{gathered} A_{11} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1.1 \end{bmatrix}, \quad A_{22} = \begin{bmatrix} 1.05 & 0.6 \\ 0 & 1 \end{bmatrix}, \quad A_{33} = \begin{bmatrix} 1 & 0.55 \\ 0 & 1.05 \end{bmatrix}, \\ A_{12} = A_{23} = A_{31} = -\begin{bmatrix} 0.1 & 0.2 \\ 0 & 0.3 \end{bmatrix}, \ B_{11} = B_{22} = B_{33} = [0, 1]^\top, \end{gathered} \tag{3.37}$$

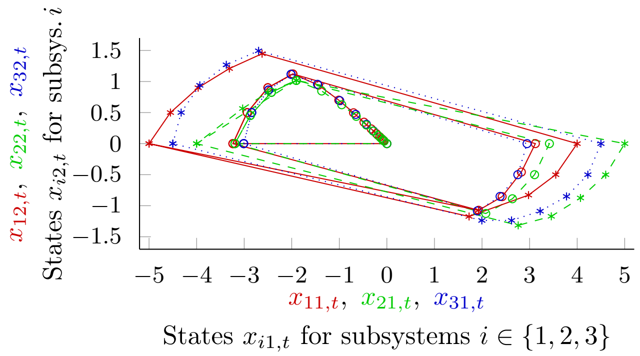


Figure 3.1: Domain enlargement for desired initial states $x_{i,0}^{\text{des},1}$, $x_{i,0}^{\text{des},2}$: $\mathcal{CS}_i^q$ of subystems $i = 1$ (solid red ——), $i = 2$ (dashed green - - -), $i = 3$ (dotted blue ......) over iterations $q = 1, 2$ ($\circ$), $q = 3, 4$ ($*$), starting with $\mathcal{CS}_i^0 = [0,\ 0]^\top$.

$$\begin{aligned}
-0.9 &\le x_{11} - x_{21} \le 0.9, \\
-0.9 &\le x_{21} - x_{31} \le 0.9, \\
-3 &\le u_i \le 3 \\
-5 &\le x_{ik} \le 5, \quad \forall i \in \mathcal{N}, \quad k \in \{1, 2\}.
\end{aligned} \tag{3.38}$$

Data Generation

First, we generate the feasible trajectories required as inputs to Algorithm 4, by making use of Algorithm 5. We choose the following desired initial states

$$\begin{aligned}
x_{1,0}^{\text{des},1} &= [-5, 0]^\top, & x_{1,0}^{\text{des},2} &= [4, 0]^\top, \\
x_{2,0}^{\text{des},1} &= [-4.5, 0]^\top, & x_{2,0}^{\text{des},2} &= [4.5, 0]^\top, \\
x_{3,0}^{\text{des},1} &= [-4, 0]^\top, & x_{3,0}^{\text{des},2} &= [5, 0]^\top.
\end{aligned} \tag{3.39}$$

We iteratively compute the inital states $x_{i,0}$ as those states closest to $x_{i,0}^{\text{des},1}$ and $x_{i,0}^{\text{des},2}$, in an alternating way, thus enlarging the domain of the DLMPC policy in both the negative and positive x_{i1} directions within the feasible region of the state space.

Remark 5. Note that if no specific initial states $x_{i,0}^{\text{des},1}$ and $x_{i,0}^{\text{des},2}$ are defined, a similar result of domain enlargement is achieved by changing the cost function in (3.35) to $x_{i,0}^q$ and $-x_{i,0}^q$, respectively.

Figure 3.1 shows the enlargement of the convex hulls of the safe sets, $\mathcal{CS}_i^q$, of the three subsystems over iterations $q = 0$ to 4 of Algorithm 5. At iteration 4, the given initial states $x_{i,0}^{\text{des},1}$ and $x_{i,0}^{\text{des},2}$ have been reached, i.e., closed-loop trajectories from $x_{i,0}^{\text{des},1}$ to $x_{i,F}$ and from $x_{i,0}^{\text{des},2}$ to $x_{i,F}$ under the DLMPC law have been generated.

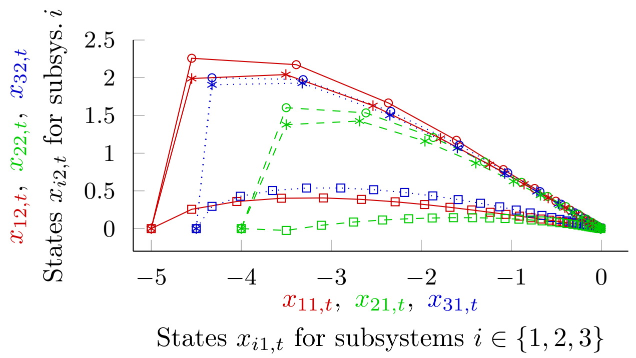


Figure 3.2: Iterative regulation task for 3 dynamically coupled subsystems. Trajectores of subystems $i = 1$ (solid red ——), $i = 2$ (dashed green - - -), $i = 3$ (dotted blue ......) shown for iterations $q = 0$ (□), $q = 1$ (○), $q = 10$ ($*$).

Table 3.1: Iteration costs *converged to global optimal solution (computed for the centralized system with $N = 200$).

Iteration	0	1	2	3	4	5	6	7	8	9	10
Sys.	295.63	216.96	216.41	216.31	216.28	216.26	216.26	216.25	216.25	216.25	216.25*
Subsys. 1	112.47	87.97	88.07	88.22	88.31	88.36	88.38	88.40	88.41	88.42	88.42
Subsys. 2	113.05	76.52	76.10	75.90	75.78	75.72	75.68	75.66	75.64	75.63	75.63
Subsys. 3	70.11	52.47	52.23	52.19	52.19	52.19	52.19	52.19	52.20	52.20	52.20

Iterative Control Task

We consider now the control task to steer the coupled subsystems in (3.36) from $x_{i,0}^{\text{des},1}$ as in (3.39) to $x_{i,F} = [0,\, 0]^\top$. The cost function is given as in (3.12) with $Q_i = I_{n_i}$ and $R_i = I_{m_i}$. We generate the first feasible trajectory by a distributed MPC with time horizon $N = 15$, without terminal sets and constraints, and with $Q_i = 0.1I_{n_i}$ and $R_i = I_{m_i}$. In real, possibly safety-critical, applications a feasible trajectory could be obtained by manual control of the subsystems, or with the data generation in Algorithm 5, as illustrated in Section 3.6. Table 3.1 shows the performance improvement over iterations of Algorithm 4 with a time horizon of $N = 4$. Figure 3.2 shows the resulting closed-loop trajectories. It is interesting to note that while the iteration costs of subsystems 2 and 3 are decreasing over the iterations, the one of subsystem 1 is increasing. As noted before, Theorem 1 guarantees that the sum, i.e., the iteration cost of the overall system is guaranteed to be non-increasing.

3.7 Chapter Summary

In this chapter, we presented a distributed learning model predictive control scheme for cooperative multi-agent settings, which exploits data in order to construct the terminal cost and constraints of the DMPC problem without imposing the distributed structure of the system. The required computation is done online in a distributed way. It was shown how the scheme can be used to safely explore the state-space and generate the required data or exploit data from iterative control tasks. In addition to recursive feasibility and asymptotic stability, performance improvement over iterations and convergence to the global centralized optimal solution under mild conditions are guaranteed.

Chapter 4

Hierarchical Predictive Control for Multi-Agent Systems

4.1 Introduction

Whereas the previous chapters investigated cooperative multi-agent scenarios, we now turn our attention to non-cooperative scenarios where the agents are coupled through constraints but where their objective functions do not explicitly depend on the other agents in the environment. This is especially relevant in mixed autonomy settings where both autonomous vehicles (AVs) and human driven vehicles are present and must co-exist safely in a shared workspace.

With the surge of machine learning and reinforcement learning (RL) over the last decade, there has been a strong interest in applying data-driven approaches in control. Among them, end-to-end planning [120, 151] constructs a direct map from perception to control but lacks interpretability and safety guarantees. Deep RL-based collision avoidance [85, 134] usually discretizes the action space or relaxes the vehicle dynamics constraints, which are highly relevant in tightly-constrained environments.

In contrast, model-based approaches can provide safety guarantees, however they tend to become highly computational demanding and sensitive to model accuracy. Conventional search or sampling-based path planning methods, such as rapidly exploring random trees (RRT), lattice planners, A* and their variants [103, 68, 40], quickly become intractable in dynamic environments. Most of these approaches are based on heuristics and do not guarantee collision avoidance or feasibility. Hierarchical approaches for autonomous driving have therefore been proposed in the literature, relying on path planning at the top level, maneuver choices at the strategic level, and trajectory planning and control at the lowest level [120, 103, 68, 56, 142].

The problem of collision avoidance is nonconvex and NP-hard in general [32]. Several, potentially conservative, approximations of the collision avoidance constraints have been proposed in the literature to tackle the computational intractability. In tightly-constrained dynamic environments, the main challenges arise from three aspects: 1) nonlinear and non-holonomic vehicle dynamics, 2) non-convexity of environments, and 3) change in obstacle configuration over time (motion of other human-driven or automated vehicles).

Recently, optimization-based algorithms such as Model Predictive Control (MPC) [25] have received a significant amount of attention due to their ability to include the exact dynamics and safety constraints in the formulation of the control problem. In [156], a novel optimization-based collision avoidance (OBCA) approach obtains a smooth reformulation of the collision avoidance constraints with exact vehicle geometry. MPC also displays great flexibility in leveraging data hierarchically, for example, for learning "strategies" at the strategic higher level, which is demonstrated in [138].

In this work, we focus on the problem of maneuvering a vehicle in a tight space, which is shared with other vehicles performing complex maneuvers, such as reverse parking. We assume that the vehicles do not use any form of motion coordination such as in [72]. This scenario is highly relevant, e.g., AV parking in a mixed autonomy setting. The ability to avoid collisions and perform the task in minimum time is critical, and greatly depends on the identification of a good high-level strategy to avoid collisions while maneuvering around other vehicles. In fact, a simple "stop and wait" strategy can be highly inefficient and might never complete the task in the crowded space. For the sake of simplicity, our presentation focuses on two cars in a parking lot. The approach is however general and can be applied to other AVs in other tight environments. We propose a hierarchical data-driven and strategy-guided control scheme to tackle the complexity of the problem. Our contributions are twofold:

- We propose a data-driven framework to construct a mapping from a high-dimensional environment encoding to a given set of high-level strategies. The latter is chosen such that it encodes the prior knowledge of behavior in the corresponding scenario. Specifically, a neural network is trained offline with optimal collision-free rollouts, which are collected in a simulated environment. After training, the mapping can be utilized as a strategy predictor while executing the control task online.

- A strategy-guided time-varying MPC policy is formulated with exact vehicle and obstacle geometry to perform a navigation task in tightly-constrained dynamic environments. We refer to this as Strategy-Guided Optimization-Based Collision Avoidance (SG-OBCA). In addition to SG-OBCA, we also design a set of control policies which can be selected based on the predicted strategy to maintain safety. The effectiveness of this control scheme design is demonstrated through extensive numerical simulations and hardware experiments.

The results presented in this chapter have also appeared in:

- X. Shen, E. Zhu, Y. Stürz, U. Rosolia, and F. Borrelli, "Collision avoidance in tightly-constrained environments without coordination: a hierarchical control approach". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2674-2680.

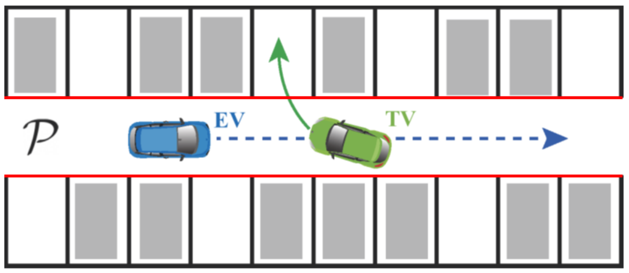


Figure 4.1: The EV tracks a reference (dashed blue) in the local region $\mathcal{P}$ while the TV executes a parking maneuver (solid green). The lane boundaries are marked in red.

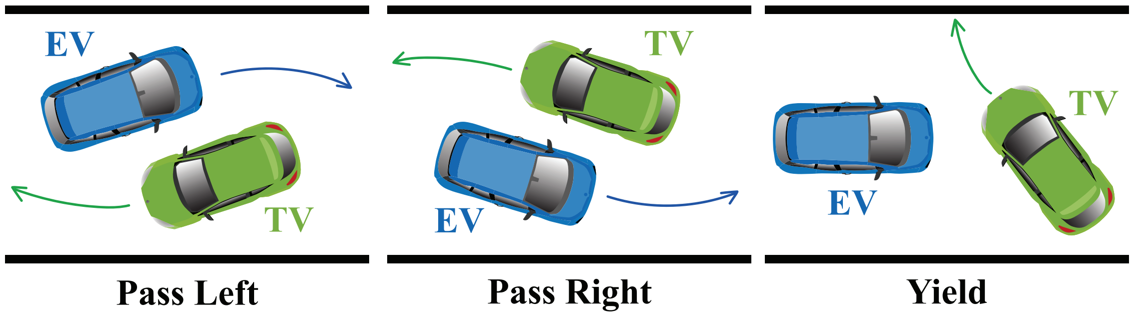


Figure 4.2: Available strategies in $\mathcal{S}$ for the EV navigation task

Notation: $\mathcal{B}_n(c,r) = \{x \in \mathbb{R}^n : \|x-c\|_2 \leq r\}$ denotes the n-D l_2-norm ball centered at $c \in \mathbb{R}^n$ with radius $r > 0$. The Minkowski sum of two sets $\mathcal{A}$ and $\mathcal{B}$ is denoted as $\mathcal{A} \oplus \mathcal{B} = \{a+b : a \in \mathcal{A}, b \in \mathcal{B}\}$. The minimum translation distance between two sets $\mathcal{A}$ and $\mathcal{B}$ is defined as $\text{dist}(\mathcal{A},\mathcal{B}) = \min_t\{\|t\| : (\mathcal{A}+t) \cap \mathcal{B} \neq \emptyset\}$ [44]. $\text{vec}(\cdot)$ denotes the vectorization and $\text{vec}(\cdot,\cdot)$ the concatenating operation to flatten a sequence of matrices into a single vector.

4.2 Problem Formulation: Strategy-Guided Collision Avoidance

We consider the problem of two-vehicle collision avoidance in a parking lot between a human-driven and autonomous vehicle, which we refer to as the target vehicle (TV) and ego vehicle (EV) respectively. In particular, the scenario of interest occurs in a local region of the vehicles' position space $\mathcal{P} \subset \mathbb{R}^2$ and involves the TV executing a parking maneuver, while the EV seeks to navigate safely and efficiently through the narrow parking lot lane (Fig. 4.1).

In our scenario, the vehicles operate at low-speeds where tire slip and inertial effects can be ignored. We therefore model the vehicle dynamics using the kinematic bicycle model [74] given by

$$\dot{z} := \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\psi+\beta) \\ v\sin(\psi+\beta) \\ \frac{v}{l_r}\sin(\beta) \\ a \end{bmatrix}, \ u = \begin{bmatrix} \delta_f \\ a \end{bmatrix}, \tag{4.1}$$

where $\beta = \tan^{-1}\left(l_r \tan(\delta_f)/(l_f + l_r)\right)$ and the following nonlinear, time-invariant, discrete-time system with state $z_k \in \mathcal{Z} \subseteq \mathbb{R}^4$ and input $u_k \in \mathcal{U} \subseteq \mathbb{R}^2$ is obtained by discretizing (4.1) with the 4-th order Runge–Kutta method

$$z_{k+1} = f(z_k, u_k), \tag{4.2}$$

The geometry of the EV is modelled as a rotated and translated rectangle with length L and width W, given as $\mathbb{B}(z_k) = R(\psi_k)\mathbb{B}_0 + [x_k, y_k]^\top$ with $\mathbb{B}_0 := \{p \in \mathbb{R}^2 : Gp \leq g\}$, where $G = [I_2, -I_2]^\top$, $g = [L/2, W/2, L/2, W/2]^\top$, $p = (x, y) \in \mathbb{R}^2$ denotes the position states, and $R(\psi_k) \in SE(2)$ is the rotation matrix corresponding to the heading angle ψ_k.

The TV and lane boundaries comprise the dynamic environment and are parameterized by $\eta_k \in \mathbb{R}^o$. Specifically, we consider M time-varying obstacles in the position space which can be each described as compact polyhedrons, i.e.

$$\mathbb{O}_k^{(m)} := \{p \in \mathbb{R}^2 : A_k^{(m)} p \leq b_k^{(m)}\}, \ m = 1, \dots, M,$$

where $A_k^{(m)} \in \mathbb{R}^{q_m \times 2}$ and $b_k^{(m)} \in \mathbb{R}^{q_m}$ are known matrices at time k, and q_m is the number of faces of the m-th polyhedron. The environment encoding η_k can then be constructed as $\eta_k = \text{vec}\left(\mathbb{O}_k^{(1:M)}\right) = \text{vec}\left(A_k^{(1:M)}, b_k^{(1:M)}\right)$. In the examples presented in this paper, we have $M = 3$.

At every discrete time step k, we require that the EV remain collision-free with all M obstacles. This is expressed as the constraint

$$\text{dist}\left(\mathbb{B}(z_k), \mathbb{O}_k^{(m)}\right) \geq d_{\min}, \ \forall m = 1, \dots, M, \tag{4.3}$$

which enforces a minimum safety distance $d_{\min} > 0$ between the EV and obstacles. We assume that measurements and predictions of the time-varying obstacles are available over an N-step time horizon, i.e., at time step k, we have access to $\boldsymbol{\eta}_k = \{\eta_k, \eta_{k+1}, \dots, \eta_{k+N}\}$, and that there is no mismatch between the predictions and realized trajectories of the time-varying obstacles. The navigation task is then defined as a constrained tracking problem for the reference $\mathbf{z}_k^{\text{ref}}$.

Our approach to maneuvering in tightly-constrained environments revolves around selecting a high-level strategy s_k from a finite and discrete set $\mathcal{S}$, which encode prior knowledge of the behavioral modes available to the EV. In the case of our two-vehicle scenario, we use the following high-level strategies, which are illustrated in Fig. 4.2.

(i) **Pass Left**: The EV passes the TV from the left side;

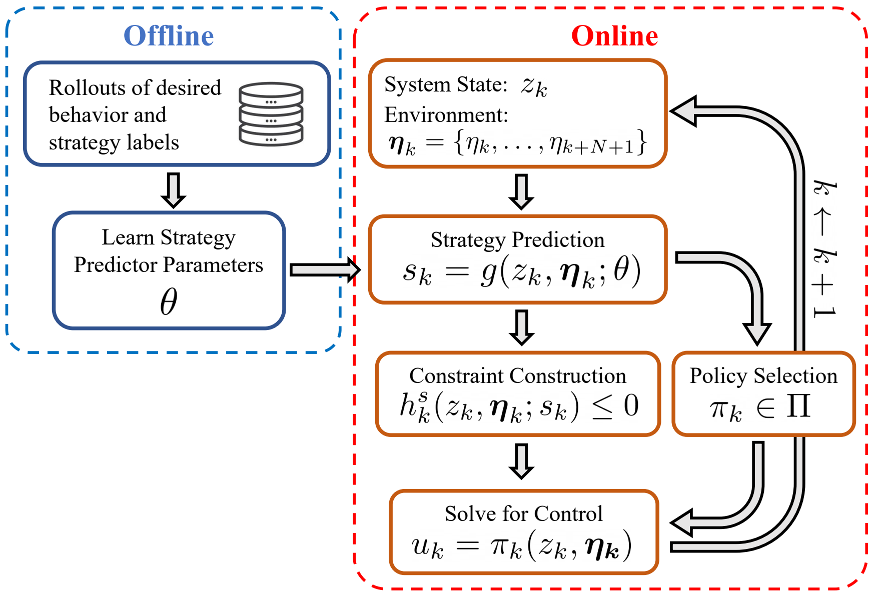

Offline
Rollouts of desired behavior and strategy labels
Learn Strategy Predictor Parameters θ
Online
System State: z_k
Environment: $\boldsymbol{\eta}_k = \{\eta_k, \ldots, \eta_{k+N+1}\}$
Strategy Prediction $s_k = g(z_k, \boldsymbol{\eta}_k; \theta)$
Constraint Construction $h_k^s(z_k, \boldsymbol{\eta}_k; s_k) \leq 0$
Policy Selection $\pi_k \in \Pi$
Solve for Control $u_k = \pi_k(z_k, \boldsymbol{\eta_k})$
$k \leftarrow k+1$

Figure 4.3: The proposed strategy-guided control scheme

(ii) **Pass Right**: The EV passes the TV from the right side;

(iii) **Yield**: The EV yields to the TV for safety.

Remark 6. The strategy set $\mathcal{S}$ is provided by the designer and will require domain expertise in its construction. The problem of identifying salient strategies given a task is outside the scope of this work.

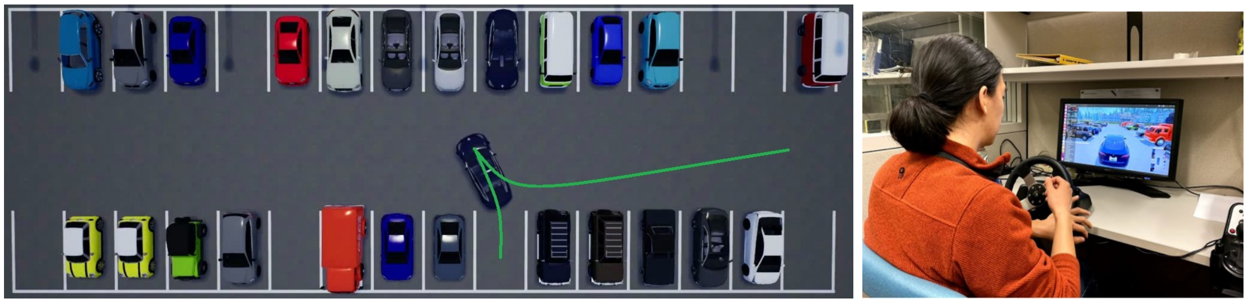
Summary of Approach

The constrained tracking task of interest is well-suited for MPC [25]. However, control horizon lengths, for which real-time MPC is possible, may result in myopic behavior leading to collision in tightly-constrained environments.

Our proposed strategy-guided control scheme for collision avoidance, shown in Fig. 4.3, attempts to address this issue by leveraging the high-level strategy as a surrogate for describing long-term behavior in the presence of other vehicles. In particular, we design a strategy predictor

$$s_k = g(z_k, \boldsymbol{\eta}_k; \theta), \tag{4.4}$$

and train the parameters θ in (4.4) using a database of human-driven parking maneuvers in a simulated environment, and locally optimal solutions to the EV navigation tasks corresponding to each of the TV maneuvers. We leverage the expressiveness of data-driven methods for strategy selection, as opposed to an end-to-end architecture directly for control as such grey-box approaches typically exhibit poor sample efficiency and require copious amounts of data.



(a) Parking Lot and Recorded Maneuver (b) Subject Driving

Figure 4.4: Data Collection in CARLA

For the lower-level online strategy-guided control scheme, we construct three policies:

$$\Pi^{\mathrm{SG}} = \{\pi^{\mathrm{SG-OBCA}}, \pi^{\mathrm{SC}}, \pi^{\mathrm{EB}}\}, \tag{4.5}$$

where $\pi^{\mathrm{SG-OBCA}}$ is a nominal MPC controller based on OBCA [156, 157], which uses the selected strategy to generate hyperplane constraints which help reduce myopic behavior and improve the ability of the EV to complete its navigation task. π^{SC} is a safety controller which can be activated due to infeasibility of SG-OBCA or ambiguity in the behavior of the TV. π^{EB} is an emergency brake controller, to be triggered in the event of impending collision. Conditions for switching between the three policies are discussed in Section 4.4.

4.3 Offline Learning of the Strategy Predictor

Data Collection

We begin with raw data in the form of a set of human-driven parking maneuvers (Fig. 4.4a). These were collected using the CARLA simulator [42] in a custom parking lot [126], where the subject (driver) controls the brake, throttle, and steering of the vehicle via a Logitech G27 racing wheel and pedals (Fig. 4.4b). For each parking trial, the static vehicles in the parking lot are randomly placed so that the subject can park into different spots either forwardly or reversely.

Given K recorded TV parking maneuvers, for the i-th recording of length $T^{[i]}$, $i = 1, \dots, K$, the environment encoding is constructed as $\boldsymbol{\eta}^{[i]}_{0:T^{[i]}} = \left\{\mathrm{vec}\left(\mathbb{O}_k^{(1:M),[i]}\right)\right\}_{k=0}^{T^{[i]}}$, which contains the lane boundaries and the time-varying TV obstacles over the entire task. After obtaining $\boldsymbol{\eta}^{[i]}_{0:T^{[i]}}$, solving for the locally optimal collision-free EV trajectory $\{\mathbf{z}^{[i],*}, \mathbf{u}^{[i],*}\}$ becomes a finite-time optimal control problem, which can be done using the OBCA [156] algorithm.

Lastly, given the strategy set $\mathcal{S}$ ={"Pass Left", "Pass Right", "Yield"}, the strategy label $s^{[i]} \in \mathcal{S}$ for the i-th task can be generated automatically by looking at the relative configurations of the EV and TV over the EV solution trajectory, as exemplified in Fig. 4.2.

Strategy Predictor Training

Using the method described in Sec. 4.3, we collect K locally optimal TV-EV rollouts and their corresponding strategy labels offline. Now, given the horizon length N, the training dataset is constructed as $\mathcal{D} = \{(X_0^{[1]}, s^{[1]}), \ldots, (X_{T^{[1]}-N}^{[1]}, s^{[1]}), (X_0^{[2]}, s^{[2]}), \ldots, (X_{T^{[K]}-N}^{[K]}, s^{[K]})\}$. We note that the strategy labels remain constant within the i-th task since the behavioral mode for each TV-EV rollout is fixed. Each data point $X_k^{[i]}$ consists of the current state of the EV and the environment encoding along the N-step horizon, i.e., $X_k^{[i]} = \text{vec}\left(z_k^{[i],*}, \boldsymbol{\eta}_k^{[i]}\right)$, $k \in \{0, \ldots, T^{[i]} - N\}$. Using this dataset, we train a neural network (NN) for the predictor $g(\cdot)$ in (4.4) to predict the strategy s_k from z_k and $\boldsymbol{\eta}_k$ at every time step. The network architecture is composed of a fully connected hidden layer with the tanh activation function and a softmax output layer. The objective function that we minimize is the cross-entropy loss for classification tasks.

4.4 Online Strategy Prediction and Control

At time step k of the online task execution, the trained network with parameters θ^* returns a probability distribution over the strategy set $\mathcal{S}$, i.e. $\hat{Y}_k = \text{NN}(z_k, \boldsymbol{\eta}_k; \theta^*)$. We refer to $\hat{Y}_k \in \mathbb{R}^3$ as the "confidence" of strategies in this work. The predicted strategy is chosen to be the one with the highest confidence, i.e., $\hat{s}_k = g(z_k, \boldsymbol{\eta}_k; \theta^*) = \arg\max_{s \in \mathcal{S}} \hat{Y}_k$. By leveraging the predicted confidences $\hat{Y}_k$ and selected strategy $\hat{s}_k$, we proceed to construct time-varying hyperplanes and select the control policy π_k from the set Π^{SG}.

Constructing Constraints from Strategies

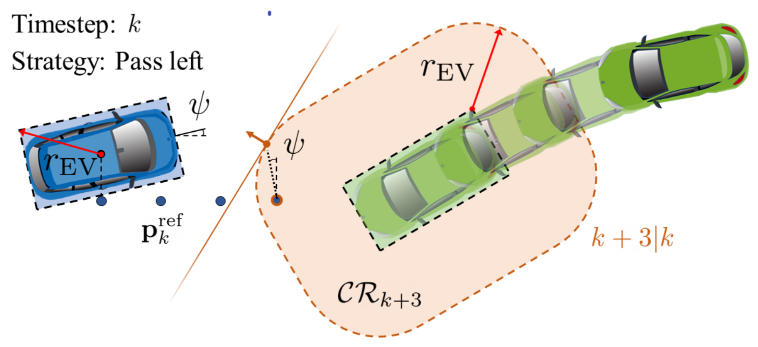
Similar to [138], in this work, we view strategies as lower dimensional encodings of long term relative behavior between the controlled system and the dynamic environment. As such, we now describe how the chosen strategy at each timestep can be used to generate constraints for online control which help guide the EV into regions of space where it is more likely to successfully navigate around the TV.

For the TV occupying the region $\mathbb{O}_k^{\text{TV}}$ at timestep k, we define the critical region as:

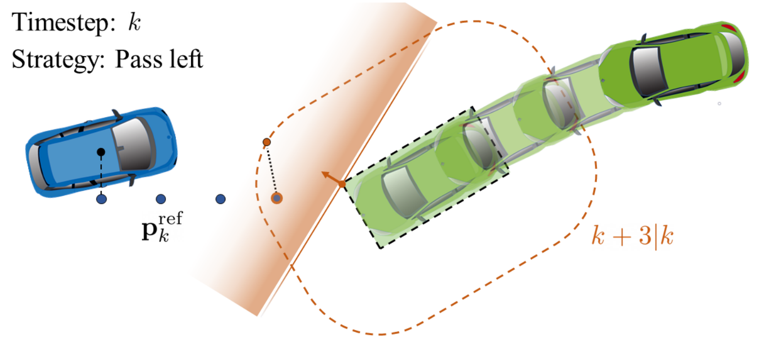
$$\mathcal{CR}_k = \mathbb{O}_k^{\text{TV}} \oplus \mathcal{B}_2(0, r_{\text{EV}}),$$

where $r_{\text{EV}} = \sqrt{L_{\text{EV}}^2 + W_{\text{EV}}^2}/2$ is the radius of the smallest disk which covers the extents of the EV as shown in Figure 4.5a. The critical region represents the area containing the TV where the collision avoidance constraints are close to or have become active.

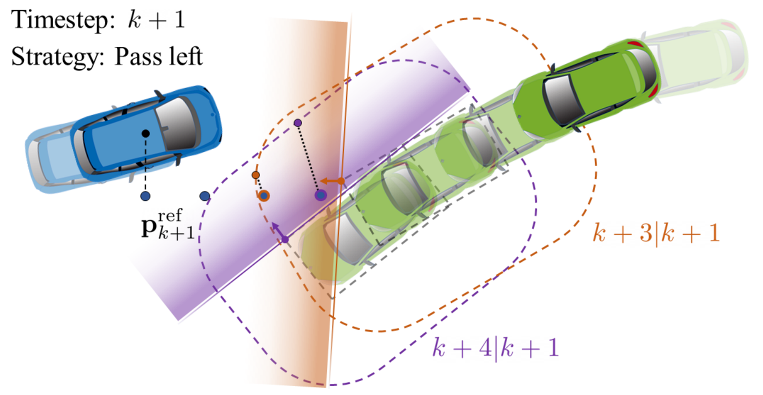
Recall that we are interested in the tracking problem where we are given the N-step reference trajectory $\mathbf{z}_k^{\text{ref}} = \{z_{k|k}^{\text{ref}}, \ldots, z_{k+N|k}^{\text{ref}}\}$ at time step k. In the scenario presented in Section 4.2, this is simply a constant velocity trajectory which follows the centerline of the lane. Denote the position states of the reference trajectory as $\mathbf{p}_k^{\text{ref}}$.



(a)



(b)



(c)

Figure 4.5: Illustration of the strategy-guided constraint generation procedure with $N = 3$ and strategy "Pass Left". a) The position reference of the EV at timestep $k+3$ falls within the critical region $\mathcal{CR}_{k+3}$. The reference point is projected to the boundary of the critical region in the positive body lateral direction and the tangent plane is found. b) The tangent plane is translated to be coincident with the TV boundary. c) Result of the constraint generation procedure at timestep $k+1$.

The main idea behind our constraint generation procedure is to project positions along the reference trajectory, which fall within the critical region (i.e., $p^{\text{ref}}_{k+t|k} \in \mathcal{CR}_{k+t}$ for some $t \in \{0, \dots, N\}$), to the boundary of the critical region $\partial\mathcal{CR}_{k+t}$. The direction of projection is determined by the chosen strategy $\hat{s}_k$ of either "Pass Left" or "Pass Right". In particular, the two aforementioned strategies correspond with the directions $(-\sin\psi, \cos\psi)$ and $(\sin\psi, -\cos\psi)$ respectively, i.e., the positive and negative body lateral axis directions in the inertial frame. Since the boundary of the critical region is smooth by definition, we can find a unique tangent plane at the projection point with an outward facing normal vector $w_{k+t|k}(\eta_{k+t}, \hat{s}_k)$ as can be seen in Figure 4.5a. The final step involves translating the tangent plane such that it is coincident with the boundary of $\mathbb{O}_k^{\text{TV}}$ as is shown in Figure 4.5b. This procedure results in a sequence of additional hyperplane constraints for time steps $k+t$ where $p^{\text{ref}}_{k+t|k} \in \mathcal{CR}_{k+t}$, $t \in \{0, \dots, N\}$. Denote the hyperplane parameters as $\phi(\eta_{k+t}, \hat{s}_k) = -\text{vec}(w_{k+t|k}(\eta_{k+t}, \hat{s}_k), b_{k+t|k}(\eta_{k+t}, \hat{s}_k))$ and the augmented state $\bar{z}_{k+t|k} = \text{vec}(z_{k+t|k}, 1)$. The constraints can then be written as

$$\phi(\eta_{k+t}, \hat{s}_k)^\top \bar{z}_{k+t|k} \leq 0. \tag{4.6}$$

Remark 7. While we may also apply the constraint generation procedure to the "Yield" strategy, we choose to instead deal with it directly by activating a safety controller, which will be described in Sec. 4.4.

Strategy-Guided Optimization-Based Collision Avoidance

For the nominal MPC policy $\pi^{\text{SG}-\text{OBCA}}$ in Π^{SG}, we leverage the generated constraints to extend the OBCA algorithm [156] which formulates the collision avoidance constraints using exact vehicle geometry to enable tight maneuvers in narrow spaces. By introducing the dual variables $\lambda^{(m)} \in \mathbb{R}^{q_m}, \mu^{(m)} \in \mathbb{R}^4$ associated with the m-th obstacle, we obtain a smooth reformulation of the collision avoidance constraint in (4.3). The resulting NLP is written as follows:

$$\begin{aligned}
\min_{\mathbf{z},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\mu}} \quad & \sum_{t=0}^{N} c(z_{k+t|k}, u_{k+t|k}, z^{\text{ref}}_{k+t|k}) \\
\text{s.t.} \quad & z_{k+t+1|k} = f(z_{k+t|k}, u_{k+t|k}),\ t = 0, \dots, N-1 && \text{(4.7a)} \\
& z_{k|k} = z_k, && \text{(4.7b)} \\
& \left(A^{(m)}_{k+t|k} t(z_{k+t|k}) - b^{(m)}_{k+t|k}\right)^\top \lambda^{(m)}_{k+t|k} - g^\top \mu^{(m)}_{k+t|k} > d_{\min} && \text{(4.7c)} \\
& G^\top \mu^{(m)}_{k+t|k} + R(z_{k+t|k})^\top A^{(m)\top}_{k+t|k} \lambda^{(m)}_{k+t|k} = 0 && \text{(4.7d)} \\
& \left\| A^{(m)\top}_{k+t|k} \lambda^{(m)}_{k+t|k} \right\| \leq 1, && \text{(4.7e)} \\
& \lambda^{(m)}_{k+t|k} \geq 0, \mu^{(m)}_{k+t|k} \geq 0, && \text{(4.7f)} \\
& \phi(\eta_{k+t}, \hat{s}_k)^\top \bar{z}_{k+t|k} \leq 0, && \text{(4.7g)} \\
& \forall t = 0, \dots, N, \quad m = 1, \dots, M,
\end{aligned}$$

where $c(z_{k+t|k}, u_{k+t|k}, z^{\text{ref}}_{k+t|k}) = \|z_{k+t|k} - z^{\text{ref}}_{k+t|k}\|^2_{Q_z} + \|u_{k+t|k}\|^2_{Q_u} + \|\Delta u_{k+t|k}\|^2_{Q_d}$ penalizes the deviation from the reference trajectory, input magnitude and rate, with positive semidefinite matrices Q_z, Q_u, and Q_d, respectively. (4.7a) are the dynamics constraints over the control horizon, (4.7b) sets the initial condition, (4.7c)-(4.7f) are the smooth and exact reformulations of (4.3) using Theorem 2 in [156], and (4.7g) are the strategy-guided hyperplane constraints. We call the NLP in (4.7): SG-OBCA.

Remark 8. Problem (4.7) is non-convex, therefore providing a good initial guess to warm-start the NLP solver is crucial to maintain feasibility. For $\mathbf{z}$ and $\mathbf{u}$, it is straightforward to use the solution from the previous iteration as the initial guess. The initial guess for the dual variables $\boldsymbol{\lambda}$, and $\boldsymbol{\mu}$ are obtained with a similar method as described in [157].

Policy Selection

In our proposed strategy-guided control scheme we defined the set (4.5), which contains three classes of policies[1]: SG-OBCA Control $\pi^{\mathrm{SG-OBCA}}$, Safety Control π^{SC}, and Emergency Brake π^{EB}. The formulation of each policy and the selection criteria are as follows.
1) SG-OBCA Control: The SG-OBCA Control policy solves the problem (4.7) at every time step k.
Selection Criteria:

(i) Problem (4.7) is solved successfully, **AND**

(ii) $\hat{s}_k \in \{$"Pass Left", "Pass Right"$\}$ with high confidence, i.e. $\max_s \hat{Y}_k \geq \xi$, where ξ is a user-defined threshold.

2) Safety Control: The Safety Control policy regulates the relative states between the EV and the TV into a safe control invariant set with respect to the two vehicles. In this work, we define this set to be the states with zero relative speed, such that the distance between EV and TV is preserved.
Selection Criteria:

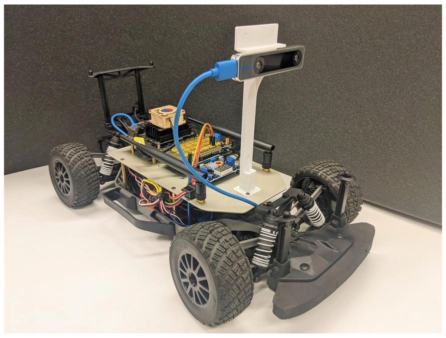
(i) Problem (4.7) is infeasible, **OR**

(ii) $\hat{s}_k =$ "Yield" with $\max_s \hat{Y}_k \geq \xi$, **OR**

(iii) $\max_s \hat{Y}_k < \xi$

3) Emergency Brake: We define the Emergency Brake policy in the context of impending system failure. When the Emergency Break policy is selected, it means that the autonomous agents cannot resolve the situation by either $\pi^{\mathrm{SG-OBCA}}$ or π^{SC} and human intervention is necessary.
Selection Criteria: A collision is anticipated and cannot be resolved by either $\pi^{\mathrm{SG-OBCA}}$ or π^{SC}.

[1]Due to space limit, the policy classes present here are an abstraction of the actual implementation.


