Learning and Control for Interactions in Mixed Human-Robot Environments

by

Wilko Schwarting

BSc, ETH Zurich (2014)
MSc, ETH Zurich (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2021

Author . . . .
Department of Electrical Engineering and Computer Science
January 14, 2021

Certified by . . . .
Daniela Rus
Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science, Chair,
Department Committee on Graduate Students

Learning and Control for Interactions in Mixed Human-Robot Environments

by

Wilko Schwarting

Submitted to the Department of Electrical Engineering and Computer Science
on January 14, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Autonomous robots will soon be a commonplace presence in our daily lives in environments such as homes, factories, and roads. In order to reap the tremendous benefits that these robots offer to society, we must ensure that they can interact with humans seamlessly and safely.

In this dissertation, we study intelligent agents that learn how to reason about human behavior and people's intentions. These agents predict others' intentions and implicitly communicate their own intentions through human-like actions that can be understood by people. They also anticipate and leverage the effect of their actions on the actions of others in the environment. When their own interests and the interests of others are not aligned, the agents quantify people's willingness to cooperate or defect and negotiate through social behavior. The agents form beliefs by perceiving the world and the actions of others. They create plans to actively gather information about themselves, others, and the environment, while simultaneously avoiding actions that lead to high uncertainty. They also reason about the beliefs of others, and can leverage how their actions influence others' beliefs.

In part (I) of this thesis, we formulate *social human-robot interactions* between agents as a best-response game wherein each agent negotiates to maximize their utility, and learn human rewards from data. We measure Social Value Orientation (SVO) to quantify an agent's degree of selfishness or altruism to better predict human behavior. In part (II) we additionally enable agents to leverage information gain and *reasoning about the beliefs* of others in stochastic environments with partial observations by combining game-theoretic and belief-space planning. In part (III) we present a multi-agent reinforcement learning algorithm that learns competitive visual control policies through self-play in imagination. The agent *learns from competition* by imagining multi-agent interaction sequences in the compact latent space of a learned world model that combines a joint transition function with opponent viewpoint prediction. Lastly, in part (IV) we introduce *Parallel Autonomy*, a *Guardian system* that uses uncertain predictions to provide safety in challenging driving scenarios while following people's desired actions as close as safely possible.

Thesis Supervisor: Daniela Rus
Title: Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science

This doctoral thesis has been examined by a Committee of the Department of Electrical Engineering and Computer Science:

Professor Daniela Rus . . . .
Chairperson, Thesis Supervisor
Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science

Professor Sertac Karaman . . . .
Member, Thesis Committee
Associate Professor of Aeronautics and Astronautics

Professor Leslie Kaelbling . . . .
Member, Thesis Committee
Panasonic Professor of Computer Science and Engineering

Acknowledgements

There are many people that I would like to thank for helping me along the road to earning a PhD. First and foremost I would like to thank my advisor Daniela Rus for giving me the freedom to follow my own research ideas and for supporting me along the way with advice, resources, and an exceptional environment. This freedom gave me the opportunity to explore many more directions throughout the PhD than I would have imagined and made the journey fun and exciting. Daniela always provides refreshing and creative perspectives. When we are stuck in the details of implementations and research, she reminds us to focus on the important higher-level aspects of the challenges we tackle. I will miss being a PhD student in her lab. I would also like to thank Sertac Karaman for sharing his deep knowledge of autonomous systems and providing a valuable feedback loop for ideas. Next, I want to thank Leslie Kaelbling for supporting me during the PhD and being part of my committee. I would like to thank the committee, Daniela Rus, Sertac Karaman, and Leslie Kaelbling for giving great feedback on the thesis and being very supportive throughout the process.

I also want to thank a host of other MIT faculty that I have interacted with throughout the years. These interactions are a big part of what made MIT such a special place to complete the PhD. Specifically, I would like to thank Russ Tedrake, Pulkit Agrawal, Josh Tenenbaum, and John Leonard. I have to thank Javier Alonso-Mora for introducing me to the power of optimization and providing me with a great start to research at MIT.

I want to thank Roland Siegwart, Margarita Chli, Ulrich Schwesinger, and Paul Furgale for giving me the opportunity to delve into research already during my time as a Bachelor's and Master's student at ETH Zurich. I would like to thank Patrick Pascheka and Somudro Gupta for introducing me to the field of autonomous driving which had a long-lasting impact on my career.

During the PhD I had the chance to collaborate with many wonderful people who introduced me to new perspectives in areas I would have never imagined. I would like to thank Alexander Amini, Alexander Wallar, Alyssa Pierson, Anshula Gandhi, Bingyu Zhou, Brandon Araki, Changhyun Choi, Cristian-Ioan Vasile, Felix Naser, Guy Rosman, Hans Andersen, Igor Gilitschenski, Javier Alonso-Mora, Jonathan DeCastro, Liam Paull, Lucas Liebenwein, Noam Buckman, Roshni Sahoo, Ryan Sanders, Scott Pendleton, Tim Seyde, You Hong Eng, and Ava P. Soleimany for working with me on exciting topics.

Without the community of the Distributed Robotics Laboratory, our interesting discussions during lunchtime or deep at night, hikes in the mountains, and Oktoberfests, this journey would not have been as amazing. I want to thank all of my current lab members for this experience: Aaron Ray, Alexander Amini, Alyssa Pierson, Andy Spielberg, Annan Zhang, Brandon Araki, Cenk Baykal, Daniel Wrafter, Hunter Hansen, Igor Gilitschenski, James Bern, John Romanishin, Johnson Wang, Joseph DelPreto, Joshie Hughes, Lillian Chin, Lucas Liebenwein, Murad Abu-Khalaf, Noam Buckman, Paul Tylkin, Ramin Hasani, Ryan Truby, Teddy Ort, Tim Seyde, Veevee Cai, Xiao Li, and Yutong Ban. Many lab members have already graduated or finished their assignments, but I enjoyed very much working with all of you. It was great learning from you and inspiring to see your future paths: Andrés Salazar-Gómez, Changhyun Choi, Cynthia Sung, David Dorhout, Guy Rosman, Javier Alonso-Mora, Liam Paull, Mikhail Volkov, Robert MacCurdy, Sebastian Claici, Stephane Bonardi, Stephanie Gil, and Tobias Nägeli.

I have had some of the most wonderful years of my life in Cambridge and Boston, and I owe that to my awesome and supportive friends. To my partner, Alex, thank you for your support, advice, and encouragement. You remind me that there is a life outside of research and academia.

I want to thank my family for always encouraging and believing in me. I especially want to thank my mom, Silke, and my dad, Hajo. Thank you for loving and supporting me throughout my journey, even if that meant risking our neighbors' cars while I was exploring water rockets. I also want to thank my sister Deike, and my grandparents, Helga, Hans-Georg, Hanna, and Ingo, for encouraging me to follow my curiosity already early on.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Vision

In the future, robots will be a part of our daily lives, supporting us with physical tasks in the household and factories, delivering our goods, driving our cars, and harvesting our crops. Society will receive tremendous benefits from the responsible use of robot capabilities as tools that enable and support people. Robots will closely interact with people while reasoning about human intentions. They will need to be robust to uncertainty from perception and prediction and leverage information gain and awareness of uncertainty to their advantage. Robots need to reason about the environment based on a higher-level intuitive understanding of the world that is learned instead of engineered.

