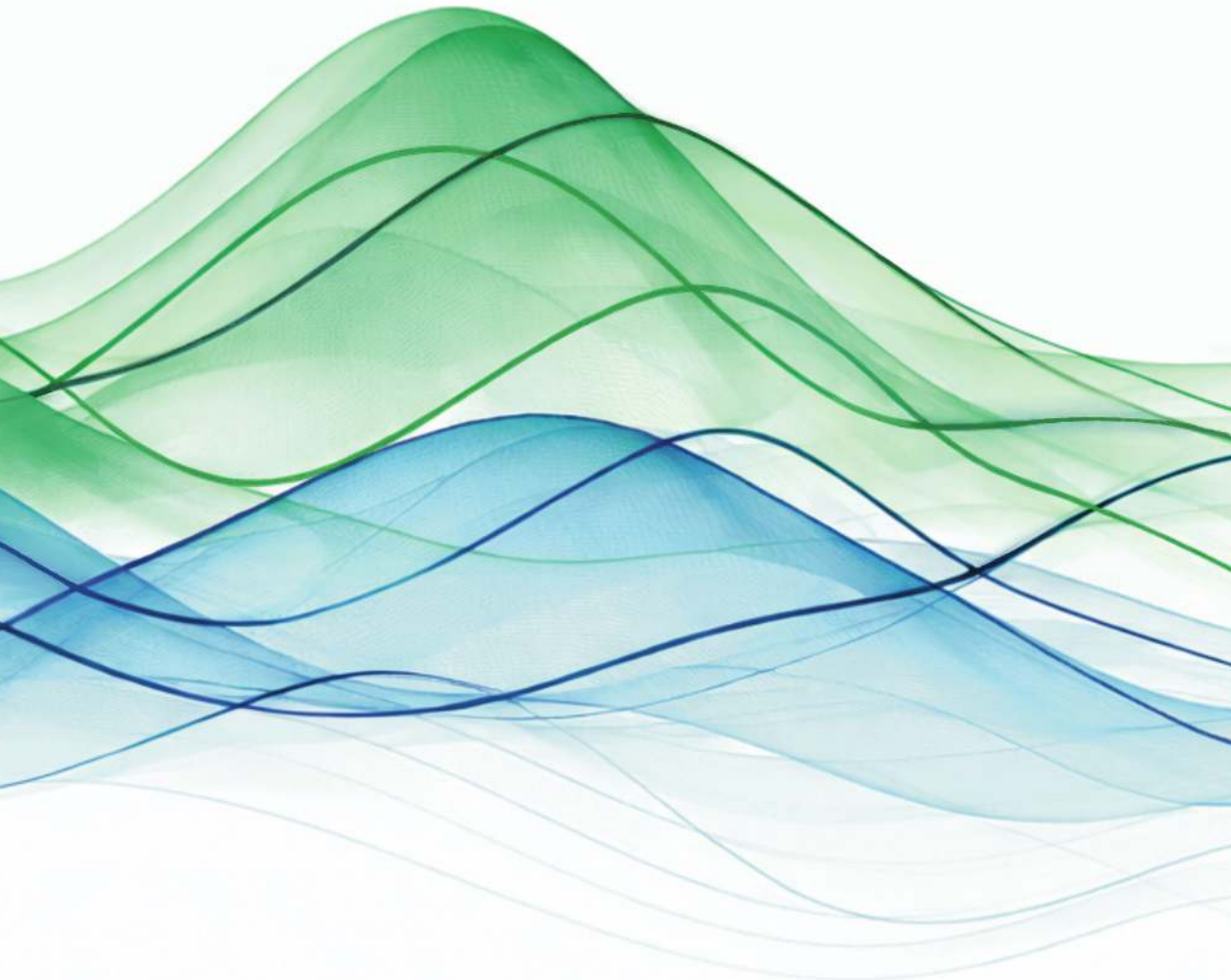
Game-Theoretic Motion Planning for Multi-Agent Interaction



Lasse Peters

Game-Theoretic Motion Planning for Multi-Agent Interaction

Game-Theoretic Motion Planning for Multi-Agent Interaction

Dissertation

for the purpose of obtaining the degree of Doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof.dr.ir. H. Bijl,
chair of the Board of Doctorates,
to be defended publicly on
Thursday, 26 March 2026 at 10:00 o'clock

by

Lasse PETERS

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	Delft University of Technology, *Chairperson*
Prof.dr. J. Alonso-Mora,	Delft University of Technology, *Promotor*
Dr. L. Ferranti,	Delft University of Technology, *Promotor*
Independent members:	
Prof.dr. M.T.J. Spaan,	Delft University of Technology
Dr.ing. S. Grammatico,	Delft University of Technology
Dr. J. Tumova,	KTH Royal Institute of Technology, Sweden
Dr. G.A. Sartoretti,	National University of Singapore, Singapore
Prof.dr. R. Babuška,	Delft University of Technology, *Reserve member*

Keywords: Game Theory, Motion Planning, Multi-Agent Interaction, Robot Learning

Printed by: Ridderprint | `www.ridderprint.nl`

Cover: Sıla Akçakoca (realization) and Lasse Peters (ideation)

Style: TU Delft House Style, with modifications by Moritz Beller
`https://github.com/Inventitech/phd-thesis-template`

ISBN/EAN: 978-94-6518-259-9 (ebook/pdf)
978-94-6384-923-4 (paperback)

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

If you're thinking without writing, you only think you're thinking.

Leslie Lamport

Contents

Summary

Robots are leaving factory floors and entering human environments, where they must move safely and efficiently while interacting with people and other autonomous systems. In these settings, decisions are interdependent: what a robot should do depends on how others will respond, and vice versa. Anticipating and shaping these responses is central to competent behavior in multi-agent settings, but it is difficult in practice because interaction unfolds under uncertainty, with limited prior knowledge of other agents' objectives and limited sensing, and often under tight computational constraints.

Because decisions made in interaction with others are often safety-critical, receding-horizon online optimization is widely adopted for its ability to optimize performance while enforcing constraints such as collision avoidance. Established approaches of this category typically decouple prediction and planning: they start by predicting the future motion of other agents (for example, by assuming constant velocity or extrapolating learned motion patterns) and then plan the robot's actions based on those predictions. While simple, this predict-then-plan paradigm cannot capture the mutual influence of agents: predictions ignore that the robot's future actions will affect how others behave. More advanced approaches go beyond this paradigm by casting interaction as *joint* decision-making via a single optimization problem. While this captures the interdependence of future decisions between agents, it implicitly forces them to share a common objective and thus cannot express strategic trade-offs, nudging, blocking, or other elements of competition.

This dissertation addresses these challenges by studying robot motion planning for interactive scenarios through the lens of game theory, which provides a principled language for modeling multiple *self-interested* decision makers who act *simultaneously* and whose objectives may only be *partially aligned*. Our focus is on settings in which we control a single robot that interacts with one or more other uncontrolled agents, be they humans or other robots, with particular attention to cases where the robot does not know other agents' intents a priori. For these settings, we develop a set of tools, grounded in game-theoretic principles, that allow a robot to understand the intents of other agents and enable it to generate motion plans that capture the *interdependence* of decisions between agents and their *self-interested* decision-making. We showcase the effectiveness of these tools in applications such as autonomous driving, mobile navigation, and multi-agent manipulation. The main contributions of this dissertation are as follows.

First, we study the problem of learning the unknown intents of other agents from observed past behavior. We formulate this so-called *inverse game problem* as maximum-likelihood estimation with equilibrium constraints and propose a transcription solvable by off-the-shelf optimization solvers. This resulting method jointly estimates game parame-

ters, current states, and future decisions of other agents, improving inference accuracy.

Second, we integrate inverse games with online decision-making so that a robot can adapt its strategy as estimates of others' intents evolve. To this end, we propose a new solution technique for the inverse game problem that handles inequality constraints, enabling safer interaction, and provides a first-order update rule that integrates naturally with neural network training for amortized inference. This yields a game-theoretic motion planner that adapts online to estimated player intents as observations become available.

Third, we study settings in which the robot has access not to a point estimate of other players' intents but explicitly reasons about a distribution over possible intents. In this context, we propose *contingency games*, an uncertainty-aware planning technique that jointly produces a *multi-hypothesis* prediction of others and a corresponding *conditional* plan for the robot. By estimating the future time at which uncertainty will be resolved, this approach *anticipates* future information gains. We show that this formulation generalizes prior approaches that either ignore uncertainty or conservatively assume that uncertainty will never be resolved. As a result, contingency games offer a middle ground between these two extremes, discovering efficient and safe interaction strategies.

Fourth, we turn to the problem of reducing the computational burden of solving game-theoretic motion planning problems online. We study this problem in the context of games in which agents are not forced to commit to a single deterministic trajectory, but may choose a distribution of trajectories, so-called *mixed* strategies. In this context, we introduce an amortized solver that learns, offline, to propose a small set of low-cost, dynamically feasible trajectory candidates per agent. Online, a discrete game is solved over these candidates to compute a mixed Nash equilibrium. The resulting solver rapidly finds feasible mixed strategies that are competitive.

Fifth, we study an interaction problem with inherently non-smooth dynamics, using the example of multi-agent manipulation. This interaction problem violates a fundamental assumption made in the majority of prior work in interactive motion planning (including those presented in this dissertation): the assumption that costs and constraints (including dynamics) are differentiable. To address this challenge, we propose a data-driven approach that combines learning from *single-agent demonstrations* with reasoning about joint costs across agents. By studying this problem through the lens of probabilistic inference and generative diffusion models, we construct a generative process of the coordinated, *joint* policy that directly leverages the pre-trained single-agent policies as a prior. This approach discovers collaborative strategies that are absent from the single-agent demonstrations while avoiding the logistical burden of providing large amounts of multi-agent data.

In summary, this thesis advances interactive motion planning by taking a game-theoretic perspective, allowing a robot to infer the intents of others, account for uncertainty in these estimates during planning, and solve challenging instances of multi-agent interaction through a mix of offline learning and online reasoning. All contributions are validated extensively in simulation of tasks such as autonomous driving, mobile navigation, and multi-agent manipulation, with selected methods additionally demonstrated on a mobile ground robot. Among the supporting contributions of this thesis are several open-source software packages that transcribe motion planning and intent inference problems into forward and inverse games and solve them efficiently. The code and data for most chapters have been made publicly available to accelerate future research.

Samenvatting

Robots verlaten de fabrieksvloer en betreden menselijke omgevingen, waar zij zich veilig en efficiënt moeten verplaatsen en interacties aangaan met mensen en andere autonome systemen. In zulke omgevingen zijn beslissingen onderling afhankelijk: wat een robot zou moeten doen, hangt af van hoe anderen zullen reageren, en omgekeerd. Het kunnen anticiperen op en sturen van deze reacties is essentieel voor competent gedrag in multi-agent omgevingen, maar in de praktijk is dit uitdagend omdat interactie gebeurt onder onzekerheid, met beperkte voorkennis over de doelen van andere agenten en beperkte waarneming, en vaak onder strikte computationele beperkingen.

Omdat beslissingen die in interacties met anderen worden genomen vaak veiligheidskritisch zijn, wordt online optimalisatie met verschuivende horizon veel gebruikt vanwege het vermogen om prestaties te optimaliseren, terwijl tegelijk beperkingen zoals het vermijden van botsingen worden afgedwongen. Gangbare methodes in deze categorie ontkoppelen doorgaans het voorspellen en het plannen: zij beginnen met het voorspellen van de toekomstige beweging van andere agenten (bijvoorbeeld door een constante snelheid aan te nemen of door aangeleerde bewegingspatronen te extrapoleren) en plannen vervolgens de acties van de robot op basis van die voorspellingen. Hoewel eenvoudig, kan dit *predict-then-plan*-paradigma de wederzijdse beïnvloeding tussen agenten niet beschrijven: de voorspellingen negeren dat de toekomstige acties van de robot van invloed zullen zijn op hoe anderen zich gedragen. Meer geavanceerde methodes gaan verder dan dit paradigma door interactie te formuleren als *gezamenlijke* besluitvorming via één enkel optimalisatieprobleem. Hoewel dit de onderlinge afhankelijkheid van toekomstige beslissingen tussen agenten vastlegt, dwingt het impliciet af dat zij een gemeenschappelijke doelstelling delen en kan het daardoor geen strategische afwegingen, subtiele beïnvloeding, blokkeren, of andere elementen van competitie uitdrukken.

Dit proefschrift pakt deze uitdagingen aan door bewegingsplanning voor robots in interactieve scenario's te bestuderen door de lens van de speltheorie, die een principiële taal biedt om meerdere *op eigenbelang gerichte* besluitvormers te modelleren die *gelijktijdig* handelen en van wie de doelstellingen slechts *gedeeltelijk op elkaar afgestemd* kunnen zijn. Onze focus ligt op scenario's waarin we één robot besturen die in interactie staat met één of meerdere niet door ons bestuurde agenten, die mensen of andere robots kunnen zijn, met bijzondere aandacht voor gevallen waarin de robot de intenties van andere agenten niet a priori kent. Voor deze scenario's ontwikkelen we een set hulpmiddelen, gegrond in speltheoretische principes, die een robot in staat stellen de intenties van andere agenten te begrijpen en bewegingsplannen te genereren die de *onderlinge afhankelijkheid* van be-

slissingen tussen agenten en hun *op eigenbelang gerichte* besluitvorming vastleggen. We tonen de effectiviteit van deze hulpmiddelen aan in toepassingen zoals autonoom rijden, mobiele navigatie en manipulatie met meerdere agenten. De belangrijkste bijdragen van dit proefschrift zijn als volgt.

Ten eerste bestuderen we het probleem om de onbekende intenties van andere agenten te leren uit waargenomen gedrag in het verleden. We formuleren dit zogeheten *inverse spelprobleem* als maximum-likelihood schatting met evenwichtsrestricties en stellen een transcriptie voor die oplosbaar is met standaard optimalisatiesolvers. De resulterende methode schat gezamenlijk spelparameters, huidige toestanden en toekomstige beslissingen van andere agenten, wat de inferentienauwkeurigheid verbetert.

Ten tweede integreren we inverse spellen met online besluitvorming zodat een robot zijn strategie kan aanpassen terwijl schattingen van de intenties van anderen evolueren. Daartoe ontwikkelen we een nieuwe oplossingstechniek voor het inverse spelprobleem die ongelijkheidsrestricties aankan, wat veiligere interactie mogelijk maakt, en een updateregel van eerste orde biedt die op natuurlijke wijze integreert met neurale-netwerktraining voor geamortiseerde inferentie. Dit leidt tot een speltheoretische bewegingsplanner die zich online aanpast aan geschatte spelersintenties naarmate observaties beschikbaar komen.

Ten derde bestuderen we scenario's waarin de robot niet over een puntschatting van de intenties van andere spelers beschikt, maar expliciet redeneert over een verdeling van mogelijke intenties. In deze context stellen we *contingency games* voor, een onzekerheidsbewuste planningsmethode die gezamenlijk een *multi-hypothese*-voorspelling van anderen produceert en een overeenkomstig *conditioneel* plan voor de robot. Door de toekomstige tijd te schatten waarop onzekerheid zal worden opgelost, *anticipeert* deze aanpak op toekomstige informatiewinsten. We laten zien dat deze formulering eerdere methodes generaliseert die ofwel onzekerheid negeren, ofwel conservatief aannemen dat onzekerheid nooit zal worden opgelost. Daardoor bieden *contingency games* een middenweg tussen deze twee extremen en vinden ze efficiënte en veilige interactiestrategieën.

Ten vierde richten we ons op het verminderen van de computationele last van het online oplossen van speltheoretische bewegingsplanningsproblemen. We bestuderen dit probleem in de context van spellen waarin agenten niet gedwongen zijn zich vast te leggen op één deterministisch traject, maar een verdeling over trajecten kunnen kiezen, zogeheten *gemengde* strategieën. In deze context introduceren we een geamortiseerde solver die offline leert om per agent een kleine set goedkope, dynamisch haalbare trajectkandidaten voor te stellen. Online wordt vervolgens een discreet spel over deze kandidaten opgelost om een gemengd Nash-evenwicht te berekenen. De resulterende solver vindt snel haalbare gemengde strategieën die competitief zijn.

Ten vijfde bestuderen we een interactieprobleem met inherent niet-gladde dynamica, met als voorbeeld manipulatie met meerdere agenten. Dit interactieprobleem schendt een fundamentele aanname in het merendeel van eerder werk over interactieve bewegingsplanning (inclusief de eerdergenoemde methodes in dit proefschrift): de aanname dat kosten en restricties (inclusief dynamica) differentieerbaar zijn. Om deze uitdaging op te lossen, stellen we een data-gedreven aanpak voor die leren uit *demonstraties van één agent* combineert met redeneren over gezamenlijke kosten over alle agenten. Door dit probleem te bestuderen door de lens van probabilistische inferentie en generatieve diffusiemodellen,

construeren we een generatief proces van het gecoördineerde, *gezamenlijke* beleid dat direct gebruikmaakt van de vooraf getrainde beleidsregels voor individuele agenten als prior. Deze aanpak vindt collaboratieve strategieën die afwezig zijn in de demonstraties van één agent, terwijl het de logistieke last van het aanleveren van grote hoeveelheden multi-agent data vermijdt.

Samengevat bevordert dit proefschrift interactieve bewegingsplanning door een speltheoretisch perspectief te nemen, waardoor een robot de intenties van anderen kan afleiden, onzekerheid in deze schattingen kan meenemen tijdens het plannen, en uitdagende instanties van multi-agent interactie kan oplossen via een combinatie van offline leren en online redeneren. Alle bijdragen worden uitgebreid gevalideerd in simulatie van taken zoals autonoom rijden, mobiele navigatie en manipulatie met meerdere agenten, met geselecteerde methoden die bovendien zijn gedemonstreerd op een mobiele grondrobot. Onder de ondersteunende bijdragen van dit proefschrift bevinden zich verschillende open-source softwarepakketten die bewegingsplanning- en intentie-inferentieproblemen omzetten in voorwaartse en inverse spellen en deze efficiënt oplossen. De code en data voor de meeste hoofdstukken zijn publiek beschikbaar gemaakt om toekomstig onderzoek te versnellen.

This Dutch summary was translated from the English version above with the help of Opus 4.6 [1] and Saray Bakker.

Chapter 1

Introduction

Robots are becoming cheaper, more available, and lighter [2–4]. These developments set the stage for a new era of robotics: one in which they are no longer bound to factory floors but can aid us in our everyday lives. In fact, this revolution has already begun: vacuum-cleaning robots are a common sight in many households [5], mobile service robots are starting to be deployed in the hospitality sector [6], and the first open-road tests of self-driving cars are underway in several cities across the world [7].

In view of these advances, it becomes increasingly clear that general autonomous robots provide a unique opportunity to improve overall welfare and promise to solve some of the most pressing challenges facing society. For example, autonomous robots offer promising technological solutions to address labor shortages posed by an aging population in many developed countries [8]. Furthermore, applications such as autonomous ride sharing, which eliminate the need for individuals to own and operate their own vehicles, have the potential to reduce traffic congestion, lower emissions, and decrease the number of accidents. Nonetheless, progress has been largely confined to domains characterized by abundant data to inform system design decisions and high potential returns that justify extensive engineering investments in domain-specific solutions, such as autonomous driving. General-purpose algorithms that enable robots to autonomously act across a wide range of domains to assist us in our everyday lives remain out of reach.

To handle such complex tasks, much like humans, robots must exploit prior knowledge, *sense* the world to *learn* from experience, and *adapt* their decision-making in order to move safely and efficiently through the world—or even to change (aspects of) the world itself. In short, they must have *agency*. Therefore, in this dissertation, we will use the term *agent* broadly to refer to any entity—human or robot—that has the capacity to act in the world by moving through it or otherwise changing it.

From Isolation to Interaction. Navigating *in* and interacting *with* a complex environment is inherently a challenging task. Beyond sensing the *current* state of the world, it requires understanding the "rules" that dictate how the world evolves in order to make predictions about its *future*. That is, it requires understanding which actions lead to which outcomes. When agents act in solitude, these "rules" of the world reduce to *laws of physics*—

1

the set of mathematical equations that describe the *dynamics* of the world. For many physical phenomena, these "rules" are well-understood: we have accurate models that capture effects ranging from the electro-mechanical dynamics of motors [9] to the behavior of entire vehicles, whether on the ground [10], on water [11], or in the air [12].

As soon as we introduce another agent into the same world—even if just *one* other agent—the set of "rules" becomes substantially more complex. In this setting, the "rules" that govern the robot's environment include not only the laws of physics but also the more subtle, implicit, and unwritten principles that govern the decisions of other agents. Hence, the added complexity is driven by the following, seemingly innocuous question that poses itself to the robot:

How will other agents react to the robot's actions?

As we bring robots closer into our lives, it is this question that we must confidently answer to ensure a robot's efficiency without compromising safety, particularly when the health and safety of humans are at stake. This dissertation seeks to provide robots with principled tools to answer this question.

This Dissertation. The focus of this dissertation is on settings where we control a *single* robot—also referred to as the "ego agent"—that must plan its motion to physically interact with one or more other agents, whether humans or other robots, without explicit communication or trust established between them. We will study settings ranging from full cooperation to full competition and pay special attention to settings where other agents' true intent and willingness to cooperate are not clear to the ego agent a priori.

The primary lens through which we will study these interactions is that of *game theory*. Beyond this main tool, key concepts underpinning this dissertation are those of optimization, receding-horizon reasoning, and learning from demonstration. We shall provide a brief introduction to these key concepts in Sections 1.1 and 1.2.

Throughout this dissertation, we develop tractable and effective methods grounded in game-theoretic principles for online motion generation in interactive settings and show how the proposed techniques enable robots to reason about and respond to the actions of other agents, facilitating safe and efficient interaction.

A Thought Experiment of Multi-Agent Interaction. Before we dive into a more technical discussion of multi-agent interaction, let us first consider a thought experiment to illustrate the challenges characterizing this class of problems.

If the reader happens to be human—rather than, e.g., a language model—they may have a good intuition for this more opaque set of "rules" that governs interactions with others.[1] In fact, we humans are remarkably skillful at navigating many of these scenarios, often without actively thinking about it. To illustrate this point, let us consider the unsignalized intersection scenario in Figure 1.1 to conduct a thought experiment. Imagine that this were an unfamiliar intersection, perhaps even in another country governed by different traffic laws and driving habits than you are used to. Even under those challenging circumstances, you would likely be able to quickly adapt to this scenario, rapidly inferring the "rules" of

[1]Some of this intuition is now slowly becoming available to language models as well [13], but it remains unclear how to make use of that knowledge for decision-making.

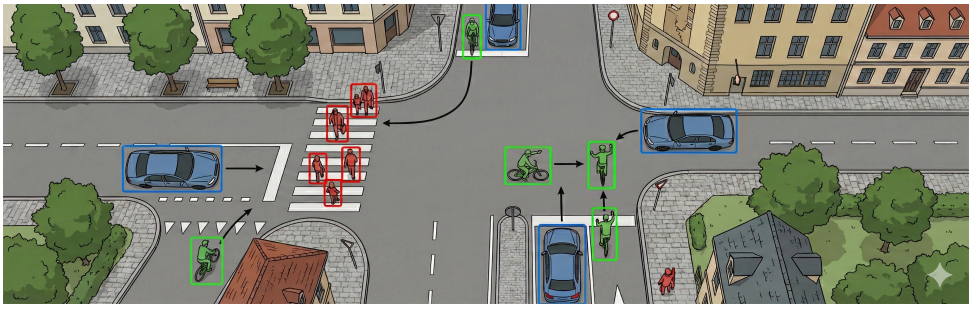


Figure 1.1: A typical multi-agent interaction scenario. *Note:* Image generated with Google Gemini [14].

the new environment while accounting for uncertainty along the way by adopting a more cautious driving strategy. In doing so, you would implicitly answer many layers of our key question above; e.g., Where are others trying to go? What are their capabilities and how fast can they go? What do *they* know? Did they even see me? What social conventions do they follow?

In this dissertation, we will formalize this kind of reasoning through the lens of game theory; an introduction to which is provided in the following section.

1.1 Interaction as a Game

Game theory provides a powerful tool for modeling multi-agent interaction because it captures the *simultaneous* yet *interdependent* nature of decision-making among multiple self-interested agents.

More formally, game theory studies how several agents, also called *players*, choose their individual *strategies*—plans of action—to achieve their individual *objectives*. These *objectives* are encoded by a *cost function*, which measures how good or bad an outcome is from the perspective of that player. Hence, players seek to choose a strategy that minimizes their own *cost function*.

For example, in the setting of Figure 1.1, a player's strategies may dictate their acceleration, braking and steering commands and their cost function may assign lower costs to decisions that satisfy traffic rules while driving efficiently. Importantly, however, a player's cost depends not only on their own actions, but also on the actions of other players. For example, the cost of entering the intersection may be higher if that would require another player to brake aggressively or change their trajectory.

Beyond soft *incentives*, a game can also capture *hard constraints* between players' strategies; such as disallowing two players to occupy the same position at the same time.[2] The dependence of a player's cost and available strategies on the decisions of other players is what makes games both powerful and challenging to solve: their solutions naturally give rise to *strategic* interactions, but finding those solutions requires, in general, reasoning about all players' strategies jointly.[3]

[2]In other words: "vehicles must not collide".

[3]This point is in contrast to more traditional approaches in the field that first *predict* the decisions of other play-

1

An Odd Game? The term "game" and vocabularies such as "player" may prompt the reader to think of "table-top games", such as chess, go, or poker. This association is not wrong, in that, much like the games studied in this dissertation, "success" in all of these settings requires strategic reasoning and careful consideration of other players' future (and, sometimes, even past) moves. These table-top games are characterized by

- finite, discrete actions: all players have a countable number of actions to choose from at every turn;
- sequential play: players take turns making their decisions;
- zero-sum cost-structure: one player's gain is another player's loss;
- and, fully known rules: all players know the rules of the game and the cost that others seek to minimize.

By contrast, the games underpinning the physical multi-agent interactions that we study in this work are characterized by:

- infinite, continuous actions: all players have an uncountable number of actions to choose from at every turn;
- simultaneous play: players make their decisions at the same time;
- general-sum cost-structure: one player's gain is not necessarily another player's loss;
- and, partially unknown rules: players may not know all aspects of the game, such as the cost that others seek to minimize.

What's more: there is typically no binary "winner" or "loser" in the games studied here. Rather, performance falls onto a continuous scale. For example, measuring a combination of safety (e.g., clearance between vehicles) and efficiency (e.g., fuel consumption and time to reach the goal) in the example of Figure 1.1.

1.2 Related Methods and Key Concepts

The contributions of this dissertation stand in close relation to several key concepts beyond game theory. Here, we discuss three cornerstone concepts that appear repeatedly throughout this chapter: *optimization*, *receding-horizon reasoning*, and *learning from demonstration*. We shall use this opportunity to highlight how they relate to the game-theoretic perspective on interaction and why established instantiations of these tools fall short of solving the challenges of multi-agent interaction highlighted above. Note that the treatment below is deliberately held non-technical to make it accessible to a broad audience. Each following chapter below includes a more detailed review of the background material relevant to understanding the contributions of that chapter.

1.2.1 Optimization

Games can be understood as a network of coupled optimization problems [16]. As such, the vocabulary and tools of optimization play an important role in this dissertation. Conventional optimization problems are concerned with choosing the *best* decision from a set of alternatives. Note that, a "decision" in that context can still be multi-dimensional—e.g.,

ers and then plan their own actions, ignoring interdependence of future actions [15]—more on this difference in Section 1.2.

assigning values for steering, acceleration, and braking commands simultaneously in the example of Figure 1.1. However, the way in which "best" is measured is only with respect to a *single* cost function against which all aspects of the decision are evaluated.

This is the key difference compared to games, where each player has their *own* cost function. Therefore, in a game, the notion of "best"—as perceived by all players jointly—is no longer well-defined, since "what is best" for one player may not be best for another. As a result, in contrast to optimization problems, games are concerned with finding *equilibria*: points at which no player can *unilaterally* reduce their cost by changing their strategy. By virtue of this subtle difference, games are fundamentally more powerful than optimization problems. In particular, games *can* capture aspects such as collaboration between players when both are sensitive to the same cost component—e.g., two players trying to avoid collision in the example of Figure 1.1—but they are equally capable of capturing partial, or even complete, competition between players—e.g., when a driver prioritizes their own progress over that of others. By contrast, if we were to model the same interaction as a joint optimization, all agents are *forced* to collaborate in a way that minimizes the *overall* cost.

1.2.2 Receding-Horizon Reasoning

Many interactions of interest in robotics unfold over an extended time window. For example, returning to the driving scenario of Figure 1.1, an individual driver in this scene may be on their way to a target destination that is several minutes, perhaps even hours, away. However, it is not practical to plan over this entire time window at once considering the full fidelity of interactions one will encounter along the way.

The Role of Receding-Horizon Reasoning for Online Decision-Making Rather than considering the entire time window at once, it is common for a robot to optimize decisions only over the next, say, 20 seconds in the immediate future, execute a portion of that plan, and then re-plan over the next 20-second window. Beyond keeping complexity manageable, this *receding-horizon* approach to decision-making takes on another important role: it allows the robot to dynamically *adapt* to any unforeseen changes in the world. After all, the model that the robot uses to predict the outcomes of their decisions is only an *approximation* of reality. As such, the robot must constantly *observe* the true outcome (i.e., the current state of the world) to ground its upcoming decisions in reality, thereby avoiding *divergence*. One of the most popular instantiations of this idea of receding-horizon reasoning is the paradigm of model predictive control (MPC) [17].

Challenges in Multi-Agent Settings. When applied to motion planning for multi-agent interaction, receding-horizon planning requires making predictions about what *other agents* will do in the immediate future. MPC approaches typically resort to first predicting the future decisions of others—often with models as simple as assuming constant velocity [15]—and then plan the robot's actions in response to those predictions. While simplifying implementation, these "predict-then-plan" approaches preclude modeling *influence* of the robot on other agents' future decisions. More advanced instantiations of MPC go beyond a "predict-then-plan" approach by modeling the *joint* decision of

all agents as a single optimization problem [18]. This approach, however, requires all agents to share the *same objective*, implicitly assuming trust and collaboration. Hence, phenomena such as nudging or blocking do not naturally emerge from these approaches.

Extending Receding-Horizon Reasoning to Games. This dissertation applies the receding-horizon approach to multi-agent interaction but, rather than deriving future plans from an optimization problem, it derives them from a *game* over the immediate future [19]. That is, at every time step, the robot constructs a game-theoretic model of the interaction in the immediate future and uses the resulting equilibrium strategies for two purposes: to predict the future decisions of others and, *simultaneously*, to find its own optimal strategy. Through this coupled reasoning about predictions and plans, the robot ensures that their predictions of others' decisions are *consistent* with its own future decisions. We will refer to this approach as model-predictive game play (MPGP).

1.2.3 Learning from Demonstration

Another key concept relevant to this dissertation is learning from demonstration; i.e., learning from observed decisions. While this thesis contends that directly learning the robot's own strategy in this way is inapplicable (since we require competent and safe behavior across diverse situations that may not be foreseen when giving demonstrations), we build on these ideas in two different ways. First, to learn models of *others'* future behavior from their past decisions (Chapters 2 and 3); and second, to accelerate the search for the robot's own strategy (Chapter 6).

Two strands of learning from demonstration popularized in the *single-agent* domain are particularly relevant to this dissertation: inverse optimal control (IOC) [20–22] and generative diffusion policies learned from demonstration [23–26]. These approaches differ in how they represent policies and structure the learning process. IOC assumes that observed behavior is the result of solving an optimal control problem with unknown parameters (e.g., unknown aspects of the cost or dynamics). Generative diffusion policies, by contrast, impose less structure: observed behavior is treated as samples from an unknown distribution, not tied to any particular objective. Due to the imposed structure, IOC is often sample-efficient, amenable to online inference, and offers an interpretable model of the observed behavior, but it is less expressive than diffusion policies [21, 27, 28]. The improved expressiveness of diffusion policies, on one hand, comes at the cost of requiring much more data and extensive offline training [23–26].

Taken alone, neither of these methods suffices for our setting. IOC enables online inference of a player's objective (encoded by their cost function); however, it does not extend to the non-cooperative settings considered in this dissertation, where players may have partially conflicting objectives. Diffusion policies provide a powerful framework for learning a robot's policy but are prohibitively data-hungry in the multi-agent setting. We build on both paradigms and combine them with game-theoretic reasoning to address these limitations.

1.3 Contributions and Outline

This dissertation contributes a set of tools—grounded in game-theoretic principles—that enable a robot to interact with other agents by (i) inferring their objectives, (ii) dealing with uncertainty in their intent, (iii) solving game-theoretic formulations of interaction efficiently, and (iv) utilizing single-agent demonstrations to learn complex policies.

Details on each contribution are listed below. Since the dissertation dedicates a separate chapter to each of these contributions, we shall use this opportunity to also clarify the structure of this document. This structure is additionally summarized in Figure 1.2. The contributions of this dissertation are:

(i) **A maximum-likelihood formulation of inverse games**, which allows a robot to learn other agents' objectives from observations of their past behavior. Unlike prior approaches, this formulation explicitly accounts for the fact that the robot receives only partial and potentially noise-corrupted observations. Two methods are proposed to solve this inverse game problem:

   (a) **Chapter 2: A constrained-optimization approach to inverse games** that transcribes the inverse game into a standard constrained optimization problem by maximizing observation likelihood while enforcing the first-order necessary conditions of the game as constraints. This transcription allows one to solve the inverse game with off-the-shelf optimization solvers. Techniques for both offline and online learning are proposed. Simulation results across several simulated traffic scenarios show that this method reliably estimates game-theoretic models from noise-corrupted data that closely match ground-truth objectives, consistently outperforming state-of-the-art approaches.

   (b) **Chapter 3: An implicit differentiation approach to inverse games** that computes the gradient of the observation likelihood by analyzing the sensitivity of the game's solution to changes in its parameters. This gradient signal facilitates first-order optimization for parameters that explain the observed behavior. Beyond online learning, this approach enables amortized inference by training an neural network (NN) offline to predict game parameters *directly* from observation histories. This chapter evaluates both parameter estimation accuracy and the closed-loop performance of a game-theoretic motion planner using these estimates. In simulated traffic scenarios, this method outperforms prior game-theoretic and non-game-theoretic approaches in terms of safety and interaction efficiency. This chapter also showcases the real-time planning capabilities and robustness of the method through hardware experiments involving interactions between mobile robots and pedestrians.

(ii) **Chapter 4: A game-theoretic approach to contingency planning** that allows a robot to jointly recover a multi-hypothesis prediction of others and a corresponding conditional plan for itself. By estimating the future time at which uncertainty will be resolved, this approach *anticipates* future information gains. This approach formalizes a middle-ground between prior game-theoretic approaches that either ignore uncertainty or conservatively assume that uncertainty will never be resolved,

recovering these prior approaches as a special case. Simulation results show that robots using this approach match the more conservative approaches in terms of safety while outperforming them in terms of interaction efficiency.

(iii) **Chapter 5: An amortized solver for trajectory games** that addresses the computational challenges of online planning in games with large strategy spaces by introducing an offline training phase. This chapter develops the approach in the context of games in which agents have access to a richer class of *mixed*—i.e., strategically randomized—strategies, allowing them to commit to a *distribution* of trajectories rather than a single deterministic plan. Simulation results of the pursuit-evasion game "tag" demonstrate the mixed strategies found by this approach provide a competitive advantage. This chapter also showcases that this approach facilitates rapid training from scratch via simulated self-play.

(iv) **Chapter 6: A technique for learning multi-agent policies while leveraging single-agent demonstrations** that applies to problems with inherently non-smooth interactions (e.g., problems with contact such as multi-agent manipulation). The method pretrains single-agent policies from demonstrations of basic skills and composes them into a coordinated multi-agent policy by reasoning about joint costs using generative diffusion models. In high-fidelity simulations of a two-robot manipulation task, we demonstrate that this method learns coordinated behavior from single-agent data, discovers collaborative strategies absent from the single-agent demonstrations, and achieves more efficient and accurate policies than a baseline trained on multi-agent demonstrations with the same data budget.

Following the presentation of these main contributions in Chapters 2 to 5, Chapter 7 concludes the dissertation with final remarks and outlines directions for future research.

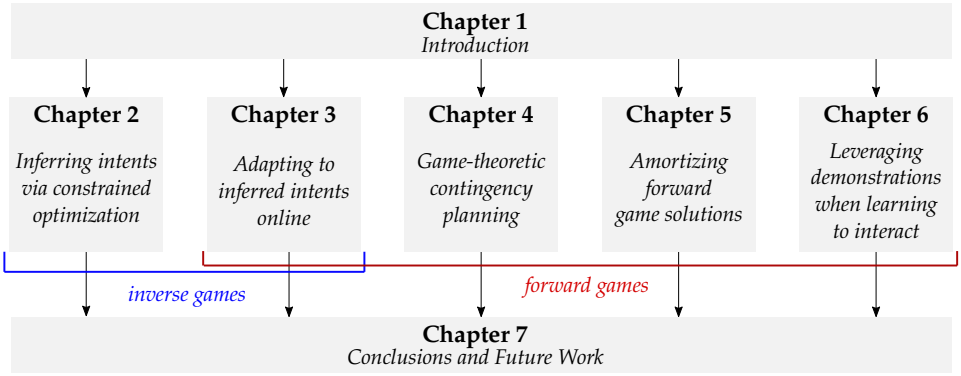


Figure 1.2: Structure of this dissertation.

1.4 Contributions Beyond this Dissertation

Beyond the contributions summarized in Section 1.3, concurrent research efforts and collaborations have produced several noteworthy contributions relevant to the topics covered in this dissertation.

1.4.1 Co-Authored Publications

Page 157 provides a complete list of publications, including those led by other authors. Below is a brief overview of the most relevant co-authored publications and their connection to this dissertation.

Additional Approaches to Inverse Games. Chapters 2 and 4 propose maximum-likelihood solutions for inverse games under open-loop information structures. Two co-authored works advance this approach. Work [29] extends the approach of Chapter 4 to games with feedback information structures; i.e., games that explicitly account for the fact that players receive additional information about the state in the future and can react accordingly. The follow-up work [30] further extends the approach of Chapter 3 beyond maximum-likelihood-based point estimates, instead inferring *distributions* of players' intents in a Bayesian framework. By integrating with the game-theoretic contingency planner of Chapter 4, this provides an example of how inverse game techniques can be incorporated into the contingency games paradigm.

Accelerating the Solution of Multi-Hypothesis Games. As we shall see in Chapter 4, contingency games result in powerful but also computationally challenging equilibrium problems. To scale this paradigm to settings with a large number of hypotheses, [31] proposes a specialized ADMM solver that parallelizes computation across hypotheses.

Safety Filters. When interacting with the world and other agents, robots must ensure that their actions are safe. In this work, safety is primarily addressed in terms of avoiding collisions with other agents. In more complex scenarios, however, collision avoidance is not the only aspect of safety. Two lines of work explore this aspect in more detail by filtering—i.e., overwriting—robot actions that violate a *semantic* notion of safety that goes beyond collision avoidance. In [32], a vision-language model (VLM) [33] interprets natural language feedback and the robot's image observations to continuously update the robot's representation of safety constraints. In [34], a latent-space generalization of Hamilton-Jacobi (HJ) reachability is proposed that enables robots to detect and avoid complex, hard-to-specify hazards—such as spilling or toppling objects—directly from high-dimensional observations like images.

1.4.2 Software

For each of the methods presented in the following chapters, this dissertation produced an open-source reference implementation. These reference implementations are built on a set of core abstractions, which were distilled into standalone software

packages and are now hosted under the GitHub organization `https://github.com/JuliaGameTheoreticPlanning`. These packages are briefly summarized below:

- **TrajectoryGamesBase.jl** [35] provides a common interface to describe trajectory games. The solvers in Chapters 3 to 5 all make use of this interface.
- **ParametricMCPs.jl** [36] provides an abstraction on top of the PATH solver [37] to automatically synthesize Mixed Complementarity Problems (MCPs) as a function of user-defined parameters. It allows automatic differentiation of the result, providing the core component for forward and inverse game solutions, including those presented in Chapters 3 and 4.
- **MCPTrajectoryGameSolver.jl** [38] automates the transcription of trajectory games into MCPs and, by building on top of ParametricMCPs.jl, implicitly facilitates differentiation of the solution with respect to game parameters.
- **DifferentiableTrajectoryOptimization.jl** [39] provides an abstraction on top of established optimization solvers such as IPOPT [40] and OSQP [41] to automatically synthesize trajectory optimization problems from user-defined costs and dynamics. By implementing back-propagation rules, this package facilitates automatic differentiation of the solution map, allowing for integration with machine-learning pipelines and providing the backbone for Chapter 5.

Chapter 2

Estimating Player Objectives from Partially Observed Interactions

Robots deployed to the real world must be able to interact with other agents in their environment. Game theory provides a powerful mathematical framework for modeling scenarios in which agents have individual objectives and interactions evolve over time. However, a key limitation of such techniques is that they require a-priori knowledge of all players' objectives.

This chapter proposes a novel method for learning players' objectives in continuous games from noise-corrupted, partial state observations. Our approach learns objectives by coupling the estimation of unknown cost parameters of each player with inference of unobserved states and inputs through Nash equilibrium constraints. By coupling past state estimates with future state predictions, our approach is amenable to simultaneous online learning and prediction in receding-horizon fashion.

This chapter is a verbatim copy, with minor modifications, of the peer-reviewed journal article [42]:

Lasse Peters, Vicenç Rubies-Royo, Claire J. Tomlin, Laura Ferranti, Javier Alonso-Mora, Cyrill Stachniss, David Fridovich-Keil. "Online and Offline Learning of Player Objectives from Partial Observations in Dynamic Games." *International Journal of Robotics Research (IJRR)*, 2023.

This journal article extends the earlier conference version of this work [43].

Contribution statement: Lasse developed and implemented the proposed method and conducted all experiments and evaluations. Lasse and David jointly proposed the problem formulation and wrote the initial draft of the manuscript. All authors contributed to technical discussions and edits of the original paper submitted to IJRR [42].

2.1 Introduction

To operate safely and efficiently in environments shared with other agents, robots must be able to predict the effects of their actions on the decisions of others. In many such settings, agents do not form a single team that shares a joint objective. Instead, each agent may have an individual objective, encoded by a cost function which they optimize unilaterally. Unless the objectives of all agents are perfectly aligned, agents must therefore compete to minimize their own cost, while accounting for the strategic behavior of others. For example, consider the highway navigation scenario in Figure 2.1. Here, each driver travels along the highway with an individual objective that encodes their preferences for speed, acceleration, and proximity to other cars. In heavy traffic, the objectives of drivers may conflict. For instance, if car 1 (blue) wishes to maintain its speed, it must overtake the slower vehicles in front. At the same time, however, the faster car 2 (orange) may wish to maintain its speed but would be forced to decelerate if the driver of car 1 changes lanes.

Mathematically, such interactions of multiple agents with individual, potentially conflicting objectives are characterized by a *noncooperative dynamic game*. The theory underpinning dynamic games is well established [44, 45] and recent work has put forth efficient algorithms to determine equilibrium solutions to these problems, given players' objectives [46, 47]. The *forward* game problem is depicted in Figure 2.1 (left to right) for the highway driving scenario: given the cost functions of all players (left), a forward game solver computes their rational strategies and corresponding future trajectories (right).

Unfortunately, the objectives of agents in a scene are often not known a priori. Therefore, in order for game-theoretic methods to find practical application in fields such as robotics, it is imperative to recover these objectives from data. This *inverse* dynamic game problem is illustrated in Figure 2.1 (right to left) for the highway driving scenario: given observations of players' strategies (right), an inverse game solver recovers objectives (left) which explain the observed behavior. This inverse dynamic game problem is the focus of this work.

The challenge of recovering objectives from observed behavior has been extensively studied in the literature on IOC [20, 21, 48] and inverse reinforcement learning (IRL) [49, 50]. Unfortunately, however, these methods are fundamentally limited to the single-player setting. While recent efforts extend these ideas to multi-agent IRL [51, 52], those approaches are limited to games with *potential* cost structures [53] and do not directly apply in general noncooperative settings. While initial work extends IOC methods to address this limitation [54–56], these inverse dynamic game solvers rely upon full observation of states and inputs of all players.

The main contribution of this work is a novel method for learning players' objectives in noncooperative dynamic games from only noise-corrupted, partial state observations. In addition to learning a cost model for all players, our method also recovers a forward game solution consistent with the learned objectives by enforcing equilibrium constraints on latent trajectory estimates. This bilevel formulation further allows us to couple observed and predicted behavior to recover player's objectives even from temporally-incomplete interactions. As a result, our approach is amenable to online learning and prediction in a receding-horizon fashion.

This work builds upon and extends our earlier work [43]. In this work, we provide a

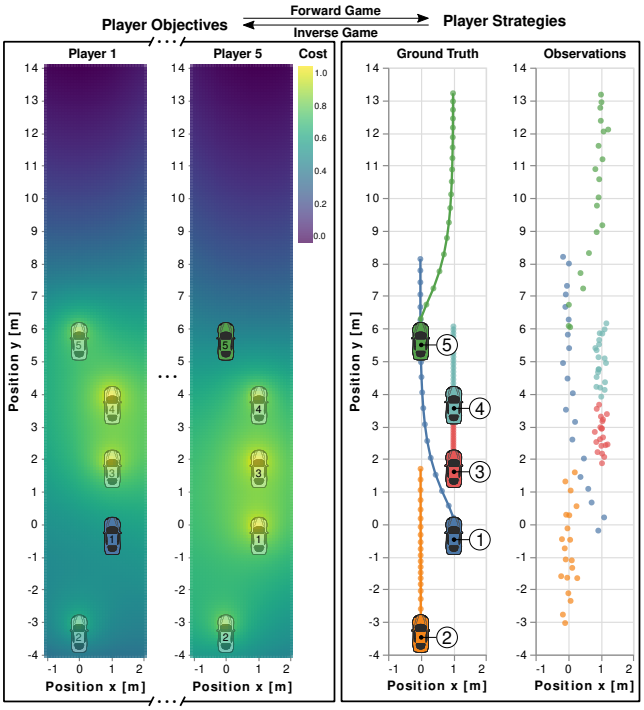


Figure 2.1: 5-player highway driving scenario, modeled as a dynamic game. Solving the "forward" problem amounts to finding optimal trajectories (right) for all cars, given their objectives (left). In contrast, this work addresses the "inverse," i.e., it estimates the objectives of each player given noise-corrupted observations of each agent's trajectories. For example, our method can infer properties such as the degree to which each player wishes to keep a safe distance from others (heatmap, left). These learned objectives constitute an abstract model which can be used to predict players' actions in the future.

more in-depth analysis of that approach. Additionally, while our original work was limited to offline operation and could therefore only recover players' objectives for interactions which had already occurred, in this work we remove this requirement.

We evaluate our method in extensive Monte Carlo simulations in several traffic scenarios with varying numbers of players and interaction geometries. Empirical results show that our approach is more robust to partial state observations, measurement noise, and unobserved time-steps than existing methods, and consequently is more suitable for predicting agents' actions in the future.

2.2 Prior Work

We begin by discussing recent advances in the well-studied area of IOC. While methods from that field address only single-player, cooperative settings, this body of work exposes many of the important mathematical and algorithmic concepts that appear in games. We discuss how some of these approaches have been applied in the noncooperative multi-player setting and emphasize the connections between existing approaches and our contributions.

2.2.1 Single-Player Inverse Optimal Control

The IOC problem has been extensively studied since the well-known work of Kalman [20]. In the context of IRL, early formulations such as that of Ng and Russell [49] and maximum-entropy variants [50, 57] have proven successful in treating problems with discrete state and control sets. In robotic applications, optimal control problems typically involve decision variables in a continuous domain. Hence, recent work in IOC differs from the IRL literature mentioned above as it is explicitly designed for smooth problems.

One common framework for addressing IOC problems with nonlinear dynamics and nonquadratic cost structures is bilevel optimization [21, 48]. Here, the *outer* problem is a least squares or maximum likelihood estimation (MLE) problem in which demonstrations are matched with a nominal trajectory estimate and decision variables parameterize the objective of the underlying optimal control problem. The *inner* problem determines the nominal trajectory estimate as the optimizer of the "forward" (i.e., standard) optimal control problem for the outer problem's decision variables. A key benefit of bilevel IOC formulations is that they naturally adapt to settings with noise-corrupted partial state observations [21].

Early bilevel formulations for IOC utilize derivative-free optimization schemes to estimate the unknown objective parameters in order to avoid explicit differentiation of the solution to the inner optimal control problem [48]. That is, the inner solver is treated as a black-box mapping from cost parameters to optimal trajectories which is utilized by the outer solver to identify the unknown parameters using a suitable derivative-free method. While black-box approaches can be simple to implement due to their modularity and lack of reliance on derivative information, they often suffer from a high sampling complexity [58]. Since each sample in the context of black-box IOC methods amounts to solving a full optimal control problem, such approaches remain intractable for scenarios with large state spaces or additional unknown parameters, such as unknown initial conditions.

Other works instead embed the Karush–Kuhn–Tucker (KKT) conditions of the inner problem as constraints on the outer problem. Since these techniques enforce only first-order necessary conditions of optimality, globally optimal observations are unnecessary and locally optimal demonstrations suffice. Yet, a key computational difficulty of KKT-constrained IOC formulations is that they yield a nonconvex optimization problem due to decision variables in the outer problem appearing nonlinearly with inner problem variables in KKT constraints. This occurs even in the relatively benign case of linear-quadratic IOC.

In contrast to bilevel optimization formulations where necessary conditions of optimality are embedded as constraints, recent methods [22, 27, 59–61] minimize the residual of these conditions directly at the demonstrations. Since the observed demonstration is assumed to satisfy any constraints of the underlying forward optimal control problem, this method can be formulated as fully unconstrained optimization. Additionally, these residual formulations yield a *convex* optimization problem if the class of objective functions is convex in the unknown parameters at the demonstration [22, 28]. This condition holds in the common setting of linearly-parameterized objective functions. Levine and Koltun [59] propose a variant of this approach that utilizes quadratic approximations of the reward model around demonstrations to derive optimality residuals in a maximum entropy framework. Englert and Toussaint [22] present an extension of this method to accommodate inequality constraints on states and inputs. Much like KKT-constrained formulations, these residual methods operate on locally optimal demonstrations. However, an important limitation of residual methods is that they require observations of full state and input sequences. More recently, Menner and Zeilinger [27] compared IOC techniques based on KKT constraints and residuals and demonstrated inferior performance of the latter even in problems with linear dynamics and quadratic target objectives.

Our work takes inspiration from the KKT-constraint formulation for single-player IOC as discussed by Albrecht et al. [21] and Menner and Zeilinger [27]. While these works apply only to single-player settings, we utilize the necessary conditions for open-loop Nash equilibria (OLNEs) [45] to generalize this approach to noncooperative multi-player scenarios.

2.2.2 Multi-Player Inverse Dynamic Games

Many of the IOC techniques discussed above have close analogues in the context of multi-player inverse dynamic games.