(a) BARC

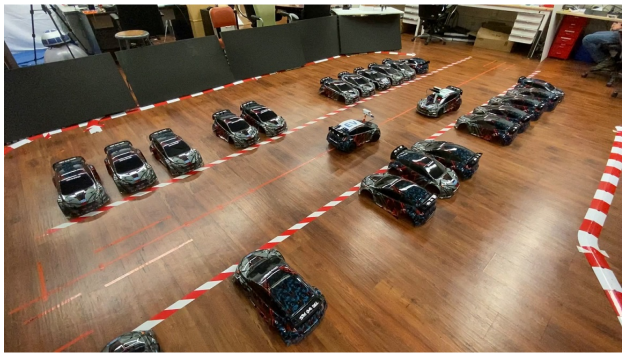

(b) Parking Lot Environment

Figure 4.6: Experimental Setup

Table 4.1: Failure Rate and Iterations to Task Completion

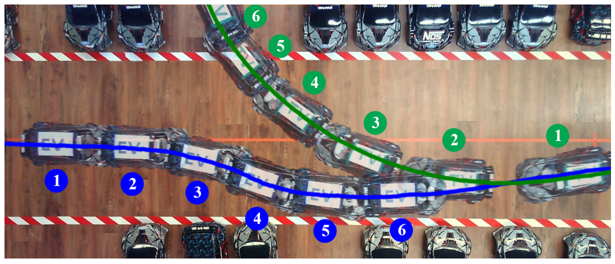
Control Method	Failure Rate	Avg. # Iter	Median # Iter
Baseline	30%	213	220
Strategy-Guided	**6%**	**270**	**227**

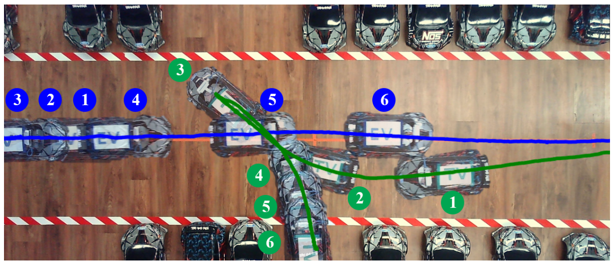
4.5 Simulation and Experimental Results

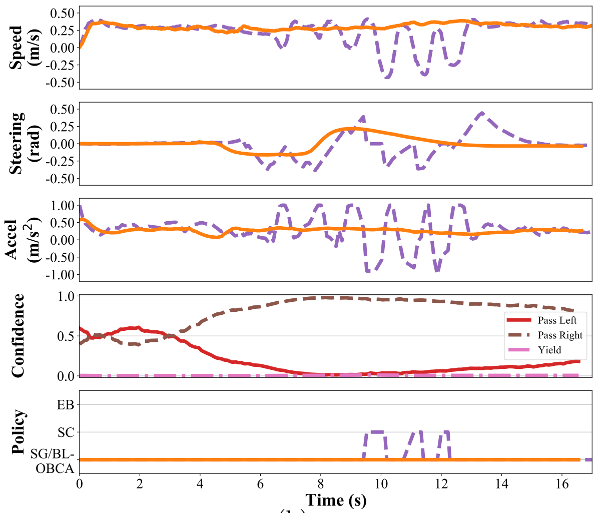
In this section, we report results from simulations and experiments with the proposed strategy-guided control approach (SG) as described in Sec. 4.4. We compare the outcomes with a baseline control scheme (BL), where the $\pi^{\mathrm{SG-OBCA}}$ in (4.5) is replaced with $\pi^{\mathrm{BL-OBCA}}$ which solves (4.7) without the additional constraints (4.7g). Moreover, for the baseline control scheme, π^{SC} is selected only when $\pi^{\mathrm{BL-OBCA}}$ is infeasible. In both simulation and hardware experiments, we chose a horizon length of $N = 20$ with a time step of $dt = 0.1s$. The safety distance $d_{\min}$ in (4.7c) is set to 0.01m and we set the confidence threshold to be 0.75. As described in Section 4.3, we recorded a total of 486 navigation tasks with TV maneuvers from which we generate $103,553$ labeled examples for the training dataset $\mathcal{D}$. Details can be found at: `bit.ly/data-sg-control`.

All tasks are attempted by both SG and BL control schemes in a MATLAB simulation environment. Table 4.1 reports the failure rate, where a collision occurs or the π^{EB} policy is selected, and the number of iterations required to complete the task. We can see that in simulation, the SG control scheme reduces the failure rate significantly, without sacrificing much in terms of task performance.

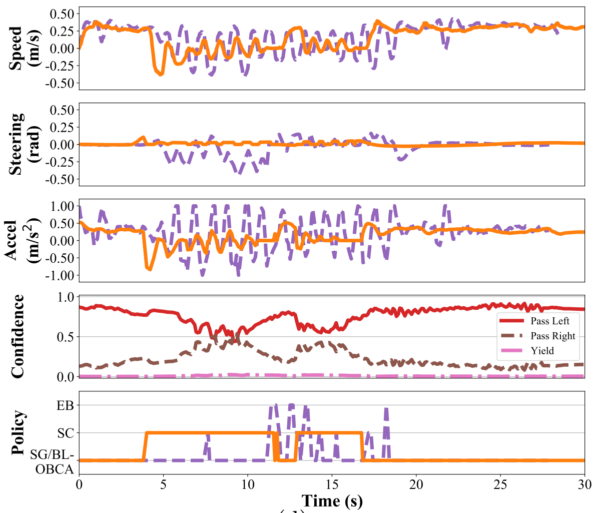
Both BL and SG approaches were implemented with the same cost matrices $Q_z = \mathrm{diag}([1,1,1,10])$, $Q_u = \mathrm{diag}([1,1])$, and $Q_d = \mathrm{diag}([50,50])$, on the 1/10 scale open source Berkeley Autonomous Race Car (BARC) platform in a laboratory parking lot environment (Fig. 4.6). The cars are equipped with an Intel RealSense T265 tracking camera for localization and an NVIDIA Jetson Nano for onboard computation. We use FORCESPRO [154, 41] to compile the NLP solver, which achieves an average solution time of 30ms for problem


(a)


(c)



(b)



(d)

Figure 4.7: a), c) Trajectories of the EV (blue) and TV (green) under the strategy-guided control scheme. Frame numbers are marked with circles; b), d) (From top to bottom) Speed and input profile of the EV under SG (solid orange) and BL (dashed purple). Confidence of strategy prediction and policy selection for the strategy-guided controller.

(4.7). 99.8% of feasible solutions are returned in less than 100ms. The strategy and policy selection, constraint generation, and NLP solver all run on a host laptop with an Intel i9 processor and 32 GB of RAM. Communication with the cars is done through the Robot Operating System (ROS). The human driven TV was represented by another car which tracked the recorded trajectories using nonlinear MPC. The MPC predictions at each time step are used to construct $\mathbb{O}_k^{\text{TV}}$ over the EV control horizon.

Results from the first navigation task are presented in Figs. 4.7a and 4.7b where the TV parks into an empty spot in the top row. The predicted strategy and selected policy over the task are visualized in Fig. 4.7b. It is clear that at the beginning of the task (before $t = 4$s), the intention of the TV is ambiguous and the confidence in the “Pass Left” strategy is slightly higher because the TV veers to the right hand side of the EV. However, once the TV begins its parking maneuver, the “Pass Right” strategy is identified with high confidence. Under SG, the EV is able to leverage the hyperplane constraints generated via the identified strategy to preemptively begin its maneuver around the TV, thereby resulting in a smoother speed and input profile compared to BL as can be seen in Fig. 4.7b. In contrast, BL performs a more aggressive and dangerous maneuver as it is only constrained by the safety distance $d_{\min}$. This results in a fragile policy which can easily lead to constraint violation when the EV is in the critical region. Indeed, in Fig. 4.7b, we observe that the BL triggers the Safety Control policy for several time steps due to infeasibility caused by the aggressive behavior.

Results from a second navigation task are presented in Figs. 4.7c and 4.7d where the TV parks in reverse into an empty spot in the bottom row and a gear change is needed. The strategy and policy selections over the task are visualized in Fig. 4.7d, where the SG control scheme selects Safety policy π^{SC} at an early stage and continues to yield to the TV for a considerable amount of time due to low confidence in the selected strategy. This is caused by the time the TV spends idling during the gear change. In contrast, the short-sighted baseline controller again exhibits overly-aggressive behavior by trying to pass the TV while it is idle. This eventually leads to activation of the policy π^{EB} and collision with the TV. Finally, we again see in Fig. 4.7d that SG results in a smoother speed and input profile when compared to BL.

4.6 Chapter Summary

In this chapter, we show that when compared to a baseline MPC approach, the proposed hierarchical data-driven control scheme significantly improves the success rate of a navigation task in a tightly-constrained multi-agent dynamic environment. The design of a data-driven strategy predictor provides a structured and transparent approach to leveraging data in control, which is crucial to expanding the adoption of learning techniques in the control of safety critical systems.

The purpose of strategy-guided constraint generation and policy selection are to coerce the system towards regions of space where successful completion (in terms of recursive feasibility and stability) of the control problem is most likely.

One interpretation of the strategy-guided constraints is that they are surrogates of the MPC terminal components, where given the selected strategy, we attempt to construct constraints which drive the state of the system (4.2) into a set which is contractive towards the reference $\mathbf{z}^{\mathrm{ref}}$ while satisfying all constraints. Despite not affording the rigorous guarantees provided by an exact terminal cost function and terminal set, the value of our proposed surrogates lies in the fact that they can be straightforward to construct even when the exact formulations are intractable, which is typically the case for complex control tasks in dynamic environments.

In terms of the policy selection, predicting the strategy allows the system to "prepare" for an upcoming safety-critical encounter by selecting an appropriate control policy. In the context of this work, when faced with ambiguous behavior from the TV, the strategy-guided control approach can select the safety control policy, so that the system will only attempt the risky passing maneuver when it is confident enough to do so. In contrast, the baseline control scheme only reverts to the safety control when the nominal MPC becomes infeasible.

This is usually at a time when the situation is already very challenging for the system to maintain safety.

In the next chapter, we will continue to explore this idea of using a data-driven learned model for constraint generation in optimal control. In particular, we will begin focusing on the application of autonomous racing, which is a competitive multi-agent scenario with no inter-agent communication.

Chapter 5

A Gaussian Process Model for Opponent Prediction in Autonomous Racing

5.1 Introduction

A major challenge in automated vehicles is the modeling of interactions between agents in highly dynamic constrained environments. Knowledge of such interactions is crucial for short-term decision making and can have a significant impact on the vehicle's safety and performance. This is especially relevant in situations where information is not shared between agents and inference must be performed to obtain a prediction of future plans of the other agents in the environment. In this chapter, we are interested in the context of racing where agents are each driven by competitive and possibly adversarial policies. Maximizing performance in racing typically requires the agent to operate in highly dynamic regimes, e.g. at the limits of tire friction, where good models are vital to maintaining stability of the vehicle. At the same time, as it is unlikely that a perfect interaction model can be obtained, it is crucial to reason about the uncertainty of the generated predictions, which can help maintain the delicate balance between safety and performance.

Obtaining exact knowledge of an opponent's dynamics and its racing policy is unlikely. However, it is certainly possible that data about them is available from past races. Our contributions are as follows:

- We propose a Gaussian process (GP) based prediction model which learns the *one-step* closed-loop behavior of an opponent, or target (TV), vehicle from a database of interactions and obtains trajectory predictions and the associated uncertainties via sampling.

- In order to show the benefits of the proposed approach, we formulate a model predictive control (MPC) policy for the ego vehicle (EV) which leverages the time-varying uncertainties provided by the GP-based predictor by constructing uncertainty-expanded collision avoidance constraints to balance performance and safety in a head-to-head racing scenario.

- We demonstrate through a Monte Carlo simulation study that the GP-based predictor achieves similar win rates while maintaining safety in up to 3x more races when compared to predictors in literature which are based on optimal control and vehicle dynamics.
- We finally demonstrate the prediction and control framework in real-time in an experimental study on a 1/10th scale racecar platform operating at high speeds of around 2.8 m/s and show a significant level of improvement over a baseline predictor.

The results presented in this chapter have also appeared in:

- E. Zhu, F. Busch, J. Johnson, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 8186–8191.

Related Work

Recent research in the construction of driver prediction models has focused on utilizing learning-based methods for short-term trajectory prediction. Notably, methods such as [61], [39], and [158] infer high level human intent, predicting agent trajectories based on goal states given a semantic map. With respect to prediction under uncertainty, [61] models uncertainty in other road agents with a bivariate Gaussian over every future position, whilst considering short-term agent history. Much of the above work takes a model-free approach to trajectory forecasting by embedding interactions into a surrounding "scene context". Alternatively, [118] proposes a graph-based prediction method that incorporates agent dynamic constraints and [36] explicitly embeds kinematic feasibility into the last layer of the prediction model. While these deep learning based methods can certainly be applied in the context of racing, their use of semantic images as model inputs leads to the requirement of large motion datasets for training. In contrast, our approach leverages closed-loop state trajectories and can achieve good performance after training on a relatively small dataset of 5000 data points.

There also exists much work which incorporates learned prediction models in closed-loop with optimal control. [139] maps features of the environment to strategy states through a GP, which are then used to construct MPC terminal constraints. This approach is complementary to our proposed method in the sense that it predicts the free space available to the EV under a certain environment configuration instead of the space occupied by the TV. [81] and [152] propose learning a mapping from vehicle state history to a finite set of behavioral parameters with a hidden Markov model and GP respectively. In [81] these parameters are used to modify an MPC reference whereas in [152] they are used to construct a measurement model which allows for trajectory predictions of other agents using an extended Kalman filter. Compared to these approaches, we not only condition the TV predictions on the current state of all agents, but also on the future plan of the EV. [30] is the most similar to our work, where a GP-based state transition model for the TV is learned and the distribution over the predicted trajectory is used in the construction of tightened half-space collision avoidance constraints for a stochastic MPC. The approach is demonstrated on straight track segments for a linearized kinematic vehicle model. Additionally, a "one-move" rule is assumed for the TV, which restricts its competitiveness. In contrast, we do not make the one-move assumption and formulate the approach for a nonlinear dynamic vehicle model and ellipsoidal collision avoidance constraints.

5.2 Problem Formulation

Vehicle Model

We model the racing agents using the dynamic bicycle model [74], where the vehicle state and input vectors are defined as $z = [p_x, p_y, \phi, v_x, v_y, \omega]^\top$ and $u = [F_x^r, \delta]^\top$. p_x and p_y are the Cartesian coordinates of the vehicle's center of gravity (CoG), ϕ is the vehicle's heading, v_x and v_y are the longitudinal and lateral velocity of the CoG respectively, and ω is the yaw rate. The control inputs to the vehicle are the rear longitudinal tire force F_x^r and the steering angle δ. We discretize the nonlinear dynamics using the 4th order Runge-Kutta method with time step T_s to obtain:

$$z_{k+1} = f(z_k, u_k). \tag{5.1}$$

We additionally make use of the vehicle pose in a curvilinear reference frame w.r.t. the centerline of a track [96], which is represented via the twice continuously differentiable mapping $\tau : [0, L] \mapsto \mathbb{R}^2$, where L is the length of the track and $\tau(0) = \tau(L)$. We denote the first and second derivatives of τ w.r.t. the track progress s as $\tau', \tau'' : [0, L] \mapsto \mathbb{R}^2$. The curvilinear pose can be computed from the global position $p = [p_x, p_y]^\top$ and heading angle ϕ, and consists of the vehicle's track progress $s(p) = \arg\min_s \|\tau(s) - p\|_2$, its lateral deviation from the centerline $e_y(p) = \min_s \|\tau(s) - p\|_2$, and its heading deviation from the centerline tangent angle $e_\phi(p, \phi) = \phi - \arctan(\tau'(s(p)))$. We can additionally define the signed curvature of the track at some $s \in [0, L]$ as $|\kappa(s)| = \|\tau''(s)\|_2$, where the sign of κ depends on the sign of the swept angle rate for the track at s. We distinguish between quantities for the EV and TV by adding a superscript, e.g. $z^{(1)}$ and $z^{(2)}$ for the state of the EV and TV respectively.

Model Predictive Contouring Control (MPCC)

In a head-to-head competition, it would be incorrect to assume that knowledge of the opponent's plans or their racing control policy are freely available. We therefore require a prediction model to generate a forecast of the opposing agent's behavior before computing our own control action. Design of such a prediction model is a key contribution of this work and will be described in further detail soon. Let us define the prediction model $\mathcal{P}$ for some horizon of length N as a function of the TV's current state $z_k^{(2)}$, which we assume can be accurately obtained through measurement, a sequence of EV states $\mathbf{z}_k^{(1)} = \{z_k^{(1)}, \dots, z_{k+N}^{(1)}\}$, which describes the EV's planned trajectory over the next N time steps, and hyperparameters θ:

$$\hat{\mathbf{z}}_k^{(2)} = \mathcal{P}(z_k^{(2)}, \mathbf{z}_k^{(1)}, \theta). \tag{5.2}$$

At time step k, given a prediction $\hat{\mathbf{z}}_k^{(2)}$ from (5.2), we define a baseline optimal control problem for the EV adapted from MPCC [87]:

$$\begin{aligned}
&\mathbf{u}^{(1),\star}(z_k^{(1)}, \hat{\mathbf{z}}_k^{(2)}, u_{k-1}^{(1)}) = \\
&\arg\min_{\mathbf{u}} \quad \sum_{t=0}^{N-1} q_c^{(1)} e_c(p_t, \bar{s}_t)^2 + u_t^\top R^{(1)} u_t \\
&\qquad\qquad\qquad + \Delta u_t^\top R_d^{(1)} \Delta u_t - q_s^{(1)} \bar{s}_N && (5.3a)\\
&\text{s.t.} \quad z_0 = z_k^{(1)} && (5.3b)\\
&\quad z_{t+1} = f^{(1)}(z_t, u_t), \ t = 0, \dots, N-1 && (5.3c)\\
&\quad \bar{s}_0 = s(p_0) && (5.3d)\\
&\quad \bar{s}_{t+1} = \bar{s}_t + T_s v_{x,t}, \ t = 0, \dots, N-1 && (5.3e)\\
&\quad z_t \in \mathcal{Z}, \ t = 0, \dots, N && (5.3f)\\
&\quad u_t \in \mathcal{U}, \ t = 0, \dots, N-1 && (5.3g)\\
&\quad h(z_t, \hat{z}_{k+t}^{(2)}) \leq 0, \ t = 0, \dots, N, && (5.3h)
\end{aligned}$$

where $\mathbf{u} = \{u_0, \dots, u_{N-1}\}$, $\Delta u_t = u_t - u_{t-1}$, and $u_{-1} = u_{k-1}^{(1)}$. The objective (5.3a) is to maximize progress while following the track. These goals are quantified by $\bar{s}_N$ and e_c, which are approximations of track progress and the centerline deviation respectively [87], and are weighted by parameters $q_s^{(1)}$, $q_c^{(1)} > 0$. We additionally wish to minimize control amplitude and rate via $R^{(1)}$, $R_d^{(1)} \succ 0$. Vehicle state and approximate track progress evolution are governed by the dynamics in (5.3b)-(5.3e). We impose track boundary and input constraints via (5.3f) and (5.3g) as follows:

$$\mathcal{Z} = \{z \mid -W/2 \leq e_y(p) \leq W/2\} \tag{5.4}$$

$$\mathcal{U} = \{u \mid u_l \leq u \leq u_u\}. \tag{5.5}$$

Where W is the width of the track and u_l and u_u are the lower and upper bounds on the input values. Constraint (5.3h) describes coupling between the predicted EV and TV trajectories which, for the purposes of this work, represent collision avoidance constraints and will be expanded upon later. Using the optimal open-loop sequence from (5.3), we can define the EV feedback policy:

$$\pi^{(1)}(z_k^{(1)}, \hat{\mathbf{z}}_k^{(2)}, u_{k-1}^{(1)}) = u_0^{(1),\star}(z_k^{(1)}, \hat{\mathbf{z}}_k^{(2)}, u_{k-1}^{(1)}), \tag{5.6}$$

which can be applied in a receding horizon fashion. See [87] for more details on the formulation of MPCC.

5.3 Learning TV Behavior via Gaussian Processes

In any competitive scenario, information about one's capabilities and strategy are closely guarded. We therefore make few assumptions about knowledge of the TV's dynamics or racing policy and simply define the evolution of the TV's state as follows:

$$z_{k+1}^{(2)} = f^{(2)}(z_k^{(2)}, \pi_k^{(2)}(z_k^{(2)}, z_k^{(1)})) = g_k(z_k^{(2)}, z_k^{(1)}), \tag{5.7}$$

where $f^{(2)}$ and $\pi_k^{(2)}$ are unknown, but are assumed to be smooth and g_k denotes the closed-loop dynamics of the TV given some EV state $z_k^{(1)}$.

We assume that a dataset $\mathcal{D}$ of head-to-head racing interactions, which are generated by a stationary TV policy, i.e. $\pi_k^{(2)} = \pi^{(2)}$, is available. Our approach is to construct a probabilistic one-step model of the TV, which approximates g_k in (5.7) and captures the associated prediction uncertainty. This would allow us to roll out a TV prediction trajectory at time step k given a TV state $z_k^{(2)}$ and EV plan $\mathbf{z}_k^{(1)}$ while maintaining a measure of the uncertainty of our predictions. This is crucial to balancing safety and performance during the highly dynamic and tightly constrained interactions which are commonplace in racing. We note that in the case of non-stationary policies, which are selected from a finite set (e.g. for overtaking, blocking, station keeping, etc.), prediction models can be constructed for each policy option given that the dataset is indexed by policy.

Gaussian Processes

We use GP regression [147] to model the unknown one-step closed-loop dynamics of the target vehicle (5.7). Consider the dataset constructed from D pairs of observations:

$$\mathcal{D} = \{(x^{[i]}, y^{[i]})\}_{i=1}^{D}, \tag{5.8}$$

where $x^{[i]} \in \mathbb{R}^n$ and $y^{[i]} \in \mathbb{R}^m$ are generated by the unknown statistical process:

$$y^{[i]} = g(x^{[i]}) + w^{[i]}, \tag{5.9}$$

with i.i.d. $w^{[i]} \sim \mathcal{N}(0, \mathrm{diag}(\sigma_1, \dots, \sigma_m))$. Assuming independence between output dimensions, GPs maintain a probability distribution over each of the functions g_j, $j = 1, \dots, m$, and are completely specified by a mean function $\mu_j(x)$ and a scalar covariance function $k_j(x, x')$ (also called the *kernel*). Without loss of generality, given a zero-mean GP prior on g_j, the posterior distribution of g_j is given by the GP:

$$g_j(x) \sim \mathcal{N}(\mu_j(x), \mathrm{Var}_j(x)) \tag{5.10a}$$

$$\mu_j(x) = \mathbf{k}_j(x)^\top (\mathbf{K}_j + \sigma_j^2 I_D)^{-1} Y_j \tag{5.10b}$$

$$\mathrm{Var}_j(x) = k_j(x, x) - \mathbf{k}_j(x)^\top (\mathbf{K}_j + \sigma_j^2 I_D)^{-1} \mathbf{k}_j(x) \tag{5.10c}$$

where

$$\begin{aligned} Y_j &= [y_j^{[1]} \dots y_j^{[D]}]^\top, \\ [\mathbf{K}_j]_{ab} &= k(x^{[a]}, x^{[b]}),\ \ a, b = 1, \dots, D, \\ \mathbf{k}_j(x) &= [k_j(x, x^{[1]})\ \dots\ k_j(x, x^{[D]})]^\top. \end{aligned}$$

In this work, we use the Matérn kernel with parameter $\nu = 1.5$ [147]:

$$k_j(x, x') = \left(1 + \frac{\sqrt{3}\|x - x'\|^2}{l_j}\right) \exp\left(-\frac{\sqrt{3}\|x - x'\|^2}{l_j}\right),$$

where l_j is a length scale parameter which, together with the prior observation variances σ_j, can be optimized by maximizing the marginal likelihood of the observations using gradient-based methods.

Data Generation and Model Training

We construct the dataset $\mathcal{D}$ from simulation rollouts of EV and TV trajectories drawn from head-to-head races. In each of these rollouts, the EV starts behind TV on a randomly generated track. EV uses the policy defined by (5.3) and (5.6) and can win the race by executing a successful overtaking maneuver. On the other hand, TV tries to remain ahead of EV by blocking attempted overtakes. In order to encourage blocking behavior, the TV policy is formulated obtained by switching the superscripts in (5.3) and (5.6), *removing* the collision avoidance constraints (5.3h), and adding to (5.3a) the following term:

$$\sum_{t=0}^{N} q_y^{(2)} \frac{(e_y(p_t^{(2)}) - e_y(p_k^{(1)}))^2}{1 + (s(p_k^{(2)}) - s(p_k^{(1)}))^2} \tag{5.11}$$

with $q_y^{(2)} > 0$ which rewards the TV for maintaining the same lateral position on the track as the EV especially when the two vehicles are close in terms of their track progress. Note that $q_y^{(2)}$ can be interpreted as an aggressiveness factor for the TV's blocking attempts. In addition, it is important to note that this cost term only depends on the state of the EV at the current time step k and does not require a prediction of the EV's trajectory over the MPC horizon. As such the overall TV policy is only a function of the EV's state at time step k. During these data generation rollouts, the predictions $\hat{\mathbf{z}}_k^{(2)}$ are replaced with the open-loop solution to (5.3) for the TV. In other words, the EV knows the true plan of its opponent.

The regression features are constructed using the resulting closed-loop trajectories as follows

$$x^{[i]} = [\Delta s_k, \Delta e_{y,k}, e_{y,k}^{(2)}, e_{\phi,k}^{(2)}, v_{x,k}^{(2)}, \omega_k^{(2)}, e_{\phi,k}^{(1)}, v_{x,k}^{(1)}, \kappa_k]^\top,$$

where $\Delta s_k = s(p_k^{(1)}) - s(p_k^{(2)})$, $\Delta e_{y,k} = e_y(p_k^{(1)}) - e_y(p_k^{(2)})$, and κ_k is a vector of track curvatures at the look-ahead points $s(p_k^{(2)}) + i\delta$, $i = 1, \dots, V$ for some chosen V and $\delta > 0$. The regression

Algorithm 6: TV trajectory prediction and uncertainty propagation

```
  Input: $N$, $M$, $z_k^{(2)}$, $\mathbf{z}_k^{(1)}$
1 $z_k^{2,[i]} \leftarrow z_k^{(2)}$, $\forall i = 1, \dots, M$;
2 for $i = 1, \dots, M$ do
3     for $t = 0, \dots, N-1$ do
4         Sample $\Delta z_{k+t}^{(2),[i]} \sim \mathcal{N}(\mu(z_{k+t}^{(2),[i]}, z_{k+t}^{(1)}), \mathrm{Var}(z_{k+t}^{(2),[i]}, z_{k+t}^{(1)}))$;
5         $z_{k+t+1}^{(2),[i]} \leftarrow z_{k+t}^{(2),[i]} + \Delta z_{k+t}^{(2),[i]}$;
6     end
7 end
8 $\hat{z}_{k+t}^{(2)} \leftarrow \frac{1}{M} \sum_{i=1}^{M} z_{k+t}^{(2),[i]}$;
9 $\hat{\Sigma}_{k+t}^{(2)} \leftarrow \frac{1}{M-1} \sum_{i=1}^{M} (z_{k+t}^{(2),[i]} - \hat{z}_{k+t}^{(2)})(z_{k+t}^{(2),[i]} - \hat{z}_{k+t}^{(2)})^\top$;
  Output: $\hat{\mathbf{z}}_k^{(2)}$, $\hat{\Sigma}_k^{(2)}$, ..., $\hat{\Sigma}_{k+N}^{(2)}$
```

targets are chosen as the one-step TV state difference, which are also transformed into the curvilinear frame via $y^{[i]} = \mathcal{C}(z_{k+1}^{(2)}) - \mathcal{C}(z_k^{(2)})$. Evaluation of the one-step predictor can be done with:

$$\hat{z}_{k+1}^{(2)} = \mathcal{C}^{-1}(\mathcal{C}(z_k^{(2)}) + \mu(z_k^{(2)}, z_k^{(1)})), \tag{5.12}$$

where μ is the vector-valued function of posterior means.

Transforming vehicle poses to the curvilinear frame has the effect of reducing the amount of data required to achieve good generalization on a wide variety of tracks. This is crucial for GPs which have complexity $\mathcal{O}(D^3)$ and which rely heavily on a distance metric in the kernel function for training and inference. Consider the case of a GP where inference is performed on a track identical to the one used for training data generation but is rotated arbitrarily. It is easy to see that any vehicle pose on the original track which undergoes the same rotation will have the same coordinate in the new curvilinear frame. However, that is obviously not true for the global pose. Therefore, a model trained on data in the curvilinear frame can be immediately applied to the same track with arbitrary orientation, whereas a global frame alternative would be expected to perform poorly as the spatial relationships learned by the model no longer hold on the rotated track.

The look-ahead curvatures κ_k were included in the feature vector to further improve generalization to unseen track configurations and is motivated by look-ahead fixations of human drivers when driving through curved roads [82].

Trajectory Prediction

After obtaining the GP prediction model, we can now construct a trajectory prediction for the TV and the associated uncertainty over a finite horizon of length N, as conditioned on a given EV plan. We propose a straight-forward sampling-based approach which estimates a nominal TV state and approximately propagates the uncertainty as described by the one-step GP model.

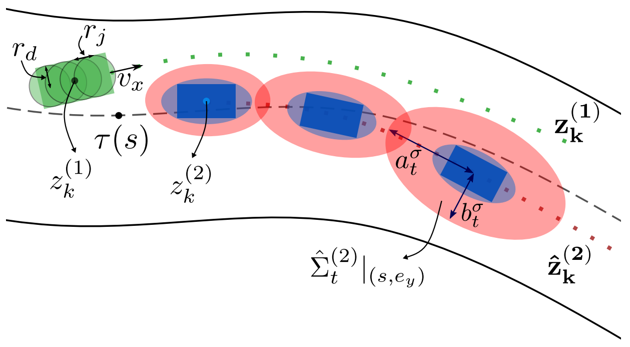

r_d
r_j
v_x
$\tau(s)$
$z_k^{(1)}$
$z_k^{(2)}$
$\mathbf{z}_\mathbf{k}^{(\mathbf{1})}$
a_t^σ
b_t^σ
$\hat{\Sigma}_t^{(2)}|_{(s,e_y)}$
$\hat{\mathbf{z}}_\mathbf{k}^{(\mathbf{2})}$

Figure 5.1: Overview of the EV obstacle avoidance constraints with uncertainty-expanded ellipses pictured in red and vehicle extent ellipses pictured in blue. The EV is represented by 4 discs of radius r_d.

The method is presented in Algorithm 6 and begins with the initialization of M sample trajectories in line 1. Lines 4 and 5 proceed with evaluation of the one-step GP prediction model for each of the M trajectories at time step $k+t$. Note that we omit the transformations in (5.12) for brevity. The resulting distribution is then sampled for the TV state difference, which is then added on to the current TV state to obtain the new state at time $k+t+1$. This procedure is repeated for N steps. In lines 8 and 9, sample means and covariances are computed over the M trajectories at each time step and returned as the predicted TV trajectory and associated uncertainty. As the trajectories as sampled independently, our procedure is amenable to parallelization.

Implementation

We train a GP with independent output dimensions using `GPyTorch` [54] on simulation rollouts where the TV policy's aggressiveness factor is set to $q_y^{(2)} = 200$. To enable real-time inference, this model is trained as a variational approximate GP as described in [58] through an inducing point method. This reduces the overall sample complexity to $\mathcal{O}(D\bar{D}^2)$ with $\bar{D} \ll D$ inducing points. Specifically, we use $\bar{D} = 200$ inducing points for a dataset of size $D = 5000$. Executing Algorithm 6 with $M = 10$ sample trajectories yields an average total run time of 45 ms for a horizon of length $N = 10$ on an NVIDIA GeForce RTX 3080.

5.4 Design of MPCC Racing Policy

In this section, we introduce modifications to (5.3) for the EV, which allow it to leverage the uncertainties provided by Algorithm 6 to intelligently trade off between safety and performance. We first define the original obstacle avoidance constraint introduced in [121], which represents the EV at time step k as a set of four circles with radii r_d and centers $c_{x,k}^{(1),j} = p_{x,k}^{(1)} + r_j \cos(\phi_k^{(1)})$ and $c_{y,k}^{(1),j} = p_{y,k}^{(1)} + r_j \sin(\phi_k^{(1)})$ for user defined coverage constants r_j and $j = 1, \ldots, 4$. The extents of the EV are assumed to be contained within the union of these circles. The TV is represented as the minimum covering ellipse of the vehicle extents, with semi-axis lengths a and b, transformed by its pose $(p_k^{(2)}, \phi_k^{(2)})$. These are illustrated in Fig. 5.1 by the green circles and blue ellipses respectively. The obstacle avoidance constraint can be written for $j = 1, \ldots, 4$ and $t = k, \ldots, k+N$ as

$$\begin{aligned} h_j(z_t^{(1)}, \hat{z}_t^{(2)}) &= 1 \\ &- \frac{((c_{x,t}^{(1),j} - \hat{p}_{x,t}^{(2)}) \cos(\hat{\phi}_t^{(2)}) - (c_{y,t}^{(1),j} - \hat{p}_{y,t}^{(2)}) \sin(\hat{\phi}_t^{(2)}))^2}{a^2} \\ &- \frac{((c_{x,t}^{(1),j} - \hat{p}_{x,t}^{(2)}) \sin(\hat{\phi}_t^{(2)}) + (c_{y,t}^{(1),j} - \hat{p}_{y,t}^{(2)}) \cos(\hat{\phi}_t^{(2)}))^2}{b^2} \end{aligned} \tag{5.13}$$

We propose an uncertainty-expanded safety bound whose area is proportional to the position prediction variance from Algorithm 6. These are illustrated by the red ellipses in Fig. 5.1. Let $\mathrm{Var}(s_t^{(2)})$ and $\mathrm{Var}(e_{y,t}^{(2)})$ be the prediction variances for track progress and lateral deviation on the diagonal of $\hat{\Sigma}_t^{(2)}$. In order to derive the tightened constraint, we transform these variances from the track-aligned curvilinear frame to the body-aligned frame of the TV:

$$\begin{aligned} \mathrm{Var}(a_t) &= \mathrm{Var}(\cos(\hat{e}_{\phi,t}^{(2)}) s_t^{(2)} + \sin(\hat{e}_{\phi,t}^{(2)}) e_{y,t}^{(2)}) \\ &= \cos^2(\hat{e}_{\phi,t}^{(2)}) \mathrm{Var}(s_t^{(2)}) + \sin^2(\hat{e}_{\phi,t}^{(2)}) \mathrm{Var}(e_{y,t}^{(2)}) \\ \mathrm{Var}(b_t) &= \sin^2(\hat{e}_{\phi,t}^{(2)}) \mathrm{Var}(s_t^{(2)}) + \cos^2(\hat{e}_{\phi,t}^{(2)}) \mathrm{Var}(e_{y,t}^{(2)}), \end{aligned}$$

where $\mathrm{Var}(a_t)$ and $\mathrm{Var}(b_t)$ correspond to the variance in the TV's longitudinal and lateral directions respectively. Note that we neglect off-diagonal terms in $\hat{\Sigma}_t^{(2)}$ and the uncertainty in the heading. The semi-axis lengths of the expanded ellipsoidal safety bounds a_t^σ and b_t^σ are then computed as

$$\begin{bmatrix} a_t^\sigma \\ b_t^\sigma \end{bmatrix} = \gamma \begin{bmatrix} \sqrt{\mathrm{Var}(a_t)} \\ \sqrt{\mathrm{Var}(b_t)} \end{bmatrix} (1 - \epsilon_t) + \begin{bmatrix} a \\ b \end{bmatrix}, \tag{5.14}$$

where $\gamma > 0$ is chosen to be the number of standard deviations we would like to account for in our safety bound, and the constraint function (5.3h) is defined by replacing a and b in (5.13) with the semi-axis lengths computed in (5.14). We allow for a certain amount of constraint violation through the slack variable $\epsilon_t \in [0, 1]$. Note that when $\epsilon_t = 1$, the original constraint (5.13) is recovered. We add the following quadratic term to the cost function in (5.3a):

$$\frac{1}{2} \epsilon^\top Q_\epsilon \epsilon + q_\epsilon^\top \epsilon, \tag{5.15}$$

where $Q_\epsilon \succeq 0$ and $q_\epsilon \geq 0$. This disincentivizes the use of slack variables, i.e. violation of the uncertainty-expanded safety bound, but will allow for a level of risk to be taken when a significant improvement in performance can be achieved. As such, the policy defined by

Table 5.1: Prediction errors [m]

	GP	CV	NL
Lateral	$\mathbf{0.02 \pm 0.077}$	0.064 ± 0.15	-0.057 ± 0.21
Longitudinal	$\mathbf{0.026 \pm 0.077}$	-0.037 ± 0.097	0.11 ± 0.16

(5.3) and (5.6) can trade off between safety and performance in a manner dictated by the weights in (5.3a) and (5.15).

The nonlinear optimal control problem is formulated using CasADi [7] and solved using sequential quadratic approximations with the QP solver `hpipm` [53]. Obtaining a solution takes 30 ms on average on a laptop with a 2.6 GHz 9th-Gen Intel Core i7 CPU.

5.5 Results and Discussion

Our approach is evaluated in simulation and hardware races, which match the EV, using the MPCC policy from Sec. 5.4, against a TV, using the blocking policy defined in Sec. 5.3. Both policies use a horizon length of $N = 10$ and sample time of $T_s = 0.1$ s. In both simulation and hardware races, the EV starts the race behind the TV and is responsible for avoiding collisions. All crashes during an overtaking attempt are counted as the EV's mistake and consequently result in a loss for the EV. Additionally, leaving the track is also counted as a loss for the EV. Hardware experiments were conducted on the 1/10th scale racecar platform, which are each equipped with Raspberry Pi 4 for onboard computation. Measurement of the vehicle state is done using an OptiTrack motion capture system. The EV and TV policies and predictors run on a host machine and communication with the cars is handled using ROS2. Code to reproduce the simulation results has been made available[1].

To show the importance of predicting interactions between the two vehicles, we take an approach similar to [152] and compare the performance of our GP-based approach against a constant velocity (CV) and progress-maximizing nonlinear MPC (NL) predictor in closed-loop with the EV MPCC policy from Sec. 5.4. CV propagates the TV's measured state under a constant linear and angular velocity assumption. NL returns the open-loop solution of the optimal control problem defined by removing the collision avoidance constraint (5.3h) from (5.3). The weights for NL are otherwise chosen to be identical to those used in the TV's blocking policy. It is important to note that NL does *not* include (5.11) in its cost function. As CV and NL do not come with any measure of uncertainty, we add a constant circular safety bound to encourage fair comparison with the uncertainty ellipses of the GP predictions. The size of this safety bound can be interpreted as a hyperparameter controlling the amount of risk the EV is willing to take in its overtaking attempts when using the CV and NL predictors.