In the following, we describe the progress in robotics in three waves. We cluster waves by increasing capabilities in versatility, sensing, planning, learning, and human-robot interaction. Other clusterings of progress in robotics and automation have been suggested. These include progress in mobility and manipulation complexity [219], progress from handcrafted methods to statistical learning and contextual adaptaion [82], and the transition from manual to autonomous automation [105].

1.1.1 First Wave of Robotics

The current state-of-the-art in the robotics and automation industry is far from the vision stated above. In the first wave of robotics, robots have been successful at scale in simple, structured, and static environments. Applications for this first wave of automation range from assembly and welding of cars and planes, through forming and processing of composite materials, to packing and dispensing of electronics. This stage of automation has been driven by the goal to free people from dangerous, repetitive, or simply unpleasant jobs; to improve quality and efficiency in manufacturing; and lastly to cut manufacturing costs [197]. Robots were designed and manually programmed for a single purpose with simple and repetitive tasks. Due to the lack of complex sensing capabilities and relying on preprogrammed plans, robots were unable to flexibly adapt to novel tasks or changes in the environment. Additionally, human workers and robots were strictly separated, dictated by safety concerns.

1.1.2 Second Wave of Robotics

The second wave of automation and robotics has enabled the use of robots in less confined environments. Robots have been equipped with multi-purpose sensors and actuators, and employ machine learning and online planning to adapt to new environments. These robots perceive the environment through sensors and estimate their state and the state of other objects in the environment. They subsequently use this information to plan actions online following objectives such as moving to a goal location and avoiding collisions. Robots in the second wave have been developed for numerous purposes beyond manufacturing. Picking unknown and unseen objects, flying drones without colliding, and autonomously vacuuming floors are some example use cases. Due to online sensing and planning capabilities, robots have gained the capability to operate in dynamic environments. This has also allowed them to work in closer proximity to people. Nonetheless, most robots have treated people as generic obstacles and remained oblivious of people's intentions. Additionally, people needed diligent training to interact safely with robots. In cases where human workers and robots have worked alongside each other, workers have been aware of the reoccurring robot tasks and could therefore anticipate and understand the artificial actions a robot executes.

In the manufacturing and factory setting, we are motivated by the realization that full automation of some tasks remains notoriously hard or expensive. Moreover, replacing the workforce through automation and robotics is undesirable due to its negative impact on society [32, 23]. Augmenting human workers through robotic tools or human-robot collaboration has increased productivity, reduced fatigue, and allowed robots to execute tasks that are too difficult to complete without assistance. Examples include robotic surgery, where surgeons direct precise motions of robot arms in augmented reality, or automotive assembly, where workers route wires and robots lift heavy parts for easier access.

1.1.3 Third Wave of Robotics

While the use of robotics in the industry is ever-increasing, the third wave of robotics will take place in the form of consumer products. It will place robots into our daily lives in the shape of household robots, autonomous cars, delivery drones, robot caregivers, robot chefs, and many more. The impact these robots will have on us and how we live our lives will be much more profound than the previous two waves. These robots have to be able to reason about the world to understand the cluttered, disorganized, and ever-changing environments they operate in. Most importantly, they need to safely and seamlessly interact with users and other people in our world. While the first two waves of robotics and automation saw tremendous progress in the areas of versatility, sensing, planning, and learning, the third wave of robotics will be characterized by skillful interactions in mixed human-robot environments.

Socially-Compliant Interactions

Because these types of robots will be interacting closely with ordinary people, they will have to be able to reason about human behavior and intentions. For instance, drivers at four-way intersections wait to signal other drivers to proceed or slowly enter the intersection to ask for permission to pass. Similarly, human drivers initiate lane-merges in heavy traffic by nudging into the neighbor's lane to communicate intent. Drivers only proceed if their counterpart responds by creating enough space to complete the merge safely. Thus, robots will not only have to predict intentions of other people but also implicitly communicate their own intentions through human-like actions that can be understood by people. In contrast to interactions in artificially constructed environments such as factories, people can not as effortlessly anticipate the behavior of robots. One reason is that in comparison to professional workers, regular people lack professional training. People are also unaware of the robot's goals, and the executed tasks are not necessarily repetitive such that they cannot predict robot behavior from experience. Thus, not complying with expected human social behavior, i.e. social compliance, can result in inefficiencies or in the worst case compromised safety. Simultaneously to implicit communication, robots will be able to anticipate and leverage the effect of their actions on the actions of others in the environment. When a driver's current lane is ending, it can be necessary to avoid a collision by forcing other drivers to comply with the merge by cutting into their lane. While against their will, drivers fall back eventually to increase the gap size and allow the merge to avoid a collision themselves. Autonomous vehicles will have to be able to perform a similar set of behaviors and plan on leveraging the influence of their actions on other drivers when necessary.

Social Behavior

Most interactions between robots and human workers in the second wave of automation consist of either simple collision avoidance or collaboration towards a known joint goal. In contrast, in our everyday lives, people's interests and the interests of others are not always well aligned. We implicitly negotiate outcomes of these social dilemmas through social behavior. Furthermore, people do not always act purely in self-interest but are frequently willing to cooperate. In the long term, cooperation can yield improved joint outcomes, reduced inequality, and increased efficiency for all.

Drivers often go out of their way for others during highway merges and slow down to yield despite not gaining any direct benefit from it. Nonetheless, the overall traffic flow improves if more drivers engage in similar behavior. Equally nuanced negotiations of their interests and the interests of others occur in many scenarios beyond navigation. Therefore, beyond goal and intent inference, robots will need to quantify the degree of altruism or selfishness of others by observing their actions and deducing whether they will cooperate or defect. Incorporating this information to improve predictions can then be leveraged during planning.

Reasoning over Beliefs

Robots often do not have ground truth information about the world. They perceive the world and others influenced by uncertainty from perception, prediction, or the environment itself. Similarly, inclement weather such as snowstorms, heavy rain, or fog may impair people's vision. Humans intuitively adjust to uncertainty in the surroundings by acting more conservatively, slowing down, increasing distances to others to maintain safety, or deeming certain activities overall unsafe under certain conditions. Other sources of uncertainty are occlusions, blind spots, or prevalent unknowns about the environment. Bicyclists slow down before occluded corners to avoid collisions with others. Similarly, robots will avoid actions that lead to high uncertainty in scenarios where this is undesirable such as when minimizing risk.

People also plan to actively gather information about themselves, others, and the environment. For instance, cliff divers first check the depth of the water before taking the plunge, and people sweep the floor with a flashlight at night to identify obstacles. Robots will not only have to reason about their own beliefs (*"I think..."*) about the world but also the beliefs of others (*"I think that they think..."*). Making the theory of mind [155] available to robots will give them the ability to attribute mental states to oneself and others. These mental states include beliefs, intents, desires, emotions, and knowledge, among others. Realizing that others' beliefs, desires, intentions, and perspectives are different from one's own is essential when interacting with people. On highways, motorcyclists minimize the time in blind spots of trucks to ensure to be seen and thus reduce the risk of collisions. Robots will similarly be able to leverage how one's actions impact the change in the confidence of others. Opening doors when someone's hands are full, offering a seat on the bus to those in need, or guiding people with visual impairments around obstacles are examples where it is necessary to first understand the state of mind of the other person before acting accordingly.