As in single-player IOC, methods akin to black-box bilinear optimization have also been studied in the context of inverse games [62, 63]. Peters [62] uses a particle-filtering technique for online estimation of human behavior parameters. This work demonstrates the importance of inferring human behavior parameters for accurate prediction in interactive scenarios. However, there, inference is limited to a single parameter and the work highlights the challenges associated with scaling this sampling based approach to high-dimensional latent parameter spaces. Le Cleac'h et al. [63] employ a similar derivative-free filtering technique based on an unscented Kalman filter. While this approach drastically reduces the overall sample complexity, it still relies on exact observations of the state to reduce the required number of solutions to full dynamic games at the inner level.

Another line of research has put forth solution techniques for inverse games that follow from the residual methods outlined in Section 2.2.1 [54–56, 64]. Köpf et al. [64] study a special case of an inverse linear-quadratic game in which the equilibrium feedback strategies of all but one player are known. This assumption reduces the estimation problem to single-player IOC to which the residual methods discussed above can be applied directly. Rothfuß et al. [54] present a more general approach that does not exploit such special structure but instead minimizes the residual of the first-order necessary conditions for a local OLNE. Inga et al. [55] present a variant of this OLNE residual method in a maximum entropy framework, generalizing the single-player IOC algorithm proposed by Levine and Koltun [59]. Recently, Awasthi and Lamperski [56] also extended the OLNE residual method of Rothfuß et al. [54] to inverse games with state and input constraints. This approach extends that of Englert and Toussaint [22] to noncooperative multi-player scenarios.

All of these inverse game KKT residual methods share many properties with their single-player counterparts. In particular, since they rely upon only local equilibrium criteria, they are able to recover player objectives even from local—rather than only global—equilibrium demonstrations. However, as in the single-player case, they rely upon observation of both state and input to evaluate the residuals.

In contrast to KKT residual methods [54–56], we enforce these conditions as constraints on a jointly estimated trajectory, rather than minimizing the residual of these conditions directly at the observation. By maintaining a trajectory estimate in this manner, our method explicitly accounts for observation noise, partial state observability, and unobserved control inputs. Furthermore, in contrast to black-box approaches to the inverse dynamic game problem [62, 63], our method does not require repeated solutions of the underlying forward game. Moreover, our method returns a full forward game solution in addition to the estimated objective parameters for all players.

2.3 Background: Open-Loop Nash Games

While this work is concerned with the *inverse* game problem of learning objectives from observed behavior, we first provide a technical introduction to the theory of *forward* open-loop dynamic Nash games. These forward games correspond to the model that we seek to recover in this work. Furthermore, as we shall discuss in Section 2.4, they may be used at the inner level of a bilevel optimization problem to formulate the inverse game problem.

As discussed in Section 2.1, dynamic games provide an expressive mathematical formalism for modeling the strategic interactions of multiple agents with differing objectives. Interested readers are directed to [45] for a more complete discussion. We note that dynamic games afford a wide variety of equilibrium concepts; our choice of open-loop Nash Equilibria in this work captures scenarios in which players do not account for future information gains and instead commit to a sequence of control decisions *a priori*. These conditions may occur when occlusions *prevent* future information gains or when bounded rationality causes players to *ignore* them. OLNEs have been demonstrated to capture dynamic interaction when embedded in receding-horizon re-planning schemes [65, 66]. Beyond that, restricting our attention to OLNEs engenders computational advantages which are discussed below. Other choices of solution concept are possible and should be explored in

future work. Recent methods such as those of Di and Lamperski [47] and Le Cleac'h et al. [66] facilitate efficient solutions to the "forward" open-loop games *given players' objectives a priori*.

2.3.1 Preliminaries

Consider a game played between N players over discrete time-steps $t \in [T] := \{1, \dots, T\}$. The game is comprised of three key components: dynamics, objectives (which are later presumed to be unknown in this work), and information structure.

We presume that the game is Markov with respect to state $x \in \mathbb{R}^n$. That is, given each player's control input $u^i \in \mathbb{R}^{m^i}$, $i \in [N]$, the state evolves according to the difference equation

$$x_{t+1} = f_t(x_t, u_t^1, \dots, u_t^N). \tag{2.1}$$

For clarity, we shall introduce the following shorthand notation:

$$\begin{aligned} \mathbf{x} &= (x_1, \dots, x_T), \\ \mathbf{u}^i &= (u_1^i, \dots, u_T^i), \\ \mathbf{u}_t &= (u_t^1, \dots, u_t^N), \\ \mathbf{u} &= (\mathbf{u}^1, \dots, \mathbf{u}^N). \end{aligned}$$

Observe that the state x pertains to the entire game, not only to a single player. In the examples presented in this work, x is simply the concatenation of individual players' states, and correspondingly the dynamics are independent for all players. However, this is not always the case and the methods developed here apply in the more general settings as well.

The objective of player i is encoded by their distinct cost function J^i, which they seek to minimize. This cost can in general depend upon the sequence of states and inputs for all players.[1] In this work, we presume that objectives are expressed in time-additive form, as is common across the optimal control and reinforcement learning literature:

$$J^i(\mathbf{x}, \mathbf{u}) := \sum_{t=1}^{T} g_t^i(x_t, u_t^1, \dots, u_t^N). \tag{2.2}$$

Since the state trajectory $\mathbf{x}$ follows (2.1), these cost functions can also be written in terms of the initial condition x_1 and the sequence of control inputs for all players $\mathbf{u}$. For this reason, we shall also use the notation $J^i(\mathbf{u}; x_1)$, and refer to the tuple of initial state, dynamics, and objectives as

$$\Gamma := \left(x_1, \{f_t\}_{t\in[T]}, \{J^i\}_{i\in[N]}\right). \tag{2.3}$$

Finally, the information structure of a dynamic game refers to the information available to each player when they are required to make a decision at each time. At time t,

[1] State and input constraints are also possible, although they complicate the notion of equilibrium solution. Solution methods such as those of Dirkse and Ferris [37] and Laine et al. [67] address constrained forward games. The present work readily extends to the constrained case; however, we neglect them for clarity of presentation.

then, Player^i's input is a function $\gamma_t^i : \mathcal{I}_t^i \to \mathbb{R}^{m^i}$, where $\mathcal{I}_t^i$ is the set of information available to Player^i at time t. In this work, we consider *open-loop* information structures, i.e., where $\mathcal{I}_t^i = \{x_1\}$.[2] In open-loop information, then, it suffices for Player^i to specify their input sequence $\mathbf{u}^i$ given a fixed initial condition x_1. For this reason, we neglect a more detailed treatment of strategy spaces and information structure, and simply refer to the finite-dimensional sequence of control inputs for each player.

This characterization of a dynamic game is intentionally general. Our solution methods will rely upon established numerical methods for smooth optimization, however, and as such we require the following assumption.

Assumption 2.1 *(Smoothness) Dynamics f and objectives J^i have well-defined second derivatives in all state and control variables, at all times and for all players.*

Most physical systems of interest and interactions thereof are naturally modeled in this way. However, we note that, for example, hybrid dynamics such as those induced by contact do not satisfy this assumption.

We shall illustrate key concepts using a consistent "running example" throughout this chapter.

Running example: *Consider an $N = 2$-player linear-quadratic (LQ) game—i.e., one in which dynamics f_t are linear in state x_t and control inputs $\mathbf{u}_t$, and costs J^i are quadratic in states and controls. Let each player independently follow the dynamics of a double integrator in the Cartesian plane. State $x = (p_x^1, p_y^1, \dot{p}_x^1, \dot{p}_y^1, p_x^2, p_y^2, \dot{p}_x^2, \dot{p}_y^2)$ then evolves with inputs $u^i = (\ddot{p}_x^i, \ddot{p}_y^i)$ according to*

$$x_{t+1} = \overbrace{\begin{bmatrix} \tilde{A} & 0 \\ 0 & \tilde{A} \end{bmatrix}}^{A} x_t + \overbrace{\begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix}}^{B^1} u_t^1 + \overbrace{\begin{bmatrix} 0 \\ \tilde{B} \end{bmatrix}}^{B^2} u_t^2, \tag{2.4}$$

$$\text{where } \tilde{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tilde{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix},$$

and Δt is a uniform time discretization, e.g., 0.1s. Each player has a quadratic objective of the form

$$J^i = \frac{1}{2} \sum_{t=1}^{T} \left(\theta_Q^i x_t Q_t^i x_t + \sum_{j=1}^{N} \theta_R^{ij} u_t^{j\top} R_t^{ij} u_t^j \right). \tag{2.5}$$

In this simple example, Q_t^i and R_t^{ij} are known, positive definite matrices encoding the preferences of each player. The scalars $\theta_Q^i \in \mathbb{R}$ and $\theta_R^{ij} \in \mathbb{R}$ weight *these known matrices. In this work, we develop a technique to learn* a priori *unknown parameters such as the costs weights above from both offline and online data. Note that this simple LQ game shall only serve to explain the general concepts of our method. For our experiments presented in Section 2.7, we*

[2] Recent work in solving forward games also considers *feedback* information in which $\mathcal{I}_t^i = \{x_t\}$; see Fridovich-Keil et al. [46] and Laine et al. [67].

consider more complex problems with nonlinear dynamics and nonquadratic costs, such as the 5-player highway navigation problem shown in Figure 2.1.

2.3.2 The Nash Solution Concept

Combining these components, each player i in an open-loop dynamic game seeks to solve the following optimization problem

$$\forall i \in [N] \begin{cases} \min\limits_{\mathbf{x},\mathbf{u}^i} \; J^i(\mathbf{u}; x_1) & \text{(2.6a)} \\ \text{s.t.} \quad x_{t+1} = f_t(x_t, \mathbf{u}_t), \forall t \in [T-1]. & \text{(2.6b)} \end{cases}$$

There exist a variety of distinct solution concepts for such smooth open-loop dynamic games. In this work, we consider the well-known Nash equilibrium concept, wherein no player has a unilateral incentive to change its strategy. Mathematically, the Nash concept is defined as follows.

Definition 2.1 *(Open-loop Nash equilibrium) The strategies* $\mathbf{u}^* := (\mathbf{u}^{1*}, \dots, \mathbf{u}^{N*})$ *constitute an open-loop Nash equilibrium (OLNE) in the game* $\Gamma = \big(x_1, \{f_t\}_{t\in[T]}, \{J^i\}_{i\in[N]}\big)$ *if the following inequalities hold:*

$$J^{i*} = J^i(\mathbf{u}^*; x_1) \leq J^i\big((\mathbf{u}^i, \mathbf{u}^{-i*}); x_1\big), \forall i \in [N]. \tag{2.7}$$

Here, we use the shorthand $(\mathbf{u}^i, \mathbf{u}^{-i*})$ *to indicate the collection of strategies in which* only *Player*^{i} *deviates from the Nash profile, i.e.,* $\mathbf{u}^i \neq \mathbf{u}^{i*}$.

Note that, at a Nash equilibrium, each player must *independently* have no incentive to deviate from its strategy. Since players' objectives may generally conflict, the Nash concept encodes noncooperative, rational, and potentially selfish behavior.

Unfortunately, Nash equilibria are known to be very difficult to find in general [68]. In this work, we seek only *local* equilibria which satisfy the Nash conditions (2.7) to first order. That is, following similar approaches in both single-player IOC [21, 22] and forward/inverse open-loop games [60, 66], we encode forward optimality via the corresponding first-order necessary conditions. These first-order necessary conditions are given by the union of the individual players' KKT conditions, i.e.,

$$\mathbf{0} = \mathbf{G}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) := \begin{bmatrix} \left.\begin{matrix} \nabla_{\mathbf{x}} J^i + \nabla_{\mathbf{x}} \mathbf{F}(\mathbf{x}, \mathbf{u})^\top \boldsymbol{\lambda}^i \\ \nabla_{\mathbf{u}^i} J^i + \nabla_{\mathbf{u}^i} \mathbf{F}(\mathbf{x}, \mathbf{u})^\top \boldsymbol{\lambda}^i \end{matrix}\right\} \forall i \in [N] \\ \mathbf{F}(\mathbf{x}, \mathbf{u}) \end{bmatrix}. \tag{2.8}$$

Here, the first two block-rows are repeated for all players, and the function $\mathbf{F}(\mathbf{x}, \mathbf{u})$ accumulates the dynamic constraints of (2.6b) at all time steps, with the t^{th} row given by $x_{t+1} - f_t(x_t, u_t^1, \dots, u_t^N)$. Note that we have also introduced costate variables $\boldsymbol{\lambda}^i := (\lambda_1^i, \dots, \lambda_{T-1}^i)$ for each player, with $\lambda_t^i \in \mathbb{R}^n$ the Lagrange multiplier corresponding to Player^i's dynamics constraint in (2.6b) at time step t. Note that, as with control inputs, we use the notation $\boldsymbol{\lambda} := (\boldsymbol{\lambda}^1, \dots, \boldsymbol{\lambda}^N)$.

Running example: *Consider the two-player LQ example above with double integrator dynamics given by (2.4) and quadratic objectives given by (2.5). The t^{th} block of the first row of (2.8) is given by*

$$\mathbf{0} = \theta_Q^i Q_t^i x_t + \lambda_{t-1}^i - A^\top \lambda_t^i \tag{2.9}$$

for Player^i. Likewise, the t^{th} block of the second row of (2.8) for Player^i is given by

$$\mathbf{0} = \theta_R^{ii} R_t^{ii} u_t^i - B^{i\top} \lambda_t^i. \tag{2.10}$$

Finally, the t^{th} block of the final row of (2.8) is given by

$$\mathbf{0} = x_{t+1} - Ax_t - B^1 u_t^1 - B^2 u_t^2. \tag{2.11}$$

Computationally, the KKT conditions of the forward game, given in (2.8), are a set of, generally nonlinear, equality constraints in the variables $\mathbf{x}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$. To find a solution—that is, a root of $\mathbf{G}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$—we may employ a root-finding algorithm such as a variant of Newton's method [58, Chapter 11]. This is the approach taken by, e.g., Le Cleac'h et al. [66].

Running example: *For our LQ example, it can be seen that a single step of Newton's method on $\mathbf{G}(\cdot)$ amounts to the well-known Riccati solution to an* open-loop *LQ game [45, Chapter 6].*[3]

2.4 Problem Setup

Solving a *forward* Nash game amounts to identifying optimal strategies for all players, provided a priori knowledge of their objectives J^i. By contrast, in this work we are concerned with the *inverse* Nash problem, i.e., that of identifying players' objectives which explain their observed behavior. To develop the inverse Nash problem, here we shall presume that learning occurs offline, given a sequence of noisy, partial observations of all players' state. The method we develop for this setting, however, is amenable to trajectory prediction and online, receding-horizon operation as discussed in Section 2.5.2.

We formulate the inverse Nash problem as one of offline learning, in which players' objectives belong to a known parametric function class. To that end, we make the following assumption.

Assumption 2.2 *(Parametric objectives) Player^i's cost function is fully described by a vector of parameters $\theta^i \in \mathbb{R}^{k^i}$. That is, $J^i(\cdot; \theta^i) \equiv \sum_{t=1}^T g_t^i(x_t, u_t^1, \ldots, u_t^N; \theta^i)$.*

Recalling Assumption 2.1, the functions $g_t^i(\cdot; \theta^i)$ have well-defined derivatives in states x_t and controls u^i. We shall also extend this smoothness assumption to include the parameters themselves.

[3] Note that this Newton step *differs* from that given by the Riccati solution to a *feedback* LQ game.

Assumption 2.3 *(Smoothness in parameter space) Extending Assumption 2.1, we require that stage cost functions $g_t^i(\cdot;\theta^i)$ have well-defined first- and second-derivatives with respect to the parameter vector θ^i.*

This smoothness assumption is quite general. For example, players' stage costs $g_t^i(\cdot;\theta^i)$ may be encoded as arbitrary function approximators such as artificial neural networks. In this work, we choose a more interpretable (though less flexible) parametric structure; we defer an investigation of more general cost structures for future work. In particular, the examples considered here use a *linearly-parameterized* structure in which $g_t^i(\cdot;\theta^i)$ is a linear function of θ^i, i.e., $g_t^i(\cdot;\theta^i) \equiv \theta^{i\top}\tilde{g}_t^i(\cdot)$ for some set of potentially nonlinear basis functions $\tilde{g}_t^i(\cdot)$. By incorporating appropriate domain-specific knowledge, however, these relatively simple cost structures are able to encapsulate complex, strategic interactions such as the highway lane changes of Figure 2.1.

Running example: *Recall the quadratic objectives of (2.5), and take cost parameters $\theta^i = (\theta_Q^i, \theta_R^{ij})_{j\in[N]}$. Observe, therefore, that Player^i's objective depends linearly upon its cost parameters θ^i.*

Thus equipped, the objective learning problem reduces to maximizing the likelihood of a sequence of partial state observations $\mathbf{y} := (y_1, \ldots, y_T)$ for the parametric class of games $\Gamma(\theta) = \big(x_1, f, \{J^{(i)}(\,\cdot\,;\theta^{(i)})\}_{i\in[N]}\big)$. Formally, we seek to solve a problem of the form

$$\max_{\theta,\mathbf{x},\mathbf{u}} \quad p(\mathbf{y} \mid \mathbf{x}, \mathbf{u}) \tag{2.12a}$$

$$\text{s.t.} \quad (\mathbf{x}, \mathbf{u}) \text{ is an OLNE of } \Gamma(\theta) \tag{2.12b}$$

$$(\mathbf{x}, \mathbf{u}) \text{ is dynamically feasible under } f, \tag{2.12c}$$

where θ aggregates all players' cost parameters, i.e., $\theta := (\theta^1, \ldots, \theta^N)$, and $p(\mathbf{y} \mid \mathbf{x}, \mathbf{u})$ constitutes a known observation likelihood, or measurement, model.

Remark 2.1 *(Initial state) Observe that x_1 is an explicit decision variable in (2.12), whereas it represents a constant (known) initial condition in the forward game problem discussed in Section 2.3. This reflects the fact that the state trajectory, including initial conditions, must be estimated as part of the inverse problem. As we shall see, estimating the state trajectory jointly with the cost parameters allows our method to be less sensitive to observation noise.*

This measurement model is arbitrary, though, following Assumption 2.1 and Assumption 2.3, it must be smooth. In the simplest instance, we may receive an exact measurement of the sequence of states and inputs for all players. In that case, the measurement model $p(\mathbf{y} \mid \mathbf{x}, \mathbf{u})$ reduces to a Dirac delta function. More generally, $p(\mathbf{y} \mid \mathbf{x}, \mathbf{u})$ may be an arbitrary smooth probability density function, making our formulation amenable to realistic sensors such as cameras or LiDARs.

Prior work in both single-player IOC, such as that of Englert and Toussaint [22], and inverse games, such as those of Awasthi and Lamperski [56] and Rothfuß et al. [54], presumes a degenerate measurement model in which states and controls are observed directly without any noise. When perfect observations are unavailable, these methods naturally extend by first estimating a sequence of likely states and controls (a standard nonlinear filtering

problem). In Section 2.6, we describe these sequential estimation methods in greater detail. In contrast, our formulation given in (2.12) encodes a coupled estimation problem in which states, control inputs, and cost parameters must all be estimated simultaneously. Thus, our method exploits the additional coupling imposed by the Nash equilibrium constraints onto the unknowns. In Section 2.7, we conduct a series of Monte Carlo experiments to quantify the advantages afforded by simultaneous learning over sequential estimation.

2.5 Equilibrium-Constrained Cost Learning

Here we present our core contribution, a mathematical formulation of objective inference in multi-agent, noncooperative games. We express this problem as a nonconvex optimization problem with equilibrium constraints, which we relax into a standard-format equality-constrained nonlinear program.

2.5.1 Offline Learning

We first consider the problem of learning each player's objective from previously recorded data of prior interactions, *offline*.

(2.12) is a mathematical program with equilibrium constraints [69, 70], with the nested equilibrium conditions of (2.12b) encoding the Nash inequalities of Definition 2.1. Equilibrium constraints generalize bilevel programming, and computational approaches tend to be less mature than those for standard-form (in)equality-constrained programming.

We relax the equilibrium constraint of (2.12b) by replacing it with its KKT conditions, i.e., by substituting (2.8). This yields:

$$\max_{\theta,\mathbf{x},\mathbf{u},\boldsymbol{\lambda}} \quad p(\mathbf{y} \mid \mathbf{x}, \mathbf{u}) \tag{2.13a}$$

$$\text{s.t.} \quad \mathbf{G}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}; \theta) = \mathbf{0}. \tag{2.13b}$$

Here, we have explicitly written the KKT conditions from (2.8) in terms of the cost parameters θ. Additionally, observe that in (2.13), the costates $\boldsymbol{\lambda}$ required to evaluate the KKT conditions $\mathbf{G}(\cdot;\theta)$ appear as *additional primal variables*. The constraints of (2.13b) will be assigned their own Lagrange multipliers, which are distinct from the original costates. By letting states, control inputs, and costates be primal variables, the KKT conditions $\mathbf{G}(\cdot)$ do not depend explicitly upon the observations $\mathbf{y}$. Thus, solving (2.13) does not require complete state or input observations; rather, the equilibrium constraints of (2.13b) allow us to reconstruct this missing information while we estimate cost parameters θ, simultaneously. Several remarks are in order.

Remark 2.2 *(Multiple observed trajectories) We have developed (2.13) for the setting in which a single trajectory* $(\mathbf{x}, \mathbf{u})$ *has been observed, yielding a measurement sequence* $\mathbf{y}$*. However, our approach affords straightforward extension to settings in which player's objectives are learned from multiple demonstrations. In this instance, the primal variables* $(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$ *would be replicated for all trajectories, although the cost parameters* θ *would be shared. The objective given by (2.13a) would be replaced by the joint probability of all measurements conditioned on all underlying trajectories, and the equilibrium constraints in (2.13b) would be concatenated for all trajectories.*

Remark 2.3 *(Regularizing parameters) Depending upon the parametric structure of players' objectives $J^i(\cdot;\theta^i)$, and hence the structure of KKT residual $\mathbf{G}(\cdot;\theta)$, it can be critical to regularize and/or constrain cost parameters. For example, if there exists a choice of θ^i for Player^i such that $J^i(\mathbf{x},\mathbf{u};\theta^i)$ is constant for all dynamically-feasible trajectories $(\mathbf{x},\mathbf{u})$, then every such trajectory would satisfy the equilibrium constraint of (2.12b). Such choices of θ must be avoided, e.g., by regularizing or otherwise constraining parameters.*

Running example: *Following Remark 2.3, we constrain the parameters $\theta^i \geq c > 0$. Moreover, to account for scale invariance, we constrain their sum to unity, i.e.,*

$$\sum_{i\in[N]} \left(\theta^i_Q + \sum_{j\in[N]} \theta^{ij}_R \right) = 1.$$

Least Squares

A common observation model $p(\mathbf{y} \mid \mathbf{x},\mathbf{u})$ is the additive white Gaussian noise (AWGN) model. Here, each observation y_t depends only upon the current state x_t and control inputs $\mathbf{u}_t$, i.e.,

$$y_t = h_t(x_t,\mathbf{u}_t) + n_t, \tag{2.14}$$

where the (potentially nonlinear) function h_t computes the expected measurement, and n_t is a zero-mean Gaussian white noise process with known covariance, i.e., $n_t \sim \mathcal{N}(0,\Sigma_t)$. In this case, following standard methods in maximum likelihood estimation [71], it is straightforward to express the maximization in (2.13a) as nonlinear least squares by taking the negative logarithm of $p(\mathbf{y} \mid \mathbf{x},\mathbf{u})$:

$$\min_{\theta,\mathbf{x},\mathbf{u},\boldsymbol{\lambda}} \quad \sum_{t=1}^{T} \big(y_t - h_t(x_t)\big)^\top \Sigma_t^{-1} \big(y_t - h_t(x_t)\big) \tag{2.15a}$$

$$\text{s.t.} \quad \mathbf{G}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda};\theta) = \mathbf{0}. \tag{2.15b}$$

In summary, this inverse problem entails the following task: Find those parameters θ for which the corresponding game solution generates expected observations near the observed data. This formulation of the inverse game problem can be encoded using well-established numerical modeling languages such as CasADi [72] or JuMP [73], and solved using off-the-shelf optimization routines such as IPOPT [40] or SNOPT [74].

Problem Complexity

Let us examine the structure of the least squares problem in (2.15) more carefully. In general, the observation map $h_t(\cdot)$ and KKT conditions $\mathbf{G}(\cdot;\cdot)$ may be arbitrarily nonlinear. Therefore, without further structural assumptions, our formulation is an equality-constrained nonlinear least squares problem. Due primarily to the nonlinearities in $\mathbf{G}$,

(2.15) is generally nonconvex. Solution methods, therefore, may be sensitive to initial values of primal variables; we discuss a straightforward initialization scheme in Section 2.6.1.

Perhaps surprisingly, this nonconvexity persists in the LQ setting of our running example, even when $h_t(\cdot)$ is the identity.

Running example: *Consider the LQ setting, with $\theta^i = (\theta^i_Q, \theta^{ij}_R)_{j\in[N]}$ as before. Let the observation map be the identity, i.e., $h_t(x_t) = x_t$ and presume AWGN. The resulting nonlinear least squares problem in (2.15) has constraints of the form given in Equations* (2.9) *to* (2.11)*. Let us consider the first of these constraints for a single time step t and Player i:*

$$\mathbf{0} = \theta^i_Q Q^i_t x_t + \lambda^i_{t-1} - A^\top \lambda^i_t.$$

Recall that the decision variables in our formulation are $(\theta, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$. Here, we see that θ^i multiplies x_t. At best, therefore, this constraint is a bilinear *equality, making the overall problem in (2.15) nonconvex even for this minimal inverse LQ game.*

When we directly observe both state and control inputs without noise, i.e., $y_t \equiv (x_t, \mathbf{u}_t)$, these constraints become *linear* even in the general non-LQ setting, so long as players' objectives are linearly parameterized. In this setting, we may rewrite (2.9) as

$$\begin{aligned}
\mathbf{0} &= \nabla_{x_t} \overbrace{g^i_t(x_t, u_t; \theta^i)}^{\theta^{i\top}\tilde{g}^i_t(\cdot)} + \lambda^i_{t-1} - \nabla_{x_t} f_t(x_t, u_t)^\top \lambda^i_t && \text{(2.16a)}\\
&= \theta^{i\top} \nabla_{x_t} \tilde{g}^i_t(x_t, u_t) + \lambda^i_{t-1} - \nabla_{x_t} f_t(x_t, u_t)^\top \lambda^i_t. && \text{(2.16b)}
\end{aligned}$$

With this observation model, then, the only decision variables are $(\theta^i, \lambda^i_t, \lambda^i_{t-1})$, which all appear linearly. Furthermore, the least squares objective in (2.15a) becomes unnecessary, since, by assumption, the measurements $\mathbf{y}$ already include the states $\mathbf{x}$ exactly. Incorporating these simplifications, the entire constrained least squares problem of (2.15) reduces to the problem

$$\begin{aligned}
\text{find} \quad & \theta, \boldsymbol{\lambda} && \text{(2.17a)}\\
\text{s.t.} \quad & \mathbf{0} = \theta^{i\top} \nabla_{x_t} \tilde{g}^i_t(x_t, u_t) + \lambda^i_{t-1} \\
& \qquad - \nabla_{x_t} f_t(x_t, u_t)^\top \lambda^i_t, \forall i, t && \text{(2.17b)}\\
& \mathbf{0} = \theta^{i\top} \nabla_{u^i_t} \tilde{g}^i_t(x_t, u_t) \\
& \qquad - \nabla_{u^i_t} f_t(x_t, u_t)^\top \lambda^i_t, \forall i, t. && \text{(2.17c)}
\end{aligned}$$

Because the constraints in (2.17) are linear, the problem is equivalent to a linear system of equations. Moreover, since the constraints are completely decoupled for each player, they may be solved separately and in parallel for all players to obtain cost parameters θ^i and costates $\boldsymbol{\lambda}^i$. This reduction forms the basis for the state-of-the-art in solving inverse dynamic games [54, 56], which only apply in settings with perfect state and input observations. To compare against these methods in more general settings that feature noise, unobserved inputs, and partial state measurements, we augment these methods with a sequential optimization procedure in Section 2.6. Comparative Monte Carlo studies of all approaches are presented in Section 2.7.

2.5.2 Online Learning

While Section 2.5.1 estimates the objectives of interacting agents from recorded data *offline*, our formulation for inverse Nash problems extends naturally to an *online* learning setting; i.e., cost learning from observations of ongoing interactions. As we shall discuss below, our method can perform online cost learning and trajectory prediction simultaneously, making it suitable for receding-horizon applications.

Learning with Prediction

Equipped with a tractable solution strategy for the setting of offline learning, we now consider a coupled *prediction* and learning problem. Similar problems have been considered in the single-agent setting by, e.g., [61, 75]. Here, we aim to learn the cost parameters θ from only a *subset* of the game horizon; i.e., we presume that observations $\mathbf{y} = (y_1, \dots, y_{\tilde{T}})$ where the observation horizon $\tilde{T} \leq T$. Despite this change, the original problem of (2.12) remains effectively unchanged; only the objective has changed. In particular, by substituting the KKT conditions for an OLNE in place of the original equilibrium constraint as in (2.13), and making AWGN assumptions, we recover a variant of the constrained least squares formulation of (2.15):

$$\min_{\theta, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}} \quad \sum_{t=1}^{\tilde{T}} \big(y_t - h_t(x_t)\big)^\top \Sigma_t^{-1} \big(y_t - h_t(x_t)\big) \tag{2.18a}$$

$$\text{s.t.} \quad \mathbf{G}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}; \theta) = \mathbf{0}. \tag{2.18b}$$

Note that the upper limit of addition is $\tilde{T}$, rather than T as in (2.15a), while the OLNE KKT conditions in (2.18b) depend upon states, inputs, and costates for all times $t \in \{1, \dots, \tilde{T}, \dots, T\}$.

Despite the similarities between this problem and (2.15), the Nash trajectory $(\mathbf{x}^*, \mathbf{u}^*)$, which emerges as a solution affords a new interpretation. In particular, for times $t \leq \tilde{T}$ these equilibrium states and controls constitute filtered estimates of the observed quantities $\mathbf{y}$, while for times $t > \tilde{T}$ they represent *predictions* of the future. Importantly, however, extending trajectories beyond the observation horizon $\tilde{T}$ adds additional constraints to (2.15). This ability to incorporate future, unobserved states makes the method more robust and data efficient when only a fraction of the game horizon is observed. Consequently, this formulation can be employed for online learning in scenarios of ongoing interactions. We provide a detailed empirical analysis of this setting in Section 2.7.2. A summary of this variant of our inverse game solver is provided in Figure 2.2a.

receding-horizon Learning

Our method is directly amenable to receding-horizon, online operation. Here, we suppose that the agents interact over the half-open time-interval $t \in \{1, \dots, \tilde{T}, \dots, \infty\}$, and that observations exist for $t \leq \tilde{T}$. Here, $\tilde{T}$ may be interpreted as the current time and, as time elapses, both $\tilde{T}$ and the overall prediction horizon T increase accordingly. Unfortunately, however, increasing the overall problem horizon increases the number of variables in (2.12), eventually making the problem intractable.

To simplify matters, we approximate the learning problem at each instant by neglecting all times outside the interval $\{\tilde{T} - s_\mathrm{o}, \ldots, \tilde{T}, \ldots, \tilde{T} + s_\mathrm{p}\}$, where s_o is the length of a fixed-lag buffer of past observations, and s_p is the horizon of future state predictions. In this setting, the total number of variables remains constant (since the length of this interval is constant), rendering (2.12) tractable to solve online. More precisely, at time $\tilde{T}$ (and under AWGN assumptions), we solve a modified version of (2.18)

$$\min_{\theta, \mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}} \quad \sum_{t=\tilde{T}-s_\mathrm{o}}^{\tilde{T}} \big(y_t - h_t(x_t)\big)^\top \Sigma_t^{-1} \big(y_t - h_t(x_t)\big) \tag{2.19a}$$

$$\text{s.t.} \quad \mathbf{G}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}; \theta) = \mathbf{0}, \tag{2.19b}$$

where the KKT constraint $\mathbf{G}(\cdot)$ is understood to depend upon times $t \in \{\tilde{T} - s_\mathrm{o}, \ldots, \tilde{T}, \ldots, \tilde{T} + s_\mathrm{p}\}$ and states, control inputs, and costates are also limited to that interval. At each later time, we solve a problem with identical structure, with the understanding that $\tilde{T}$ will have changed to reflect the elapsed time. In effect, this procedure amounts to simultaneous fixed-lag smoothing and receding-horizon prediction. We simulate this online learning procedure in Section 2.7.3.

2.6 Baseline

Recall the discussion of Section 2.5.1, in which we show that—with noiseless observations of states $\mathbf{x}$ and controls $\mathbf{u}$, and linear cost parameterization $g_t^i(\cdot; \theta^i) \equiv \theta^{i\top} \tilde{g}_t^i(\cdot)$—our formulation reduces to the linear system of equations of (2.17). This reduction underlies state of the art methods for learning the objectives of players in games [54, 56]. Therefore, such methods unfortunately require noiseless observations of the full state and input sequences for all players. In contrast, our approach in (2.13) is amenable to noisy, partial observations.

2.6.1 Recovering Unobserved Variables

To provide a meaningful comparison between our proposed technique and the state-of-the-art in settings with imperfect observations, we augment [54, 56] with a pre-processing to estimate unobserved states and inputs. To that end, we solve the following relaxed version of (2.13):

$$\tilde{\mathbf{x}}, \tilde{\mathbf{u}} := \arg\max_{\mathbf{x}, \mathbf{u}} \quad p(\mathbf{y} \mid \mathbf{x}, \mathbf{u}) \tag{2.20a}$$

$$\text{s.t.} \quad \mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{0}. \tag{2.20b}$$

As in Section 2.5.1, under a AWGN assumption (2.20) becomes equality-constrained nonlinear least squares. However, unlike (2.15), we have neglected the first two rows of the equilibrium constraint given in (2.8). That is, (2.20) computes a maximum likelihood estimate of states and inputs irrespective of the underlying game structure.

The solution of this smoothing problem is used as an estimate of states and inputs when the baseline is employed in partially observed settings. Beyond that, the same procedure

serves as simple, yet effective initialization scheme for our method to tackle issues of non-convexity discussed in Section 2.5.1.

2.6.2 Minimizing KKT Residuals

Like our proposed method, the state-of-the-art methods developed by Rothfuß et al. [54] and Awasthi and Lamperski [56] use the forward game's KKT conditions to measure the quality of a set of cost parameters θ. While we compare to this derivative-based, KKT condition approach, we note that other approaches outlined in Section 2.2.2 such as [63] utilize black-box optimization methods and do not require or exploit derivative information. These significant algorithmic differences—and the resulting differences in sample complexity, locality of solutions, etc.—make a direct comparison difficult to interpret.

Specifically, the KKT residual method of [54, 56] fixes the state and input sequences to their observed—or in our case, estimated via (2.20)—values. Fixing these variables, however, the resulting linearly-constrained satisfiability problem of (2.17) may be infeasible, depending upon the parametric structure of costs $g_t^i(\cdot;\theta^i)$. In lieu, state-of-the-art approaches minimize the KKT residual itself, i.e.,

$$\min_{\theta,\boldsymbol{\lambda}} \|\mathbf{G}(\tilde{\mathbf{x}},\tilde{\mathbf{u}},\boldsymbol{\lambda};\theta)\|_2^2 \,. \tag{2.21}$$

In prior work [54, 56], $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ are assumed to be directly observed. As discussed in Section 2.6.1, here we presume they are the results of the pre-processing step given in (2.20). Additionally, like the linear system of equations in (2.17), the only decision variables here are the objective parameters θ and the costates $\boldsymbol{\lambda}$. In effect, the baseline does *not* refine the state and input estimates given by the pre-processing step of (2.20). Furthermore, as in (2.17), the problem may be decomposed into separate problems for each player and solved in parallel. In essence, then, this KKT residual formulation neglects the coupling between players' actions which is encoded in the equilibrium conditions; computationally, it reduces to solving separate IOC problems for each player neglecting game-theoretic interactions with others.

A schematic overview of this baseline approach is depicted in Figure 2.2b. By first estimating the states $\mathbf{x}$ and inputs $\mathbf{u}$ from measurements $\mathbf{y}$, and only afterward learning the cost parameters θ and associated costates $\boldsymbol{\lambda}$, the KKT residual method can be thought of as a sequential decomposition of our approach. By contrast, our formulation maintains $(\mathbf{x},\mathbf{u})$ as decision variables and refines the initial guess of $(\tilde{\mathbf{x}},\tilde{\mathbf{u}})$ by *identifying all variables simultaneously.*

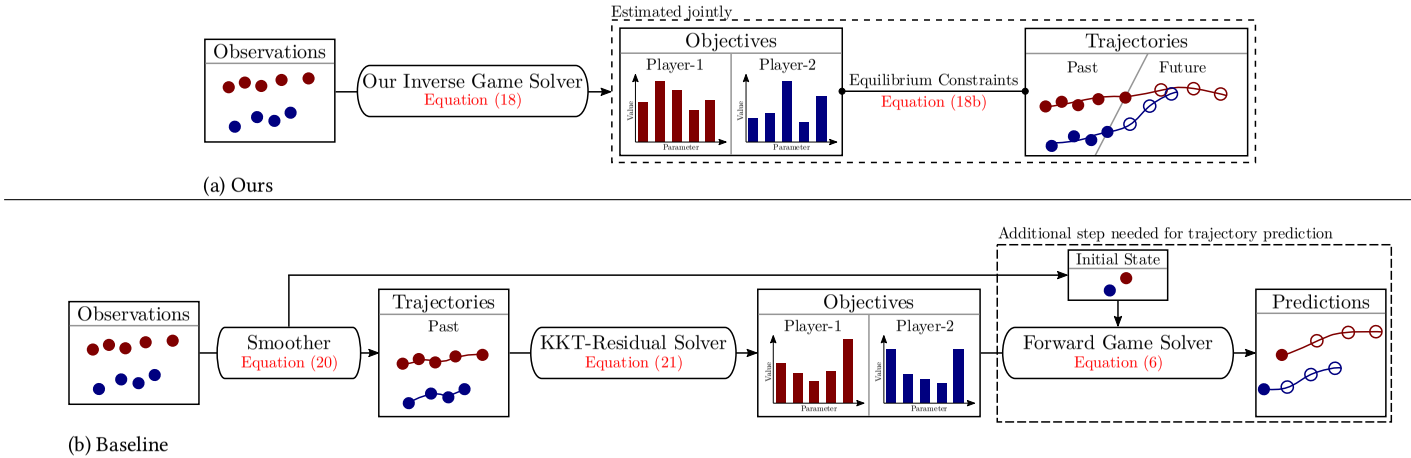


Figure 2.2: Schematic overview of inverse game solvers set up for online operation. (a) Our method computes player's objectives, state estimates, and trajectory predictions jointly. (b) The baseline requires full knowledge of states and inputs and therefore must preprocess raw observations before it can estimate players' objectives. In order to generate trajectory predictions, the baseline must solve an additional forward game formulated over the estimated initial states and objectives.

2.7 Experiments

In this work, we develop a technique for learning players' objectives in continuous dynamic games from noise-corrupted, partial state observations. We conduct a series of Monte Carlo studies to examine the relative performance of our proposed methods and the KKT residual baseline in both offline and online learning settings.

2.7.1 Experimental Setup

We implement our proposed approach as well as the KKT residual baseline of [54] in the Julia programming language [76], using the mathematical modeling framework JuMP [73]. As a consequence, our implementation encodes an abstract description of (2.13), making it straightforward to use in concert with a variety of optimization routines. In this work, we use the open source COIN-OR IPOPT algorithm [40]. The source code for our implementation is publicly available.[4]

To evaluate the relative performance of our proposed approach with the KKT residual baseline, we perform several Monte Carlo studies. The details of these studies are described below. However, all of these studies share the following overall setup: we fix a cost parameterization for each player, find corresponding OLNE trajectories as roots of (2.8) using the well-known iterated best response (IBR) algorithm [65], and simulate noisy observations thereof with additive white Gaussian noise (AWGN) as in (2.14). Each study then presents samples across a different problem parameter to test the sensitivity of both approaches to observation noise (Sections 2.7.2 and 2.7.3) and unobserved time-steps (Section 2.7.2) in two different problem settings.

In each of the studies below, we consider N vehicles navigating traffic, and instantiate game dynamics and player objectives as follows. Each vehicle has its own state x^i such that the global game state is concatenated as $x = (x^1, \ldots, x^N)$. Further, each vehicle follows unicycle dynamics at time discretization Δt:

$$x^i_{t+1} = \begin{cases} \textit{(x-position)} & p^i_{x,t+1} = p^i_{x,t} + \Delta t\, v^i_t \cos \psi^i_t \\ \textit{(y-position)} & p^i_{y,t+1} = p^i_{y,t} + \Delta t\, v^i_t \sin \psi^i_t \\ \textit{(heading)} & \psi^i_{t+1} = \psi^i_t + \Delta t\, \omega^i_t \\ \textit{(speed)} & v^i_{t+1} = v^i_t + \Delta t\, a^i_t, \end{cases} \tag{2.22}$$

where $u^i_t = (\omega_t, a_t)$ includes the yaw rate and longitudinal acceleration. Finally, each player's objective is characterized by a stage cost g^i_t which is a weighted sum of several basis functions, i.e.,

$$g^i_t = \sum_{\ell=1}^{5} w^i_\ell g^i_{\ell,t} \begin{cases} g^i_{1,t} = \mathbb{1}(t \geq T - t_{\text{goal}}) \mathrm{d}(x^i_t, x^i_{\text{goal}}) & \text{(2.23a)} \\ g^i_{2,t} = -\sum_{j \neq i} \log(\|p_i - p_j\|_2^2) & \text{(2.23b)} \\ g^i_{3,t} = (v^i)^2 & \text{(2.23c)} \\ g^i_{4,t} = (\omega^i_t)^2 & \text{(2.23d)} \\ g^i_{5,t} = (a^i_t)^2. & \text{(2.23e)} \end{cases}$$

[4] https://github.com/PRBonn/PartiallyObservedInverseGames.jl

Here, the cost parameters $\theta^i = (w_\ell^i)_{\ell\in[5]}, w_\ell^i \in \mathbb{R}_+$ are positive weights for each cost component. Further, $p_i = (p_x^i, p_y^i)$ denotes the planar position of Player^i, and $\mathrm{d}(\cdot,\cdot)$ is an arbitrary distance mapping. For example, we may choose $\mathrm{d}(x_t^i, x_{\text{goal}}^i) = \|p_t^i - p_{\text{goal}}^i\|_2^2$ to compute squared distance from a fixed goal position. Note, however, that this map is generic and can also be used to encode more complex goal-reaching specifications as in the highway lane-changing example depicted in Figure 2.1. Taken together, the basis functions encode the following aspects of each player's preferences:

1. Be close to the goal state within the last t_{goal} time steps (2.23a)
2. Avoid close proximity to other vehicles (2.23b)
3. Avoid high speeds (2.23c)
4. Avoid large control efforts (2.23d, 2.23e)

Games of this form are inherently noncooperative since players must compete to reach their own goals efficiently while avoiding collision with one another. Hence, they must negotiate these conflicting objectives and thereby find an equilibrium of the underlying game.

In all of the Monte Carlo studies, we evaluate the approaches for two different noisy observation models h_t^{full} and h_t^{partial}. In $h_t^{\text{full}}(x_t) := x_t$, estimators observe the *full* state, and in $h_t^{\text{partial}}(x_t) := (p_t^1, \psi_t^1, \ldots, p_t^N, \psi_t^N)$, estimators observe the position and heading but not the speed of each agent; i.e., they receive a *partial* state observation.

2.7.2 Detailed Analysis of a 2-Player Game

We first study the performance of our method in a simplified, $N = 2$-player game. This set of experiments demonstrates the performance gap of our approach and the KKT residual baseline in methods in a conceptually simple and easily interpretable scenario. Here, the game dynamics are given as in (2.22), and player objectives are parameterized as in (2.23). In particular, we let $\mathrm{d}(x_t^i, x_{\text{goal}}^i) = \|p_t^i - p_{\text{goal}}^i\|_2^2$. In summary, therefore, each vehicle wishes to reach a fixed, known goal position in the plane while avoiding collision with the other.

Offline Learning

We begin by studying both our method's and the baseline's ability to infer the unknown objective parameters θ, as developed in Section 2.5.1. To do so, we conduct a Monte Carlo study for the aforementioned 2-player collision-avoidance application.

We generate 40 random observation sequences at each of 22 different levels of isotropic observation noise. For each of the resulting 880 observation sequences we run both our method and the baseline to recover estimates of weights $\theta^i = (w_\ell^i)_{\ell\in[5]}$ for each player. Note that in this offline setting both methods learn these objective parameters from noisy observations of a single, complete game trajectory. That is, each estimate relies upon 25 s of simulated interaction history from a single scenario.

Figure 2.3 shows the estimator performance for varying levels of observation noise in two different metrics. Figure 2.3a reports the mean cosine error of the objective parameter estimates. That is, we measure cosine-dissimilarity between the unobserved true model parameters θ_{true} and the learned estimates θ_{est} according to

$$D_{\cos}(\theta_{\text{true}}, \theta_{\text{est}}) = 1 - \frac{1}{N} \sum_{i \in [N]} \frac{\theta_{\text{true}}^{i\top} \theta_{\text{est}}^{i}}{\left\|\theta_{\text{true}}^{i}\right\|_2 \left\|\theta_{\text{est}}^{i}\right\|_2}, \tag{2.24}$$

where the mean is taken over the N players. The normalization of the parameter vectors in (2.24) reflects the fact that the absolute scaling of each player's objective parameters does not effect their optimal behavior, holding other players' parameters fixed. In sum, this metric measures the estimator performance in objective *parameter space*.

Figure 2.3b shows the mean absolute position error for trajectory *reconstructions* computed by finding a root of (2.8) using the estimated objective parameters. Reconstruction error allows us to inspect the quality of learned cost parameters for explaining observed vehicle motion, providing a more tangible metric of algorithmic quality. In addition to the raw data, we highlight the median as well as the interquartile range (IQR) of the estimation error over a rolling window of 60 data points.

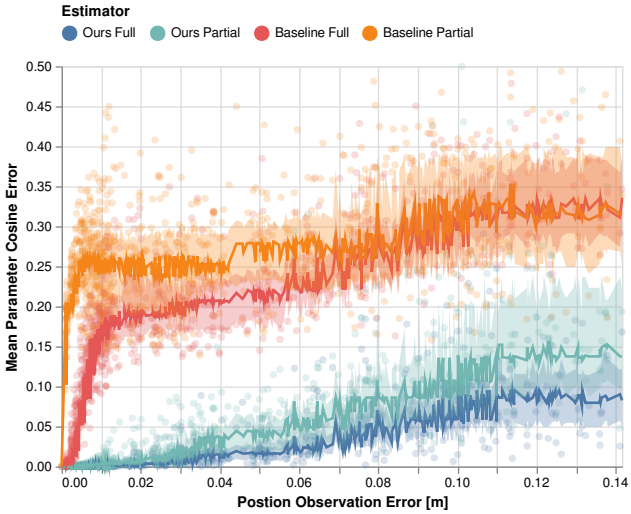
Figure 2.3a shows that both our method and the baseline recover the true parameters θ reliably even for partial observations, if the observations are noiseless. However, the performance of the baseline degrades rapidly with increasing noise variance. This pattern is particularly pronounced in the setting of partial observations. On the other hand, our estimator recovers the unknown cost parameters more accurately in both settings, and with a smaller variance than the baseline. Thus, compared to the KKT residual baseline, the performance of our method degrades gracefully when both full and partial observations are corrupted by noise.

Next, we study these methods' relative performance as measured by reconstruction error, as shown in Figure 2.3b. Here, reconstruction error is measured according to

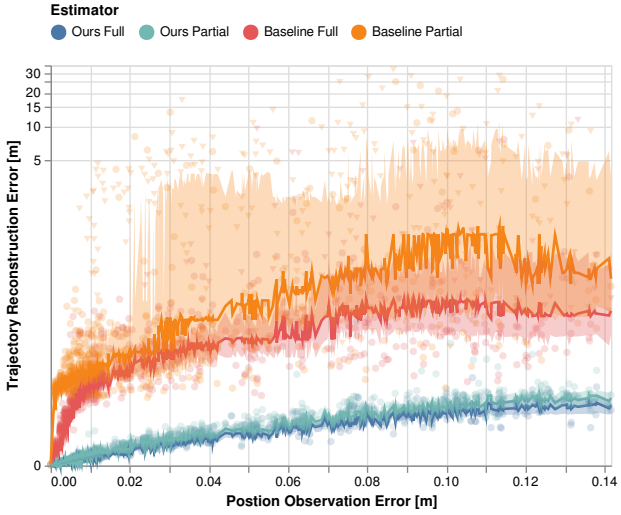
$$D_{\text{rec}}(\theta_{\text{true}}, \theta_{\text{est}}) = \frac{1}{NT} \sum_{i \in [N]} \sum_{t \in [T]} \|p_{\text{rec},t}^{i} - p_{\text{true},t}^{i}\|_2, \tag{2.25}$$

where $p_{\text{true},t}^{i}$ denotes the true position of Player^{i} at time step t, and $p_{\text{rec},t}^{i}$ denotes the position reconstructed from a Nash solution to the game with estimated cost parameters θ_{est}. We see similar patterns here as in the parameter error space, indicating the reliability of our method in both noisy full and partial observation settings.

Additionally, note that we have denoted some data points for the baseline method with triangular markers. For these Monte Carlo samples, the learned parameters θ_{est} specify ill-conditioned objectives that prevent us from recovering roots of (2.8)—essentially rendering the parameter estimates useless for downstream applications. This can happen, for example, when proximity costs dominate control input costs. For the baseline, a total of 104 out of 880 estimates result in an ill-conditioned forward game when states are fully observed. In the case of partial observations, the number of learning failures increases to 218. In contrast, our method recovers well-conditioned player objectives for all demonstrations and allows for accurate reconstruction of the game trajectory.



(a) Parameter estimation



(b) Trajectory reconstruction

Figure 2.3: Estimation performance of our method and the baseline for the 2-player collision-avoidance example, with noisy full and partial state observations. (a) Error measured directly in parameter space using (2.24). (b) Error measured in position space using (2.25). Triangular data markers in (b) highlight objective estimates which lead to ill-conditioned games. Solid lines and ribbons indicate the median and IQR of the error for each case.

For additional intuition of the performance gap, Figure 2.4 visualizes the reconstruction results in trajectory space for a fixed initial condition. Figure 2.4a shows the noise corrupted demonstrations generated for isotropic AWGN with standard deviation $\sigma = 0.1$. Figure 2.4b and Figure 2.4c show the corresponding trajectories reconstructed by solving the game using the objective parameters learned by our method and the baseline, respectively. Note that our method generates a far smaller fraction of outliers than the baseline. Furthermore, the performance of our method is only marginally affected by partial state observability, whereas baseline performance degrades substantially.