[1]`https://github.com/MPC-Berkeley/gp-opponent-prediction-models`

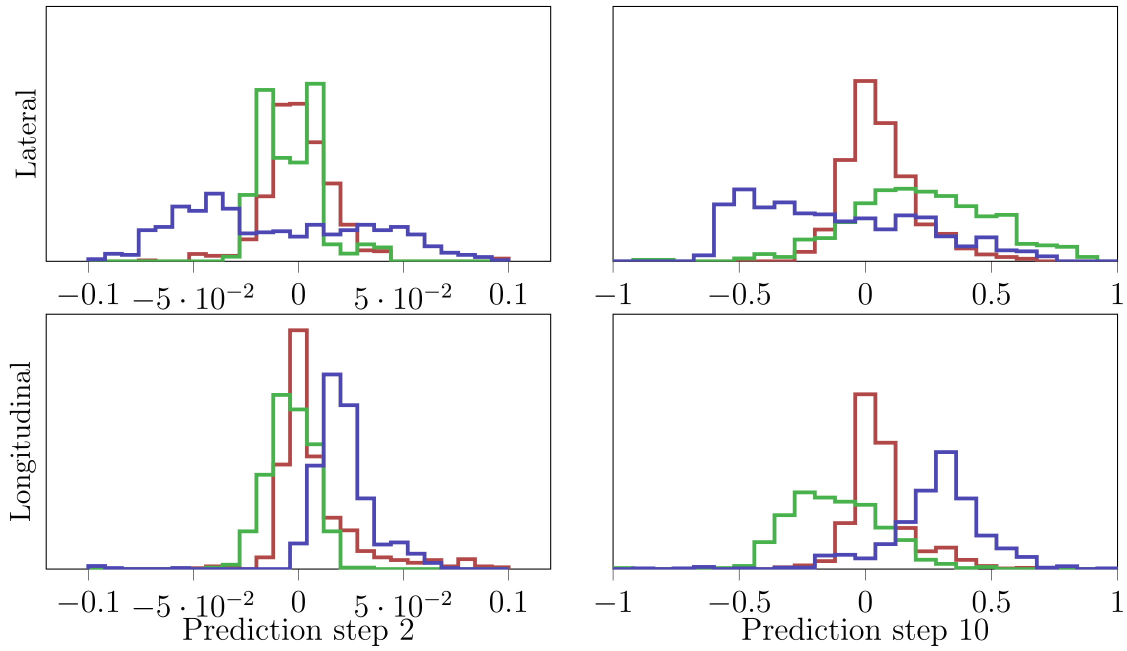


Figure 5.2: Distribution of lateral (top) and longitudinal (bottom) prediction errors [m] at the second (left) and last (right) prediction step for the GP (—), CV (—), and NL (—) predictors.

Simulation Study

In simulation, we perform a Monte Carlo study where 100 starting configurations are sampled and races are run from those initial conditions using each of the aforementioned predictors in close-loop with the EV's policy. For the purposes of the studies below, we vary the value of γ from (5.14) for GP, the radius of the safety bound for CV and NL, and the blocking aggressiveness parameter $q_y^{(2)}$ of the TV policy.

We first examine the longitudinal and lateral errors of the nominal open-loop predictions with respect to the *actual* trajectory driven by the TV. The errors are defined in terms of track progress s and lateral deviation from the centerline e_y respectively. We restrict our error analysis to predictions made during vehicle interactions, i.e. situations where the two vehicles are within two car lengths of each other. As seen in Table 5.1, GP clearly outperforms both CV and NL, particularly when predicting the blocking behavior of the TV as evidenced by the lower lateral error. This is further illustrated in the prediction error distributions in Fig. 5.2, where it can be seen that GP maintains a lower bias and variance in both longitudinal and lateral prediction error even at the end of the prediction horizon.

Next, the closed-loop performance in the presence of TV model mismatch is evaluated. To simulate model mismatch, we vary the blocking aggressiveness $q_y^{(2)}$ of the TV policy while keeping the predictors unchanged, i.e. no retraining or additional tuning is done. As a performance baseline, we include race results where the open-loop solutions of the TV policy are used in lieu of a prediction when computing the control action of the EV. This corresponds to the case where exact knowledge of the TV's future plans are known to the

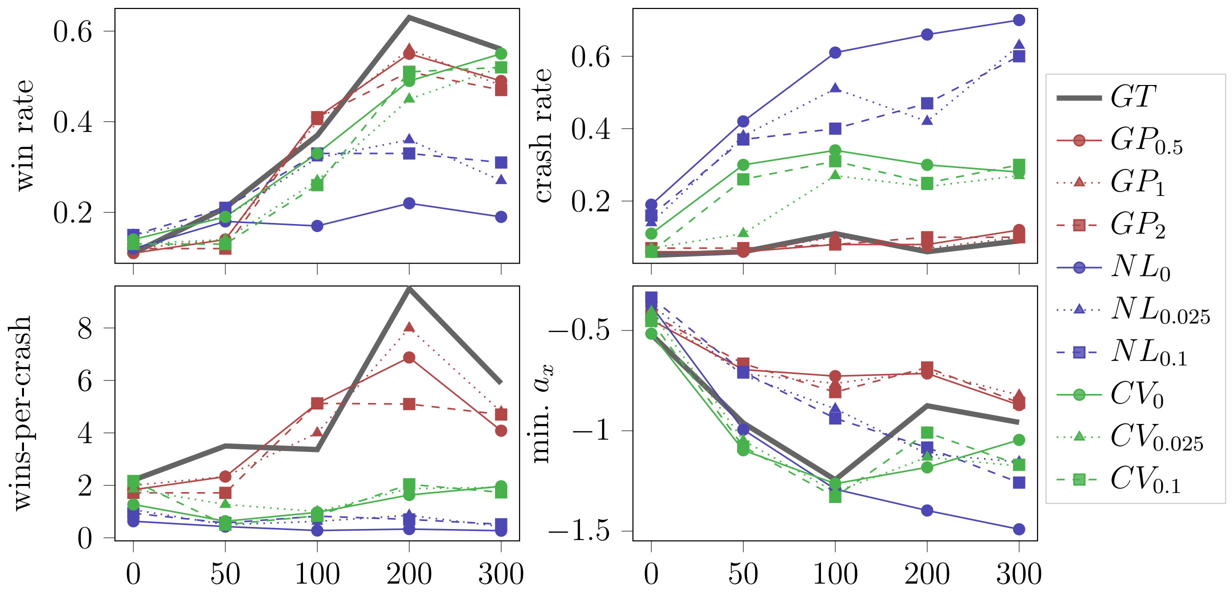


Figure 5.3: Monte Carlo closed-loop performance comparison between the studied prediction models. The x axis corresponds to the values of $q_y^{(2)}$, the weight on the TV blocking cost. Higher values correspond to more aggressive blocking behavior by the TV during a race. The subscripts denote the values used for the safety bound parameters. For GP, it corresponds to the number of standard deviations γ. For CV and NL, it corresponds to the radius of the circular safety bound in meters. The grey line (GT) indicates the best performance achieved by using the ground truth open-loop solutions in lieu of predictions.

EV, we call this the ground truth (GT) case. For each class of predictor, we additionally vary the size of the safety bound to investigate its effect on closed-loop race performance. For the GP, we pick safety bounds with $\gamma = 0.5, 1, 2$. For the CAV and NL predictors, circular safety bounds of radius 0.025, and 0.1 m are added. These are chosen such that the safety bound size on average is similar to the safety bounds induced by the GP's uncertainties in the one and two standard deviation cases. The performance of the EV policy in closed-loop with each of these predictors against the TV blocking policy with the various aggressiveness factor settings are depicted in Fig. 5.3.

In terms of win rate, shown in the upper left, GP clearly outperforms the other predictors for the TV policy that it was trained on, i.e. $q_y^{(2)} = 200$. For the other settings of $q_y^{(2)}$, while GP does not impart a clear advantage in win rate, it remains competitive with the other predictors, including the GT case when we use the TV open-loop solutions directly instead of the output of a predictor. In addition, GP seems to be less sensitive against variations in the size of the safety bound, likely because it provides consistently accurate predictions and does not have to rely on large safety bounds to keep the vehicle safe. In general, all predictors are able to exploit overly aggressive behavior by the target vehicle by overtaking through the space which is made free from aggressive blocking attempts. This is clearly seen in the results as the win rate of the ego vehicle increases with the aggressiveness of the target vehicle.

As for the crash rate, shown in the upper right, GP is able to maintain a crash rate below

10% for all values of $q_y^{(2)}$, whereas the other predictors cannot keep the EV safe when the TV exhibits more aggressive blocking behavior. In fact, it is clear from this plot that in terms of safety, GP is essentially able to achieve identical performance to GT where the TV's future plans are known to the EV. Generally, we observe that GP outperforms the other predictors in terms of safety as it is able to accurately anticipate the TV's blocking attempts and react accordingly by braking or steering out of the way, ultimately reducing the number of crashes.

While the ultimate objective of a race is to win, it is also worthwhile to look at the the outcome of races where the EV loses to the TV. In particular, we are interested in whether the EV was able to finish the race safely despite the loss or if the loss was caused by a crash. The reason for this concerns the practical aspects of car racing where crashes can be catastrophic and expensive. As such, consider two predictors A and B where Predictor A has a win rate of 40% and a crash rate of 5% and Predictor B has a win rate of 50% and a crash rate of 20%. Despite the lower win rate, it would be prudent to prefer Predictor A over Predictor B due to its significantly lower crash rate. In order to quantify this preference, we introduce the metric wins-per-crash, to normalize for the effect of crashing in the win rate and to measure the trade-off between wins and safety. A high score for this metric requires that the predictor not only lead to more wins but also few crashes. The results for this metric are shown in the lower left, where it is clear that GP outperforms the other predictors across all TV policies. We additionally see that GP closely tracks the results from using GT.

We finally examine the largest deceleration (averaged over all races) experienced by the EV under each predictor, which is shown in the plot in the lower right. This acts as a proxy to illustrate how well the predictors anticipate the TV's behavior. If blocking attempts are predicted accurately, we expect the EV to be able to react in such a way that minimizes acceleration, which is encouraged by the input cost in (5.3a). On the other hand, if blocking maneuvers by the TV are not predicted, we expect the EV to be more reactionary to the current state of the TV, which would lead to larger throttle and braking commands. From the plot, it can be clearly seen that under NL and CV, the EV generally experiences larger decelerations, with more extreme values being observed for more aggressive TV behavior. On the other hand, GP is less sensitive to the target vehicle's aggressiveness and experiences less extreme deceleration indicating that it is able to anticipate the incoming blocking attempts by the TV and can begin to react earlier. We note that in this aspect, GP actually outperforms GT. We believe that this is due to the fact that GP was trained on closed-loop TV behavior and is predicting the *actual* trajectory which will be driven by the TV. On the other hand, the open-loop predictions from the TV policies, which are used in GT, can change drastically between consecutive time steps due to local optimality of NLPs. This can lead to significant mismatch between the open-loop predictions and the *actual* driven trajectory, which ultimately results in larger braking commands to react to those mismatches.

Hardware Study

In our hardware experiments, we compare the performance of the EV MPCC policy in closed-loop with the NL and GP predictors in ten races on the L-shaped track shown in Fig. 5.4.

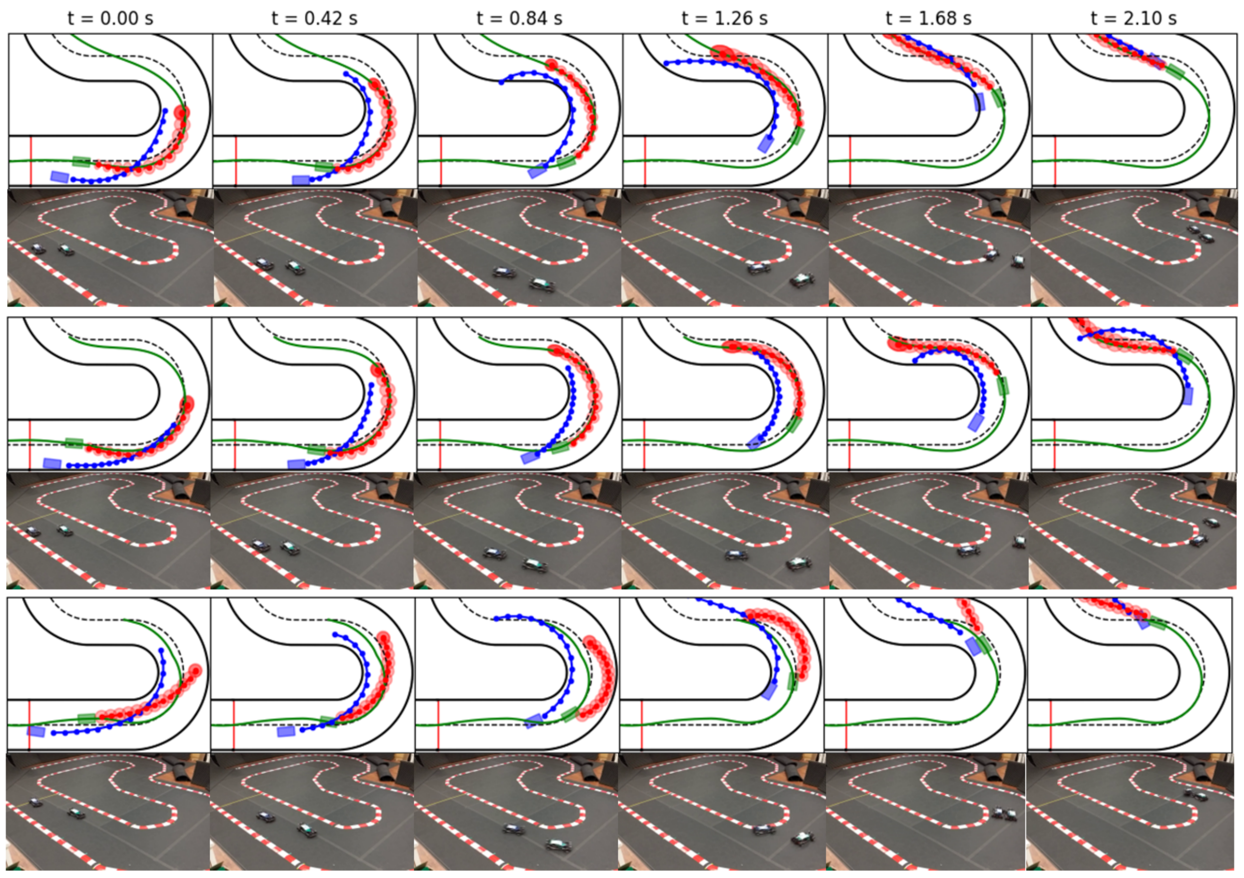


Figure 5.4: EV (blue) overtaking attempts against the TV (green) with predictions in red at six time instances during the hardware experiment. The first two rows show successful and failed attempts with GP and the third row shows a failed attempt with NL. The red ellipses and circles correspond to the uncertainty-expanded or artificial safety bounds around the nominal prediction at each step in the MPC horizon. The solid green line indicates the closed-loop trajectory of TV. Video recordings of all hardware experiments can be found at `https://youtu.be/KMSs4ofDfIs`.

We chose $\gamma = 1$ for GP and a radius of 0.14 m for NL such that the safety bound for NL is similar in size to those produced by GP. Using each of the two predictors, we run ten races with the EV starting at the same position. The races end after the EV has driven three laps or after a major collision occurs. As for the TV, we chose different starting positions ahead of the EV for each race. These starting positions are kept constant in the corresponding race where GP and NL predictions are used. For all experiments, we impose a longitudinal speed constraint of 2.8 and 2.0 m/s for the EV and TV respectively. This is done to ensure that close interactions will occur between the vehicles during each race. As in the simulation study, the TV uses the blocking policy described in Section 5.3 whereas the EV is encouraged to overtake the TV through a higher weight on the progress maximization cost in (5.3a). We note that both control policies use a horizon length of $N = 15$ and no additional controller tuning was performed when switching between the GP and NL predictors. The outcomes of the ten races with the GP and NL predictors is summarized in Table 5.2, where the EV was

Table 5.2: Race outcomes

	GP	NL
Overtake	4	3
Safe (no overtake)	5	0
Collision (minor)	1	2
Collision (major)	0	5

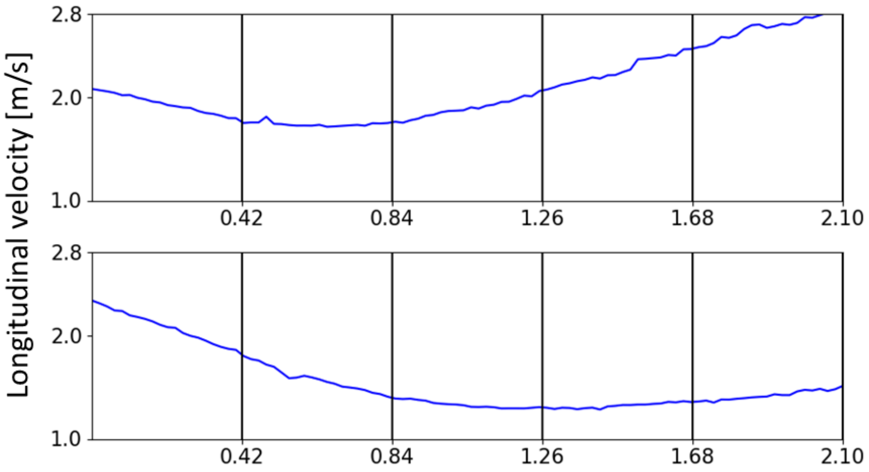


Figure 5.5: Longitudinal velocity profiles for the successful (top) and failed (bottom) overtaking attempts using GP. The black vertical lines correspond to the time instances of the frames in Fig. 5.4.

able to perform safe overtaking maneuvers (all with longitudinal velocity greater than 2.7 m/s) and win in four races with GP and in three races with NL. For the remaining races, the EV lost but was able to remain safe in five races with GP and saw only one minor collision where the wheels of the two vehicles touched. On the other hand, the TV saw collisions in all seven remaining races. Out of these collisions, two were minor wheel touches, but the other five were major collisions which significantly altered the trajectory of both cars. We note that these outcomes are in close agreement with the results from our simulation study.

To further understand the advantages that GP affords us when compared to NL, we turn to a similar interaction which arose in multiple instances with both predictors, which is illustrated in Fig. 5.4. The scenario begins with EV on the outside of TV and the two cars approaching the first 180turn. The top frames show the case when GP is used where a successful inside overtake is performed. In this interaction, it can be seen that the TV's closed-loop blocking maneuver is contained in the uncertainty-expanded safety bounds. By accurately predicting the behavior of the TV and recognizing that it cannot successfully block the EV, this allows the EV to exploit the free space on the inside of the TV to perform a safe overtake. The middle frames show the case when GP is used, but the EV was not able to successfully overtake the TV. However, it is able to remain safe and avoid collisions. In this scenario, the EV begins by slowing down in anticipation of the TV's outside blocking maneuver as seen in Fig. 5.5. This lower speed then prevents the EV from finding a safe solution for performing an inside overtake as it accurately predicts that the TV will be able to block it even as it takes the inside line. The EV therefore slows down and remains behind the TV. Finally, the bottom frames show the case when NL is used and the EV collides with the TV during its overtaking attempt. This is primarily due to the inaccurate predictions from NL, which do not take into account the blocking incentive of the TV, which, as seen in (5.11), dominates the TV's MPC cost especially when the two cars are in close proximity with each other. The added safety bounds are of no use here either due to the significant divergence of the NL predictions from the closed-loop trajectory of the TV.

5.6 Chapter Summary

In this chapter, we presented a learning-based method for predicting opponent behavior in the context of head-to-head competitive autonomous racing. In particular, Gaussian processes were trained on data from past races to generate trajectory predictions and their corresponding uncertainties, which are then used in closed-loop with an MPCC policy. We show through both simulation and hardware experiments that our approach outperforms non-data-driven predictors in terms of safety while maintaining a high win rate.

It should be noted that the method proposed in this chapter follows a somewhat standard sequential architecture where the prediction stage is upstream of the planning stage and therefore the predictions are treated as constant when solving the planning sub-problem. While this produces a conceptually simple framework, we note that the GP model can only make predictions of interactive behavior to the extent that it exists in the dataset used for training. Furthermore, due to the static nature of the opponent predictions during the planning stage, the opponent is unable to "react" to the actions chosen by the racing policy which can limit the interactivity of the solutions. In the following chapters, we will examine a solution approach to the prediction and planning problem which is able to explicitly reason over interactions between agents by leveraging ideas from game theory.

Chapter 6

On the Solution of Game-Theoretic Equilibria for Autonomous Racing

6.1 Introduction

Autonomous racing has steadily gained attention as an area of research due to the unique combination of challenges that it presents [17]. Like many widely studied autonomous navigation problems, racing scenarios typically involve multiple agents with no information sharing who are subject to coupling through safety related constraints. However, what sets racing apart is the aggressive and direct competition which necessitates performance at the limit (of speed, tire friction, thrust, etc.) in order to achieve victory against one's opponents. As such, we are faced with the challenge in prediction and planning for autonomous racing where, we need to solve for interactive multi-agent behavior arising from direct competition under shared constraints, while subject to performance limiting factors such as friction limits from non-linear tire models.

Approaches to tackling these challenges fall broadly into data-driven and model-based approaches, which have both seen impressive successes. In [149], autonomous agents were trained using deep reinforcement learning (DRL) in a video game environment and achieved superior performance in head-to-head races against human players. While this approach was certainly able to learn complex vehicle dynamics and highly expressive multi-agent interaction models, it required days of training and, as with most RL-based methods, is dependent on large quantities of data to reach super-human performance in a simulation environment. In [69], DRL was used to achieve super-human performance in real-world drone racing. However, due to the sparsity of close-proximity interactions in drone races, no interaction models were used in the solution. On the other hand, [18, 67] used model-based approaches in a predict-then-plan architecture to achieve real-world racing of full scale race cars at speeds of up to 270 km/h. In these works, an upstream prediction module provides behavior forecasts of the opponents, which are then treated as constant during the downstream planning phase when an action plan is formulated for the ego vehicle (EV) via optimal control techniques which leverage explicit nonlinear vehicle models. These approaches have the benefit of being highly transparent, which is crucial for deployment on expensive and safety critical hardware, but can be limited in expressiveness in terms of interactivity between the opponent predictions and ego plan. Our work seeks to address this limitation for model-based approaches by producing highly interactive solutions to the prediction and planning problem for autonomous racing while maintaining transparency through the use of explicit vehicle models.

Recent work has posed the prediction and planning problem as an equilibrium finding problem for a dynamic game [160, 129, 123]. These approaches search for equilibria over the joint trajectory space of the EV and its opponents, which allows for simultaneous prediction and planning, as they are not conditioned on a static prior EV plan and can in fact solve over the space of multi-agent interactions. Such methods are based on the theory of non-cooperative dynamic games [14] and have been applied to a wide variety of trajectory optimization tasks for multi-agent robotic systems [75, 78, 52, 90, 104, 70]. Two solution or equilibrium concepts are common for dynamic games, namely the Stackelberg and Nash equilibria, which make different assumptions about the information structure of the game. A Stackelberg equilibrium can be found for a game with an explicit leader-follower hierarchy [50], whereas a Nash equilibrium models the case when agents make their decisions simultaneously. Our work focuses on the selection of Nash equilibria, which we believe to be a good fit for modeling the behavior of ego-centric agents in racing scenarios where no *a priori* structure is imposed on the order of the interactions and where agents can have zero-sum terms in their objective functions, which describe direct competition with their opponents.

Motivated by the considerations above, this chapter focuses on finding generalized Nash equilibria (GNE) [45] for open-loop dynamic games with state and input constraints with a numerically robust solver. We present three main contributions.

- The first is the DG-SQP solver, which is a novel iterative approach for finding GNE of a discrete-time open-loop dynamic game based on sequential quadratic programming (SQP). In particular, the method is able to account for nonlinear game dynamics and constraints on both the game state and agent actions. The three key elements of the method are a non-monotonic line search for solving the associated KKT equations, a merit function to handle zero sum costs, and a decaying regularization scheme for the SQP step selection.

- The second contribution is a novel application of model predictive contouring control in the context of dynamic games which is used to approximate Frenet-frame kinematics for improved numerical robustness.

- Finally, we perform an extensive simulation study comparing the performance of our solver to the to the state-of-the-art PATH solver based on mixed complementary problems [38]. We show comparable performance to the PATH solver, in terms of success rate, on dynamic games formulated using exact Frenet-frame kinematics and significant improvements when the approximate formulation is used.

The results presented in this chapter have also appeared in:

- E. Zhu and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 3211-3217.

6.2 Related Work

Numerical Solvers for Nash Equilibria of Dynamic Games

For the solution of GNE for dynamic games, many numerical approaches have been proposed. [52] takes a differential dynamic programming approach to obtain a linear quadratic approximation of the dynamic game and its associated feedback Nash equilibrium. This is built upon in [123] to address stochastic games over belief spaces, which takes into account the effect of noisy measurements on state estimation in the game dynamics. However, these approaches are unable to explicitly account for inequality constraints and instead must include them in the cost function via barrier functions. This can obfuscate the meaning of the game as the cost measures both performance and constraint violation. [70, 21] propose an approach for the subclass of potential dynamic games which was shown to be computationally efficient. However, by requiring that the dynamic game be described by a single potential function over all agents, we note that direct competition, in the form of zero-sum terms in the agents' objective functions, cannot be captured in this approach. [129, 145, 143, 144] formulate a method akin to block coordinate descent called Sensitivity Enhanced Iterative Best Response (SE-IBR) where the optimal control problems for each agent, which are coupled in the dynamic game setting, are decoupled but augmented with a term in the objective function which captures the sensitivity of the TV's solution w.r.t. the EV's solution. Agents then improve their solutions in a sequential manner while holding the behavior of all other agents fixed. It was shown that fixed points of this algorithm correspond to GNE. However, each iteration of the method requires the solution of the same number of optimization problems as there are agents and can be slow to converge in practice. In contrast, our proposed solver only requires the solution of a single convex optimization problem at each iteration. [78] proposes a solver based on the augmented Lagrangian method. The solver, called ALGAMES, shows good performance when compared to [52]. However, we show in prior work that ALGAMES appears to struggle with convergence in the context of car racing where more complex dynamics and environments are introduced [160]. Our proposed solver is perhaps most similar to [76] which also leverages SQP for the computation of *feedback* Nash equilibria, but does not investigate its local behavior. Compared to [76], we also introduce an approximation scheme which improves solver convergence in racing scenarios. Finally, we note that the problem of finding GNE for open-loop dynamic games can be formulated as a mixed complementarity problem for which the PATH solver is the state-of-the-art [38]. This solver was used to great effect in multiple recent works to compute GNE for multi-agent navigation tasks [90, 104]. As such, the PATH solver will be the main target of our comparisons, where we show, through a numerical study, that our proposed solver achieves greater success rate in the context of head-to-head racing scenarios.

Game-Theoretic Methods in Autonomous Racing

Due to its competitive nature, autonomous racing has been a popular test bed for game-theoretic methods of prediction and planning and many of the previously mentioned methods have been applied in this context. [123] formulates the racing problem as a stochastic dynamic game which can take into account state and measurement uncertainty when solving for a Nash equilibrium. However, this approach is unable to explicitly incorporate safety-critical track boundary and collision avoidance constraints, which are crucial in racing. [65] poses the problem as a constrained potential game. This results in good computational efficiency for game-theoretic prediction and planning in a real-time, model predictive game play (MPGP) manner, but importantly precludes zero-sum components in the agents' cost functions by definition of the potential game. This makes it difficult to capture direct competition between the agents and instead the approach relies on heuristics and mode switching in order to induce competitive behavior. [129, 144, 145, 143] leverage the aforementioned SE-IBR algorithm to approximately solve a dynamic game in a MPGP manner and use the approximate GNEs as high-level plans for a tracking policy in hardware races. However, this work, like all of the ones discussed so far in this section, have only shown GNE solution results for linear models such as the integrator model or simple nonlinear models such as the kinematic bicycle model. The only exception to this is in [88] where a bimatrix game was defined over the incurred costs of sampled trajectory rollouts for two vehicles each using the dynamic bicycle model. The trajectory pair corresponding to the NE of this bimatrix game was then chosen as the opponent prediction and EV plan. This approach can be considered as a zeroth order method for the solution of GNEs as it only requires evaluations of the components of the dynamic game. (This is in contrast to [78, 38] and our approach, which leverage gradients of the game components). As such, it provides a simple framework for leveraging high fidelity vehicle models. However, it is straightforward to see that the effectiveness of this approach largely depends on the number of trajectory samples taken and it is unclear if the trajectory pair corresponding to the NE of the bimatrix game is a GNE of the dynamic game.

6.3 Problem Formulation

Consider an M-agent, finite-horizon, discrete-time, general-sum, open-loop, dynamic game whose state is characterized by the joint dynamical system:

$$x_{k+1} = f(x_k, u_k), \tag{6.1}$$

where $x_k^i \in \mathcal{X}^i$ and $u_k^i \in \mathcal{U}^i$ are the state and input of agent i at time step k and

$$x_k = \left[x_k^{1\top}, \dots, x_k^{M\top}\right]^\top \in \mathcal{X}^1 \times \dots \times \mathcal{X}^M = \mathcal{X} \subseteq \mathbb{R}^n$$
$$u_k = \left[u_k^{1\top}, \dots, u_k^{M\top}\right]^\top \in \mathcal{U}^1 \times \dots \times \mathcal{U}^M = \mathcal{U} \subseteq \mathbb{R}^m,$$

are the concatenated states and inputs of all agents. In this work, we will use the notation $x_k^{\neg i}$ and $u_k^{\neg i}$ to denote the collection of states and inputs for all but the i-th agent.

Each agent i attempts to minimize its own cost function, which is comprised of stage costs l_k^i and terminal cost l_N^i, over a horizon of length N:

$$\bar{J}^i(\mathbf{x}, \mathbf{u}^i) = \sum_{k=0}^{N-1} l_k^i(x_k, u_k^i) + l_N^i(x_N) \tag{6.2a}$$
$$= J^i(\mathbf{u}^1, \dots, \mathbf{u}^M, x_0), \tag{6.2b}$$

where $\mathbf{x} = \{x_0, \dots, x_N\}$ and $\mathbf{u}^i = \{u_0^i, \dots, u_{N-1}^i\}$ denote state and input sequences over the horizon. Note that the cost in (6.2a) for agent i depends on its *own* inputs and the *joint* state, which can capture dependence on the behavior of the other agents. We arrive at (6.2b) by recursively substituting in the dynamics (6.1) to the cost function, which are naturally a function of the open-loop input sequences for all agents. The agents are additionally subject to n_c constraints

$$C(\mathbf{u}^1, \dots, \mathbf{u}^M, x_0) \leq 0, \tag{6.3}$$

which can be used to describe individual constraints as well as coupling between agents and where we have once again made the dependence on the joint dynamics implicit. For the sake of brevity, when focusing on agent i, we omit the initial state x_0 and write the cost and constraint functions as $J^i(\mathbf{u}^i, \mathbf{u}^{\neg i})$ and $C(\mathbf{u}^i, \mathbf{u}^{\neg i})$. Let us now define the conditional constraint set

$$\mathcal{U}^i(\mathbf{u}^{\neg i}) = \{\mathbf{u}^i \mid C(\mathbf{u}^i, \mathbf{u}^{\neg i}) \leq 0\},$$

which can be interpreted as a restriction of the joint constraint set for agent i given some $\mathbf{u}^{\neg i}$. We make the following assumption about the functions and constraint sets.

Assumption 3. The sets $\mathcal{X}^i$ and $\mathcal{U}^i$ are compact and the functions f, J^i, and C are three times continuously differentiable on $\mathcal{X}$ and $\mathcal{U}$ for all $i \in \{1, \dots, M\}$.

Generalized Nash Equilibrium

We define the constrained dynamic game as the tuple:

$$\Gamma = (N, \mathcal{X}, \mathcal{U}, f, \{J^i\}_{i=1}^M, C). \tag{6.4}$$

For such a game, a GNE [45] is attained at the set of feasible input sequences $\mathbf{u} = \{\mathbf{u}^i\}_{i=1}^M$ which minimize (6.2) for all agents i. Formally, we define this solution concept as follows:

Definition 1. A generalized Nash equilibrium (GNE) for the dynamic game Γ is the set of open-loop solutions $\mathbf{u}^\star = \{\mathbf{u}^{i,\star}\}_{i=1}^M$ such that for each agent i:

$$J^i(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star}) \leq J^i(\mathbf{u}^i, \mathbf{u}^{\neg i,\star}),\ \forall \mathbf{u}^i \in \mathcal{U}^i(\mathbf{u}^{\neg i,\star}).$$

If the condition holds only in some local neighborhood of $\mathbf{u}^{i,\star}$, then $\mathbf{u}^\star$ is denoted as a local GNE.

In other words, at a local GNE, agents cannot improve their cost by unilaterally perturbing their open-loop solution in a locally feasible direction. Furthermore, it was shown in [76, Theorem 2.2] that the local GNE for agent i can be obtained equivalently by solving the following constrained finite horizon optimal control problems (FHOCP):

$$\begin{aligned} \mathbf{u}^{i,\star}(\mathbf{u}^{\neg i,\star}) = \arg\min_{\mathbf{u}^i} \quad & J^i(\mathbf{u}^i, \mathbf{u}^{\neg i,\star}) \\ \text{subject to} \quad & C(\mathbf{u}^i, \mathbf{u}^{\neg i,\star}) \leq 0. \end{aligned} \tag{6.5}$$

where $\mathbf{u}^{\neg i,\star}$ correspond to local GNE solutions for the other agents. Note that we are assuming uniqueness of the local GNE of (6.4). This assumption will be made formal in the next section. A distinct advantage of using (6.5) to model agent interactions is that a dynamic game allows for a direct representation of agents with competing objectives as the M objectives are considered separately instead of being summed together, which is typical in cooperative multi-agent approaches [162].

6.4 An SQP Approach to Dynamic Games

In this section, we propose a method which iteratively solves for open-loop local GNE of dynamic games using sequential quadratic approximations. In particular, we will derive the algorithm and present guarantees on local convergence, which is based on established SQP theory [23]. We begin by defining the Lagrangian functions for the M coupled FHOCPs in (6.5):

$$\mathcal{L}^i(\mathbf{u}^i, \mathbf{u}^{\neg i,\star}, \lambda^i) = J^i(\mathbf{u}^i, \mathbf{u}^{\neg i,\star}) + C(\mathbf{u}^i, \mathbf{u}^{\neg i,\star})^\top \lambda^i,$$

where we have again omitted the dependence on the initial state x_0 for brevity. As in [78], we require that the Lagrange multipliers $\lambda^i \geq 0$ are equal over all agents, i.e $\lambda^i = \lambda^j = \lambda$, $\forall i, j \in \{1, \ldots, M\}$. Since the multipliers reflect the sensitivity of the optimal cost w.r.t. constraint violation, this can be interpreted as a requirement for parity in terms of constraint violation for all agents. Under this condition, the GNE from (6.5) are also known as normalized Nash equilibria [110].

A direct consequence of writing the constrained dynamic game in the coupled nonlinear optimization form of (6.5) is that, subject to regularity conditions, solutions of (6.5) must satisfy the KKT conditions below:

$$\begin{aligned}
\nabla_{\mathbf{u}^i}\mathcal{L}^i(\mathbf{u}^{i,\star},\mathbf{u}^{\neg i,\star},\lambda^\star) &= 0,\ \forall i=1,\dots,M, && \text{(6.6a)}\\
C(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star}) &\le 0, && \text{(6.6b)}\\
C(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star})^\top\lambda^\star &= 0, && \text{(6.6c)}\\
\lambda^\star &\ge 0. && \text{(6.6d)}
\end{aligned}$$

We therefore propose to find a local GNE as a solution to the KKT system (6.6) in an iterative fashion starting from an initial guess for the primal and dual solution, which we denote as $\mathbf{u}_0^i$ and $\lambda_0 \ge 0$ respectively, and taking steps p_q^i and p_q^λ, at iteration q, to obtain the sequence of iterates:

$$\mathbf{u}_{q+1}^i = \mathbf{u}_q^i + p_q^i,\ \ \lambda_{q+1} = \lambda_q + p_q^\lambda. \tag{6.7}$$

In particular, we form a quadratic approximation of (6.6a) and linearize the constraints in (6.6b) about the primal and dual solution at iteration q in a SQP manner [148] as follows:

$$\begin{aligned}
L_q &= \begin{bmatrix} \nabla^2_{\mathbf{u}^1}\mathcal{L}_q^1 & \nabla_{\mathbf{u}^2,\mathbf{u}^1}\mathcal{L}_q^1 & \dots & \nabla_{\mathbf{u}^M,\mathbf{u}^1}\mathcal{L}_q^1 \\ \nabla_{\mathbf{u}^1,\mathbf{u}^2}\mathcal{L}_q^2 & \nabla^2_{\mathbf{u}^2}\mathcal{L}_q^2 & \dots & \nabla_{\mathbf{u}^M,\mathbf{u}^2}\mathcal{L}_q^2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{\mathbf{u}^1,\mathbf{u}^M}\mathcal{L}_q^M & \nabla_{\mathbf{u}^2,\mathbf{u}^M}\mathcal{L}_q^M & \dots & \nabla^2_{\mathbf{u}^M}\mathcal{L}_q^M \end{bmatrix},\\
h_q &= \begin{bmatrix} \nabla_{\mathbf{u}^1}J_q^1 & \nabla_{\mathbf{u}^2}J_q^2 & \dots & \nabla_{\mathbf{u}^M}J_q^M \end{bmatrix}^\top,\\
G_q &= \begin{bmatrix} \nabla_{\mathbf{u}^1}C_q & \nabla_{\mathbf{u}^2}C_q & \dots & \nabla_{\mathbf{u}^M}C_q \end{bmatrix},\\
B_q &= \mathrm{proj}_{\succeq 0}((L_q + L_q^\top)/2) + \epsilon I,
\end{aligned} \tag{6.8}$$

where the subscript q indicates that the corresponding quantity is evaluated at the primal and dual iterate $\mathbf{u}_q$ and λ_q. Here, $\epsilon \ge 0$ is a regularization coefficient, I is the identity matrix of appropriate size, and

$$\mathrm{proj}_{\succeq 0}(X) = \arg\min_{Y \succeq 0} \|X - Y\|_F^2 \tag{6.9}$$

denotes the operation which projects the symmetric matrix $X \in \mathbb{S}^n$ onto the positive semidefinite cone, where $\|\cdot\|_F$ denotes the Frobenius norm. We note that the semidefinite program in (6.9) admits the closed-form solution $\mathrm{proj}_{\succeq 0}(X) = \sum_{i=1}^n \max\{0, s_i\} v_i v_i^\top$, where s_i and v_i denote the i-th eigenvalue and eigenvector of X respectively. Using the approximation in (6.8), we solve for the step in the primal variables via the following convex quadratic program (QP):

$$\begin{aligned}
p_q^{\mathbf{u}} = \arg\min_{p^1,\dots,p^M}\ & \frac{1}{2} p^{\mathbf{u}\top} B_q p^{\mathbf{u}} + h_q^\top p^{\mathbf{u}} && \text{(6.10a)}\\
\text{subject to}\ \ & C_q + G_q p^{\mathbf{u}} \le 0. && \text{(6.10b)}
\end{aligned}$$

where we denote $p^{\mathbf{u}} = [p^{1\top}, \ldots, p^{M\top}]^\top$. Denote the Lagrange multipliers corresponding to the solution of (6.10) as d_q. We then define the step in the dual variables as

$$p_q^\lambda = d_q - \lambda_q, \tag{6.11}$$

which maintains nonnegativity of the dual iterates given $\lambda_0 \geq 0$. Finally, we note that in contrast to the approach used in [129], which requires the solution of M optimization problems, our SQP procedure requires the solution of only a single QP at each iteration.