Complex Emergent Behavior from Interactions

At an early age, children learn complex skills and behaviors. They quickly learn to identify and manipulate objects, to communicate, and to move by either crawling or walking. In contrast, in the first wave of robotics, motions were manually programmed and engineered. In factory settings, robot arms repetitively move from one position to another to manipulate, form, weld, or inspect objects. For these types of problems, the task and problem formulation is straight forward. For more complex tasks such as tying shoelaces, opening a door, or folding clothes, it can be very challenging to engineer a controller or even mathematically formulate the problem. Precisely stating a problem for a task is also very demanding if we knew accurate models of the world and if uncertainty was negligible. Encoding complex behaviors is even more challenging in mixed human-robot and multi-agent environments. For instance, in competitive multi-agent racing, we will expect behaviors such as cutting off other drivers in turns, blocking overtaking maneuvers, or executing a PIT maneuver (turning an opponent's car abruptly sideways, causing the driver to lose control and stop). Instead of manually crafting all these necessary behaviors for racing, they should emerge from a general problem formulation, such as winning the race, and general-purpose algorithms. Ideally, we will develop robots with algorithms that do not require detailed manually engineered models of the world, as they can be hard to obtain, and only rely on minimal prior information. Instead, robots will either optimize their behavior directly or learn models of the world from past experience of environment interactions. Additionally, robots will learn from each other to increase capabilities and improve performance through competition and self-play.

1.2 Challenges

To progress towards the vision presented in the previous section, a number of conceptual and technical challenges need to be overcome. A large amount of literature related to this thesis exists. We will discuss this thesis and its related work in the context of these main challenges.

1.2.1 Modeling Interactions

Similarly to how people interact with each other, a robot's actions influence the actions of other people, and people's actions affect the robot's actions. Constructing or learning a model of interactions allows the robot to predict, plan, and leverage their influence on other people in the environment.

Many approaches learn to imitate human behavior directly from data. A variety of neural network architectures have been proposed including Long Short-Term Memory (LSTM) Networks [8], Generatic Adversarial Networks (GAN) [15, 73, 101], attention mechanisms [207, 161], Graph Neural Networks (GNN) [91], Encoder-Decoders [221], or Variational Autoencoders (VAE) and Inverse Optimal Control (IOC) [107]. These purely data-driven approaches predict future trajectories of people based on the current state. Nonetheless, the predictions are not conditioned on the robot's future actions and thus cannot anticipate how people change their behavior based on the robots actions. Therefore, they are not directly suitable for planning.

A similar thread in learning interaction models is to employ joint distributions. Joint distributions essentially model the robot as one of the other agents. Applications include Gaussian processes [199], joint reward functions [98], corresponding to distributions according to the principle of maximum entropy, or joint energy potentials [158]. These approaches, leveraging manually engineered features to handle interactions, optimize both the robot's future actions as well as all other agents' predicted actions by jointly minimizing the sum of all agents' costs. This formulation implies that agents' interests are well aligned, and that agents are willing to sacrifice their utility for the joint utility. Nonetheless, in practice people's interests are not always well aligned.

Game-theoretic planning successfully solves problems where an agent's objective is at odds with the objective of other agents, such as in modeling human behavior [111, 172, 200], and leveraging the effects on humans by autonomous cars [163, 62, 173]. [125] presents a recent review on game theory and control. In game theory, the Nash equilibrium is a proposed solution of a non-cooperative game involving two or more players. At the equilibrium point, no player has anything to gain by changing only their own strategy. Solving for Nash equilibria has been applied to competitive racing of cars [217, 119], drones [183] and navigating vehicles through traffic [54, 173].

In this thesis we model interactions between agents as a best response game wherein each agent negotiates to maximize their own utility and calibrate reward functions to human data through Inverse Reinforcement Learning (IRL). We also ensure safety by integrating general constraints. With the game-theoretic formulation, the autonomous system can leverage the effect of their actions on other people. The Nash equilibrium pf the game yields predictions for all other agents in the environment and a control policy for the robot.

1.2.2 Computing Nash Equilibria

We often pose single agent motion-planning as an optimization problem. Solving for Nash equilibria in multi-agent environments can be thought of as solving an interdependent optimization problem for each agent and is therefore computationally challenging. Solution methods for Nash equilibria include Iterated Best Response (IBR) [217, 183, 49, 65], using discrete payoff matrices [119], or solving the optimality conditions. We present a tractable reformulation [173] of the necessary optimality conditions respecting general constraints and solve for the Nash equilibrium of the game one order of magnitude faster than state-of-the-art methods such as IBR. This allows the system to operate in real-time. We also show an alternative method that achieves linear complexity per iteration over the planning horizon by solving a locally quadratic game in the backward pass of the Iterative Linear-Quadratic regulator control (iLQR) [174].

1.2.3 Human Social Preferences in Robotics

Behavioral and experimental economics shows that people have unique and individual social preferences, including: interpersonal altruism, fairness, reciprocity, inequity aversion, and egalitarianism. Self-interested models, like the *homo economicus* [42], assume agents maximize only their own reward, which fails to account for nuances in real human behavior. In contrast, Social Value Orientation (SVO) indicates a person's preference of how to allocate rewards between themselves and another person. SVO can predict cooperative motives, negotiation strategies, and choice behavior [52, 126, 202, 132, 153, 5]. People observe and estimate SVO from actions and social cues [178]. While some works attempt to estimate human internal state [162, 191, 189], this thesis presents a method to estimate peoples' social preferences in the form of SVOs from observed trajectories and integrates the estimates into an interactive prediction and planning framework.

We also enable robots to show behavior corresponding to social preferences. Prosocials exhibit more fairness and considerateness compared to individualists [52], and engage in more volunteering, pro-environment, pro-community, and charitable efforts [126, 70, 203, 57]. They also tend to minimize differences in outcomes between self and others (inequality aversion, egalitarianism) [134, 202]. Additional findings suggest reciprocity in SVO and resulting cooperation [133, 4, 130]. Therefore, instilling prosocial behavior into autonomous agents is desirable and deviation from purely selfish behavior is necessary.

1.2.4 Socially-Compliant Behavior and Implicit Communication

Achieving predictable behavior in robots is fundamental for the safety of other people, since it enables them to understand and appropriately respond to the robot's actions. One way to obtain predictable behavior is through behaving according to the social expectations of the group, i.e. social compliance. To accomplish social-compliance, the autonomous system must behave as human-like as possible [98, 206, 199, 100], which requires an intrinsic understanding of human behavior as well as the social expectations of the group. Human behavior may be imitated by learning human policies from data through Imitation Learning [159, 86, 8, 15, 73, 101, 207, 161, 91, 221, 107]. Our autonomous system design enables social compliance by learning human reward functions through IRL [222]. The utility optimizing policy is human-imitating [1, 223, 66, 34, 218, 164, 100, 98]. Furthermore, integrating the optimal control policy into a best response game with learned rewards yields an interactive human-imitating policy capable of leveraging the effect of actions on the actions of others. We additionally incorporate SVO to mirror the utility-maximization strategies of humans with heterogeneous social preferences in social dilemmas [133].