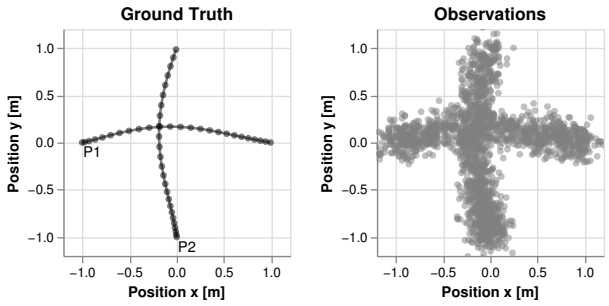
Online Learning with Prediction

Next, we study the performance of both our proposed method and the KKT residual baseline in the setting of objective learning with prediction. Following the problem description of Section 2.5.2, here, only the beginning of an unfolding dynamic game is observed. This problem naturally describes a single time frame of online operations where observations accumulate as time evolves.

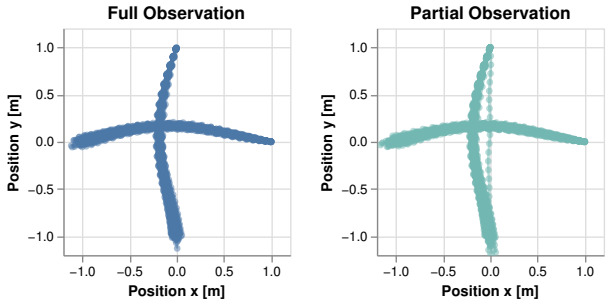
We conduct a Monte Carlo analysis of the two-player collision-avoidance game from Section 2.7.1 in which we vary the number of observed time steps of a fixed-length game. For this truncated observation sequence, each method is tasked to learn the players' underlying cost parameters θ^i *and* predict their motion for the next $s_\mathrm{p} = 10$ time steps. Our method accomplishes these coupled tasks jointly by solving (2.18). The KKT residual baseline, however, operates on the estimates provided by the preceding smoothing step, therefore, cannot couple unobserved, future time steps with cost inference. Instead, it achieves this task in a two-stage procedure: First, parameter estimates are recovered from a truncated game over only the observed $\tilde{T}$ time steps. With these parameters in hand, the baseline then predicts future game states by re-solving a forward game starting from the final state estimate $\tilde{x}_{\tilde{T}}$ with time steps simulated from $t \in \{\tilde{T}, \dots, \tilde{T} + 10\}$.

In Figure 2.5, we vary the observation horizon $\tilde{T} \in \{5, \dots, 15\}$ for a ground-truth game played over 25 time steps. For each value of $\tilde{T}$, we sample 40 sequences of observations $\{y_t\}_{t=1}^{\tilde{T}}$. Here, we fix an isotropic Gaussian noise level of $\sigma = 0.05$, and measure the performance of both our method and the baseline using two distinct metrics. In Figure 2.5a, we measure learning performance in parameter space using the metric given in (2.24). As shown, our approach consistently estimates the cost parameters more accurately than the baseline. Furthermore, as the observation horizon $\tilde{T}$ increases, both methods improve. In Figure 2.5b, we see that these patterns persist when we measure performance in trajectory space, applying the metric of (2.25) to the predicted states $x_t, t \in \{\tilde{T}, \dots, \tilde{T}+10\}$. Indeed, in this case, the performance gap is even more pronounced. By observing only $\tilde{T} = 5$ steps, our method reliably outperforms the baseline even when the baseline is given triple the number of observations.

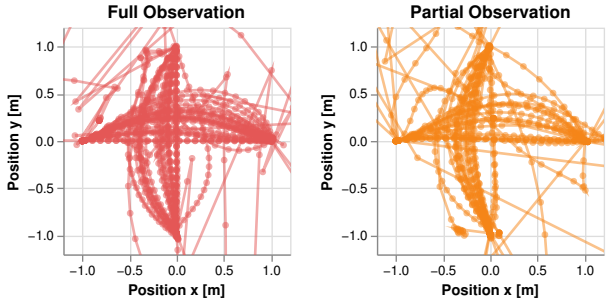
To inspect these results more closely, in Figure 2.6 we show the output of both methods for a single observation sequence of length $\tilde{T} = 10$. This visualization highlights a key advantage of our approach compared with the baseline. In this scenario, Player^2 (bottom) turns left early on in order to avoid Player^1 (left) later along the path to its goal. Their ground truth trajectories are shown in black. However, the methods only receive noise-corrupted partial state observations of the first $\tilde{T} = 10$ time steps shown in gray.



(a) Demonstrations

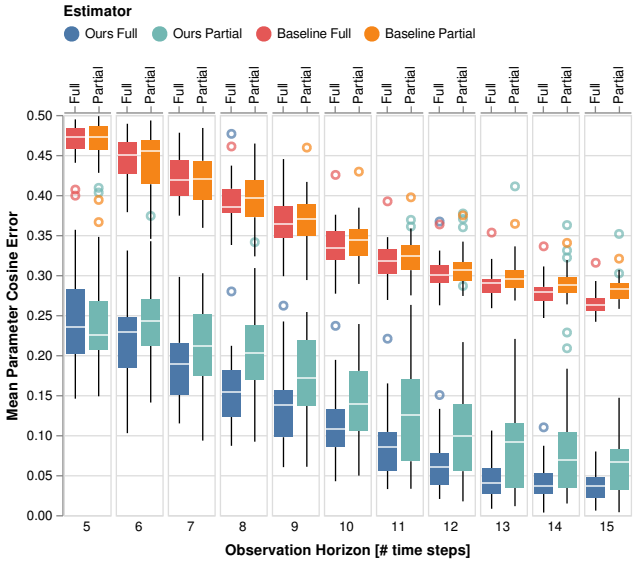


(b) Ours

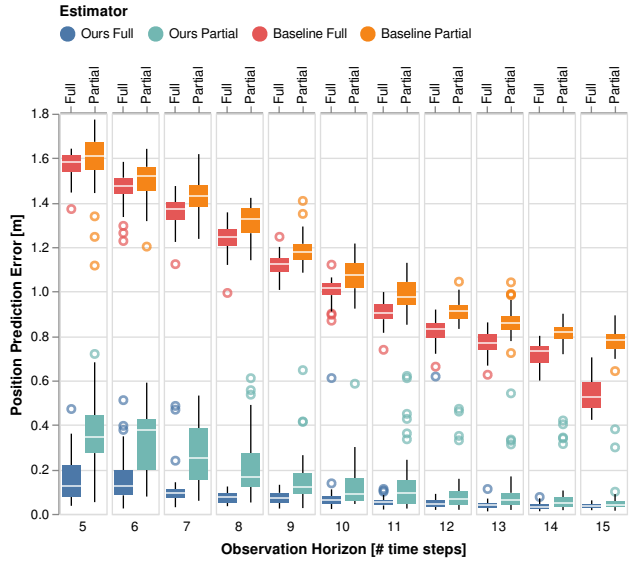


(c) Baseline

Figure 2.4: Qualitative reconstruction performance for the 2-player collision avoidance example at noise level $\sigma = 0.1$ for 40 different observation sequences. (a) Ground truth trajectory and observations, where each player wishes to reach a goal location opposite their initial position. (b, c) Trajectories recovered by solving the game at the estimated parameters for our method and the baseline using noisy full and partial state observations.



(a)



(b)

Figure 2.5: Estimation performance for our method and the baseline for varying numbers of observations of the 2-player collision-avoidance example at a fixed noise level of $\sigma = 0.05$. (a) Estimation performance measured directly in parameter space using (2.24). (b) Prediction error over the next 10 s beyond the observation horizon using (2.25).

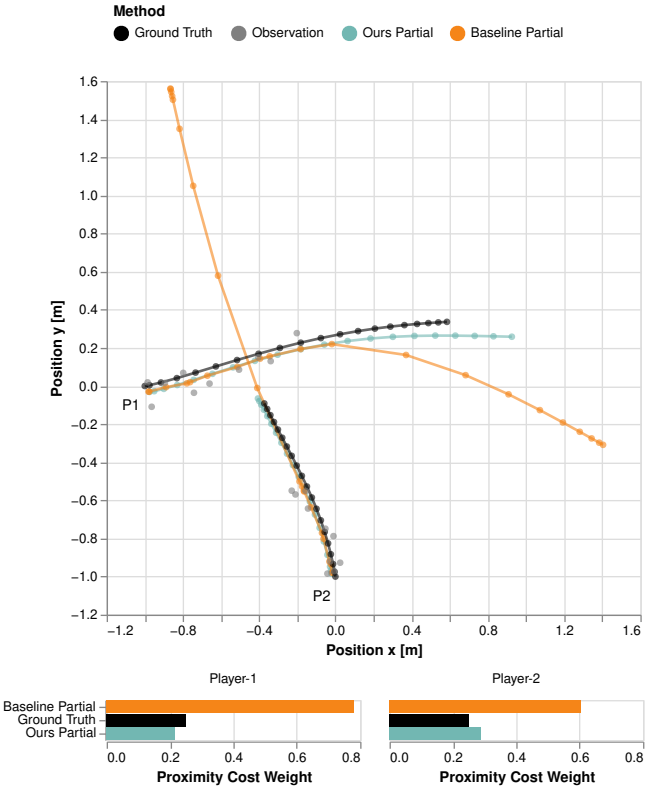


Figure 2.6: Qualitative prediction performance of our method and the baseline for the 2-player collision avoidance example when only the first 10 out of 25 time steps are observed.

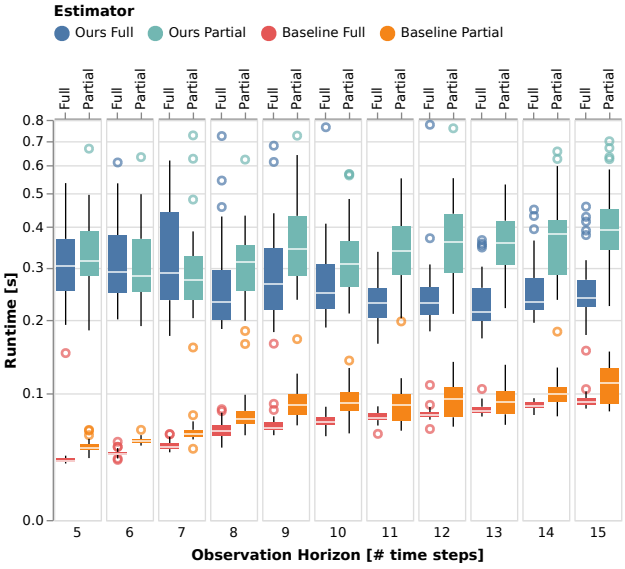


Figure 2.7: Runtime of our method and the baseline for varying numbers of observations of the 2-player collision-avoidance example at a fixed noise level of $\sigma = 0.05$.

Our method models the players' interactions as continuing into the future, allowing it to attribute observed behavior to future costs. In this instance, our method correctly explains Player^2's observed left turn as the result of a modest penalty on proximity, which becomes important only later in the trajectory when the players are close to one another. Cost estimation is shown at the bottom of Figure 2.6. The KKT residual baseline is incapable of such attributions. More precisely, it can only consider the KKT residuals $\mathbf{G}(\cdot;\theta)$ of (2.21) for time steps $t \in [\tilde{T}]$. Hence, the baseline must presume that the game terminates at $\tilde{T}$ rather than at some time in the future. Thus, it cannot anticipate the immediate future consequences of particular cost models. In Figure 2.6, the baseline can only explain the players' early observed collision avoidance maneuver with an extremely large penalty on proximity to their opponents. As a result, it predicts that the players will quickly drive away from one another. Unlike our method, the baseline's prediction rapidly diverges from the ground truth.

Beyond inference and prediction accuracy, a key factor for online operation is the computational complexity. To investigate this point, Figure 2.7 shows the computation time of both methods for the same dataset underpinning Figure 2.5. These timing results were obtained on a AMD Ryzen 9 5900HX laptop CPU. Overall, we observe that the KKT residual baseline has a lower runtime than our approach. The reduced runtime can be attributed to the fact that, by fixing the states and inputs a priori, the KKT residual formulation yields a simpler *convex* optimization problem in (2.21). Nonetheless, our method's runtime still remains moderate and scales gracefully with the observation horizon. We note that our current implementation is not optimized for speed. In practical applications in the context of receding-horizon applications—a topic that we shall discuss in Section 2.7.3—the

runtime may be further reduced via improved warm-starting and memory sharing across planner invocations.

2.7.3 Scaling to Larger Games

While our approach is more easily analyzed in the small, two-player collision-avoidance game of Section 2.7.2, it readily extends to larger multi-agent interactions. In order to demonstrate scalability of the approach, we therefore replicate the offline learning analysis of Section 2.7.2 in a larger 5-player highway driving scenario depicted in Figure 2.1. Finally, we demonstrate a proof of concept for online, receding-horizon learning in this scaled setting following the setup of Section 2.5.2.

In the highway scenario discussed through the remainder of this section, each player wishes to make forward progress in a particular lane at an unknown nominal speed, rather than reach a desired position as above. Therefore, ground-truth objectives use a quadratic penalty on deviation from a desired state that encodes each player's target lane and preferred travel speed rather than a specific goal location. Despite these differences, this class of objectives is still captured by the cost structure introduced in (2.23).

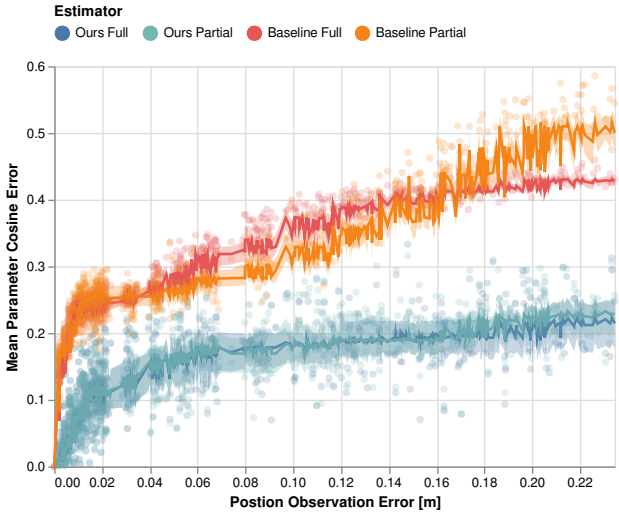
Offline Learning

First, we study the performance of our method and the KKT residual baseline in the setting of offline learning without trajectory prediction. Figure 2.8 displays these results, using the same metrics as in Section 2.7.2 to measure performance in parameter space—Figure 2.8a—and position space—Figure 2.8b. As before, our method demonstrably outperforms the baseline in both fully and partially observed settings. Furthermore, whereas our method performs comparably according to both metrics in the full and partial observation settings, the baseline performance differs between the two metrics. That is, while the performance of the baseline measured in parameter space is not significantly effected by less informative observations, the effect is significant in trajectory space. This inconsistency can be attributed to the fact that certain objective parameters have stronger influence on the resulting game trajectory than others. Since our method's objective is observation fidelity, here measured by the measurement likelihood of (2.13a), it directly accounts for these varying sensitivities. The baseline, however, greedily optimizes the KKT residual of (2.21), irrespective of the resulting equilibrium trajectory.

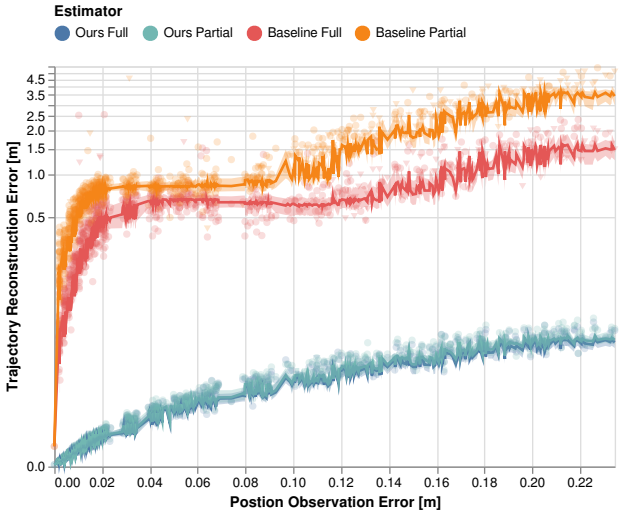
Online Learning and Receding-Horizon Prediction

Finally, we demonstrate the application of our method for simultaneous online learning and receding-horizon prediction in the 5-player highway navigation scenario depicted in Figure 2.1.

Here, the information available to the estimator evolves over time and the problem only admits access to *past* observations of the game state for cost learning. Following the proposed procedure of Section 2.5.2, here, we limit the computational complexity of the estimation problem by considering only a fixed-lag buffer of observations over the last $5s$ and predict all player's behavior over the next $10s$. The qualitative performance of our method under noise-corrupted partial state observation is shown in Figure 2.9. As can be



(a) Parameter estimation



(b) Trajectory Reconstruction

Figure 2.8: Estimation performance of our method and the baseline for the 5-player highway overtaking example, with noisy full and partial state observations. (a) Error measured directly in parameter space using (2.24). (b) Error measured in position space using (2.25). Triangular data markers in (b) highlight objective estimates which lead to ill-conditioned games. Solid lines and ribbons indicate the median and IQR of the error for each case.

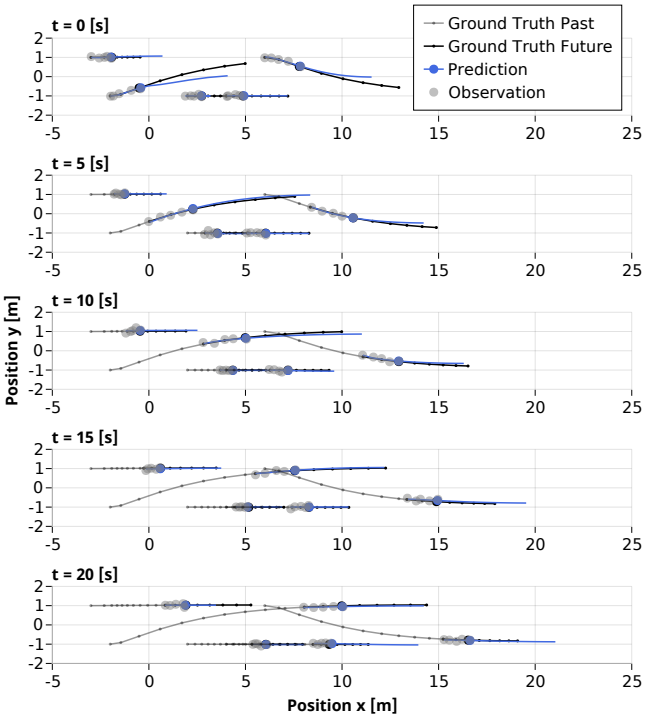


Figure 2.9: Demonstration of our method in an *online* application of simultaneous objective learning and trajectory prediction for the 5-player highway navigation scenario. At each time step, objective learning is performed on a fixed-lag buffer of 5 s of observation data which is coupled with trajectory prediction 10 s into the future.

seen, from only a few seconds of data, our method learns player objectives that accurately predict the evolution of the game over a receding prediction horizon. Note that, by design, objective learning and behavior prediction is achieved *simultaneously* by solving a single joint optimization problem as in (2.13). This ability to couple online learning and prediction makes it particularly suitable for online applications.

2.8 Conclusion

In this work, we have introduced a novel approach to learn the parameters of players' objectives in dynamic, noncooperative interactions, given only noisy, partial observations. This *inverse* dynamic game arises in a wide variety of multi-robot and human-robot interactions and generalizes well-studied problems such as inverse optimal control, inverse reinforcement learning, and learning from demonstrations. Contrary to prior work, our method learns players' cost parameters while *simultaneously* recovering the forward game trajectory consistent with those parameters, with overall performance measured according to observation fidelity. We have shown how this formulation naturally extends to both *offline* learning and prediction problems, as well as *online*, receding-horizon learning.

We have conducted extensive numerical simulations to characterize the performance of our method and compare it to a state-of-the-art baseline method [54, 56]. These simulations clearly demonstrate our method's improved robustness to both observation noise and partial observations. Indeed, existing methods presume noiseless, full state observations and thus require *a priori* estimation of states and inputs. Our method recovers objective parameters, reconstructs past game trajectories, and predicts future trajectories far more accurately than the baseline. Beyond that, our method's structure allows performing all of these tasks jointly as the solution of a single optimization problem. This feature renders our method suitable for online learning and prediction in a receding-horizon fashion.

In light of these encouraging results, there are several directions for future research. Most immediately, our method lends itself naturally to deployment onboard physical robotic systems such as the autonomous vehicles considered in the examples of Section 2.7. In particular, the online, receding-horizon learning and prediction procedure of Section 2.5.2 may be run onboard an autonomous car. Here, the "ego" agent would seek to learn other vehicles' objective parameters while simultaneously using the receding-horizon game solution to respond to predicted opponent strategies.

Another exciting, more theoretical direction consists of extending our formulation to more complex equilibrium concepts than OLNE. For example, recent solution methods for forward games in state feedback Nash equilibria [46, 67, 77] might be adapted to solve inverse games along the lines of (2.12).

Chapter 3

Learning to Interact with Agents That Have Unknown Objectives

In Chapter 2, we proposed a method for estimating players' objectives in dynamic games from noise-corrupted, partial state observations by casting the problem as constrained maximum likelihood estimation (MLE) and solving it via canonical constrained optimization methods. However, results in the previous chapter were limited to inference.

This chapter builds on the MLE inverse game formulation of Chapter 2, and moves beyond inference by integrating inferred intents into online decision-making. To enable this, we develop a new solution technique for the MLE inverse game problem that (i) handles inequality constraints for safer interaction and (ii) yields a simple, first-order update rule for parameters that composes naturally with neural networks (NNs) training for amortized inference. Large-scale simulations show that our approach outperforms game-theoretic and non-game-theoretic baselines in terms of safety and interaction efficiency. Finally, this chapter demonstrates the method's real-time planning capabilities and robustness through hardware experiments of interactions between mobile robots and pedestrians.

This chapter is a verbatim copy, with minor modifications, of the peer-reviewed journal article [78]:

Lasse Peters*, Xinjie Liu*, Javier Alonso-Mora. "Learning to Play Trajectory Games Against Opponents with Unknown Objectives." *IEEE Robotics and Automation Letters (RA-L)*, 2023.

* indicates equal contribution.

Contribution statement: Lasse proposed the key idea of solving MLE inverse games via differentiable programming, inspired by his prior work in [42, 79, 80]. Lasse and Xinjie jointly implemented the underpinning differentiable trajectory game solver. Under Lasse's guidance, Xinjie performed all experiments. Lasse and Xinjie wrote the initial draft of the manuscript. All authors contributed to technical discussions and edits of the original manuscript submitted to RA-L [78].

3.1 Introduction

Many robot planning problems, such as robot navigation in a crowded environment, involve rich interactions with other agents. Classic "predict-then-plan" frameworks neglect the fact that other agents in the scene are responsive to the ego-agent's actions. This simplification can result in inefficient or even unsafe behavior [81]. Dynamic game theory explicitly models the interactions as coupled trajectory optimization problems from a multi-agent perspective. A noncooperative equilibrium solution of this game-theoretic model then provides strategies for all players that account for the strategic coupling of plans. Beyond that, general constraints between players, such as collision avoidance, can also be handled explicitly. All of these features render game-theoretic reasoning an attractive approach to interactive motion planning.

In order to apply game-theoretic methods for interactive motion planning from an *egocentric* rather than *omniscient* perspective, such methods must be capable of operating only based on local information. For instance, in driving scenarios as shown in Figure 3.1, the red ego-vehicle may only have partial-state observations of the surrounding vehicles and incomplete knowledge of their objectives due to unknown preferences for travel velocity, target lane, or driving style. Since vanilla game-theoretic methods require an objective model of *all* players [46, 82], this requirement constitutes a key obstacle in applying such techniques for autonomous strategic decision-making.

To address this challenge, we introduce our main contribution: a model-predictive game solver, which adapts to unknown opponents' objectives and solves for generalized Nash equilibrium (GNE) strategies. The adaptivity of our approach is enabled by a differentiable trajectory game solver whose gradient signal is used for MLE of opponents' objectives.

We perform thorough experiments in simulation and on hardware to support the following three key claims: our solver (i) outperforms both game-theoretic and non-game-theoretic baselines in highly interactive scenarios, (ii) can be combined with other differentiable components such as NNs, and (iii) is fast and robust enough for real-time planning on a hardware platform.

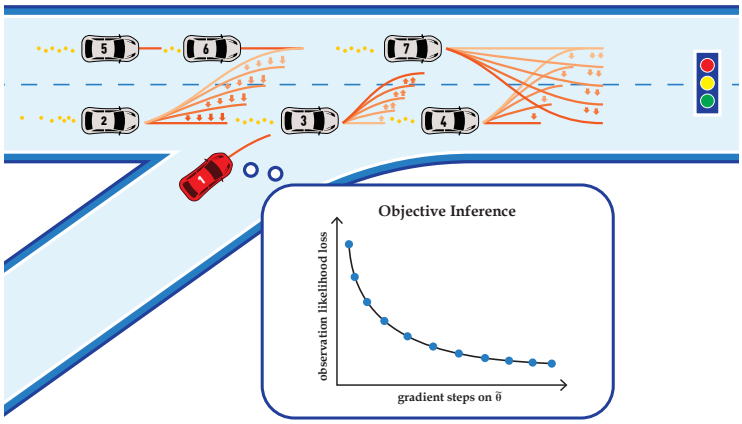


Figure 3.1: An ego-agent (red) merging onto a busy road populated by six surrounding vehicles whose preferences for travel velocity and lane are initially unknown. Our approach adapts the ego agent's strategy by inferring opponents' intention parameters $\tilde{\theta}$ from partial state observations.

3.2 Related Work

To put our contribution into context, this section discusses four main bodies of related work. First, we discuss works on trajectory games which assume access to the objectives of all players in the scene. Then, we introduce works on inverse dynamic games that infer unknown objectives from data. Thereafter, we also relate our work to non-game-theoretic interaction-aware planning-techniques. Finally, we survey recent advances in differentiable optimization, which provide the underpinning for our proposed differentiable game solver.

3.2.1 N-Player General-Sum Dynamic Games

Dynamic games are well-studied in the literature [45]. In robotics, a particular focus is on multi-player general-sum games in which players may have differing yet non-adversarial objectives, and states and inputs are continuous.

Various equilibrium concepts exist in dynamic games. The Stackelberg equilibrium concept [83] assumes a "leader-follower" hierarchy, while the Nash equilibrium problem (NEP) [46, 83] does not presume such a hierarchy. Within the scope of NEP, there exist open-loop NEPs [82] and feedback NEPs [46, 67]. We refer the readers to [45] for more details about the difference between the concepts. When shared constraints exist between players, such as collision avoidance constraints, one player's feasible set may depend on other players' decisions. In that case, the problem becomes a generalized Nash equilibrium problem (GNEP) [84]. In this work, we focus on GNEPs under an open-loop information pattern which we solve by converting to an equivalent MCP [85].

3.2.2 Inverse Games

There are three main paradigms for solving inverse games: (i) Bayesian inference, (ii) minimization of KKT residuals, and (iii) equilibrium-constrained maximum-likelihood estimation. In type (i) methods, Le Cleac'h et al. [63] employ an Unscented Kalman Filter (UKF). This sigma-point sampling scheme drastically reduces the sampling complexity compared to vanilla particle filtering. However, a UKF is only applicable for uni-modal distributions, and extra care needs to be taken when uncertainty is multi-modal, e.g., due to multiple Nash equilibria. Type (ii) methods require *full* demonstration trajectories, i.e., including noise-free states and inputs, to cast the N-player inverse game as N independent unconstrained optimization problems [54, 56]. However, they assume full constraint satisfaction at the demonstration and have limited scalability with noisy data [79]. The type (iii) methods use KKT conditions of an OLNE as constraints to formulate a constrained optimization problem [79]. This type of method finds the same solution as type (ii) methods in the noise-free cases but can additionally handle partial and noisy state observations. However, encoding the equilibrium constraints is challenging, as it typically yields a nonconvex problem, even in relatively simple linear-quadratic game settings. This challenge is even more pronounced when considering inequality constraints of the observed game, as this results in complementarity constraints in the inverse problem.

Our solution approach also matches the observed trajectory data in an MLE framework. In contrast to all methods above, we do so by making a generalized Nash equilibrium (GNE) solver differentiable. This approach yields two important benefits over existing methods: (i) general (coupled) inequality constraints can be handled explicitly, and (ii) the entire pipeline supports direct integration with other differentiable elements, such as NNs. This latter benefit is a key motivation for our approach that is not enabled by the formulations in [63] and [79].

Note that Geiger and Straehle [86] explore a similar differentiable pipeline for inference of game parameters. In contrast to their work, however, our method is not limited to the special class of potential games and applies to general GNEPs.

3.2.3 Non-Game-Theoretic Interaction Models

Besides game-theoretic methods, two categories of interaction-aware decision-making techniques have been studied extensively in the context of collision avoidance and autonomous driving: (i) approaches that learn a navigation policy for the ego-agent directly without explicitly modeling the responses of others [87–89], and (ii) techniques that explicitly predict the opponents' actions to inform the ego-agent's decisions [57, 90–93]. This latter category may be further split by the granularity of coupling between the ego-agent's decision-making process and the predictions of others. In the simplest case, prediction depends only upon the current physical state of other agents [94]. More advanced interaction models condition the behavior prediction on additional information such as the interaction history [57], the ego-agent's goal [91, 92], or even the ego-agent's future trajectory [90, 93].

Our approach is most closely related to this latter body of work: by solving a trajectory game, our method captures the interdependence of future decisions of all agents; and by additionally inferring the objectives of others, predictions are conditioned on the in-

teraction history. However, a key difference of our method is that it explicitly models others as rational agents unilaterally optimizing their own cost. This assumption provides additional structure and offers a level of interpretability of the inferred behavior.

3.2.4 Differentiable Optimization

Our work is enabled by differentiating through a GNE solver. Several works have explored the idea of propagating gradient information through optimization algorithms [95–97], enabling more expressive neural architectures. However, these works focus on optimization problems and thus only apply to special cases of games, such as potential games studied by Geiger and Straehle [86]. By contrast, differentiating through a GNEP involves N *coupled* optimization problems. We address this challenge in section 3.4.2.

3.3 Preliminaries

This section introduces two key concepts underpinning our work: forward and inverse dynamic games. In *forward* games, the objectives of players are known, and the task is to find players' strategies. By contrast, *inverse* games take (partial) observations of strategies as inputs to recover initially *unknown objectives*. In Section 3.4, we combine these two approaches into an adaptive solver that computes forward game solutions while estimating player objectives.

3.3.1 General-Sum Trajectory Games

Consider an N-player discrete-time general-sum trajectory game with horizon of T. In this setting, each player i has a control input $u_t^i \in \mathbb{R}^{m^i}$ which they may use to influence their state $x_t^i \in \mathbb{R}^{n^i}$ at each discrete time $t \in [T]$. In this work, we assume that the evolution of each player's state is characterized by an individual dynamical system $x_{t+1}^i = f^i(x_t^i, u_t^i)$. For brevity throughout the remainder of this chapter, we shall use boldface to indicate aggregation over players and capitalization for aggregation over time, e.g., $\mathbf{x}_t := (x_t^1, \ldots, x_t^N)$, $U^i := (u_1^i, \ldots, u_T^i)$, $\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_T)$. With a joint trajectory starting at a given initial state $\hat{\mathbf{x}}_1 := (\hat{x}_1^1, \ldots, \hat{x}_1^N)$, each player seeks to find a control sequence U^i to minimize their own cost function $J^i(\mathbf{X}, U^i; \theta^i)$, which depends upon the joint state trajectory $\mathbf{X}$ as well as the player's control input sequence U^i and, additionally, takes in a parameter vector θ^i.[1] Each player must additionally consider private inequality constraints ${}^p g^i(X^i, U^i) \geq 0$ as well as shared constraints ${}^s g(\mathbf{X}, \mathbf{U}) \geq 0$. This latter type of constraint is characterized by the fact that all players have a shared responsibility to satisfy it, with a common example being collision avoidance constraints between players. In summary, this noncooperative trajectory game can be cast as a tuple of N coupled

[1] The role of the parameters will become clear later in this chapter when we move on to *inverse* dynamic games.

trajectory optimization problems:

$$\forall i \in [N] \begin{cases} \min\limits_{X^i, U^i} & J^i(\mathbf{X}, U^i; \theta^i) \\ \text{s.t.} & x^i_{t+1} = f^i(x^i_t, u^i_t), \forall t \in [T-1] \\ & x^i_1 = \hat{x}^i_1 \\ & {}^p g^i(X^i, U^i) \geq 0 \\ & {}^s g(\mathbf{X}, \mathbf{U}) \geq 0. \end{cases} \tag{3.1}$$

Note that each player's feasible set in this problem may depend upon the decision variables of others, which makes it a GNEP rather than a standard NEP [84].

A solution of this problem is a tuple of GNE strategies $\mathbf{U}^* := (U^{1*}, \ldots, U^{N*})$ that satisfies the inequalities $J^i(\mathbf{X}^*, U^{i*}; \theta^i) \leq J^i((X^i, \mathbf{X}^{\neg i*}), U^i; \theta^i)$ for any feasible deviation (X^i, U^i) of any player i, with $\mathbf{X}^{\neg i}$ denoting all but player i's states. Since identifying a global GNE is generally intractable, we require these conditions only to hold locally. At a local GNE, then, no player has a unilateral incentive to deviate *locally* in feasible directions to reduce their cost.

Running example: We introduce a simple running example[2] which we shall use throughout the presentation to concretize the key concepts. Consider a tracking game played between $N = 2$ players. Let each agent's dynamics be characterized by those of a planar double-integrator, where states $x^i_t = (p^i_{x,t}, p^i_{y,t}, v^i_{x,t}, v^i_{y,t})$ are position and velocity, and control inputs $u^i_t = (a^i_{x,t}, a^i_{y,t})$ are acceleration in horizontal and vertical axes in a Cartesian frame. We define the game's state as the concatenation of the two players' individual states $\mathbf{x}_t := (x^1_t, x^2_t)$. Each player's objective is characterized by an individual cost

$$J^i = \sum_{t=1}^{T-1} \|p^i_{t+1} - p^i_{\text{goal}}\|_2^2 + 0.1\|u^i_t\|_2^2 + 50 \max(0, d_{\min} - \|p^i_{t+1} - p^{-i}_{t+1}\|_2)^3, \tag{3.2}$$

where we set $p^1_{\text{goal}} = p^2_t$ so that player 1, the tracking robot, is tasked to track player 2, the target robot. Player 2 has a fixed goal point p^2_{goal}. Both agents wish to get to their goal position efficiently while avoiding proximity beyond a minimal distance $d_{\min}$. Players also have shared collision avoidance constraints ${}^s g_{t+1}(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) = \|p^1_{t+1} - p^2_{t+1}\|_2 - d_{\min} \geq 0, \forall t \in [T-1]$ and private bounds on state and controls ${}^p g^i(X^i, U^i)$. Agents need to negotiate and find an underlying equilibrium strategy in this noncooperative game, as no one wants to deviate from the direct path to their goal.

3.3.2 Inverse Games

We now switch context to the *inverse* dynamic game setting. Let $\theta := (\hat{\mathbf{x}}_1, \theta^2, ..., \theta^N)$ denote the aggregated tuple of parameters initially unknown to the ego-agent with index 1. Note that we explicitly infer the initial state of a game $\hat{\mathbf{x}}_1$ to account for the potential

[2] Our final evaluation in Section 3.5 features denser interaction such as the 7-player ramp-merging scenario shown in Figure 3.1.

sensing noise and partial state observations. To model the inference task over these parameters, we assume that the ego-agent observes behavior originating from an unknown Nash game $\Gamma(\theta) := (\hat{\mathbf{x}}_1, {}^s g, \{{}^p g^i, J^i(\cdot;\theta^i)\}_{i\in[N]})$, with objective functions and constraints parameterized by initially unknown values θ^i and $\hat{\mathbf{x}}_1$, respectively.

Similar to the existing method [79], we employ an MLE formulation to allow observations to be *partial* and *noise-corrupted*. In contrast to that method, however, we also allow for inequality constraints in the hidden game. That is, we propose to solve

$$\begin{aligned} \max_{\theta,\mathbf{X},\mathbf{U}} \quad & p(\mathbf{Y} \mid \mathbf{X}, \mathbf{U}) \\ \text{s.t.} \quad & (\mathbf{X}, \mathbf{U}) \text{ is a GNE of } \Gamma(\theta) \end{aligned} \tag{3.3}$$

where $p(\mathbf{Y} \mid \mathbf{X}, \mathbf{U})$ denotes the likelihood of observations $\mathbf{Y} := (\mathbf{y}_1, ..., \mathbf{y}_T)$ given the estimated game trajectory $(\mathbf{X}, \mathbf{U})$ induced by parameters θ. This formulation yields an *mathematical program with equilibrium constraints (MPEC)* [69], where the outer problem is an estimation problem while the inner problem involves solving a dynamic game. When the observed game includes inequality constraints, the resulting inverse problem necessarily contains complementarity constraints and only few tools are available to solve the resulting problem. In the next section, we show how to transform (3.3) into an unconstrained problem by making the inner game differentiable, which also enables combination with other differentiable components.

Running example: We assign the tracker (player 1) to be the ego-agent and parameterize the game with the goal position of the target robot $\theta^2 = p^2_{\text{goal}}$. That is, the tracker does not know the target agent's goal and tries to infer this parameter from position observations. To ensure that (3.3) remains tractable, the ego-agent maintains only a fixed-length buffer of observed opponent's positions. Note that solving the inverse game requires solving games rather than optimal control problems at the inner level to account for the noncooperative nature of observed interactions, which is different from IOC even in the 2-player case. We employ a Gaussian observation model, which we represent with an equivalent negative log-likelihood objective $\|\mathbf{Y} - r(\mathbf{X}, \mathbf{U})\|_2^2$ in (3.3), where $r(\mathbf{X}, \mathbf{U})$ maps $(\mathbf{X}, \mathbf{U})$ to the corresponding sequence of expected positions.

3.4 Adaptive Model-Predictive Game Play

We wish to solve the problem of MPGP from an ego-centric perspective, i.e., without prior knowledge of other players' objectives. To this end, we present an adaptive model-predictive game solver that combines the tools of Section 3.3: first, we perform MLE of unknown objectives by solving an *inverse game* (Section 3.3.2); then, we solve a *forward game* using this estimate to recover a strategic motion plan (Section 3.3.1).

3.4.1 Forward Games as MCPs

We first discuss the conversion of the GNEP in (3.1) to an equivalent MCP. There are three main advantages of taking this view. First, there exists a wide range of off-the-shelf solvers for this problem class [98]. Furthermore, MCP solvers directly recover strategies for all players *simultaneously*. Finally, this formulation makes it easier to reason about

derivatives of the solution w.r.t. problem data. As we shall discuss in Section 3.4.3, this derivative information can be leveraged to solve the inverse game problem of (3.3).

In order to solve the GNEP presented in (3.1) we derive its first-order necessary conditions. We collect all equality constraints for player i in (3.1) into a vector-valued function $h^i(X^i, U^i; \hat{x}_1^i)$, introduce Lagrange multipliers $\mu^i, {}^p\lambda^i$ and ${}^s\lambda$ for constraints $h^i(X^i, U^i; \hat{x}_1^i)$, ${}^pg^i(X^i, U^i)$, and ${}^sg(\mathbf{X}, \mathbf{U})$ and write the Lagrangian for player i as

$$\begin{aligned}\mathcal{L}^i(\mathbf{X}, \mathbf{U}, \mu^i, {}^p\lambda^i, {}^s\lambda; \theta) &= J^i(\mathbf{X}, \mathbf{U}; \theta^i) \\ &+ \mu^{i\top} h^i(X^i, U^i; \hat{x}_1^i) - {}^s\lambda^\top {}^sg(\mathbf{X}, \mathbf{U}) - {}^p\lambda^{i\top} {}^pg^i(X^i, U^i).\end{aligned} \tag{3.4}$$

Note that we share the multipliers associated with shared constraints between the players to encode equal constraint satisfaction responsibility [99]. Under mild regularity conditions, e.g., linear independence constraint qualification (LICQ), a solution of (3.1) must satisfy the following joint KKT conditions:

$$\forall i \in [N] \begin{cases} \nabla_{(X^i, U^i)} \mathcal{L}^i(\mathbf{X}, \mathbf{U}, \mu^i, {}^p\lambda^i, {}^s\lambda; \theta) = 0 \\ 0 \leq {}^pg^i(X^i, U^i) \perp {}^p\lambda^i \geq 0 \\ h(\mathbf{X}, \mathbf{U}; \hat{\mathbf{x}}_1) = 0 \\ 0 \leq {}^sg(\mathbf{X}, \mathbf{U}) \perp {}^s\lambda \geq 0, \end{cases} \tag{3.5}$$

where, for brevity, we denote by $h(\mathbf{X}, \mathbf{U}; \hat{\mathbf{x}}_1)$ the aggregation of all equality constraints. If the second directional derivative of the Lagrangian is positive along all feasible directions at a solution of (3.5)—a condition that can be checked a posteriori—this point is also a solution of the original game. In this work, we solve trajectory games by viewing their KKT conditions through the lens of MCPs [85, Section 1.4.2].

Definition 3.1 *(Mixed Complementarity Problem (MCP)) A Mixed Complementarity Problem (MCP) is defined by the following problem data: a function $F(z) : \mathbb{R}^d \mapsto \mathbb{R}^d$, lower bounds $\ell_j \in \mathbb{R} \cup \{-\infty\}$ and upper bounds $u_j \in \mathbb{R} \cup \{\infty\}$, each for $j \in [d]$. The solution of an MCP is a vector $z^* \in \mathbb{R}^n$, such that for each element with index $j \in [d]$ one of the following equations holds:*

$$z_j^* = \ell_j, F_j(z^*) \geq 0 \tag{3.6a}$$

$$\ell_j < z_j^* < u_j, F_j(z^*) = 0 \tag{3.6b}$$

$$z_j^* = u_j, F_j(z^*) \leq 0. \tag{3.6c}$$

The parameterized KKT system of (3.5) can be expressed as a *parameterized family* of MCPs with decision variables corresponding to the primal and dual variables of (3.5),

$$z = \left[\mathbf{X}^\top, \mathbf{U}^\top, \boldsymbol{\mu}^\top, {}^p\lambda^{1\top}, \ldots, {}^p\lambda^{N\top}, {}^s\lambda^\top\right]^\top,$$

and problem data

$$F(z;\theta) = \begin{bmatrix} \nabla_{(X^1,U^1)}\mathcal{L}^i \\ \vdots \\ \nabla_{(X^N,U^N)}\mathcal{L}^N \\ h \\ {}^p g^1 \\ \vdots \\ {}^p g^N \\ {}^s g \end{bmatrix}, \quad \ell = \begin{bmatrix} -\infty \\ \vdots \\ -\infty \\ -\infty \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad u = \begin{bmatrix} \infty \\ \vdots \\ \infty \\ \infty \\ \infty \\ \vdots \\ \infty \\ \infty \end{bmatrix}, \tag{3.7}$$

where, by slight abuse of notation, we overload F to be parametrized by θ via $\mathcal{L}^i$ and use ∞ to denote elements for which upper or lower bounds are dropped.

3.4.2 Differentiation of an MCP solver

An MCP solver may be viewed as a function, mapping problem data to a solution vector. Taking this perspective, for a parameterized family of MCPs as in (3.7), we wish to compute the function's derivatives to answer the following question: How does the solution z^* respond to local changes of the problem parameters θ?

The Nominal Case

Let $\Psi(\theta) := (F(\cdot;\theta), \ell, u)$ denote an MCP parameterized by $\theta \in \mathbb{R}^p$ and let $z^* \in \mathbb{R}^n$ denote a solution of that MCP, which is implicitly a function of θ. For this nominal case, we consider only solutions at which *strict complementarity* holds. We shall relax this assumption later. If F is smooth, i.e., $F(\cdot;\theta), F(z^*;\cdot) \in C^1$, we can recover the Jacobian matrix $\nabla_\theta z^* = \left(\frac{\partial z_j^*}{\partial \theta_k}\right) \in \mathbb{R}^{n\times p}$ by distinguishing two possible cases. For brevity, below, gradients are understood to be evaluated at z^* and θ.

Active bounds Consider first the elements z_j^* that are either at their lower or upper bound, i.e., z_j^* satisfies (3.6a) or (3.6c). Since strict complementarity holds at the solution, $F_j(z^*;\theta)$ must be bounded away from zero with a finite margin. Hence, the smoothness of F guarantees that a local perturbation of θ will retain the sign of $F_j(z^*;\theta)$. As a result, z_j^* remains at its bound and, locally, is identically zero. Let $\tilde{\mathcal{I}} := \{k \in [n] \mid z_k^* = \ell_k \vee z_k^* = u_k\}$ denote the index set of all elements matching this condition and $\tilde{z}^* := [z^*]_{\tilde{\mathcal{I}}}$ denote the solution vector reduced to that set. Trivially, then, the Jacobian of this vector vanishes, i.e., $\nabla_\theta \tilde{z}^* = 0$.

Inactive bounds The second case comprises elements that are strictly between the bounds, i.e., z_j^* satisfying (3.6b). In this case, under mild assumptions on F, for any local perturbation of θ there exists a perturbed solution such that F remains at its root. Therefore, the gradient $\nabla_\theta z_j^*$ for these elements is generally non-zero, and we can compute it

via the implicit function theorem (IFT). Let $\bar{\mathcal{I}} := \{k \in [n] \mid F_k(z^*;\theta) = 0, \ell_k < z_k^* < u_k\}$ be the index set of all elements satisfying case (b) and let

$$\bar{z}^* := [z^*]_{\bar{\mathcal{I}}}, \quad \bar{F}(z^*,\theta) := [F(z^*;\theta)]_{\bar{\mathcal{I}}} \tag{3.8}$$

denote the solution vector and its complement reduced to said index set. By the IFT, the relationship between parameters θ and solution $z^*(\theta)$ is characterized by the stationarity of $\bar{F}$:

$$0 = \nabla_\theta \left[\bar{F}(z^*(\theta),\theta)\right] = \nabla_\theta \bar{F} + (\nabla_{\bar{z}^*}\bar{F})(\nabla_\theta \bar{z}^*) + (\nabla_{\bar{z}^*}\bar{F}) \underbrace{(\nabla_\theta \tilde{z}^*)}_{\equiv 0} \tag{3.9}$$

Note that, as per the discussion in case (a), the last term in this equation is identically zero. Hence, if the Jacobian $\nabla_{\bar{z}^*}\bar{F}$ is invertible, we recover the derivatives as the unique solution of the above system of equations,

$$\nabla_\theta \bar{z}^* = -\left(\nabla_{\bar{z}^*}\bar{F}\right)^{-1}(\nabla_\theta \bar{F}). \tag{3.10}$$

Note that (3.9) may not always have a unique solution, in which case (3.10) cannot be evaluated. We discuss practical considerations for this special case below.

Remarks on Special Cases and Practical Realization

The above derivation of gradients for the nominal case involves several assumptions on the structure of the problem. We discuss considerations to improve numerical robustness for practical realization of this approach below. We note that both special cases discussed hereafter are rare in practice. In fact, across 100 simulations of the running example with varying initial states and objectives, neither of them occurred.

Weak Complementarity The nominal case discussed above assumes strict complementarity at the solution. If this assumption does not hold, the derivative of the MCP is not defined. Nevertheless, we can still compute subderivatives at θ. Let the set of all indices for which this condition holds be denoted by $\hat{\mathcal{I}} := \{k \in [n] \mid F_k(z^*;\theta) = 0 \wedge z_k^* \in \{\ell_k, u_k\}\}$. Then by selecting a subset of $\hat{\mathcal{I}}$ and including it in $\bar{\mathcal{I}}$ for evaluation of (3.10), we recover a subderivative.

Invertibility The evaluation (3.10) requires invertibility of $\nabla_{\bar{z}^*}\bar{F}$. To this end, we compute the least-squares solution of (3.9) rather than explicitly inverting $\nabla_{\bar{z}}\bar{F}$.

3.4.3 Model-Predictive Game Play with Gradient Descent

Finally, we present our pipeline for adaptive game-play against opponents with unknown objectives. Our adaptive MPGP scheme is summarized in Algorithm 1. At each time step, we first update our estimate of the parameters by approximating the inverse game in (3.3) via gradient descent. To obtain an unconstrained optimization problem, we substitute the

Algorithm 1 Adaptive MPGP

Hyper-parameters: stopping tolerance: stop_tol, learning rate: lr
Input: initial $\tilde{\theta}$, current observation buffer $\mathbf{Y}$, new observation $\mathbf{y}$
$\mathbf{Y} \leftarrow \text{updateBuffer}(\mathbf{Y}, \mathbf{y})$
while not stop_tol and not max_steps_reached **do**
  $(z^*, \nabla_\theta z^*) \leftarrow \text{solveDiffMCP}(\tilde{\theta})$ ▷ sec. 3.4.2
  $\nabla_\theta p \leftarrow \text{composeGradient}(z^*, \nabla_\theta z^*, \mathbf{Y})$ ▷ eq. (3.12)
  $\tilde{\theta} \leftarrow \tilde{\theta} - \nabla_\theta p \cdot \text{lr}$
$z^* \leftarrow \text{solveMCP}(\tilde{\theta})$ ▷ forward game, eq. (3.7)
$\text{applyFirstEgoInput}(z^*)$
return $\tilde{\theta}, \mathbf{Y}$

constraints in (3.3) with our differentiable game solver. Following the discussion of (3.7), we denote by $z^*(\theta)$ the solution of the MCP formulation of the game parameterized by θ. Furthermore, by slight abuse of notation, we overload $\mathbf{X}(z^*), \mathbf{U}(z^*)$ to denote functions that extract the state and input vectors from z^*. Then, the inverse game of (3.3) can be written as unconstrained optimization,

$$\max_{\theta} \quad p(\mathbf{Y} \mid \mathbf{X}(z^*(\theta)), \mathbf{U}(z^*(\theta))). \tag{3.11}$$

Online, we approximate solutions of this problem by taking gradient descent steps on the negative logarithm of this objective, with gradients computed by chain rule,

$$\begin{aligned} \nabla_\theta \left[p(\mathbf{Y} \mid \mathbf{X}(z^*(\theta)), \mathbf{U}(z^*(\theta))\right] = \\ (\nabla_{\mathbf{X}} p)(\nabla_{z^*} \mathbf{X})(\nabla_\theta z^*) + (\nabla_{\mathbf{U}} p)(\nabla_{z^*} \mathbf{U})(\nabla_\theta z^*). \end{aligned} \tag{3.12}$$

Here, the only non-trivial term is $\nabla_\theta z^*$, whose computation we discussed in Section 3.4.2. To reduce the computational cost, we warm-start using the estimate of the previous time step and terminate early if a maximum number of steps is reached. Then, we solve a forward game parametrized by the estimated $\tilde{\theta}$ to compute control commands. We execute the first control input for the ego agent and repeat the procedure.

3.5 Experiments

To evaluate our method, we compare against two baselines in Monte Carlo studies of simulated interaction. Beyond these quantitative results, we showcase our method deployed on Jackal ground robots in two hardware experiments.

The experiments below are designed to support the key claims that our method (i) outperforms both game-theoretic and non-game-theoretic baselines in highly interactive scenarios, (ii) can be combined with other differentiable components such as NNs, and (iii) is sufficiently fast and robust for real-time planning on a hardware platform. A supplementary video of qualitative results as well as the code for our method and experiments can be found at `https://xinjie-liu.github.io/projects/game`.