Local Behavior of Dynamic Game SQP

We make the following assumptions about the primal and dual solutions of (6.5):

Assumption 4. Solutions $\{\mathbf{u}^{i,\star}\}_{i=1}^M$ and $\lambda^\star$ of (6.5) satisfy the following, for each $i \in \{1, \ldots, M\}$:

- $\lambda^\star \perp C(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star})$ and $\lambda_j^\star = 0 \iff C_j(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star}) < 0$ for $j = 1, \ldots, n_c$.
- The rows of the active constraint Jacobians at the local GNE, i.e. $\nabla_{\mathbf{u}^i} \bar{C}(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star})$, are linearly independent.
- $d^\top \nabla^2_{\mathbf{u}^i} \mathcal{L}^i(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star}, \lambda^\star) d > 0$, $\forall d \neq 0$ such that $\nabla_{\mathbf{u}^i} \bar{C}(\mathbf{u}^{i,\star}, \mathbf{u}^{\neg i,\star})^\top d = 0$.

The first assumption is strict complementary slackness, the second is the linear independence constraint qualification (LICQ), and the third states that the Hessian of the Lagrangian function is positive definite on the null space of the active constraint Jacobians at the solution. It is straight forward to see that By standard optimization theory, (6.6) and Assumption 4 together constitute necessary and sufficient conditions for a primal and dual solution of (6.5) for agent i to be locally optimal and unique given $\mathbf{u}^{\neg i,\star}$. When these conditions hold for the solutions over all agents, it was shown in [76] that satisfaction of the requirements for a unique local GNE follow immediately. Note that, as in [23] and [76], Assumption 4 is standard and can be verified a posteriori.

To analyze the local behavior of the iterative procedure as defined by (6.7), (6.8), and (6.10), let us assume that $\mathbf{u}_0$ and λ_0 are close to the optimal solution and the subset of active constraints at the local GNE, which we denote as $\bar{C}$, with Jacobian $\bar{G}$, is known and constant at each iteration q. Therefore, for the purposes of this section, we can replace the inequality constraint in (6.10b) with the equality constraint:

$$\bar{C}_q + \bar{G}_q p^{\mathbf{u}} = 0. \tag{6.12}$$

We refer to the QP constructed from (6.10a) and (6.12) as EQP.

In the traditional derivation of the SQP procedure [23, 148], it was shown that under the aforementioned assumptions, the SQP step computed is identical to a Newton step for the corresponding KKT system. The SQP step therefore inherits the quadratic convergence rate of Newton's method in a local neighborhood of the optimal solution [23, Theorem 3.1]. However, in the case of dynamic games, the equivalence between the SQP procedure and Newton's method is no longer exact since the matrix L_q is not symmetric in general. To see this, let us first state the joint KKT system for the equality constrained version of (6.5):

$$F(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star},\lambda^\star) = \begin{bmatrix} \nabla_{\mathbf{u}^1}\mathcal{L}^1(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star},\lambda^\star) \\ \vdots \\ \nabla_{\mathbf{u}^M}\mathcal{L}^M(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star},\lambda^\star) \\ \bar{C}(\mathbf{u}^{1,\star},\dots,\mathbf{u}^{M,\star}) \end{bmatrix} = 0, \tag{6.13}$$

For the system of equations (6.13), the Newton step at iteration q is the solution of the linear system:

$$\begin{bmatrix} L_q & \bar{G}_q^\top \\ \bar{G}_q & 0 \end{bmatrix} \begin{bmatrix} \bar{p}_q^{\mathbf{u}} \\ \bar{p}_q^{\lambda} \end{bmatrix} = - \begin{bmatrix} h_q + \bar{G}_q^\top \lambda_q \\ \bar{C}_q \end{bmatrix}. \tag{6.14}$$

On the other hand, by the first order optimality conditions for EQP, we have that the SQP step must satisfy

$$\begin{bmatrix} B_q & \bar{G}_q^\top \\ \bar{G}_q & 0 \end{bmatrix} \begin{bmatrix} p_q^{\mathbf{u}} \\ p_q^{\lambda} \end{bmatrix} = - \begin{bmatrix} h_q + \bar{G}_q^\top \lambda_q \\ \bar{C}_q \end{bmatrix}. \tag{6.15}$$

When the matrix L_q is positive definite and $\epsilon = 0$, (6.15) and (6.14) are equivalent. This corresponds to the special case of potential games [163]. However, this is not true in general for our SQP step, which implies that we cannot inherit the quadratic convergence of Newton's method. Instead, the SQP step from (6.10a) and (6.12) can be seen as a symmetric approximation to the Newton step. As such, we establish guaranteed local linear convergence for our SQP procedure via established theory for SQP with approximate Hessians. Before proving the main result of this section, let us first define the bounded deterioration property, which essentially requires that the distance between a matrix and its approximations are bounded.

Definition 2. A sequence of matrix approximations $\{B_q\}$ to $L^\star$ for the SQP method is said to have the property of *bounded deterioration* if there exist constants α_1 and α_2 independent of q such that:

$$\|B_{q+1} - L^\star\| \leq (1 + \alpha_1\sigma_q)\|B_q - L^\star\| + \alpha_2\sigma_q,$$

where $\sigma_q = \max(\|\mathbf{u}_{q+1} - \mathbf{u}^\star\|, \|\mathbf{u}_q - \mathbf{u}^\star\|, \|\lambda_{q+1} - \lambda^\star\|, \|\lambda_q - \lambda^\star\|)$, and $L^\star$ denotes the matrix L in (6.8) evaluated at the solution $\mathbf{u}^\star$, $\lambda^\star$.

Theorem 2. *Consider the dynamic game defined by* (6.4)*. Let Assumptions 3 and 4 hold. Then there exist positive constants ϵ_1 and ϵ_2 such that if*

$$\|\mathbf{u}_0 - \mathbf{u}^\star\| \leq \epsilon_1, \ \|B_0 - L^\star\| \leq \epsilon_2,$$

and $\lambda_0 = -(\bar{G}_0\bar{G}_0^\top)^{-1}\bar{G}_0 h_0$*, then the sequence* $(\mathbf{u}_q, \lambda_q)$ *generated by the SQP procedure* (6.7) *and* (6.10) *converges linearly to* $(\mathbf{u}^\star, \lambda^\star)$*.*

Proof. We obtain the result by showing that the conditions of [23, Theorem 3.3] are satisfied. The first condition requires that the approximations B_q are positive definite on the null space of $\bar{G}_q$. Since B_q is constructed by projecting L_q into the positive definite cone, this condition is satisfied trivially.

The second condition requires that the sequence $\{B_q\}$ satisfies the bounded deterioration property from Definition 2. To show this, we begin with the following derivation:

$$\begin{aligned}\|B_{q+1} - L^\star\| &= \|B_{q+1} - B_q + B_q - L^\star\| \\ &\leq \|B_q - L^\star\| + \|B_{q+1} - B_q\|.\end{aligned}$$

From the above, it can be seen that $\{B_q\}$ satisfies the bounded deterioration property with $\alpha_1 = 0$ if there exists α_2 such that $\|B_{q+1} - B_q\| \leq \alpha_2\sigma_q$. We will show this via the mean value theorem, where we first observe that by Assumption 3, $L(\mathbf{u}_q, \lambda_q)$ is continuous and differentiable in $\mathbf{u}_q$ and λ_q. The smoothness of $B(\mathbf{u}_q, \lambda_q)$ is therefore dependent on the smoothness of the projection operator in (6.9). It was established in [2] that for parametric strongly convex conic optimization problems, there exists a unique, continuous, and differentiable mapping from the parameters to the optimal solution of the problem. The projection operator (6.9) is a semi-definite program which admits a unique solution [29]. Therefore, we may conclude that $B(\mathbf{u}_q, \lambda_q)$ is continuous and differentiable in $\mathbf{u}_q$ and λ_q. This allows us to establish the following inequalities:

$$\begin{aligned}\|B_{q+1} - B_q\| &= \|B_{q+1} - B^\star + B^\star - B_q\| \\ &\leq \|B_{q+1} - B^\star\| + \|B_q - B^\star\| \\ &= \|B_{q+1} - B(\mathbf{u}^\star, \lambda_{q+1}) + B(\mathbf{u}^\star, \lambda_{q+1}) - B^\star\| \\ &\quad + \|B_q - B(\mathbf{u}^\star, \lambda_q) + B(\mathbf{u}^\star, \lambda_q) - B^\star\| \\ &\leq \|B_{q+1} - B(\mathbf{u}^\star, \lambda_{q+1})\| + \|B(\mathbf{u}^\star, \lambda_{q+1}) - B^\star\| \\ &\quad + \|B_q - B(\mathbf{u}^\star, \lambda_q)\| + \|B(\mathbf{u}^\star, \lambda_q) - B^\star\|.\end{aligned}$$

Since B is continuous and differentiable, by the mean value theorem:

$$\begin{aligned}&\|B_{q+1} - B(\mathbf{u}^\star, \lambda_{q+1})\| + \|B(\mathbf{u}^\star, \lambda_{q+1}) - B^\star\| + \|B_q - B(\mathbf{u}^\star, \lambda_q)\| + \|B(\mathbf{u}^\star, \lambda_q) - B^\star\| \\ \leq\ &\beta_1\|\mathbf{u}_{q+1} - \mathbf{u}^\star\| + \beta_2\|\lambda_{q+1} - \lambda^\star\| + \beta_3\|\mathbf{u}_q - \mathbf{u}^\star\| + \beta_4\|\lambda_q - \lambda^\star\| \\ \leq\ &\max(\beta_1, \beta_2, \beta_3, \beta_4)\sigma_q,\end{aligned}$$

where

$$\begin{aligned}
\beta_1 &= \sup_{t\in[0,1]} \|\nabla_{\mathbf{u}} B(t\mathbf{u}_{q+1} + (1-t)\mathbf{u}^\star, \lambda_{q+1})\|, \\
\beta_2 &= \sup_{t\in[0,1]} \|\nabla_{\lambda} B(\mathbf{u}^\star, t\lambda_{q+1} + (1-t)\lambda^\star)\|, \\
\beta_3 &= \sup_{t\in[0,1]} \|\nabla_{\mathbf{u}} B(t\mathbf{u}_{q} + (1-t)\mathbf{u}^\star, \lambda_{q})\|, \\
\beta_4 &= \sup_{t\in[0,1]} \|\nabla_{\lambda} B(\mathbf{u}^\star, t\lambda_{q} + (1-t)\lambda^\star)\|.
\end{aligned}$$

This satisfies the bounded deterioration property with $\alpha_2 = \max(\beta_1, \beta_2, \beta_3, \beta_4)$. Given Assumption 4 holds, linear convergence of the SQP iterations follows directly as a consequence of [23, Theorem 3.3]. ■

6.5 Numerical Robustness of Dynamic Game SQP

We have shown that our proposed SQP approach exhibits linear convergence when close to a local GNE. However, as is commonly seen with numerical methods for nonlinear optimization, a naïve implementation of the procedure defined by (6.7), (6.8), and (6.10) often performs poorly due to overly aggressive steps leading to diverging iterates. In this section, we introduce multiple practical considerations which will help address this problem in practice. Namely, a merit function, a non-monotonic line search method, and a decaying regularization scheme. These components will be used to determine how much of the SQP step $p_q^{\mathbf{u}}$ and p_q^{λ} can be taken to make progress towards a local GNE while remaining in a region about the current iterate where the QP approximation (6.10) is valid. We will show through examples that these additions are crucial in improving the numerical robustness of the proposed SQP approach.

A Novel Merit Function

Merit functions are commonly used in conjunction with a backtracking line search technique as a mechanism for measuring progress towards the optimal solution and limiting the step size taken by an iterative solver [148]. This is important as the quality of the QP approximation (6.10) may be poor when far away from the iterate, which could result in divergence of the iterates if large steps are taken.

In traditional constrained optimization, merit functions typically track a combination of cost value and constraint violation. In the context of dynamic games, this is not as straightforward as the agents may have conflicting objectives and a proposed step may result in an increase in the objectives of some agents along with a decrease in others'. We also cannot just simply sum the objectives as minimizers of the combined cost function may not be local GNE. We therefore propose the following merit function:

$$\phi(\mathbf{u},\lambda, s; \mu) = \frac{1}{2}\| \underbrace{\begin{bmatrix} \nabla_{\mathbf{u}^1}\mathcal{L}^1(\mathbf{u},\lambda) \\ \vdots \\ \nabla_{\mathbf{u}^M}\mathcal{L}^M(\mathbf{u},\lambda) \end{bmatrix}}_{=\nabla\mathcal{L}(\mathbf{u},\lambda)} \|_2^2 + \mu\|C(\mathbf{u}) - s\|_1, \tag{6.16}$$

and define $\gamma(\mathbf{u},\lambda) = (1/2)\|\nabla\mathcal{L}(\mathbf{u},\lambda)\|_2^2$. The slack variable $s = \min(0, C(\mathbf{u}))$ is defined element-wise such that $C - s$ captures violation of the inequality constraints and we define the step $p^s = C(\mathbf{u}) + G(\mathbf{u})p^{\mathbf{u}} - s$. Compared to the merit function from [76], which only included the first term of (6.16), the novelty of our's is the l^1 norm term, whose purpose will be described shortly. Instead of measuring the agent objectives, our merit function tracks the first order optimality conditions in addition to constraint violation. It should be easy to see that the merit function attains a minimum of zero at any local GNE. However, we note that this merit function is not *exact* [148] since the first order conditions are only necessary for optimality.

Since we would like the sequence of iterates to converge to the minimizers of ϕ, it follows that at each iteration we would like take a step in a descent direction of ϕ. As such, we will now, in a manner similar to [148], analyze the directional derivative of the proposed merit function in the direction of the steps computed from (6.10) and describe a procedure for choosing the parameter μ and the corresponding *conditions* such that a descent in ϕ is guaranteed.

Since the l^1 norm is not differentiable everywhere, we begin by taking a Taylor series expansion of γ and $C - s$ for $\alpha \in (0, 1]$:

$$\begin{aligned} &\phi(\mathbf{u} + \alpha p^{\mathbf{u}}, \lambda + \alpha p^{\lambda}, s + \alpha p^s; \mu) - \phi(\mathbf{u}, \lambda, s; \mu) \\ &\leq \alpha\nabla_{\mathbf{u},\lambda}\gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} + \mu\|C - s + \alpha(Gp^{\mathbf{u}} - p^s)\|_1 - \mu\|C - s\|_1 + \beta\alpha^2\|p\|_2^2 \\ &= \alpha\left(\nabla_{\mathbf{u},\lambda}\gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} - \mu\|C - s\|_1\right) + \beta\alpha^2\|p\|_2^2, \end{aligned}$$

where the term $\beta\alpha^2\|p\|_2^2$ bounds the second derivative terms for some $\beta > 0$. Following a similar logic, we can obtain the bound in the other direction:

$$\begin{aligned} &\phi(\mathbf{u} + \alpha p^{\mathbf{u}}, \lambda + \alpha p^{\lambda}, s + \alpha p^s; \mu) - \phi(\mathbf{u}, \lambda, s; \mu) \\ &\geq \alpha\left(\nabla_{\mathbf{u},\lambda}\gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} - \mu\|C - s\|_1\right) - \beta\alpha^2\|p\|_2^2. \end{aligned}$$

Dividing the inequality chain by α and taking the limit $\alpha \to 0$, we obtain the directional derivative:

$$D(\phi(\mathbf{u}, \lambda, s; \mu), p^{\mathbf{u}}, p^{\lambda}) = \nabla_{\mathbf{u},\lambda}\gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} - \mu\|C - s\|_1 \tag{6.17}$$

From (6.17), it should be clear that given $C - s \neq 0$, there exists a value for $\mu > 0$ such that the directional derivative is negative. As such, we propose the following expression to compute μ, given some $\rho \in (0, 1)$:

$$\mu \geq \left(\nabla_{\mathbf{u},\lambda} \gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} \right) / ((1 - \rho) \|C - s\|_1), \tag{6.18}$$

which results in $D(\phi(\mathbf{u}, \lambda; \mu), p^{\mathbf{u}}, p^{\lambda}) \leq -\rho\mu \|C - s\|_1$.

In the case when $C - s = 0$, we unfortunately cannot guarantee that the directional derivative will always be negative. To see this, let us analyze the first term of (6.17):

$$\begin{aligned} \nabla_{\mathbf{u},\lambda} \gamma \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} &= \nabla \mathcal{L}^\top \begin{bmatrix} L & G^\top \end{bmatrix} \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} \\ &= \nabla \mathcal{L}^\top \begin{bmatrix} B + R & G^\top \end{bmatrix} \begin{bmatrix} p^{\mathbf{u}} \\ p^{\lambda} \end{bmatrix} \\ &= -\nabla \mathcal{L}^\top \nabla \mathcal{L} + \nabla \mathcal{L}^\top R p^{\mathbf{u}}, \end{aligned} \tag{6.19}$$

where R denotes the residual matrix i.e. $R = L - B$ and we arrive at the third equality by plugging in for the stationarity condition from (6.10). From (6.19), it should be immediately apparent that when the residual matrix is large and dominates the first term, it is possible for the directional derivative to be positive. This can be interpreted as a condition on how well the positive definite B actually approximates the original stacked Hessian matrix L. For dynamic games where the agents have highly coupled and differing objectives, i.e. when L is highly non-symmetric, it is reasonable to expect that the approximation would suffer and that we may not be able to achieve a decrease in the merit function. For this reason, we utilize a non-monotone strategy for the line search step, which will be discussed in the following.

A Non-Monotone Line Search Strategy

Line search methods are used in conjunction with merit functions to achieve a compromise between the goals of making rapid progress towards the optimal solution and keeping the iterates from diverging. This is done by finding the largest step size $\alpha \in (0, 1]$ such that the following standard decrease condition is satisfied [23]:

$$\begin{aligned} &\phi(\mathbf{u}_q + \alpha p_q^{\mathbf{u}}, \lambda_q + \alpha p_q^{\lambda}, s_q + \alpha p_q^{s}; \mu) \\ &\quad \leq \phi(\mathbf{u}_q, \lambda_q, s_q; \mu) + \zeta \alpha D(\phi(\mathbf{u}_q, \lambda_q, s_q; \mu), p_q^{\mathbf{u}}, p_q^{\lambda}), \end{aligned} \tag{6.20}$$

where $\zeta \in (0, 0.5)$. However, since our merit function is not exact, the line search procedure can be susceptible to poor local minima which do not correspond to local GNE. Moreover, there is evidence that requiring monotonic decrease in the merit function at each iteration may actually impede the solution process [49]. We therefore include in our approach a non-monotone approach to line search called the *watchdog* strategy [34]. Instead of insisting on sufficient decrease in the merit function at every iteration, this approach allows for relaxed steps to be taken for a certain number of iterations where increases in the merit function are allowed. The decrease requirement (6.20) is then enforced after the prescribed number of relaxed iterations or if the step size exceeds some given threshold. The rationale behind this strategy is that we can use the relaxed steps as a way to escape regions where it is difficult to make progress w.r.t. the merit function. The algorithm we implement is based on the non-monotone stabilization scheme presented in [38] and involves the taking of *d-steps* and *m-steps*, where *d-steps* correspond to relaxed steps ($\alpha = 1$) which disregard the merit decrease condition and *m-steps* which enforce merit decrease through an Armijo backtracking linesearch on α in (6.20). Furthermore, the method stores the list of iterates which satisfy the merit decrease condition as checkpoints. This allows for a *watchdog* step to reset the iterate to a checkpoint in the event that an *m-step* was unsuccessful and to perform a backtracking linesearch from there.

A Decaying Regularization Scheme

The regularization parameter ϵ introduced in (6.8) is a common mechanism used in iterative Newton-based procedures to control the size of $p^{\mathbf{u}}$. However, the selection of the regularization parameter is non-trivial in many cases as the convergence behavior of the iterative procedure can be highly sensitive to the value of ϵ. Often, it is observed that iterates can diverge if ϵ is too small, whereas progress is slow if ϵ is too big. To address this challenge, one method which has seen success is that of decaying regularization which given some initial setting of ϵ gradually decreases its value over iterations of the algorithm [78, 141, 95]. In this work, we take a similar approach to updating the values of ϵ, but importantly, integrate it into the non-monotone line search procedure. Specifically, for some initial regularization value $\epsilon_0 \geq 0$ and decay multiplier $\eta \in (0, 1]$, we impose the following update rule:

$$\epsilon_{q+1} = \begin{cases} \epsilon_q & \text{if } \textit{d-step}, \\ \eta \epsilon_q & \text{if } \textit{m-step}, \end{cases} \tag{6.21}$$

where the regularization value only decays after a step which satisfies the merit function decrease condition is found. The reason for this is to avoid reducing the regularization strength when potentially spurious relaxed *d-steps* are taken and the iterate is reset to a checkpoint.

6.6 The Dynamic Game SQP Algorithm

By combining the elements discussed in the previous section, we arrive at the dynamic game SQP (DG-SQP) algorithm, which is presented in Algorithm 7. The algorithm requires as

Algorithm 7: Dynamic Game SQP (DG-SQP)

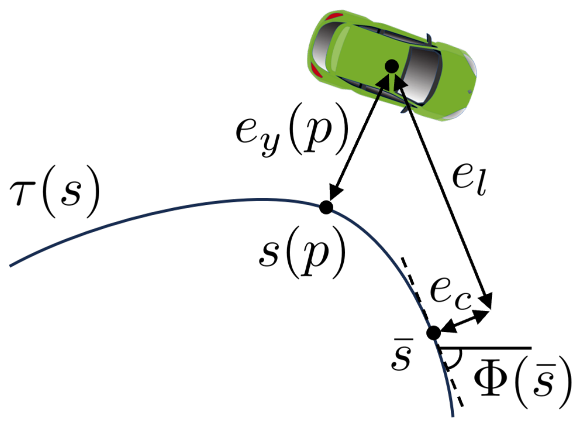
```
   Input: $\mathbf{u}_0$, $\epsilon_0$
1  $q \leftarrow 0$;
2  $\mathbf{u}_q \leftarrow \mathbf{u}_0$, $\lambda_q \leftarrow \max(0, -(G_0 G_0^\top)^{-1} G_0 h_0)$;
3  while not converged do
4      $B_q(\epsilon_q),\ h_q,\ G_q,\ C_q \leftarrow$ (6.8);
5      $p_q^{\mathbf{u}},\ p_q^{\lambda} \leftarrow$ (6.10), (6.11);
6      $s_q \leftarrow \min(0, C_q)$, $p_q^s \leftarrow C_q + G_q p_q^{\mathbf{u}} - s_q$;
7      if $C_q - s_q \neq 0$ then
8          Compute $\mu$ from (6.18);
9      else
10         $\mu \leftarrow 0$;
11     end
12     $\mathbf{u}_{q+1},\ \lambda_{q+1}, \epsilon_{q+1} \leftarrow$ watchdog line search;
13     $q \leftarrow q + 1$
14 end
15 return $\mathbf{u}^\star \leftarrow \mathbf{u}_q$, $\lambda^\star \leftarrow \lambda_q$;
```

input initial guesses of open-loop input sequences for each agent and the initial regularization value ϵ_0. Line 2 initializes the primal and dual iterates, where the dual variables are initialized as the least squares solution to (6.6a). Lines 3 to 14 perform the SQP iteration which has been described in Sections 6.4 and 6.5. An iterate is said to have converged to a local GNE if it satisfies the KKT conditions described in (6.6) up to some user specified tolerance. Namely, for some given $\rho_1,\ \rho_2,\ \rho_3 > 0$, we require the conditions $\|\nabla \mathcal{L}(\mathbf{u}_q, \lambda_q)\|_\infty \leq \rho_1,\ \|C(\mathbf{u}_q)\|_\infty \leq \rho_2,\ |\lambda_q^\top C(\mathbf{u}_q)| \leq \rho_3$ be satisfied in order for the algorithm to terminate successfully. The algorithm outputs the open-loop strategies for the M agents and the corresponding Lagrange multipliers.

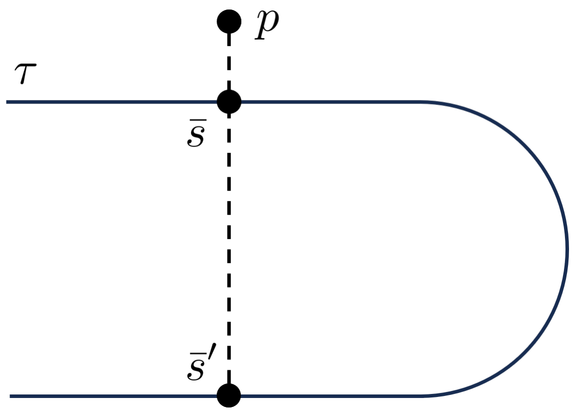
6.7 Frenet-Frame Games for Autonomous Racing

This section presents the second main contribution of this paper. We formulate the autonomous racing problem as a dynamic game using ideas from contouring control [87, 47] and leverage the solver presented in the previous section to compute the solution to such game. Numerical studies in Section 6.8 will show that with this formulation, we achieve significantly higher success rates when solving for GNE of racing dynamic games.

We consider the class of racing dynamic games where the race track is represented as a parametric path $\tau : [0, L] \mapsto \mathbb{R}^2$, which is assumed to be twice differentiable and of total length L. In particular, when given an arclength $s \in [0, L]$, τ returns the inertial x-y position of the path and we may write the path tangent angle at s is $\Phi(s) = \arctan(\tau_y'(s)/\tau_x'(s))$. In the case where τ forms a closed circuit we have that $\tau(0) = \tau(L)$, $\tau'(0) = \tau'(L)$, $\tau''(0) = \tau''(L)$, where τ' and τ'' are the element-wise first and second derivatives of τ. Given a path τ, we



(a)



(b)

Figure 6.1: (a) Illustration of the contouring and lag errors e_c and e_l. (b) An example of a poor approximation $\bar{s}'$ where $e_l(p, \bar{s}) = e_l(p, \bar{s}') = 0$, but $\bar{s}' \neq s(p)$.

may then transform an inertial frame position $p = (x, y)$ and yaw angle ψ into a path-relative pose as follows:

$$s(p) = \arg\min_s \|\tau(s) - p\|_2, \tag{6.22a}$$

$$e_y(p) = [-\sin\Phi(s(p)), \cos\Phi(s(p))](p - \tau(s(p))), \tag{6.22b}$$

$$e_\psi(p, \psi) = \psi - \Phi(s(p)),$$

which are the path progress, the lateral displacement from the path, and the heading deviation from the path tangent corresponding to p and ψ respectively. These quantities, which are illustrated in Figure 6.1a, are known as the coordinates of a Frenet reference frame [96] and they enable a straightforward expression of quantities which are important to the formulation of the autonomous racing problem as a dynamic game [129]. First, we may express the difference in track progress between two agents as $s(p^i) - s(p^j)$. Maximization of this quantity in the game objective J^i (6.2b) captures the direct competition between agents i and j in racing. Furthermore, using the lateral deviation in (6.22b), we may easily express track boundary constraints for agent i as simple bounds on $e_y^i(p^i)$, which are then included in the game constraints C (6.3). Repeated evaluation of the potentially nonlinear optimization problem (NLP) in (6.22a) can be avoided by directly describing the evolution of a vehicle's kinematics in the Frenet reference frame [96]. In this work, we consider two vehicle models which can be expressed this way. The first is the Frenet-frame kinematic bicycle model which, for vehicle i, has the state and input vector:

$$x^i_{\text{kin}} = [v, s, e_y, e_\psi]^\top \in \mathcal{X}^i_{\text{kin}}, \tag{6.23a}$$

$$u^i_{\text{kin}} = [a, \delta_f]^\top \in \mathcal{U}^i_{\text{kin}}, \tag{6.23b}$$

where v and a are the velocity and acceleration of vehicle i along its direction of travel and δ_f is the steering angle of the front wheel. The second is the Frenet-frame dynamic bicycle model which, for vehicle i, has the state and input vector:

$$x^i_{\text{dyn}} = [v_x, v_y, \omega, s, e_y, e_\psi]^\top \in \mathcal{X}^i_{\text{dyn}}, \tag{6.24a}$$
$$u^i_{\text{dyn}} = [a_x, \delta_f]^\top \in \mathcal{U}^i_{\text{dyn}}, \tag{6.24b}$$

where v_x and v_y are the longitudinal and lateral velocity, ω is the yaw rate, and a_x is the longitudinal acceleration of vehicle i. The following functions describe the state evolution of the kinematic or dynamic bicycle models and expressions for both can be found in the Appendix.

$$x^{i,+}_{\text{kin}} = f^i_{\text{kin}}(x^i_{\text{kin}}, u^i_{\text{kin}}) \tag{6.25}$$
$$x^{i,+}_{\text{dyn}} = f^i_{\text{dyn}}(x^i_{\text{dyn}}, u^i_{\text{dyn}}) \tag{6.26}$$

These can then be concatenated to construct the game dynamics in (6.1):

$$f_* = \text{vec}(f^1_*, \dots, f^M_*), \tag{6.27}$$

where the subscript $* \in \{\text{kin}, \text{dyn}\}$ indicates the vehicle model. A Frenet-frame dynamic game with either the kinematic or dynamic bicycle model can therefore be defined as:

$$\Gamma_* = (N, \mathcal{X}_*, \mathcal{U}_*, f_*, \{J^i_*\}^M_{i=1}, C_*). \tag{6.28}$$

Direct expression of the vehicle kinematics in a Frenet reference frame offers two benefits in the formulation of racing tasks. The first is that it allows for the expression of progress maximization as a linear function of the state variable s. The second is that it allows for track boundary constraints to be imposed as simple bounds on the state variable e_y [111, 67, 161]. However, this comes at the cost of introducing singularities at the centers of curvature (as seen in the numerator of (A.1)) which can cause numerical instabilities especially on tracks with tight turns. Furthermore, Frenet-frame kinematics also complicate the expression of obstacle avoidance constraints in multi-agent settings, which will be seen in our description of the racing scenarios in Section 6.8. Our formulation, which applies ideas from contouring control [87, 47, 77] in the context of dynamic games, approximates the evolution of s and e_y without introducing singularities, and results in a dynamic game with inertial kinematics, where the approximations of s and e_y can be leveraged in a similar manner as before for simple description of racing tasks.

Let us first define the inertial-frame counterparts of the kinematic and dynamic bicycle models for vehicle i, which have as state and input vectors

$$\bar{x}^i_{\text{kin}} = [v, x, y, \psi]^\top, \in \bar{\mathcal{X}}^i_{\text{kin}} \tag{6.29a}$$
$$\bar{u}^i_{\text{kin}} = [a, \delta_f]^\top \in \bar{\mathcal{U}}^i_{\text{kin}}. \tag{6.29b}$$

and

$$\bar{x}^i_{\text{dyn}} = [v_x, v_y, \omega, x, y, \psi]^\top \in \bar{\mathcal{X}}^i_{\text{dyn}}, \tag{6.30a}$$
$$\bar{u}^i_{\text{dyn}} = [a_x, \delta_f]^\top \in \bar{\mathcal{U}}^i_{\text{dyn}}, \tag{6.30b}$$

respectively. The dynamics functions describe the state evolution of the inertial-frame kinematic or dynamic bicycle models and expressions for both are again included in the Appendix.

$$\bar{x}^{i,+}_{\text{kin}} = \bar{f}^i_{\text{kin}}(\bar{x}^i_{\text{kin}}, \bar{u}^i_{\text{kin}}) \tag{6.31}$$

$$\bar{x}^{i,+}_{\text{dyn}} = \bar{f}^i_{\text{dyn}}(\bar{x}^i_{\text{dyn}}, \bar{u}^i_{\text{dyn}}) \tag{6.32}$$

We begin the derivation of our approximation by augmenting the state and input vectors for vehicle i with the variables $\bar{s}^i$ and $\bar{v}^i$ to obtain

$$\hat{x}^i_* = (\bar{x}^i_*, \bar{s}^i) \in \hat{\mathcal{X}}^i_*, \;\; \hat{u}^i_* = (\bar{u}^i_*, \bar{v}^i) \in \hat{\mathcal{U}}^i_*, \tag{6.33}$$

where we use the subscript $* \in \{\text{kin}, \text{dyn}\}$ to indicate that this augmentation may be performed on the state and input vectors for either the kinematic and dynamic bicycle models. The variable $\bar{s}^i$ represents our approximation of the progress s for vehicle i along a given path τ and evolves according to simple integrator dynamics:

$$\bar{s}^i_{k+1} = \bar{s}^i_k + \Delta t \cdot \bar{v}^i_k, \tag{6.34}$$

with initial condition $\bar{s}^i_0 = s(p^i_0)$, where $\bar{v}^i$ is a decision variable that can be thought of as an approximate arcspeed for agent i. We denote the augmented dynamics function as

$$\hat{x}^{i,+}_* = \hat{f}^i_*(\hat{x}^i_*, \hat{u}^i_*), \tag{6.35}$$

which is defined as the concatenation of the original inertial-frame vehicle dynamics $\bar{f}^i_*$ in (6.31) or (6.32) with (6.34). Construction of the game dynamics $\hat{f}_*$ can then be done by simply concatenating the augmented dynamics functions $\hat{f}^i_*$ in (6.35) for all vehicles:

$$\hat{f}_* = \text{vec}(\hat{f}^1_*, \ldots, \hat{f}^M_*). \tag{6.36}$$

Now, given an inertial position p^i and approximate progress $\bar{s}^i$, we define the lag error for vehicle i as follows:

$$e_l(p^i, \bar{s}^i) = [-\cos\Phi(\bar{s}^i), -\sin\Phi(\bar{s}^i)](p^i - \tau(\bar{s}^i)). \tag{6.37}$$

Specifically, the lag error approximates the difference between $\bar{s}^i$ and $s(p^i)$. We note that the complement of the lag error can be written as $e_c(p^i, \bar{s}^i) = [-\sin\Phi(\bar{s}^i), \cos\Phi(\bar{s}^i)](p^i - \tau(\bar{s}^i))$. This term is known as the contouring error [77] and it approximates the lateral deviation of the vehicle from the path at $\bar{s}^i$. It is straightforward to see through the illustration in Figure 6.1a that when $\bar{s}$ is in a local neighborhood of $s(p)$ then $e_l \approx 0$ implies that $\bar{s} \approx s(p)$ and $e_c \approx e_y(p)$. However, if $\bar{s}$ is far from $s(p)$, as in the case of Figure 6.1b, then it is entirely possible that the difference between $\bar{s}$ and $s(p)$ is arbitrarily large despite $e_l = 0$. Therefore, in order to for $\bar{s}$ to be a good approximation, we must not only drive e_l to zero, but we must also attempt to keep $\bar{s}$ close to $s(p)$. This leads us to make the following two modifications to the game objectives and constraints.

First, to minimize the lag error, we introduce it into the augmented game objective for agent i as follows:

$$\hat{J}^i_*(\hat{\mathbf{u}}^i_*, \hat{\mathbf{u}}^{\neg i}_*) = J^i_*(\hat{\mathbf{u}}^i_*, \hat{\mathbf{u}}^{\neg i}_*) + \sum_{k=1}^{N} q_l e_l(p^i_k(\bar{\mathbf{u}}^i_*), \bar{s}^i_k(\bar{\mathbf{v}}^i))^2, \tag{6.38}$$

where J^i_* is the original game objective for agent i in (6.28), $q_l \gg 0$ is a weight on the lag error cost, and $p^i_k(\cdot)$ and $\bar{s}^i_k(\cdot)$ denote functions which, given the input sequences $\bar{\mathbf{u}}^i$ and $\bar{\mathbf{v}}^i$, roll out the dynamics function $\hat{f}^i_*$ up to time step k and return the x-y position and approximate progress respectively. For brevity, we have omitted the explicit dependence of $\hat{J}^i_*$ on $\bar{s}^i_0$: the path progress corresponding to the state $\hat{x}^i_{*,0}$. This can be easily computed via (6.22a) as $\bar{s}^i_0 = s(p^i_0)$. Furthermore, under a slight abuse of notation, we replace the arguments of the original Frenet-frame game objectives in (6.28) (i.e. the first term in (6.38)) with the augmented input sequences to reflect that any terms in J^i_* which originally depended on s^i and $s^{\neg i}$ should be replaced with the approximations $\bar{s}^i$ and $\bar{s}^{\neg i}$. Second, to keep $\bar{s}^i_k$ close to $s(p^i_k)$, we impose the simple bounds on the approximate arc speed input: $|\bar{v}^i_k| \leq v_{\max}$. This helps in avoiding undesirable local minima of the lag error term in (6.38) which correspond to poor approximations with $\bar{s}^i_k$ as shown in Figure 6.1b. We denote the concatenation of the arcspeed bounds for all agents $i \in \{1, \dots, M\}$ and time steps $k \in \{1, \dots, N\}$ as $C_v(\bar{\mathbf{v}}^1, \dots, \bar{\mathbf{v}}^M)$.

Next, we turn to the track boundary constraints which can no longer be expressed as simple bounds on e_y. Instead, we may conveniently use the contouring error e_c in Figure 6.1a as an approximation to e_y to define the following constraints for agent i.

$$-w^-(\bar{s}^i_k(\bar{\mathbf{v}}^i)) \leq e_c(p^i_k(\bar{\mathbf{u}}^i_*), \bar{s}^i_k(\bar{\mathbf{v}}^i)) \leq w^+(\bar{s}^i_k(\bar{\mathbf{v}}^i)), \tag{6.39}$$

where $w^+, w^- : [0, L] \mapsto \mathbb{R}_{++}$ are functions which, for a value of s, return the distance from $\tau(s)$ to the track boundaries in the directions normal to τ at s.

Finally, we define the augmented constraint function $\hat{C}_*$ as the concatenation of the original constraints of the dynamic game C_* in (6.28) with the arc speed bounds:

$$\hat{C}_*(\hat{\mathbf{u}}^1, \dots, \hat{\mathbf{u}}^M) = \mathrm{vec}(C_*(\hat{\mathbf{u}}^1, \dots, \hat{\mathbf{u}}^M), C_v(\bar{\mathbf{v}}^1, \dots, \bar{\mathbf{v}}^M)), \tag{6.40}$$

where through a similar abuse of notation as before, we replace the arguments of the original Frenet-frame game constraints C_* with the augmented input sequences to reflect that the original track boundary constraints in e_y should be replaced with (6.39). Combining the elements discussed above, we define the following dynamic game:

$$\hat{\Gamma}_* = (N, \hat{\mathcal{X}}_*, \hat{\mathcal{U}}_*, \hat{f}_*, \{\hat{J}^i_*\}_{i=1}^M, \hat{C}_*), \tag{6.41}$$

which can be viewed as an approximation to the racing dynamic game Γ_* in (6.28), which uses the exact Frenet-frame kinematics when defining the game dynamics, agent objective, and constraint functions. Table 6.1 summarizes the components of each of the exact and approximate dynamic games which will be solved in the following numerical studies.