1.2.5 Reasoning about Own and Others' Beliefs

While game-theoretic planning models the interaction and dependency among agents, it does not address the uncertainty of the information accumulated by the agent for decision making. Belief-space planning [94] uses beliefs, which are the distributions of the robot's state estimate, to represent the uncertainties in the perception of the robot. The problem of computing a control policy over the space of belief states is formally described as a Partially Observable Markov Decision Process (POMDP), and has been studied extensively. Solving a POMDP to global optimality is NP-hard: solutions such as point-based algorithms [24, 102, 151, 78] in discrete space as well as sampling based solvers [40, 143, 156] are bound to the curse of history, i.e., that the computational complexity grows exponentially with the planning horizon. Optimization-based approaches have been developed for planning in continuous belief space [106, 142, 152, 201, 190, 152, 142], by approximating beliefs as Gaussian distributions and computing a value function valid in local regions of the belief space. Optimization-based methods scale linearly in the planning horizon while point-based algorithms only scale exponentially. In this thesis we integrate game-theoretic planning into Gaussian-belief space planning to reason about the own belief and the belief of others.

1.2.6 Complex Emergent Behavior

While coping with interactions and reasoning about beliefs is challenging, as outlined above, agents also need be capable of complex behaviors and skills which can be hard to manually engineer. Instead, complex behaviors must be learned from interacting with others and the environment without the need for engineering. Recently, Reinforcement Learning (RL)-agents have received increased attention sparked by surpassing human-level performance in many applications. This is especially the case for single agent environments such as ATARI games [129] or control tasks in simulation environments [37, 194, 56]. The agents' ability to learn behaviors scales to complex tasks [117, 128, 168] including humanoid locomotion. Progress on simpler but real-world tasks such as object manipulation and grasping [109, 48, 72] is equally impressive. Stabilizing training and reducing sample complexity enabled much of the progress through algorithms like DDPG [117], PPO [169], TRPO [167], MPO [3], SAC [75], or a combination thereof such as Rainbow [85].

More interestingly, Multi-Agent Reinforcement Learning (MARL) approaches also surpassed human-level performance in many multi-agent environments including complex board games such as GO [180, 182], chess, shogi [181], as well as two-player [131] and multi-player poker [39]. A MARL approach achieved grandmaster-level performance in the real-time strategy game Star Craft II [208, 209]. Nonetheless, agents need access to privileged ground truth information, e.g. categorized entity lists and a pre-processed multi-layer map. We are motivated by complex emergent behaviors from competition such as complex interactions [29] or tool use [25]. While often based on principled game-theoretic modelling [123, 104, 83, 84], most MARL approaches focus on domains involving discrete low-dimensional action spaces [38, 185]. Recently, MARL approaches have also been developed for continuous control tasks in cooperative, competitive, and team competition environments [122, 112, 90], although on low-dimensional observations. A common thread in all of these approaches is the use of self-play auto-curricula: Agents gain competitiveness by playing against themselves and iteratively improve performance. In this thesis we present a MARL algorithm for continuous control tasks that learns competitive strategies directly from image observations without relying on privileged information. Additionally, we reduce sample complexity by learning a world model in latent space. Agents can then gain competitiveness from imagined self-play game rollouts reducing the number of expensive real-world interactions needed.

1.2.7 Learning from High-Dimensional Observations

While the previously outlined challenges mostly tackle how agent behavior should be modeled and learned, another important aspect is dealing with real-world observations. In most applications we do not have direct access to low-dimensional state observations, e.g. the state of the vehicle. Instead, robots have to rely on high-dimensional observations such as camera images or LIDAR. An autonomous vehicle has to sense the road through camera, LIDAR, or other modalities for a planner to compute controls to keep the car on the road. To avoid relying on prior knowledge, in this case the road, we can also learn a generic world transition model in a compact latent space based on high-dimensional observations. The use of neural networks, particularly recurrent neural networks, for modeling the evolution of the environment allowing for *"mental imagination"* has been proposed as early as 1990 [166] and recently revisited in [74]. In a similar spirit, variational inference approaches have been combined with linear-quadratic-regulators for learning to control from raw images [215, 220]. Another line of recent algorithms combines latent (multi-step) imagination with video prediction [95, 76] achieving state-of-the-art performance on several standard benchmarks. We draw inspiration from these ideas and generalize the concept of multi-step latent imagination to multi-player settings. Imagining the outcomes of interactions in a latent world model allows us to apply game-theoretic methods such as self-play to learn complex emergent behaviors.

1.2.8 Autonomous Guardian Systems

The previously outlined challenges focused on interactions between robots and agents. Another mode of interaction we propose is augmenting agents with robots and autonomous systems. These systems can enable, elevate, and protect people. In this work, we present a Parallel Autonomy system that acts as a guardian angel to ensure people's safety. We apply the concepts to automated driving.

In theory, safety can be guaranteed for deterministic systems by computing the set of the states for which a vehicle will *inevitably* have a collision. We can then ensure that the vehicle never enters that set. The set is referred to as the capture set [63, 7, 35, 87], inevitable collision states (ICS) [64, 31, 11, 47, 114], the region of inevitable collision (RIC) [45], the target set [127], or risk level set [146, 149, 88, 147]. However, necessary assumptions limit the approach to simplistic scenarios, since the set cannot be computed in real time. We follow the idea of [63, 31, 11] and define a set of probabilistic constraints for collision avoidance, which produce a safe behavior.

The most intuitive way of merging the human input with the output of a safety system is by linear combination of the two [17, 19] or completely taking over control [50]. In contrast, in this work we directly incorporate the human inputs into a minimally invasive optimization framework and add a soft *nudging* behavior to guide the driver. Providing feedback to the driver already before scenarios become critical prevents high intervention later on, while also avoiding to startle the driver. While other approaches also minimize the deviation from human input [179, 68, 59, 10], they do not generalize to complex environments with many dynamic obstacles and environments requiring advanced vehicle models including slip and normal-load transfer.

1.3 Outline and Contributions

This section outlines the approaches and contributions presented in this thesis. Section 1.3.1 describes contributions to model *Social Human-Robot Interactions* with a focus on social aspects of human behavior applied to autonomous vehicles and reasoning about the own beliefs and the beliefs of others in stochastic dynamic games.

Section 1.3.3 details contributions for intelligent agents that can leverage *Learning from Competition* to obtain complex skills without manual engineering of behaviors. These agents will learn to race using visual control policies in a learned latent-space model of the world. Finally, Section 1.3.4 outlines contributions for a *Guardian Angel* instantiated in the *Parallel Autonomy* framework for automated vehicles. It protects drivers in dynamic traffic scenarios and when road conditions are challenging.

1.3.1 Social Human-Robot Interactions

Social Behavior for Autonomous Vehicles

Deployment of autonomous vehicles on public roads promises increased efficiency and safety. It requires understanding the intent of human drivers and adapting to their driving styles. Autonomous vehicles must also behave in safe and predictable ways without requiring explicit communication. We integrate tools from social psychology into autonomous vehicle decision-making to quantify and predict the social behavior of other drivers and to behave in a socially-compliant way. A key component is Social Value Orientation (SVO), which quantifies the degree of an agent's selfishness or altruism, allowing us to better predict how the agent will interact and cooperate with others. We model interactions between agents as a best-response game wherein each agent negotiates to maximize their own utility. We solve the dynamic game by finding the Nash equilibrium, yielding an online method of predicting multi-agent interactions given their SVOs. This approach allows autonomous vehicles to observe human drivers, estimate their SVOs, and generate an autonomous control policy in real time. We demonstrate the capabilities and performance of our algorithm in challenging traffic scenarios: merging lanes and unprotected left turns. We validate our results in simulation and on human driving data from the NGSIM dataset. Our results illustrate how the algorithm's behavior adapts to social preferences of other drivers. By incorporating SVO, we improve autonomous performance and reduce errors in human trajectory predictions by 25%.