3.5.1 Experiment Setup

Scenarios

We evaluate our method in two scenarios.

2-player running example To test the inference accuracy and convergence of our method in an intuitive setting, we first consider the 2-player running example. For evaluation in simulation, we sample the opponent's intent—i.e., their unknown goal position in (3.2)— uniformly from the environment. Partial observations comprise the position of each agent.

Ramp merging To demonstrate the scalability of our approach and support the claim that our solver outperforms the baselines in highly interactive settings, we also test our method on a ramp merging scenario with varying numbers of players. This experiment is inspired by the setup used in [82] and is schematically visualized in Figure 3.1. We model each player's dynamics by a discrete-time kinematic bicycle with the state comprising position, velocity and orientation, i.e., $x_t^i = (p_{x,t}^i, p_{y,t}^i, v_t^i, \psi_t^i)$, and controls comprising acceleration and steering angle, i.e., $u_t^i = (a_t^i, \phi)$. We capture their individual behavior by a cost function that penalizes deviation from a reference travel velocity and target lane; i.e., $\theta^i = (v_{\text{ref}}^i, p_{\text{y,lane}}^i)$. We add constraints for lane boundaries, for limits on speed, steering, and acceleration, for the traffic light, and for collision avoidance. To encourage rich interaction in simulation, we sample each agent's initial state by sampling their speed and longitudinal positions uniformly at random from the intervals from zero to maximum velocity v_{max} and four times the vehicle length l_{car}, respectively. The ego-agent always starts on the ramp and all agents are initially aligned with their current lane. Finally, we sample each opponent's intent from the uniform distribution over the two lane centers and the target speed interval $[0.4v_{\text{max}}, v_{\text{max}}]$. Partial observations comprise the position and orientation of each agent.

Baselines

We consider the following three baselines.

KKT-Constrained Solver In contrast to our method, the solver by Peters et al. [79] has no support for either private or shared inequality constraints. Consequently, this baseline can be viewed as solving a simplified version of the problem in (3.3) where the inequality constraints associated with the inner-level GNEP are dropped. Nonetheless, we still use a cubic penalty term as in (3.2) to encode soft collision avoidance. Furthermore, for fair comparison, we only use the baseline to *estimate* the objectives but compute control commands from a GNEP considering all constraints.

MPC with Constant-Velocity Predictions This baseline assumes that opponents move with constant velocity as observed at the latest time step. We use this baseline as a representative method for predictive planning approaches that do not explicitly model interaction.

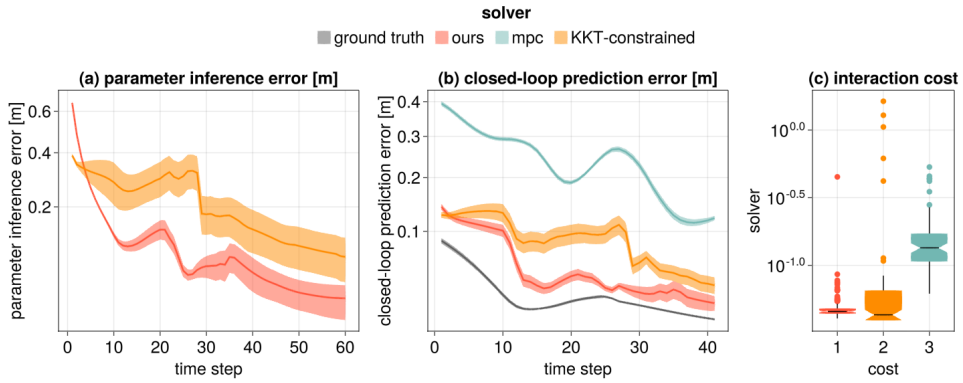


Figure 3.2: Monte Carlo study for the 2-player tracking game for 100 trials. Solid lines and ribbons in (a) and (b) indicate the mean and standard error of the mean. Cost distributions in (c) are normalized by subtracting ground truth costs.

Heuristic Estimation MPGP To highlight the importance of online intent inference, for the ramp merging evaluation, we also compare against a game-theoretic baseline that assumes a fixed intent for all opponents. This fixed intent is recovered by taking each agent's initial lane and velocity as a heuristic preference estimate.

To ensure a fair comparison, we use the same MCP backend [37] to solve all GNEPs and optimization problems with a default convergence tolerance of $1e^{-6}$. Furthermore, all planners utilize the same planning horizon and history buffer size of 10 time steps with a time-discretization of 0.1 s. For the iterative MLE solve procedure in the 2-player running example and the ramp merging scenario, we employ a learning rate of $2e^{-2}$ for objective parameters and $1e^{-3}$ for initial states. We terminate maximum likelihood estimation iteration when the norm of the parameter update step is smaller than $1e^{-4}$, or after a maximum of 30 steps. Finally, opponent behavior is generated by solving a separate ground-truth game whose parameters are hidden from the ego-agent.

3.5.2 Simulation Results

To compare the performance of our method to the baselines described in Section 3.5.1, we conduct a Monte Carlo study for the two scenarios described in Section 3.5.1.

2-Player Running Example

Figure 3.2 summarizes the results for the 2-player running example. For this evaluation, we filter out any runs for which a solver resulted in a collision. For our solver, the KKT-constrained baseline, and the MPC baseline this amounts to 2, 2 and 13 out of 100 episodes, respectively.

Figures 3.2(a-b) show the prediction error of the goal position and opponent's trajectory, each of which is measured by ℓ^2-norm. Since the MPC baseline does not explicitly reason about costs of others, we do not report parameter inference error for it

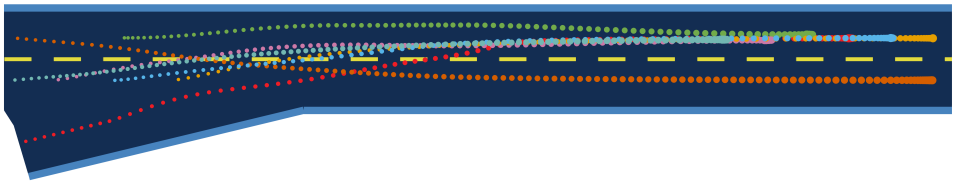
Set.	Method	Ego cost	Opp. cost	Coll.	Inf.	Traj. err. [m]	Param. err.	Time [s]
3 player	Ours	0.64 ± 0.36	0.06 ± 0.03	0	0	1.29 ± 0.05	0.41 ± 0.03	0.081 ± 0.002
	KKT-con	1.85 ± 1.21	0.05 ± 0.02	0	1	1.32 ± 0.06	2.39 ± 0.11	0.060 ± 0.002
	Heuristic	6.73 ± 2.40	0.09 ± 0.07	0	11	7.89 ± 0.26	3.96 ± 0.13	0.008 ± 0.001
	MPC	1.50 ± 0.45	0.33 ± 0.07	28	218	2.40 ± 0.11	n/a	0.009 ± 0.002
5 player	Ours	0.56 ± 0.43	0.16 ± 0.06	0	2	1.66 ± 0.07	0.47 ± 0.03	0.29 ± 0.02
	KKT-con	0.07 ± 0.32	0.06 ± 0.02	1	4	1.70 ± 0.06	2.15 ± 0.06	0.28 ± 0.02
	Heuristic	2.06 ± 0.44	0.35 ± 0.10	5	25	8.05 ± 0.19	2.91 ± 0.07	0.015 ± 0.001
	MPC	5.73 ± 2.91	0.42 ± 0.13	44	552	2.87 ± 0.13	n/a	0.014 ± 0.002
7 player	Ours	1.60 ± 1.19	0.06 ± 0.02	1	1	1.89 ± 0.05	0.46 ± 0.02	0.68 ± 0.02
	KKT-con	3.11 ± 1.72	0.09 ± 0.04	7	22	2.01 ± 0.06	1.93 ± 0.03	0.63 ± 0.06
	Heuristic	6.60 ± 1.67	0.27 ± 0.06	8	8	8.18 ± 0.15	2.44 ± 0.05	0.031 ± 0.002
	MPC	8.41 ± 1.45	0.59 ± 0.09	43	848	3.07 ± 0.08	n/a	0.0274 ± 0.004

Table 3.1: Monte Carlo study for the ramp merging scenario depicted in Figure 3.1 with 100 trials for settings with 3, 5, and 7 players. Except for collision and infeasible solve times, all metrics are reported by mean and standard error of the mean.

in Figure 3.2a. As evident from this visualization, both game-theoretic methods give relatively accurate parameter estimates and trajectory predictions. Among these methods, our solver converges more quickly and consistently yields a lower error. By contrast, MPC gives inferior prediction performance with reduced errors only in trivial cases, when the target robot is already at the goal. Figure 3.2c shows the distribution of costs incurred by the ego-agent for the same set of experiments. Again, game-theoretic methods yield better performance and our method outperforms the baselines with more consistent and robust behaviors, indicated by fewer outliers and lower variance in performance.

Ramp Merging

Table 3.1 summarizes the results for the simulated ramp-merging scenario for 3, 5, and 7 players.



(a) Qualitative performance.

Ego cost	Opp. cost	Coll.	Inf.	Traj. err. [m]	Param. err.	Time [2]
2.19 ± 1.21	0.17 ± 0.07	3	5	2.34 ± 0.08	0.91 ± 0.08	0.274 ± 0.01

(b) Quantitative performance.

Figure 3.3: Performance of our solver in combination with an NN for 100 trials of the 7-player ramp merging scenario.

Task Performance To quantify the task performance, we report costs as an indicator for interaction efficiency, the number of collisions as a measure of safety, number of infeasible solves as an indicator of robustness, and trajectory and parameter error as a measure of inference accuracy. On a high level, we observe that the game-theoretic methods generally outperform the other baselines; especially for the settings with higher traffic density. While MPC achieves high efficiency (ego-cost) in the 3-player case, it collides significantly more often than the other methods across all settings. Among the game-theoretic approaches, we observe that online inference of opponent intents—as performed by our method and the KKT-constrained baseline—yields better performance than a game that uses a heuristic estimate of the intents. Within the inference-based game solvers, a Manning-Whitney U-test reveals that, across all settings, both methods achieve an ego-cost that is significantly lower than all other baselines but not significantly higher than solving the game with ground truth opponent intents. Despite this tie in terms of interaction *efficiency*, we observe a statistically significant improvement of our method over the KKT-constrained baseline in terms of *safety*: in the highly interactive 7-player case, the KKT-constrained baseline collides seven times more often than our method. This advantage is enabled by our method's ability to model inequality constraints within the inverse game.

Computation Time We also measure the computation time of each approach. The inference-based game solvers have generally a higher runtime than the remaining methods due to the added complexity. Within the inference methods, our method is only marginally slower than the KKT-constrained baseline, despite solving a more complex problem that includes inequality constraints. The average number of MLE updates for our method was 11.0, 19.2, and 22.7 for the 3, 5, and 7-player setting, respectively. While our current implementation achieves real-time planning rates only for up to three players, we note that additional optimizations may further reduce the runtime of our approach.

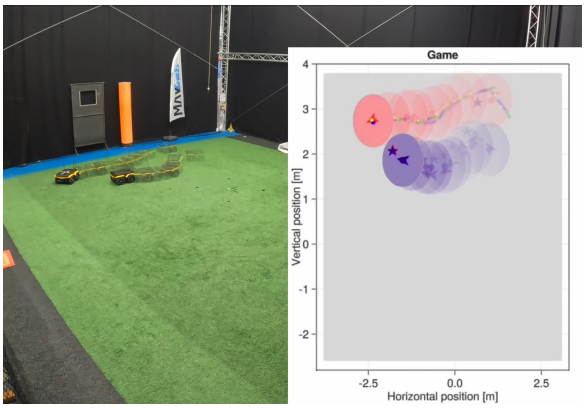
Among such optimizations are low-level changes such as sharing memory between MLE updates as well as algorithmic changes to perform intent inference asynchronously at an update rate lower than the control rate. We briefly explore another algorithmic optimization in the next section.

Combination with an NN

To support the claim that our method can be combined with other differentiable modules, we demonstrate the integration with an NN. For this proof of concept, we use a two-layer feed-forward NN, which takes the buffer of recent partial state observations as input and predicts other players' objectives. Training of this module is enabled by propagating the gradient of the observation likelihood loss of (3.11) through the differentiable game solver to the parameters of the NN. Online, we use the network's prediction as an initial guess to reduce the number of gradient steps. As summarized in Figure 3.3, this combination reduces the computation time by more than 60% while incurring only a marginal loss in performance.

3.5.3 Hardware Experiments

To support the claim that our method is sufficiently fast and robust for hardware deployment, we demonstrate the tracking game in the running example in Section 3.3.1 with a Jackal ground robot tracking (i) another Jackal robot (Figure 3.4a) and (ii) a human player (Figure 3.4b), each with initially unknown goals. Plans are computed online on a mobile i7 CPU. We generate plans using the point mass dynamics with a velocity constraint of $0.8\,\mathrm{m\,s^{-1}}$ and realize low-level control via the feedback controller of [100]. A video of these hardware demonstrations can be found at `https://xinjieliu.github.io/projects/game`. In both experiments, we observe that our adaptive MPGP planner enables the robot to infer the unknown goal position to track the target while avoiding collisions. The average computation time in both experiments was $0.035\,\mathrm{s}$.



(a) Interacting with another Jackal.

(b) Interacting with a human.

Figure 3.4: Time lapse of the running-example in which a Jackal tracks (a) another Jackal and (b) a human. Overlaid in (a) are the position of target robot (red) its true goal (red star), the tracker (blue), and its goal estimate (blue star).

3.6 Conclusion

In this work, we presented a model-predictive game solver that adapts strategic motion plans to initially unknown opponents' objectives. The adaptivity of our approach is enabled by a differentiable trajectory game solver whose gradient signal is used for MLE of unknown game parameters. As a result, our adaptive MPGP planner allows for safe and efficient interaction with other strategic agents without assuming prior knowledge of their objectives or observations of full states. We evaluated our method in two simulated interaction scenarios and demonstrated superior performance over a state-of-the-art game-theoretic planner and a non-interactive MPC baseline. Beyond that, we demonstrated the real-time planning capability and robustness of our approach in two hardware experiments.

In this work, we have limited inference to parameters that appear in the objectives of other players. Since the derivation of the gradient in Section 3.4.2 can also handle other parameterizations of F—so long as they are smooth—future work may extend this framework to infer additional parameters of constraints or aspects of the observation model. Furthermore, encouraged by the improved scalability when combining our method with learning modules such as NNs, we seek to extend this learning pipeline in the future. One such extension would be to operate directly on raw sensor data, such as images, to exploit additional visual cues for intent inference. Another extension is to move beyond MLE-based point estimates to inference of potentially multi-modal distributions over opponent intents, which may be achieved by embedding our differentiable method within a variational autoencoder. Finally, our framework could be tested on large-scale datasets of real autonomous-driving behavior.

Chapter 4

Contingency Games: Strategic Interaction Under Uncertainty

In Chapters 2 and 3, we introduced methods for estimating and adapting to players' objectives using an MLE formulation of inverse games. However, a key limitation of this MLE perspective is that it only considers a point estimate *of other players' objectives. When uncertainty about other players' objectives is high, deriving decisions from such a point estimate may be unreliable.*

In this chapter, we go beyond point estimates *and instead consider a* distribution *of possible players' objectives.*[1] *To facilitate this, we take inspiration from the concept of contingency planning, wherein an agent generates a set of possible plans conditioned on the outcome of an uncertain event. Building on this key idea, this chapter develops a game-theoretic variant of contingency planning, tailored to multi-agent scenarios in which a robot's actions impact the decisions of other agents and vice versa. Contingency games are parameterized via a scalar variable which represents a future time when intent uncertainty will be resolved. By estimating this parameter online, we construct a game-theoretic motion planner that adapts to changing beliefs while anticipating future certainty. We show that existing variants of game-theoretic planning under uncertainty are readily obtained as special cases of contingency games. Through a series of simulated autonomous driving scenarios, we demonstrate that contingency games close the gap between certainty-equivalent games that commit to a single hypothesis and non-contingent multi-hypothesis games that do not account for future uncertainty reduction.*

This chapter is a verbatim copy, with minor modifications, of the peer-reviewed journal article [101]:

> **Lasse Peters**, Andrea Bajcsy, Chih-Yuan Chiu, David Fridovich-Keil, Forrest Laine, Laura Ferranti, Javier Alonso-Mora. "Contingency Games for Multi-Agent Interaction." *IEEE Robotics and Automation Letters (RA-L)*, 2024.

Contribution statement: Lasse developed the mathematical framework for contingency games, implemented the proposed method, and performed all experiments. Andrea put forth the initial idea of taking a game-theoretic perspective on contingency planning. Lasse, Andrea, Chih-Yuan, David, and Forrest wrote the initial draft of the manuscript. All authors contributed to technical discussions and edits of the original manuscript submitted to RA-L [101].

[1] Such distributions of intents could come from [30], a work that is beyond the scope of this dissertation but is briefly discussed in Section 1.4.

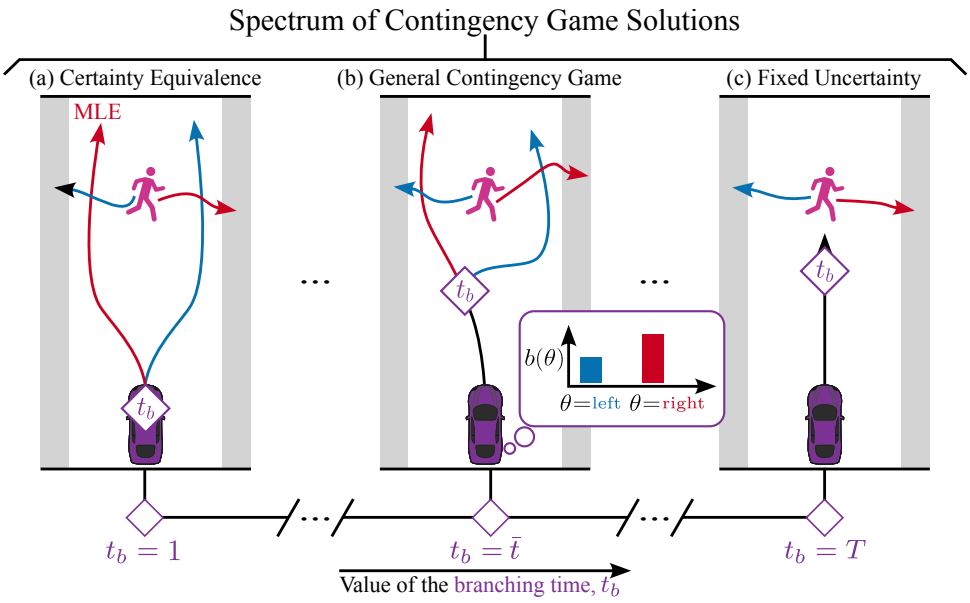


Figure 4.1: A vehicle approaching a jaywalking pedestrian with uncertain intent. (a) A risk-taking driver may gamble for the most likely outcome, ignore uncertainty, and pass on the left. (c) A risk-averse driver may hedge against all outcomes by bringing their vehicle to a full halt, waiting for the situation to resolve. (b) An experienced driver may realize that this uncertainty will resolve in the near future (at time t_b) and thus commit to an *immediate plan* that can be continued safely and efficiently under both outcomes. Our contingency games formalize this middle ground between the two extremes.

4.1 Introduction

Imagine you are driving and you see a pedestrian in the middle of the road as shown in Figure 4.1. The pedestrian is likely to continue walking to the right, but you also saw them turning their head around; so maybe they want to walk back to the left? You think to yourself, "If the pedestrian continues to the right, I just need to decelerate slightly and can safely pass on the left; but if they suddenly turn around, I need to brake and pass them on the right." Moreover, you understand that your actions influence the pedestrian's decision regarding whether and how quickly to cross the street. You decide to take your foot off the gas pedal and drive forward, aiming to pass the pedestrian on the left, but you are ready to brake and swerve to the right should the pedestrian turn around.

This example captures three important aspects of real-world multi-agent reasoning: (i) strategic interdependence of agents' actions due to their (partially) conflicting intents—e.g., the pedestrian's actions do not only depend on their own intent but also on your actions, and vice versa; (ii) accounting for uncertainty—e.g., how likely is it that the pedestrian wants to move left or right?; and (iii) planning contingencies—e.g., by anticipating that uncertainty will be resolved in the future, a driver can commit to an *immediate plan* (shown in black in Figure 4.1b) that can be continued safely and efficiently under each outcome (shown in red and blue in Figure 4.1b).

In this work, we formalize this kind of reasoning by introducing *contingency games*: a mathematical model for planning contingencies through the lens of dynamic game theory. Specifically, we focus on MPGP settings, wherein an ego agent (i.e., a robot) plans by choosing its future trajectory in strategic *equilibrium* with those of other nearby agents (e.g., humans) at each planning invocation. Importantly, the ego agent must account for any uncertainty about other agents' intents—i.e., their optimization objectives—when solving this game. For computational tractability, most established methods take one of two approaches. One class of methods ignores uncertainty by performing MLE and solving a certainty-equivalent game [78, 102, 103]. The other class accounts for uncertainty by planning with a full distribution—or "belief"—conservatively assuming that intent uncertainty will never be resolved during the planning horizon [63, 104]. Our **main contribution** is a game-theoretic interaction model that bridges the gap between these two extremes:

> A contingency game is a model for *strategic interactions* which allows a robot to consider the full *distribution* of other agents' intents while *anticipating intent certainty* in the near future.

Importantly, unlike existing formulations of trajectory games with parametric uncertainty [63, 104, 105], contingency games capture the fact that future belief updates will reduce uncertainty and hence, eventually, the *true* intent of the human will be clear at a future "branching" time, t_b. As a result, solutions of contingency games are *conditional* robot strategies which take a tree structure as shown in Figure 4.1b. The "trunk" of the conditional plan encodes decisions that are made before certainty is reached (before t_b). After t_b, the robot generates separate conditional trajectories for each possibility $\theta \in \Theta$.

Beyond our main contribution of an uncertainty-aware game-theoretic interaction model, we also (i) show how general-sum N-player contingency games can be transformed into Mixed Complementarity Problems, for which off-the-shelf solvers [37] are available and (ii) discuss how beliefs and branching times may be estimated online for receding-horizon operation. We also highlight the desirable modeling flexibility of contingency games as a function of the parameter t_b, recovering certainty-equivalent games on one extreme, and conservative solutions on the other (see Figure 4.1). Through a series of simulation experiments, we demonstrate that contingency games close the gap between these two extremes, and highlight the utility of estimating the branching time online.

4.2 Related Work

4.2.1 Game-Theoretic Motion Planning

Game-theoretic planning has become increasingly popular in interactive robotics domains like autonomous driving [102, 106, 107], drone racing [108], and shared control [109] due to its ability to model influence among agents. A crucial axis in which prior works differ is in the modeling of the robot's uncertainty regarding other agents' objectives, dynamics, and state at each time. Methods which assume no uncertainty in the trajectory game model (e.g., taking the most probable hypothesis as truth) result in the simplest game formulations, and have been explored extensively [46, 82, 103, 110]. However, in real-world settings it is unrealistic to assume that a robot has full certainty, especially with respect

to other agents' intents. Instead, robots will often maintain a probability distribution, or "belief," over uncertain aspects of the game.

There are several ways in which such a belief can be incorporated in a trajectory game. On one hand, the robot could simply optimize for its expected cost under the distribution of uncertain game elements. We call this a "fixed uncertainty" approach, since the game ignores the fact that as the game evolves, the robot could gain information leading to belief updates [63, 104, 105]. While these methods do utilize the robot's uncertainty, they often lead to overly conservative plans because the robot cannot reason about future information that would make it more certain (i.e., confident) in its decisions.

On the other hand, the agents in a game may reason about how their actions could lead to information gain; we refer to this as "dynamic uncertainty." Games which exactly model dynamic information gain are inherently more complex, and are generally intractable to solve, especially when the belief space is large and has non-trivial update dynamics [111, 112]. Recent methods attempted to alleviate the computational burden of an exact solution via linear-quadratic-Gaussian approximations of the dynamics, objectives, and beliefs [113]. While the computational benefits of such approximations are significant, they introduce artifacts that make them inappropriate for scenarios such as that shown in Figure 4.1 in which uncertainty is fundamentally multimodal. It remains an open challenge to solve "dynamic uncertainty" games tractably.

Our *contingency games* approach presents a middle ground between these paradigms via a belief-update model simple enough to compute exact solutions to the resulting dynamic-uncertainty game, and realistic enough to generate intelligent behavior that anticipates future uncertainty reduction.

4.2.2 Non-Game-Theoretic Contingency Planning

There is a growing literature of non-game-theoretic interaction planners [114], including various flavors of contingency planning. Online-optimization-based approaches primarily focus on predict-then-plan contingency planning [115–118]. Recent learning-based contingency planners leverage deep neural networks to generate human predictions conditioned on candidate robot plans [119, 120]; a robot plan is then selected via methods like sampling-based model-predictive control [119], dynamic programming [121], or neural trajectory decoders [120].

Other approaches draw inspiration from deep reinforcement learning to accomplish both intention prediction and motion planning. To predict multi-agent trajectories in the near future, Packer et al. [122] and Rhinehart et al. [91] construct a flow-based generative model and a likelihood-based generative model, respectively. Meanwhile, Rhinehart et al. [123] develop an end-to-end contingency planning framework for both intention prediction and motion planning. We bring the notion of contingency planning to a different modeling domain—dynamic game theory—to extend its capabilities to handle "dynamic uncertainty" in strategic interactions. This game-theoretic perspective captures interdependent behavior by modeling other agents as minimizing their own cost as a function of the decisions of all players in the scene.

4.3 Formalizing Contingency Games

In this work, we consider settings where a game-theoretic interaction model is not fully specified. For example, an autonomous car may not be sure if a nearby pedestrian intends to jaywalk; an assistive robot may not know which tool a surgeon will wish to use next. In such instances, the robot (agent R) can construct a dynamic game in which components of the model depend upon an unknown parameter $\theta \in \Theta$, with $|\Theta| = K < \infty$. When the robot has some prior information—e.g., from observations of past behavior—it can maintain a probability distribution or *belief*, $b(\theta)$, over the set of possible games. Naturally, however, this belief changes as a function of the human's behavior *and* the robots actions. It is this *dynamic* nature of the uncertainty that the robot can exploit to come up with more efficient plans. In the formal description below, we adopt a two-player convention for clarity. We note, however, that the formalism can incorporate more players as illustrated in Section 4.7.2.

Approximations and modeling assumptions. Contingency games approximate a game which exactly models dynamic uncertainty, which are generally intractable to solve. Specifically, we introduce the following key modeling assumptions to facilitate tractable online computation:

1. We assume unilateral uncertainty with discrete support, i.e. $|\Theta| = K$. That is, while the robot R has uncertainty in the form of a discrete probability mass function $b(\theta)$, the human H acts rationally under the true hypothesis $\hat{\theta}$.

2. We assume the robot has access to (an estimate of) the so-called branching time, t_b, which models the future time at which additional state observations will have resolved all uncertainty about the initially unknown θ.

3. We simplify the robot's belief dynamics. Instead of capturing the exact belief dynamics across the planning horizon, the robot distinguishes two phases: before t_b the belief is fixed at $b(\cdot)$; after t_b the belief collapses to certainty about a single hypothesis.

We envision contingency games to be employed in a receding-horizon (MPGP) fashion. In that context, beliefs are updated between each planner invocation and the branching time may vary and can be estimated online.[2] We discuss considerations and results for this case in Sections 4.6 and 4.8.

Notation conventions. We consider interaction of agents $i \in \{\mathrm{R}, \mathrm{H}\}$ over $T < \infty$ time steps. In contrast to existing game-theoretic formulations [45, 46, 67, 82], in a contingency game we endow each player with multiple trajectories; one for each hypothesis θ. At each $t \in [T] = \{1, 2, \ldots, T\}$ and hypothesis $\theta \in \Theta$ the *state* of the game is comprised of separate variables for each agent i, i.e., $x_{\theta,t} := (x^{\mathrm{R}}_{\theta,t}, x^{\mathrm{H}}_{\theta,t})$. Each agent i begins at a fixed initial state $\hat{x} := (\hat{x}^{\mathrm{R}}, \hat{x}^{\mathrm{H}})$, i.e., $x^i_{\theta,1} = \hat{x}^i$, which evolves over time as a function of that agent's control action, $u^i_{\theta,t}$. States and control actions are assumed to be real vectors of arbitrary, finite dimension. For brevity, we introduce the following shorthand: we use

[2]Note that, despite the use of assumption 3 within our game formulation, we will employ a belief updater that still captures more accurate belief dynamics as we shall discuss in Section 4.6.

$z^i_{\theta,t} := (x^i_{\theta,t}, u^i_{\theta,t})$ to denote a state-control tuple, we use boldface to denote aggregation over time, e.g. $\boldsymbol{z}^i_\theta := (z^i_\theta)_{t\in[T]}$, we omit player indices to denote aggregation, e.g. $\boldsymbol{z}_\theta = (\boldsymbol{z}^{\mathrm{R}}_\theta, \boldsymbol{z}^{\mathrm{H}}_\theta)$, and we denote the finite collection of all $K = |\Theta|$ trajectories for player i as $\boldsymbol{z}^i_\Theta = (\boldsymbol{x}^i_\Theta, \boldsymbol{u}^i_\Theta) := (\boldsymbol{z}^i_\theta)_{\theta\in\Theta}$.

Contingency game formulation. With these conventions in place, we formulate a contingency game as follows. The robot wishes to optimize its expected performance over all hypotheses (4.1a) while restricting all contingency plans to be feasible with respect to hypothesis-dependent constraints h_θ (4.1b) and enforcing the *contingency constraint* (4.1c) that the first $t_b - 1$ control inputs must be identical across all hypotheses $\theta \in \Theta$:

$$\mathrm{R}: \ \mathcal{S}^{\mathrm{R}}\left(\boldsymbol{z}^{\mathrm{H}}_\Theta\right) := \arg\min_{\boldsymbol{z}^{\mathrm{R}}_\Theta} \ \sum_{\theta\in\Theta} b(\theta) J^{\mathrm{R}}(\boldsymbol{z}^{\mathrm{R}}_\theta, \boldsymbol{z}^{\mathrm{H}}_\theta) \tag{4.1a}$$

$$h^{\mathrm{R}}_\theta(\boldsymbol{z}^{\mathrm{R}}_\theta, \boldsymbol{z}^{\mathrm{H}}_\theta) \geq 0, \ \forall\theta\in\Theta \tag{4.1b}$$

$$c(\boldsymbol{u}^{\mathrm{R}}_\Theta; t_b) = 0. \tag{4.1c}$$

Simultaneously, the robot interacts with K versions of agent H, each of which is guided by a different intent θ which parameterizes both the hypothesis-dependent cost J^{H}_θ (4.2a) and constraints h^{H}_θ (4.2b):

$$\forall\theta\in\Theta: \begin{cases} \mathrm{H}_\theta : \mathcal{S}^{\mathrm{H}_\theta}\left(\boldsymbol{z}^{\mathrm{R}}_\theta\right) := \arg\min_{\boldsymbol{z}^{\mathrm{H}}_\theta} J^{\mathrm{H}}_\theta(\boldsymbol{z}^{\mathrm{R}}_\theta, \boldsymbol{z}^{\mathrm{H}}_\theta) & (4.2a) \\ \qquad\qquad h^{\mathrm{H}}_\theta(\boldsymbol{z}^{\mathrm{R}}_\theta, \boldsymbol{z}^{\mathrm{H}}_\theta) \geq 0. & (4.2b) \end{cases}$$

In this contingency game formulation, it is important to appreciate that the robot's strategy depends on the distribution $b(\theta)$ and *all* the trajectories $\boldsymbol{z}_\Theta = (\boldsymbol{z}_\theta)_{\theta\in\Theta}$, while H_θ's strategy depends *only* on the trajectories under hypothesis θ, $\boldsymbol{z}_\theta$. Since the objectives of R and H may in general conflict with one another, and we model agents as being self-interested, such a game is termed *noncooperative*.

Taken together, optimization problems Equations (4.1) and (4.2) take the form of a GNEP. "Solutions" to this problem are defined as follows.

Definition 4.1 *(Generalized Nash Equilibrium, [85, Ch. 1]). A trajectory profile* $(\boldsymbol{z}^{\mathrm{R}*}_\Theta, \boldsymbol{z}^{\mathrm{H}*}_\Theta)$ *is a* GNE *of the game from Equations* (4.1) *and* (4.2) *if and only if*

$$\boldsymbol{z}^{\mathrm{R}*}_\Theta \in \mathcal{S}^{\mathrm{R}}\left(\boldsymbol{z}^{\mathrm{H}*}_\Theta\right) \quad \textit{and} \quad \bigwedge_{\theta\in\Theta} \boldsymbol{z}^{\mathrm{H}*}_\theta \in \mathcal{S}^{\mathrm{H}_\theta}\left(\boldsymbol{z}^{\mathrm{R}*}_\theta\right).$$

In practice, we relax Definition 4.1 and only require *local* minimizers of Equations (4.1) and (4.2). Under appropriate technical qualifications, such local equilibria are characterized by first and second order conditions commensurate with concepts in optimization. Readers are directed to [85] for further details.

A solution method for contingency games is discussed in Section 4.5; but first, we take a step back to build intuition about the behavior induced by the proposed interaction model.

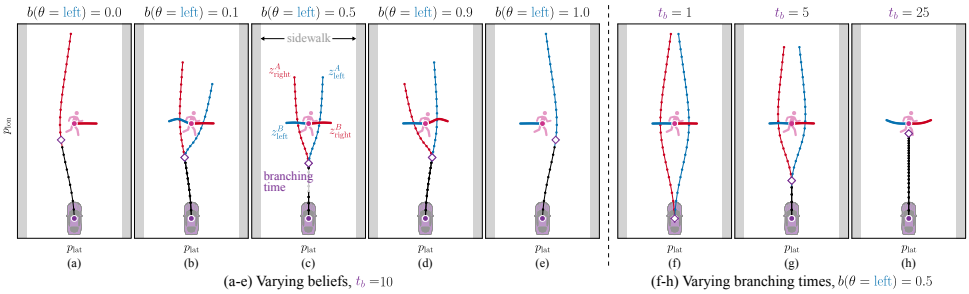


Figure 4.2: Contingency game solutions for the jaywalking scenario at varying beliefs and branching times.

4.4 Features of Contingency Game Solutions

4

Scenario 1: driving near a jaywalking pedestrian. Consider the scenario shown in Figure 4.2. Here, the robot (R) wants to move forward but does not know if the pedestrian (H) wants to reach the left or right side of the road. To model this ambiguity, let $\theta \in \Theta := \{\text{left}, \text{right}\}$. Robot R plans a trajectory for each hypothesis, i.e. $\boldsymbol{z}_\Theta^{\text{R}} := (\boldsymbol{z}_{\text{left}}^{\text{R}}, \boldsymbol{z}_{\text{right}}^{\text{R}})$, each of which is tailored to react appropriately to human H_θ in the corresponding scenario. Similarly, H maintains $\boldsymbol{z}_\Theta^{\text{H}} := (\boldsymbol{z}_{\text{left}}^{\text{H}}, \boldsymbol{z}_{\text{right}}^{\text{H}})$, which capture its "ground truth" behavior under each hypothesis. Each of H's trajectories will react to the corresponding trajectory of R. We model the robot as a kinematic unicycle and the pedestrian as a planar point mass. To ensure collision avoidance, the robot must pass behind the pedestrian, i.e. not between the pedestrian and its (initially unknown) goal position. We capture all of these constraints via a single vector-valued function $h_\theta^i(\boldsymbol{z}_\theta^{\text{R}}, \boldsymbol{z}_\theta^{\text{H}}) \geq 0$, which explicitly depends on hypothesis θ.

How each agent acts in a contingency game formulation of this problem is largely affected by two quantities: (i) the initial belief that the robot holds over the human's intent and (ii) the branching time which models the time at which the robot will get certainty. We discuss the role of both quantities below.

Qualitative behavior: Role of the belief. First, we keep the branching time fixed and analyze the contingency plans for a suite of beliefs. Figures 4.2a-e show how both the robot's contingency plan and the pedestrian's reaction change as a function of the robot's intent uncertainty. At extreme values of the belief, the robot is certain which hypothesis is accurate and the contingency strategy is equivalent to that of a single-hypothesis game with intent certainty. At intermediate beliefs, the contingency game balances hypotheses' cost according to their likelihood, yielding interesting behaviors: in Figure 4.2d the robot plans to inch forward at first (black), but biases its initial motion towards the scenario where the pedestrian will go left (blue). Nevertheless, it still generates a plan for the less likely event that the pedestrian goes right (red).

Qualitative behavior: Role of the branching time. Next, we assume that the robot's belief is always a uniform distribution, and vary the contingency game's t_b parameter. Figures 4.2f-h show how extreme values of this parameter automatically recover existing variants of game-theoretic planning under uncertainty: certainty-equivalent games [78, 102, 103] and non-contingent games that plan in expectation [104, 105]. At one extreme,

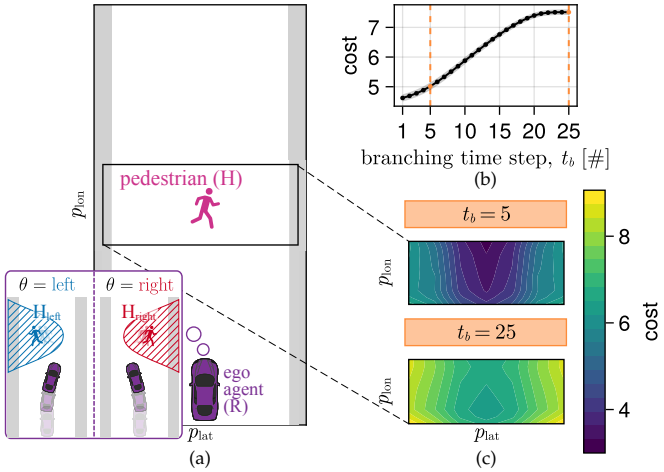


Figure 4.3: Cost of contingency plans in the jaywalking scenario. (a) state sampling region, (b) cost per t_b averaged over all initial states, (c) spatial cost distribution for two fixed t_b.

when $t_b = 1$, the contingency constraint (4.1c) is removed entirely and we obtain the solutions of the fully observed game under each hypothesis (see Figure 4.2f). These are precisely the solutions found for the certainty-equivalent games in Figures 4.2a,e. Note, however, that in this special case the contingency plan is not immediately actionable since it first requires the ego agent to commit to a single branch, e.g., by considering only the most likely hypothesis [78, 102, 103]. While easy to implement, such an approach can be overly optimistic and can lead to unsafe behavior since the control sequence from the selected branch may be infeasible for another intent hypothesis. For example, if the robot were to commit to θ = left in Figure 4.2f, it would be on a collision course with 50% probability.

By contrast, at the upper extreme of the branching time, $t_b = T$, the contingency plan no longer branches and it consists of a single control sequence for the entire planning horizon, c.f. [104, 105]. As a result, this plan is substantially more conservative since it must trade off the cost under *all* hypotheses while respecting the constraints for any hypothesis with non-zero probability mass: in Figure 4.2g, the ego agent plans to slow down aggressively since its uncertainty about the pedestrian's intent renders both sides of the roadway blocked.

Overall, this analysis shows that contingency games (i) unify various popular approaches for game-theoretic planning under uncertainty, and (ii) go beyond these existing formulations towards games that consider the *distribution* of other players' intents while *anticipating intent certainty* in the future.

Quantitative impact of the branching time. Given the central role of the branching time in our interaction model, we further analyze the quantitative impact of this parameter on the robot's contingency plan. For this purpose, we generate contingency plans across varying branching times, $t_b \in \{1, \dots, 25\}$, for each of 70 different initial pedestrian positions sampled from a uniform grid around the nominal position as shown in Figure 4.3a.

Figure 4.3b shows that, by anticipating future certainty at earlier branching times ($t_b = 5$), the robot discovers lower-cost plans than a method that assumes uncertainty will never resolve ($t_b = 25$). Furthermore, the spatial distribution of the cost in Figure 4.3c reveals that the robot generates particularly low-cost plans for $t_b = 5$ if the human is initially in the center of the road. Here, the contingency plan exploits the fact that—irrespective of the human's true intent—the road will be cleared by the time the robot arrives, c.f. Figure 4.2g. Of course, this analysis pertains to the *open-loop* plan. However, as we shall demonstrate in Section 4.8, a performance advantage persists under the added effect of receding-horizon planning.

4.5 Transforming Contingency Games into Mixed Complementarity Problems

Next, we discuss how to compute strategies from this interaction model. Rather than developing a specialized solver, we demonstrate how contingency games can be transformed into MCPs for which large-scale off-the-shelf solvers are readily available [37]. Our implementation of a game-theoretic contingency planner internally synthesizes such MCPs from user-provided descriptions of dynamics, costs, constraints, and beliefs.

We begin by deriving the KKT conditions for the contingency game in Equations (4.1) and (4.2). Under an appropriate constraint qualification (e.g., Abadie, or linear independence [85, Ch. 1], [58, Ch. 12]), these first-order conditions are jointly necessary for any generalized Nash equilibrium of the game. The Lagrangians of both players are:

$$\begin{aligned}\mathrm{R}: \mathcal{L}^{\mathrm{R}}(\boldsymbol{z}^{\mathrm{R}}_{\Theta}, \boldsymbol{z}^{\mathrm{H}}_{\Theta}, \lambda^{\mathrm{R}}_{\Theta}, \rho) &= \rho^{\top} c(\boldsymbol{u}^{\mathrm{R}}_{\Theta}; t_b) + \\ &\sum_{\theta\in\Theta} \Big(b(\theta) J^{\mathrm{R}}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}) - \lambda^{\mathrm{R}\top}_{\theta} h^{\mathrm{R}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}) \Big), \\ \mathrm{H}_{\theta}: \mathcal{L}^{\mathrm{H}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}, \lambda^{\mathrm{H}}_{\theta}) &= J^{\mathrm{H}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}) - \lambda^{\mathrm{H}\top}_{\theta} h^{\mathrm{H}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}),\end{aligned} \tag{4.3}$$

where ρ is the Lagrange multiplier for player R's contingency constraint, and λ^i_θ are Lagrange multipliers for all other constraints of player i at hypothesis θ. Denoting complementarity by "$\perp$", we derive the following KKT system for all players:

$$\forall \theta \in \Theta: \quad \begin{cases} \nabla_{\boldsymbol{z}^{\mathrm{R}}_{\theta}} \mathcal{L}^{\mathrm{R}}(\boldsymbol{z}^{\mathrm{R}}_{\Theta}, \boldsymbol{z}^{\mathrm{H}}_{\Theta}, \lambda^{\mathrm{R}}_{\Theta}, \rho) = 0, & \text{(4.4a)} \\ \nabla_{\boldsymbol{z}^{\mathrm{H}}_{\theta}} \mathcal{L}^{\mathrm{H}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}, \lambda^{\mathrm{H}}_{\theta}) = 0, & \text{(4.4b)} \\ 0 \le h^{\mathrm{R}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}) \perp \lambda^{\mathrm{R}}_{\theta} \ge 0, & \text{(4.4c)} \\ 0 \le h^{\mathrm{H}}_{\theta}(\boldsymbol{z}^{\mathrm{R}}_{\theta}, \boldsymbol{z}^{\mathrm{H}}_{\theta}) \perp \lambda^{\mathrm{H}}_{\theta} \ge 0, & \text{(4.4d)} \end{cases}$$

$$c(\boldsymbol{u}^{\mathrm{R}}_{\Theta}; t_b) = 0. \tag{4.4e}$$

Collectively, (4.4) forms an MCP, as defined below [85, Ch. 1].

Definition 4.2 (Mixed Complementarity Problem) *A Mixed Complementarity Problem (MCP) takes the following form: Given $G: \mathbb{R}^d \to \mathbb{R}^d$, lower bounds $v_{lo} \in [-\infty, \infty)^d$*

and upper bounds $v_{up} \in (-\infty, \infty]^d$, solve for $v^\star \in \mathbb{R}^d$ such that, for each $j \in \{1, \cdots, d\}$, one of the equations below holds:

$$\begin{aligned} [v]_j^\star = [v_{lo}]_j, [G]_j(v^\star) \geq 0, \\ [v_{lo}]_j < [v]_j^\star < [v_{up}]_j, [G]_j(v^\star) = 0, \\ [v]_j^\star = [v_{up}]_j, [G]_j(v^\star) \leq 0, \end{aligned}$$

where $[G]_j$ denotes the j^{th} component of G, and $[v_{lo}]_j$ and $[v_{up}]_j$ denote the j^{th} components of v_{lo} and v_{up}, respectively.

Observe that the KKT conditions (4.4) encode an MCP with variable v, function G, and bounds v_{lo}, v_{up} block-wise defined (by slight abuse of notation): one block for each $\theta \in \Theta$,

$$[v]_\theta = (\boldsymbol{z}_\theta^{\text{R}}, \boldsymbol{z}_\theta^{\text{H}}, \lambda_\theta^{\text{R}}, \lambda_\theta^{\text{H}}), \tag{4.5a}$$

$$[v_{\text{lo}}]_\theta = (-\infty, -\infty, 0, 0), \tag{4.5b}$$

$$[v_{\text{up}}]_\theta = (\infty, \infty, \infty, \infty), \tag{4.5c}$$

$$[G(v)]_\theta = \begin{bmatrix} \nabla_{\boldsymbol{z}_\theta^{\text{R}}} \mathcal{L}^{\text{R}}(\boldsymbol{z}_\Theta^{\text{R}}, \boldsymbol{z}_\Theta^{\text{H}}, \lambda_\Theta^{\text{R}}, \rho) \\ \nabla_{\boldsymbol{z}_\theta^{\text{H}}} \mathcal{L}^{\text{H}}(\boldsymbol{z}_\theta^{\text{R}}, \boldsymbol{z}_\theta^{\text{H}}, \lambda_\theta^{\text{H}}) \\ h_\theta^{\text{R}}(\boldsymbol{z}_\theta^{\text{R}}, \boldsymbol{z}_\theta^{\text{H}}) \\ h_\theta^{\text{H}}(\boldsymbol{z}_\theta^{\text{R}}, \boldsymbol{z}_\theta^{\text{H}}) \end{bmatrix}, \tag{4.5d}$$

and an additional block for the contingency constraint

$$[v]_c = \rho, [v_{\text{lo}}]_c = -\infty, [v_{\text{up}}]_c = \infty, [G(v)]_c = c(\boldsymbol{u}_\Theta^{\text{R}}; t_b). \tag{4.6}$$

To establish that the MCP solution is indeed a local equilibrium of the contingency game, sufficient second-order conditions [58, Thm. 12.6] can be checked for (4.1) and (4.2).

4.6 Online Planning with Contingency Games

We envision contingency games to be deployed in a MPGP framework where beliefs and branching times are estimated online. Next, we discuss considerations for this setting.

4.6.1 Belief Updates

While the true human intent $\hat{\theta}$ is hidden from the robot, the robot can utilize observations of past human decisions to update its current belief $b_\tau(\theta) = P(\theta \mid \hat{\boldsymbol{x}})$ about this quantity, where $\hat{\boldsymbol{x}} := \{(\hat{x}_1^{\text{R}}, \hat{x}_1^{\text{H}}), \dots (\hat{x}_\tau^{\text{R}}, \hat{x}_\tau^{\text{H}})\}$ is the sequence of joint human-robot states observed up until the current time τ. We use the model in Figure 4.4 to cast this inference problem in a Bayesian framework. In this model, we use our game-theoretic planner to compute jointly a *nominal* state-action trajectory for each agent and for each hypothesis. The robot's portion of this solution computed at time τ, $\boldsymbol{z}_{\Theta,\tau}^{\text{R}}$, serves as our receding-horizon motion plan, and the human's portion of the plan, $\boldsymbol{z}_{\theta,\tau}^{\text{H}}$, constitutes a nominal receding-horizon prediction of their future actions under each hypothesis $\theta \in \Theta$. However, due to bounded rationality [124], the human may not execute exactly this plan. Hence, at the

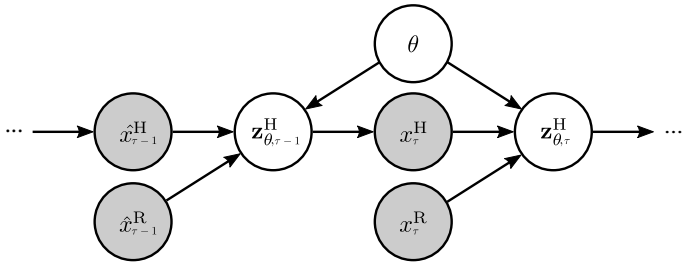


Figure 4.4: Bayesian network modeling the robot's intent inference problem. Shaded nodes represent observed variables.

next time step $\tau + 1$ when we observe a new human physical state, $\hat{x}^{\mathrm{H}}_{\tau+1}$, we treat it as a random emission of the *previous* nominal predictions. Similar to prior works [118, 120], in our experiments we assume that human states are distributed according to a Gaussian mixture model with one mode for each hypothesis θ; i.e., $p(\hat{x}^{\mathrm{H}}_{\tau+1} \mid \boldsymbol{z}^{\mathrm{H}}_{\theta,\tau}) = \mathcal{N}(\mu_\theta, \Sigma)$ where the mean μ_θ is the expected human state extracted from the *previous* human prediction, $\boldsymbol{z}^{\mathrm{H}}_{\theta,\tau}$, and Σ is a covariance parameter characterizing human rationality. In summary, the robot can recursively update their belief via

$$b_{\tau+1}(\theta) = \frac{p(\hat{x}^{\mathrm{H}}_{\tau+1} | \boldsymbol{z}^{\mathrm{H}}_{\theta,\tau}) b_\tau(\theta)}{\sum_{\theta' \in \Theta} p(\hat{x}^{\mathrm{H}}_{\tau+1} \mid \boldsymbol{z}^{\mathrm{H}}_{\theta',\tau}) b_\tau(\theta')}. \tag{4.7}$$

4.6.2 Estimating Branching Times

Prior work on non-game-theoretic contingency planning either chooses the branching time as informed by a careful heuristic design [115] or through offline analysis of the belief updater [125]. A thorough theoretical analysis on how to choose the branching time t_b in the more general game-theoretic case is beyond the scope of this work. Nonetheless, to demonstrate the utility of anticipating future intent certainty, we propose a branching time heuristic which only requires access to the previous game solution and current belief.