Table 6.1: Summary of Exact and Approximate Dynamic Games

	Dynamics	Objectives	Constraints
Γ_{kin}	f_{kin} (6.25),(6.27)	$\{J_{\text{kin}}\}_{i=1}^{M}$ (6.2b)	C_{kin} (6.3)
Γ_{dyn}	f_{dyn} (6.26),(6.27)	$\{J_{\text{dyn}}\}_{i=1}^{M}$ (6.2b)	C_{dyn} (6.3)
$\hat{\Gamma}_{\text{kin}}$	$\hat{f}_{\text{kin}}$ (6.31),(6.35),(6.36)	$\{\hat{J}_{\text{kin}}\}_{i=1}^{M}$ (6.2b),(6.38)	$\hat{C}_{\text{kin}}(6.3), (6.40)$
$\hat{\Gamma}_{\text{dyn}}$	$\hat{f}_{\text{dyn}}$ (6.32),(6.35),(6.36)	$\{\hat{J}_{\text{dyn}}\}_{i=1}^{M}$ (6.2b),(6.38)	$\hat{C}_{\text{dyn}}(6.3), (6.40)$

6.8 Numerical Study

In the following numerical studies, we examine the performance of our DG-SQP solver in solving for *open-loop* GNE of three head-to-head racing scenarios, which are defined in Part A. In Part B, we examine the effect of the proposed merit function and non-monotone line search strategy on the success rate of the DG-SQP solver via an ablation study on these components. In Part C, we investigate the effect of the proposed decaying regularization scheme through a sensitivity study on the regularization value and decay rate. In Part D, we compare the performance of our DG-SQP solver against the PATH solver from [38] and show improved performance when solving both the exact and approximate dynamic games Γ_* and $\hat{\Gamma}_*$. Our DG-SQP solver was implemented in Python and for the PATH solver, we use the Julia implementation `PATHSolver.jl` for comparison, but call it through a custom Python wrapper in order for us to easily pass identical definitions of the following dynamic games to it. Our implementation can be found at `https://github.com/zhu-edward/DGSQP`.

Scenario Description

We perform our numerical studies on three head-to-head racing scenarios which were constructed to showcase the performance of our DG-SQP solver with different vehicle models and track geometries.

Scenario 1

In this scenario, we use the L-shaped track shown in the left plot of Figure 6.2, where the vehicles travel in the counter clockwise direction from the red start line. We model both vehicles using the kinematic bicycle model f_{kin}^{i} with parameters that correspond to a 1/10th scale RC car. The discrete time dynamics are obtained via 4th order Runge-Kutta discretization with a time step of $\Delta t = 0.1s$. The Frenet-frame, or exact, dynamic game Γ_{kin}

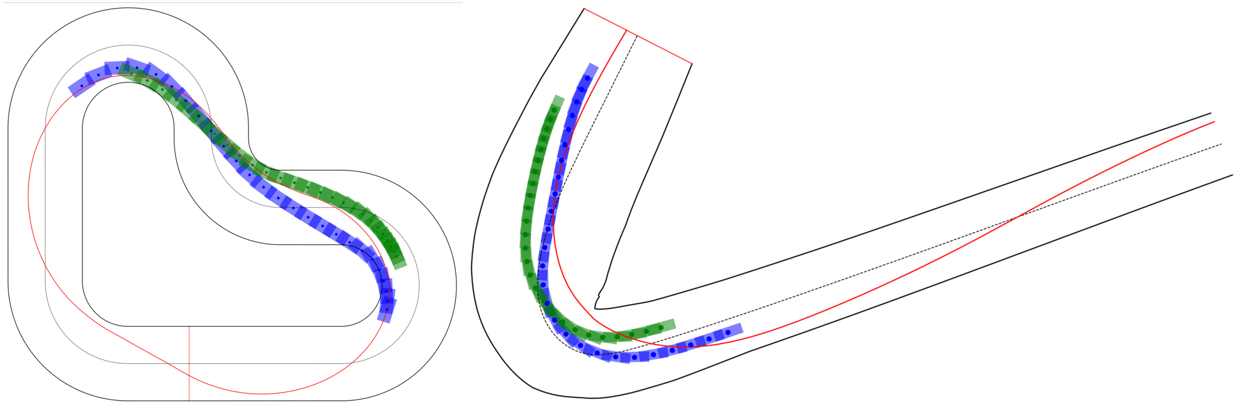


Figure 6.2: Example head-to-head racing open-loop GNE solutions with $N = 25$ for the L-shaped track and 1/10th scale RC car (left) and the first hairpin turn of the Austin F1 track and a full scale race car (right). The red trajectories show pre-computed race lines which are used to warm start Scenarios 2 and 3 of our numerical study. In both plots, the cars are traveling in the counter-clockwise direction.

is defined with the game dynamics $f_{\text{kin}} = \text{vec}(f^1_{\text{kin}}, f^2_{\text{kin}})$ and objectives:

$$J^1_{\text{kin}}(\mathbf{u}^1, \mathbf{u}^2) = \sum_{k=0}^{N-1} \|u^1_k\|^2_{R^1} + \|\Delta u^1_k\|^2_{P^1} + q^1_2 s^2_N(\mathbf{u}^2) - q^1_1 s^1_N(\mathbf{u}^1)$$
$$J^2_{\text{kin}}(\mathbf{u}^2, \mathbf{u}^1) = \sum_{k=0}^{N-1} \|u^2_k\|^2_{R^2} + \|\Delta u^2_k\|^2_{P^2} + q^2_1 s^1_N(\mathbf{u}^1) - q^2_2 s^2_N(\mathbf{u}^2)$$

where $\Delta u^i_k = u^i_k - u^i_{k-1}$, u^i_{-1} is the input from the previous time step, $R^i, P^i \succeq 0$ are weights on the quadratic input and input rate penalties, $q^i_1, q^i_2 > 0$ are weights on the competition costs, and $\|x\|^2_A = x^\top Ax$. The constraints C_{kin} are defined as follows:

$$u_l \leq u^i_k \leq u_u, \; i = 1, 2 \tag{6.42a}$$
$$-w/2 \leq e^i_{y,k}(\mathbf{u}^i) \leq w/2, \; i = 1, 2 \tag{6.42b}$$
$$\|p^1_k(\mathbf{u}^1) - p^2_k(\mathbf{u}^2)\|^2_2 \geq (r^1 + r^2)^2, \tag{6.42c}$$

which consist of input limits (6.42a), track boundary constraints (6.42b), and collision avoidance constraints (6.42c). In the above, u_l and u_u are the lower and upper bounds on the input, w is the track width, and r^1, r^2 are the radii of the circular collision buffers for the two vehicles. When implementing the collision avoidance constraint for the exact dynamic game Γ_{kin}, the mapping $p^i_k(\mathbf{u}^i)$, which returns the inertial x-y position of agent i at time step k, is written as:

$$p^i_k(\mathbf{u}^i) = \tau(s^i_k(\mathbf{u}^i)) + \frac{e^i_{y,k}(\mathbf{u}^i)}{\|\tau'(s^i_k(\mathbf{u}^i))\|_2} \begin{bmatrix} -\tau'_y(s^i_k(\mathbf{u}^i)) \\ \tau'_x(s^i_k(\mathbf{u}^i)) \end{bmatrix}. \tag{6.43}$$

For the approximate game $\hat{\Gamma}_{\text{kin}}$, we construct the game dynamics as $\hat{f}_{\text{kin}} = \text{vec}(\hat{f}^1_{\text{kin}}, \hat{f}^2_{\text{kin}})$. The game objectives J^i_{kin} are obtained by replacing any occurrences of $s^i_N(\cdot)$ with $\bar{s}^i_N(\cdot)$ in the expressions for J^i_{kin} above. The game constraints are constructed as $\hat{C}_{\text{kin}} = \text{vec}(C_{\text{kin}}, C_v)$ where $e^i_{y,k}(\mathbf{u}^i)$ is replaced with $e_c(p^i_k(\bar{\mathbf{u}}^i), \bar{s}^i_k(\bar{\mathbf{v}}^i))$ in the original track boundary constraints (6.42b) in C_{kin}. When implementing the collision avoidance constraint for the approximate dynamic game $\hat{\Gamma}_{\text{kin}}$, the mapping $p^i_k(\bar{\mathbf{u}}^i)$ simply extracts elements corresponding to x and y from the state vector of vehicle i at time k. Finally, for this scenario, given an initial condition, we warm start the solver using trajectories obtained via a PID controller which attempt to maintain the speed v and lateral deviation e_y as specified in the initial condition.

Scenario 2

This scenario is identical to the previously described Scenario 1 in all aspects except for the game dynamics and warm start strategy, where instead, the vehicle is modeled using the dynamic bicycle model [73] and the tire forces are described via Pacejka tire models [102] with parameters that correspond to a 1/10th scale RC car. Γ_{dyn} and $\hat{\Gamma}_{\text{dyn}}$ are therefore constructed in a similar manner as Scenario 1. Warm starting of the solvers is done with trajectories computed from an optimal control problem which attempts to track a pre-computed race line (the red line in the left plot of Figure 6.2) for the two vehicles but *does not* consider collision avoidance.

Scenario 3

In this scenario, we use a segment of the Austin F1 track corresponding to the first hairpin turn, as shown in the right plot of Figure 6.2, and again model the vehicles using the dynamic bicycle model. Model parameters are chosen which match those of a full scale race car. The agent objectives in this scenario remain identical to Scenario 1 and the constraints are also the same, with the exception of the the track boundary constraints which, due to the variable track width, are now written as:

$$-w_-(s^i_k(\mathbf{u}^i)) \leq e^i_{y,k}(\mathbf{u}^i) \leq w_+(s^i_k(\mathbf{u}^i)),$$

where $w_-, w_+ : [0, L] \mapsto \mathbb{R}_{++}$ are functions which return the width of the track in the directions normal to the parametric path at the given arclength s. We also use the same warm start strategy as in Scenario 2 with the race line shown in the right plot of Figure 6.2.

An Ablation Study on the Merit Function and Non-Monotone Line Search Strategy

We first demonstrate the value of our proposed merit function and non-monotone line search strategy, by performing an ablation study on these components for the dynamic game $\hat{\gamma}_{\text{kin}}$ described in Scenario 1 with a horizon of $N = 25$ and the cost parameter settings $q^1_1 = q^1_2 = q^2_1 = q^2_2 = 1$ and $R^1 = R^2 = P^1 = P^2 = \text{diag}(1e-1, 1e-1, 1e-4)$. We compare convergence results from the full DG-SQP approach presented in Algorithm 7 with three ablative cases.

Table 6.2: Ablation study convergence results

	no ablation	ablation merit function	ablation line search	ablation both
Conv.	45	5	9	4

In the first, we replace the merit function (6.16) with the sum of all of the agents' objectives:

$$\phi(\mathbf{u}, s; \mu) = \sum_{i=1}^{M} J^i(\mathbf{u}^1, \dots, \mathbf{u}^M) + \mu \|C(\mathbf{u}) - s\|_1, \tag{6.44}$$

which is a generalization of the standard choice in nonlinear optimization approaches [148]. In the second, we do not use the non-monotone line search strategy and instead employ a standard backtracking line search which enforces the merit decrease condition (6.20) at every iteration. In the third, we ablate both the merit function and line search scheme. The results are shown in Table 6.2 where GNE are solved for using DG-SQP and the two ablative cases. This is done over 100 randomly sampled joint initial conditions on the L-shaped track in Figure 6.2 where the vehicles start within 1.2 car lengths of each other and are traveling at velocities that exhibit at most a 25% difference. We count the number of trials where the solvers reached convergence with thresholds $\rho_1, \rho_2, \rho_3 = 10^{-4}$. From the results, it can be clearly seen that certainly both the proposed merit function and the non-monotone line search strategy have a significant impact on the numerical robustness of the solver. In particular, we observe a 89% and 80% reduction in solver success rate when the two components are ablated respectively. The choice in merit function plays an important role due to the agent objectives being very close to zero-sum. As for the non-monotone line search, we believe it's impact is due to the globalizing effect that the non-monotone scheme has on the solution process, which was shown in [38] for the PATH solver.

A Sensitivity Study on the Decaying Regularization Scheme

Next, we investigate the sensitivity of our approach to the regularization values and decay rates, by performing a sensitivity study over a grid of these values for the dynamic game from Scenario 1 with $q_1^1 = 6,\ q_2^1 = 5,\ q_2^2 = 6,\ q_1^2 = 5$ and $R^1 = R^2 = P^1 = P^2 = \text{diag}(1, 1, 1e-4)$. In particular, we again randomly sample 100 joint initial conditions and count the number of trials where the solver reached convergence for values of $\epsilon_0 \in \{0, 0.1, 1, 10, 1000\}$ and $\eta \in \{0.5, 0.65, 0.8, 0.95, 1\}$. Note that $\epsilon_0 = 0$ corresponds to no regularization (in which case the value of the decay rate becomes irrelevant), and $\eta = 1$ corresponds to no regularization decay, which corresponds to standard regularization schemes with a constant regularizer. The results of this study are shown in Figure 6.3 for a horizon length of $N = 15$, where we may observe 1) the importance of regularization, as the success rate is lowest when no regularization is used (first column), and 2) the sensitivity of solver performance to the choice of a constant ϵ_0, where the solver's success rate is highly dependent on ϵ_0 when no decay is performed (last row). This is especially apparent for larger values of ϵ_0 where we can see that

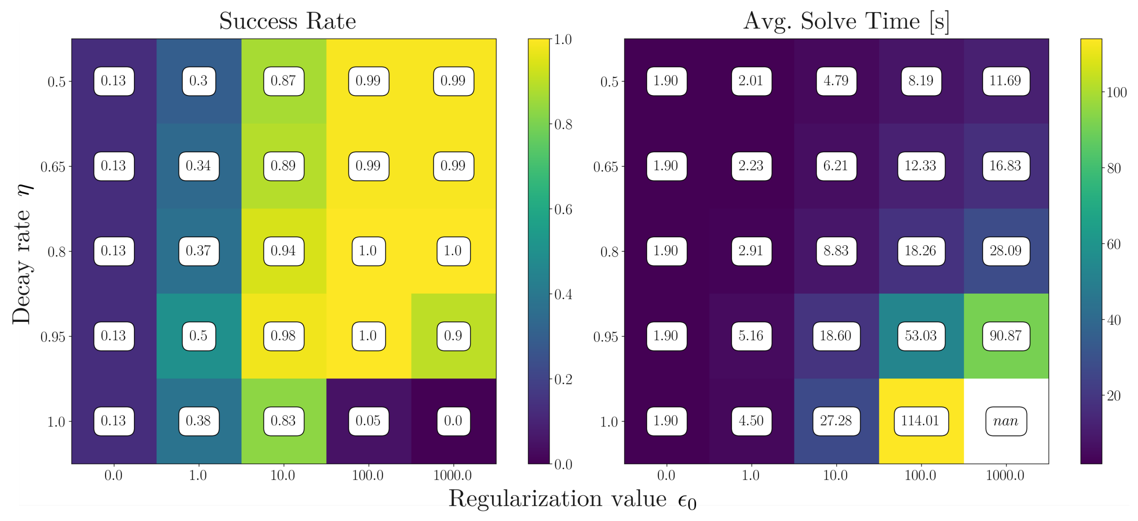


Figure 6.3: DG-SQP success rates (left) and average solve times (right) for $\hat{\Gamma}_{\text{kin}}$ with $N = 15$ over different regularization weight (x-axis) and decay rate (y-axis) settings. "nan" indicates that no data was available for that setting.

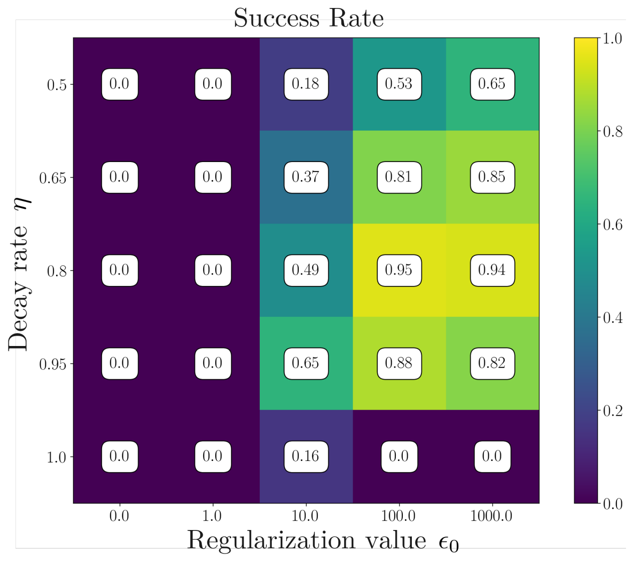


Figure 6.4: DG-SQP success rates for $\hat{\Gamma}_{\text{kin}}$ with $N = 25$.

the average solve time increases dramatically due to the small step sizes. Next, by looking along the rows of Figure 6.3, it can be seen that decaying regularization helps to alleviate the sensitivity of solver convergence to the choice in ϵ_0 as we observe nontrivial success rates over a range of magnitudes for ϵ_0. Finally, by looking along the columns of Figure 6.3, we see that while some settings of decay rate can certainly lead to better success rates, it appears that a fairly large range of values are sufficient to stabilize the solution process to achieve high success rates of >80%. In Figure 6.4, we show success rates for a horizon length of $N = 25$ which largely follow the trends in Figure 6.3, but are more exaggerated due to the longer horizon. We note that the best success rates for both horizon lengths are observed for the same settings of decay rate and initial value, i.e. $\eta = 0.8$ and $\epsilon_0 = 100$, and these values are what we use in the following experiments.

Solver and Model Comparison for Racing Tasks

The PATH solver [38] is considered to be the state-of-the-art in solving for GNE of open-loop dynamic games and has been used effectively in many multi-agent navigation tasks [90, 104]. However, these works have primarily demonstrated its performance on games with inertial kinematic models. We now compare its performance against our DG-SQP approach for the exact and approximate games Γ_* and $\hat{\Gamma}_*$ in the head-to-head racing scenarios described above.

Scenario 1

For this scenario, we randomly sample 500 collision-free initial conditions where the agents start within 1.2 car lengths of each other and are traveling at velocities that exhibit at most a 25% difference. For each initial condition, we solve for GNE of Γ_{kin} and $\hat{\Gamma}_{\text{kin}}$ with horizon lengths of $N \in \{15, 20, 25\}$ using both the PATH and DG-SQP solvers. Convergence thresholds are again set to $\rho_1, \rho_2, \rho_3 = 10^{-4}$. The results are shown in Figure 6.5 where we first observe that for the exact dynamic game Γ_{kin}, our DG-SQP solver shows better, performance when compared against the PATH solver, achieving higher success rates for all three horizon lengths. However, it is clear that both solvers struggle with finding a solution for longer horizon dynamic games. We note from the first column of Figure 6.5, that for the exact dynamic game Γ_{kin}, the majority of the failure cases for the PATH and DG-SQP solvers occur at the entrance of the chicane. Due to the many tight turns, this section of the track is where the numerical shortcomings of the Frenet-frame dynamics, as described in Section 6.7, can be especially apparent. Next, we examine the performance of the solvers on the approximate dynamic game $\hat{\Gamma}_{\text{kin}}$. Here, we observe significant improvements in success rate for the DG-SQP solver over all horizon lengths, with a success rate of over 90% in the case of $N = 25$. This is an ∼5x improvement when compared to the success rate of solving the exact dynamic game Γ_{kin} with the same DG-SQP solver. When solving $\hat{\Gamma}_{\text{kin}}$ with the PATH solver, we see improvements over its performance with the exact dynamic game for the longer horizons of $N = 20$ and $N = 25$. However, we again see that DG-SQP achieves superior performance, with a >13x improvement compared to the PATH solver in the case of $N = 25$.

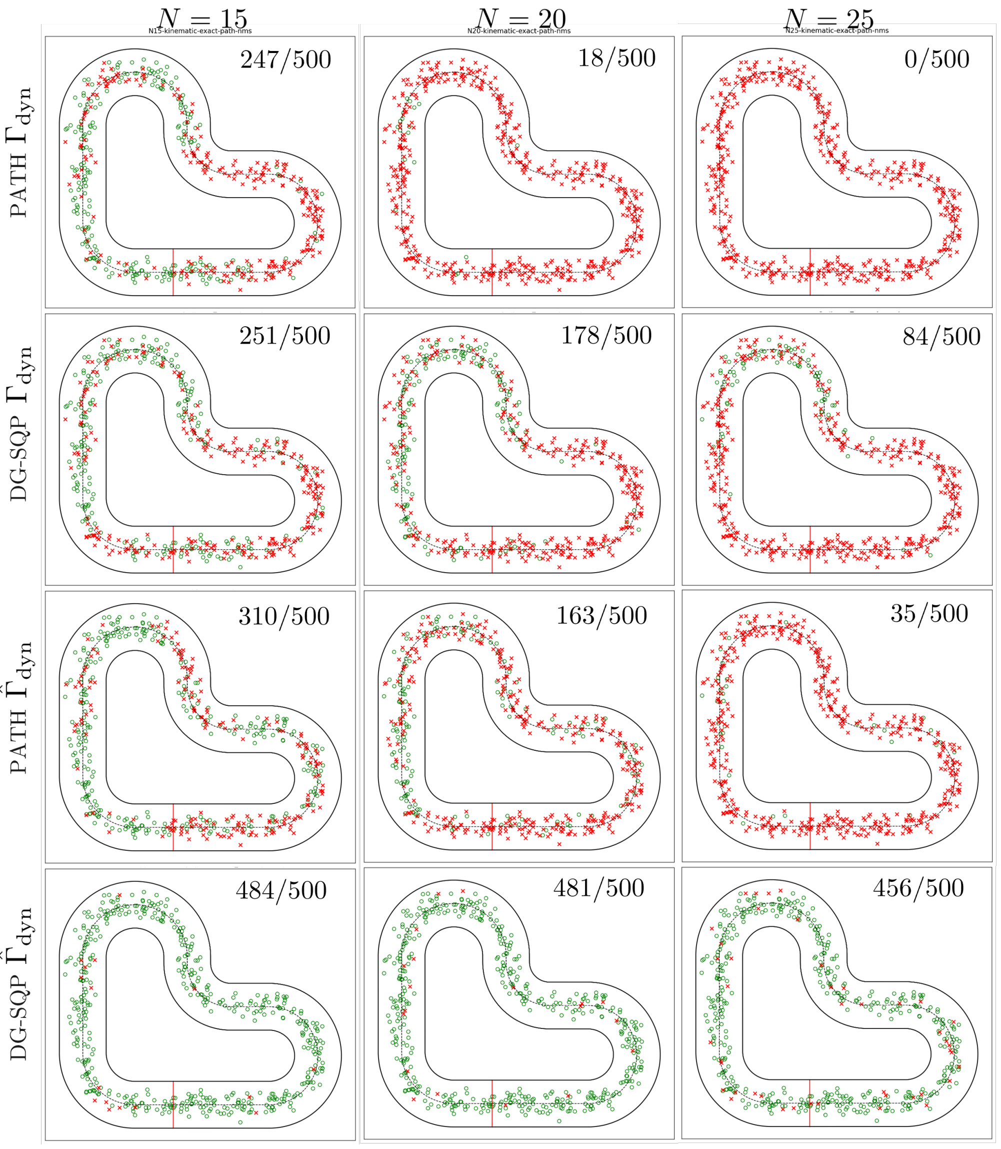


Figure 6.5: Results of the solver comparison study on the L-shaped track with the kinematic bicycle model from Scenario 1. Each point corresponds to the average of the sampled initial x-y positions for the two agents. A green $\circ$, and red $\times$ denote successful and failed trials respectively. The number in the top right of each plot shows the number of successful GNE solves out of the 500 sampled inital conditions.

Scenario 2

For this scenario, we randomly sample 500 collision-free initial conditions about a pre-computed race line (the red line in the left plot of Figure 6.2) where the agents start within 1.2 car lengths of each other and at velocities which are within ± 0.75 m/s of the race line velocity at the sampled s_0. We compare the success rates of solving for GNE from these initial conditions for the exact and approximate dynamic games Γ_{dyn} and $\hat{\Gamma}_{\text{dyn}}$. This is done again for the set of horizon lengths of $N \in \{15, 20, 25\}$. The results are summarized in

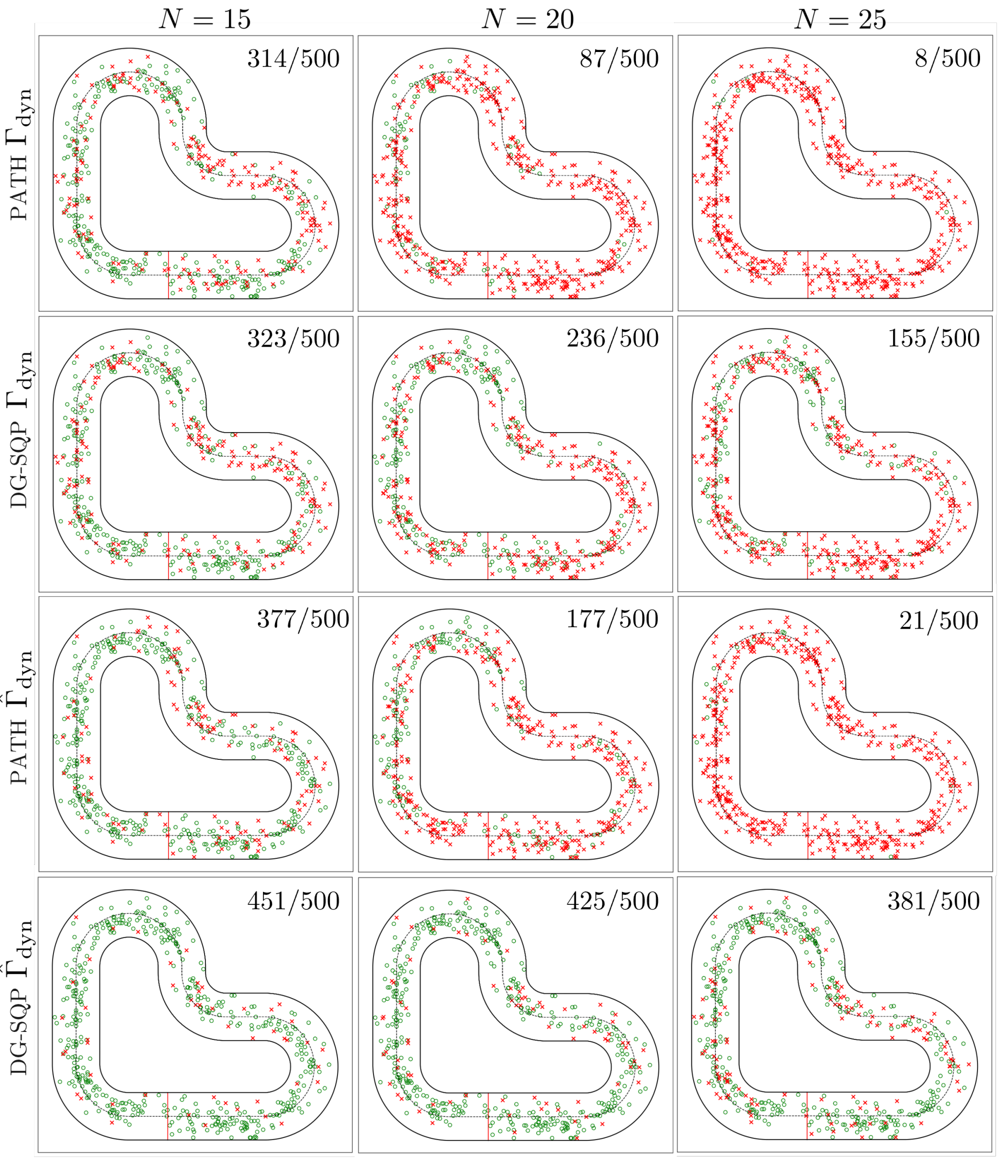


Figure 6.6: Results of the solver comparison study on the L-shaped track with the dynamic bicycle model from Scenario 2.

Figure 6.6, where we observe similar trends to the results from Scenario 1 in the case of the exact dynamic game Γ_{dyn}, which shows that our DG-SQP solver achieves similar or superior performance to PATH for all three values of horizon lengths. Turning our attention now to the approximate dynamic game, we again see that our DG-SQP solver achieves significantly higher success rates when solving $\hat{\Gamma}_{\text{dyn}}$, with an over 2x improvement over the case of Γ_{dyn} with DG-SQP and $N = 25$.

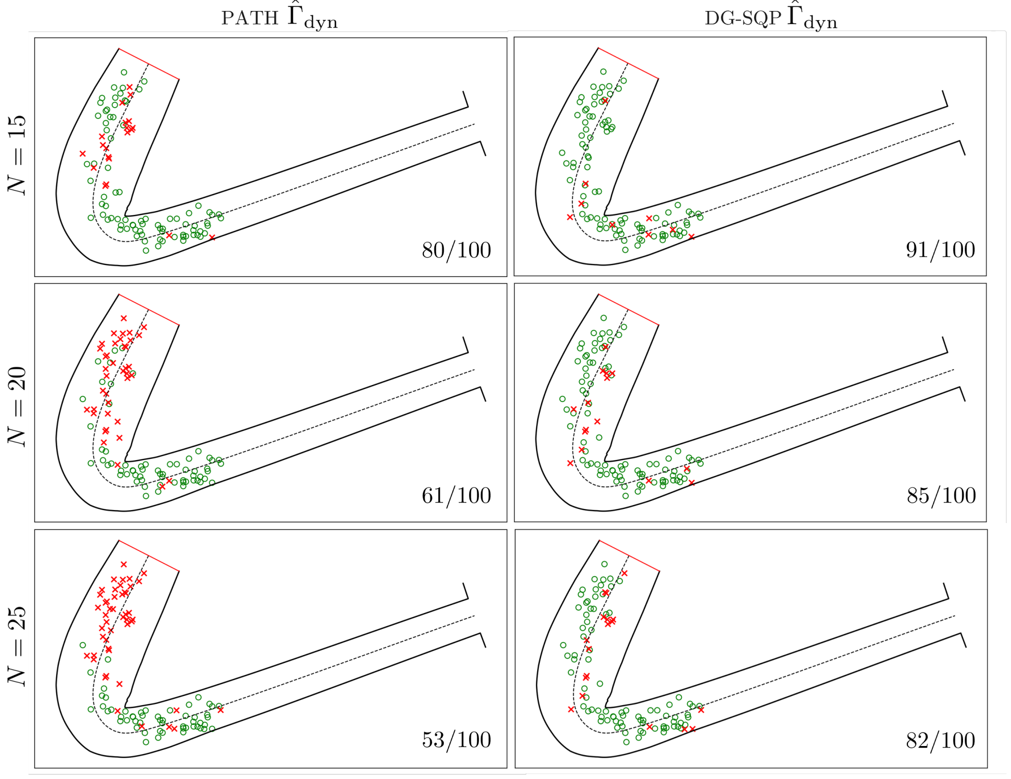


Figure 6.7: Results of the solver comparison study on the Austin F1 track from Scenario 3. The number in the bottom right of each plot shows the number of successful GNE solves out of the 100 sampled inital conditions.

Scenario 3

Using a similar procedure to Scenario 2, we randomly sample 100 joint initial conditions about a precomputed raceline (the red line in the right plot of Figure 6.2) where the agents start within 3 car lengths of each other and at velocities which are within ± 7.5 m/s of the race line velocity at the sampled s_0. The success rates of solving for GNE of the approximate dynamic game $\hat{\Gamma}_{\text{dyn}}$ from these initial conditions are shown in Figure 6.7, where we observe that, in the case of $N = 25$, our DG-SQP solver is able to achieve a success rate of 82% of the sampled initial conditions and out-performs the PATH solver by over 50%.

Comparing GNE From the Exact and Approximate Dynamic Games

We have just shown using three scenarios that significantly higher success rates can be achieved when solving for GNE of the approximate dynamic games $\hat{\Gamma}_*$ with the DG-SQP solver. However, it should be clear that as our approximation scheme modifies the components of the exact dynamic game Γ_*, we are therefore solving for the GNE of a related, but nevertheless different dynamic game. The question that naturally arises is then how the

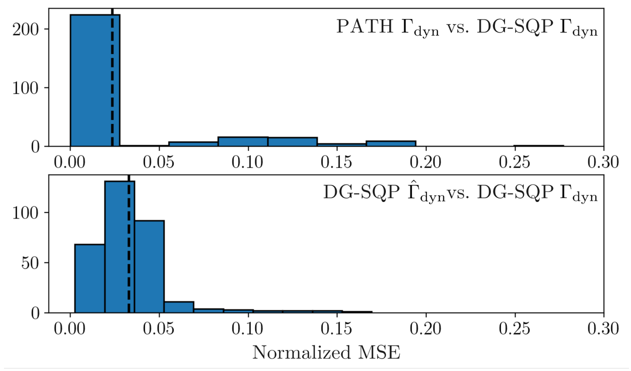


Figure 6.8: Distribution of the normalized mean squared error (MSE) between the GNEs from the three approaches. The dashed black line corresponds to the mean.

GNEs from the approximate dynamic game $\hat{\Gamma}_*$ compare with those from the exact one Γ_*. In order to answer this question, we now examine the solutions from Scenario 2 with $N = 15$. Specifically, compute the pairwise mean squared error (MSE) between the GNE from two comparison cases. The first compares the GNE of Γ_{dyn} for the samples where both PATH and DG-SQP were successful. The normalized MSE for this case is defined as follows:

$$\frac{1}{N}\sum_{k=0}^{N}\sum_{i=1}^{M}\|u^{i,\star}_{DGSQP,k} - u^{i,\star}_{PATH,k}\|^2_{D^{-1}}.$$

The second case compares the GNE of Γ_{dyn} and $\hat{\Gamma}_{\text{dyn}}$ from DG-SQP, where we only consider samples where DG-SQP was able to successfully solve for GNE of both the exact and approximate dynamic games. The normalized MSE for this case is defined as follows:

$$\frac{1}{N}\sum_{k=0}^{N}\sum_{i=1}^{M}\|\hat{u}^{i,\star}_{DGSQP,k}[1:2] - u^{i,\star}_{DGSQP,k}\|^2_{D^{-1}},$$

where $D = \text{diag}(u_u)$ normalizes the errors using the input upper bound u_u from (6.42a) and $\hat{u}^{i,\star}_{DGSQP,k}[1:2]$ denotes the first two elements of the GNE solution at time step k (recall that $\hat{u}$ is the augmented input vector which includes the approximate arc speed $\bar{v}$). The distribution of these errors are shown in Figure 6.8, where we may first observe from the top histogram that, when solving the exact Frenet-frame dynamic game, the DG-SQP and PATH solvers, for the most part, produce the same solutions with a minimum, median, and mean MSE of 8.71×10^{-8}, 1.17×10^{-5}, and 2.36×10^{-2} respectively. This serves as a baseline to illustrate the correctness of our approach. Looking at the bottom histogram, we see that although there are certainly outliers, the majority of the GNE from the approximate dynamic game are close to those from the exact dynamic game when the same solver is used. In this comparison case, we observe a minimum, median, and mean MSE of 2.54×10^{-3}, 3.07×10^{-2}, and 3.30×10^{-2} respectively.

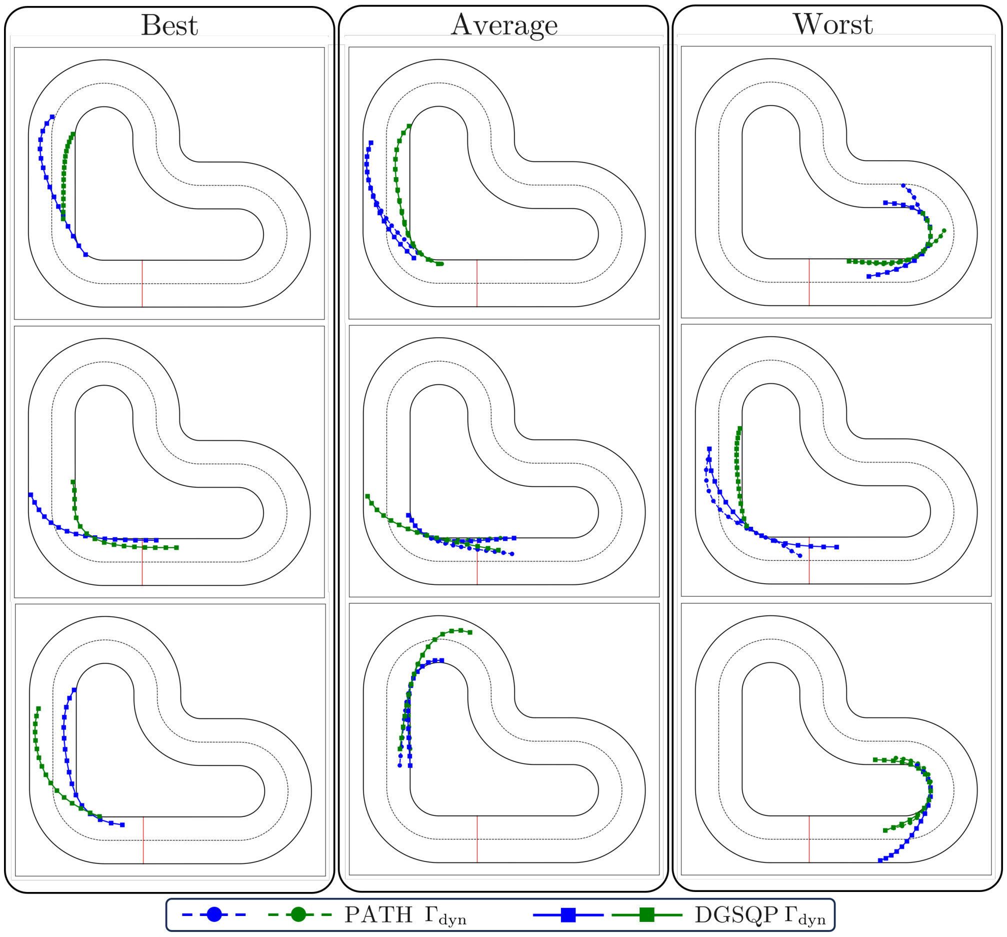


Figure 6.9: Comparison of GNEs of the exact dynamic game from the DG-SQP and PATH solvers. The blue and green traces represent the GNE trajectories of agents 1 and 2 respectively. The three plots in each column correspond to the top three cases in that category.

We now turn to specific examples from the histograms to examine the qualitative differences in the GNE from the two comparison cases in Figures 6.9 and 6.10. In particular, we visualize the three GNE pairs with the smallest MSE (“Best" column), largest MSE (“Worst" column), and the three pairs with a MSE that is closest to the mean value (“Average" column).