Contributions Our approach proposes a system to measure, quantify, and predict human behavior to better inform an autonomous system. A game-theoretic formulation models driving as a series of social dilemmas to represent the dynamic interaction between drivers. We formulate a direct solution of the best response game, allowing for fast, online predictions and planning, while integrating environmental and planning constraints to ensure safety. The game's reward functions are dynamic and dependent on the vehicles' states and the environment. Since we learn the reward functions from human driving data, we expect that our approach translates to other traffic scenarios and broadly, human-robot interactions, where we can derive similar predictions trained on relevant data. Using SVO, a common metric from psychology, we quantify human social preferences and their corresponding levels of cooperation. SVO measures how an individual weights their reward against the rewards of others, which translates into altruistic, prosocial, egoistic or competitive preferences. We estimate the human drivers' SVOs from observed motion, and set the Autonomous Vehicle (AV)'s SVO based on preference in a given scenario.

The main contributions are:

1. Modeling driving as a dynamic game and computing its Nash Equilibrium
2. Reformulating the Nash Equilibrium through Karush-Kuhn-Tucker (KKT) conditions for a speedup of more than 10x while preserving safety constraints
3. Predicting human actions from expected utility maximization
4. Integrating SVO preferences into the utility-maximizing framework
5. Estimating SVO online from observed driving trajectories
6. Evaluation of emerging socially-compliant autonomous driving behavior
7. Validation on NGSIM driving data

We present the approach, previously published in [173], in Chapter 2 with additional additional analysis and results.

1.3.2 Reasoning about own Beliefs and the Beliefs of Others

Stochastic Dynamic Games in Belief Space

Information gathering while interacting with other agents is critical in many emerging domains, such as self-driving cars, service robots, drone racing, and active surveillance. In these interactions, the interests of agents may be at odds with others, resulting in a non-cooperative dynamic game.

Since unveiling one's own strategy to adversaries is undesirable, each agent must independently predict the other agents' future actions without communication. In the face of uncertainty from sensor and actuator noise, agents have to gain information over their own state, the states of others, and the environment. They must also consider how their own actions reveal information to others.

We formulate this non-cooperative multi-agent planning problem as a stochastic dynamic game. Our solution uses local iterative dynamic programming in the belief space to find a Nash equilibrium of the game. We present three applications: active surveillance, guiding eyes for a blind agent, and autonomous racing. Agents with game-theoretic belief space planning win 44% more races compared to a baseline without game theory and 34% more than without belief space planning.

We describe the work, previously presented in [174], in Chapter 3.

Contributions We present a computationally-tractable solution to multi-agent planning that combines game-theoretic planning and belief space planning to interact within a problem formulated as a game, gain information, and leverage the information gain to improve the agents' control policies. The main limiting factor in applying either game theory or belief space planning, and even more so the combination of both to robotic control problems lies in the associated computational complexity. To the best of our knowledge this is the first work to combine general dynamic games and planning in belief space into an efficient real-time algorithm. The main contributions of this research thrust are:

1. A method for computing Nash equilibria for dynamic games in belief space in

real time

2. Solving a quadratic game at each stage of the recursive backward pass of a belief space variant of iLQG yields linear time-complexity in the planning horizon

3. A linear feedback law, similar to linear-quadratic Gaussian control (LQG) for the robot resulting from the solution, and also a predicted linear feedback law for all other agents

4. State and control trajectory based regularization to ensure and improve convergence

5. Evaluation of the proposed method in three stochastic dynamic games: racing with autonomous vehicles, active surveillance, and guiding eyes for a blind agent

1.3.3 Learning from Competition

Deep Latent Competition: Learning to Race Using Visual Control Policies in Latent Space

Learning competitive behaviors in multi-agent settings such as racing requires long-term reasoning about potential adversarial interactions. This research thrust presents Deep Latent Competition (DLC), a novel reinforcement learning algorithm that learns competitive visual control policies through self-play in imagination. The DLC agent imagines multi-agent interaction sequences in the compact latent space based on a world model that combines a joint transition function with opponent viewpoint prediction. Imagined self-play reduces costly sample generation in the real world, while the latent representation enables planning to scale gracefully with observation dimensionality. We demonstrate the effectiveness of our algorithm in learning competitive behaviors on a novel multi-agent racing benchmark that requires planning from image observations.

Contributions While our method can be applied to a range of problems, we focus on demonstrating it in the context of two-player racing. Our approach learns a world-model for imagining competitive behavior in latent-space. This allows for training agents via imagined self-play such that they can predict opponent behavior and incorporate the expected outcomes of action sequences in their policy selection process. We further learn to predict the belief of other agents purely based on observations from the ego agent's perspective. This methodology is validated in extensive evaluations on a novel multi-agent racing benchmark based on OpenAI Gym that requires application of continuous visual control policies. Concluding, this work contains the following contributions:

1. A novel model-based reinforcement learning algorithm for learning competitive continuous control policies for multi-agent problems from high-dimensional observations such as images
2. A world-model structure that allows for estimation and imagination of competing agents' behavior in a learned latent space
3. Extensive evaluations in a new multi-agent racing benchmark demonstrating superiority over approaches that do not reason about other agents in imagination

We present the approach in Chapter 3. The work has been published in [175].

1.3.4 Guardian Angel

Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model

High-end vehicles are already equipped with safety systems, such as assistive braking and automatic lane following, enhancing vehicle safety. Yet, these current solutions can only help in low-complexity driving situations. In this work, we introduce a Parallel Autonomy, or shared control, framework that computes safe trajectories for an automated vehicle, based on human inputs. We minimize the deviation from the human inputs while ensuring safety via a set of collision avoidance constraints. Our method achieves safe motion even in complex driving scenarios, such as those commonly encountered in an urban setting. We introduce a receding horizon planner formulated as nonlinear model predictive control (NMPC), which includes the analytic descriptions of road boundaries and the configuration and future uncertainties of other road participants. The NMPC operates over both steering and acceleration simultaneously. We introduce a nonslip model suitable for handling complex environments with dynamic obstacles, and a nonlinear combined slip vehicle model including normal load transfer capable of handling static environments. We validate the proposed approach in two complex driving scenarios. First, in an urban environment that includes a left-turn across traffic and passing on a busy street. And second, under snow conditions on a race track with sharp turns and under complex dynamic constraints. We evaluate the performance of the method with various human driving styles. We consequently observe that the method successfully avoids collisions and generates motions with minimal intervention for Parallel Autonomy. We note that the method can also be applied to generate safe motion for fully autonomous vehicles.

Contributions The main contribution of this work are:

1. A formulation of Parallel Autonomy as a shared control approach between people and intelligent vehicles, which adheres to the minimal intervention principle and is able to handle complex driving scenarios
2. The development of a real-time nonlinear Model Predictive Controller (NMPC) suitable for trajectory generation in intelligent and autonomous vehicles
3. Simulation of complex traffic scenarios, such as left-turns across traffic and driving on a snowy race track, with real human inputs of different driving styles

We will employ two motion models of different complexity to handle both environments subject to slip such as on an icy road, and in difficult driving scenarios with many other drivers such as in left-turns across traffic. We introduce

- A dynamical nonlinear combined slip vehicle model including load transfer for static environments
- A kinematic model for dynamic and complex environments

We detail the method in Chapter 5, which has previously been published in [170, 171].