Heuristic branching time estimator. Intuitively, the branching time should be lower when the future human actions are more "distinct" under each hypothesis. Our heuristic captures this relationship as follows. Let $\mathcal{H}[b_\tau] = -\sum_{\theta \in \Theta} b_\tau(\theta) \log_{|\Theta|}(b_\tau(\theta))$ denote the entropy of belief b_τ. Furthermore, let $B(\boldsymbol{z}^{\mathrm{H}}_{\Theta,\tau-1}, \theta, k)[\cdot]$ denote the operator that, for a *given* θ, takes the first k states from the previously-computed $\boldsymbol{z}^{\mathrm{H}}_{\Theta,\tau-1}$ as hypothetical observations and returns the thus updated belief. We approximate the branching time as

$$t_b(b_\tau, \boldsymbol{z}^{\mathrm{H}}_{\Theta,\tau-1}) = \max_{\theta \in \Theta} \min_{k \in \{2,\dots,T\}} k \quad \text{s.t.} \quad \mathcal{H}\left[B(\boldsymbol{z}^{\mathrm{H}}_{\theta,\tau-1}, \theta, k)[b_\tau]\right] \leq \epsilon \tag{4.8}$$

Note that the minimum branching time chosen by this heuristic is $t_b = 2$, ensuring that the robot has a unique first input to apply during receding-horizon operation. Procedurally, this heuristic is straightforward to implement: for *each* hypothesis, we predict the belief via a hypothetical observation sequence as if the human (i) is perfectly rational and (ii) does not re-plan in the future; then, we return the first time at which *all* predicted

beliefs reach entropy threshold ϵ[3]. Assumptions (i) and (ii) make this heuristic cheap to evaluate since they avoid the need to re-compute game solutions within (4.8). While, these approximations may affect the accuracy of the estimator we shall demonstrate the utility of this approach in Section 4.8.

4.7 Experimental Setup

We wish to study the value of *anticipating* future certainty in game-theoretic planning. Therefore, we compare our method against a non-contingent game-theoretic baselines on two simulated interaction scenarios in which dynamic uncertainty naturally occurs.

4.7.1 Compared Methods

Beyond the contingency game that uses our branching time heuristic, we consider the following methods, all of which operate in receding-horizon fashion with online belief updates according to (4.7). Note that, following the discussion in Section 4.4, all game-theoretic methods below can be understood as a contingency game with a special branching time choice.

Baseline 1: Certainty-equivalent ($t_b = 1$). This baseline assumes certainty, making a point estimate by considering only the most probable hypothesis at each time step.

Baseline 2: Fixed uncertainty ($t_b = T$). Similar to [104], this baseline ignores future information gains, assuming fixed uncertainty along the entire planning horizon.

Baseline 3: MPC. This baseline uses non-game-theoretic model-predictive control (MPC), forecasting opponent trajectories assuming constant ground-truth velocity.

Contingency game with $t_b = 2$. To test the utility of our branching time heuristic, we also consider a contingency game that assumes certainty one step into the future—an assumption also used in non-game-theoretic contingency planning [117].

Contingency game with oracle branching time. Additionally, we consider an "oracle" branching time estimator that recovers the true branching time by first simulating receding-horizon interaction with a nominal branching time and then extracting the time of certainty from the belief evolution in *hindsight*. Naturally, this oracle requires access to the *true* human intent and hence is not realizable in practice. Nonetheless, we include this variant to demonstrate the potential performance achievable with our interaction model.

4.7.2 Driving Scenarios

Beyond the jaywalking example ("Scenario 1") introduced in Section 4.4, we evaluate our method on the following three-player scenario.

Scenario 2: highway overtaking. In this scenario, an autonomous vehicle attempts to overtake a human-operated vehicle with additional slow traffic in front, c.f. Figure 4.5. To perform this overtaking maneuver safely, the robot must reason about possible lane

[3]A low threshold results in more conservative behavior. As informed by a parameter sweep over ϵ, we choose $\epsilon = 2^{-2}$ for all experiments for best performance.

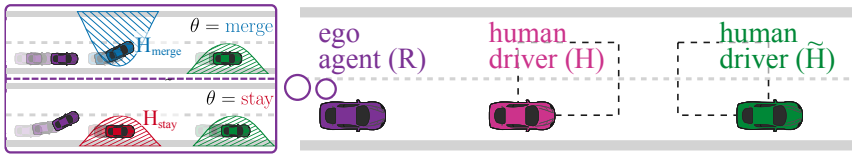


Figure 4.5: Scenario 2: an autonomous vehicle seeks to overtake slow traffic on a highway while being uncertain about the lane changing intentions of the vehicle ahead.

changing intentions of the other vehicle. Since the robot is uncertain about the target lane of human-driven car in front, it maintains a belief over the hypothesis space $\Theta = \{\text{merge}, \text{stay}\}$. We add a non-convex collision avoidance constraint between pairs of players which enforces that cars cannot be overtaken on the side of their target lane.

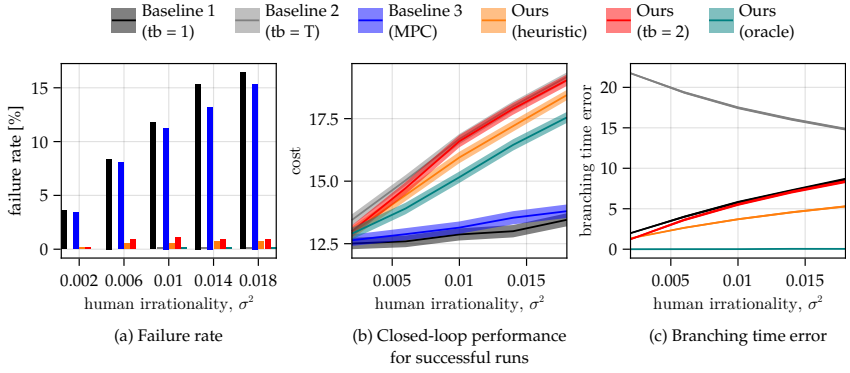
Implementation details. Throughout all experiments, we model cars as kinematic unicycles, and pedestrians as planar point masses. All systems evolve in discrete-time with time discretization of 0.2 s and agents plan over a horizon of $T = 25$ time steps. In both scenarios, road users' costs comprise of a quadratic control penalty and their intent $\theta \in \Theta$ dictates a goal position for pedestrians and a reference lane for cars via a quadratic state cost. Collision avoidance constraints are shared between all agents.

4.8 Simulated Interaction Results

The following evaluations are designed to support the claims that (**C1**) contingency games close the gap between the two extremes shown in Figure 4.1: providing more efficient plans than fixed-uncertainty games at higher levels of safety than certainty-equivalent games; and that (**C2**) our branching time heuristic improves the performance of contingency games over a naive fixed branching time estimate of $t_b = 2$.

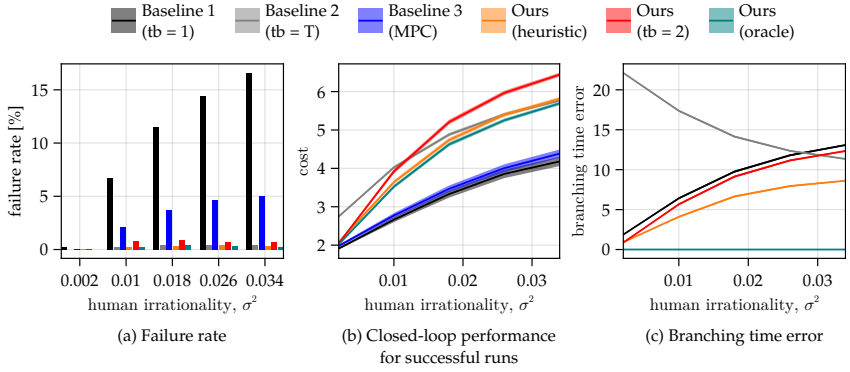
Data collection. We evaluate all methods in a large-scale Monte Carlo study as follows. For each scenario, we simulate receding-horizon interactions of 6 s duration. As in the open-loop evaluation of Section 4.4, we repeat the simulation for 70 initial states of each non-ego agent. These initial states are drawn from a uniform grid over the state regions shown in Figure 4.3a and 4.5. We generate the ground-truth human behavior from a game solution at each fixed hypothesis $\hat{\theta} \in \Theta$. Since this true human intent is initially unknown to the robot, the robot starts with a uniform belief. To test the methods under varying information gain dynamics—and thereby varying branching times—we consider five levels of human rationality, σ^2, parameterizing an isotropic observation model, i.e., $\Sigma = \sigma^2 \mathbf{I}$ where $\mathbf{I}$ denotes the identity matrix. In contrast to the open-loop evaluation of Section 4.4, here the robot re-plans at every time step with the latest online estimate of the belief and branching time. Figures 4.6 and 4.7 summarize the results of this Monte Carlo study.

Quantitative results. In terms of safety, Figures 4.6a and 4.7a show that the methods making a single prediction about the future, Baseline 1 and Baseline 3, fail significantly more often than the remaining approaches, all of which achieve failure rates below 1% across all levels of human rationality. In terms of efficiency, Figures 4.6b and 4.7b show that contingency games incur a lower interaction cost than the more conservative fixed-



(a) Failure rate (b) Closed-loop performance for successful runs (c) Branching time error

Figure 4.6: Quantitative closed-loop results for the jaywalking example.



(a) Failure rate (b) Closed-loop performance for successful runs (c) Branching time error

Figure 4.7: Quantitative closed-loop results for the overtaking example.

uncertainty Baseline 2. However, this performance advantage relies on a dynamic branching time estimate, as indicated by the performance advantage of Ours (heuristic) over Ours ($t_b = 2$). Finally, Ours (oracle) further improves efficiency due to the tight branching time estimate (c.f. Figures 4.6c and 4.7c), demonstrating the potential of our interaction model. In summary, these results support our claims **C1** and **C2** above.

Qualitative results. To further contextualize these results with respect to claim **C1**, we visualize examples of the closed-loop behavior generated by our method, the certainty-equivalent Baseline 1, and the fixed-uncertainty Baseline 2 in Figures 4.8 and 4.9. Here, we show both the sequence of states traced out by all players and the robot's plan five time steps into the interaction. In both scenarios, the robot's initial observations cause its belief to favor an incorrect hypothesis. This belief prompts Baseline 1 to commit to an unsafe strategy, causing a collision with the human. Baseline 2, on the other hand, brakes conservatively in the face of this belief and only accelerates once the uncertainty fully resolves. Finally, our contingency game planner *anticipates* the future information gain and avoids excessive braking while remaining safe.

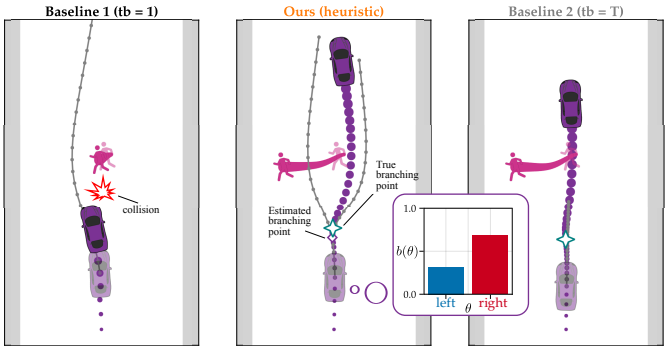


Figure 4.8: Qualitative closed-loop results for the jaywalking example.

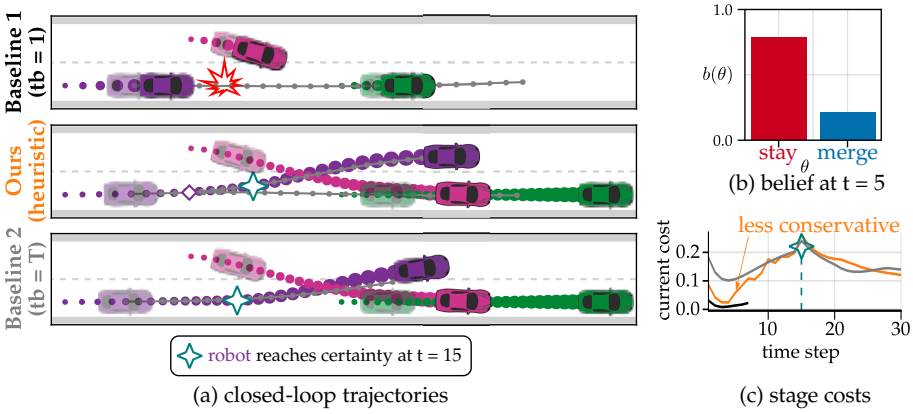


Figure 4.9: Qualitative closed-loop results for the overtaking example.

4.9 Limitations & Future Work

Even with a simple branching time heuristic, our approach outperforms the baselines. Nonetheless, the observed performance gains for the branching time "oracle" motivate further research into more precise estimators. Beyond that, in this work we assumed access to a fixed set of suitable intent hypotheses and our approach relies on receding-horizon re-planning to adapt to changes of this set. Future work should seek to automate the discovery of intent hypotheses, test our approach in scenarios with more complex intent dynamics, and consider extensions that explicitly capture these effects in the contingency game. Furthermore, the complexity of our approach is proportional to the product of the number of individual intents of each player, and the technique we employ to solve these games generally scales cubically with regards to total strategy size. Future work could consider employing a learning-based predictor [92] to automatically identify high-likelihood intents in complex scenarios and sub-select local players [126]. Finally, future work may extend our approach to continuous hypothesis spaces, e.g., by sampling as in [63, 127].

4.10 Conclusion

We present contingency games, a game-theoretic motion planning framework that enables a robot to efficiently interact with other agents in the face of uncertainty about their intents. By capturing simplified belief dynamics, our method allows a robot to anticipate future changes in its belief during strategic interactions. In detailed simulated driving experiments, we characterized both the qualitative behavior induced by contingency games and the quantitative performance gains with respect to non-contingent baselines.

Chapter 5

Amortized Equilibrium Approximation through Offline Learning

In Chapters 3 and 4, we have developed game-theoretic motion planners that adapt to changing beliefs of unknown game parameters online. Despite their demonstrated utility, these techniques share an important limitation: they require reasoning over large strategy spaces online, which can be computationally challenging.

The size of the strategy space is influenced by a variety of factors. In Chapter 3, the driving factor of the problem size is the number of players. In Chapter 4, the presence of multiple game-parameter-hypotheses further increases the problem size.

In this chapter, we tackle the computational challenges induced by large strategy spaces by developing a game solver that amortizes the computation of equilibrium solutions via an offline training phase. While the core principles of this chapter also apply to the games in Chapters 3 and 4, we develop our contribution in the context of another class of problems that naturally gives rise to enlarged strategy spaces: games in which agents are not forced to commit to a single, deterministic trajectory, but may choose a distribution of trajectories—so-called "mixed" strategies. We validate our approach on a number of experiments using the pursuit-evasion game "tag."

This chapter is a verbatim copy, with minor modifications, of the peer-reviewed conference publication [80]:

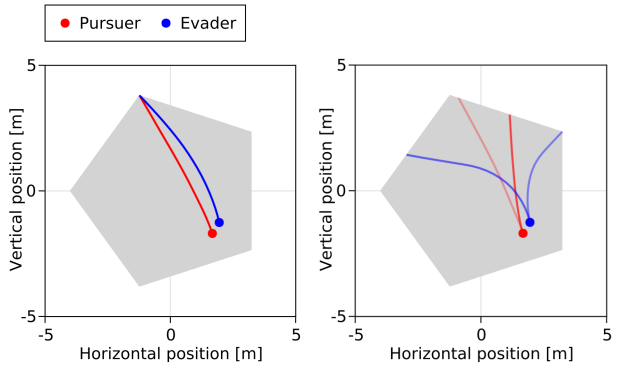
Lasse Peters, David Fridovich-Keil, Laura Ferranti, Cyrill Stachniss, Javier Alonso-Mora, Forrest Laine. "Learning Mixed Strategies in Trajectory Games." *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

Contribution statement: Lasse and Forrest developed the key ideas for the lifted game formulation and the offline training phase. Lasse implemented the lifted game solver, the underpinning trajectory optimization routines, the infrastructure for making all components differentiable, and conducted all experiments. Forrest implemented the underpinning finite game solver and proved the theoretical results in Appendix 5.A and 5.B. Lasse, David, and Forrest jointly wrote the initial draft of the manuscript. All authors contributed to technical discussions and edits of the original manuscript submitted to RSS [80].

5.1 Introduction

Trajectory optimization techniques have become increasingly common in motion planning. So long as vehicle dynamics, design objectives, and safety constraints satisfy mild regularity conditions, a motion planning problem may be encoded as a nonlinear program and solved efficiently to a locally-optimal solution. The widespread successes of trajectory optimization have sparked growing interest in similar techniques for multi-agent, noncooperative decision-making and motion planning. In this context, game theory offers an elegant mathematical framework for modeling the strategic interactions of rational agents with distinct interests. By reasoning about interactions with others as a *trajectory game*, an autonomous agent can plan future decisions while accounting for the strategic reactions of others.

Since they involve multiple players with distinct, potentially competing objectives, trajectory games can be far more complex to solve than single-agent trajectory optimization problems. Recent algorithmic advances make solving trajectory games tractable in some instances [46, 47]. Still, they remain fundamentally more challenging to solve than single-agent problems, and consequently, trajectory games have not been widely adopted in the robotics community.



(a) Pure strategies (b) Mixed strategies

Figure 5.1: A zero-sum game of tag played between two agents with planer point-mass dynamics in a pentagonal environment. (a) In pure strategies, players are bound to deterministic behavior, and the evader is quickly captured. (b) Our approach lifts the strategy space to learn more competitive, mixed strategies, i.e., distributions over multiple trajectory candidates per player. The opacity of each trajectory in (b) encodes probability of selecting that learned candidate.

Perhaps more importantly, however, equilibrium solutions to trajectory games do not always exist. Nonexistence arises even in extremely simple static games such as rock-paper-scissors, in which neither player wishes to commit to a fixed, deterministic action which could be exploited by its opponent. Unsurprisingly, the same phenomenon can arise in more complex trajectory games. For example, consider the game of tag shown in Figure 5.1, where the red pursuer wishes to catch the blue evader. Here, if the evader chooses a single, deterministic trajectory, it will certainly be caught by a rational pursuer. In the context of small, discrete games such as rock-paper-scissors, these non-existence issues

are commonly avoided by allowing players to "mix" their actions, i.e., to choose an action at random from a distribution of their choice. This distribution is called a "mixed strategy," in contrast to the choice of a single deterministic action or "pure strategy." However, for continuous trajectory games it can be difficult to represent mixed strategies. Hence, it is common to regularize players' objectives in order to encourage the existence of pure solutions.[1] For example, in Figure 5.1a each player is penalized for large accelerations, leading to an equilibrium in which the evader is cornered by the pursuer.

With these issues in mind, this work introduces the following key contributions:

1. a principled method for reducing the online computation needed to solve trajectory games via the introduction of an offline training phase, and
2. a formulation of *lifted* games over multiple trajectory candidates, which admit a natural class of high-performance mixed strategies.

Together, these contributions enable efficient and reliable on-line trajectory planning for autonomous agents in noncooperative settings, such as the tag example of Figure 5.1. We validate our methods in a suite of Monte Carlo studies, in which we demonstrate that lifting gives rise to mixed strategies as shown in Figure 5.1b, providing a significant competitive advantage in both open-loop and receding-horizon play. Our method's reliable convergence and its ability to explicitly account for constraints enables training from scratch within only a few minutes of simulated self-play. Once fully trained, learning can be disabled and our method generates mixed strategies within 2 ms for the tag example in Figure 5.1.

5.2 Related Work

Our contributions build upon recent work in trajectory optimization and game-theoretic planning, and bear a close relationship with work in learning motion primitives and implicit differentiation. We discuss these relationships in further detail below.

5.2.1 Trajectory Optimization

Trajectory optimization refers to a finite-horizon optimal control problem in which a robot seeks a sequence of control inputs which minimize a performance criterion [129]. It is common to use trajectory optimization for model predictive control (MPC), whereby a robot quickly re-optimizes a new sequence of control inputs as new sensor data becomes available [130]. While a host of trajectory optimization techniques have been proposed in recent years, most common algorithms build upon the iterative linear-quadratic regulator [131–133] and differential dynamic programming [134–136]. In turn, these may be understood as specific approximations to standard algorithms in nonlinear programming (NLP), such as sequential quadratic programming [58, 137]. As discussed below, this fundamental NLP representation underlies the proposed approach for multi-agent trajectory games.

[1] Regularizing players' control inputs to ensure the existence of equilibria is well-established in the literature on dynamic games and robust control [45], [128].

5.2.2 Trajectory Games

Recent work has sought to generalize the aforementioned trajectory optimization techniques to address multi-agent, competitive planning. Here, each player seeks to minimize an individual performance criterion subject to constraints arising from, e.g., dynamics and actuator limits. The objectives and constraints for different players may, in general, depend upon the trajectories of others. Solutions to these problems are characterized by equilibrium points at which all players' strategies are unilaterally optimal.

The theoretical underpinnings of *dynamic* games were established in the context of state feedback [44, 45, 138, 139]. However, computational methods were historically limited to highly-structured problems such as those found in linear robust control [128, 140]. Recent work on iterative linear-quadratic methods [46, 67, 104] extends these ideas to more general games such as those found in noncooperative robotic planning.

Closely related problems have also been studied in the context of *static* games. Here, equilibrium points are found by treating the trajectory of each player as a single action, and assuming the players choose these actions simultaneously [45]. This results in a Generalized Nash Equilibrium Problem (GNEP), for which general-purpose solution methods exist [37, 84, 141]. Several domain-specific solvers have been developed to exploit the structure of trajectory games, ranging from augmented Lagrangian [66] to iterated best response methods [65, 142]. Still, these methods can have a high computational burden in challenging settings.

Regardless of the equilibrium definition (dynamic or static), solving trajectory games is fundamentally harder than solving single-agent trajectory optimization problems, if for nothing else but the increased problem dimension. The number of decision variables scales linearly with the number of players involved, and even with proper handling of sparsity, computation generally scales cubically with the number of players [46]. In Section 5.4.1, we introduce an offline training phase for trajectory games which effectively reduces the online computational burden to that of solving a trajectory optimization problem for each player in parallel.

5.2.3 Motion Primitives

In this work, we introduce a *trajectory lifting* technique, which may be understood in the context of motion primitives [143]. As we discuss in Section 5.4, this reformulation endows each player with a distribution over finitely many trajectory candidates, which may be learned. However, learning trajectories, or motion primitives, is also meaningful in the context of a single agent, and recent work has proposed this concept in the contexts of quadcopter navigation [144] and robot manipulation [145]. In this light, the present work may be viewed as a multi-agent generalization of these techniques. Additionally, our work constitutes an adaptive, learning-enabled generalization of the multi-agent motion primitive games formulated for autonomous racing in [83]. In Section 5.4.2, we show that these trajectory primitives may be learned efficiently with first-order optimization.

5.2.4 Differentiable Optimization

To improve learning efficiency, we employ implicit differentiation to propagate derivative information through all steps of our proposed trajectory lifting approach. Recent works in end-to-end neural architectures for autonomous driving have developed specialized network layers that embed optimization problems [96, 97, 146]. Like these methods, we obtain derivatives of players' game values with respect to learnable parameters by implicitly differentiating through the first order optimality conditions for all players in a lifted trajectory game.

5.3 Formulation

We develop our approach in the context of games played between two[2] agents over time, in which each agent's motion is characterized by a smooth discrete-time dynamical system. That is, we model agent i's motion as the temporal evolution of its state $\bar{x}_i(t) \in \mathbb{R}^n$ and control input $\bar{u}_i(t) \in \mathbb{R}^m$ over discrete time-steps $t \in \{1, \dots, T\}$ with $\bar{x}_i(t+1) = F\big(\bar{x}_i(t), \bar{u}_i(t)\big)$ for differentiable vector field $F(\cdot, \cdot)$.

Taking an egocentric approach, we investigate using *model-predictive game play* (MPGP) [46, 65, 66] as a method by which each player can plan strategically while accounting for the predicted reactions of its opponent. MPGP constitutes a natural analogue to MPC [130] for *noncooperative*, multi-agent settings. That is, at regular intervals the 'ego' agent formulates a finite-horizon trajectory game between itself and its opponent. The equilibrium of this game specifies optimal trajectories for both players; the ego agent begins to execute its equilibrium trajectory, and the procedure repeats after a short time once the players have moved.

The finite-horizon trajectory games formulated at each planning interval can be modeled as a pair of coupled optimization problems, as is common in the literature [45, 67]:

$$\begin{aligned} \mathrm{OPT}_1(\tau_2, x_1) \;:=\; & \underset{\tau_1}{\arg\min} \quad f_1(\tau_1, \tau_2) \\ & \text{s.t.} \quad \tau_1 \in \mathcal{K}_1(x_1) \end{aligned} \tag{5.1a}$$

$$\begin{aligned} \mathrm{OPT}_2(\tau_1, x_2) \;:=\; & \underset{\tau_2}{\arg\min} \quad f_2(\tau_1, \tau_2) \\ & \text{s.t.} \quad \tau_2 \in \mathcal{K}_2(x_2) \end{aligned} \tag{5.1b}$$

The decision variables τ_i for each player $i \in \{1, 2\}$ represent discrete-time state-control trajectories starting from initial configuration x_i. Therefore, the constraint sets $\mathcal{K}_i(x_i)$ represent the set of all trajectories satisfying dynamic constraints, control limits, etc. Note that these sets need not be compact or convex, and that the players' constraint sets are independent of one another's trajectory. In contrast, the differentiable cost functions $f_i(\tau_1, \tau_2)$ in each problem can depend upon *both* players' trajectories. Thus, the f_i can encode preferences such as goal-reaching and collision-avoidance. In particular, since constraints are

[2] Although we limit our discussion to two players, our formulation may be extended to the general case. For further discussion, refer to Section 5.4.3.

decoupled, we assume that any aspect of interaction in the game is modeled via the cost functions and not through constraints.

As discussed in Section 5.2.2, existing methods to find local equilibrium solutions of (5.1) include iterative best response [65, 65] and iterative linear-quadratic methods [46, 47, 66, 67]. A Nash equilibrium for Game (5.1) starting from initial configuration (x_1, x_2) is defined to be a pair of trajectories, (τ_1^*, τ_2^*), satisfying

$$\tau_1^* \in \text{OPT}_1(\tau_2^*, x_1) \text{ and } \tau_2^* \in \text{OPT}_2(\tau_1^*, x_2). \tag{5.2}$$

Nash equilibrium points encode rational strategic play for both players, and hence serve as a natural solution concept in trajectory games (5.1). For this reason, most recent MPGP methods [46, 47, 65–67] aim to compute a Nash equilibrium of the trajectory game (5.1). As a practical matter, however, Nash equilibria can be intractable to compute and modern methods often settle for *local* equilibria, in which players' trajectories are only locally optimal.

Several important issues arise when employing an MPGP approach. The first is that solving for a Nash equilibrium—even a local Nash—is harder than solving for a locally optimal trajectory (as would be done in the single-agent setting of MPC). Not only is the search space larger due to the inclusion of both players' trajectory variables, but potential complications are also introduced by agents' different and potentially conflicting objectives. As in MPC, real-world applications depend upon our ability to compute solutions to (5.1) quickly; unfortunately though, this increased complexity can make MPGP unsuitable for real-time applications.

The second issue is that a Nash equilibrium point may not even exist for Game (5.1), particularly when one or both of the subproblems (5.1a) and (5.1b) are non-convex. Relatedly, even if a Nash equilibrium does exist, it may not be unique. Consequently, in MPGP an agent may spend significant computational effort searching for an equilibrium point that does not exist. Worse, non-uniqueness implies that even if an agent finds an equilibrium, the opponent's predicted Nash trajectory may not be representative of its true strategy.

To make these issues more concrete, consider the following "toy" variant of the tag game in Figure 5.1. Let τ_1 and τ_2 be scalars, $f_1(\tau_1, \tau_2) = \|\tau_1 - \tau_2\|_2^2 = -f_2(\tau_1, \tau_2)$, and $\mathcal{K}_1 = \mathcal{K}_2 = [-1, 1]$. Here, the pursuer (Player 1) and evader (Player 2) choose positions in the interval $[-1, 1]$. By inspection, we may verify that no Nash equilibrium exists. With additional regularization, however, this example can be modified to admit *local* equilibria. With f_1 defined as above, if we redefine the function $f_2(\tau_1, \tau_2) = -\|\tau_1 - \tau_2\|_2^2 - \|\tau_2\|_2^2$, two local equilibrium points result: $(-1, -1)$ and $(1, 1)$. Unfortunately, the locality of these equilibria causes a significant problem: if Player 1 computed one of these equilibria, and Player 2 computed the other, the resulting pairing of actions, e.g. $(-1, 1)$, would have a significantly different outcome for the players than what occurs at either local equilibrium.

5.4 Approach

We propose a novel *lifted* trajectory game formulation which ameliorates the complexity and existence/uniqueness issues discussed in Section 5.3.

5.4.1 Reducing Run-time Computation

To begin, we propose a technique for offloading the complexity introduced by multi-agent interactions to an offline training phase. The result of this pre-training is that at run-time, only a single-agent trajectory optimization problem remains for each player, and these problems can be solved in parallel. To do so, we introduce auxiliary trajectory references ξ_i for each Player i, and with a slight abuse of notation, reformulate Game (5.1) as:

$$\text{OPT}_1(\xi_2, x_1, x_2) \;:=\; \underset{\xi_1}{\arg\min} \quad f_1(\tau_1, \tau_2) \tag{5.3a}$$

$$\text{OPT}_2(\xi_1, x_1, x_2) \;:=\; \underset{\xi_2}{\arg\min} \quad f_2(\tau_1, \tau_2) \tag{5.3b}$$

Here, the decision variables ξ_i and the initial states x_i determine trajectory variables $\tau_i = \text{TRAJ}_i(\xi_i, x_i)$, which we presume to have the form:

$$\begin{aligned} \text{TRAJ}_i(\xi_i, x_i) \;:=\; \underset{\tau}{\arg\min}\; & \frac{1}{2}\|G_i\tau - \xi_i\|_2^2 + \frac{1}{2}\|H_i\tau\|_2^2 \\ \text{s.t.} \quad & \tau \in \mathcal{K}_i(x_i). \end{aligned} \tag{5.4}$$

The first term of the cost functions in problem (5.4) enables ξ_i to serve as a reference for τ_i. For example, if $\tau_i = [X_i^\mathsf{T}\; U_i^\mathsf{T}]^\mathsf{T}$, with X_i and U_i representing the state and control variables of the trajectory, then G_i could be $[0\; I]$, giving ξ_i the interpretation of a control reference signal. Alternatively, ξ_i could represent a reference for the terminal state of the trajectory. The second term allows regularization of the trajectory, which may be needed if the reference and constraint sets are otherwise insufficient to isolate solutions.

In Appendix 5.A, we prove that for any stationary point (τ_1, τ_2) of Game (5.1), there exists a stationary point (ξ_1, ξ_2) of Game (5.3) such that for both players i, $\tau_i = \text{TRAJ}_i(\xi_i, x_i)$. This implies that no stationary points are "lost" in the reformulation from (5.1) to (5.3). Furthermore, we discuss practical methods to guarantee that all computed stationary points of (5.3) result in stationary points of (5.1). This implies that no spurious stationary points are "introduced" in the reformulation.

With this reformulation, it is now possible to offload a significant amount of computation to an offline training phase. To do so, we propose training a *reference generator* for each player, denoted by the function $\pi_{\theta_i}(x_1, x_2)$, which maps both player's initial states (x_1, x_2) to reference ξ_i. Generator π_{θ_i} is parameterized by θ_i and, e.g., may be a multi-layer perceptron as described in Section 5.4.4. Given a data set[3] of initial MPGP configurations $D := \{x_1^k, x_2^k\}_{k=1}^d$, we train the reference generators (π_{θ_1} and π_{θ_2}) by solving the following game offline:

$$\text{GEN}_1(\theta_2, D) := \underset{\theta_1}{\arg\min}\; \frac{1}{d}\sum_{k=1}^{d} f_1(\tau_1^k, \tau_2^k), \tag{5.5a}$$

$$\text{GEN}_2(\theta_1, D) := \underset{\theta_2}{\arg\min}\; \frac{1}{d}\sum_{k=1}^{d} f_2(\tau_1^k, \tau_2^k). \tag{5.5b}$$

[3] D need not be constructed laboriously; in Section 5.5.5 we show that it can even be accumulated during online operation.

Similar to Game (5.3), each trajectory τ_i^k appearing in (5.5) is a function of θ_i, x_1^k and x_2^k, via the relationships

$$\begin{aligned} \tau_1^k &= \text{TRAJ}_1\Big(\pi_{\theta_1}(x_1^k, x_2^k), x_1^k\Big), \\ \tau_2^k &= \text{TRAJ}_2\Big(\pi_{\theta_2}(x_1^k, x_2^k), x_2^k\Big). \end{aligned} \tag{5.6}$$

A Nash equilibrium for Game (5.5) can be found by simultaneous gradient descent over each player's reference generator parameters, θ_1 and θ_2. Simultaneous gradient play is widely used in adversarial machine learning, and is particularly important in both generative adversarial networks [147] and multi-agent reinforcement learning [148]. Here, each player's parameter θ_i is iteratively updated as $\theta_i \leftarrow \theta_i - \delta\theta_i$, where

$$\begin{aligned} \delta\theta_1 &= \frac{\alpha_1}{d}\nabla_{\theta_1}\sum_{k=1}^{d} f_1\Big(\text{TRAJ}_1(\theta_1, x^k), \text{TRAJ}_2(\theta_2, x^k)\Big) \\ \delta\theta_2 &= \frac{\alpha_2}{d}\nabla_{\theta_2}\sum_{k=1}^{k} f_2\Big(\text{TRAJ}_1(\theta_1, x^k), \text{TRAJ}_2(\theta_2, x^k)\Big) \end{aligned} \tag{5.7}$$

Note that in (5.7), we use the shorthand $x^k \equiv (x_1^k, x_2^k)$, and although we abbreviate the arguments to the TRAJ functions, they should be interpreted exactly as in (5.6). The values α_1 and α_2 are learning rates used for the respective reference generators. To compute these gradients, we must differentiate through each player's objective f_i and through each TRAJ_i. We have assumed *a priori* that the f_i were differentiable. To differentiate through the trajectory optimization step of (5.4), we follow a procedure similar to what is outlined in [96, 97, 146].

Assuming that offline gradient play converges to a Nash equilibrium over the training set D, and that the resulting trajectory generators generalize to instances of (5.3) defined by configurations (x_1, x_2) not included in D, then an approximate equilibrium solution to Game (5.1), denoted by $(\hat{\tau}_1^*, \hat{\tau}_2^*)$ can be found via the following evaluations:

$$\begin{aligned} \xi_1 &= \pi_{\theta_1}(x_1, x_2), \qquad \xi_2 = \pi_{\theta_2}(x_1, x_2) \\ \hat{\tau}_1^* &= \text{TRAJ}_1(\xi_1, x_1), \quad \hat{\tau}_2^* = \text{TRAJ}_2(\xi_2, x_2) \end{aligned} \tag{5.8}$$

Hence, at run-time, solving this reformulated game only requires evaluating the reference generators and solving the optimization problems TRAJ_i to compute the corresponding trajectories. These problems can be solved in parallel. Furthermore, since trajectories are generated according to (5.4), each player's constraints defined by $\mathcal{K}_i(x_i)$ are guaranteed to be satisfied. Thus, if the reference generator does not generalize well, the only negative consequence is suboptimality (but not infeasibility).[4]

In summary, by pre-training a reference generator for each player offline, the run-time concerns of MPGP can be alleviated. Unfortunately, however, potential issues persist due to the possible non-existence or non-uniqueness of Nash equilibrium solutions. To address this concern, we introduce a concept we refer to as *strategy lifting*.

[4]Recall that each player's constraints do not depend upon the trajectory of the other player. Extension to this more complex case is possible, but beyond the scope of this work.

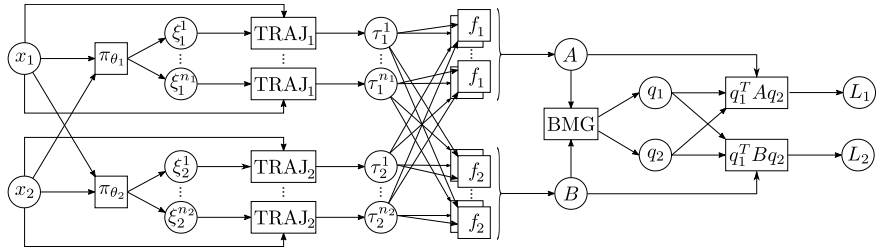


Figure 5.2: Overview of our proposed lifted game solver using reference generators. Generators π_{θ_i} for both players are trained jointly to minimize their respective average losses L_i (5.10a). At run-time deployment of this pipeline, the pre-trained generators produce references ξ_i which approximate a Nash equilibrium of (5.9). These references evaluate to equilibrium motion plan candidates τ_i and mixing strategies q_i for each player. When used in an ego-centric MPGP fashion, e.g. for Player 1, (τ_1, q_1) serves as a distribution over ego motion plans, and (τ_2, q_2) constitutes a probabilistic opponent prediction.

5.4.2 Lifted Trajectory Games

Rather than endowing each player with a single reference and its resulting trajectory, we allow each player to choose among multiple independent references according to the equilibrium solution of the bimatrix game formulated below:

$$\text{OPT}_1^{\text{lifted}}(\xi_2, x_1, x_2) := \arg\min_{\xi_1} L_1(\xi_1, \xi_2) \tag{5.9a}$$

$$\text{OPT}_2^{\text{lifted}}(\xi_1, x_1, x_2) := \arg\min_{\xi_2} L_2(\xi_1, \xi_2) \tag{5.9b}$$

where the dependence of L_1 and L_2 on ξ_1 and ξ_2 is made explicit through the following relationships:

$$L_1 = q_1^\top A q_2, \qquad L_2 = q_1^\top B q_2 \tag{5.10a}$$

$$A_{i,j} = f_1(\tau_1^i, \tau_2^j), \qquad B_{i,j} = f_2(\tau_1^i, \tau_2^j) \tag{5.10b}$$

$$\tau_1^i = \text{TRAJ}_1(\xi_1^i, x_1),\ i \in N_1 \tag{5.10c}$$

$$\tau_2^j = \text{TRAJ}_2(\xi_2^j, x_2),\ j \in N_2 \tag{5.10d}$$

$$(q_1, q_2) = \text{BMG}(A, B). \tag{5.10e}$$

Here, $N_1 := \{1, ..., n_1\}$, and $N_2 := \{1, ..., n_2\}$, where n_1 and n_2 are the number of trajectories for Player 1 and Player 2, respectively. Specifically, each variable τ_1^i represents one of n_1 trajectories that Player 1 optimizes over (and similar for Player 2). The reference variables $\xi_i := (\xi_i^1, \ldots, \xi_i^{n_i})$ are now collections of trajectory references, with each ξ_i^j associated to τ_i^j. The function $\text{BMG}(A, B)$ maps cost matrices A and B to mixed equilibrium strategies for the resultant bimatrix game. Specifically, $\text{BMG}(A, B)$ returns a point $(q_1^*, q_2^*) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ such that

$$\begin{aligned} (q_1^*)^\mathsf{T} A q_2^* &\leq q_1^\mathsf{T} A q_2^*, \qquad \forall q_1 \in \Delta^{n_1 - 1}, \\ (q_1^*)^\mathsf{T} B q_2^* &\leq (q_1^*)^\mathsf{T} B q_2, \quad \forall q_2 \in \Delta^{n_2 - 1}. \end{aligned} \tag{5.11}$$

In (5.11), Δ^k is the k-simplex, representing the space of valid parameters for a categorical distribution over $k+1$ elements. Note that when $n_1 = n_2 = 1$, Game (5.9) reduces exactly to Game (5.3), since $\text{BMG}(A, B) \equiv (1, 1)$.

Continuous games, such as (5.1), may suffer from non-existence of equilibrium points, but when those games are *separable*, a mixed strategy equilibrium is known to exist with finite support [149, 150]. This theoretical result motivates the lifting (5.9) of our reference-based formulation of Game (5.1).

For comparative purposes, Game (5.9) is presented analogously to (5.3), i.e. without any explicit dependence on reference generators. Nevertheless, generators can be trained analogously to (5.5), using a similar simultaneous gradient procedure. As before, solving (5.9) or an analogous version of (5.5) via gradient play requires that each of the function evaluations in (5.10) are differentiable in their arguments. It has already been discussed how each of these functions are differentiable, with the exception of the bimatrix game in (5.10e). We discuss in Appendix 5.B how this function is also differentiable.

A summary of the lifted game solver that utilizes reference generators for reduced online computation is provided in Figure 5.2. With this computation graph, the cost of approximating solutions to Game (5.9) is that of evaluating the two generator calls, solving the resultant $n_1 + n_2$ trajectory optimization problems (in parallel, if warranted), and solving a bimatrix game formed by considering all combinations of player trajectories.

5.4.3 Extension to Many-Player Games

We reiterate that, although we present this formulation in the two player setting, generalizations to larger games are straightforward. In this case, each player would consider multiple trajectory candidates, and a cost *tensor* would be created for each player, representing the costs for all possible combinations of players' trajectories. A finite Nash equilibrium could be identified over these cost tensors to compute the equilibrium mixing weights q_i [151], and computed by solving a nonlinear, mixed complementarity program [37, 152]. Note that the majority of computation required to construct these cost tensors can be trivially parallelized, making our framework particularly promising for many-player settings. We defer further study of such games to future work.

5.4.4 Implementation

We implement the lifted game solver depicted in Figure 5.2 in the Julia programming language [76]. For the experiments conducted in this work, reference generators π_{θ_i} are realized as multi-layer perceptrons, trajectory optimization problems TRAJ_i are solved via OSQP [41], and bimatrix games are solved using a custom implementation of the Lemke-Howson algorithm [153].

In order to facilitate back-propagation of gradients through this computation graph, we utilize the auto-differentiation tool Zygote [154]. For those components that cannot be efficiently differentiated automatically, namely TRAJ_i and BMG in Figure 5.2, we provide custom gradient rules via the implicit function theorem, c.f. [96, 97, 146] and Appendix 5.B. Our implementation can be found at `https://lasse-peters.net/pub/lifted-games`.

5.5 Results

We have presented a novel formulation of lifted trajectory games in which learned reference generators facilitate the efficient online computation of mixed strategies. In this section, we evaluate the performance of our proposed lifted game solver on variants of the "tag" game shown in Figure 5.1 and described below in Section 5.5.1. Concretely, we aim to quantify the utility of learning trajectory references rather than choosing them *a priori* (Section 5.5.2), characterize the equilibria identified by trajectory lifting (Section 5.5.3), evaluate the performance of trajectory lifting in head-to-head decentralized competition (Section 5.5.4), and demonstrate our method's capacity for online training in receding-horizon MPGP (Section 5.5.5).

5.5.1 Environment: The Tag Game

We validate our methods in a two-player tag game, illustrated in Figure 5.1. Here, each player's trajectory τ_i follows time-discretized planar double-integrator dynamics $\ddot{p}_i = u_i$, where $p_i \in \mathbb{R}^2$ is understood to represent horizontal and vertical position in the plane. The set $\mathcal{K}_i(x_i)$ then encompasses all dynamically-feasible trajectories that also satisfy input saturation limits and state constraints. In particular, we require that positions remain within a closed set, such as the pentagon illustrated in Figure 5.1, and that speeds remain below a fixed magnitude. These choices yield *linear* constraints, so that (5.4) becomes a quadratic program. We note, however, that our approach does not rely upon this convenient structure and is compatible with more general embedded nonlinear programs.

For the purposes of this example, we shall designate Player 1 to be the "pursuer" and Player 2 to be the "evader." Hence, the pursuer's objective $f_1(\tau_1, \tau_2)$ measures the average distance between players' trajectories over time and is regularized by the difference in control effort between the two players to ensure the existence of at least local pure Nash equilibria for the original game (5.1). The evader's objective is $f_2(\tau_1, \tau_2) = -f_1(\tau_1, \tau_2)$. Since the tag game has zero-sum cost structure, throughout the following evaluations we only report the cost for the pursuer and refer to this quantity as the *game value*. Furthermore, unless otherwise stated, we use an input reference signal ξ_i for all players in (5.3) and (5.9).

5.5.2 The Importance of Learning Trajectory Candidates

Without lifting, it is still possible to approximate mixed strategies for the trajectory game by discretizing the trajectory space (e.g., via sampling [83]). We compare to a sampling-based mixed-strategy baseline to study the isolated effects of learning in a lifted space.

Setup. We instantiate an evader with $n_2 = 20$ pre-sampled trajectory references. To strengthen the evader, we ensure that these samples cover a large region of the trajectory space. To that end, in this experiment (only) we use ξ_i as a reference for Player i's goal state rather than their input sequence. We compare the pursuer's performance for two different schemes of generating trajectory candidates. The non-learning baseline *samples* $n_1 \in \{1, \dots, 20\}$ pursuer trajectory references from the same distribution as the evader. Our method computes the pursuer strategy by performing gradient play on (5.9a) to *learn*

2 trajectory candidates via the goal reference parameterization. The mixed Nash equilibrium (q_1, q_2) over the players' trajectory candidates is computed according to (5.10). We evaluate both methods for 50 random initial conditions, and record the game value for each trial.

Discussion. Figure 5.3 summarizes the results of this experiment. As shown, the baseline steadily improves its performance with increasing numbers of sampled trajectory references to mix over. However, even with 20 trajectory samples, it cannot match the performance of our approach with only two learned candidates. Moreover, learning only a few trajectory references drastically reduces the number of trajectory optimizations and, consequently, the size of the bimatrix game in (5.10).

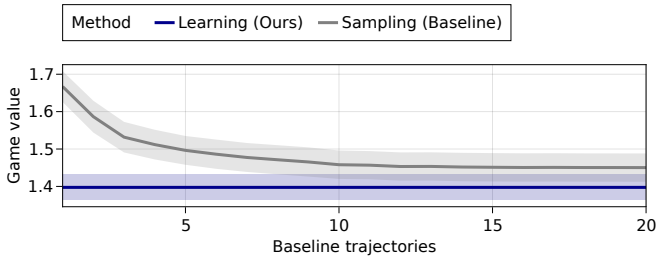


Figure 5.3: Comparison of game value for both sampled and learned pursuer trajectories. Lines trace the sample mean over 50 randomized trials, and the surrounding ribbons denote the SEM. On the horizontal axis, we vary the number of sampled *baseline* trajectories n_1, while fixing *our* approach to learn only two trajectories.

5.5.3 Convergence and Characteristics of Lifted Equilibria

In this experiment, we analyze *mixed* strategies found by our lifted solver and compare them to *pure* strategies computed by a non-lifting baseline. We shall demonstrate that both approaches reliably converge to different equilibria, and characterize these differences.

Setup. We perform a Monte Carlo study in which we randomly sample 20 initial states of the tag game. On each sample, we invoke two solvers which perform gradient play on different strategy spaces. The baseline solver is restricted to pure strategies as in Game (5.3).[5] Our method utilizes lifting to find mixed strategies which solve Game (5.9). In each iteration of gradient play, we record the game value.

Discussion. Figure 5.4 shows the reliable convergence of both methods in this Monte Carlo study. Since both players learn competitively via simultaneous gradient play, the game value ought not to evolve monotonically; an equilibrium is reached when neither player can improve its strategy unilaterally. At convergence, we observe that the mixed strategies found by our lifting procedure result in a higher game value. This higher value implies that, by operating in a lifted strategy space, the evader can secure a greater average distance between itself and the pursuer.

This gap in value may be understood intuitively by examining the strategy profiles for each method shown in Figure 5.1. In Figure 5.1a, players are restricted to pure strategies, and a rational pursuer can exploit the evader's deterministic choice of trajectory. In

[5]Such pure Nash solutions could also be found using iterated best response [65], iterative linear-quadratic methods [47], or mixed complementarity methods [37].

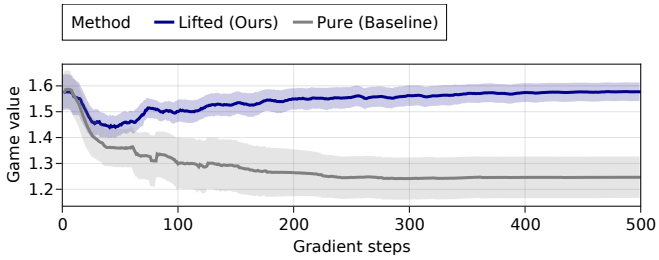


Figure 5.4: Equilibrium value convergence averaged over 20 initial states. Ribbons indicate the SEM. The baseline approximates Nash equilibria in *pure* strategies via gradient play on (5.3). Our method approximates Nash equilibria in *lifted* strategies via gradient play on (5.9).

contrast, our proposed lifting formulation allows the evader to mix over multiple trajectory candidates, c.f. Figure 5.1b, making its motion less predictable and hence increasing the chance of escaping the pursuer. In response, the pursuer also mixes between two trajectory candidates. However, each of the pursuer's candidates must account for the full distribution of evader trajectories; hence, the pursuer plans to turn less aggressively than the evader.

In this experiment, we have studied a *centralized* setting in which each method computes strategies for both players from a single game. Therefore, the results presented above are only suitable to characterize the solution points of (5.3) and (5.9), but do not justify conclusions about the competitive performance of these solutions in decentralized settings, such as MPGP. In the next section, we extend our analysis to settings in which the opponent's decision-making process is unknown.

5.5.4 Competitive Evaluation Against Non-Lifted Strategies

This experiment is designed to examine the performance of both pure (Baseline) and lifted (Ours) strategies in *decentralized* head-to-head competition. For this purpose, we perform two additional Monte Carlo studies which simulate tournaments among players in each strategy class.

Note that, in contrast to previous experiments, here, player strategies are not computed as the solution to a single, centralized game. Rather, each player is oblivious to their opponent's decision making process and solves its own version of the game from a known initial state over a finite time interval.

Open-Loop Competition

To begin, we evaluate both methods in open-loop on a fixed, 20-step time horizon.

Setup. For this Monte Carlo study, we randomly sample 100 initial states. For every sampled state, we invoke pure and lifted game solvers twice with randomly sampled initial strategies; once to obtain pursuer strategies, and once to obtain evader strategies.[6] For all

[6]This initialization procedure avoids leaking information about players' decision making processes to one another.

Table 5.1: Open-loop competition.

	Evader	
Pursuer	Lifted	Pure
Lifted	1.577 ± 0.021	1.502 ± 0.022
Pure	1.672 ± 0.022	1.370 ± 0.027

Table 5.2: Receding-horizon competition.

	Evader	
Pursuer	Lifted	Pure
Lifted	1.360 ± 0.003	1.289 ± 0.005
Pure	1.463 ± 0.004	0.903 ± 0.009

possible solver pairings on these 100 state samples we record the resultant value of the competing strategies; i.e., if Player i chooses trajectory τ_i, we record $f_1(\tau_1, \tau_2)$.

Discussion. Table 5.1 summarizes the mean and the standard error of the mean (SEM) of the resultant game value for this open-loop tournament. The evader has a clear incentive to utilize lifted strategies, since they secure the highest game value irrespective of the solution technique used by the pursuer. The best response of the pursuer is then also to play a lifted strategy to minimize value within this column. Hence, (Ours, Ours) is the unique Nash equilibrium in this meta game between solvers.