In Figure 6.9, we first compare the position traces arising from the GNE of the exact dynamic game Γ_{dyn} from the DG-SQP and PATH solvers. In the “Best” column, we observe that the GNE from the two solvers are are identical. We note that in this comparison case, these examples are indicative of the majority of successful GNEs as evidenced by the minimum and median MSEs. In the “Average” and “Worst” columns, we observe that some differences arise especially towards the end of the game horizon. These instances correspond to the solvers finding different local GNE. However, we note that in all of the “Average” and “Worst” examples, the outcome, i.e. the relative ordering of the two vehicles at the end of

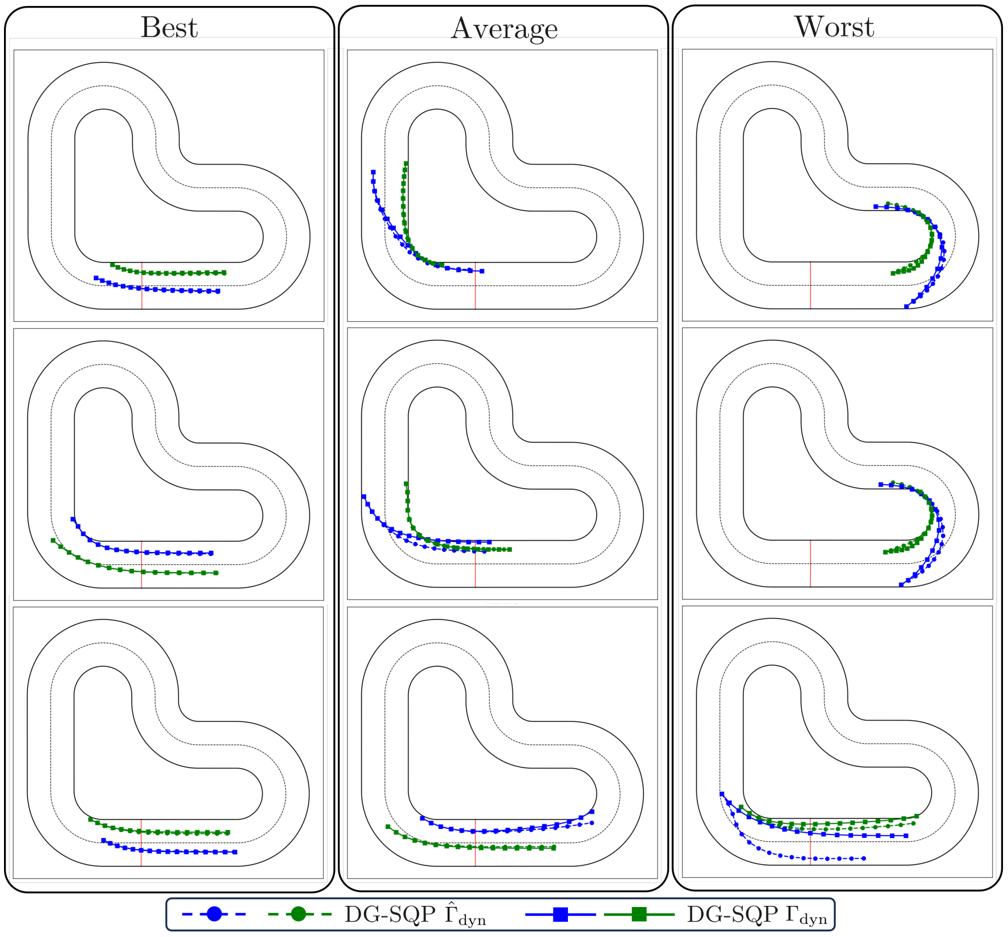


Figure 6.10: Comparison of GNEs of the approximate and exact dynamic games from the DG-SQP solver.

the game horizon, is the same.

In Figure 6.10, we compare GNE obtained with the DG-SQP solver for the exact and approximate dynamic games. We first observe that in the "Best" column, the GNE of the approximate and exact dynamic games are identical. The fact that this occurs on a straight section of the track is unsurprising as this is where $\bar{s}$ can most accurately approximate the true path progress s through the lag error in (6.37). To see this, simply evaluate (6.37) with $\Phi = 0$, which leads to $e_l(p, \bar{s}) = \tau_x(\bar{s}) - x$. From the "Average" column, we see that differences between the GNE are minor and that importantly, the GNE of the approximate dynamic game still capture the competitive nature of the racing scenario. Furthermore, we note that as in the previous comparison, the outcome remains the same despite small perturbations to the GNE trajectories between the exact and approximate dynamic games. In the "Worst" column, we may observe a reversal in outcome in the top two plots where the leading vehicle at the end of the game horizon is different between the two solutions. However, we note that such examples are rare as evidenced by the long tail of the histograms in Figure 6.8.

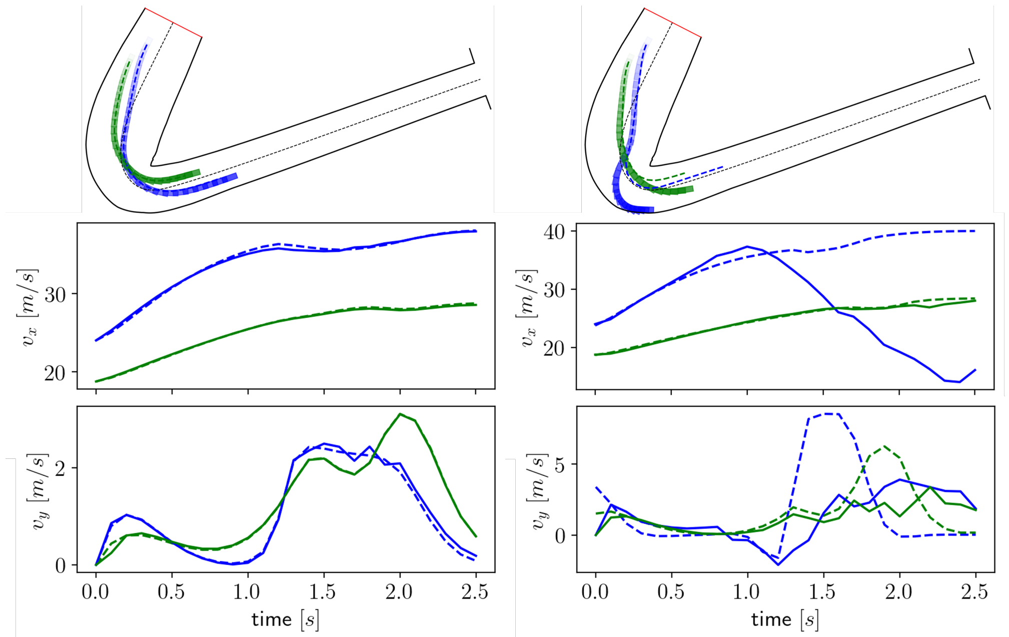


Figure 6.11: Results of tracking the GNE computed using a dynamic (left) and kinematic (right) bicycle model with an MPC policy. In the top plots, darker rectangles correspond to the vehicle's pose later in time. In all plots, the dashed lines represents the GNE reference and the solid lines represent the actual tracking performance of the vehicle.

Tracking Feasibility of Dynamic and Kinematic GNE

We finally show, through an example in Figure 6.11, the importance of higher fidelity models in the solution of GNE for racing scenarios. In particular, we take GNE computed with kinematic and dynamic bicycle models from the same initial conditions on the Austin F1 track from Scenario 3 and attempt to track the solutions using a MPC policy in a manner similar to [145] and [143]. As the true system is modeled using a dynamic bicycle model, we first see in the left plots of Figure 6.11 that it is able to track the dynamic GNE accurately. On the other hand, if the kinematic GNE is used as a reference, we observe significant mismatch between the reference and actual trajectories. This is due to the lack of tire forces in the kinematic model which results in an overestimation of cornering performance of the vehicles. This is especially evident in the trajectory of the blue vehicle, where the dynamic infeasibility of the reference induces poor numerical performance in the QP solver of the MPC policy. This actually results in a reversal of outcome, where instead of being overtaken by the blue vehicle (as was predicted by the GNE), the green vehicle is able to maintain its lead over the blue vehicle at the end of the game horizon.

6.9 Chapter Summary

In this chapter, we have presented DG-SQP, an SQP approach to the solution of generalized Nash equilibria for open-loop dynamic games. We show that the method exhibits local linear convergence in the neighborhood of GNE and present several practical improvements to the vanilla SQP algorithm including a non-monotone line search strategy with a novel merit function and decaying regularization scheme. We further present an approximation scheme to Frenet-frame dynamic games which can be used to improve the performance of our solver in racing scenarios. We conducted an extensive numerical study on various head-to-head racing scenarios with both kinematic and dynamic vehicle models and different race tracks. In particular, we measured the performance of our DG-SQP solver against the state-of-the-art PATH solver using the metric of success rate and showed significant improvements in racing scenarios especially when approximate dynamics are used. Another metric which we have not discussed is that of solution time. This is especially important if we would like to use our DG-SQP solver in a real-time MPGP manner like in [90] and [104]. However, we note that main shortcoming of our approach at this time is computational efficiency, with solutions taking up to two minutes for some of the successful trials with $N = 25$ in Scenario 3. This is due to two reasons. The first is that our solver requires the solution of a sequence of constrained quadratic programs, which can be computationally demanding especially when compared to the PATH solver, which at its core, solves a sequence of linear systems. Secondly, our solver is implemented in Python, which is again significantly slower than the C++ implementation of the PATH solver. Though we can certainly implement our solver in a different language, we anticipate that it would still be difficult to achieve real-time solutions for the long-horizon racing problems presented here due to the solution of QPs. As such, an important direction of future work is to improve the computational efficiency of our DG-SQP solver, which would allow us to leverage its solutions for real-time racing experiments. One possible approach could be to improve the quality of the initial guess through supervised learning of GNE as in [146] and [106]. By warm starting the solver with the a neural network, which predicts GNE for a given joint initial condition, this would potentially reduce the number of solver iterations required to reach convergence. An alternative approach would be to leverage the numerical robustness of our DG-SQP solver to generate an extensive dataset of GNE from various initial conditions and track environments. This data could then be used to learn a game-theoretic value function which can be integrated in an optimal control scheme to aid in long-term strategic planning without the need for online solutions from our solver. We will explore this idea in the next chapter.

Chapter 7

Strategy Learning for Autonomous Racing using Game-Theoretic Equilibria

7.1 Introduction

In this chapter, we continue our investigation into the use of game-theoretic methods for prediction and planning in autonomous racing. Previously in Chapter 6, we had introduced DG-SQP, which was a solver for generalized Nash equilibria (GNE) of open-loop dynamic games. Through the introduction of a merit function, nonmonontone line search, and decaying regularization scheme in the step selection procedure for the solver, we showed that our approach was able to out-perform the state of the art PATH solver for mixed complementarity problems, which are a super class of the GNE finding problem for open-loop dynamic games, in terms of success rate. Now, we would like to leverage the GNE from our solver for the real-time control of an autonomous race car in a head-to-head race. Broadly speaking, there are two complementary approaches to this problem, namely in the *tactical* and *strategic* utilization of GNE for prediction and/or planning. In tactical approaches, such as those employed in [129, 123, 88], GNE are used directly to provide both a prediction of ones opponents and a motion plan for the ego agent over the planning horizon. Therefore, solve rates between 1 to 10 Hz are typically required of the GNE solver in order for these solutions to be used either for real-time control [122] or as a high-level plan for a tracking policy [129]. In strategic approaches, such as the one presented in [50], GNE are used to estimate the value of long-term behavior beyond the planning horizon, which importantly can be performed offline, thereby removing the real-time requirement for the solver. We will discuss the distinctions between the two approaches in greater detail in Chapter 8. As was discussed in Section 6.9, one drawback of our current implementation of DG-SQP is its high computational burden. This limits its applicability in one of the aforementioned tactical approaches. However, due to our emphasis on the robustness of the solver, we believe that it is particularly well-suited for the class of strategic approaches which benefit from data generated under a wide range of dynamic game descriptions and initial conditions. As such, we now present our strategic

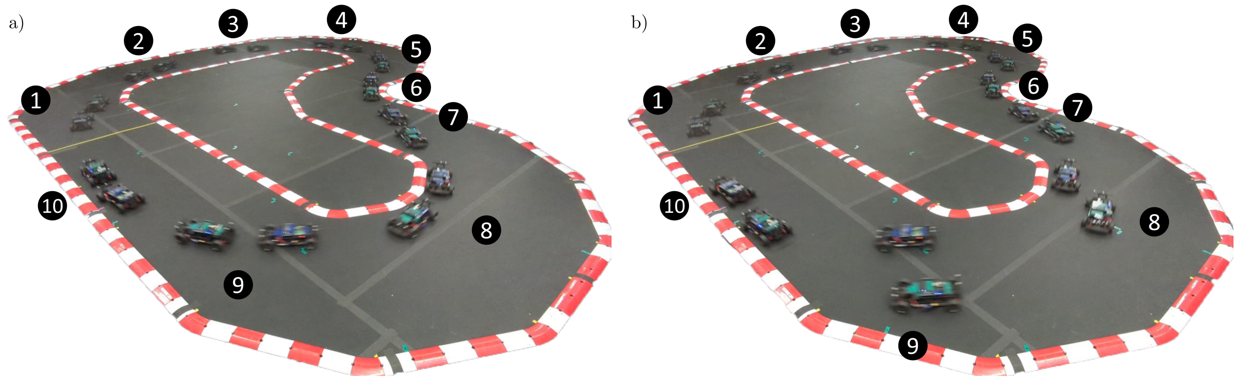


Figure 7.1: Time-lapse of head-to-head races on a 1/10th scale RC car platform where a) The green car uses MPC with a game-theoretic value function to successfully defend against the blue car. b) The blue car uses the same to successfully overtake the green car. The numbers indicate the order of the video frames, which are equally spaced in time.

approach, which leverages the GNE solutions from DG-SQP to learn a game-theoretic value function which can be embedded in a real-time model-predictive control (MPC) scheme to guide the solution of short-horizon optimal control problems. In particular, we aim to endow the racing agent with the ability to reason about long-term *strategic* value through game-theoretic state space equilibria. Our results show that by leveraging this reasoning in a MPC scheme, we observe the emergence of competitive behaviors in car racing such as position defense and overtaking at the limit of tire friction as shown in Figure 7.1. The contributions of this chapter are as follows:

- A strategy learning approach which constructs a value function estimate using GNE solutions from our DG-SQP solver.
- The design of a racing MPC policy which leverages the learned value function as a terminal cost-to-go function.
- Experiments in simulation and hardware which demonstrate the performance of the MPC policy in closed-loop with the learned value function as compared to a progress maximization baseline.

7.2 Problem Formulation

Consider the head-to-head racing problem on the L-shaped track shown in Figure 7.2 where we describe the motion of the racing agents using the Frenet frame dynamic bicycle model in (A.2) which is discretized with a time step of $\Delta t = 0.1s$. In particular, for Agents 1 and 2, with which we will associate the colors blue and green respectively, we denote their states and inputs as $x^i = [v_x, v_y, \omega, e_\psi, s, e_y]^\top$ and $u^i = [a_x, \delta]^\top$ for $i = 1, 2$. The agent dynamics functions are assumed to be identical, i.e. $f^1 = f^2 = f$. Like in Chapter 6, the race track is represented as a parametric path $\tau : [0, L] \mapsto \mathbb{R}^2$, which is assumed to be twice differentiable,

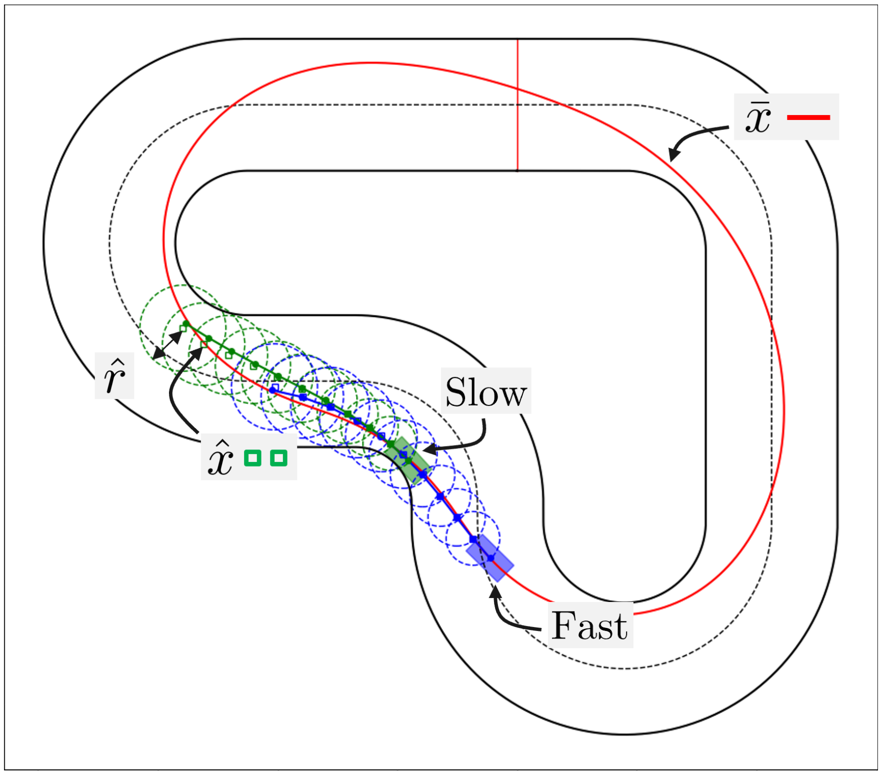


Figure 7.2: Illustration of the head-to-head racing problem.

of length L, and of constant width W. The definition of the Frenet frame pose (s, e_y, e_ψ) is given in (6.22). The agents are each subject to identical private state and input constraints

$$-W/2 \leq e_y^i \leq W/2, \tag{7.1a}$$
$$u_l \leq u^i \leq u_u, \tag{7.1b}$$
$$\Delta u_l \leq \Delta u^i \leq \Delta u_u, \tag{7.1c}$$
$$a^f(x^i, u^i), a^r(x^i, u^i) \leq \mu g \tag{7.1d}$$

where (7.1a) are the track boundary constraints, (7.1b) and (7.1c) denote magnitude and rate limits on the control inputs, and (7.1d) denotes friction circle constraints on the front and rear tires for the tire-road friction coefficient μ. Here, a^f and a^r are functions which return the magnitude of the acceleration experienced by the front and rear tires respectively for a given state and input. Finally, each agent imposes circular collision avoidance constraints with respect to the prediction of their opponent:

$$\|p(x_k^i) - \hat{p}_k\|_2^2 \geq (r^i + \hat{r}_k)^2 \tag{7.2}$$

where $p(x_k^i)$ is the function defined in (6.43) which performs the transformation of the agent's Frenet frame position coordinates (s, e_y) into the inertial frame (x, y). r^i is the radius of the collision buffer for agent i. $\hat{p}_k$ and $\hat{r}_k$ are the predicted position of the opponent for agent i and the radius of the collision buffer for that corresponding prediction at time step k. Finally, for each agent $i = 1, 2$, we assume that a pre-computed racing line for the track, i.e. a state reference trajectory $\bar{x}^i(l) : [0, T] \mapsto \mathbb{R}^6$ has been made available, where $T > 0$ is the lap time associated with the racing line. This is shown as the red curve in Figure 7.2. Now, given a state x_k^i, we may obtain the N-step racing line reference at time step k for agent i as

$$\bar{\mathbf{x}}_k^i = \{\bar{x}^i(l^\star), \bar{x}^i(l^\star + \Delta t), \dots, \bar{x}^i(l^\star + N\Delta t)\} \tag{7.3}$$

where $l^\star = \arg\min_{l \in [0,T]} \|\bar{x}^i(l) - x_k^i\|_2^2$.

Race Setup

All races in simulation and hardware are initialized with Agent 1 (the blue car) trailing Agent 2 (the green car) by at least 3 car lengths and end after both agents finish a prescribed number of laps around the track. Just as human racing drivers will modify their driving behavior depending whether there are opponents in close proximity, we distinguish between a race line tracking mode and an interaction mode for the operation of the two agents. The race line tracking mode will be used by the agents when they are far from each other and there is no possibility of direct interactions. In this mode, the agents will ignore the presence of their opponent and simply focus on optimizing their own driving in terms of tracking the pre-computed race line. The interaction mode will be used when the agents are within a certain distance of each other, where for each agent, the behavior of their opponent will have a direct effect on their own motion plan. In this mode, the agents must take care to avoid collisions, but will also engage in competition to achieve the best possible race outcome. The transition from the race line tracking mode to the interaction mode occurs when the trailing car closes its gap with the leading car to within 3 car lengths. The cars revert to race line tracking upon a successful overtake by the trailing Agent 1 or if Agent 1 falls behind Agent 2 by more than 3 car lengths.

Race Line Tracking MPC Policy

We now define the path tracking optimal control problem for agent i which will be used by both agents when operating in the race line tracking mode:

$$\begin{aligned}
\mathbf{u}_k^{i,\star}(x_k^i, \bar{\mathbf{x}}_k^i) = \arg\min_{\mathbf{u}^i} \quad & \sum_{t=0}^{N-1} \|x_t^i - \bar{x}_{k+t}^i\|_{Q^i}^2 + \|u_t^i\|_{R_1^i}^2 + \|\Delta u_t^i\|_{R_2^i}^2 && (7.4a) \\
\text{subject to} \quad & x_0^i = x_k^i, && (7.4b) \\
& x_{t+1}^i = f^i(x_t^i, u_t^i),\ k = 0, \dots, N-1, && (7.4c) \\
& x_l \le x_t^i \le x_u,\ k = 0, \dots, N, && (7.4d) \\
& u_l \le u_t^i \le u_u,\ \Delta u_l \le \Delta u_t^i \le \Delta u_u,\ k = 0, \dots, N-1, && (7.4e) \\
& a^f(x_t^i, u_t^i), a^r(x_t^i, u_t^i) \le \mu g,\ k = 0, \dots, N-1. && (7.4f)
\end{aligned}$$

We denote the solution to (7.4) as $\mathbf{u}_k^{i,\star}(x_k^i, \bar{\mathbf{x}}_k^i)$ which, at time step k, is a function of the state x_k^i and the race line reference $\bar{\mathbf{x}}_k^i = \{\bar{x}_k^i, \bar{x}_{k+1}^i, \dots, \bar{x}_{k+N}^i\}$, which was defined in (7.3). The stage costs in (7.4a) are comprised of tracking, input, and input rate penalties with $Q^i, R_1^i, R_2^i \succeq 0$. (7.4b) and (7.4c) are the initial condition and dynamics constraints respectively and the remainder of the constraints correspond those described earlier in this section. Note that we do not impose the collision avoidance constraint defined in (7.2) since this policy is only used when the agents are greater than three car lengths apart where there is no possibility of collision. In typical receding horizon fashion, the tracking policy applies the first element of $\mathbf{u}_k^{i,\star}$ to the system of agent i at time k, then resolves (7.4) at the next time step.

7.3 Learning a Game-Theoretic Value Function

In this section, we describe a simple technique for learning a value function for our head-to-head racing task using the GNE of an open-loop dynamic game. We define the dynamic game via the following coupled optimal control problems:

$$\begin{aligned}
\mathbf{u}^{1,\star}(\mathbf{u}^{2,\star}) = \arg\min_{\mathbf{u}^1} \quad & \sum_{k=0}^{\bar{N}-1} \|u_k^1\|_{P_1^1}^2 + \|\Delta u_k^1\|_{P_2^1}^2 + b^1 s_{\bar{N}}^2(\mathbf{u}^{2,\star}) - a^1 s_{\bar{N}}^1(\mathbf{u}^1) && (7.5a)\\
\text{subject to} \quad & (r^1 + r^2) - \|p_k^1(\mathbf{u}^1) - p_k^2(\mathbf{u}^{2,\star})\|_2 \le 0,\ k = 0, \dots, \bar{N}, && (7.5b)\\
& -W/2 \le e_{y,k}^1(\mathbf{u}^1) \le W/2,\ k = 0, \dots, \bar{N}, && (7.5c)\\
& u_l \le u_k^1 \le u_u,\ k = 0, \dots, \bar{N}-1. && (7.5d)
\end{aligned}$$

$$\begin{aligned}
\mathbf{u}^{2,\star}(\mathbf{u}^{1,\star}) = \arg\min_{\mathbf{u}^2} \quad & \sum_{k=0}^{\bar{N}-1} \|u_k^2\|_{P_1^2}^2 + \|\Delta u_k^2\|_{P_2^2}^2 + a^2 s_{\bar{N}}^1(\mathbf{u}^{1,\star}) - b^2 s_{\bar{N}}^2(\mathbf{u}^2) && (7.6a)\\
\text{subject to} \quad & (r^1 + r^2) - \|p_k^1(\mathbf{u}^{1,\star}) - p_k^2(\mathbf{u}^2)\|_2 \le 0,\ k = 0, \dots, \bar{N}, && (7.6b)\\
& -W/2 \le e_{y,k}^2(\mathbf{u}^2) \le W/2,\ k = 0, \dots, \bar{N}, && (7.6c)\\
& u_l \le u_k^2 \le u_u,\ k = 0, \dots, \bar{N}-1. && (7.6d)
\end{aligned}$$

Note that the horizon $\bar{N}$ of the dynamic game does not have to be the same as the horizon of the optimal control problem N. In practice, we choose $\bar{N} = 25$.

As previously mentioned, tactical approaches to learning with GNE of dynamic games for racing typically attempt to replicate the GNE over the planning horizon. This can be thought of as a prediction of the *trajectory* outcome of the dynamic game where for some initial condition x_0^1 and x_0^2, we obtain a prediction in the form $\hat{\mathbf{u}}^1$ and $\hat{\mathbf{u}}^2$ which should ideally be close to the GNE $\mathbf{u}^{1,\star}$ and $\mathbf{u}^{2,\star}$. In contrast, our strategic approach attempts to learn the *reward* outcome of the dynamic game where instead of the entire trajectory, we would like to predict a scalar representation of the value or benefit of being at x_0^1 and x_0^2 given that the agents behave rationally. In particular, since it is straightforward in racing to use the progress difference as a way to measure the relative standing of two agents, we propose to use the following learning target computed from the GNE solution of the dynamic game with initial conditions x_0^1 and x_0^2

$$s_N^1(\mathbf{u}^{1,\star}) - s_N^2(\mathbf{u}^{2,\star}). \tag{7.7}$$

This target is associated with the feature vector

$$\text{vec}(x_0^1, x_0^2 - x_0^1). \tag{7.8}$$

Here we are treating Agent 1 as the ego agent, which means that we are predicting the progress advantage that Agent 1 will have over Agent 2. We may also construct features and targets from the perspective of Agent 2 by swapping the superscripts in (7.8) and (7.7). The dataset generation procedure then involves 1. sampling of initial conditions x_0^1 and x_0^2, 2. solving for a corresponding GNE of the dynamic game, and 3. the construction of (7.8) and (7.7) which are added to the dataset.

With this dataset, we can then solve a simple regression problem to obtain the game-theoretic value function $\Delta s \approx V_{GT}(x, \hat{x})$, which estimates the difference in track progress Δs for a given ego state x and opponent state $\hat{x}$. We model this function as an MLP with two hidden layers of size 48 and `tanh` activation function. Training is done on a dataset of $\sim$1800 GNE solutions from joint initial conditions that were sampled from the L-shaped track in Figure 7.2. We use the Adam optimizer with a scheduled learning rate initialized at 1e-3 and stop training if no improvement is observed after 10 epochs.

7.4 Closed-Loop Head-to-Head Racing Policy

With the learned game-theoretic value function V_{GT}, we may now introduce the MPC policy which will be used during the interactive periods of the race, i.e. when the two cars are within 3 car lengths of each other. The optimal control problem is defined as follows:

$$\begin{aligned}
\min_{\mathbf{u}^i} \quad & \sum_{t=0}^{N-1} \|x_t^i - \bar{x}_{k+t}^i\|_{Q^i}^2 + \|u_t^i\|_{R_1^i}^2 + \|\Delta u_t^i\|_{R_2^i}^2 - V_{GT}(x_N^i, \hat{x}_{k+N}) && \text{(7.9a)}\\
\text{subject to} \quad & x_0^i = x_k^i, && \text{(7.9b)}\\
& x_{t+1}^i = f^i(x_t^i, u_t^i),\ k = 0, \dots, N-1, && \text{(7.9c)}\\
& x_l \leq x_t^i \leq x_u,\ k = 0, \dots, N, && \text{(7.9d)}\\
& u_l \leq u_t^i \leq u_u,\ \Delta u_l \leq \Delta u_t^i \leq \Delta u_u,\ k = 0, \dots, N-1, && \text{(7.9e)}\\
& a^f(x_t^i, u_t^i), a^r(x_t^i, u_t^i) \leq \mu g,\ k = 0, \dots, N-1 && \text{(7.9f)}\\
& \|p(x_t^i) - \hat{p}_{k+t}\|_2^2 \geq (r + \hat{r}_{k+t})^2,\ k = 0, \dots, N, && \text{(7.9g)}
\end{aligned}$$

where one can quickly observe that it is largely similar to the race line tracking policy defined in (7.4). The differences are the addition of the game-theoretic value function as the terminal cost function in (7.9a) and the collision avoidance constraint in (7.9g), which is a function of the opponent prediction $\hat{\mathbf{z}}_k = \{\hat{z}_k, \hat{z}_{k+1}, \dots, \hat{z}_{k+N}\}$, where $\hat{z}_{k+t} = (\hat{x}_{k+t}, \hat{r}_{k+t})$ denotes a nominal state and collision buffer radius of the opponent at time step $k + t$ for $t = 0, \dots, N$. We note that since our goal is to demonstrate the impact of the value function, for simplicity, we generate $\hat{\mathbf{z}}_k$ using an oracle but corrupt it with Gaussian noise. We use time-varying collision buffer radii, i.e. $\hat{r}_k < \hat{r}_{k+1} < \dots < \hat{r}_{k+N}$, to capture robustness against growing prediction uncertainty. Of course, this could be replaced by any number of existing trajectory prediction schemes, though we defer an investigation of their impact on racing performance to future work.

In the following simulation and hardware races, we denote the vehicle using (7.9) as GT and we compare against a non-game-theoretic optimal control policy which is identical to (7.9) in all but the terminal cost function. Namely, we replace V_{GT} with a maximum progress value function

$$V_{MP}(x, \hat{x}) = s(x) - s(\hat{x}), \tag{7.10}$$

where the function $s(x)$ extracts the element of the state vector corresponding to the track progress s. Of course, since the opponent prediction is treated as a parameter in (7.9), $s(\hat{x})$ is a constant and therefore has no effect on the optimal solution. However, we keep the value function in this form in order to make a direct comparison to V_{GT}, which, as described in Section 7.3, predicts the progress advantage of the ego vehicle against its opponent. We denote the vehicle using this baseline as MP.

7.5 Results

We now present the results of our strategic learning approach. In particular, we will first examine the reward outcome predictions of the learned game-theoretic value function in isolation to identify the high-value regions of the state space. We will then show the results of head-to-head races between agents using the GT and MP policies as defined in Section 7.4.

Open-Loop Results

Let us first examine the learned game-theoretic value function $V_{GT}(x, \hat{x})$ by evaluating it over a grid of ego state values and for a set of opponent states. Each of the plots shown in Figure 7.3 correspond to an evaluation of $V_{GT}(x, \hat{x})$ over a grid of positions states in x and for a fixed $\hat{x}$. The rows of the two figures correspond to different settings of the lateral position state e_y for $\hat{x}$. The columns correspond to different settings of longitudinal velocity for x, where positive values of Δv indicate that the longitudinal velocity for x is greater than that of $\hat{x}$. Note that in both figures, we chose the longitudinal velocity of $\hat{x}$ to be 2.5 m/s. In Figure 7.3, we would like to highlight two visual elements which are important in

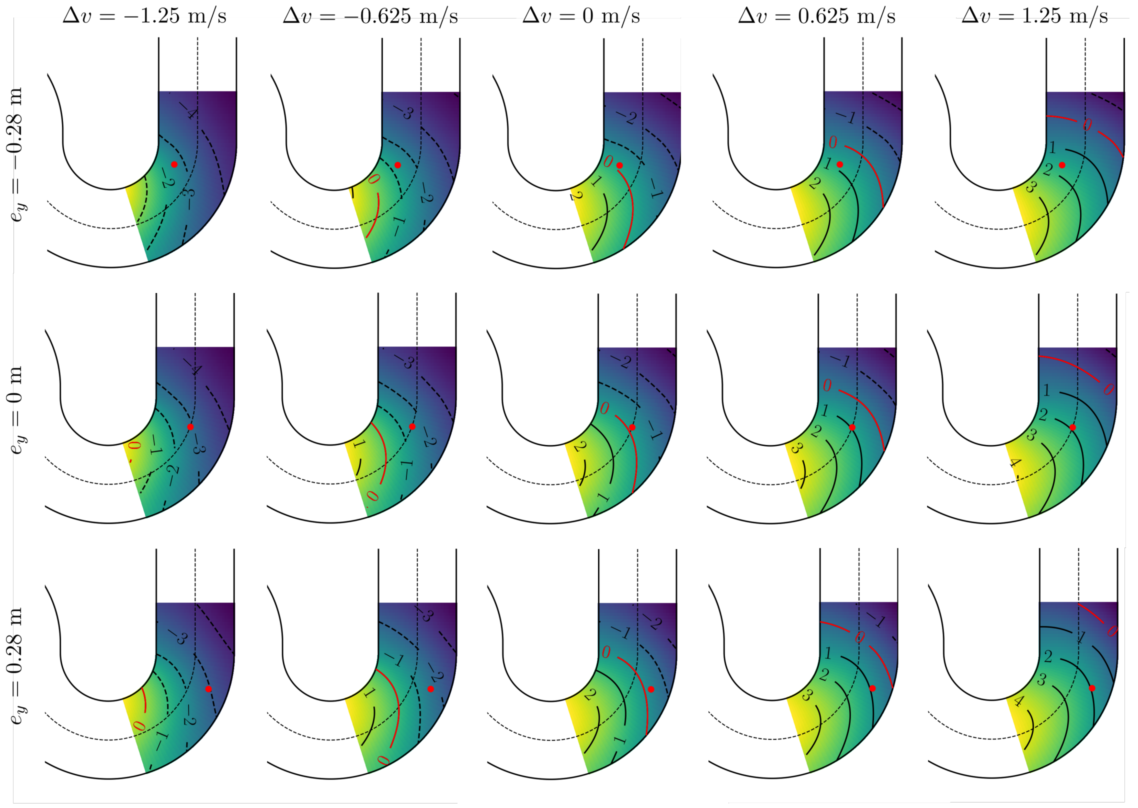


Figure 7.3: Evaluation of the learned game-theoretic value function $V_{GT}(x, \hat{x})$ for different values of lateral position e_y (illustrated by the red dot in each plot) and longitudinal velocity difference Δv.

understanding outputs of the value function. The first is the red dot in each plot which denotes the position of $\hat{x}$. This can be thought of as the position of the opponent and we center our evaluation of $V_{GT}(x, \hat{x})$ about this position. The second is the red contour line which is a projection of the level set $\{x \mid V_{GT}(x, \hat{x}) = 0\}$ (given a constant $\hat{x}$) on to the position states. This line can be thought of as the "tie" line which indicates the set of positions for an ego vehicle where the progress difference between the two agents is predicted to be zero.

In Figure 7.3, we show the dependency of the reward predictions from V_{GT} on the lateral position of the opponent and the velocity difference between the two agents. First, we can see that as a general trend, for positions which have the same s coordinate, the value function predicts a greater reward for positions closer to the inside of the turn. This preference for inside lateral positions aligns with standard car racing strategy as many head-to-head battles between drivers are determined by whomever attains control of these crucial parts of the track. Now, let us examine the rows of plots where the lateral position of the opponent is fixed. Going from left to right, we see that as the velocity difference increases, the "tie" line shifts backwards which indicates an enlargement of the set of states where the ego vehicle is predicted to end up ahead of its opponent. Unsurprisingly, this means that the value function shows preference for greater velocity differences which are another key ingredient of

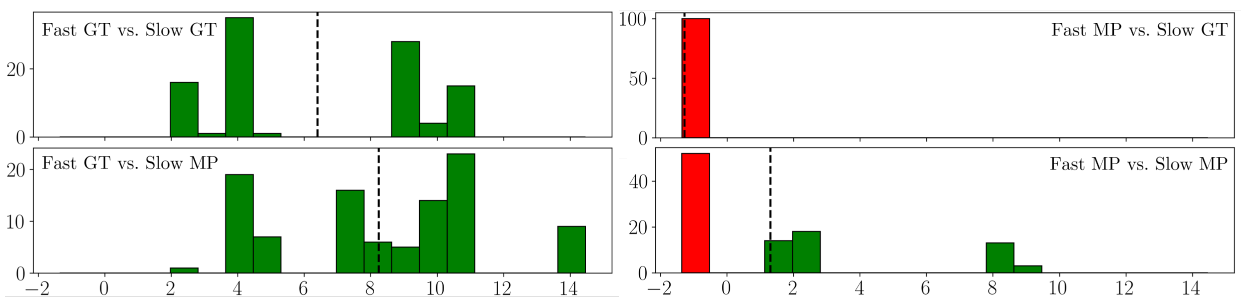


Figure 7.4: Histogram of the lead (in number of car lengths) of the faster agent over the slower agent at the end of the simulation races. Green bars indicate races where the faster agent wins, red bars indicate losses. The black dashed line indicates the average lead over all trials.

successful overtakes or position defenses in real-world car racing. Finally, let us look at the columns of the plots where the velocity difference is fixed. Going from top to bottom, we see that as the opponent's position goes moves towards the outside of the turn, the "tie" line also shifts backwards. This builds on top of the previously discussed preference in absolute lateral position and shows that, when entering into a turn, relative lateral positioning with respect to your opponent can have a significant effect on the predicted reward. In particular, the value function shows preference for positions that have the inside advantage over an opponent which, again is a strategy seen in real-world racing behavior.

Closed-Loop Results

We now move on to the results of head-to-head races which are set up according to Section 7.2. While both vehicles track pre-computed racelines of the same shape (i.e. the red line in Figure 7.5), similar to [123] and [143], we force the initially leading Agent 2 to be slower than the initially trailing Agent 1 by reducing its velocity profile by 10%. In the remainder, we refer to these two cars as "slow" and "fast" respectively. Finally, we formulate both the GT and MP optimal control problems using CasADi [7] with a horizon length of $N = 10$. These are solved using real-time iteration with the QP solver HPIPM [53].