Chapter 2

Social Human-Robot Interactions: Social Behavior in Autonomous Driving

2.1 Introduction

Interacting with human drivers is one of the great challenges of autonomous driving. To operate in the real world, Autonomous Vehicles (AVs) need to cope with situations requiring complex observations and interactions, such as highway merging and unprotected left-hand turns, which are challenging even for human drivers. For example, over 450,000 lane-change/merging accidents and 1.4 million right/left turn accidents occurred in the United States in 2015 alone [6]. Currently, AVs lack an understanding of human behavior, thus requiring conservative behavior for safe operation. Conservative driving creates bottlenecks in traffic flow, especially in intersections. For example, even companies considered leaders in the autonomous driving industry still struggle with left turns and acting in predictable manners [58]. This conservative behavior not only leaves AVs vulnerable to aggressive human drivers and inhibits the interpretability of intentions, but also can result in unexpected reactions that confuse and endanger others. In a recent analysis of California traffic incidents with AVs, in 57% of crashes the AV was rear-ended by human drivers [186], with many of these crashes occurring because the AV behaved in an unexpected way that the human driver did not anticipate. For AVs to integrate onto roadways with human drivers, they must understand the intent of the human drivers and respond in a predictable and interpretable way.

2.2 Overview

This chapter proposes a system to measure, quantify, and predict human behavior to better inform an autonomous system. A game-theoretic formulation models driving as a series of social dilemmas to represent the dynamic interaction between drivers. We formulate a direct solution of the best response game, allowing for fast, online predictions and planning, while integrating environmental and planning constraints to ensure safety. The game's reward functions are dynamic and dependent on the vehicles' states and the environment. Since we learn the reward functions from human driving data, we expect that our approach translates to other traffic scenarios and broadly, human-robot interactions, where we can derive similar predictions trained on relevant data. Using Social Value Orientation (SVO), a common metric from psychology, we quantify human social preferences and their corresponding levels of cooperation. SVO measures how an individual weights their reward against the rewards of others, which translates into altruistic, prosocial, egoistic or competitive preferences. We estimate the human drivers' SVOs from observed motion, and set the AV's SVO based on the scenario.

2.2.1 Main Assumptions

We assume that a perception system provides state estimates for the ego vehicle and all other vehicles nearby. We also assume knowledge of lanes and the general road network. These can be furnished through either prior maps or perceived online. We assume agents to be rational and to maximize utility. Drivers account for other drivers either through experience or by explicitly incorporating how future trajectories are affected by their own and others' interests. We motivate this assumption in Section 2.6.1. Since we present a local solution algorithm for the game, we need to provide a suitable initialization. While we can compute these through computationally expensive search-based or sampling-based methods, we present a heuristic in Section 2.6.1 that works well in practice. To learn human reward functions, we assume access to a set of human demonstrations. We presume that the distribution shift between training and deployment is negligible, which is common in inverse reinforcement learning. We also assume that the learned reward functions are universal for all drivers. We do not account for personal differences in reward functions besides their social preferences in the form of SVO. We discuss SVO preferences as quantities that are not intrinsic but can change over time based on experiences or the current situation.

2.2.2 Contributions

The main contributions of this chapter are:

1. modeling driving as a dynamic game and computing its Nash equilibrium;
2. predicting human actions from expected utility maximization;
3. integrating SVO preferences into the utility-maximizing framework;
4. estimating SVO online from observed driving trajectories;
5. simulations of emerging socially-compliant autonomous driving behavior; and validation on NGSIM [67] driving data.

2.2.3 Driving as a Game

We model driving as a non-cooperative dynamic game [30], where the driving agents maximize their accumulated reward, or "payout," over time. At each point in time, the agent receives a reward, which may be defined by factors like delay, comfort, distance between cars, progress to goal, and other priorities of the driver. Figure 2-1 illustrates an example of a driving game: an unprotected left turn. Here, the blue

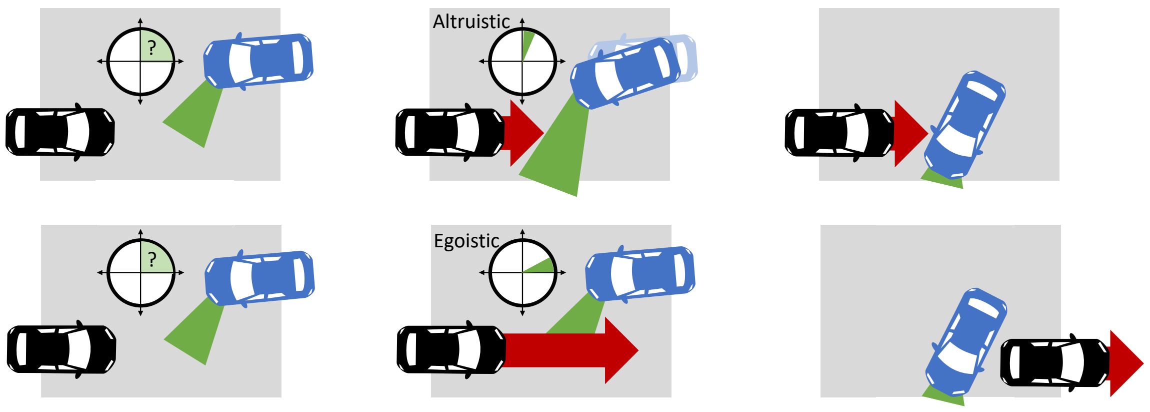


Figure 2-1: **SVO at left turn:** The autonomous car (blue) attempts to turn left across human drivers (black) in traffic. The social preferences of a driver help determine if they will yield. Here, the blue car observes the other black car and estimates its SVO. **Top:** The altruistic human driver yields and the autonomous car can safely turn. **Bottom:** If the human driver is egoistic, they will not yield and the autonomous car must wait to turn.

car must make a left turn across the path of the black car. Depending on how the interaction is resolved, the agents accrue different rewards for decisions such as comfortable braking, waiting for others to pass, or safety. In Figure 2-1, if each driver only maximizes their own reward, then the black vehicle would never brake for the blue vehicle making the unprotected left turn. However, we know human drivers often brake for others in an act of altruism or cooperation. Similarly, in highway driving, we observe human drivers open gaps for merging vehicles. If all agents were to act in pure selfishness, the result would be increased congestion and therefore a decrease in the overall group's reward. We thus conclude that driving poses a sequence of social dilemmas.

2.2.4 Social Coordination

Social dilemmas involve a conflict between the agent's short-term self-interest and the group's longer-term collective interest. Social dilemmas occur in driving, where drivers must coordinate their actions for safe and efficient joint maneuvers. Other examples include: resource depletion, low voter turnout, overpopulation, the prisoner's dilemma, or the public goods game. The autonomous control system proposed in this chapter builds on social preferences of human drivers to predict outcomes of social dilemmas: whether individuals cooperate or defect, such as opening or closing a gap during a traffic merge. It allows us to better predict human behavior, thus offering a better basis for decision-making. It may also improve the efficiency of the group as a whole through emerging cooperation, for example by reducing congestion.