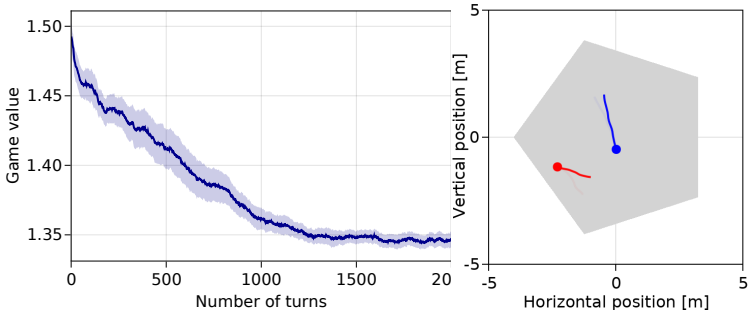
Additionally, observe that the baseline pursuer performs very well against the baseline evader, as deterministic evasion strategies can always be exploited by a rational pursuer. However, the tournament value reported in the bottom right of Table 5.1 is inconsistent with the equilibrium value for the baseline found earlier in Figure 5.4. This discrepancy suggests that players in this decentralized setup find different local solutions depending on the initialization of the baseline solver. Hence, random initialization effectively makes even a pure strategy evader slightly unpredictable, thereby allowing it to attain a higher average value. By contrast, the value of the lifted strategy computed by our method (top left, Table 5.1) closely agrees with the equilibrium value computed in Figure 5.4, which indicates that non-uniqueness of solutions is not an issue for our approach.[7]

Receding-Horizon Competition

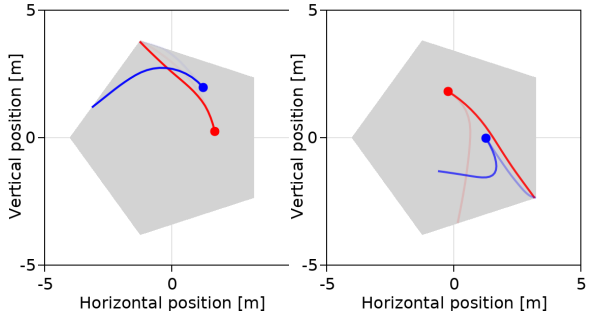
As MPGP is naturally applied in a receding-horizon fashion, we replicate the previous Monte Carlo study in that setting.

Setup. For each of 5 state samples, we simulate receding-horizon competitions for all possible solver pairings. As before, we use a planning horizon of 20 time steps for all players, and in order to simulate latency, we only allow players to update their plans every

[7] This close agreement in value suggests that our method identifies global (rather than local) Nash equilibria which satisfy the so-called *ordered interchangeability property* [45]. Unfortunately, as in continuous optimization, it is generally intractable to properly verify that these solutions are global.



(a) Closed-loop game value (b) Initial strategies



(c) Intermediate strategies (d) Final strategies

Figure 5.5: Training a reference generator for the lifted game solver in Figure 5.2 in simulated self-play. Transparency denotes the mixing probabilities associated with each trajectory which arise from BMG. (a) Game value mean and SEM over a moving window of 500 turns. (b-d) Strategies at different phases of learning.

9 time steps. Each simulation terminates once players have updated their strategy for the 500^{th} time. From one such trial, we compute the game value by evaluating the pursuer's objective f_1 on the entire *closed-loop* trajectories of both players. Note that, in contrast to previous experiments, here we use pre-trained reference generators for all solvers as described in Section 5.4.1 to accelerate the computation in this large simulation.

Discussion. Table 5.2 summarizes both the mean and SEM for the resultant game value in this receding-horizon Monte Carlo tournament. Overall, we observe the same patterns as in the open-loop setting: lifting is the dominant strategy for the evader and the corresponding best response for the pursuer. However, the game values found for this receding-horizon setting are generally lower than in open-loop. By replanning in a receding-horizon fashion, the pursuer can react to the evader's decision before the distance between them grows very large.

5.5.5 Learning in Receding-Horizon Self-Play

Finally, we demonstrate that a lifted game solver with trajectory generators, as shown in Figure 5.2, can be rapidly trained from scratch in simulated self-play.

Setup. We repeat the following experiment 10 times. For each player, we randomly sample an initial state x_i and initialize their reference generator π_{θ_i} with parameters θ_i sampled from a uniform distribution. Subsequently, we simulate receding-horizon learning over 2500 turns with the lifted game solver in the loop. In contrast to the setup used in Section 5.5.4, here, we do *not* use pre-trained reference generators. Instead, the network parameters are updated on the fly using gradient descent. That is, at every turn, we first perform a forward pass through the computation graph of Figure 5.2 to compute a lifted strategy profile, followed by a backwards pass to compute a gradient step on each player's reference generator parameters θ_i. For each experiment, we record players' strategies as well as the game value over a moving window of 500 turns.

Discussion. Figure 5.5 summarizes the results of lifted learning in self-play. Initially, the untrained reference generators cause both players to move haphazardly, c.f. Figure 5.5b. As learning progresses, players become more competitive, resulting in purposeful, dynamic maneuvers, c.f. Figure 5.5c. Within approximately 1500 turns, learning converges, the game value stabilizes, and the solver has learned to generate highly competitive mixed strategies as shown in Figure 5.5d.

Note that, throughout the learning procedure, state and input constraints are explicitly enforced in the TRAJ step of the pipeline in Figure 5.2. Moreover, since our proposed pipeline is end-to-end differentiable, it provides a strong learning signal. Therefore, training in simulated self-play over 2500 turns can be performed in less than three minutes on a standard laptop. Then, once the reference generators π_{θ_i} are fully trained, learning can be disabled, and a forward pass on the pipeline in Figure 5.2 can be computed with an average run-time of 2 ms. In summary, these results indicate that our method learns quickly and reliably, making it well-suited for online learning in real systems with embedded computational hardware.

5.6 Conclusion

In this work, we have proposed two key contributions to the field of noncooperative, multi-agent motion planning. First, we have introduced a principled technique to reduce the online computational complexity of solving these trajectory games. Second, we extended this approach to optimize over a richer, probabilistic class of *lifted* strategies for each player. Taken together, these innovations facilitate efficiently-computable, high-performance online trajectory planning for multiple autonomous agents in competitive settings. Moreover, our method directly accounts for problem constraints and hence guarantees that learned trajectories satisfy these constraints whenever they are feasible.

While our formulations readily extend to games with many players and arbitrary cost structure, we demonstrate our results in a two-player, zero-sum game of tag. We validate our approach in extensive Monte Carlo studies, in which we observe rapid and reliable convergence to solutions which outperform those which emerge in the original, non-lifted strategy space.

Finally, we showcase our approach in online learning, where each player solves lifted trajectory games in a receding time horizon. Despite the additional complexity present in this setting—e.g., non-stationary training data and potential limit cycles—our method converges reliably to competitive mixed strategies. These initial results are extremely encouraging, and future work should investigate online learning and adaptation in noncooperative settings more extensively. In particular, we note that our method is limited to so-called *open-loop* information structures, in which each agent in a trajectory game must choose future control inputs as a function only of the current state. We believe that the incorporation of feedback structures at the trajectory-level will be an exciting direction for future research.

Appendix for Chapter 5

Amortized Equilibrium Approximation through Offline Learning

5.A Equivalence of Game (5.1) and Game (5.3)

In this section we establish an equivalence result between (5.1) and (5.3). We prove this for a particular interpretation of the reference variables ξ_i, and forms of $G_i, H_i, \mathcal{K}_i(x_i)$, noting that similar results can be established for other settings.

For each Player i, consider the instance of (5.4) in which $G_i := \mathrm{I}$ and $H_i := 0$, representing the identity and zero matrices of appropriate dimension. Furthermore, assume that the set $\mathcal{K}_i(x_i) := \{\tau : \mathrm{lb}_i \leq g_i(\tau) \leq \mathrm{ub}_i\}$, for some vector-valued and twice-differentiable function g_i, and lower and upper bounds lb_i and ub_i. It is assumed that a suitable constraint qualification applies to this constraint set, such as the LICQ [58]. This implies that ξ_i has dimension equal to that of the decision variable τ, and the objective of (5.4) is to find a trajectory $\tau \in \mathcal{K}_i(x_i)$ which is as close as possible to ξ_i as measured by the ℓ_2-norm.

Theorem 5.1 *In the setting as stated above,*

1. *For any stationary point (τ_1, τ_2) of Game (5.1), there exists a stationary point (ξ_1, ξ_2) of Game (5.3) such that $\tau_i = \mathrm{TRAJ}_i(\xi_i, x_i)$ for all players i.*
2. *For any stationary point (ξ_1, ξ_2) of Game (5.3) satisfying $\xi_i \in \mathcal{K}_i(x_i)$, the trajectories $\tau_i = \mathrm{TRAJ}_i(\xi_i, x_i)$ constitute a stationary point for (5.1).*

Proof. To prove this result, we first make explicit the definition of a stationary point for (5.1) and (5.3). A stationary point for (5.1) is a point (τ_1, τ_2), such that for both players i,

$$d^\intercal \nabla_{\tau_i} f_i(\tau_1, \tau_2) \geq 0, \forall d \in T_{\mathcal{K}_i}(\tau_i). \tag{5.12}$$

Here, $T_{\mathcal{K}_i}(\tau_i)$ is the set of linearized feasible directions with respect to constraint set $\mathcal{K}_i(x_i)$ at τ_i, which because we have assumed a suitable constraint qualification, is equivalent to the tangent cone at this point [58]. Specifically, at a feasible point τ, let $\mathcal{I}_l(\tau) := \{j : g_{i,j}(\tau) = \mathrm{lb}_j\}$, and $\mathcal{I}_u(\tau) := \{j : g_{i,j}(\tau) = \mathrm{ub}_j\}$. Then

$$\begin{aligned} T_{\mathcal{K}_i}(\tau) := \{d : d^\intercal \nabla g_{i,j}(\tau) \geq 0, j \in \mathcal{I}_l(\tau), \\ d^\intercal \nabla g_{i,j}(\tau) \leq 0, j \in \mathcal{I}_u(\tau)\} \end{aligned} \tag{5.13}$$

A stationary point for (5.3) is a point (ξ_1, ξ_2) such that

$$(\nabla_{\xi_i} \tau_i \cdot d)^\intercal \nabla_{\tau_i} f_i(\tau_1, \tau_2) \geq 0, \forall d, \tag{5.14}$$

where $\tau_i = \mathrm{TRAJ}_i(\xi_i, x_i)$. Note that $(\nabla_{\xi_i} \tau_i \cdot d) := (\nabla_{\xi_i} \mathrm{TRAJ}_i(\xi_i, x_i) d)$ as appearing above is the directional derivative of $\mathrm{TRAJ}_i(\xi_i, x_i)$ with respect to changes of ξ in the direction d. This directional derivative is defined to be e, where e solves the following quadratic program [95]:

$$\begin{aligned} \min_{e} \quad & \frac{1}{2} e^\intercal Q_1 e + d^\intercal Q_2 e \\ \text{s.t.} \quad & e \in \mathcal{C}_{\lambda_i}(\tau_i), \end{aligned} \tag{5.15}$$

where $Q_1 := I - \nabla^2_{\tau,\tau}(g(\tau_i)^\mathsf{T}\lambda_i)$, $Q_2 := -I$, λ_i are the dual variables associated with the primal solution τ_i to $\text{TRAJ}_i(\xi_i, x_i)$, and $\mathcal{C}_{\lambda_i}(\tau)$ is the critical cone to the constraint set $g_i(\tau)$ with respect to λ_i at τ:

$$C_{\lambda_i}(\tau) := \{d : \bar{\text{lb}}_{i,j} \leq d^\mathsf{T}\nabla g_{i,j}(\tau) \leq \bar{\text{ub}}_{i,j})\}. \tag{5.16}$$

The bounds $\bar{\text{lb}}_{i,j}$ and $\bar{\text{ub}}_{i,j}$ are defined as:

$$(\bar{\text{lb}}_{i,j}, \bar{\text{ub}}_{i,j}) := \begin{cases} (0,0) & j \in \mathcal{I}_l(\tau) \;\&\; \lambda_{i,j} > 0 \\ (0,\infty) & j \in \mathcal{I}_l(\tau) \;\&\; \lambda_{i,j} = 0 \\ (-\infty,\infty) & i \notin (\mathcal{I}_l(\tau) \cup \mathcal{I}_u(\tau)) \\ (-\infty,0) & j \in \mathcal{I}_u(\tau) \;\&\; \lambda_{i,j} = 0 \\ (0,0) & j \in \mathcal{I}_u(\tau) \;\&\; \lambda_{i,j} < 0 \end{cases} \tag{5.17}$$

Now, to prove 1), we show that $\xi_i = \tau_i$ satisfies the claim. It follows directly that $\tau_i = \text{TRAJ}_i(\xi_i, x_i)$. It can be verified that, because $\tau_i \in \mathcal{K}_i(x_i)$ by definition, then all constraints appearing in TRAJ_i are only weakly active, implying $\lambda_i = 0$. This implies that the constraint set appearing in (5.15) is precisely the tangent cone (5.13). Therefore, for all directions d, $\nabla_{\xi_i}\text{TRAJ}_i(\tau_i, x_i) \cdot d \in T_{\mathcal{K}_i}(\tau_i)$, which by (5.12), implies that (5.14) holds, establishing the result.

To prove 2), we simply note that if $\xi_i \in \mathcal{K}_i(x_i)$, then $\tau_i = \xi_i$. Furthermore, in this setting $\lambda_i = 0$ as before, and therefore the critical cone appearing in (5.15) is again equivalent to the tangent cone (5.13). This implies that the directional derivative $(\nabla_{\xi_i}\tau_i \cdot d)$ is defined to simply be the projection of the direction d into the tangent cone at τ_i. The set of all directions d mapped through this projection results precisely in $T_{\mathcal{K}_i}(\tau_i)$. Therefore, the conditions (5.14) imply (5.12) for this setting, implying our result. □

The result as stated in Theorem 5.1 does not imply that an *arbitrary* stationary point found for (5.3) corresponds to a stationary point for (5.1), since it may be that either of the references $\xi_i \notin \mathcal{K}_i(x_i)$. For such reference points, it is possible that for some direction d the expression in (5.14) holds with equality, yet the expression in (5.12) is violated. This situation results in "sticky constraints," in which a descent direction exists for $f_i(\tau_1, \tau_2)$, yet that direction is not in the range of $\nabla_{\xi_i}\text{TRAJ}_i(\xi_i, x_i)$, i.e. small changes to the reference are not enough to release τ_i away from the active constraint boundaries.

To address this issue, we propose a modest regularization scheme to eliminate the possibility of reference stationary points of (5.3) which do not correspond to trajectory stationary points of (5.1). One such approach could be to enforce constraints in (5.3) such that $\xi_i \in \mathcal{K}_i(x_i)$. This, however, would render the reformulation from (5.1) to (5.3) pointless. Instead, we impose a simple regularization in the objectives of each player in (5.3). Namely, instead of minimizing over $f_i(\tau_1, \tau_2)$ w.r.t. ξ_i, we minimize over

$$f_i(\tau_1, \tau_2) + \|(g(\xi_i) - \text{ub})_+ + (\text{lb} - g(\xi_i))_+\|_2^2, \tag{5.18}$$

where $(\cdot)_+ := \max(\cdot, 0)$.

Note that this introduced regularization is exact, and has precisely the effect of eliminating any stationary points for (5.3) in which $\xi_i \notin \mathcal{K}_i(x_i)$. If the regularization term is non-zero, then necessarily from the definition of the directional derivative (5.15), the

gradient of the regularization component is in the null-space of $\nabla_{\xi_i}\text{TRAJ}_i(\xi_i, x_i)$. This implies the regularization can be driven to zero without changing the resultant solution τ_i. This is true irrespective of the scale factor multiplying the regularization term. Furthermore, if $\xi_i \in \mathcal{K}_i(x_i)$, then the regularization term is zero, and has no effect on stationary points of the un-regularized game (5.3).

We note that the particular choice of regularization (5.18) is only applicable for the interpretation of the references ξ_i made throughout this section. For more general parameterizations of the reference, as discussed in the main text, a suitable regularization is the norm of inequality constraint multipliers associated with the solution of $\text{TRAJ}_i(\xi_i, x_i)$. The use of this dual-variable regularization is effective at eliminating the spurious stationary points for (5.3), so long as the parameterization of the reference is rich enough such that for any ξ_i and associated τ_i, λ_i, there exists directions d in which the ξ_i can be perturbed and the directional derivative of τ_i is 0, and the directional derivative of $\lambda_{i,j}$ is negative for all j. This is true, for example, of the control signal reference used throughout this work.

Therefore, with use of the introduced regularization (5.18), the stationary points of Games (5.1) and (5.3) have a one-to-one correspondence, warranting the use of Game (5.3) in place of Game (5.1).

5

5.B Differentiating through BMG

The problem of finding q_1, q_2 which satisfy (5.11) (as is the task of the function BMG), can be equivalently expressed as the linear complementarity problem [155]

$$\begin{aligned} &\text{find} \;\; p_1, p_2 \\ \text{s.t.} \;\; &p_1 \geq 0 \perp \bar{A} p_2 \geq 1 \\ &p_2 \geq 0 \perp \bar{B}^\mathsf{T} p_1 \geq 1. \end{aligned} \tag{5.19}$$

The solution (q_1, q_2) to the BMG are related to the solution to (5.19) by the relations

$$(q_1)_i = \frac{(p_1)_i}{\sum_k (p_1)_k}, \quad (q_2)_i = \frac{(p_2)_i}{\sum_k (p_2)_k}. \tag{5.20}$$

It is assumed that $\bar{A}$ and $\bar{B}$ are derived from the original matrices A, B, as the following. $\bar{A}_{i,j} := A_{i,j} + \alpha$, $\bar{B}_{i,j} := B_{i,j} + \beta$, for some positive constants α, β such that every element of $\bar{A}$ and $\bar{B}$ are strictly positive. Furthermore, the 1s appearing in the right-hand side of the constraints in (5.19) are assumed to represent vectors of appropriate dimension with each value equal to 1.

Consider some solution p_1, p_2 to (5.19) in which strict complementarity holds for each condition, e.g. either $p_{1,j} = 0$ or $(\bar{A} p_2)_j = 1$, but not both. For each $j \in \{1, 2\}$, Denote the index sets $\mathcal{I}_j^+ := \{i : (p_j)_i > 0\}$. Then let

$$\begin{aligned} p_1^+ &:= [p_1]_{\mathcal{I}_1^+}, & p_2^+ &:= [p_2]_{\mathcal{I}_2^+}, \\ \bar{A}^+ &:= [A]_{\mathcal{I}_1^+, \mathcal{I}_2^+}, & \bar{B}^+ &:= [B]_{\mathcal{I}_1^+, \mathcal{I}_2^+}. \end{aligned}$$

In words, p_1^+ is the vector formed by only considering the non-zero elements of p_1, and $\bar{B}^+$ is the matrix formed by considering the columns specified by $\mathcal{I}_1^+$ and rows specified by $\mathcal{I}_2^+$. By the strict complementarity, at equilibrium, it is that

$$\begin{bmatrix} 0 & \bar{A}^+ \\ (\bar{B}^+)^\top & 0 \end{bmatrix} \begin{bmatrix} p_1^+ \\ p_2^+ \end{bmatrix} = 1, \tag{5.21}$$

where, as before, the right-hand side 1 is a vector consisting of all 1s.

The values $p_j^- := [p_j]_i, i \notin \mathcal{I}_j^+$ are defined to be identically 0, and as such have 0 derivative with respect to the values $\bar{A}, \bar{B}$. The derivatives of remaining portion of the solution, p_1^+, p_2^+, can be evaluated from (5.21). If the matrix on the left-hand-side of (5.21) is singular, then the resulting solution is in fact non-isolated (there exist a continuum of solutions satisfying (5.19)), and the derivatives of the solution are not defined. If the matrix is non-singular, then necessarily so are $\bar{A}^+$ and $\bar{B}^+$, and the isolated solutions of p_1^+, p_2^+ are locally related to the matrices $\bar{A}$, $\bar{B}$ as

$$\begin{aligned} p_1^+ &= (\bar{B}^+)^{-\top} 1, \\ p_2^+ &= (\bar{A}^+)^{-1} 1. \end{aligned} \tag{5.22}$$

In this form, the derivatives of each element of p_j^+ can be found by differentiating through the expressions (5.22). Combining the above results, the derivatives of the solution vector p_1, p_2 with respect to the problem data A, B, can be established as the following:

$$\begin{aligned} \frac{\partial (p_1)_i}{\partial A_{j,k}} &:= 0, \\ \frac{\partial (p_1)_i}{\partial B_{j,k}} &:= \begin{cases} 0 & : i \notin \mathcal{I}_1^+ \\ -((\bar{B}^+)^{-\top} I_{k,j} p_1^+)_i & : \text{else} \end{cases}, \\ \frac{\partial (p_2)_i}{\partial A_{j,k}} &:= \begin{cases} 0 & : i \notin \mathcal{I}_2^+ \\ -((\bar{A}^+)^{-1} I_{j,k} p_2^+)_i & : \text{else} \end{cases}, \\ \frac{\partial (p_2)_i}{\partial B_{j,k}} &:= 0. \end{aligned} \tag{5.23}$$

Above, the term $I_{j,k}$ is used to refer to the matrix consisting of zero everywhere except at the (j,k)-th position, which has value 1.

When strict complementarity does not hold at the solution to (5.19), then only directional derivatives of the solution vectors exist w.r.t. the problem data. The various directional derivatives are found by, for each condition which does not hold with strict complementarity, making a selection on whether that index should be included the sets $\mathcal{I}_j^+$ or not. Then proceeding with the remainder of calculations, the result forms one of the directional derivative for the system. The directions for which this derivative is valid are defined to be those which make the directional derivative consistent with the selected index sets.

The derivatives of the elements of p_1 and p_2 with respect to the cost matrices are formed by propagating the derivatives (5.23) through the relationships (5.20).

Chapter 6

Utilizing Single-Agent Demonstrations to Learn Multi-Agent Policies

Chapters 2 to 5 introduced several game-theoretic approaches to multi-agent interaction problems, including methods for intent inference, handling uncertainty, and efficient game solving. In these works, we relied on the assumption that the problems are smooth*, meaning that the costs and constraints (including dynamics) have well-defined first and second derivatives. This enabled solution techniques that leverage the local geometry to iteratively search for a local equilibrium. While this assumption holds for many problems of interest, such as autonomous driving, there are important settings where it becomes limiting. For instance, in multi-agent manipulation, the contact dynamics involved are often inherently non-smooth. Furthermore, the requirement for smoothness can make it challenging to specify the* task *itself, as users may be forced to create smooth approximations of the cost (and/or constraints) they truly care about, reducing expressiveness.*

In this chapter, we address these challenges by adopting a fundamentally different approach: we learn a multi-agent policy directly from a combination of expert demonstrations and a (potentially non-smooth) joint *cost function. Our main contribution is a training framework that first learns* single-agent policies *from demonstrations of basic skills (e.g. the ability to pick and place objects) and then composes these policies into* coordinated *multi-agent behavior by reasoning about the* joint cost *of actions across agents. Unlike previous chapters, we present this contribution primarily in the language of probabilistic inference rather than game theory. This perspective enables us to draw on recent tools for generative modeling of complex distributions. Despite this shift in perspective, we maintain links to game-theoretic reasoning to connect this work to the broader context of the dissertation. We validate our approach in a high-fidelity simulation of a two-agent manipulation task.*

6.1 Introduction

Imitation learning (IL) enables individual robots to learn complex behaviors such as object manipulation from expert demonstrations. Generative models such as diffusion policies have recently further enhanced this capability [23–26]. However, many real-world tasks require multi-agent interaction—e.g., a robotic manipulator relying on another arm to grab an out-of-reach object, clearing a table after dinner, or assembling a piece of furniture.

A key challenge in solving complex multi-agent tasks is the scarcity of data in these settings, which can be attributed to two main issues: (i) the product space of states and actions grows exponentially with the number of players and (ii) the cost of acquiring data for multi-agent systems is inevitably higher than for single-agent systems.

To tackle these challenges, this work focuses on the following question:

How can we generate multi-agent behavior while leveraging single-agent demonstrations?

We investigate this question based on the observation that the gap between single-agent and multi-agent behavior often does not arise in the low-level actuation of robots, but rather at a higher level of decision-making. In the out-of-reach pick-and-place setup, for instance, the core challenge lies in the coordination between agents (e.g., which robot should pick up the object and where to place it next) rather than on the low-level skill of object interaction (e.g. *how* to pick an object). As a result, single-agent data can serve as a strong prior for learning multi-agent policies.[1]

The main contribution of this chapter is CoDi, a method for synthesizing a single cohesive multi-agent policy by *guiding* multiple single-agent diffusion policies towards *coordinated* behavior. We achieve this via a multi-stage training procedure as summarized in Figure 6.1. First, the user provides single-agent demonstrations of basic single-agent skills (e.g. picking and placing objects). We use this data to train single-agent diffusion policies that capture the low-level skills of each agent. Next, the user provides a multi-agent cost function that penalizes deviations from desired multi-agent behavior (e.g. rewarding object movement towards the goal location while penalizing collisions with other agents). We construct a centralized guidance model from this cost function that bridges the gap between the marginal (single-agent) diffusion policies and the joint (multi-agent) diffusion policy.

As a result, our approach effectively inverts the conventional *centralized training, decentralized execution* paradigm [156]: we enable *decentralized* (single-agent) pre-training at the expense of requiring *centralized* execution. We contend that in many settings, this trade-off is preferable; enabling centralized execution via multi-agent communication is often less demanding than collecting multi-agent demonstration data.

We validate our approach in simulation, where we demonstrate that our method can (i) learn multi-agent policies while pre-training only on single-agent data, (ii) discover new collaboration strategies not present in the original single-agent demonstrations, and (iii) generates more efficient and accurate manipulation policies than a baseline that directly relies on multi-agent demonstrations, given the same amount of data.

[1] While we will use multi-agent manipulation as a running example throughout this chapter, the method applies more generally to multi-agent tasks in other domains.

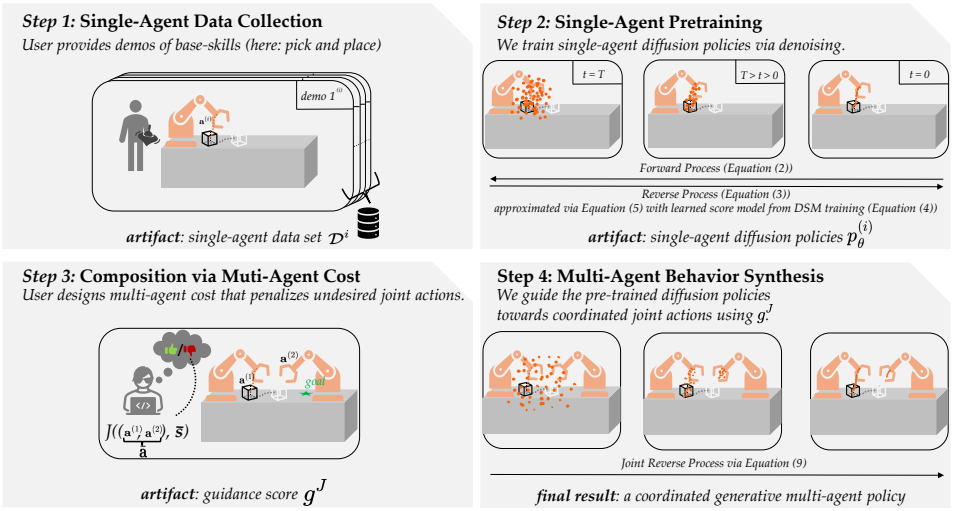


Figure 6.1: We present *Coordinated Diffusion (CoDi)*, a method for synthesizing coordinated multi-agent behavior from single-agent demonstrations.

6

6.2 Background: Forward- and Reverse-time Diffusion

This section provides a brief background to diffusion policies and their theoretical underpinnings in forward and reverse-time diffusion processes. For a comprehensive treatment of these foundations, we refer the reader to the tutorial by Lai et al. [157].

6.2.1 Forward-time Diffusion Processes: From Data to Noise

In order to understand the diffusion models as a generative paradigm, it is useful to first understand their sibling process: the forward-time diffusion process. This process models the act of incrementally corrupting *structured data* with *noise*. When the data $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is distributed according to $p^*(\mathbf{x})$, and $q_{\sigma(t)}(\mathbf{x}(t) \mid \mathbf{x})$ denotes the perturbation kernel at noise-time $t \in [0, T]$, then the noise-corrupted data is distributed according to

$$\mathbf{x}(t) \sim p^*_{\sigma(t)}(\mathbf{x}(t)) \coloneqq \int_{\mathcal{X}} p^*(\mathbf{x}) q_{\sigma(t)}(\mathbf{x}(t) \mid \mathbf{x}) \mathrm{d}\mathbf{x}, \tag{6.1}$$

If we choose a *Gaussian* perturbation kernel, i.e., $q_{\sigma(t)}(\mathbf{x}(t) \mid \mathbf{x}) \coloneqq \mathcal{N}(\mathbf{x}(t); \mathbf{x}, \sigma(t)^2 \mathbf{I})$, at monotonically increasing isotropic variance $\sigma(t)^2$, the process $\{\mathbf{x}(t)\}_{t \in [0,T]}$ matches the target distribution p^* at noise-time $t = 0$ (i.e., $p^* \equiv p^*_{\sigma(0)}$) and satisfies the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}(t) = \sqrt{2\dot{\sigma}(t)\sigma(t)} \mathrm{d}\mathbf{w}_t \tag{6.2}$$

where $\mathbf{w}_t$ is a standard Wiener process.

6.2.2 Reverse-time Diffusion Processes: From Noise to Data

Generative diffusion models [158–161] are grounded in the key idea of *reversing* the time direction in (6.2). Intuitively, this means that, while the forward-time diffusion process corrupts the structured data distribution p^* with noise (as in (6.2)) until it is indistinguishable from a (high-variance) Gaussian, the reverse-time diffusion process transforms samples of (high-variance) Gaussian noise into samples from the structured data distribution p^*.

This time reversal is enabled by the celebrated result of Anderson [162], which establishes that the following reverse-time SDE governs the noise-corrupted data $\mathbf{x}(t)$ as noise-time flows backwards:

$$\mathrm{d}\mathbf{x}(t) = -2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}(t))\mathrm{d}t + \sqrt{2\dot{\sigma}(t)\sigma(t)}\mathrm{d}\bar{\mathbf{w}}_t, \quad \mathbf{x}(T) \sim p^*_{\sigma(T)}(\mathbf{x}(T)). \tag{6.3}$$

Here, $\bar{\mathbf{w}}_t$ is a standard Wiener process when time flows backwards and $\mathrm{d}t$ is a *negative* time differential. Generative diffusion models leverage this SDE by learning a model of the only unknown in this equation: the term $\nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}(t))$. This term—the gradient of the log-likelihood of the mollified target distribution $p^*_{\sigma(t)}(\mathbf{x}(t))$—is commonly referred to as the *score* of $p^*_{\sigma(t)}(\mathbf{x}(t))$.

6

Training. Vincent [163] show that, when samples from the data distribution $p^*(\mathbf{x})$ are available, a *score model* $s^{p^*}_\theta(\mathbf{x}; \sigma)$ can be trained by minimizing the denoising score matching (DSM) loss

$$\mathcal{L}(\theta) \coloneqq \mathop{\mathbb{E}}_{\mathbf{x}\sim p^*(\mathbf{x}), t\sim\mathcal{W}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\left\| \sigma(t)^2 s^{p^*}_\theta(\mathbf{x} + \sigma(t)\epsilon; \sigma) + \sigma(t)\epsilon \right\|^2 \right], \tag{6.4}$$

where θ are the trainable parameters of the score model $s^{p^*}_\theta$, and $\mathcal{W}$ is a distribution over the noise-time t following [164].

Generating Samples. Once the score model is trained, it satisfies $s^{p^*}_\theta(\mathbf{x}; \sigma) \approx \nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}; \sigma)$. Additionally, when the terminal noise scale $\sigma(T)$ is sufficiently large, we have $p^*_{\sigma(T)}(\mathbf{x}) \approx \mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma(T)^2\mathbf{I})$. Therefore, we can generate samples from the target distribution $p^*(\mathbf{x})$ by integrating the reverse-time SDE

$$\mathrm{d}\mathbf{x}(t) = -2\dot{\sigma}(t)\sigma(t) \underbrace{s^{p^*}_\theta(\mathbf{x}(t); \sigma(t))}_{\approx \nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}(t))} \mathrm{d}t + \sqrt{2\dot{\sigma}(t)\sigma(t)}\mathrm{d}\bar{\mathbf{w}}_t, \quad \mathbf{x}(T) \sim \underbrace{\mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma(T)^2\mathbf{I})}_{\approx p^*_{\sigma(T)}(\mathbf{x}(T))} \tag{6.5}$$

backwards in time from T to 0 (c.f. (6.3)).

Remarks on Notation and Properties. We denote the implicit distribution induced by the generative process governed by the score model $s^{p^*}_\theta$ as $p_\theta(\mathbf{x})$. Importantly, however, while we can draw samples $\mathbf{x} \sim p_\theta(\mathbf{x})$, we cannot evaluate its density $p_\theta(\cdot)$.

6.2.3 Diffusion Policies

While diffusion models have been popularized in the context of image generation [161, 165], in this work we will use them to represent a stochastic policy for decision-making as in [23–26]; i.e., to represent $\pi(\mathbf{a} \mid \mathbf{s})$—a distribution of actions $\mathbf{a} \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ to take given a specific state $\mathbf{s} \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$.

One simple way of training *diffusion policies* is via imitation learning: given a dataset $\mathcal{D} \coloneqq \{(\mathbf{s}_{(d)}, \mathbf{a}_{(d)})\}_{d=1}^{D}$ of D state-action pairs $(\mathbf{s}_{(d)}, \mathbf{a}_{(d)})$ with $\mathbf{a}_{(d)} \sim \pi^*(\mathbf{a} \mid \mathbf{s}_{(d)})$, we train a score model s_θ^π via DSM as in (6.4). Mirroring the sampling procedure of Section 6.2.2, sampling actions from the policy, π, amounts to integrating the reverse-time SDE with the learned score model for a given state $\mathbf{s}$, which approximates the true distribution π^*.

Action Representation. Like prior works [24, 25], rather than predicting an action for a *single* time step, we predict an *action sequence* $\mathbf{a}_{1:K} \in \mathbb{R}^{n_a \times K}$—where column $\mathbf{a}_k$ represents the action applied k steps into the future. The action sequence is executed in a receding-horizon fashion and re-planning is warm-started from the previous solution to promote temporal consistency and computational efficiency [24, 25]. From here on forwards, we will overload $\mathbf{a}$ to denote the action *sequence* $\mathbf{a}_{1:K}$ for brevity and use $\mathcal{A}$ to denote the corresponding space of action sequences.

6.3 Composing Multi-Agent Behavior from Single-Agent Diffusion Policies

In this work, we are interested in synthesizing policies that allow *multiple* agents to *collaboratively* interact with each other. In theory, the concept of diffusion policies introduced in Section 6.2.3 naturally extends to such a multi-agent setting. However, naive training of such a multi-agent diffusion policy would require vast amounts of *multi-agent* demonstrations: expert demonstrations of how *all* agents should behave *jointly* in a given state.

Therefore, a naive extension of diffusion policies to the multi-agent setting (i.e, training a joint policy from multi-agent expert demonstrations) faces two key challenges:

(i) the joint state-action space grows exponentially with the number of agents, making it difficult to ensure good coverage of the entire space with expert demonstrations,

(ii) providing demonstrations for multi-agent systems is cumbersome, even simply due to the logistical challenge of controlling N agents simultaneously.

To address these challenges, this section introduces our **main contribution**: *a framework for composing multi-agent behavior from single-agent diffusion policies.*

A Word on Notation: Distinguishing Single-Agent and Multi-Agent Quantities. Before we proceed to present our main contribution, we introduce some notation to clearly distinguish between single-agent and multi-agent quantities. We will use superscripts, i.e., $(\cdot)^{(i)}$, to denote quantities relating to the i-th of N agents and omit such superscripts when we refer to quantities pertaining to all N agents jointly. Following this convention, we use

- $\mathbf{s}^{(i)} \in \mathcal{S}^{(i)}$ to denote the state of the i-th agent, and $\mathbf{s} \in \mathcal{S}$ to denote the joint state of all N agents, both of which are related via the mapping $f^{(i)}_{\text{single}} : \mathcal{S} \to \mathcal{S}^{(i)}$ so that $\mathbf{s}^{(i)} = f^{(i)}_{\text{single}}(\mathbf{s})$,
- $\mathbf{a}^{(i)} \in \mathcal{A}^{(i)}$ to denote the action of the i-th agent, $\mathbf{a} \coloneqq (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(N)}) \in \mathcal{A} \coloneqq \prod_{i=1}^{N} \mathcal{A}^{(i)}$ to denote the joint action of all N agents,
- $\pi(\mathbf{a} \mid \mathbf{s})$ to denote the multi-agent policy that characterizes a distribution over joint actions given a joint state,
- and $\mathcal{D}^{(i)} = \{(\mathbf{s}^{(i)}_{(d)}, \mathbf{a}^{(i)}_{(d)})\}_{d=1}^{D_i}$ to denote a dataset of single-agent demonstrations for agent i.

6.3.1 Key Idea: Multi-Agent Learning with Single-Agent Data Constraints

Consider a multi-agent task such as moving an object to a specific goal location on a wide table with two robots as shown in Figure 6.1. Since each robot can only reach a limited area of the table, they need to collaborate to complete the task. It is clear that many skills involved in this task pertain to individual agents, most importantly the ability to pick up objects and place them elsewhere. Once both robots possess these skills, more complex behaviors—such as passing an object to another agent who can otherwise not reach the object—can emerge through careful coordination of individual skills across agents.

Inspired by this intuition, we propose the following workflow, dubbed Coordinated Diffusion (CoDi), summarized in Figure 6.1.

Step 1: Provide Single-Agent Demonstrations. The user provides single-agent demonstrations of basic skills for each agent. In our running example of Figure 6.1, this means providing demonstrations of picking and placing objects at various positions on the table. Let $\mathcal{D}^{(i)}$ denote the dataset of single-agent demonstrations for agent i.

Step 2: Single-Agent Pretraining. We fit a *single-agent diffusion policy* $p^{(i)}_\theta$ for each agent based on its corresponding dataset $\mathcal{D}^{(i)}$. This diffusion policy thus captures a probabilistic "library" of base-skills for each agent. Note that when agents are homogeneous, the same diffusion policy can be shared across agents.

Step 3: Multi-Agent Cost Function. The user provides a *multi-agent cost function* J: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$, that penalizes deviations from desired multi-agent behavior (in our running example, rewarding object movement towards the goal location while penalizing collisions with other agents).

Step 4: Deployment via Multi-Agent Policy Composition. At test time, we generate coordinated multi-agent behavior by composing the single-agent diffusion policies with the multi-agent cost function into a multi-agent diffusion policy, π.

Mathematically, our proposed multi-agent policy takes the following composite form:

$$\pi(\mathbf{a} \mid \mathbf{s}) \coloneqq \frac{1}{Z(\mathbf{s})} \exp\left(\frac{-J(\mathbf{s}, \mathbf{a})}{\lambda}\right) p_\theta(\mathbf{a} \mid \mathbf{s}), \quad p_\theta(\mathbf{a} \mid \mathbf{s}) \coloneqq \prod_{i=1}^{N} p_\theta^{(i)}(\mathbf{a}^{(i)} \mid f_{\text{single}}^{(i)}(\mathbf{s})), \tag{6.6}$$

user-defined multi-agent cost

learned from single-agent data

where $p_\theta(\mathbf{a} \mid \mathbf{s})$ takes the role of a prior, structured as the product of (independent) single-agent diffusion policies, and $Z(\mathbf{s})^{-1} \exp\left(-J(\mathbf{s}, \mathbf{a})\lambda^{-1}\right)$ takes the role of a coupling term, assigning higher probability to joint actions that result in lower cost, with $\lambda \in \mathbb{R}_{\geq 0}$ controlling the strength of this coupling and the scalar $Z(\mathbf{s}) \coloneqq \int_{\mathcal{A}} \exp\left(-J(\mathbf{s}, \mathbf{a})\lambda^{-1}\right) \mathrm{d}\mathbf{a}$ being the normalization constant.

Below, we provide two main motivations for this policy structure: a connection to game-theoretic principles and an analysis of its design space.

6.3.2 A Connection to Game-Theoretic Concepts

To further motivate the proposed policy structure, we offer a game-theoretic interpretation of the proposed policy structure.

To this end, we remark that the proposed policy π in (6.6) is the solution of the following joint optimization problem:

$$\pi(\mathbf{a} \mid \mathbf{s}) \in \arg\min_{\tilde{\pi} \in \mathcal{P}\{\mathcal{A} \mid \mathcal{S}\}} \mathbb{E}_{\mathbf{a} \sim \tilde{\pi}} \left[J(\mathbf{a}, \mathbf{s})\right] - \lambda D_{\mathrm{KL}}\left(\tilde{\pi} \| p_\theta\right), \tag{6.7}$$

where $\mathcal{P}\{\mathcal{A} \mid \mathcal{S}\}$ is the space of all possible *joint* policies.[2] Intuitively, in this game, players *jointly* seek to minimize their expected cost while regularizing their strategy towards the data-driven prior distribution p_θ. This connection reveals that, by finding a generative model for our policy π, we are finding the solution of a *cooperative* game.[3]

6.3.3 Exploring the Design Space of CoDi

Let us take a moment to understand what kind of multi-agent behavior we can capture via the proposed policy structure of (6.6). The key question guiding this analysis is: *where does the "knowledge" of the multi-agent behavior really come from?* Is it baked into the single-agent demonstrations provided by the user? Or is it encoded in the user-defined multi-agent cost function?

To formalize this analysis, let π^* denote a *hypothetical multi-agent target policy* that we wish to approximate with π via our proposed approach. Note that this target policy can be of arbitrary structure, i.e., it can be any distribution over the joint action space $\mathcal{A}$; not limited to the composite form of (6.6). Under which conditions can our learned policy, π, approximate such a target policy π^*?

Decomposing the Approximation Error. To answer this question, we can decompose the Kullback–Leibler (KL)-divergence between the target policy π^* and our learned policy

[2] This connection is well-established in prior work [166–168], known as the Donsker-Varadhan variational formula; therefore, we omit the proof.

[3] If we were to restrict ourselves to the language of non-cooperative games, we can also interpret equation (6.7) as a correlated quantal response equilibrium [169] in a game with identical incentives, but the interpretation as joint optimization in (6.7) more clearly reveals that the cooperative nature of the resulting solution.

π as follows:

$$D_{\mathrm{KL}}(\pi^* \| \pi) = \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi^*}\Big[\log \pi^*(\mathbf{a} \mid \mathbf{s}) - \log \pi(\mathbf{a} \mid \mathbf{s})\Big] \tag{6.8a}$$

$$= \underbrace{\mathop{\mathbb{E}}_{\mathbf{a}\sim\pi^*}\left[\log \frac{\pi^*(\mathbf{a} \mid \mathbf{s})}{p_\theta(\mathbf{a} \mid \mathbf{s})}\right]}_{=D_{\mathrm{KL}}(\pi^*\|p_\theta)\geq 0} + \underbrace{\log Z + \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi^*}\left[\frac{J(\mathbf{s},\mathbf{a})}{\lambda}\right]}_{\neq 0}. \tag{6.8b}$$

Here, the first corresponds to the KL-divergence between the target policy π^* and the data-driven component p_θ, and the second corresponds to the contribution of the user-defined multi-agent cost function J.

Analyzing the Approximation Error. From this decomposition, it is clear that, if the data-driven component p_θ (learned from single-agent demonstrations) is not able to capture the target behavior π^* perfectly, i.e., $D_{\mathrm{KL}}(\pi^* \| p_\theta) > 0$, then the multi-agent cost function J must compensate for this error. Analyzing (6.8b) further, we can see under which condition this error compensation is achievable: the target behavior π^* must be absolutely continuous w.r.t. the data-driven component p_θ ($\pi^* \gg p_\theta$) in the sense that $p_\theta(\mathbf{a} \mid \mathbf{s}) = 0$ must imply $\pi^*(\mathbf{a} \mid \mathbf{s}) = 0$ for all $\mathbf{a} \in \tilde{\mathcal{A}} \subseteq \mathcal{A}$ for which $\tilde{\mathcal{A}}$ has non-zero measure under π^*. When this condition is satisfied, the optimally compensating multi-agent cost function is given by

$$J^*(\mathbf{s},\mathbf{a}) = -\lambda \log Z(\mathbf{s}) + \lambda \log \frac{p_\theta(\mathbf{a} \mid \mathbf{s})}{\pi^*(\mathbf{a} \mid \mathbf{s})}. \tag{6.9}$$

This result highlights that the role of J is to account for the log-likelihood gap between π^* and p_θ.[4] If the demonstrations are perfectly informative, we would have $\log \frac{p_\theta(\mathbf{a}|\mathbf{s})}{\pi^*(\mathbf{a}|\mathbf{s})} = 0$ and it is easy to see that $J^*(\mathbf{s},\mathbf{a}) \propto 1$. The larger the KL-gap between p_θ and π^* is, the more information must be encoded in the cost function to compensate for this error.

Key Insight. In conclusion, this analysis reveals two important results when trying to learn multi-agent behavior π^* with π in the proposed form of (6.6) with our method. First, it reveals an important requirement for the single-agent demonstrations provided by the user: *The single-agent demonstrations must be sufficiently rich to cover the support of the target behavior π^*.* And second, (6.9) reveals that the more informative the single-agent demonstrations are for the targeted multi-agent behavior, the less design effort needs to go into J.

6.4 Sampling from CoDi Policies

Sampling from the multi-agent policy in (6.6) is non-trivial for several reasons. First, even computing the correct normalizing constant is challenging. Moreover, since the data-driven component is represented by diffusion policies, its distribution is only known in terms of a generative model: generating samples is easy but computing the density function is not (c.f. Section 6.2).

[4] See Section 6.A.1 for more details.

In this section, we show how to exploit the structure of (6.6) to directly construct a generative model for the multi-agent policy, thereby tackling the challenges of sampling from the multi-agent policy. We present this key result in two steps. First, in Section 6.4.1, we derive a reverse-time SDE following the structure of (6.3) whose numerical integration generates samples from the multi-agent policy $\pi(\mathbf{a} \mid \mathbf{s})$. Then, in Section 6.4.2, we show how to construct estimators for the required ingredients of this generative model.

6.4.1 Constructing a Generative Model for the Multi-Agent Policy

To construct a generative model for the multi-agent policy, we seek a SDE of the form of (6.3) whose solution matches the target distribution $\pi(\mathbf{a} \mid \mathbf{s})$.

The Multi-Agent Reverse-Time SDE. In Section 6.2.2, we saw that a generative diffusion model for a distribution $p^*(\mathbf{x})$ can be constructed by learning a score model that satisfies $s_\theta(\mathbf{x}; \sigma) \approx \nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}; \sigma)$ and solving the reverse-time SDE in (6.5) backwards in time from T to 0. By symmetry, in order to construct a generative model for $\pi(\mathbf{a} \mid \mathbf{s})$, we need a score model that satisfies $s^\pi_\theta(\mathbf{a}; \sigma(t), \mathbf{s}) \approx \nabla_{\mathbf{a}} \log \pi_{\sigma(t)}(\mathbf{a}; \mathbf{s})$, where $\pi_{\sigma(t)}(\mathbf{a}; \mathbf{s}) \coloneqq \int_{\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{s}) q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \mathrm{d}\mathbf{a}$ and solve

$$\mathrm{d}\mathbf{a}(t) = -2\dot{\sigma}(t)\sigma(t) \underbrace{s^\pi_\theta(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\approx \nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t); \mathbf{s})} \mathrm{d}t + \sqrt{2\dot{\sigma}(t)\sigma(t)} \mathrm{d}\bar{\mathbf{w}}_t; \ \mathbf{a}(T) \sim \underbrace{\mathcal{N}(\mathbf{x}; \mathbf{0}, \sigma(T)^2 \mathbf{I})}_{\approx \pi_{\sigma(T)}(\mathbf{a}(T); \mathbf{s})}. \tag{6.10}$$

Observe that the only unknown in this equation is the *multi-agent score model* $s^\pi_\theta(\mathbf{a}; \sigma(t), \mathbf{s})$. Unlike in conventional diffusion models, however, we cannot trivially learn this score model via DSM from samples as in (6.4) because we precisely do not have access to any samples from the multi-agent policy. Therefore, to overcome this chicken-and-egg problem, we must construct a model of the multi-agent score $s^\pi_\theta(\mathbf{a}; \sigma(t), \mathbf{s})$ in a different way.

Recognizing the Single-Agent Score Contribution to the Multi-Agent Score. To find an estimator of the multi-agent score model, we first decompose it in terms of the single-agent score models. To this end, as we detail in Appendix 6.A.2, we can decompose the multi-agent score as follows:

$$\nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t); \mathbf{s}) = \tag{6.11}$$
$$\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) + \underbrace{\nabla_{\mathbf{a}(t)} \log \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{s}, \mathbf{a})}{\lambda} \right) \right]}_{g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) \coloneqq}.$$

Analyzing the first term reveals that it is the score of the data-driven component p_θ whose score model can be trivially constructed from the scores learned by the single-agent diffu-

sion policies:

$$\nabla_{\mathbf{a}} \log p_{\theta\,\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) = \nabla_{\mathbf{a}} \log \prod_{i=1}^{N} p_{\theta}^{(i)}{}_{\sigma(t)}(\mathbf{a}^{(i)} \mid f_{\text{single}}^{(i)}(\mathbf{s})) \tag{6.12}$$

$$= \begin{bmatrix} \nabla_{\mathbf{a}^{(1)}} \log p_{\theta}^{(1)}{}_{\sigma(t)}(\mathbf{a}^{(1)} \mid f_{\text{single}}^{(1)}(\mathbf{s})) \\ \vdots \\ \nabla_{\mathbf{a}^{(N)}} \log p_{\theta}^{(N)}{}_{\sigma(t)}(\mathbf{a}^{(N)} \mid f_{\text{single}}^{(N)}(\mathbf{s})) \end{bmatrix} \tag{6.13}$$

$$= \begin{bmatrix} s_{\theta}^{p^{(1)}}(\mathbf{a}^{(1)}; \sigma(t), f_{\text{single}}^{(1)}(\mathbf{s})) \\ \vdots \\ s_{\theta}^{p^{(N)}}(\mathbf{a}^{(N)}; \sigma(t), f_{\text{single}}^{(N)}(\mathbf{s})) \end{bmatrix}$$

$$=: s_{\theta}^{p}(\mathbf{a}; \sigma(t), \mathbf{s}),$$

where $s_{\theta}^{p^{(i)}}$ denotes the score model underpinning the i-th agent's single-agent diffusion policy $p_{\theta}^{(i)}$. This reveals that we do not need to approximate the multi-agent score from scratch: we can re-use the single-agent score models already available from the underpinning single-agent diffusion policies. Consequently, the only missing ingredient is the second term in (6.11) which encodes the contribution of the user-defined multi-agent cost function J. As we show in Section 6.B, this term can be related to the framework of classifier guidance [165]. We therefore refer to $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$ as the guidance score.

Key Insight. We can sample from our multi-agent policy π in (6.6) by numerically integrating the reverse diffusion SDE in (6.10). The score model involved in this SDE can be constructed from the single-agent scores (which we already have from pre-training the single-agent diffusion policies) plus an extra guidance score term. Constructing an estimator for this guidance score term will be the focus of the next section.