Simulation Monte-Carlo Experiment

For our simulation study, we sample 100 joint initial conditions from the pre-computed raceline for the L-shaped track such that the two cars begin with a difference in track progress of 2m. For each initial condition, four scenarios are simulated. Namely, Fast GT vs. Slow GT, Fast GT vs Slow MP, Fast MP vs. Slow GT, and Fast MP vs. Slow MP, which indicate the assignment of the aforementioned MPC policies defined in (7.9) to each of the two cars. The outcomes of the races are summarized in Figure 7.4. From the results, we first note that when using GT, the faster car is able to win all races. However, there is a clear reduction in its average lead at the end of the race when the slower car also uses GT, which indicates that GT allows the slower car to defend its position more effectively

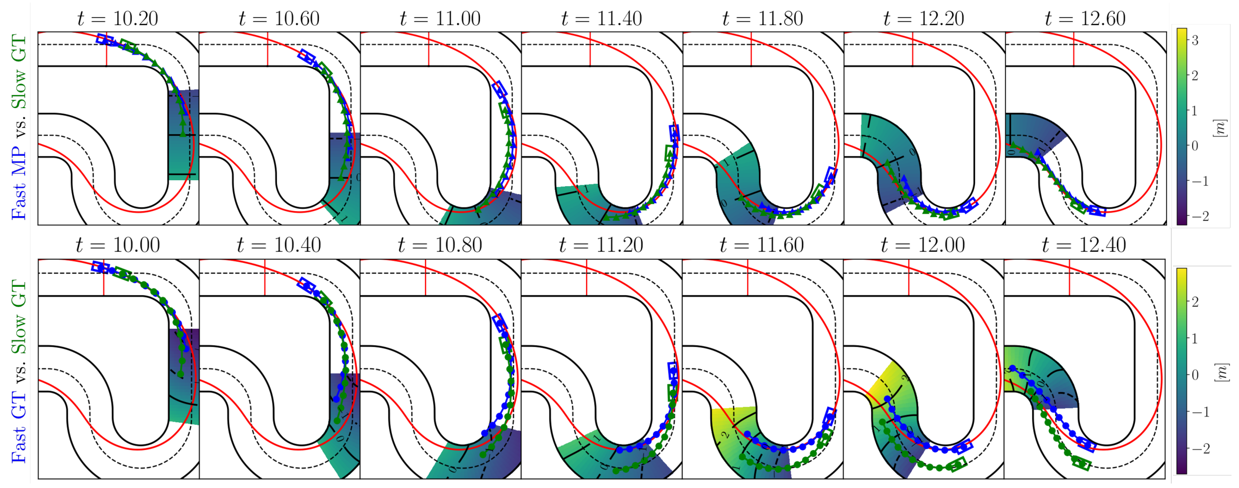


Figure 7.5: In a simulation experiment, the faster (blue) car with MP is unable to overtake the slower (green) car with GT (Top). The faster car with GT is able to overtake the slower car with GT (Bottom). The heatmaps and black level curves visualize the value functions used in this scenario by the faster (blue) car.

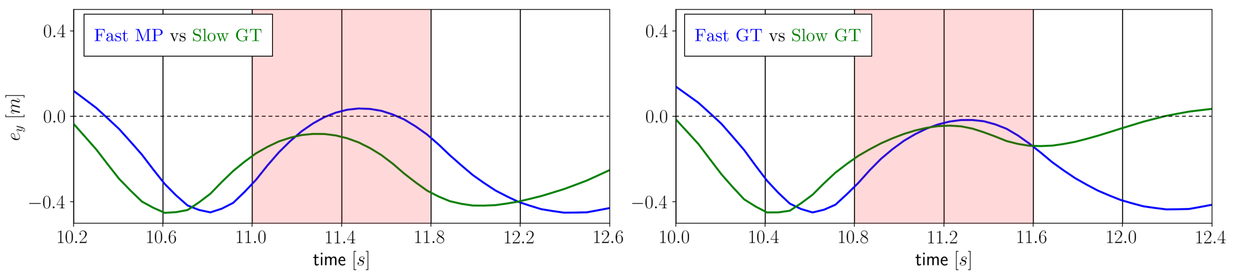


Figure 7.6: Lateral position traces from two racing scenarios. The vertical black lines correspond to the time stamps in Figure 7.5.

than MP by leveraging the game-theoretic value function. This is further emphasized in the case of Fast MP vs. Slow GT where the slower car is able to successfully defend its lead and win all races with the game-theoretic MPC policy. We include results from the Fast MP vs. Slow MP scenario as a baseline to show that when no game-theoretic component is included, the faster car has a win rate of approximately 50%. From these results, it is clear that the learned game-theoretic value function imparts a clear competitive advantage in both overtaking and position defense scenarios. Next, we will highlight the differences in how the GT and MP value functions guide the short-horizon MPC solution through two interactions in Figures 7.5 and 7.7 where we visualize the MPC plans and value function heatmaps.

In Figure 7.5, we show an overtaking attempt by the faster agent and where the slower agent uses the GT policy. The top row of frames shows the scenario when the faster agent uses the MP value function. While this value function directly encourages the agent to make progress along the track by maximizing s, it does not provide any guidance in terms of desireable lateral positions as all points with the same s coordinate are of equal value. On

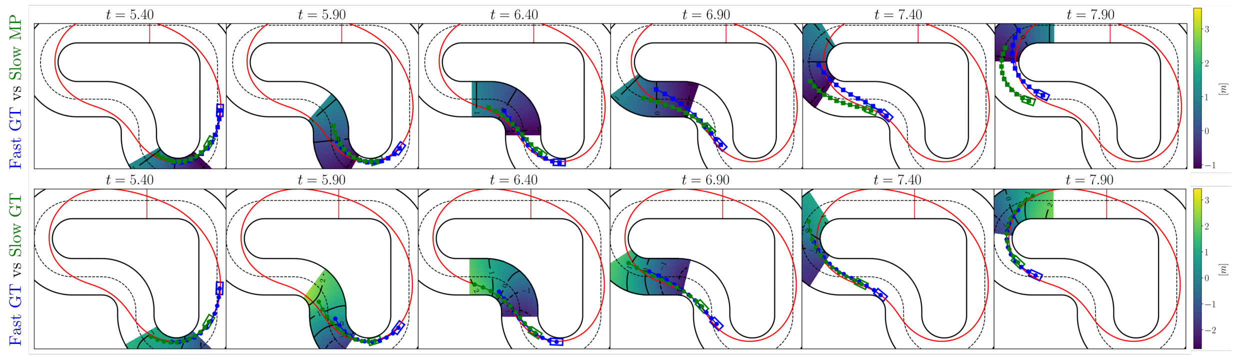


Figure 7.7: In a simulation experiment, the slower (green) car with MP is unable to defend its position against the faster (blue) car with GT (Top). The slower car with GT is able to defend its position against the faster car with GT (Bottom). The heatmaps and black level curves visualize the value functions used in this scenario by the slower (green) car.

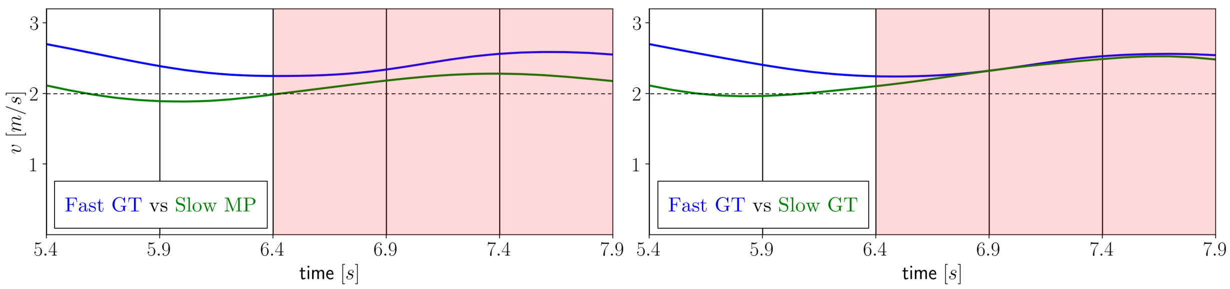


Figure 7.8: Speed traces from two racing scenarios. The vertical black lines correspond to the time stamps in Figure 7.7.

the other hand, when the faster agent uses the GT value function, we can see in the bottom frames that not only does the learned value function encourage track progression, but we also observe non-linear level curves which indicate a preference for certain lateral positions for a given s. In fact, we note that the high value regions shown in these frames match target regions in conventional racing strategy which look to minimize the curvature of the path taken through turns, thereby allowing the agent to carry higher speed. By leveraging guidance from the GT value function, the faster agent is able to find an inside line through the 180° turn which forces the slower agent to deviate from its raceline in order to avoid collisions. This can be clearly seen in the shaded region of Figure 7.6 where the faster agent stays to the right of the centerline as the MPC solution is being guided towards the inside of the turn. This finally results in a successful overtake by the faster agent.

In Figure 7.7, we show a position defense attempt by the slower agent and where the faster agent uses the GT policy. From these frames, we see that the GT value function indicates preference for lateral track positions in the vicinity of the precomputed raceline (of which the learned value function had no prior knowledge). Moreover, we can clearly see from the shaded regions of Figure 7.8 that the GT value function causes the slower agent to speed up in order to defend its lead against the faster agent. This is because the GT value function allows for explicit reasoning over the value of the velocity states by the optimal

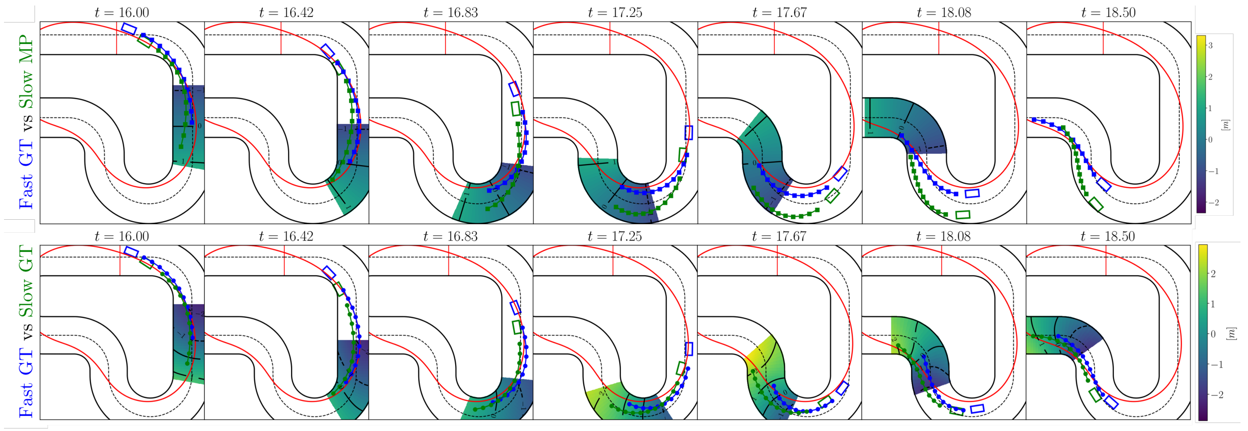


Figure 7.9: In a hardware experiment, the slower (green) car with MP is unable to defend its position against the faster (blue) car with GT (Top). The slower car with GT is able to defend its position against the faster car with GT (Bottom). The heatmaps and black level curves visualize the value functions used in this scenario by the slower (green) car.

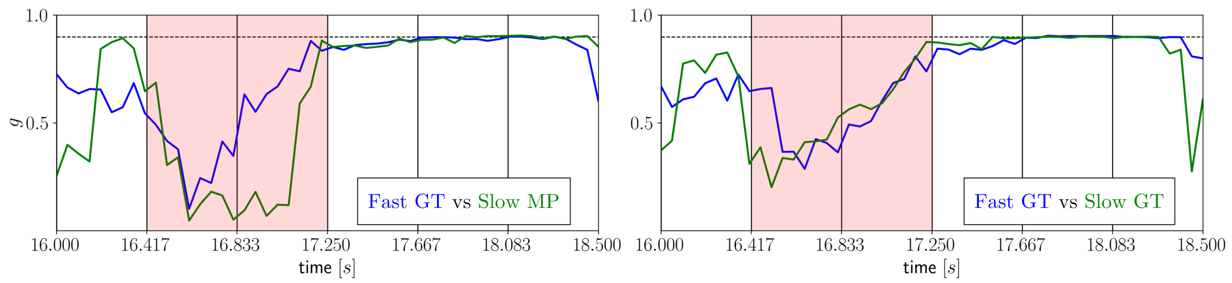


Figure 7.10: Rear tire g traces. The horizontal dashed line indicates the limit of friction with $\mu = 0.9$. The vertical black lines correspond to the time stamps in Figure 7.9.

control policy as conditioned on opponent behavior. On the other hand, when using the MP value function, any reasoning about the velocity of the car can only occur indirectly through the propagation of gradient information through the s coordinate and the vehicle dynamics.

It is important to note that while it is certainly possible to obtain similarly competitive behavior via more a sophisticated hand-engineered value function, our emphasis here is on how we can arrive at such behavior without excessive reward shaping. Instead, by learning from GNE, our GT value function can automatically encode various aspects of conventional racing strategy which can then be leveraged to achieve good competitive performance against an opponent in both overtaking and position defense scenarios.

Hardware Experiments

We conducted hardware experiments on a 1/10th scale RC car platform where state feedback is provided by an OptiTrack motion capture system. The MPC policies for the two cars are both run at 10 Hz on a laptop with a 2.6 GHz 9th-Gen Intel Core i7 CPU and communication of control commands and state measurements are handled using ROS2. We pick 5 different

Table 7.1: Outcomes of the hardware experiments

Scenario	Races won (fast/slow)	Avg. lead (fast)
Fast GT vs. Slow GT	4/1	6.10
Fast GT vs. Slow MP	5/0	9.14
Fast MP vs. Slow GT	0/5	-1.28

joint initial poses for the two cars and run three racing scenarios of 4 laps from each of these starting conditions, i.e. Fast GT vs. Slow GT, Fast GT vs. Slow MP, and Fast MP vs. Slow GT. The results are summarized in Table 7.1, where we can see that the outcomes of these races closely mirror those from the simulation study in both the win rates as well as the average lead of the faster car.

We finally illustrate the impact of the value function on tire friction, which is the limiting factor of performance in car racing. This is shown through the position defense interaction and corresponding friction traces in Figures 7.9 and 7.10, where once again we see that the slower car is able to defend its position when using the GT value function, but is overtaken when using the MP value function. The key differences are highlighted in the shaded regions of 7.10 where it can be clearly seen that although the slower car begins cornering at the limit of friction at approximately the same time for both GT and MP policies, the under-utiliziation of friction by MP as the car enters into the turn results in a much wider trajectory that leaves room for the faster car to overtake. On the other hand, when using the GT value function, the slower car is able to enter the turn more aggresively and therefore defend its position through the turn.

7.6 Chapter Summary

In this chapter, we proposed a strategic learning approach which leveraged GNE from our DG-SQP solver to learn a game-theoretic value function for predicting the reward outcomes of a head-to-head racing dynamic game. We showed that the reward predictions made by the learned value function align with conventional car racing strategies which place an emphasis on maintaining a speed advantage and a relative lateral position advantage over one's opponent when approaching a turn. When incorporated in an MPC policy as a terminal cost function, we show that an agent using the resulting game-theoretic MPC racing policy is able to significantly outperform the progress maximizing baseline in both overtaking and position defense scenarios in simulation and hardware races.

Chapter 8

A Discussion on Game-Theoretic Methods for Prediction and Planning

8.1 Introduction

In Chapter 6, we proposed a solver for generalized Nash equilibria (GNE) of open-loop dynamic games as an approach to obtain highly interactive solutions to the coupled prediction and planning problem. We then used the GNE obtained from the solver in Chapter 7 to learn a game-theoretic value function which, when used as the terminal cost function for an MPC racing policy, allowed it to leverage game-theoretic reward outcomes to guide the solution of the optimal control problem to win races against a baseline policy. The work in these chapters showed that dynamic games are certainly a powerful formalism for describing multi-agent scenarios and that the equilibria of these games can capture highly interactive behavior which are "optimal" in the sense of achieving a strategic equilibrium between agents. Furthermore, we showed that this interactive behavior could indeed be leveraged for real-time planning in competitive scenarios. However, there are some important limitations which come with the dynamic game formalism presented in Chapter 6, namely in the assumptions made about knowledge of the game components in (6.4). In this chapter, we will explore the benefits and, perhaps more importantly, the drawbacks of utilizing equilibria of open-loop dynamic games for joint prediction and planning. We will also examine the state-of-the-art in the field of game-theoretic prediction and planning for robotic systems and discuss solution approaches which are attempting to address the drawbacks we identified.

Running Example

let us first introduce the running example that will be used in this chapter. In particular, we use a head-to-head car racing dynamic game with two agents on a race track which is similar to the one used in the numerical studies of Chapter 6. The dynamic game which describes this competitive scenario is defined, as in (6.5), via the coupled optimal control problems in (8.1) and (8.2) below, which are played over a horizon of length N. The input

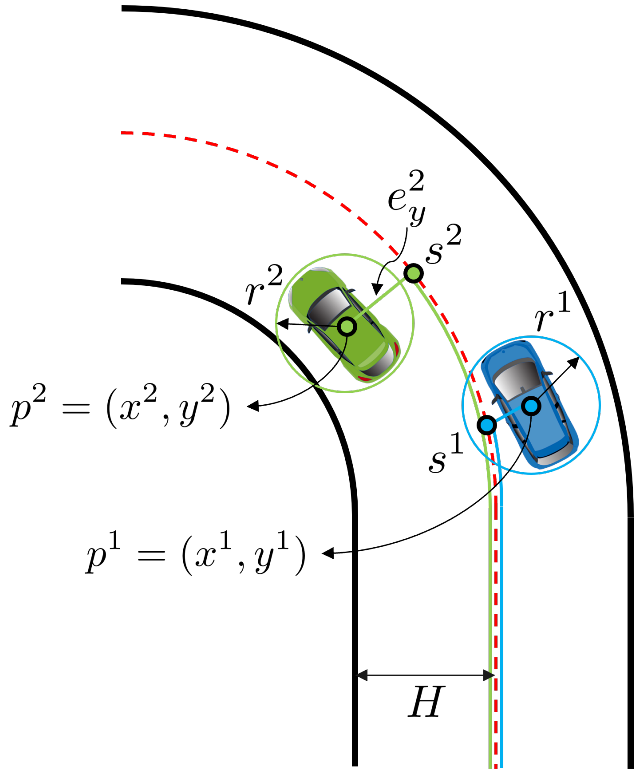


Figure 8.1: Illustration of the car racing running example.

sequences are written as $\mathbf{u}^1 = \{u_0^1, \dots, u_{N-1}^1\}$ and $\mathbf{u}^2 = \{u_0^2, \dots, u_{N-1}^2\}$ and the functions $s_k^i(\mathbf{u}^i)$, $p_k^i(\mathbf{u}^i)$, and $e_{y,k}^i(\mathbf{u}^i)$ apply the input sequence $\mathbf{u}^i$ to the system dynamics, described using a kinematic bicycle model, starting from a given initial condition x_0 and return the value of the specified state at time step k for agent $i = 1, 2$. Note that as in Chapter 6, we have omitted the explicit dependence on x_0 for simplicity of notation.

$$\begin{aligned} \mathbf{u}^{1,\star}(\mathbf{u}^{2,\star}) = \arg\min_{\mathbf{u}^1} \quad & s_N^2(\mathbf{u}^{2,\star}) - s_N^1(\mathbf{u}^1) && (8.1a)\\ \text{subject to} \quad & (r^1 + r^2) - \|p_k^1(\mathbf{u}^1) - p_k^2(\mathbf{u}^{2,\star})\|_2 \leq 0,\ k = 0, \dots, N, && (8.1b)\\ & -H \leq e_{y,k}^1(\mathbf{u}^1) \leq H,\ k = 0, \dots, N, && (8.1c)\\ & u_l^1 \leq u_k^1 \leq u_u^1,\ k = 0, \dots, N-1. && (8.1d) \end{aligned}$$

$$\begin{aligned} \mathbf{u}^{2,\star}(\mathbf{u}^{1,\star}) = \arg\min_{\mathbf{u}^2} \quad & s_N^1(\mathbf{u}^{1,\star}) - s_N^2(\mathbf{u}^2) && (8.2a)\\ \text{subject to} \quad & (r^1 + r^2) - \|p_k^1(\mathbf{u}^{1,\star}) - p_k^2(\mathbf{u}^2)\|_2 \leq 0,\ k = 0, \dots, N, && (8.2b)\\ & -H \leq e_{y,k}^2(\mathbf{u}^2) \leq H,\ k = 0, \dots, N, && (8.2c)\\ & u_l^2 \leq u_k^2 \leq u_u^2,\ k = 0, \dots, N-1. && (8.2d) \end{aligned}$$

Looking at the optimal control problem for Agent 1 in (8.1), we see that the objective (8.1a) consists of the progress difference between the two agents which encourages the maximization of the agent's own progress s_N^1 and the minimization of their opponent's progress s_N^2. The constraints consist of track boundaries and actuation limits in (8.1c) and (8.1d) respectively, which are private to Agent 1, and collision avoidance constraints in (8.1b), which are shared between the agents. As for the optimal control problem for Agent 2 in (8.2), one can quickly see that it mostly identical, with the key difference being the sign changes in the cost function (8.2a), which results in a zero-sum dynamic game. This captures the competitive nature of the agents, where each agent would like to make progress along the race track preferably at the expense of their opponent. However, the two agents are coupled in part by a shared requirement for safety, which is described by their common collision avoidance constraint (8.1b) and (8.2b).

8.2 Advantanges of a GNE Solver for Prediction and Planning

Using this car racing example, let us first discuss the advantages of using GNE of open-loop dynamic games as a model-based solution to the prediction and planning problem. Consider now an alternative solution approach to the joint prediction and planning problem where we construct a centralized optimal control problem from (8.1) and (8.2) and solve jointly for $\mathbf{u}^1$ and $\mathbf{u}^2$. Specifically, let us define the following optimal control problem:

$$\begin{aligned}
\mathbf{u}^{1,\star}, \mathbf{u}^{2,\star} = \arg\min_{\mathbf{u}^1,\mathbf{u}^2} \quad & s_N^2(\mathbf{u}^2) - s_N^1(\mathbf{u}^1) + s_N^1(\mathbf{u}^1) - s_N^2(\mathbf{u}^2) = 0 && \text{(8.3a)}\\
\text{subject to} \quad & (r^1 + r^2) - \|p_k^1(\mathbf{u}^1) - p_k^2(\mathbf{u}^2)\|_2 \leq 0,\ k = 0, \dots, N, && \text{(8.3b)}\\
& -H \leq e_{y,k}^1(\mathbf{u}^1) \leq H,\ k = 0, \dots, N, && \text{(8.3c)}\\
& u_l^1 \leq u_k^1 \leq u_u^1,\ k = 0, \dots, N-1 && \text{(8.3d)}\\
& -H \leq e_{y,k}^2(\mathbf{u}^2) \leq H,\ k = 0, \dots, N, && \text{(8.3e)}\\
& u_l^2 \leq u_k^2 \leq u_u^2,\ k = 0, \dots, N-1. && \text{(8.3f)}
\end{aligned}$$

which was constructed by summing together the agent objectives in (8.1a) and (8.2a) and concatenating the agent constraints. Simplifying the objective in (8.3a), we can easily find that the objective is identically zero regardless of the choice of $\mathbf{u}^1$ and $\mathbf{u}^2$. This is to be expected since, as we mentioned in the previous section, this racing example is an instance of a zero-sum dynamic game. This means that the centralized optimal control problem defined in (8.3) reduces to a feasibility problem where the notion of competition between agents is entirely lost. Indeed, if we compare a GNE of the dynamic game and a solution of the centralized optimal control problem from the same initial condition, we may obtain the trajectories shown in Figure 8.2 where the solution of the centralized optimal control problem is feasible but does not at all capture the competitive goals of the agents. We note that this issue of information loss in the centralized objective function is not only limited to strictly zero-sum games, but can occur in any general-sum game which has zero-sum components in the agent objective function, i.e. any terms which may cancel out when the

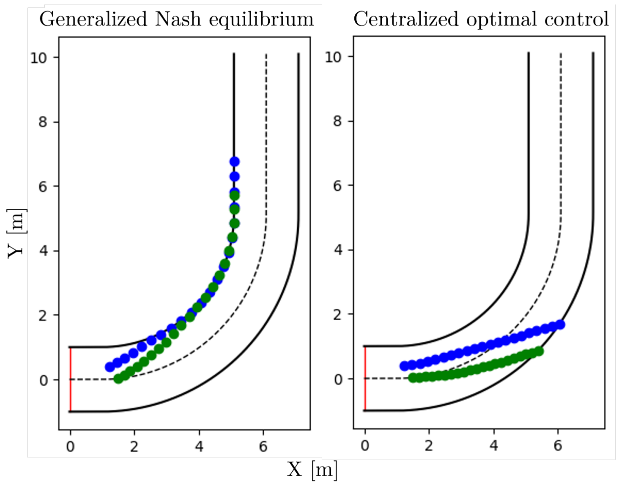


Figure 8.2: Solutions of the joint prediction and planning problem as a GNE of a dynamic game (left) and as the optimal solution of a centralized optimal control problem (right). The blue and green trajectories illustrate the position traces corresponding to the input sequences $\mathbf{u}^{1,\star}$ and $\mathbf{u}^{2,\star}$ respectively.

agent objectives are summed together. A subclass of dynamic games which does not suffer from this limitation is that of potential dynamic games [97, 21, 70, 22] whose equilibria can be equivalently found as the solutions of a centralized optimal control problem. However, potential dynamic games are somewhat restrictive in terms of allowable cost structures and importantly cannot model zero-sum interactions between agents except under very strict conditions [83].

8.3 Limitations of GNE Solvers for Prediction and Planning

We have shown the advantage of GNE of open-loop dynamic games as a solution to the joint prediction and planning problem when compared to a centralized model-based approach. However, it is also important to understand the challenges and limitations of GNE which are currently limiting its application to real-world systems. These are twofold:

1. High computational burden of solving for equilibria of dynamic games.
2. Strict modeling assumptions which are required in the definition of dynamic games.

The first challenge is illustrated by the solve time performance of DG-SQP in Chapter 6. Even for other GNE solvers such as ALGAMES [78] and PATH [38] where users have reported real-time performance for robotic navigation tasks, this has only been shown for up to three agents in a receding horizon manner [90]. While optimizing the implementations of these solvers can certainly help to reduce the computational burden, there are interesting approaches which attempt to leverage learning to accelerate the solution process, which we will discuss in the next section.

The focus of this section will be on the second challenge, where the effectiveness of game-theoretic prediction and planning can easily be hamstrung by possibly restrictive assumptions on knowledge about the other agents in the environment. This challenge can manifest in two ways. The first is in the assumption that agents are "rational", which, in the game-theoretic sense, means that agents have full knowledge of their available actions and the associated costs and will always choose the action with the lowest cost. The second is in the requirement that all components of the game definition (6.4) are known exactly a priori.

Now, of the two manifestations, the first one is perhaps more benign. Though it certainly precludes treatment of completely random agents, such agents are somewhat rare in multi-agent robotic systems. Regardless of whether they leverage a model-based or learned control policy the behavior of an autonomous robot largely follows from the optimization of an objective or reward function, which explicitly quantifies the value of the actions available to it. As such, in the case of multi-robot interactions, this assumption is largely satisfied. In mixed autonomy settings where robots interact with humans, this assumption may be violated by human agents due to their bounded rationality [117, 127]. We will discuss approaches which account for this possible discrepancy in the next section.

The second manifestation, which is more directly an engineering challenge, asks the question "even if we know all agents are rational and are playing a game, how do we know what game they are playing?". Whereas it is reasonable that an ego agent would know their own dynamics, constraint, and objective functions to a high degree of certainty, it is impossible to have exact knowledge of these components for the other agents in the environment when there is no communication between agents. Therefore, the formulation of a stationary dynamic game must be done under a certain hypothesis about the realizations of these components, where it is then entirely likely that mismatch can occur between, say, Agent 1's hypothesis of Agent 2's objective function and the actual objective function used by Agent 2. To illustrate this challenge and its consequences for agent behavior, let us return to the running example, but modify it slightly by replacing the objective functions in (8.1a) and (8.2a) with

$$b^1 s_N^2(\mathbf{u}^2) - a^1 s_N^1(\mathbf{u}^1) \tag{8.4}$$

$$a^2 s_N^1(\mathbf{u}^1) - b^2 s_N^2(\mathbf{u}^2) \tag{8.5}$$

respectively, where $a^1, a^2, b^1, b^2 > 0$ are scalar weights on the terms in the progress difference objectives for the two agents. Consider now a simple case of mismatch where we assume that the agents each know the correct functional form of their opponent's objective, but there can be a mismatch in the weight parameters. We use a subscript to denote which agent "owns" the parameter, i.e. $a_1^1, a_1^2, b_1^1, b_1^2$ and $a_2^1, a_2^2, b_2^1, b_2^2$ correspond to the parameters which are hypothesized by Agents 1 and 2 respectively. We will now show two scenarios to illustrate the impact of mismatch on the open-loop behavior of the agents.

In the first scenario, we set $a_1^1, a_1^2, b_1^1, b_1^2 = 1$ for the dynamic game of Agent 1 and $a_2^1 = 2, b_2^2 = 2, a_2^2, b_2^1 = 1$ for the dynamic game of Agent 2. The GNE for these two dynamic games are shown in the left and middle panels of Figure 8.3, which show that in this case, the

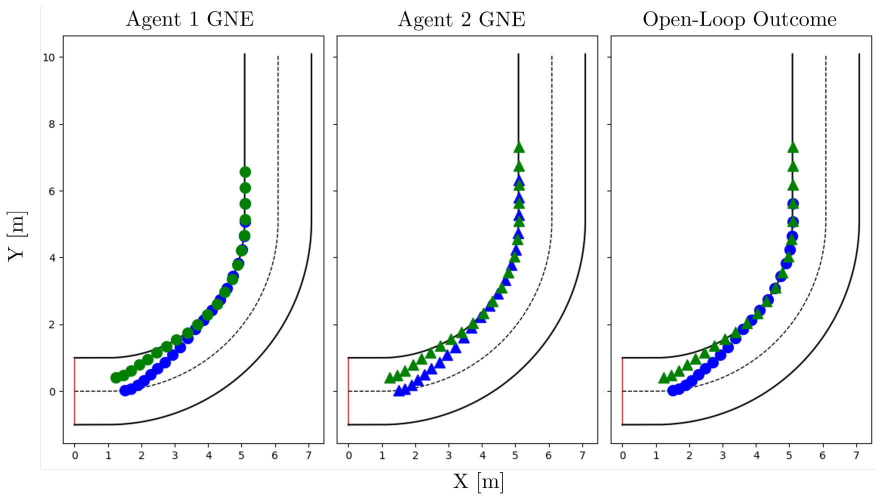


Figure 8.3: Solution GNEs of Agent 1 (left) and Agent 2 (middle) for scenario 1 and the corresponding outcome of each agent applying their open-loop plans, which is collision-free (right). The trajectories progress CCW from the bottom left.

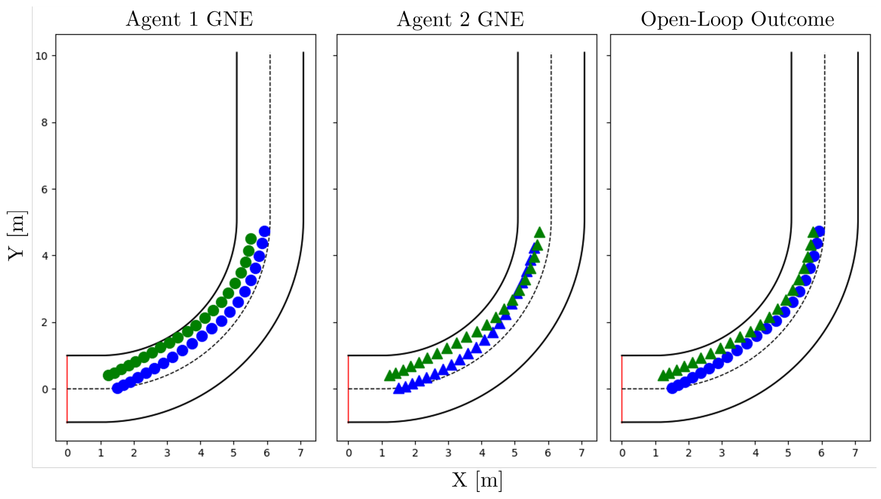


Figure 8.4: Solution GNEs of Agent 1 (left) and Agent 2 (middle) for scenario 2 and the corresponding outcome of each agent applying their open-loop plans, where a collision would occur (right). The trajectories progress CCW from the bottom left.

GNE for the two agents are similar in that both prediction Agent 2 (the green trajectory) leading Agent 1 (the blue trajectory) at the end of the game horizon. Now if both agents execute their open-loop plans, i.e. the trajectories drawn with blue circles and green triangles for Agents 1 and 2 respectively, then the outcome is illustrated in the right panel of Figure 8.3 where the lead of Agent 2 over Agent 1 at the end of the horizon is larger than either had anticipated, but the overall outcome is consistent with the predictions made by both agents and is also collision-free.

In the second scenario, we set $a_1^1 = 5, a_1^2, b_1^1, b_1^2 = 1$ for the dynamic game of Agent 1 and

$b_2^2 = 5, a_2^1, a_2^2, b_2^1 = 1$ for the dynamic game of Agent 2 and show the corresponding GNE in the left and middle panels of Figure 8.4. In this case, we can see that both agents in fact predict that they will be ahead of their opponent at the end of the game horizon. However, if we again have both agents execute their open-loop plans, then the outcome, illustrated in the right panel of Figure 8.4, shows that first of all, there will be a collision between the two agents. Secondly, even if we ignore the collision, we can see that the actual overall outcome is reversed for Agent 2, with Agent 1 ahead at the end of the horizon.

These two scenarios show that mismatch in the game components will lead to diverging predictions and plans over agents. Whereas the differences in scenario 1 are relatively minor and benign in terms of outcome, the differences in scenario 2 are much more consequential in terms of both safety and performance. Of course, in practice for online prediction and planning, it is likely that the GNE of dynamic games would be implemented in a receding horizon manner, thereby incorporating the element of feedback which has been observed to help alleviate the issues caused by discrepancies in the agent GNEs [78]. However, this obviously does not solve the issue of mismatch and can lead to highly suboptimal behavior. In the next section, we will discuss approaches which leverage learning and inference to tackle this problem.

8.4 Going Beyond Stationary Open-Loop Dynamic Games via Learning

In order to address the aforementioned challenges of computational burden of solving for GNE and model mismatch in stationary dynamic games, recent works have proposed data-driven methods which follow one of two main paradigms. Namely *tactical* and *strategic* learning with game-theoretic equilibrium.

The first, which we call *tactical* learning of game-theoretic equilibrium, involves replicating or adapting equilibria with data for use *over* the planning horizon. Replication-based methods attempt to directly reproduce equilibria for dynamic games by training on datasets of equilibria and have been primarily used to alleviate the computational burden of the solution process. In [146] a neural network is trained to directly predict an equilibrium for a given initial condition. This estimate was then used to warm start the Sensitivity Enahanced Iterative Best Response (SE-IBR) algorithm in order to accelerate the solution process. The authors showed significant improvements in solution accuracy when initializing their solver with the neural network predictions. In [106] a similar approach is taken to train so called reference generators for each of the two-players in a pursuit-evasion game. This allowed for approximate equilibria to be found in parallel by querying the reference generator for the opposing agent and solving a trajectory optimization problem for the ego agent. These works are part of an effort to address the first challenge as presented in Section 8.3 from a tactical learning standpoint.

On the other hand, adaptation-based methods have addressed the problem of model mismatch by updating parameters θ of a parametric dynamic game $\Gamma(\theta)$ given observations of the other agents in the environment. $\Gamma(\theta)$ can be obtained from (6.4) by exposing the parameters of the dynamics, constraint, and objective functions as additional arguments to those functions. Such methods still require the explicit definition of the functional form of the game components, but leave the parameters of these components variable such that they can be updated over iterations of learning. We will discuss three classes of approaches which fall under this category.

The first leverages ideas from inverse optimal control (IOC) and attempts to minimize the residuals of the first-order optimality criteria, i.e. KKT conditions, of the dynamic game in (6.6) with respect to its parameters [11, 116, 64]. In particular, the following problem is solved:

$$\min_{\theta,\lambda} \ \|F(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \lambda, \theta)\|, \tag{8.6}$$

where F represents the concatenated KKT conditions in (6.6), $\bar{\mathbf{x}}, \bar{\mathbf{u}}$ are some given state and input sequence, and λ are the dual variables associated with the KKT conditions F. Given that $\|F\|$ is sufficiently small for the solution $\theta^\star$ and $\lambda^\star$, then under regularity conditions, we may say that $\bar{\mathbf{x}}, \bar{\mathbf{u}}$ are a GNE for the parameteric dynamic game $\Gamma(\theta^\star)$.

An important limitation of the IOC-based approach is its dependence on full state and input sequences. This can be relaxed to partially observable settings by optimizing the game parameters in the maximum likelihood sense [90, 95, 105, 84, 107]. These methods take gradient steps in the parameter space to maximize the likelihood of observations under some probabilistic observation model with equilibrium constraints. In particular, variations of the following problem are solved:

$$\begin{aligned} \max_{\theta} \quad & p(\mathbf{y}|\mathbf{x}, \mathbf{u}) && \text{(8.7a)} \\ \text{subject to} \quad & \mathbf{x}, \mathbf{u} \text{ is an equilibrium of } \Gamma(\theta), && \text{(8.7b)} \end{aligned}$$

where $\mathbf{y}$ denotes a sequence of observations. This is done by computing the gradient of the likelihood p in (8.7a) and taking gradient steps in θ. Note that computing the gradient of the likelihood p with respect to the parameters θ requires differentiation through equilibrium solutions of $\Gamma(\theta)$. Such methods therefore leverage the implicit function theorem to construct differentiable game solvers.

A third class of approach performs Bayesian inference over a belief space to estimate the full distribution of the parameter θ. This is done by solving the Bayesian filtering problem as follows to find the posterior distribution of the parameter θ conditioned on the observations $\mathbf{y}$:

$$p(\theta_{k+1}|\mathbf{y}_{k+1}) = \frac{p(y_{k+1}|\theta_{k+1})p(\theta_{k+1}|\mathbf{y}_k)}{\int_{\bar{\theta}} p(y_{k+1}|\bar{\theta})p(\bar{\theta}|\mathbf{y}_k)d\bar{\theta}}, \tag{8.8}$$

which can be done exactly when θ has discrete values [104], or approximately via Kalman or particle filtering when θ is continuous [122, 79]. Since the full probabilistic measurement model is included in (8.8), this allows for estimation of the parameters θ to account for randomness in the measurements y, which can help to model the bounded rationality of human agents [104]. Furthermore, since the full probability distribution is tracked, such approaches can also account for multi-modal behavior of target agents. Ultimately, the goal of all three of these approaches is to align the game description with the observed behavior by refining the value of game parameters based on data. We note that our DG-SQP solver presented in Chapter 6 is complementary to most of these methods as the robustness of the parameter estimation scheme is directly tied to the robustness of the equilibrium solver, which was a focus in our solution approach.