2.3 Social Value Orientation

Behavioral and experimental economics shows that people have unique and individual social preferences, including: interpersonal altruism, fairness, reciprocity, inequity aversion, and egalitarianism. Self-interested models, like the *homo economicus* [42], assume agents maximize only their own reward in a game, which fails to account for nuances in real human behavior. In contrast, Social Value Orientation (SVO) indicates a person's preference of how to allocate rewards between themselves and another person. SVO can predict cooperative motives, negotiation strategies, and choice behavior [52, 126, 202, 132, 153, 5]. SVO preferences can be represented with a slider measure [134], a discrete-form triple dominance measure [204], or as an angle φ within a ring [115]. We denote SVO in angular notation, shown in Figure 2-2.

Returning to the example of Figure 2-1, SVO helps explain when the black car yields. Here, the black car considers both its reward and the reward of the blue car, weighted by SVO. As the angular preference increases from egoistic to prosocial, the weight of the other agent's reward increases, making it more likely the black car will yield. Knowing a vehicle's SVO helps an AV better predict the actions of that vehicle, and allows it to complete the turn if cooperation is expected. Without SVO it would wait conservatively until all cars cleared the intersection.

An AV needs to estimate SVO, since humans cannot communicate this directly. Instead, humans observe and estimate SVO from actions and social cues [178]. SVO preference distributions of individuals are largely individualistic ($\sim 40\%$) and prosocial ($\sim 50\%$), [27, 69, 43, 61, 134], which emphasizes that a SVO-based model will be more accurate than a purely selfish model. Figure 2-3 shows how we measure

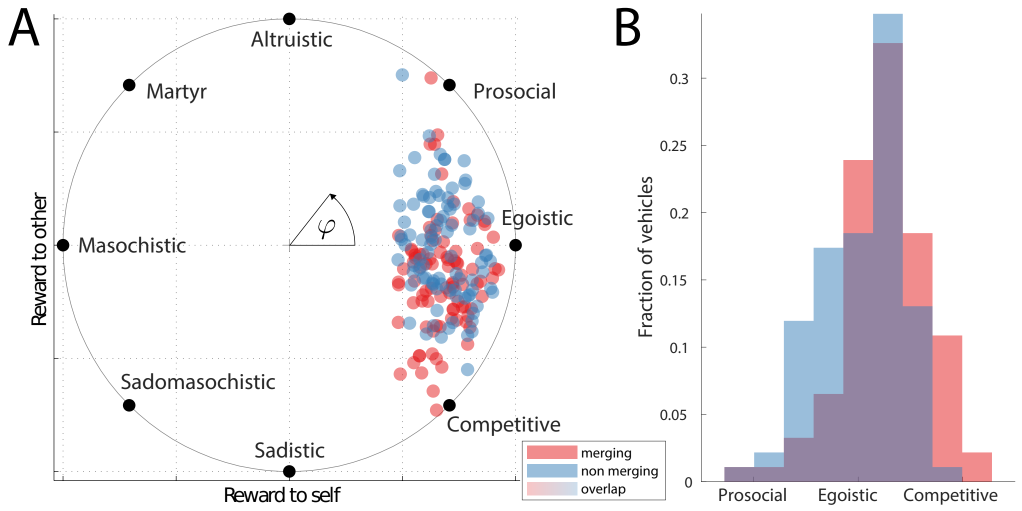


Figure 2-2: **SVO measurements from driving data: (A)** SVO is represented as an angular preference φ that relates how individuals weight rewards in a social dilemma game. Here, we plot the estimated SVOs for drivers merging in the NGSIM data set. **(B)** The distribution of mean SVO estimates during interactions. We find merging drivers (red) to be more competitive than non-merging drivers (blue).

SVO from observed motion of other drivers. We estimate SVOs of other drivers by determining the SVO that best fits predicted trajectories to the actual driver trajectories. This technique enables the estimation and study of SVO distributions of agent populations directly from trajectory data, extending beyond driving. We plot the estimated SVOs for drivers merging in the NGSIM data set in Figure 2-2.

2.4 Socially-Compliant Driving

Using SVO estimates of human drivers, we can design the control policy of the AV. We define *socially-compliant driving* as behaving predictably to other human and autonomous agents during the sequence of driving social dilemmas. Achieving socially-compliant driving in AVs is fundamental for the safety of passengers and surrounding vehicles, since behaving in a predictable manner enables humans to understand and appropriately respond to the AV's actions. To achieve socially-compliant driving, the autonomous system must behave as human-like as possible, which requires an intrinsic understanding of human behavior as well as the social expectations of the group.

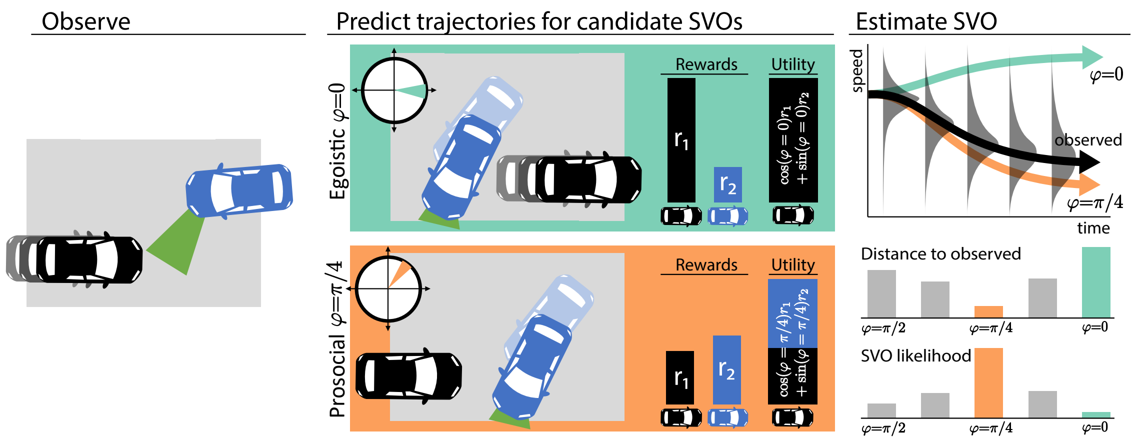


Figure 2-3: **Measuring SVO from observations:** Knowing a driver's SVO helps predict their behavior. Here, the AV (blue) observes the trajectories of the other human driver (black). We can predict future motion of the black vehicle for candidate SVOs based on a utility-maximizing decision model. If the human driver is egoistic, they will not yield and the AV must wait to turn. If the human driver is prosocial, they will yield and the AV can safely turn. In both cases, the driver is utility-maximizing, but the utility function varies by SVO. An egoistic driver considers only its own reward in computing its utility. A prosocial driver weights its reward with the reward of the other car. The most-likely SVO is the one that best matches a candidate trajectory to the actual observed trajectory. The AV predicts future motion using the most-likely SVO estimate.

Human behavior may be imitated by learning human policies from data through Imitation Learning [159, 86]. Our autonomous system design enables social compliance by learning human reward functions through *Inverse Reinforcement Learning* (IRL) [98, 223, 164]. The imitating policy is the expectation of human behavior based on past observed actions, capable of predicting and mimicking human trajectories. Combined with SVO this enables an AV to behave as a human driver is expected to behave in traffic scenarios, such as acting more competitively during merges, and mirroring the utility-maximization strategies of humans with heterogeneous social preferences in social dilemmas [133].