6.4.2 Multi-Agent Guidance Score Estimation

Having established that the only missing ingredient required for sampling from the multi-agent policy is the guidance score, $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$, in (6.11), we now discuss two techniques for estimating this term.

Design Constraints. When constructing these estimators, we specifically consider two design constraints: (i) the user-provided cost function J may not be differentiable; and (ii) we do not have access to any samples from the multi-agent policy $p_{\theta}(\mathbf{a} \mid \mathbf{s})$. Hence, these estimators must be constructed from point-wise evaluations of the black-box cost J, without access to its derivatives.

High-Level Overview. In the following, we propose two estimators that satisfy these design constraints. Section 6.4.2 proposes an online estimator that approximates g^J in a sampling-based manner. This estimator is useful for design-time exploration of different cost functions, but is computationally expensive for final deployment because it requires taking many samples online. Therefore, Section 6.4.2 proposes an offline estimator that

amortizes this computation into a guidance network, learning an approximation of the vector field $g^J(\cdot; \mathbf{s}, \sigma(t))$ directly. This estimator is computationally efficient for final deployment at the expense of an additional training phase.

Online Guidance Score Estimation

Our online guidance score estimator approximates $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$ in a sampling-based (i.e., gradient-free) manner. To achieve this, as shown in Appendix 6.A.3, we rewrite $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$ as follows:

$$g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) = \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} | \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[w(\mathbf{a}(t), \mathbf{a}, \mathbf{s}, t) \nabla_{\mathbf{a}(t)} \log q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \right] \tag{6.14}$$

where

$$w(\mathbf{a}(t), \mathbf{a}, \mathbf{s}, t) \coloneqq \frac{\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) - \mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} | \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) \right]}{\mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} | \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) \right]} \tag{6.15}$$

This formulation side-steps the need to compute gradients of J.[5] Next, we approximate the expectations in (6.14) in a sampling-based manner using Monte Carlo integration. This requires tractable generation of samples from the posterior $p(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))$ (sampling *noise-free* data $\mathbf{a}$ given *noisy* data $\mathbf{a}(t)$). To this end, we construct a Gaussian posterior approximation using Tweedie's formula [170, 171] as shown in Appendix 6.A.4, admitting

$$\begin{aligned} &p(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) \approx \\ &\quad \mathcal{N}(\mathbf{a}; \underbrace{\mathbf{a}(t) + \sigma(t)^2 s_\theta^p(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\mu_{\text{online}} \coloneqq}, \underbrace{\sigma(t)^2 \mathbf{I} + \sigma(t)^4 \nabla_{\mathbf{a}(t)} s_\theta^p(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\Sigma_{\text{online}} \coloneqq}). \end{aligned} \tag{6.16}$$

Observe that the parameters μ_{online} and Σ_{online} of this distribution are constructed directly from the single-agent score models characterizing $s_\theta^p(\mathbf{a}(t); \sigma(t), \mathbf{s})$ (c.f. (6.12)), requiring no additional training. Since, in practice, the score model s_θ^p is a neural network, its Jacobian, $\nabla_{\mathbf{a}(t)} s_\theta^p(\mathbf{a}(t); \sigma(t), \mathbf{s})$ (required to compute Σ_{online}), can be computed using automatic differentiation.

Key Insight. In summary, exploiting Equations (6.14) to (6.16) and applying Monte Carlo integration, we can approximate the guidance score of (6.44) with M samples as

$$\begin{aligned} &g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) \approx \\ &\quad g^J_{\text{online}}(\mathbf{a}(t); \mathbf{s}, \sigma(t)) \coloneqq \frac{1}{M} \sum_{m=1}^{M} w^J_{\text{online}}(\mathbf{a}^{(m)}, \mathbf{s}) \nabla_{\mathbf{a}(t)} \log q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}^{(m)}) \end{aligned} \tag{6.17}$$

[5] When J is differentiable in $\mathbf{a}$, we can alternatively compute a lower-variance estimate of the expectation in (6.14) using the reparameterization trick. Here, however, we are interested in the general case where the user-provided multi-agent cost need not be differentiable.

where $\mathbf{a}^{(m)}$ are sampled from (6.16) and

$$w^J_{\text{online}}(\mathbf{a}, \mathbf{s}) \coloneqq \exp\left(\frac{-J(\mathbf{s}, \mathbf{a})}{\lambda}\right) / \mu_w - 1, \text{ where } \mu_w \quad \coloneqq \frac{1}{M} \sum_{m=1}^{M} \exp\left(\frac{-J(\mathbf{s}, \mathbf{a}^{(m)})}{\lambda}\right). \tag{6.18}$$

This estimator can be evaluated for varying choices of J without retraining, making it a flexible design-time tool that allows a user to try out different cost functions quickly.

Learning a Guidance Score Estimator Offline

While the online estimator provides design-time flexibility, evaluating g^J for many samples can be computationally expensive. Therefore, for final deployment, we also propose an amortized guidance score estimator.

As we show in Appendix 6.A.5, we can train such an estimator by minimizing the following loss:

$$\mathcal{L}_g(\psi) \coloneqq \mathop{\mathbb{E}}_{\mathbf{a}\sim p_\theta(\mathbf{a}|\mathbf{s}), t\sim\mathcal{W}, \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) \left\|\sigma(t)^2 s^\pi_\psi(\mathbf{a} + \sigma(t)\epsilon; \sigma) + \sigma(t)\epsilon\right\|^2\right], \tag{6.19a}$$

$$\text{where} \quad s^\pi_\psi(\mathbf{a}(t); \mathbf{s}, \sigma(t)) = \underbrace{s^p_\theta(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\text{base model}} + \underbrace{g^J_\psi(\mathbf{a}(t); \mathbf{s}, \sigma(t))}_{\text{learned guide}}. \tag{6.19b}$$

This loss takes the form of a re-weighted DSM loss (c.f. (6.4).[6] During training, the base score model s^π_θ (previously trained on single-agent data) is frozen, and only the parameters of the residual guidance term g^J_ψ are updated. Thus, (6.19) fine-tunes this residual model using the user-defined cost J as the learning signal. Finally, observe that evaluation of the loss (6.19) requires only samples from the base-model $p_\theta(\mathbf{a} \mid \mathbf{s})$, which are readily available via the single-agent diffusion models.

Key Insight. In summary, an offline guidance score estimator can be trained by fine-tuning the residual model of (6.19b) by minimizing the re-weighted DSM loss (6.40a) over the product of single-agent data distributions, $p_\theta(\mathbf{a} \mid \mathbf{s})$.

6.5 Experimental Evaluation

The following simulation experiments are designed to support the key claims that CoDi (i) learns multi-agent policies while pre-training only on single-agent data, (ii) discovers new collaboration strategies not present in the original single-agent demonstrations, and (iii) generates more efficient and accurate manipulation policies than a baseline that directly relies on multi-agent demonstrations, given the same amount of data.

[6] Interestingly, Ma et al. [172] find a similar re-weighted DSM loss when training (single-agent) diffusion-policies from scratch. By contrast, our approach additionally accounts for priors via the base model s^π_θ.

6.5.1 Task

To test our method, we instantiate the out-of-reach manipulation task from Figure 6.1 in a high-fidelity simulation environment using the Isaac Gym simulator [173]. As shown in Figure 6.2, in this task, two 7-DoF Franka arms are placed at opposite ends of a wide table. The dimensions of the table (width $1.8\,\text{m}$, depth $1.2\,\text{m}$) are such that no single robot can reach across the entire table. The task requires both robots to collaboratively move the object (a red cube with an edge length of $5\,\text{cm}$) to the goal location (marked by a green star in Figure 6.2) within a tolerance of $5\,\text{cm}$. For each simulated episode, we randomly generate initial configurations, i.e., pairs of initial object poses and goal locations.

6.5.2 Compared Methods

In the simulation environment, we compare the following methods:

CoDi (online) This variant of our approach instantiates CoDi with the online guidance score estimator proposed in Section 6.4.2. We include this method to showcase the performance observed when using our online score estimator during design-time exploration without requiring additional training.

CoDi (offline) This variant of our approach instantiates CoDi with the amortized guidance score estimator learned offline as proposed in Section 6.4.2. We include this method to showcase the final performance achievable once the user has settled on a cost function design and trains a dedicated guidance score estimator.

Multi-Agent Classifier-Guidance (MACG): This method instantiates vanilla classifier-guidance [165] applied to a *multi-agent* diffusion policy—amounting to a multi-agent instantiation of the approach presented in Janner et al. [25]. We include this method to provide a reference for the performance achievable when multi-agent data were available.

Demonstration Data. We generate pick-and-place demonstrations using a simple state-machine-based controller. This controller drives the end-effector via inverse kinematics to move the object to randomly sampled goal locations on the table. At each invocation, the controller checks if the object is within reach; if so, it picks up the object and places it at the goal. For MACG's multi-agent demonstrations, when multiple robots can reach the object, the controller randomly assigns it to one robot. If no robot interacts with the object for 100 time steps, the position of the cube is reset to a new random location on the table. The datasets for all methods contain the same *number* of samples: we provide 2k demonstrations, split into 100k receding-horizon segments of 16 action predictions at a rate of $10\,\text{Hz}$.

Policy Representation. We represent the score model underpinning single- and multi-agent diffusion policies via a denoising UNet architecture as in Chi et al. [23]. In our experiments, diffusion policies predict desired end-effector translation and rotation velocities as well as desired gripper width. We use inverse kinematics to convert end-effector velocities into joint velocities.

Cost Function Design. The demonstration data only encodes knowledge of the base *skills* (here, the ability to pick up and place the object); not of the exact task to be performed (here, the position where the object should be placed). Instead, the task-specific information is encoded in the cost function. We chose this setup to capture a scenario where the user may want to use the same base policy for multiple different tasks (because generating demonstrations for each task is annoying). This philosophy is consistent with Janner et al. [25].

The cost function used for guidance takes the form of a weighted sum of cost components,

$$J(\mathbf{s}, \mathbf{a}) = w_{\text{goal}} J_{\text{goal}}(\mathbf{s}, \mathbf{a}) + w_{\text{collision}} J_{\text{collision}}(\mathbf{s}, \mathbf{a}) + w_{\text{engage}} J_{\text{engage}}(\mathbf{s}, \mathbf{a}). \tag{6.20}$$

The cost components are defined as follows: $J_{\text{goal}}(\mathbf{s}, \mathbf{a})$ measures the distance of the object from the goal resulting from applying action-sequence $\mathbf{a}$ from state $\mathbf{s}$; $J_{\text{collision}}(\mathbf{s}, \mathbf{a})$ is a binary indicator, equal to one if end-effectors are closer than a given safety distance as a result of taking the joint action sequence $\mathbf{a}$ from state $\mathbf{s}$ and zero otherwise; and $J_{\text{engage}}(\mathbf{s}, \mathbf{a})$ measures the distance of the closer robot to the object. Section 6.C provides additional implementation details on this cost function.

Note that, in order to evaluate J_{goal}, we need to make a prediction about the *combined effect* of all agents' actions on the future position of the object. Since such a prediction is not part of the diffusion policies' output, we obtain the prediction by *simulating* the outcome of the joint action sequence $\mathbf{a}$ from state $\mathbf{s}$ using Isaac Gym. Note that this simulation can be done for many samples in parallel during online-guidance score estimation of (6.17). Nonetheless, the computational cost of this simulation is non-negligible, motivating the use of an offline estimator at test time. We provide an additional discussion of this point in Section 6.6.

6.5.3 Evaluation & Metrics

We perform closed-loop simulations of all methods for 10 episodes. Initial states are sampled uniformly at random from the table; these include settings where robots must collaborate to complete the task since no single robot can reach both the initial object position and the goal location.

To assess the performance of the different methods, we compute the following metrics on closed-loop rollouts of the different policies:

Minimum Goal Distance: The minimum distance of the object to the goal. This metric provides insights into the manipulation accuracy.

Task Completion Time: The time it takes for the object to be placed within 5 cm of the designated goal location. This metric provides insights into task efficiency.

6.5.4 Results

Figures 6.3 and 6.4 summarize the results of our simulation experiments by reporting the empirical cumulative distribution function (ECDF) of each metric by method. These results provide the following key insights.

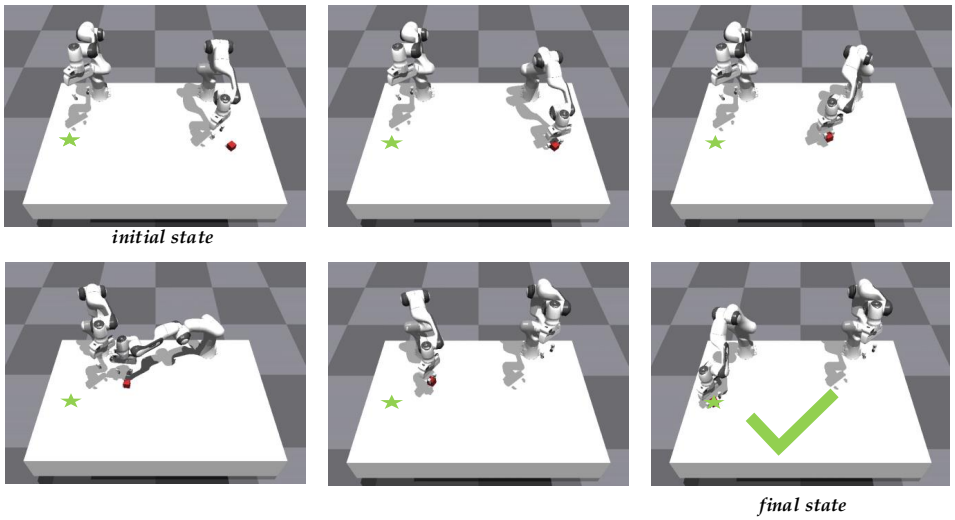


Figure 6.2: Qualitative example of the closed-loop behavior of CoDi. The goal is not reachable for the right robot. Our approach causes the right robot to pick up the object and place it at an intermediate location reachable by the left robot, which then completes the task.

We observe that both instantiations of CoDi achieve a fast task completion time (cf. Figure 6.3) and better manipulation accuracy (cf. Figure 6.4) than MACG. Since both methods are trained on the same amount of data, this suggests that—by learning manipulation skills in the smaller single-agent state-action space—CoDi makes use of the training data more efficiently than MACG.

Furthermore, note that object handovers are not part of single-agent demonstrations. Hence, these results reveal that CoDi is able to discover new collaboration strategies not present in the original single-agent demonstrations.

In sum, these results support the key claims enumerated above.

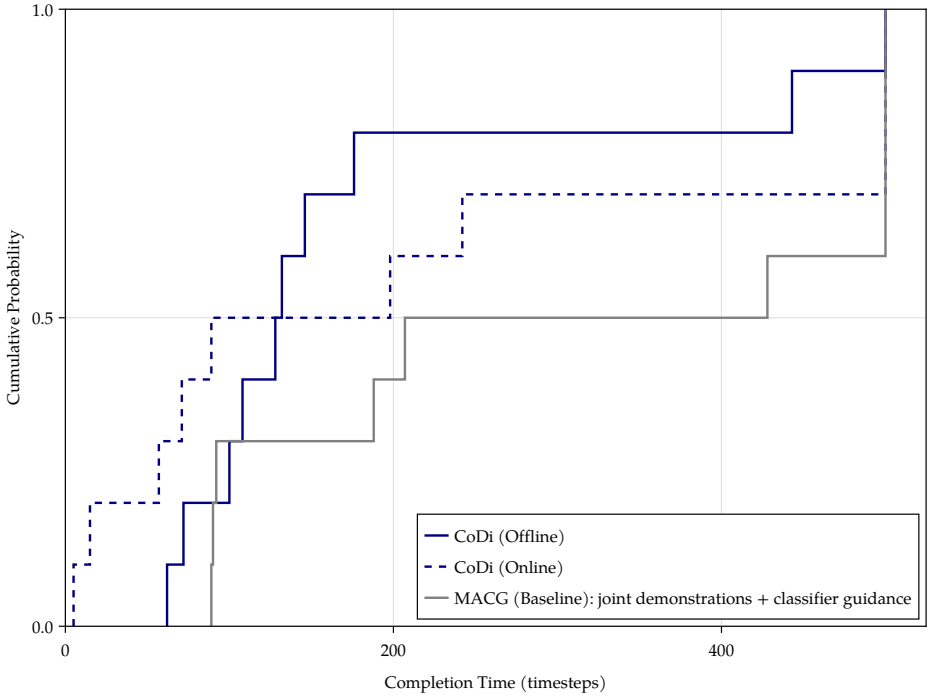


Figure 6.3: ECDF of task completion times by method.

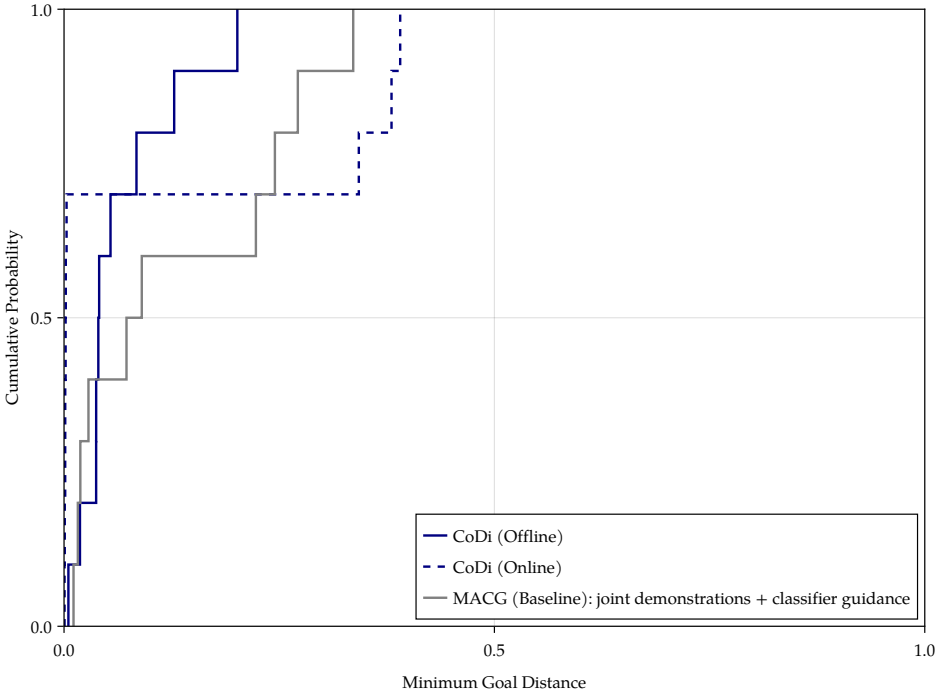


Figure 6.4: ECDF of minimum goal distances by method.

6.6 Limitations & Future Work

This work introduces CoDi, a method for synthesizing multi-agent behavior from single-agent demonstrations by guiding the product of single-agent diffusion policies toward coordinated multi-agent behavior using a multi-agent cost function.

Our simulation results demonstrate that CoDi discovers novel collaboration strategies absent from the original single-agent demonstrations and achieves superior manipulation accuracy and task efficiency compared to vanilla classifier guidance applied to diffusion policies trained on multi-agent demonstrations.

Despite these promising results, several limitations suggest directions for future work. A primary limitation of the current approach is the need to evaluate the joint effect of all agents' actions on the future environment state. In our manipulation example, this requires a simulator to assess the cost function. Although our offline score estimator mitigates the computational cost of these simulations, such a simulator may not always be available—especially for tasks involving complex physical phenomena (e.g., manipulating soft objects or even liquids). Additionally, while our proposed online score estimator facilitates rapid evaluation of a given cost design on the joint behavior of all agents without retraining, specifying an appropriate cost function remains challenging.

Combined, these limitations motivate future research into learning the guidance term directly from a *small* number of multi-agent demonstrations, thereby eliminating the need to define the multi-agent cost function explicitly. While this approach would relax the reliance on single-agent demonstrations alone, such an approach still shows promise in being more data-efficient overall than naive multi-agent imitation learning.

Furthermore, here we assumed fully *cooperative* behavior, requiring all players to optimize the same cost function and assuming correlated action distributions across agents. Future work should explore the more general *non-cooperative* setting, including those with distinct costs and independent action distributions.

Appendix for Chapter 6

Utilizing Single-Agent Demonstrations to Learn Multi-Agent Policies

6.A Derivations

6.A.1 Optimally Compensating Multi-Agent Cost Function

For completeness, we begin by repeating the decomposition result of (6.8), labeling the second term as $\gamma[J]$:

$$D_{\mathrm{KL}}(\pi^* \| \pi) = \underbrace{\mathop{\mathbb{E}}_{\mathbf{a}\sim\pi^*}\left[\log \frac{\pi^*(\mathbf{a} \mid \mathbf{s})}{p_\theta(\mathbf{a} \mid \mathbf{s})}\right]}_{D_{\mathrm{KL}}(\pi^* \| p_\theta) \geq 0} + \underbrace{\log Z + \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi^*}\left[\frac{J(\mathbf{s},\mathbf{a})}{\lambda}\right]}_{:= \gamma[J]}.$$

To achieve a perfect match of the target behavior, i.e., $D_{\mathrm{KL}}(\pi^* \| \pi) = 0$, we need $\pi^*(\mathbf{a} \mid \mathbf{s}) = \pi(\mathbf{a} \mid \mathbf{s})$ for *all* $\mathbf{a} \in \tilde{\mathcal{A}} \subseteq \mathcal{A}$ for which $\tilde{\mathcal{A}}$ has non-zero measure under π^*. Solving for J, this yields the optimally compensating multi-agent cost function

$$\pi^*(\mathbf{a} \mid \mathbf{s}) = \frac{1}{Z(\mathbf{s})} \exp\left(\frac{-J^*(\mathbf{a};\mathbf{s})}{\lambda}\right) p_\theta(\mathbf{a} \mid \mathbf{s}) \tag{6.21}$$

$$\implies J^*(\mathbf{a};\mathbf{s}) = -\lambda \log Z(\mathbf{s}) + \lambda \log \frac{p_\theta(\mathbf{a} \mid \mathbf{s})}{\pi^*(\mathbf{a} \mid \mathbf{s})} \tag{6.22}$$

6

6.A.2 Decomposition of π

$$\pi_{\sigma(t)}(\mathbf{a}(t);\mathbf{s}) = \int_{\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{s}) q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \mathrm{d}\mathbf{a} \tag{6.23a}$$

$$= p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) \frac{\int_{\mathcal{A}} \pi(\mathbf{a} \mid \mathbf{s}) q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \mathrm{d}\mathbf{a}}{p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s})} \tag{6.23b}$$

$$= p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) \int_{\mathcal{A}} \frac{\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) p_\theta(\mathbf{a} \mid \mathbf{s})}{Z(\mathbf{s})} \frac{q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a})}{p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s})} \mathrm{d}\mathbf{a} \tag{6.23c}$$

$$= p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) \int_{\mathcal{A}} \frac{\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right)}{Z(\mathbf{s})} p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) \mathrm{d}\mathbf{a} \tag{6.23d}$$

$$= p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) \frac{1}{Z(\mathbf{s})} \mathop{\mathbb{E}}_{\mathbf{a}\sim p_\theta(\mathbf{a}|\mathbf{a}(t),\mathbf{s},\sigma(t))}\left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right)\right], \tag{6.23e}$$

and thus we can express the multi-agent score as

$$\nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t);\mathbf{s}) = \nabla_{\mathbf{a}(t)} \log p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) - \cancelto{0}{\nabla_{\mathbf{a}(t)} \log Z(\mathbf{s})} \tag{6.24}$$

$$+ \underbrace{\nabla_{\mathbf{a}(t)} \log \mathop{\mathbb{E}}_{\mathbf{a}\sim p_\theta(\mathbf{a}|\mathbf{a}(t),\mathbf{s},\sigma(t))}\left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right)\right]}_{g^J(\mathbf{a}(t);\mathbf{s},\sigma(t)) :=}.$$

6.A.3 Simplification of $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$

We can simplify $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$:

$$g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) = \nabla_{\mathbf{a}(t)} \log \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \right] \tag{6.25a}$$

$$= \frac{\nabla_{\mathbf{a}(t)} \mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \right]}{\mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \right]} \tag{6.25b}$$

$$= \frac{\mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \nabla_{\mathbf{a}(t)} \log p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) \right]}{\mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \right]} \tag{6.25c}$$

Due to the identities

$$\nabla_{\mathbf{a}(t)} \log p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) = \nabla_{\mathbf{a}(t)} \log q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) - \nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \tag{6.26}$$

and

$$\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) = \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\nabla_{\mathbf{a}(t)} \log q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \right] \tag{6.27}$$

we conclude that

$$g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) = \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[w(\mathbf{a}(t), \mathbf{a}, \mathbf{s}, t) \nabla_{\mathbf{a}(t)} \log q_{\sigma(t)}(\mathbf{a}(t) \mid \mathbf{a}) \right] \tag{6.28}$$

where

$$w(\mathbf{a}(t), \mathbf{a}, \mathbf{s}, t) := \frac{\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right)}{\mathbb{E}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right) \right]} - 1 \tag{6.29}$$

6.A.4 Gaussian approximation of the posterior $p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))$

We can obtain a Gaussian approximation of the posterior $p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))$ by exploiting Tweedie's formula [170, 171] to recover the first two moments.

$$\mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a}] = \mathbf{a}(t) + \sigma(t)^2 \nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \tag{6.30}$$

$$\mathop{\mathrm{Cov}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a}] = \sigma(t)^2 \mathbf{I} + \sigma(t)^4 \nabla^2_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \tag{6.31}$$

Recall that the score model satisfies $s^\pi_\theta(\mathbf{a}(t); \sigma(t), \mathbf{s}) \approx \nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})$. Hence, we can approximate

$$p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) \approx \mathcal{N}(\mathbf{a}; \underbrace{\mathbf{a}(t) + \sigma(t)^2 s^\pi_\theta(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\mu_{\text{online}} :=}, \underbrace{\sigma(t)^2 \mathbf{I} + \sigma(t)^4 \nabla_{\mathbf{a}(t)} s^\pi_\theta(\mathbf{a}(t); \sigma(t), \mathbf{s})}_{\Sigma_{\text{online}} :=}). \tag{6.32}$$

The Jacobian of the score model, $\nabla_{\mathbf{a}(t)} s_\theta^\pi(\mathbf{a}(t); \sigma(t), \mathbf{s})$, is computed using automatic differentiation.

The formula for the first moment arises by expanding $\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})$:

$$\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) = \frac{\nabla_{\mathbf{a}(t)} p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})}{p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})} \tag{6.33a}$$

$$= \frac{\int_{\mathcal{A}} p_\theta(\mathbf{a} \mid \mathbf{s}) \nabla_{\mathbf{a}(t)} \overbrace{\mathcal{N}(\mathbf{a}(t) \mid \mathbf{a}, \sigma(t)^2 \mathbf{I})}^{q_{\sigma(t)}(\mathbf{a}(t)|\mathbf{a})} \mathrm{d}\mathbf{a}}{p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})} \tag{6.33b}$$

$$= \frac{\int_{\mathcal{A}} p_\theta(\mathbf{a} \mid \mathbf{s})(\mathbf{a} - \mathbf{a}(t)) \mathcal{N}(\mathbf{a}(t) \mid \mathbf{a}, \sigma(t)^2 \mathbf{I}) \mathrm{d}\mathbf{a}}{\sigma(t)^2 p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})} \tag{6.33c}$$

$$= \frac{1}{\sigma(t)^2} \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{s})} [\mathbf{a} - \mathbf{a}(t)] \tag{6.33d}$$

$$\implies \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{s})} [\mathbf{a}] = \mathbf{a}(t) + \sigma(t)^2 \nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \tag{6.33e}$$

The formula for the second moment arises by expanding $\nabla^2_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})$,

$$\begin{aligned} &\nabla^2_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) = \\ &\quad \frac{1}{p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})} \nabla^2_{\mathbf{a}(t)} p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \\ &\quad - \frac{1}{p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})^2} (\nabla_{\mathbf{a}(t)} p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}))(\nabla_{\mathbf{a}(t)} p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}))^\top, \end{aligned} \tag{6.34}$$

and exploiting the identities $\frac{\nabla p}{p} = \nabla \log p$ and

$$\nabla^2_{\mathbf{a}(t)} p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \tag{6.35a}$$

$$= \nabla_{\mathbf{a}(t)} \left(\frac{p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})}{\sigma(t)^2} \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a} - \mathbf{a}(t)] \right) \tag{6.35b}$$

$$= p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) \left(-\frac{1}{\sigma(t)^2} I + \frac{1}{\sigma(t)^4} \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[(\mathbf{a} - \mathbf{a}(t))(\mathbf{a} - \mathbf{a}(t))^\top \right] \right) \tag{6.35c}$$

to recover

$$\nabla^2_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}) = \tag{6.36a}$$

$$- \frac{1}{\sigma(t)^2} I + \frac{1}{\sigma(t)^4} \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[(\mathbf{a} - \mathbf{a}(t))(\mathbf{a} - \mathbf{a}(t))^\top \right] \tag{6.36b}$$

$$- (\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}))(\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}))^\top. \tag{6.36c}$$

Recognizing

$$\begin{aligned} &\mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[(\mathbf{a} - \mathbf{a}(t))(\mathbf{a} - \mathbf{a}(t))^\top \right] = \\ &\quad \mathop{\mathrm{Cov}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a}] + \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a}(t)] \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a}|\mathbf{a}(t), \mathbf{s}, \sigma(t))} [\mathbf{a}(t)]^\top \end{aligned} \tag{6.37}$$

we conclude that

$$\operatorname*{Cov}_{\mathbf{a}\sim p_\theta(\mathbf{a}|\mathbf{a}(t),\mathbf{s},\sigma(t))}[\mathbf{a}] = \sigma(t)^2\mathbf{I} + \sigma(t)^4\nabla^2_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s}). \tag{6.38}$$

6.A.5 Offline Guidance Score Estimation

As outlined in Section 6.2, generative diffusion models approximate a target distribution $p^*(\mathbf{x})$ by approximating the score of the mollified target distribution $\nabla_{\mathbf{x}(t)} \log p^*_{\sigma(t)}(\mathbf{x}(t))$ and DSM [163] learns the required score model $s^{p^*}_\theta(\mathbf{x};\sigma)$ of $\nabla_{\mathbf{x}(t)} \log p^*_{\sigma(t)}(\mathbf{x}(t))$ by minimizing the DSM loss given in (6.4), repeated here for convenience:

$$\mathcal{L}(\theta) \coloneqq \mathop{\mathbb{E}}_{\mathbf{x}\sim p^*(\mathbf{x}),t\sim\mathcal{W},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\left\| \sigma(t)^2 s^{p^*}_\theta(\mathbf{x}+\sigma(t)\epsilon;\sigma) + \sigma(t)\epsilon \right\|^2 \right]. \tag{6.39}$$

By symmetry, in order to learn $\nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t);\mathbf{s})$, we must find a score model $s^\pi_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t))$ that minimizes the following loss:

$$\tilde{\mathcal{L}}(\psi) \coloneqq \mathop{\mathbb{E}}_{\mathbf{a}(t)\sim\pi(\mathbf{a}(t);\mathbf{s}),t\sim\mathcal{W},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\left\| \sigma(t)^2 s^\pi_\psi(\mathbf{a}+\sigma(t)\epsilon;\sigma) + \sigma(t)\epsilon \right\|^2 \right] \tag{6.40a}$$

$$\propto \mathop{\mathbb{E}}_{\mathbf{a}\sim p_\theta(\mathbf{a}|\mathbf{s}),t\sim\mathcal{W},\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \left[\exp\left(\frac{-J(\mathbf{a};\mathbf{s})}{\lambda}\right) \left\| \sigma(t)^2 s^\pi_\psi(\mathbf{a}+\sigma(t)\epsilon;\sigma) + \sigma(t)\epsilon \right\|^2 \right] \tag{6.40b}$$

The latter formulation has the advantage of requiring only samples from the base-model $p_\theta(\mathbf{a} \mid \mathbf{s})$ rather than from the joint policy $\pi(\mathbf{a}(t);\mathbf{s})$.[7]

To leverage the single-agent priors, we parameterize $s^\pi_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t))$ in a residual fashion as

$$s^\pi_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t)) = \underbrace{s^p_\theta(\mathbf{a}(t);\sigma(t),\mathbf{s})}_{\text{base model}} + \underbrace{g^J_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t))}_{\text{learned guide}}, \tag{6.41}$$

where s^p_θ is the score model obtained from single-agent pre-training and g^J_ψ is the learned guidance score model tasked to match g^J of (6.11).

It is easy to verify that minimization of this loss formulation indeed causes g^J_ψ to match the optimal guidance score g^J of (6.11). To see this, observe that by standard DSM-score-equivalence, this loss causes s^π_ψ to match $\nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t);\mathbf{s})$ which in turn decomposes as derived in (6.11). Therefore,

$$s^\pi_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t)) = s^p_\theta(\mathbf{a}(t);\sigma(t),\mathbf{s}) + g^J_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t)) \tag{6.42a}$$

$$\approx \nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t);\mathbf{s}) = \underbrace{\nabla_{\mathbf{a}(t)} \log p_{\theta\sigma(t)}(\mathbf{a}(t) \mid \mathbf{s})}_{=s^p_\theta(\mathbf{a}(t);\sigma(t),\mathbf{s}) \text{ (c.f. (6.12))}} + g^J(\mathbf{a}(t);\mathbf{s},\sigma(t)), \tag{6.42b}$$

revealing that optimization of (6.40b) results in $g^J_\psi(\mathbf{a}(t);\mathbf{s},\sigma(t)) \approx g^J(\mathbf{a}(t);\mathbf{s},\sigma(t))$.

[7] We could, of course, synthesize such samples by using our online estimator from Section 6.4.2 but this data-generation strategy would come at a substantial computational cost which is precisely what we want to avoid by proposing this offline estimator.

6.B Relationship to Conventional Classifier Guidance

Background on Classifier Guidance. A useful property of diffusion models is that their generative process can be adapted at test-time by composing multiple score models. A prominent instance of this composition is classifier-based guidance [165], which adapts a diffusion model of an *unconditional* distribution $p^*(\mathbf{x})$ to instead match a new *conditional* distribution $p^{\mathrm{c}}(\mathbf{x} \mid \mathbf{y})$, where $\mathbf{y} \in \mathbb{R}^{n_y}$ denotes test-time context information not known during training. This is enabled by the identity

$$\nabla_{\mathbf{x}(t)} \log p^{\mathrm{c}}(\mathbf{x}(t) \mid \sigma(t), \mathbf{y}) = \nabla_{\mathbf{x}(t)} \log p^*_{\sigma(t)}(\mathbf{x}(t)) + \nabla_{\mathbf{x}(t)} \log p^{\mathrm{c}}_{\sigma(t)}(\mathbf{y} \mid \mathbf{x}(t)) \\ \cancelto{0}{- \nabla_{\mathbf{x}(t)} \log p^{\mathrm{c}}_{\sigma(t)}(\mathbf{y})}, \tag{6.43}$$

where $p^{\mathrm{c}}_{\sigma(t)}(\mathbf{y} \mid \mathbf{x}(t)) \coloneqq \mathbb{E}_{\mathbf{x} \sim p^*(\mathbf{x} \mid \mathbf{x}(t), \sigma(t))} \left[p^{\mathrm{c}}(\mathbf{y} \mid \mathbf{x}) \right]$ takes the interpretation of a noise-conditioned classifier, assessing the (unnormalized) log-likelihood of the context $\mathbf{y}$ given the *noisy* sample $\mathbf{x}(t)$ at noise level $\sigma(t)$. Thus, given an unconditional score model $s^{p^*}_\theta(\mathbf{x}(t); \sigma(t)) \approx \nabla_{\mathbf{x}} \log p^*_{\sigma(t)}(\mathbf{x}(t); \sigma(t))$, we can obtain conditional samples from $p^{\mathrm{c}}(\mathbf{x} \mid \mathbf{y})$ by adding the classifier score $\nabla_{\mathbf{x}(t)} \log p^{\mathrm{c}}_{\sigma(t)}(\mathbf{y} \mid \mathbf{x}(t))$ to $s^{p^*}_\theta$ in the reverse SDE of (6.5).

Recognizing g^J as a classifier score. When we interpret $\exp\left(-J(\mathbf{a}; \mathbf{s})/\lambda\right)$ as a classifier $p^{\mathrm{c}}(\mathbf{y} \mid \mathbf{a}, \mathbf{s})$, we can rewrite $g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))$ as the gradient of a noise-conditioned log classifier

$$g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t)) \coloneqq \nabla_{\mathbf{a}(t)} \log \mathop{\mathbb{E}}_{\mathbf{a} \sim p_\theta(\mathbf{a} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \left[\overbrace{\exp\left(\frac{-J(\mathbf{a}; \mathbf{s})}{\lambda} \right)}^{p^{\mathrm{c}}(\mathbf{y} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))} \right] \tag{6.44}$$

$$= \nabla_{\mathbf{a}} \log p^{\mathrm{c}}(\mathbf{y} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t)) \tag{6.45}$$

making (6.11) reminiscent of the vanilla classifier guidance framework (c.f. (6.43)) with the composed score

$$\nabla_{\mathbf{a}(t)} \log \pi_{\sigma(t)}(\mathbf{a}(t); \mathbf{s}) = \nabla_{\mathbf{a}(t)} \log p_{\theta_{\sigma(t)}}(\mathbf{a}(t) \mid \mathbf{s}) + \underbrace{\nabla_{\mathbf{a}(t)} \log p^{\mathrm{c}}(\mathbf{y} \mid \mathbf{a}(t), \mathbf{s}, \sigma(t))}_{g^J(\mathbf{a}(t); \mathbf{s}, \sigma(t))}.$$

6.C Cost Function Implementation Details

We use the following cost function structure, repeated here for convenience:

$$J(\mathbf{s}, \mathbf{a}) = w_{\text{goal}} J_{\text{goal}}(\mathbf{s}, \mathbf{a}) + w_{\text{collision}} J_{\text{collision}}(\mathbf{s}, \mathbf{a}) + w_{\text{engage}} J_{\text{engage}}(\mathbf{s}, \mathbf{a}). \tag{6.46}$$

Below, we provide the implementation details for this cost structure. Table 6.1 lists the values we use for any parameters referenced below.

Notation. Let $\tau : \{1, \dots, K\} \to \mathcal{S}$ denote the joint state trajectory resulting from applying the action-sequence $\mathbf{a}$ from state $\mathbf{s}$ over the receding-horizon window of length K

so that $\tau(k) = \mathbf{s}_k$ denotes the joint state at future time step k. Let $p_k^{(i)} \in \mathbb{R}^3$ denote the position of the ith end-effector at time step k in the workspace and $p_k^{\text{object}} \in \mathbb{R}^3$ denote the position of the object at time step k. Let $d^{\text{ws}}(p^{\text{a}}, p^{\text{b}})$ denote the Euclidean distance between two points $p^{\text{a}} \in \mathbb{R}^3$ and $p^{\text{b}} \in \mathbb{R}^3$ in the workspace. Let $d_{\text{t2p}}(\tilde{\tau}, \tilde{p})$ denote the minimum distance between any point along the trajectory $\tilde{\tau}$ and the point $\tilde{p}$, i.e.

$$d_{\text{t2p}}(\tilde{\tau}, \tilde{p}) = \min_{k=1,\dots,K} \left\| p_k^{(i)} - \tilde{p} \right\|, \tag{6.47}$$

and analogously for a pair of trajectories

$$d_{\text{t2t}}(\tilde{\tau}^{\text{a}}, \tilde{\tau}^{\text{b}}) = \min_{k=1,\dots,K} \left\| p_k^{(i)} - p_k^{(j)} \right\|, \tag{6.48}$$

Finally, let p_{goal} denote the goal position and p_{object} denote the position of the object associated with state $\mathbf{s}$. The cost components are defined as follows.

Goal Cost. The goal cost component penalizes the minimum distance of the object to the goal along the receding-horizon trajectory

$$J_{\text{goal}}(\mathbf{s}, \mathbf{a}) = \min\{d_{\text{t2p}}(\tau^{\text{object}}, p_{\text{goal}}) - \Delta_{\text{goal}}, 0\}. \tag{6.49}$$

where Δ_{goal} denotes the threshold at which this penalty is applied.

Collision-Avoidance Cost. The collision-avoidance cost component penalizes the largest safety-distance violation between both end-effectors via

$$J_{\text{collision}}(\mathbf{s}, \mathbf{a}) = \min\{\Delta_{\text{collision}} - d_{\text{t2t}}(\tau^{(1)}, \tau^{(2)}), 0\}, \tag{6.50}$$

where $\Delta_{\text{collision}}$ denotes the threshold at which this penalty is applied.

Engagement Cost. The engagement cost encourages the closer robot to engage with the object via

$$J_{\text{engage}}(\mathbf{s}, \mathbf{a}) = \begin{cases} \min\{d_{\text{t2t}}(\tau^{(1)}, \tau^{\text{object}}) - \Delta_{\text{engage}}, 0\} & \text{if } \left\| p_1^{(1)} - p_1^{\text{object}} \right\| < \left\| p_1^{(2)} - p_1^{\text{object}} \right\| \\ \min\{d_{\text{t2t}}(\tau^{(2)}, \tau^{\text{object}}) - \Delta_{\text{engage}}, 0\} & \text{otherwise.} \end{cases} \tag{6.51}$$

Parameter	Symbol	Value
Goal penalty threshold	Δ_{goal}	1 cm
Goal reaching cost weight	w_{goal}	1
Engage penalty threshold	Δ_{engage}	20 cm
Engage cost weight	w_{engage}	10
Collision avoidance radius	$\Delta_{\text{collision}}$	30 cm
Collision avoidance cost weight	$w_{\text{collision}}$	10

Table 6.1: Cost function parameters.

Chapter 7

Conclusions and Future Work

This chapter discusses the main takeaways of this dissertation and presents directions for future research.

7.1 Conclusions

This dissertation advances robotic motion planning in scenarios where a robot interacts with other agents—i.e., with humans or other robots. By studying this problem through the lens of game theory, this dissertation developed algorithms that enable a robot to estimate the intents of others (Chapter 2), adapt its strategic plans to these estimates (Chapter 3), account for uncertainty (Chapter 4), and compute its own strategies efficiently (Chapter 5). A common thread across these chapters was the reliance on the assumption that the underlying problem data—i.e., the costs and constraints (including dynamics)—are *smooth* to facilitate iterative search for local equilibria. While this assumption holds for many applications, such as autonomous driving, it becomes limiting in settings where non-smoothness is inherent, such as multi-agent manipulation. Chapter 6 adopts a distinct approach, leveraging probabilistic inference and generative modeling to address complex, non-smooth interactions by combining learning from demonstrations with reasoning about joint costs across agents. Below, we summarize the key conclusions of Chapters 2 to 6.

7.1.1 Estimating Player Objectives from Partially Observed Interactions

Chapter 2 studied the problem of inferring other agents' objectives from observations of their past behavior—a problem commonly referred to as the *inverse* game problem. The central contribution of this chapter was a maximum-likelihood formulation of this inverse game problem. The key innovation of this formulation was that it captured scenarios characterized by partial and noise-corrupted observations. In contrast, existing methods assumed full observability of all agents' states and could only handle partial observations when combined with a separate state estimation preprocessing step. After formalizing the problem, we demonstrated how to transcribe the proposed partially observed inverse

games into standard constrained optimization problems that can be solved using off-the-shelf optimization solvers. The resulting approach estimated players' cost parameters while simultaneously recovering the forward game trajectory consistent with those parameters, with overall performance measured according to observation likelihood.

To validate this approach, we conducted extensive numerical simulations across multiple autonomous driving scenarios involving up to 5 vehicles, where key aspects of other vehicles' behavior—such as goal locations, target lanes, and desired travel speeds—were initially unknown to the robot. Our simulation results demonstrated several key advantages of the joint inference approach compared to prior methods that treated state estimation and objective inference as separate problems. Most notably, the results showed that this joint formulation consistently produced more accurate inference of other agents' objectives, which in turn enabled more accurate predictions of future trajectories for all players compared to the baseline approach. Finally, this joint formulation improved state estimation of the *past* trajectory, further highlighting the benefits of coupling inference and trajectory reconstruction.

To reduce the computational burden, we proposed a receding-horizon variant of the inverse game problem, in which the robot performs inference as observations become available and is tasked with updating its predictions on the fly. Similar to the offline setting, our approach showed improved robustness to noise and partial observations. Furthermore, we found a pronounced performance advantage over the baseline when the robot had access to relatively short observation horizons.

Despite these advances, the computational complexity of the joint formulation remained a key limitation of the methods proposed in this chapter, as they required solving challenging nonlinear optimization problems. Furthermore, the solution approach used in this chapter did not handle inequality constraints. Finally, this chapter studied intent inference in isolation, without integrating it with online decision making.

7.1.2 Learning to Interact with Agents That Have Unknown Objectives

Chapter 3 built on the inverse game problem of Chapter 2 by integrating this paradigm with online decision making. Hence, this chapter moved beyond pure *inference* to adaptive *motion planning*—simultaneously estimating other agents' objectives and responding to them in real time. To facilitate this integration, we developed a new solution approach that addressed the key limitations of the constrained optimization approach.

The key innovation facilitating these advancements was the contribution of an implicit differentiation scheme that efficiently computes gradients of the observation likelihood with respect to the game's parameters. We showed how this gradient signal can be used to estimate these game parameters online, yielding an adaptive game solver that updates the game-theoretic interaction model as observations become available. Relative to the formulation in Chapter 2, this new approach had two technical advantages: (i) it handled inequality constraints, facilitating safer interaction; and (ii) it gave rise to a simple, first-order update rule for parameters that integrates naturally with NNs training for amortized inference.

To validate this approach, we conducted extensive simulations of autonomous driving scenarios similar to those in Chapter 2, but now focused on *decision making* rather than pure *inference*. Beyond the approach from Chapter 2, we compared against a non-adaptive game-theoretic planner that assumes a prior game model and a non-game-theoretic MPC baseline that ignores interdependence of actions.

In terms of interaction performance, our simulation results showed that the implicit differentiation approach to inverse games predicted game parameters more accurately than the baselines, and thereby enabled safer and more efficient online interaction. This advantage was particularly pronounced in high-density traffic scenarios, where the complexity of multi-agent interactions made adaptive planning most beneficial.

In terms of computational efficiency, we presented two key findings. First, when instantiated as first-order online optimization, the implicit differentiation approach of this chapter was only marginally slower than an optimized implementation of the method from Chapter 2, despite solving a harder inverse problem due to the consideration of inequality constraints. And second, when instantiated as amortized inference via an offline training phase, we found that this approach was about 3 times faster than the amortized form of Chapter 2 while still having better interaction performance.

Finally, we also conducted hardware experiments that showcased the real-time planning capacity of the proposed adaptive game solver in two variants of a 2-player interaction problem with (i) another robot and (ii) a human player.

Despite these advances, the implicit differentiation approach had a key limitation: it produced only a single point estimate of other agents' objectives, thus planning as if this estimate represented the true underlying game model without accounting for uncertainty in the estimate.

7.1.3 Contingency Games: Strategic Interaction Under Uncertainty

Chapter 4 set out to address the neglect of uncertainty in the previous chapter. To this end, we formalized a new uncertainty-aware planning paradigm, termed *contingency games*, in which the robot generates plans for a *distribution* of possible game models[1].

The key innovation of contingency games was their ability to capture the fact that future belief updates will reduce uncertainty and hence, eventually, the *true* intent of others will be revealed at a future "branching time". By encoding this feature in the game formulation, equilibrium solutions of contingency games take the form of a conditional plan, allowing the robot to develop distinct interactive strategies after the branching time for each possible intent of others. We demonstrated that contingency games generalize prior approaches that either assumed full certainty [78, 102, 103] or assumed fixed uncertainty [63, 104], recovering prior approaches as special cases under extreme choices of the branching time. To formulate a middle ground between these two extremes, we proposed a simple yet effective heuristic to estimate the branching time online, enabling adaptive, uncertainty-aware game-theoretic motion planning.

[1]Such intents could come from [30], a work that is beyond the scope of this dissertation but is briefly discussed in Section 1.4.

We evaluated contingency games in simulations of several driving scenarios with up to 3 players. In these simulations, we compared against game-theoretic baselines from both extremes: certainty-equivalent games (as in Chapter 3) and fixed uncertainty games that assume uncertainty will never be resolved (as in [63, 104]).

Our simulation results demonstrated several key findings. First, we found that all methods accounting for uncertainty improved safety over those techniques that ignored uncertainty by avoiding commitment to overly optimistic plans. Among the uncertainty-aware game formulations, we found that contingency games matched the fixed uncertainty baseline in terms of safety but improved over it in terms of efficiency when using the proposed adaptive branching time estimator.

7.1.4 Amortized Equilibrium Approximation through Offline Learning

Chapter 5 set out to address a challenge observed repeatedly throughout this dissertation: the difficulty of computing game-theoretic equilibria, particularly when the strategy space is large. To study this challenge, this chapter focused on games with enlarged strategy spaces due to mixed strategies; i.e., games in which players may strategically randomize their actions to avoid exploitation by opponents.

The contribution of this chapter was a game solver that amortized the computation of equilibrium solutions by decomposing the problem into an *offline* training phase and an *online* reasoning phase. Offline, the method trained a NN to predict a finite number of low-cost trajectory candidates. Online, a discrete game solver computed a mixed Nash equilibrium over the trajectory candidates proposed by the pre-trained NN for the current game state. Two technical advances enabled this approach. First, by embedding a trajectory optimization layer in the NN architecture, this approach ensured that, despite amortization, all candidate trajectories are feasible. And second, by making both trajectory optimization and game solver differentiable, this approach ensured that the NN training could directly use the game value as supervision signal.

Extensive evaluation of this amortized approach in simulations of the pursuit-evasion game "tag" revealed the following key results. First, a comparison against a baseline that sampled a fixed number of trajectory candidates a priori revealed the utility of learning the trajectory candidates: even with just two learned trajectory candidates, the amortized approach found more competitive strategies than the sampling-based baseline with up to 20 candidates. Second, a comparison against a baseline that considers only pure strategies revealed the utility of reasoning over the enlarged mixed-strategy space: in head-to-head competition, agents utilizing mixed strategies consistently outperformed opponents that considered only the simplified pure-strategy space. Finally, we also demonstrated the efficiency of the targeted use of learning for amortization in our approach. This efficiency was highlighted by the average runtime of 2ms for online decision making and the ability to train from scratch in only a few minutes of simulated self-play, converging within approximately 1500 turns.

7.1.5 Utilizing Single-Agent Demonstrations to Learn Multi-Agent Policies

Chapter 6 shifted gears to address an assumption underpinning all of the previous chapters: the requirement that the problem data—i.e., the costs and constraints (including dynamics)—are *smooth*. While this assumption holds for many applications, such as autonomous driving, it becomes limiting in settings where non-smoothness is inherent, such as multi-agent manipulation due to contact dynamics. Furthermore, insisting on smoothness may require users to formulate artificial, smoothed versions of the costs or constraints they genuinely intend, potentially complicating task specification.