The second paradigm, which is where our racing approach from Chapter 7 belongs, is *strategic* learning with game-theoretic equilibria. This involves distilling high-level strategy from equilibria in the form of a value function to capture multi-agent system performance *beyond* the planning horizon. This idea of value function learning has been a popular approach in reinforcement learning and has been extensively explored in single agent cases to synthesize the terminal cost-to-go function for optimal control [60, 91, 159, 19, 20]. However, there is no game-theoretic component in these methods as they act in environments which are largely stationary. On the other hand, value function learning is also a critical component in many multi-agent reinforcement learning approaches which admit game-theoretic interpretations [119, 135, 108]. However, such methods largely eschew model-based optimal control in favor of learned policies, which limits their application to hardware with safety-critical constraints. So far, to the best of our knowledge, the idea of strategic learning for optimal control with explicit game-theoretic equilibria is relatively unexplored. [50] is the most similar to our work where a value function is learned directly from game equilibria. In particular, the approach formulates a Stackelberg dynamic game over gridded state and input spaces for simplified agent dynamics and solves for the game values via dynamic programming. These values are then interpolated to synthesize a value function which is used as the terminal cost function for an MPC policy. Our work is distinct in that we consider GNE over continuous state and input spaces. We note that, like [50], our racing approach does not require the real-time solution of game equilibria, which helps to address the challenge of computational burden. Instead, we capture long-term strategic information about the multi-agent system in a value function whose output and gradients which can be quickly evaluated. To the best of our knowledge, the work presented in Chapter 7 is the first time a strategic learning approach with game-theoretic equilibria was demonstrated in hardware experiments for car racing.

8.5 Chapter Summary

In this chapter, we examined the benefits and drawbacks of utilizing GNE of open-loop dynamic games for joint prediction and planning. In particular, we showed that GNE of dynamic games are especially well-suited in modeling multi-agent scenarios involving zero-sum objectives, where a centralized optimal control approach would reduce to a feasibility problem which is not representative of the agent intentions. However, two challenges prevent the widespread application of game-theoretic methods for prediction and planning. Namely the high computational burden of solving for equilibria of dynamic games and the strict modeling assumptions which are required in the definition of dynamic games. We illustrated the second challenge by showing that mismatch in the game components leads to diverging predictions and plans over agents which can not only result in suboptimal behavior but also constraint violation. We finally presented a discussion on recent research directions which are tackling these limitations and showed how our work in Chapters 6 and 7 fits in to this broader landscape.

Chapter 9

Conclusions

As robotic systems advance in sophistication and in the complexity of their application, it is not enough to only consider a robot's behavior in a vacuum when they are increasingly expected to operate in environments populated with other intelligent agents. In order to perform their tasks well while obeying constraints, these robotic systems must be endowed with the ability to predict the behavior of the other agents in the environment in addition to planning their own actions over a time horizon. This is especially important in scenarios where agents do not communicate with each other and may behave in an adversarial manner. In this thesis, we have investigated methods for prediction and planning for multi-agent systems over a variety of reward structures and information structures. Namely, we have examined approaches which tackle the problem in cooperative, non-cooperative, and competitive scenarios where agents engage in partial or no communication about their intentions and future plans. These approaches are formulated through a combination of model-based optimal control and data-driven learning techniques, where we have used data in a principled manner to construct or augment the objective and constraint functions of optimal control problems. This allows us to incorporate the rich and expressive behavior stemming from learned models in a transparent manner though the integration in and solution of constrained optimization problems for control. Namely, in Chapters 2 and 3, we introduced a learning model predictive control method for decoupled nonlinear and coupled linear multi-agent systems in a cooperative setting with partial communication between agents. The methods leveraged data from prior executions of an iterative task for the synthesis of the terminal cost function and terminal set of an optimal control problem to improve the performance of the group as a whole while guaranteeing constraint satisfaction over all iterations of the task. In Chapter 4, we proposed an MPC policy for an agent operating in a non-cooperative tightly constrained setting with no communication. Here, we used a dataset of optimal trajectories for a parking lot navigation task to learn a high-level strategy predictor conditioned on the behavior of a dynamic obstacle. This predictor produced a target set which informed the construction of collision avoidance constraints for effectively circumventing the obstacle. In Chapter 5, we begin investigating the problem of autonomous car racing which is a competitive setting with no communication between agents. We formulated a MPC policy for head-to-head racing where the opponent predictions were obtained through a Gaussian process model which was trained on the one-step closed-loop behavior of an opponent from prior races. We leveraged the predictions and uncertainty from the GP to construct tightened collision avoidance constraints which were instrumental in enabling the ego vehicle to overtake its opponent safely in simulation and hardware races. In Chapters 6 and 7, we pose the racing problem as an open-loop dynamic game and present a novel solver for generalized Nash equilibria as the solution to the coupled prediction and planning problem. To address the computational burden of our solver which prevented a real-time receding horizon implementation, we leveraged the robustness of the solver to generate a dataset of GNE which were then used to learn a game-theoretic value function for head-to-head racing. By using this value function as the terminal cost function of an MPC racing policy, we were able to observe highly competitive behaviors of overtaking and position defense in simulation and hardware races. We finally provide some perspectives on the area of game-theoretic methods for prediction and planning in Chapter 8.

9.1 Future Work

We now outline a few exciting directions for future work in the area of prediction and planning for multi-agent systems

Prediction and Planning under Partial Observability

In this thesis, we made the blanket assumption that we have access to the state vector for the multi-agent systems we are predicting and planning for. Of course, in reality this is rarely the case due to limitations in communication and/or sensing. As an example, in the autonomous racing case, agents are unlikely to be broadcasting their state to the field of competitors. We must therefore obtain measurements of our opponents using the onboard sensors. However, these measurements are unlikely to capture the entire vehicle state. In particular, while distance measurements can be obtained in a straightforward manner, e.g. using radar or lidar, the orientation and velocities (linear and angular) of one's opponents must be inferred. This notion of incomplete information can be captured through a measurement function $y = h(x)$. One straightforward method for dealing with partial observability is to use common filtering methods, e.g. the Kalman filter, to obtain a state estimate $\hat{x}$ from an observation. The methods presented in this thesis can then be immediately applied to the state estimate. Now, this decoupled approach to estimation and planning is commonly used and is in fact optimal under certain assumptions on linearity. However, an interesting direction of work could investigate methods where state estimation, especially for the other agents in the environment, is coupled or implicit. This may produce better performance for both estimation and prediction and planning for nonlinear systems.

Prediction and Planning under Uncertainty

Even if partial observability is considered, another simplifying assumption made in this thesis is that the multi-agent systems are deterministic. In reality, the system may exhibit both aleatoric and epistemic uncertainty, which typically manifests as uncertainty in the process model and in the measurements respectively. All of the approaches in the work could be extended to account for both of these types of uncertainty. In particular, this would be helpful with addressing the limitation of model mismatch in dynamic games that we illustrated in Chapter 8, which is a source of aleatoric uncertainty and we may leverage ideas of maximum likelihood estimation and (approximate) Bayesian filtering for addressing this uncertainty. An especially interesting direction of future work for the strategy learning approach in Chapter 7 is an extension to account for a set of different opponent policies. Currently, we generate data for a single setting of parameters for the dynamic game. The stationarity of the generating distribution then allows for even a simple model to accurately capture the reward outcomes. By sampling over both the parameters and the initial conditions of the dynamic game, we could obtain a rich dataset of opponent behavior over a wide spectrum of interactive policies. Further work could explore learning methods which can efficiently leverage the richness in the dataset, though ideally with a minimal increase in model complexity such that it could still be incorporated in a real-time receding horizon control policy.

Prediction and Planning under Closed-Loop Information Structures

The methods described in this thesis have formulated the prediction and planning problem under an open-loop information structure where at each time step, where the resulting prediction and motion plan are in the form of a state and input vector sequence such as $\{u_0^\star, u_1^\star, \ldots, u_{N-1}^\star\}$. Though applying this output in a receding horizon offers some robustness in the closed-loop behavior of the system, such an open-loop structure for policy synthesis can be fragile to model mismatch and measurement noise. Instead of solving the prediction and planning problem over the sequences of control inputs, we could instead solve over the space of feedback policies $\pi_k^\star$ such that the prediction and motion plan are of the form $\{\pi_0^\star(x_0), \pi_1^\star(x_1), \ldots, \pi_{N-1}^\star(x_{N-1})\}$. Such approaches typically result in a larger decision space as we are now solving over the parameters of a parametric family of functions, but can improve the performance of the system in the presence of uncertainty. Future work could examine ways of extending the work in this thesis to leverage a closed-loop information structure.

Efficient Prediction and Planning at Scale

Finally, we note that though much of the work in this thesis applies to multi-agent systems with an arbitrary number of agents, in practice the real-time tractability of these approaches may not extend beyond a handful of agents. In fact, all hardware demonstrations in this thesis were done with two agents. Therefore, an important direction for future work is the improvement of computational efficiency for prediction and planning at a larger scale. This may be done by further leveraging offline computation in the synthesis of (possibly data-driven) components for decentralized online optimal control. We note that for the approaches which were designed for two agents, generalization to an arbitrary number of agents may not be trivial due to the combinatorial aspect of enumerating pairwise strategies or reward outcomes and is also an interesting direction for further investigation.

Bibliography

[1] Ahmed Aboudonia, Annika Eichler, and John Lygeros. "Distributed model predictive control with asymmetric adaptive terminal sets for the regulation of large-scale systems". In: *arXiv:2005.04077* (2020).

[2] Akshay Agrawal et al. "Differentiating Through a Cone Program". In: *Journal of Applied & Numerical Optimization* 1.2 (2019).

[3] Alessandro Alessio, Davide Barcelli, and Alberto Bemporad. "Decentralized model predictive control of dynamically coupled linear systems". In: *Journal of Process Control* 21.5 (2011), pp. 705–714.

[4] Alessandro Alessio and Alberto Bemporad. "Decentralized model predictive control of constrained linear systems". In: *ECC*. 2007, pp. 2813–2818.

[5] Carmen Amo Alonso and Nikolai Matni. "Distributed and localized model predictive control via system level synthesis". In: (2019). arXiv: `1909.10074`. URL: `http://arxiv.org/abs/1909.10074`.

[6] Javier Alonso-Mora et al. "Distributed multi-robot formation control in dynamic environments". In: *Autonomous Robots* 43 (2019), pp. 1079–1100.

[7] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: `10.1007/s12532-018-0139-4`.

[8] Anil Aswani et al. "Practical comparison of optimization algorithms for learning-based MPC with linear models". In: *arXiv preprint arXiv:1404.2843* (2014).

[9] Anil Aswani et al. "Provably safe and robust learning-based model predictive control". In: *Automatica* 49.5 (2013), pp. 1216–1226.

[10] Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea. "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach". In: *IEEE/RSJ Int. Conf. Intell. Robots Sys.* IEEE. 2012, pp. 1917–1922.

[11] Chaitanya Awasthi and Andrew Lamperski. "Inverse differential games with mixed inequality constraints". In: *2020 American control conference (ACC)*. IEEE. 2020, pp. 2182–2187.

[12] Andrea Bajcsy et al. “Learning Vision-based Pursuit-Evasion Robot Policies”. In: *arXiv preprint arXiv:2308.16185* (2023).

[13] Goran Banjac et al. “Decentralized resource allocation via dual consensus ADMM”. In: *Proc. Am. Control Conf.* 2019-July (2019), pp. 2789–2794. ISSN: 07431619. arXiv: `1809.07376`.

[14] Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[15] Alberto Bemporad, Maurice Heemels, Mikael Johansson, et al. *Networked control systems*. Vol. 406. Springer, 2010.

[16] Dimitri Panteli Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, Inc., 1989.

[17] Johannes Betz et al. “Autonomous vehicles on the edge: A survey on autonomous vehicle racing”. In: *IEEE Open Journal of Intelligent Transportation Systems* (2022).

[18] Johannes Betz et al. “TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge”. In: *Journal of Field Robotics* 40.4 (2023), pp. 783–809.

[19] Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. “Blending MPC & Value Function Approximation for Efficient Reinforcement Learning”. In: *International Conference on Learning Representations*. 2020.

[20] Mohak Bhardwaj et al. “Information theoretic model predictive q-learning”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 840–850.

[21] Maulik Bhatt, Yixuan Jia, and Negar Mehr. “Efficient Constrained Multi-Agent Trajectory Optimization Using Dynamic Potential Games”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 7303–7310.

[22] Maulik Bhatt and Negar Mehr. “Strategic Decision-Making in Multi-Agent Domains: A Weighted Potential Dynamic Game Approach”. In: *arXiv preprint arXiv:2308.05876* (2023).

[23] Paul T Boggs and Jon W Tolle. “Sequential quadratic programming”. In: *Acta numerica* 4 (1995), pp. 1–51.

[24] Francesco Borrelli. *Constrained optimal control of linear and hybrid systems*. Berlin: Springer, 2003. ISBN: 354000257X. DOI: `10.1007/BF03047352`.

[25] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[26] Carlo Bosio et al. “Automated Layout Design and Control of Robust Cooperative Grasped-Load Aerial Transportation Systems”. In: *arXiv preprint arXiv:2310.07649* (2023).

[27] Stephen Boyd. *Sequential Convex Programming.* Lecture Notes. 2008.

[28] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Found. Trends Mach. Learn.* 3.1 (2010), pp. 1–122. ISSN: 1935-8237. DOI: `10.1561/2200000016`.

[29] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[30] Tim Brüdigam et al. "Gaussian Process-based Stochastic Model Predictive Control for Overtaking in Autonomous Racing". In: *ArXiv* abs/2105.12236 (2021).

[31] Eduardo Camponogara et al. "Distributed model predictive control". In: *Control Syst. Mag.* 22.1 (2002), pp. 44–52. ISSN: 10991514. DOI: `10.1002/oca.2167`.

[32] John F. Canny. *The Complexity of Robot Motion Planning.* Cambridge, MA, USA: MIT Press, 1988. ISBN: 0262031361.

[33] Yufan Chen, Mark Cutler, and Jonathan P How. "Decoupled multiagent path planning via incremental sequential convex programming". In: *IEEE Int. Conf. Robot. Autom.* 2015, pp. 5954–5961.

[34] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods.* SIAM, 2000.

[35] Christian Conte et al. "Distributed synthesis and stability of cooperative distributed model predictive control for linear systems". In: *Automatica* 69 (2016), pp. 117–125.

[36] Henggang Cui et al. "Deep Kinematic Models for Kinematically Feasible Vehicle Trajectory Predictions". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 10563–10569.

[37] Georgios Darivianakis, Annika Eichler, and John Lygeros. "Distributed model predictive control for linear systems with adaptive terminal sets". In: *IEEE Trans. Automat. Contr.* 65.3 (2020), pp. 1044–1056. ISSN: 0018-9286. DOI: `10.1109/tac.2019.2916774`.

[38] Steven P Dirkse and Michael C Ferris. "The path solver: a nommonotone stabilization scheme for mixed complementarity problems". In: *Optimization methods and software* 5.2 (1995), pp. 123–156.

[39] Nemanja Djuric et al. "Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving". In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2020), pp. 2084–2093.

[40] Dmitri Dolgov et al. "Path planning for autonomous vehicles in unknown semi-structured environments". In: *International Journal of Robotics Research* 29.5 (2010), pp. 485–501.

[41] Alexander Domahidi and Juan Jerez. *FORCES Professional.* Embotech AG. 2014–2023.

[42] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[43] William B Dunbar and Richard M Murray. “Distributed receding horizon control for multi-vehicle formation stabilization”. In: *Automatica* 42.4 (2006), pp. 549–558.

[44] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.

[45] Francisco Facchinei and Christian Kanzow. “Generalized Nash equilibrium problems”. In: *Annals of Operations Research* 175.1 (2010), pp. 177–211.

[46] F Farokhi, I Shames, and K H Johansson. “Distributed MPC via dual decomposition and alternative direction method of multipliers”. In: *Distrib. Model Predict. Control Made Easy*. Ed. by José M Maestre and Rudy R Negenborn. Dordrecht: Springer Netherlands, 2014, pp. 115–131. ISBN: 978-94-007-7006-5. DOI: 10.1007/978-94-007-7006-5_7.

[47] Timm Faulwasser, Benjamin Kern, and Rolf Findeisen. “Model predictive path-following for constrained nonlinear systems”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE. 2009, pp. 8642–8647.

[48] Laura Ferranti et al. “Coordination of multiple vessels via distributed nonlinear model predictive control”. In: *ECC*. 2018, pp. 2523–2528.

[49] MC Ferris and S Lucidi. “Nonmonotone stabilization methods for nonlinear equations”. In: *Journal of Optimization Theory and Applications* 81.1 (1994), pp. 53–71.

[50] Jaime F Fisac et al. “Hierarchical game-theoretic planning for autonomous vehicles”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9590–9596.

[51] Jaime F Fisac et al. “Probabilistically Safe Robot Planning with Confidence-Based Human Predictions”. In: *14th Robotics: Science and Systems, RSS 2018*. MIT Press Journals. 2018.

[52] David Fridovich-Keil et al. “Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games”. In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 1475–1481.

[53] Gianluca Frison and Moritz Diehl. “HPIPM: a high-performance quadratic programming framework for model predictive control”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 6563–6569.

[54] Jacob Gardner et al. “Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration”. In: *Advances in neural information processing systems* 31 (2018).

[55] Pontus Giselsson and Anders Rantzer. “On feasibility, stability and performance in distributed model predictive control”. In: *IEEE Trans. Automat. Contr.* 59.4 (2014), pp. 1031–1036. ISSN: 00189286. DOI: 10.1109/TAC.2013.2285779. arXiv: 1302.1974.

[56] David González et al. "A Review of Motion Planning Techniques for Automated Vehicles". In: *IEEE Trans. Intell. Transp. Syst.* 17.4 (2016), pp. 1135–1145.

[57] Erico Guizzo. *Kiva Systems: Three engineers, hundreds of robots, one warehouse.* Sept. 2022. URL: `https://spectrum.ieee.org/three-engineers-hundreds-of-robots-one-warehouse`.

[58] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. "Scalable Variational Gaussian Process Classification". In: *AISTATS*. 2015.

[59] Andreas Hock and Angela P Schoellig. "Distributed iterative learning control for multi-agent systems: Theoretic developments and application to formation flying". In: *Auton. Rob.* 43.8 (2019), pp. 1989–2010.

[60] David Hoeller, Farbod Farshidian, and Marco Hutter. "Deep value model predictive control". In: *Conference on Robot Learning*. PMLR. 2020, pp. 990–1004.

[61] Joey Hong, Benjamin Sapp, and James Philbin. "Rules of the Road: Predicting Driving Behavior With a Convolutional Model of Semantic Interactions". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 8446–8454.

[62] Mokter Hossain. *Self-Driving Robots: A Revolution in the Local Delivery.* Apr. 2022. URL: `https://cmr.berkeley.edu/2022/04/self-driving-robots-a-revolution-in-the-local-delivery/`.

[63] Haimin Hu et al. "Non-Cooperative Distributed MPC with Iterative Learning". In: *IFAC-PapersOnLine* (2020).

[64] Jairo Inga et al. "Inverse dynamic games based on maximum entropy inverse reinforcement learning". In: *arXiv preprint arXiv:1911.07503* (2019).

[65] Yixuan Jia, Maulik Bhatt, and Negar Mehr. "RAPID: Autonomous Multi-Agent Racing using Constrained Potential Dynamic Games". In: *arXiv preprint arXiv:2305.00579* (2023).

[66] Colin N. Jones and Manfred Morari. "Polytopic approximation of explicit model predictive controllers". In: *IEEE Trans. Automat. Contr.* 55.11 (2010), pp. 2542–2553. ISSN: 00189286. DOI: `10.1109/TAC.2010.2047437`.

[67] Chanyoung Jung et al. "An Autonomous System for Head-to-Head Race: Design, Implementation and Analysis; Team KAIST at the Indy Autonomous Challenge". In: *arXiv preprint arXiv:2303.09463* (2023).

[68] Christos Katrakazas et al. "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". In: *Transp. Res. Part C Emerg. Technol.* 60 (2015), pp. 416–442.

[69] Elia Kaufmann et al. "Champion-level drone racing using deep reinforcement learning". In: *Nature* 620.7976 (2023), pp. 982–987.

[70] Talha Kavuncu, Ayberk Yaraneri, and Negar Mehr. “Potential iLQR: A Potential-Minimizing Controller for Planning Multi-Agent Interactive Trajectories”. In: *17th Robotics: Science and Systems, RSS 2021*. MIT Press Journals. 2021.

[71] TamáS Keviczky, Francesco Borrelli, and Gary J Balas. “Decentralized receding horizon control for large scale dynamically decoupled systems”. In: *Automatica* 42.12 (2006), pp. 2105–2115.

[72] Maximilian Kneissl et al. “A Multi-Vehicle Control Framework With Application to Automated Valet Parking”. In: *IEEE Trans. Intell. Transp. Syst.* (2020), pp. 1–11. ISSN: 1524-9050.

[73] Jason Kong et al. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *2015 IEEE intelligent vehicles symposium (IV)*. IEEE. 2015, pp. 1094–1099.

[74] Jason Kong et al. “Kinematic and dynamic vehicle models for autonomous driving control design”. In: *IEEE Intelligent Vehicles Symposium, Proceedings* 2015-August (2015), pp. 1094–1099.

[75] Forrest Laine et al. “Multi-Hypothesis Interactions in Game-Theoretic Motion Planning”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 8016–8023.

[76] Forrest Laine et al. “The computation of approximate generalized feedback nash equilibria”. In: *SIAM Journal on Optimization* 33.1 (2023), pp. 294–318.

[77] Denise Lam, Chris Manzie, and Malcolm Good. “Model predictive contouring control”. In: *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 6137–6142.

[78] Simon Le Cleac’h, Mac Schwager, and Zachary Manchester. “ALGAMES: a fast augmented Lagrangian solver for constrained dynamic games”. In: *Autonomous Robots* 46.1 (2022), pp. 201–215.

[79] Simon Le Cleac’h, Mac Schwager, and Zachary Manchester. “LUCIDGames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5485–5492.

[80] Jonathan W Lee et al. “Traffic Control via Connected and Automated Vehicles: An Open-Road Field Experiment with 100 CAVs”. In: *arXiv preprint arXiv:2402.17043* (2024).

[81] Stéphanie Lefèvre et al. “Driver models for personalised driving assistance”. In: *Vehicle System Dynamics* 53.12 (2015), pp. 1705–1720.

[82] Esko Lehtonen et al. “Look-ahead fixations in curve driving”. In: *Ergonomics* 56.1 (2013), pp. 34–44.

[83] Changxi Li, Fenghua He, and Ning Hao. “Verification and design of zero-sum potential games”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 16932–16937.

[84] Jingqi Li et al. “Cost Inference for Feedback Dynamic Games from Noisy Partial State Observations and Incomplete Trajectories”. In: *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 2023, pp. 1062–1070.

[85] Sheng Li, Maxim Egorov, and Mykel J. Kochenderfer. “Optimizing collision avoidance in dense airspace using deep reinforcement learning”. In: *13th USA/Europe Air Traffic Management Research and Development Seminar 2019* 3 (2019).

[86] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.

[87] Alexander Liniger, Alexander Domahidi, and Manfred Morari. “Optimization-based autonomous racing of 1: 43 scale RC cars”. In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647.

[88] Alexander Liniger and John Lygeros. “A noncooperative game approach to autonomous racing”. In: *IEEE Transactions on Control Systems Technology* 28.3 (2019), pp. 884–897.

[89] Jinfeng Liu, David Muñoz de la Peña, and Panagiotis D Christofides. “Distributed model predictive control of nonlinear systems subject to asynchronous and delayed measurements”. In: *Automatica* 46.1 (2010), pp. 52–61.

[90] Xinjie Liu, Lasse Peters, and Javier Alonso-Mora. “Learning to Play Trajectory Games Against Opponents with Unknown Objectives”. In: *IEEE Robotics and Automation Letters* (2023).

[91] Kendall Lowrey et al. “Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control”. In: *International Conference on Learning Representations*. 2018.

[92] Sergio Lucia, Markus Kögel, and Rolf Findeisen. “Contract-based predictive control of distributed systems with plug and play capabilities”. In: *IFAC-PapersOnLine* 48.23 (2015), pp. 205–211. ISSN: 24058963. DOI: `10.1016/j.ifacol.2015.11.284`.

[93] Carlos E Luis and Angela P Schoellig. “Trajectory generation for multiagent point-to-point transitions via distributed model predictive control”. In: *IEEE Robot. Autom. Letters* 4.2 (2019), pp. 375–382.

[94] David Q Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.

[95] Negar Mehr et al. “Maximum-entropy multi-agent dynamic games: Forward and inverse solutions”. In: *IEEE Transactions on Robotics* (2023).

[96] Alain Micaelli and Claude Samson. “Trajectory tracking for unicycle-type and two-steering-wheels mobile robots”. PhD thesis. INRIA, 1993.

[97] Dov Monderer and Lloyd S Shapley. “Potential games”. In: *Games and economic behavior* 14.1 (1996), pp. 124–143.

[98] Simon Muntwiler et al. "Distributed Model Predictive Safety Certification for Learning-based Control". In: *ArXiv:1911.01832* (2019). DOI: 2004.01298.

[99] Ofir Nachum et al. "Multi-Agent Manipulation via Locomotion using Hierarchical Sim2Real". In: *Conference on Robot Learning*. PMLR. 2020, pp. 110–121.

[100] Rudy R Negenborn, Bart De Schutter, and J Hellendoorn. "Multi-agent model predictive control: A survey". In: *arXiv:0908.1076* (2009).

[101] Rudy R Negenborn and Jose Maria Maestre. "Distributed model predictive control: An overview and roadmap of future research opportunities". In: *IEEE Control Systems Magazine* 34.4 (2014), pp. 87–97.

[102] Hans B Pacejka and Egbert Bakker. "The magic formula tyre model". In: *Vehicle system dynamics* 21.S1 (1992), pp. 1–18.

[103] Brian Paden et al. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Trans. Intell. Veh.* 1.1 (2016), pp. 33–55.

[104] Lasse Peters et al. "Contingency Games for Multi-Agent Interaction". In: *arXiv preprint arXiv:2304.05483* (2023).

[105] Lasse Peters et al. "Inferring Objectives in Continuous Dynamic Games from Noise-Corrupted Partial State Observations". In: *Robotics: Science and Systems XVII, 2021*. 2021.

[106] Lasse Peters et al. "Learning Mixed Strategies in Trajectory Games". In: *Proceedings Robotics: Science and System XVIII* (2022).

[107] Lasse Peters et al. "Online and offline learning of player objectives from partial observations in dynamic games". In: *The International Journal of Robotics Research* (2023), p. 02783649231182453.

[108] Tabish Rashid et al. "Monotonic value function factorisation for deep multi-agent reinforcement learning". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 7234–7284.

[109] Stefano Riverso and Giancarlo Ferrari-Trecate. "Tube-based distributed control of linear constrained systems". In: (2011), pp. 1–10.

[110] J Ben Rosen. "Existence and uniqueness of equilibrium points for concave n-person games". In: *Econometrica: Journal of the Econometric Society* (1965), pp. 520–534.

[111] Ugo Rosolia and Francesco Borrelli. "Learning how to autonomously race a car: a predictive control approach". In: *IEEE Transactions on Control Systems Technology* 28.6 (2019), pp. 2713–2719.

[112] Ugo Rosolia and Francesco Borrelli. "Learning model predictive control for iterative tasks: A computationally efficient approach for linear system". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 3142–3147. ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.324. arXiv: 1702.07064.

[113] Ugo Rosolia and Francesco Borrelli. “Learning model predictive control for iterative tasks. a data-driven control framework”. In: *IEEE Transactions on Automatic Control* 63.7 (2017), pp. 1883–1896.

[114] Ugo Rosolia and Francesco Borrelli. “Minimum Time Learning Model Predictive Control”. In: *arXiv preprint arXiv:1911.09239* (2019).

[115] Ugo Rosolia, Xiaojing Zhang, and Francesco Borrelli. “Robust learning model predictive control for linear systems”. In: (2019). arXiv: `arXiv:1911.09234v1`.

[116] Simon Rothfuß et al. “Inverse optimal control for identification in non-cooperative differential games”. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 14909–14915.

[117] Ariel Rubinstein. *Modeling bounded rationality.* MIT press, 1998.

[118] Tim Salzmann et al. “Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data”. In: *ECCV*. 2020.

[119] Julian Schrittwieser et al. “Mastering atari, go, chess and shogi by planning with a learned model”. In: *Nature* 588.7839 (2020), pp. 604–609.

[120] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. “Planning and decision-making for autonomous vehicles”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[121] Wilko Schwarting et al. “Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), pp. 1928–1935.

[122] Wilko Schwarting et al. “Social behavior for autonomous vehicles”. In: *Proceedings of the National Academy of Sciences* 116.50 (2019), pp. 24972–24978.

[123] Wilko Schwarting et al. “Stochastic dynamic games in belief space”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 2157–2172.

[124] Xu Shen and Francesco Borrelli. “Multi-vehicle Conflict Resolution in Highly Constrained Spaces by Merging Optimal Control and Reinforcement Learning”. In: *IFAC-PapersOnLine* 56.2 (2023), pp. 3308–3313.

[125] Xu Shen and Francesco Borrelli. “Reinforcement learning and distributed model predictive control for conflict resolution in highly constrained spaces”. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2023, pp. 1–6.

[126] Xu Shen et al. “ParkPredict: Motion and Intent Prediction of Vehicles in Parking Lots”. In: *arXiv preprint arXiv:2004.10293* (2020).

[127] Herbert A Simon. “Bounded rationality”. In: *Utility and probability* (1990), pp. 15–18.

[128] Stanley W Smith et al. “Improving urban traffic throughput with vehicle platooning: Theory and experiments”. In: *IEEE Access* 8 (2020), pp. 141208–141223.

[129] Riccardo Spica et al. “A real-time game theoretic planner for autonomous two-player drone racing”. In: *IEEE Transactions on Robotics* 36.5 (2020), pp. 1389–1403.

[130] Raphael E Stern et al. “Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments”. In: *Transportation Research Part C: Emerging Technologies* 89 (2018), pp. 205–221.

[131] Yvonne R Stürz et al. “Distributed learning model predictive control for linear systems”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 4366–4373.

[132] Yvonne R. Stürz, Annika Eichler, and Roy S. Smith. “Distributed control design for heterogeneous interconnected systems”. In: *arXiv preprint arXiv:2004.04876* (2020).

[133] Yvonne Rebecca Stürz, Annika Eichler, and Roy Stephen Smith. “Scalable controller synthesis for heterogeneous interconnected systems applicable to an overlapping control framework”. In: *Eur. Control Conf.* 1 (2018), pp. 2561–2568.

[134] Libo Sun, Jinfeng Zhai, and Wenhu Qin. “Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning”. In: *IEEE Access* 7 (2019), pp. 109544–109554.

[135] Peter Sunehag et al. “Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 2085–2087.

[136] Peter Trautman and Andreas Krause. “Unfreezing the robot: Navigation in dense, interacting crowds”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 797–803.

[137] Paul A. Trodden and J. M. Maestre. “Distributed predictive control with minimization of mutual disturbances”. In: *Automatica* 77 (2017), pp. 31–43. ISSN: 00051098. DOI: `10.1016/j.automatica.2016.11.023`.

[138] Charlott Vallon and Francesco Borrelli. “Data-Driven Hierarchical Predictive Learning in Unknown Environments”. In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. 2020, pp. 104–109.

[139] Charlott Vallon and Francesco Borrelli. “Data-Driven Strategies for Hierarchical Predictive Control in Unknown Environments”. In: *ArXiv* abs/2105.06005 (2022).

[140] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *Nature* 575.7782 (2019), pp. 350–354.

[141] Andreas Wächter and Lorenz T Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106 (2006), pp. 25–57.

[142] Marcel Walch et al. “Autonomous driving”. In: 21.5 (2015), pp. 11–18. DOI: `10.1145/2799250.2799268`.

[143] Mingyu Wang et al. “Game Theoretic Planning for Self-Driving Cars in Competitive Scenarios.” In: *Robotics: Science and Systems*. 2019, pp. 1–9.

[144] Mingyu Wang et al. "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios". In: *IEEE Transactions on Robotics* 37.4 (2021), pp. 1313–1325.

[145] Zijian Wang, Riccardo Spica, and Mac Schwager. "Game theoretic motion planning for multi-robot racing". In: *Distributed Autonomous Robotic Systems: The 14th International Symposium*. Springer. 2019, pp. 225–238.

[146] Zijian Wang, Tim Taubner, and Mac Schwager. "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments". In: *Robotics and Autonomous Systems* 125 (2020), p. 103410.

[147] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[148] Stephen Wright, Jorge Nocedal, et al. "Numerical optimization". In: *Springer Science* 35.67-68 (1999), p. 7.

[149] Peter R Wurman et al. "Outracing champion Gran Turismo drivers with deep reinforcement learning". In: *Nature* 602.7896 (2022), pp. 223–228.

[150] Chenyu Yang et al. "Collaborative navigation and manipulation of a cable-towed load by multiple quadrupedal robots". In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 10041–10048.

[151] Zhengyuan Yang et al. "End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 2289–2294.

[152] Youngmin Yoon et al. "Interaction-Aware Probabilistic Trajectory Prediction of Cut-In Vehicles Using Gaussian Process for Proactive Control of Autonomous Vehicles". In: *IEEE Access* 9 (2021), pp. 63440–63455.

[153] Ekim Yurtsever et al. "A survey of autonomous driving: Common practices and emerging technologies". In: *IEEE access* 8 (2020), pp. 58443–58469.

[154] A. Zanelli et al. "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs". In: *International Journal of Control* (2017), pp. 1–17.

[155] Ornella Zerlenga, Vincenzo Cirillo, and Rosina Iaderosa. "Once upon a time there were fireworks. The new nocturnal drones light shows". In: *img Journal* 4 (2021), pp. 402–425.

[156] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. "Optimization-Based Collision Avoidance". In: *IEEE Transactions on Control Systems Technology* (2020), pp. 1–12.

[157] Xiaojing Zhang et al. "Autonomous Parking Using Optimization-Based Collision Avoidance". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, Dec. 2019, pp. 4327–4332.

[158] Hang Zhao et al. "TNT: Target-driveN Trajectory Prediction". In: *CoRL*. 2020.

[159] Mingyuan Zhong et al. "Value function approximation and model predictive control". In: *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*. IEEE. 2013, pp. 100–107.

[160] Edward L Zhu and Francesco Borrelli. "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 3211–3217.

[161] Edward L Zhu et al. "A Gaussian Process Model for Opponent Prediction in Autonomous Racing". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 8186–8191.

[162] Edward L Zhu et al. "Trajectory optimization for nonlinear multi-agent systems using decentralized learning model predictive control". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 6198–6203.

[163] Quanyan Zhu. "A Lagrangian approach to constrained potential games: Theory and examples". In: *2008 47th IEEE Conference on Decision and Control*. IEEE. 2008, pp. 2420–2425.

Appendix A

Vehicle Dynamics Models

Here, we describe the kinematic and dynamic bicycle models which are commonly used throughout this dissertation.

The Frenet-frame kinematic bicycle model has the state and input vectors:

$$x_{\text{kin}} = [v, s, e_y, e_\psi]^\top,\ u_{\text{kin}} = [a, \delta]^\top,$$

and the continuous-time dynamics are written as:

$$\begin{aligned}
\dot{v} &= a, \\
\dot{s} &= v\cos(e_\psi + \beta(\delta))/(1 - e_y\kappa(s)), \\
\dot{e}_y &= v\sin(e_\psi + \beta(\delta)), \\
\dot{e}_\psi &= v\sin(\beta(\delta))/l^r - \kappa(s)v\cos(e_\psi + \beta(\delta))/(1 - e_y\kappa(s)),
\end{aligned} \tag{A.1}$$

where $\beta(\delta) = \arctan(\tan\delta \cdot l^f/(l^f + l^r))$ is the side slip angle, l^f, l^r are the distance from the center of mass to the front and rear axles respectively, and $\kappa(s) = (\tau'_x(s)\tau''_y(s) - \tau'_y(s)\tau''_x(s))/(\tau'_x(s)^2 + \tau'_y(s)^2)^{3/2}$ is the curvature of the path τ at a given s.

The Frenet-frame dynamic bicycle model has the state and input vectors:

$$x_{\text{dyn}} = [v_x, v_y, \omega, s, e_y, e_\psi]^\top,\ u_{\text{dyn}} = [a_x, \delta]^\top,$$

and the continuous-time dynamics are written as:

$$\begin{aligned}
\dot{v}_x &= a_x - \frac{1}{m}F_y^f\sin\delta - c_d v_x + \omega v_y, \\
\dot{v}_y &= \frac{1}{m}(F_y^r + F_y^f\cos\delta) - \omega v_x, \\
\dot{\omega} &= \frac{1}{I_z}(l^f F_y^f\cos\delta - l^r F_y^r), \\
\dot{s} &= (v_x\cos(e_\psi) - v_y\sin(e_\psi))/(1 - e_y\kappa(s)), \\
\dot{e}_y &= v_x\sin(e_\psi) + v_y\cos(e_\psi), \\
\dot{e}_\psi &= \omega - \kappa(s)(v_x\cos(e_\psi) - v_y\sin(e_\psi))/(1 - e_y\kappa(s)),
\end{aligned}$$

where m and I_z are the mass and yaw moment of inertia of the vehicle and c_d is an aerodynamic drag coefficient. The lateral tire forces are modeled using a simplified Pacejka tire model [102]:

$$\begin{aligned} F_y^f &= D^f \sin(C^f + \arctan(B^f \alpha^f)) \\ F_y^r &= D^r \sin(C^r + \arctan(B^r \alpha^r)), \end{aligned}$$

where B, C, and D are parameters obtained experimentally and α^f and α^r are the slip angles of the front and rear tires respectively:

$$\begin{aligned} \alpha^f &= -\arctan\left(\frac{\omega l^f + v_y}{v_x}\right) + \delta \\ \alpha^r &= \arctan\left(\frac{\omega l^r - v_y}{v_x}\right). \end{aligned}$$

The inertial-frame kinematic bicycle model has the state and input vectors:

$$\bar{x}_{\text{kin}} = [v, x, y, \psi]^\top,\ \bar{u}_{\text{kin}} = [a, \delta]^\top, \tag{A.2}$$

and the continuous-time dynamics are written as:

$$\begin{aligned} \dot{v} &= a, \\ \dot{x} &= v\cos(\beta(\delta) + \psi), \\ \dot{y} &= v\sin(\beta(\delta) + \psi), \\ \dot{\psi} &= v\sin(\beta(\delta))/l^r, \end{aligned}$$

The inertial-frame dynamic bicycle model has the state and input vectors:

$$\bar{x}_{\text{dyn}} = [v_x, v_y, \omega, x, y, \psi]^\top,\ \bar{u}_{\text{dyn}} = [a_x, \delta]^\top,$$

and the continuous-time dynamics are written as:

$$\begin{aligned} \dot{v}_x &= a_x - \frac{1}{m}F_y^f \sin\delta - c_d v_x + \omega v_y, \\ \dot{v}_y &= \frac{1}{m}(F_y^r + F_y^f \cos\delta) - \omega v_x, \\ \dot{\omega} &= \frac{1}{I_z}(l^f F_y^f \cos\delta - l^r F_y^r), \\ \dot{x} &= v_x\cos\psi - v_y\sin\psi, \\ \dot{y} &= v_x\sin\psi + v_y\cos\psi, \\ \dot{\psi} &= \omega. \end{aligned}$$

We obtain the discrete-time Frenet-frame dynamics f_{kin}, f_{dyn}, and inertial-frame dynamics $\bar{f}_{\text{kin}}$, $\bar{f}_{\text{dyn}}$ via 4-th order Runge-Kutta discretization with a time step of Δt.