The key innovation of this chapter was a method for synthesizing a single, cohesive multi-agent policy while leveraging single-agent demonstrations. This was achieved through a multi-stage learning process: we first trained *single-agent policies* from demonstrations of basic skills (e.g., pick-and-place), and then composed these policies into *coordinated* multi-agent behavior by reasoning about the *joint cost* of actions across agents. By studying this problem through the lens of probabilistic inference and generative diffusion models [158–161], we constructed a generative process of the coordinated, *joint* policy while reusing pre-training results from single-agent learning. We also discussed how this probabilistic inference approach can be tied back to a game-theoretic perspective.

We validated our approach in high-fidelity simulations of a two-agent manipulation task, where two robots must collaborate to move an object to a specified location on a wide table. We demonstrated that our method is able to learn multi-agent policies from single-agent demonstrations, discover new collaborative strategies not present in the original data, and generate more efficient and accurate manipulation policies than a baseline that relies on multi-agent demonstrations, given the same amount of data.

7.2 Limitations & Future Work

The methods developed in this dissertation provide a foundation for several lines of future work to further advance robotic motion planning for multi-agent interaction. The preceding Chapters 2 to 6 each concluded with chapter-specific suggestions for future research. This section identifies overarching key directions for future work that extend beyond the scope of individual chapters.

A more holistic view on uncertainty. While this dissertation introduced methods for inferring unknown game parameters (Chapters 2 and 3) and planning under the resulting uncertainty (Chapter 4), important opportunities remain to reason about uncertainty in a more integrated manner—primarily along two key dimensions. First, although Chapters 2 and 3 unified intent inference and state estimation, our work treated this estimation pipeline as distinct from planning. As a result, the planners in Chapters 3 and 4 did not consider how their actions might influence future state estimation or intent inference. Second, in all studied scenarios, uncertainty was strictly one-directional: the robot was uncertain about other agents, but assumed others had perfect knowledge. Future work should address these limitations by moving toward a more comprehensive perspective of a full partially observed stochastic game (POSG)—the game-theoretic analogue of dual

control—where each agent's actions depend on the entire observation history. Such extensions would enable robots to reason not only about their own uncertainty, but also about how their actions shape the beliefs of others, supporting more effective implicit communication through informative actions. Although POSGs are notoriously challenging, there is significant potential for tractable methods that approximate belief dynamics and strategy spaces to capture the most relevant effects in a manner that exploits the structure of the motion planning problem.

Towards a mature ecosystem of modern game solvers for motion planning. For non-game-theoretic instances of motion planning—i.e., settings in which a robot acts in solitude or treats others as part of the environment rather than as strategic agents—practitioners benefit from a broad range of tools spanning three main categories, each with distinct advantages: (i) gradient-free solvers [174–176] handle non-smooth problem data, such as those arising in problems with contact, naturally exploit parallel hardware for real-time applications, and offer relatively simple user interfaces since they can operate on black-box specifications of the problem; (ii) gradient-based first- and second-order solvers [177–179] enable fast online optimization for problems with smooth dynamics, constraints, and objectives, and (iii) amortized approaches, from structured learning [180, 181] to general reinforcement learning [182–186], address otherwise intractable problems by shifting computation offline when large-scale experience is available. This rich toolbox in the single-agent domain enables practitioners to select methods tailored to their application's needs. While approaches in each of these categories exist also for game-theoretic problems [46, 66, 80, 187–194], options in that domain are far more limited. For example, while in single-agent tasks with non-smooth dynamics, methods like Model Predictive Path Integral Control (MPPI) [175] are well established [195–197], no comparable standard exists for games. Similarly, while recent work has put forth robust amortized approaches such as Dreamer [198–201] or TD-MPC [202, 203] that provide general-purpose reinforcement learning solutions in the single-agent domain with little tuning effort, and algorithms like Proximal Policy Optimization (PPO) [182] are the de facto standard for quickly learning new skills in the single-agent domain with well-explored best practices and guidelines for practitioners [204], learning-based methods are much less explored in the game-theoretic domain and require extensive problem-specific tuning, in particular for general-sum games. To make game-theoretic planning broadly accessible, future research should seek to advance game solvers in each of these categories, aiming for a robust ecosystem of "plug-and-play" tools. Beyond considerations of simple user interfaces, these solvers should be designed from the ground up with modern hardware in mind, particularly targeting GPU acceleration for massive parallelization.

Model-order reduction. This dissertation includes games involving up to seven players. In real-world scenarios, however, there can often be many more players, posing a computational challenge. Interestingly, cognitive psychology consistently shows that humans can attend to no more than five dynamic targets at once [205–207]. Motivated by these findings, future research in game-theoretic motion planning should focus on principled model-order reduction techniques that distill many-player games into manageable

representations, for example, by aggregating agents into macro-entities or pruning those whose influence on the ego agent's strategy is marginal.

Interaction beyond collision avoidance. The majority of this dissertation (Chapters 2 to 5) focused on multi-agent scenarios in which collision avoidance was the primary source of interaction. Yet, many real-world applications—such as autonomous assembly, construction, and household robotics—require richer forms of interaction that go well beyond mere collision avoidance, involving tightly coupled collaboration, physical contact, or shared manipulation tasks. Such interactions are often governed by complex, sometimes non-smooth dynamics, and they typically involve high-dimensional states that must be inferred indirectly from rich sensory data such as RGB images or LiDAR point clouds. Chapter 6 took an step beyond pure collision avoidance by studying a manipulation task in which multiple agents must actively coordinate to achieve a joint goal. However, this chapter made two key simplifying assumptions: all agents were required to share a joint cost function (thus ruling out tasks with elements of competition), and the state was assumed to be directly observed and low-dimensional rather than operating directly on high-dimensional sensory inputs.

In the single-agent domain, the emergence of world models pretrained on internet-scale data [201] has shown promise for enabling robot control directly from these rich sensory inputs. These methods jointly learn a state representation and state dynamics alongside a corresponding state estimator, thereby enabling decision-making in a latent space. Multi-agent extensions of these approaches have begun to emerge [208–212], but they remain largely confined to fully cooperative settings where all agents share a common objective [210–212]. A notable exception is [209], which supports distinct cost functions for each player to capture non-cooperative settings. Notwithstanding, this work has been demonstrated only in a simple two-player simulation with shared observations and zero-sum cost structure, and it requires training from scratch on large-scale multi-agent experience.

Future work should seek to build more flexible multi-agent world models that are capable of capturing the full complexity of non-cooperative partially observable general-sum games. Two concrete directions appear most promising in this regard. First, future work should develop new model architectures that capture arbitrary individual costs per player, accommodate distinct observations and beliefs while building on the latest architectural advances in the single-agent domain [201, 203, 213, 214]. Second, to facilitate cost-effective training, this new generation of multi-agent world models should leverage single-agent data exhaustively, since this data is far more abundant than multi-agent data. Here, a key challenge is distilling and transferring knowledge from single-agent data to multi-agent settings. Beyond learning transferable data representations, this requires developing techniques to compensate for the lack of interaction in these single-agent data sources. Here, one promising approach is to apply game-theoretic reasoning to single-agent policies to synthesize new interaction-aware data, building on the initial results of Chapter 6.

When these efforts to build general multi-agent world models succeed, such models can serve as versatile "adapters" that connect game-theoretic tools to complex real-world interactions in a data-efficient manner. Future work should investigate specialized game solvers that harness these models' latent spaces—including non-cooperative multi-agent

reinforcement learning (MARL), online equilibrium-seeking approaches, and hybrids of offline learning with online reasoning [202, 203, 215–217]—all while exploiting the massively parallelizable evaluation of these world models.

Bibliography

References

[1] Anthropic. Claude 4.6 opus. Software, 2 2026. URL https://www.anthropic.com/news/claude-opus-4-6. Large language model.

[2] Jonathan Tilley. Automation, robotics, and the factory of the future. *McKinsey & Company*, 67(1):67–72, 2017.

[3] Daniel Zhang, Nestor Maslej, Erik Brynjolfsson, John Etchemendy, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Michael Sellitto, Ellie Sakhaee, et al. The ai index 2022 annual report. *arXiv preprint arXiv:2205.03468*, 2022.

[4] Yuchuang Tong, Haotian Liu, and Zhengtao Zhang. Advancements in humanoid robots: A comprehensive review and future prospects. *IEEE/CAA Journal of Automatica Sinica*, 11(2):301–328, 2024.

[5] International Federation of Robotics. 2.1 million domestic floor cleaning robots sold in 2023, 2025. URL https://ifr.org/post/21-million-domestic-floor-cleaning-robots-sold-in-2023. Accessed: 2025-01-15.

[6] Juan Angel Gonzalez-Aguirre, Ricardo Osorio-Oliveros, Karen L Rodríguez-Hernández, Javier Lizárraga-Iturralde, Ruben Morales Menendez, Ricardo A Ramirez-Mendoza, Mauricio Adolfo Ramirez-Moreno, and Jorge de Jesus Lozoya-Santos. Service robots: Trends and technology. *Applied Sciences*, 11(22):10702, 2021.

[7] Gwyn Topham. Driverless taxis from waymo will be on london's roads next year, us firm announces, 2025. URL https://www.theguardian.com/technology/2025/oct/15/driverless-taxis-from-waymo-will-be-on-londons-roads-next-year-us-firm-announces. Accessed: 2025-01-15.

[8] Daron Acemoglu and Pascual Restrepo. Demographics and automation. *The Review of Economic Studies*, 89(1):1–44, 2022.

[9] Berker Bilgin, Jianbin Liang, Mladen V Terzic, Jianning Dong, Romina Rodriguez, Elizabeth Trickett, and Ali Emadi. Modeling and analysis of electric motors: State-of-the-art review. *IEEE Transactions on Transportation Electrification*, 5(3):602–617, 2019.

[10] Thomas Gillespie. *Fundamentals of vehicle dynamics*. SAE international, 2021.

[11] Ying He, Dao Bo Wang, and Zain Anwar Ali. A review of different designs and control models of remotely operated underwater vehicle. *Measurement and Control*, 53(9-10):1561–1570, 2020.

[12] Leonard Bauersfeld, Elia Kaufmann, Philipp Foehn, Sihao Sun, and Davide Scaramuzza. Neurobem: Hybrid aerodynamic quadrotor model. In *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[13] Tom Driessen, Dimitra Dodou, Pavlo Bazilinskyy, and Joost De Winter. Putting chatgpt vision (gpt-4v) to the test: risk perception in traffic images. *Royal Society open science*, 11(5):231676, 2024.

[14] Illustration of a multi-agent interaction scenario, 2025. Generated with Google Gemini, `https://gemini.google.com/app`, Accessed: 2025-12-06.

[15] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):187–210, 2018.

[16] David Fridovich-Keil. Smooth game theory, 2024.

[17] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

[18] Laura Ferranti, Lorenzo Lyons, Rudy R Negenborn, Tamás Keviczky, and Javier Alonso-Mora. Distributed nonlinear trajectory optimization for multi-robot motion planning. *IEEE Trans. on Robotics (TRO)*, 31(2):809–824, 2022.

[19] Alessandro Zanardi, Saverio Bolognani, Andrea Censi, and Emilio Frazzoli. Game theoretical motion planning: Tutorial ICRA 2021, 2021.

[20] Rudolf E. Kalman. When Is a Linear Control System Optimal? *ASME Journal of Basic Engineering*, 86(1):51–60, 1964.

[21] Sebastian Albrecht, Karinne Ramirez-Amaro, Federico Ruiz-Ugalde, David Weikersdorfer, Marion Leibold, Michael Ulbrich, and Michael Beetz. Imitating human reaching motions using physically inspired optimization principles. In *Proc. of the IEEE Intl. Conf. on Humanoid Robots*. IEEE, 2011.

[22] Peter Englert and Marc Toussaint. Inverse KKT: Learning cost functions of manipulation tasks from demonstrations. *Intl. Journal of Robotics Research (IJRR)*, pages 57–72, 2018.

[23] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Proc. of Robotics: Science and Systems (RSS)*, 2023.

[24] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Intl. Journal of Robotics Research (IJRR)*, 44(10-11):1684–1704, 2025.

[25] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint*, 2022. arXiv:2205.09991.

[26] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *Proc. of Robotics: Science and Systems (RSS)*, 2023.

[27] Marcel Menner and Melanie N. Zeilinger. Maximum likelihood methods for inverse learning of optimal controllers. *arXiv preprint arXiv:2005.02767*, 2020.

[28] Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *Proc. of the Intl. Symp. on Intelligent Control (ISIC)*. IEEE, 2011.

[29] Jingqi Li, Chih-Yuan Chiu, Lasse Peters, Somayeh Sojoudi, Claire J Tomlin, and David Fridovich-Keil. Cost inference for feedback dynamic games from noisy partial state observations and incomplete trajectories. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2023.

[30] Lasse Peters, Xinjie Liu, Javier Alonso-Mora, Ufuk Topcu, and David Fridovich-Keil. Auto-encoding bayesian inverse games. In *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2024.

[31] Jingqi Li, Chih-Yuan Chiu, Lasse Peters, Fernando Palafox, Mustafa Karabag, Javier Alonso-Mora, Somayeh Sojoudi, Claire Tomlin, and David Fridovich-Keil. Scenario-game admm: A parallelized scenario-based solver for stochastic noncooperative games. *Proceedings of the Conference on Decision Making and Control (CDC)*, 2023.

[32] Leonardo Santos, Zirui Li, Lasse Peters, Somil Bansal, and Andrea Bajcsy. Updating robot safety representations online from natural language feedback. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2025.

[33] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. In *Proc. of Robotics: Science and Systems (RSS)*, 2025.

[35] Lasse Peters, David Fridovich-Keil, and Forrest Laine. TrajectoryGamesBase.jl, 2022. URL `https://github.com/JuliaGameTheoreticPlanning/TrajectoryGamesBase.jl`. Accessed: 2025-12-06.

[36] Lasse Peters. ParametricMCPs.jl, 2022. URL `https://github.com/JuliaGameTheoreticPlanning/ParametricMCPs.jl`. Accessed: 2025-12-06.

[37] Steven P Dirkse and Michael C Ferris. The Path Solver: a Non-monotone Stabilization Scheme for Mixed Complementarity Problems. *Optimization Methods and Software*, 1995.

[38] Lasse Peters. MCPTrajectoryGameSolver.jl, 2022. URL `https://github.com/JuliaGameTheoreticPlanning/MCPTrajectoryGameSolver.jl`. Accessed: 2025-12-06.

[39] Lasse Peters, David Fridovich-Keil, and Forrest Laine. DifferentiableTrajectoryOptimization.jl, 2022. URL `https://github.com/lassepe/DifferentiableTrajectoryOptimization.jl`. Accessed: 2025-12-06.

[40] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[41] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[42] Lasse Peters, Vicenc Rubies-Royo, Claire J. Tomlin, Laura Ferranti, Javier Alonso-Mora, Cyrill Stachniss, and David Fridovich-Keil. Online and offline learning of player objectives from partial observations in dynamic games. In *Intl. Journal of Robotics Research (IJRR)*, 2023.

[43] Lasse Peters, David Fridovich-Keil, Vicenc Rubies-Royo, Claire J. Tomlin, and Cyrill Stachniss. Inferring objectives in continuous dynamic games from noise-corrupted partial state observations. In *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[44] Rufus Isaacs. Differential games i-iv. Technical report, RAND CORP SANTA MONICA CA SANTA MONICA, 1954-1955.

[45] Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*. Society for Industrial and Applied Mathematics (SIAM), 2. edition, 1998.

[46] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D. Dragan, and Claire J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2020.

[47] Bolei Di and Andrew Lamperski. Newton's method and differential dynamic programming for unconstrained nonlinear dynamic games. In *Proceedings of the Conference on Decision Making and Control (CDC)*, 2019.

[48] Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots*, 28(3):369–383, 2010.

[49] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2000.

[50] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, 2008.

[51] Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. Inverse reinforcement learning in swarm systems. *arXiv preprint arXiv:1602.05450*, 2016.

[52] Sriraam Natarajan, Gautam Kunapuli, Kshitij Judah, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. Multi-agent inverse reinforcement learning. In *2010 ninth international conference on machine learning and applications*, pages 395–400. IEEE, 2010.

[53] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[54] Simon Rothfuß, Jairo Inga, Florian Köpf, Michael Flad, and Sören Hohmann. Inverse optimal control for identification in non-cooperative differential games. *IFAC-PapersOnLine*, 50(1):14909–14915, 2017.

[55] Jairo Inga, Esther Bischoff, Florian Köpf, and Sören Hohmann. Inverse dynamic games based on maximum entropy inverse reinforcement learning. *arXiv preprint arXiv:1911.07503*, 2019.

[56] Chaitanya Awasthi and Andrew Lamperski. Inverse differential games with mixed inequality constraints. In *Proc. of the IEEE American Control Conference (ACC)*, 2020.

[57] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *Intl. Journal of Robotics Research (IJRR)*, 35(11):1289–1307, 2016.

[58] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer Verlag, 2. edition, 2006.

[59] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2012.

[60] Chaitanya Awasthi. Forward and inverse methods in optimal control and dynamic game theory. Master's thesis, University of Minnesota, 2019.

[61] Wanxin Jin, Dana Kulić, Shaoshuai Mou, and Sandra Hirche. Inverse optimal control from incomplete trajectory observations. *Intl. Journal of Robotics Research (IJRR)*, 40 (6-7):848–865, 2021.

[62] Lasse Peters. Accommodating intention uncertainty in general-sum games for human-robot interaction. Master's thesis, Hamburg University of Technology, 2020.

[63] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. LUCIDGames: Online unscented inverse dynamic games for adaptive trajectory prediction and planning. *IEEE Robotics and Automation Letters (RA-L)*, 6(3):5485–5492, 2021.

[64] Florian Köpf, Jairo Inga, Simon Rothfuß, Michael Flad, and Sören Hohmann. Inverse reinforcement learning for identification in linear-quadratic dynamic games. *IFAC-PapersOnLine*, 50(1):14902–14908, 2017.

[65] Zijian Wang, Riccardo Spica, and Mac Schwager. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pages 225–238. Springer Verlag, 2019.

[66] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. ALGAMES: A fast solver for constrained dynamic games. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[67] Forrest Laine, David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin. The computation of approximate generalized feedback nash equilibria. *SIAM Journal on Optimization*, 33(1):294–318, 2023.

[68] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1): 195–259, 2009.

[69] Zhi-Quan Luo, Jong-Shi Pang, and Daniel Ralph. *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[70] Michael C Ferris, Steven P Dirkse, and Alexander Meeraus. Mathematical programs with equilibrium constraints: Automatic reformulation and solution via constrained optimization. *Frontiers in applied general equilibrium modeling*, pages 67–93, 2005.

[71] Robert G Gallager. *Stochastic processes: theory for applications*. Cambridge University Press, 2013.

[72] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[73] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review (SIREV)*, 59(2):295–320, 2017.

[74] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review (SIREV)*, 47:99–131, 2005.

[75] Mustafa Mukadam, Jing Dong, Frank Dellaert, and Byron Boots. Steap: simultaneous trajectory estimation and planning. *Autonomous Robots*, 43(2):415–434, 2019.

[76] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review (SIREV)*, 59(1):65–98, 2017.

[77] Bolei Di and Andrew Lamperski. Newton's method, Bellman recursion and differential dynamic programming for unconstrained nonlinear dynamic games. *Dynamic Games and Applications*, pages 1–49, 2021.

[78] Lasse Peters, Xinjie Liu, and Javier Alonso-Mora. Learning to play trajectory games against opponents with unknown objectives. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[79] Lasse Peters, David Fridovich-Keil, Claire J Tomlin, and Zachary N Sunberg. Inferring objectives from demonstrations with unknown rewards. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2021.

[80] Lasse Peters, David Fridovich-Keil, Laura Ferranti, Cyrill Stachniss, Javier Alonso-Mora, and Forrest Laine. Learning mixed strategies in trajectory games. In *Proc. of Robotics: Science and Systems (RSS)*, 2022.

[81] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[82] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. ALGAMES: a fast augmented lagrangian solver for constrained dynamic games. *Autonomous Robots*, 2022.

[83] Alexander Liniger and John Lygeros. A noncooperative game approach to autonomous racing. *IEEE Trans. on Control Systems Technology (TCST)*, 28(3):884–897, 2019.

[84] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, 2010.

[85] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional Variational Inequalities and Complementarity Problems*, volume 1. Springer Verlag, 2003.

[86] Philipp Geiger and Christoph-Nikolas Straehle. Learning game-theoretic models of multiagent trajectories using implicit layers. In *Proc. of the Conference on Advancements of Artificial Intelligence (AAAI)*, volume 35, 2021.

[87] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[88] Bruno Brito, Michael Everett, Jonathan P How, and Javier Alonso-Mora. Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments. *IEEE Robotics and Automation Letters (RA-L)*, 6(3):4616–4623, 2021.

[89] Varun Tolani, Somil Bansal, Aleksandra Faust, and Claire Tomlin. Visual navigation among humans with optimal control as a supervisor. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2288–2295, 2021.

[90] Edward Schmerling, Karen Leung, Wolf Vollprecht, and Marco Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[91] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[92] Junha Roh, Christoforos Mavrogiannis, Rishabh Madan, Dieter Fox, and Siddhartha Srinivasa. Multimodal trajectory prediction via topological invariance for navigation at uncontrolled intersections. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

[93] Muchen Sun, Francesca Baldini, Peter Trautman, and Todd Murphey. Move beyond trajectories: Distribution space coupling for crowd navigation. *Proc. of Robotics: Science and Systems (RSS)*, 2021.

[94] Christoph Schöller, Vincent Aravantinos, Florian Lay, and Alois Knoll. What the constant velocity model can teach us about pedestrian motion prediction. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1696–1703, 2020.

[95] Daniel Ralph and Stephan Dempe. Directional derivatives of the solution of a parametric nonlinear program. *Mathematical Programming*, 70(1):159–172, 1995.

[96] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2017.

[97] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[98] Stephen C Billups, Steven P Dirkse, and Michael C Ferris. A comparison of large scale mixed complementarity problem solvers. *Computational Optimization and Applications*, 7(1):3–25, 1997.

[99] Ankur A Kulkarni and Uday V Shanbhag. On the variational equilibrium as a refinement of the generalized nash equilibrium. *Automatica*, 48(1):45–55, 2012.

[100] Yutaka Kanayama, Yoshihiko Kimura, Fumio Miyazaki, and Tetsuo Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 1990.

[101] Lasse Peters, Andrea Bajcsy, Chih-Yuan Chiu, David Fridovich-Keil, Forrest Laine, Laura Ferranti, and Javier Alonso-Mora. Contingency games for multi-agent interaction. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[102] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Proc. of Robotics: Science and Systems (RSS)*, 2016.

[103] Negar Mehr, Mingyu Wang, Maulik Bhatt, and Mac Schwager. Maximum-entropy multi-agent dynamic games: Forward and inverse solutions. *IEEE Trans. on Robotics (TRO)*, 2023.

[104] Forrest Laine, David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin. Multi-Hypothesis Interactions in Game-Theoretic Motion Planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8016–8023, 2021. doi: 10.1109/ICRA48506.2021.9561695.

[105] Ran Tian, Liting Sun, Andrea Bajcsy, Masayoshi Tomizuka, and Anca D Dragan. Safety assurances for human-robot interaction via confidence-aware game-theoretic human models. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[106] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[107] Oswin So, Paul Drews, Thomas Balch, Velin Dimitrov, Guy Rosman, and Evangelos A Theodorou. MPOGames: Efficient multimodal partially observable dynamic games. *arXiv preprint arXiv:2210.10814*, 2022.

[108] Riccardo Spica, Eric Cristofalo, Zijian Wang, Eduardo Montijano, and Mac Schwager. A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Trans. on Robotics (TRO)*, 2020.

[109] Selma Musić and Sandra Hirche. Haptic shared control for human-robot collaboration: a game-theoretical approach. *IFAC-PapersOnLine*, 2020.

[110] Edward L Zhu and Francesco Borrelli. A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria. *arXiv preprint arXiv:2203.16478*, 2022.

[111] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 2002.

[112] Judy Goldsmith and Martin Mundhenk. Competition adds complexity. *Advances in Neural Information Processing Systems (NeurIPS)*, 2007.

[113] Wilko Schwarting, Alyssa Pierson, Sertac Karaman, and Daniela Rus. Stochastic dynamic games in belief space. *IEEE Trans. on Robotics (TRO)*, 2021.

[114] Wenshuo Wang, Letian Wang, Chengyuan Zhang, Changliu Liu, Lijun Sun, et al. Social interactions for autonomous driving: A review and perspectives. *Foundations and Trends in Robotics*, 10(3-4):198–376, 2022.

[115] Jason Hardy and Mark Campbell. Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Trans. on Robotics (TRO)*, 2013.

[116] Wei Zhan, Changliu Liu, Ching-Yao Chan, and Masayoshi Tomizuka. A non-conservatively defensive strategy for urban autonomous driving. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, 2016.

[117] Yuxiao Chen, Ugo Rosolia, Wyatt Ubellacker, Noel Csomay-Shanklin, and Aaron D Ames. Interactive multi-modal motion planning with branch model predictive control. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):5365–5372, 2022.

[118] Siddharth H Nair, Vijay Govindarajan, Theresa Lin, Chris Meissen, H Eric Tseng, and Francesco Borrelli. Stochastic mpc with multi-modal predictions for traffic intersections. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, pages 635–640, 2022.

[119] Alexander Cui, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun. LookOut: Diverse multi-future prediction and planning for self-driving. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021.

[120] Ekaterina Tolstaya, Reza Mahjourian, Carlton Downey, Balakrishnan Vadarajan, Benjamin Sapp, and Dragomir Anguelov. Identifying driver interactions via conditional behavior prediction. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[121] Yuxiao Chen, Peter Karkus, Boris Ivanovic, Xinshuo Weng, and Marco Pavone. Tree-structured policy planning with learned behavior models. *arXiv preprint arXiv:2301.11902*, 2023.

[122] Charles Packer, Nicholas Rhinehart, Rowan Thomas McAllister, Matthew A Wright, Xin Wang, Jeff He, Sergey Levine, and Joseph E Gonzalez. Is anyone there? learning a planner contingent on perceptual uncertainty. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2022.

[123] Nicholas Rhinehart, Jeff He, Charles Packer, Matthew A Wright, Rowan McAllister, Joseph E Gonzalez, and Sergey Levine. Contingencies from observations: Tractable contingency planning with learned behavior models. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[124] Ariel Rubinstein. *Modeling bounded rationality*. MIT Press, 1998.

[125] Andrea Bajcsy, Anand Siththaranjan, Claire J Tomlin, and Anca D Dragan. Analyzing human models that adapt online. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[126] Makram Chahine, Roya Firoozi, Wei Xiao, Mac Schwager, and Daniela Rus. Local non-cooperative games with principled player selection for scalable motion planning. *arXiv preprint arXiv:2310.12958*, 2023.

[127] Dario Paccagnan and Marco C Campi. The scenario approach meets uncertain game theory and variational inequalities. In *Proceedings of the Conference on Decision Making and Control (CDC)*, pages 6124–6129, 2019.

[128] Tamer Başar and Pierre Bernhard. *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media, 2008.

[129] Daniel Liberzon. *Calculus of variations and optimal control theory*. Princeton University Press, 2011.

[130] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[131] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proc. of the Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, pages 222–229, 2004.

[132] Weiwei Li and Emanuel Todorov. Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. *Intl. Journal of Control*, 80(9):1439–1453, 2007.

[133] Emanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proc. of the IEEE American Control Conference (ACC)*, pages 300–306. IEEE, 2005.

[134] David H Jacobson and David Q Mayne. *Differential dynamic programming*. Modern analytic and computational methods in science and mathematics. Elsevier, 1970.

[135] Zhaoming Xie, C Karen Liu, and Kris Hauser. Differential dynamic programming with nonlinear constraints. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 695–702, 2017.

[136] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1168–1175, 2014.

[137] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[138] Alan Wilbor Starr and Yu-Chi Ho. Further properties of nonzero-sum differential games. *Journal of Optimization Theory and Applications*, 3(4):207–219, 1969.

[139] Alan Wilbor Starr and Yu-Chi Ho. Nonzero-sum differential games. *Journal of Optimization Theory and Applications*, 3(3):184–206, 1969.

[140] Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

[141] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. Generalized nash equilibrium problems and newton methods. *Mathematical Programming*, 117(1-2):163–194, 2009.

[142] Mingyu Wang, Zijian Wang, John Talbot, J Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Trans. on Robotics (TRO)*, 2021.

[143] Oussama Khatib, Sean Quinlan, and David Williams. Robot planning and control. *Journal on Robotics and Autonomous Systems (RAS)*, 21(3):249–261, 1997.

[144] Efe Camci and Erdal Kayacan. Learning motion primitives for planning swift maneuvers of quadrotor. *Autonomous Robots*, 43(7):1733–1745, 2019.

[145] Freek Stulp, Evangelos Theodorou, Mrinal Kalakrishnan, Peter Pastor, Ludovic Righetti, and Stefan Schaal. Learning motion primitive goals for robust manipulation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 325–331. IEEE, 2011.

[146] Brandon Amos. *Differentiable optimization-based modeling for machine learning*. PhD thesis, Carnegie Mellon University, 2019.

[147] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[148] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[149] Noah D Stein, Asuman Ozdaglar, and Pablo A Parrilo. Separable and low-rank continuous games. *International Journal of Game Theory*, 37(4):475–504, 2008.

[150] I Glicksberg and Oliver Gross. 9. notes on games over the square. In *Contributions to the Theory of Games (AM-28), Volume II*, pages 173–182. Princeton University Press, 2016.

[151] Christos H Papadimitriou and Tim Roughgarden. Computing equilibria in multiplayer games. In *SODA*, volume 5, pages 82–91, 2005.

[152] Forrest Laine. TensorGames, 2022. URL `https://github.com/4estlaine/TensorGames.jl`.

[153] Carlton E Lemke and Joseph T Howson, Jr. Equilibrium points of bimatrix games. *Society for Industrial and Applied Mathematics (SIAM)*, 12(2):413–423, 1964.

[154] Michael Innes. Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951*, 2018.

[155] Katta G Murty and Feng-Tien Yu. *Linear complementarity, linear and nonlinear programming*, volume 3. Citeseer, 1988.

[156] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

[157] Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint*, 2025. arXiv:2510.21890.

[158] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 2256–2265. Proceedings of Machine Learning Research, 2015.

[159] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[160] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2021.

[161] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[162] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[163] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[164] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[165] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[166] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on pure and applied mathematics*, 28(1):1–47, 1975.

[167] Emanuel Todorov. Linearly-solvable markov decision problems. *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

[168] Pierre H Richemond and Brendan Maginnis. A short variational proof of equivalence between policy gradients and soft q learning. *arXiv preprint arXiv:1712.08650*, 2017.

[169] Luis E Ortiz, Robert E Schapire, and Sham M Kakade. Maximum entropy correlated equilibria. In *Artificial Intelligence and Statistics*, pages 347–354. PMLR, 2007.

[170] Herbert E Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer Verlag, 1992.

[171] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[172] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Soft diffusion actor-critic: Efficient online reinforcement learning for diffusion policy. *arXiv preprint*, pages arXiv–2502, 2025.

[173] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[174] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2011.

[175] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[176] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2022.

[177] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2009.

[178] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Intl. Journal of Robotics Research (IJRR)*, 33(9):1251–1270, 2014.

[179] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2012.

[180] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Trans. on Automatic Control (TAC)*, 63 (7):1883–1896, 2017.

[181] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3363–3370, 2019.

[182] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[183] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, and Marcin Michalski. What matters in on-policy reinforcement learning? a large-scale empirical study. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2021.

[184] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2018.

[185] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2016.

[186] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2018.

[187] Pravesh Koirala and Forrest Laine. Monte carlo optimization for solving multilevel stackelberg games. *arXiv preprint arXiv:2312.03282*, 2023.

[188] Yuanhanqing Huang and Jianghai Hu. Zeroth-order learning in continuous games via residual pseudogradient estimates. *IEEE Transactions on Automatic Control*, 2024.

[189] Mattia Bianchi, Giuseppe Belgioioso, and Sergio Grammatico. Fast generalized nash equilibrium seeking under partial-decision information. *Automatica*, 136:110080, 2022.

[190] Carlo Cenedese, Giuseppe Belgioioso, Sergio Grammatico, and Ming Cao. An asynchronous distributed and scalable generalized nash equilibrium seeking algorithm for strongly monotone games. *European Journal of Control*, 58:143–151, 2021.

[191] Eric Mazumdar, Lillian J Ratliff, and S Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

[192] Amélie Héliou, Panayotis Mertikopoulos, and Zhengyuan Zhou. Gradient-free online learning in continuous games with delayed rewards. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[193] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

[194] Yaodong Yang, Chengdong Ma, Zihan Ding, Stephen McAleer, Chi Jin, Jun Wang, and Tuomas Sandholm. Game-theoretic multiagent reinforcement learning. *arXiv preprint arXiv:2011.00583*, 2025.

[195] Cunxi Dai, Xiaohan Liu, Koushil Sreenath, Zhongyu Li, and Ralph Hollis. Interactive navigation with adaptive non-prehensile mobile manipulation. *arXiv preprint arXiv:2410.13418*, 2024.

[196] Juan Alvarez-Padilla, John Z Zhang, Sofia Kwok, John M Dolan, and Zachary Manchester. Real-time whole-body control of legged robots with model-predictive path integral control. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2025.

[197] Corrado Pezzato, Chadi Salmi, Elia Trevisan, Max Spahn, Javier Alonso-Mora, and Carlos Hernández Corbato. Sampling-based model predictive control leveraging parallelizable physics simulations. *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[198] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2020.

[199] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2021.

[200] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[201] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[202] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2022.

[203] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2024.

[204] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. Accessed: 2025-12-06.

[205] Zenon W Pylyshyn and Ron W Storm. Tracking multiple independent targets: Evidence for a parallel tracking mechanism. *Spatial vision*, 3(3):179–197, 1988.

[206] Daryl Fougnie and Rene Marois. Distinct capacity limits for attention and working memory: Evidence from attentive tracking and visual working memory paradigms. *Psychological science*, 17(6):526–534, 2006.

[207] George A Alvarez and Steven L Franconeri. How many objects can you track?: Evidence for a resource-limited attentive tracking mechanism. *Journal of vision*, 7 (13):14–14, 2007.

[208] Mark Cavolowsky. Building better multi-agent systems: The theory and practice of multi-agent world models. Technical report, University of Maryland, College Park, 2025. URL `https://www.cs.umd.edu/sites/default/files/scholarly_papers/202501_Cavolowsky%2C_Mark_Scholarly_Paper.pdf`. Accessed: 2025-12-06.

[209] Wilko Schwarting, Tim Seyde, Igor Gilitschenski, Lucas Liebenwein, Ryan Sander, Sertac Karaman, and Daniela Rus. Deep latent competition: Learning to race using visual control policies in latent space. In *Proc. of the Conf. on Robot Learning (CoRL)*, 2021.

[210] Yang Zhang, Xinran Li, Jianing Ye, Shuang Qiu, Delin Qu, Xiu Li, Chongjie Zhang, and Chenjia Bai. Revisiting multi-agent world modeling from a diffusion-inspired perspective. *arXiv preprint arXiv:2505.20922*, 2025.

[211] Zifeng Shi, Meiqin Liu, Senlin Zhang, Ronghao Zheng, Shanling Dong, and Ping Wei. Gawm: Global-aware world model for multi-agent reinforcement learning. *arXiv preprint arXiv:2501.10116*, 2025.

[212] Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Behzad Dariush, Kwonjoon Lee, Yilun Du, and Chuang Gan. Combo: Compositional world models for embodied multi-agent cooperation. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2025.

[213] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[214] Riccardo Mereu, Aidan Scannell, Yuxin Hou, Yi Zhao, Aditya Jitta, Antonio Dominguez, Luigi Acerbi, Amos Storkey, and Paul Chang. Generative world modelling for humanoids: 1x world model challenge technical report. *arXiv preprint arXiv:2510.07092*, 2025.

[215] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[216] Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2025.

[217] Álvaro Serra-Gomez, Daniel Jarne Ornia, Dhruva Tirumala, and Thomas Moerland. A kl-regularization framework for learning to plan with adaptive priors. *arXiv preprint arXiv:2510.04280*, 2025.

Abbreviations

AWGN additive white Gaussian noise.

CoDi Coordinated Diffusion.

DSM denoising score matching.

ECDF empirical cumulative distribution function.

GNE generalized Nash equilibrium.

GNEP generalized Nash equilibrium problem.

HJ Hamilton-Jacobi.

i.i.d. independent and identically distributed.

IBR iterated best response.

IFT implicit function theorem.

IL imitation learning.

IOC inverse optimal control.

IQR interquartile range.

IRL inverse reinforcement learning.

KKT Karush–Kuhn–Tucker.

KL Kullback–Leibler.

LICQ linear independence constraint qualification.

LQ linear-quadratic.

LQR linear-quadratic regulator.

MARL multi-agent reinforcement learning.

MCP Mixed Complementarity Problem.

MLE maximum likelihood estimation.

MPC model predictive control.

MPEC mathematical program with equilibrium constraints.

MPGP model-predictive game play.

MPPI Model Predictive Path Integral Control.

NEP Nash equilibrium problem.

NN neural network.

OLNE open-loop Nash equilibrium.

PAC probably approximately correct.

POMDP partially observable Markov decision process.

POSG partially observed stochastic game.

PPO Proximal Policy Optimization.

QRE quantal-response equilibrium.

RL reinforcement learning.

RSS Robotics: Science and Systems.

SDE stochastic differential equation.

SEM standard error of the mean.

SVO social value orientation.

T-RO IEEE Transactions on Robotics.

UKF unscented Kalman filter.

VLM vision-language model.

Curriculum Vitæ

Lasse Peters

Education

2021–2026	Ph.D. Robotics, TU Delft, Delft, Netherlands. Advisors: Javier Alonso-Mora and Laura Ferranti.
2017–2020	M.Sc. Mechatronics, TUHH, Hamburg, Germany. Specialization: Intelligent Systems and Robotics.
2014–2017	B.Sc. Mechanical Engineering, TUHH, Hamburg, Germany. Specialization: Theoretical Mechanical Engineering.

Experience

2024	Visiting Researcher, Intent Robotics Lab, Carnegie Mellon University, Pittsburgh, USA. Host: Andrea Bajcsy, Robotics Institute. Projects: game-theoretic methods for multi-agent manipulation; latent safety filters; safe navigation.
2020–2021	Full-Time Researcher, Photogrammetry & Robotics Lab, University of Bonn, Bonn, Germany. Advisor: Cyrill Stachniss.
2019	Visiting Researcher, Hybrid Systems Laboratory, UC Berkeley, Berkeley, USA. Host: Claire J. Tomlin. Master's thesis on game-theoretic planning under uncertainty.
2018–2019	Visiting Student, UC Berkeley Extension, Berkeley, USA. Topics: model predictive control; control of unmanned aerial vehicles; introduction to artificial intelligence; sensor fusion in autonomous driving.

Honors & Awards

2025	Best Paper Award Finalist, TC on Robot Control Best Paper Award.

2025	RSS Pioneers: Selected as one of 33 early-career researchers.
2018–2019	DAAD ISAP Scholarship for study at UC Berkeley.
2018	Delmes-Buxmann-Award of the Rotary Club Hamburg-Haake for best Mechanical Engineering B.Sc. (TUHH, 2017).
2017–2019	Deutschlandstipendium.

Academic Service

Contributions Committee	ICRA 2023 Workshop on Multi-Robot Learning
Reviewer	Science Robotics; IEEE T-RO; IEEE TAC; IEEE RA-L; IEEE L-CSS, RSS; L4DC; IEEE ICRA; IEEE IROS; IEEE CDC; AAMAS

Invited Talks and Guest Lectures

2025	Guest Lecture: Perspectives on Inverse Games, UT Austin (Online).
2024	Invited Talk: Game-Theoretic Models for Multi-Agent Interaction, Princeton University, Princeton, USA.
2024	Guest Lecture: Game-Theoretic Models for Multi-Agent Interaction, Carnegie Mellon University (Online).
2023	Invited Talk: Contingency Games, Nuro (Online).
2023	Invited Talk: Contingency Games, Motional (Online).
2022	Guest Lecture: Uncertainty in Game-Theoretic Planning, UT Austin (Online).
2021	Guest Lecture: Inference and Learning in Games, UT Austin (Online).
2021	Guest Lecture: Model-Predictive Control, University of Bonn (Online).

Advising & Mentorship

2025–2026	Pathway Mentor: Joel Adebayo, CMU Africa (Rwanda).
2024–2025	Master's Thesis Advisor: Andrei Papuc, TU Delft. Thesis: "Interaction aware autonomous drone racing".
2022–2023	Master's Thesis Advisor: Xinjie Liu, TU Delft. Thesis: "On Game-Theoretic Planning with Unknown Opponents' Objectives".
2022	Master's Project Advisor: Maximilian Schmidt, TU Delft / TUHH. Thesis: "Game-Theoretic Motion Planning on a Real-World Hardware Platform".

Teaching

2022–2025	Teaching Assistant: Planning & Decision-Making, TU Delft, Delft, Netherlands. Responsibilities: course logistics; lecture material; discussion sessions.

List of Publications

Journal Articles

2025. Dong Ho Lee, **Lasse Peters**, and David Fridovich-Keil. *You Can't Always Get What You Want: Games of Ordered Preference.* IEEE Robotics and Automation Letters (RA-L).

2024. **Lasse Peters**, Andrea Bajcsy, Chih-Yuan Chiu, David Fridovich-Keil, Forrest Laine, Laura Ferranti, and Javier Alonso-Mora. *Contingency Games for Multi-Agent Interaction.* IEEE Robotics and Automation Letters (RA-L).

2023. **Lasse Peters***, Xinjie Liu*, and Javier Alonso-Mora. *Learning to Play Trajectory Games Against Opponents with Unknown Objectives.* IEEE Robotics and Automation Letters (RA-L).

2023. **Lasse Peters**, Vicenc Rubies-Royo, Claire J. Tomlin, Laura Ferranti, Javier Alonso-Mora, Cyrill Stachniss, and David Fridovich-Keil. *Online and Offline Learning of Player Objectives from Partial Observations in Dynamic Games.* The International Journal of Robotics Research (IJRR).

Peer-Reviewed Conference Publications

2025. Leonardo Santos*, Zirui Li*, **Lasse Peters**, Somil Bansal*, and Andrea Bajcsy*. *Updating Robot Safety Representations Online from Natural Language Feedback.* ICRA.

2025. Kensuke Nakamura, **Lasse Peters**, and Andrea Bajcsy. *Generalizing Safety Beyond Collision-Avoidance via Latent-Space Reachability Analysis.* RSS.

2025. Pau de las Heras Molins, Eric Roy-Almonacid, Dong Ho Lee, **Lasse Peters**, David Fridovich-Keil, and Georgios Bakirtzis. *Approximate solutions to games of ordered preference.* IEEE International Conference on Intelligent Transportation Systems (ITSC).

2024. **Lasse Peters***, Xinjie Liu*, Javier Alonso-Mora, Ufuk Topcu, and David Fridovich-Keil. *Auto-Encoding Bayesian Inverse Games.* WAFR.

2023. Jingqi Li, Chih-Yuan Chiu, **Lasse Peters**, Fernando Palafox, Mustafa Karabag, Javier Alonso-Mora, Somayeh Sojoudi, Claire Tomlin, and David Fridovich-Keil. *Scenario-Game ADMM: A Parallelized Scenario-Based Solver for Stochastic Noncooperative Games.* CDC.

2023. Jingqi Li, Chih-Yuan Chiu, **Lasse Peters**, Somayeh Sojoudi, Claire Tomlin, and David Fridovich-Keil. *Cost Inference for Feedback Dynamic Games from Noisy Partial State Observations and Incomplete Trajectories.* Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 1062–1070.

2022. **Lasse Peters**, David Fridovich-Keil, Laura Ferranti, Cyrill Stachniss, Javier Alonso-Mora, and Forrest Laine. *Learning Mixed Strategies in Trajectory Games.* RSS.

2021. **Lasse Peters**, David Fridovich-Keil, Vicenc Rubies-Royo, Claire J. Tomlin, and Cyrill Stachniss. *Inferring Objectives in Continuous Dynamic Games from Noise-Corrupted Partial State Observations.* RSS. code

2020. **Lasse Peters**, David Fridovich-Keil, Claire J. Tomlin, and Zachary N. Sunberg. *Inference-Based Strategy Alignment for General-Sum Differential Games.* AAMAS.

2020. David Fridovich-Keil, Ellis Ratner, **Lasse Peters**, Anca D. Dragan, and Claire J. Tomlin. *Efficient Iterative Linear-Quadratic Approximations for Nonlinear Multi-Player General-Sum Differential Games.* ICRA.

Included in this thesis.
Finalist for the *TC on Robot Control Best Paper Award (2025).*
* indicates equal contribution.

Acknowledgments

I am deeply thankful for the opportunities to learn and grow over the past four years, and for the many wonderful people who have entered my life on the path to this degree. This work would not have been possible without the support of all of you: my colleagues, collaborators, friends, and family. Therefore, I would like to take this opportunity to express my gratitude to those who have had a part in this effort.

First and foremost, I would like to thank my advisors and promotors, **Javier** and **Laura**, for their support and guidance throughout my PhD. Thank you for guiding me through the ups and downs of the PhD journey, for giving me professional and personal advice, for giving me the freedom to pursue my own research interests, and, above all, for adapting your supervision style to the many different stages of my PhD: cheering me on when I had a run, pushing me when I needed it, encouraging me when I felt stuck, and giving me space and time when life outside of research did not go as planned. You two make a great team and I am very grateful for your support.

I would also like to express my gratitude to **Matthijs Spaan**, **Sergio Grammatico**, **Jana Tumova**, **Guillaume Sartoretti**, and **Robert Babuška** for kindly agreeing to serve on my doctoral committee, for taking the time to review my thesis, and for providing valuable feedback.

Furthermore, I would like to thank my **colleagues at TU Delft**, especially the members of the Cognitive Robotics group, for your camaraderie and for the many inspiring conversations, some of which inspired the ideas that have made it into this dissertation. A special thanks also goes to **the technical support staff at TU Delft**, especially to **Kseniia**, **Maurits**, **André**, **Thomas**, and **Roel**, for helping my Master's students and me with the harsh reality of getting things to work in real life—patiently fighting hardware failures, network issues, 3D printer failures, and administrative mazes.

My most vivid professional memories are of tackling problems that could only be solved in a team, requiring everyone to step out of their comfort zone to learn new tools and techniques. In this context, a special thanks to **David** and **Forrest** for being the best remote collaborators I could have asked for. Beyond being inspiring role models and mentors, you two turned the period of the global pandemic into some of the most exciting times of my PhD. Another key figure who shaped the way I think about research and academic leadership is **Andrea**. Visiting your lab at CMU was a time of some of the most intense and rewarding learning experiences I have had in the past four years. In this context, a special thanks to **Ken**, **Yilin**, and **Ravi** for making me feel welcome in the lab right from the start, and to my fellow visiting students **Leo**, **Joe**, **Michał**, and **Ignacy** for looping me into your lively friend group, making it a truly great summer of learning, fun, and pickled food.

This dissertation marks the end of both my doctoral studies and, at least for now, my time in Delft. I consider myself incredibly lucky to have made many close friends over the past four years here, and I would like to thank all of you for making Delft feel like home very quickly—through dinners, board game evenings, BBQs, weekend food tours, bouldering, and squash sessions. Some of my greatest memories from the last four years are shared with **Álvaro** and **Natalia** over private salsa lessons and food tours; with **Max** and **Andreu** over some of the most interesting conversations beyond research I have had in this time; with **Corrado** and **Patricija** over always cheerfully agreeing to every weekend side quest and request for a squash session; with **Bea** and **Giovanni** over BBQs, hot chocolates on Dutch rainy days, and debating on the living room floor; with **Elia** and **Daniela** over home-cooked pizza; with **Italo**, **Tomás**, and **Anton** over group dinners at Pietheijnstraat; and with **Lorenzo** and **Mariano** over pasta and board game nights. Above all, and with intentional repetition, thank you again to **Álvaro** and **Natalia** for being my family and home away from home.

Before I came to Delft, I took a stab at a PhD in the Photogrammetry & Robotics Lab at the University of Bonn. While I didn't end up staying long (because I found out that my research interests were not sufficiently aligned with the group), I am very grateful for the year that I spent there. It has greatly impacted the way that I think about *how* to conduct research, organize a research group, and communicate my research to others. Beyond the team of students and postdocs, I am especially grateful to **Cyrill Stachniss** for being an inspiring leader and mentor, and for supporting my move to TU Delft.

Until 2018, I had no plans to pursue a PhD, but I am very glad I did. A special thanks goes to **Robert Seifried** for encouraging me to apply for the UC Berkeley exchange program that would ultimately set me on this path. My time in Berkeley is where I fell in love with research. To the Berkeley folks: working alongside you during my Master's and witnessing the inspiring and stimulating environment that all of you created is the reason I decided to pursue this degree. A special thanks to **Claire Tomlin**, **Zach**, **David**, and **Forrest**, who played a major role in setting me on this path and supported my PhD applications.

Going further back in time, I would like to thank my former RoboCup teammates for getting me hooked on robotics in the first place. Among this large crew, a special thanks goes to **Ploth**, **Robert**, **Chris**, **Wege**, **Warmuth**, **Thomas**, **Georg**, **Maxi**, **René**, **Erik**, and **Flinzer** for the great times in and outside the robot lab. Among my favorite memories are days and nights of coding and debugging, countless BBQs that were planned for 16:00 but pushed to 20:00 "because…robots", and traveling for robot soccer competitions, coding marathons, and team events. The group of friends that formed through this experience and has grown over the years to include **Jasmin**, **Caro**, **Hannah**, and many more is a place that I will always call home, no matter how long I've been away.

My final thanks go to my **family**, especially my **parents**, for always supporting me and being there for me. The steadfast stability and support that you provide are the reason I can pursue my ambitions and dreams. You allow me to go where my interests take me because I know that home is only a phone call away and that things will be all the same when I get back.

Lasse
Delft, February 2026

Propositions

accompanying the dissertation

Game-Theoretic Motion Planning for Multi-Agent Interaction

by

Lasse Peters

1. To understand the intentions of others, robots must reason about the interaction *in the context of their own sensory limitations.* [Chapters 2 and 3]

2. By reasoning about the actions of all players jointly, robots can discover safer and more efficient plans than traditional "predict-then-plan" approaches. [Chapters 3 and 4]

3. While games provide a powerful interaction model, real-time implementations of game-theoretic motion planning for complex interactions remain intractable without additional approximations [Chapters 3, 5, and 6].

4. Game-theoretic reasoning has the potential to drastically improve the data efficiency of learning-based approaches [Chapters 5 and 6].

5. Offline learning alone is doomed to fail: resilient autonomy requires some degree of online reasoning.

6. Model-free approaches are a hoax: ultimately, every method that works in practice needs a model.

7. The role of academia is to explore, so that industry can exploit.

8. A key challenge in robotics and in life is knowing what to learn next in order to improve.

9. At least 30% of people in a robotics institution should focus on state estimation and perception, working closely with planning and control teams to understand each other's needs.

10. To keep pace with the rapid evolution of the field, robotics research funding should be allocated dynamically rather than through rigid, long-term grant cycles that risk locking researchers into outdated goals.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors Prof. Dr. Javier Alonso-Mora and Dr. Laura Ferranti.