When designing a cooperative AV, it may be desirable to assign the AV a prosocial SVO. Prosocials exhibit more fairness and considerateness compared to individualists [52], and engage in more volunteering, pro-environment, pro-community, and charitable efforts [126, 70, 203, 57]. They also tend to minimize differences in outcomes between self and others (inequality aversion, egalitarianism) [134, 202]. Additional findings suggest reciprocity in SVO and resulting cooperation [133, 4, 130].

To make the unprotected turn in Figure 2-1 and 2-3, the AV first observes the trajectory of the oncoming car, which can be done with onboard sensors. Using the reward (payoff) structure learned from data and our utility-maximizing behavior model, it generates candidate trajectories based on possible SVO values. The most likely SVO is the one that best matches a candidate trajectory to the actual observed trajectory. With this estimated SVO, the AV then generates future motion predictions and plans when to turn safely.

2.5 Estimating Driver Behavior with SVO

Our approach integrates SVO into a non-cooperative dynamic game, and we model the agents as making utility-maximizing decisions, with the optimization framework presented in Section 2.6. To integrate SVO into our game-theoretic formulation, we define a utility function $g(\cdot)$ that combines the rewards of the ego agent with other agents, weighted by the ego agent's SVO angular preference φ. For a two-agent game,

$$g_1 = \cos(\varphi_1)r_1(\cdot) + \sin(\varphi_1)r_2(\cdot), \tag{2.1}$$

where r_1 and r_2 are the "reward to self" and "reward to other," respectively, and φ_1 is the ego agent's SVO. We see that the orientation of φ_1 will weight the reward r_1 against r_2 based on the ego agent's actions. The following definitions of social preferences [115, 134] are based on these weights:

Altruistic: Altruistic agents maximize the other party's reward, without consideration of their own outcome, with $\varphi \approx \frac{\pi}{2}$.

Prosocial: Prosocial agents behave with the intention of benefiting a group as a whole, with $\varphi \approx \frac{\pi}{4}$. This is usually defined by maximizing the joint reward.

Individualistic/Egoistic: Individualistic agents maximize their own outcome, without concern of the reward of other agents, with $\varphi \approx 0$. The term egoistic is also used.

Competitive: Competitive agents maximize their relative gain over others, i.e. $\varphi \approx -\frac{\pi}{4}$.

We limit our definitions to rational social preferences, with more in [134, 115]. While our definitions give specific values of SVO preferences for clarity, we also note that SVO exists on a continuum. For example, values in the range $0 < \varphi < \frac{\pi}{2}$ all exhibit a certain degree of altruism. We denote **cooperative actions** as actions that improve the outcome for all agents. For example, two egoistic agents may cooperate if both benefit in the outcome. Prosocials make cooperative choices, as their utility-maximizing policy also values a positive outcome of others. These cooperative choices improve the efficiency of the interaction and create collective value.

Given that other drivers maximize utility, we can predict their trajectories from observations and an estimate of their SVO. The choice of SVO changes the predicted trajectories. In Figure 2-3 a prosocial SVO generates a braking trajectory prediction, while an egoistic SVO generates a non-braking trajectory. In Section 2.7.1 we compute the likelihood of candidate SVOs from evaluating the Gaussian kernel on the distance between predicted and actual trajectories. We utilize these methods to estimate SVO

Table 2.1: Social Behavior in Autonomous Driving: Main Symbols and Notation

m	number of human and autonomous agents
$\mathbf{x}_i^k, \mathbf{u}_i^k$	state and control of agent i at time k
φ	Social Value Orientation (SVO)
$\mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k)$	state transition function
$\tau = \sum_{k=1}^{N} \Delta t$	time horizon
$g_i(\mathbf{x}, \mathbf{u}, \varphi_i)$	step utility weighted by SVO of agent i
$r_i(\mathbf{x}, \mathbf{u}, \varphi_i)$	step reward of agent i
$G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i)$	accumulated utility of agent i
$c_i(\mathbf{x}, \mathbf{u}) \leq 0$	constraints
$\mathbf{u}_i^* = \arg\max_{\mathbf{u}_i} \; G_i\left(\mathbf{x}^0, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i\right)$	optimal control of agent i
λ	dual variables
$p(\varphi_i^k \| \varphi_i^{k-1}) \propto \mathcal{M}(\varphi_i^k \| \varphi_i^{k-1}, \sigma^2_{\text{SVO}})$	SVO dynamics (von Mises distribution)
$\mathbf{b}, \mathbf{H}$	gradient and Hessian of utility G

from human driver trajectories in Section 2.9.

We improve predictions of interactions by estimating SVO of other drivers online. Incorporating SVO into the model increases social compliance of vehicles in the system, by improving predictability and blending in better. For the AVs, SVO adds the capability of nuanced cooperation with only a single variable. The AV's SVO can be specified as user input, or change dynamically according to the driving scenario, such as becoming more competitive during merging.

2.6 Driving as a Game in Mixed Human-Robot Systems

To create a socially-compliant autonomous system, our autonomous agents must determine their control strategies based on the decisions of the human and other agents. This section details how we incorporate a human decision-making model into an optimization framework. We formulate the utility-maximizing optimization problem as a multi-agent dynamic game, then derive the Nash equilibrium to solve for a socially-compliant control policy.

Consider a system of m human drivers and autonomous agents, with states such as position, heading, and speed, at time k denoted $\mathbf{x}_i^k \in \mathcal{X}$, where $i = \{1, ..., m\}$ and

$\mathcal{X} \in \mathbb{R}^n$ is the set of all possible states. We denote $\mathbf{u}_i^k \in \mathcal{U}$ as the control input, such as acceleration and steering angle, of agent i and $\varphi_i \in \Phi$ as SVO preference, where $\mathcal{U} \in \mathbb{R}^n$ is the set of all possible control inputs and Φ is the set of possible SVO preferences. For brevity, we write the state of all agents in the system as $\mathbf{x} = [\mathbf{x}_1^\intercal, ..., \mathbf{x}_m^\intercal]^\intercal$, all control inputs as $\mathbf{u} = [\mathbf{u}_1^\intercal, ..., \mathbf{u}_m^\intercal]^\intercal$. The states evolve according to dynamics $\mathcal{F}_i(\mathbf{x}_i^k, \mathbf{u}_i^k)$ subject to constraints $c_i(\cdot) \leq 0$ with the discrete-time transition function

$$\mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k) = \left[\mathcal{F}_1(\mathbf{x}_1^k, \mathbf{u}_1^k)^\intercal, ..., \mathcal{F}_m(\mathbf{x}_m^k, \mathbf{u}_m^k)^\intercal\right]^\intercal . \tag{2.2}$$

The notation $\mathbf{x}_{\neg i}$ refers to the set of agents excluding agent i. For example, we can write the state vector $\mathbf{x} = [x_1^\intercal \mid x_{\neg 1}^\intercal]^\intercal$, with $\mathbf{x}_{\neg 1} = [\mathbf{x}_2^\intercal, ..., \mathbf{x}_m^\intercal]^\intercal$. The agents calculate their individual control policies $\mathbf{u}_i$ by solving a general discrete-time constrained optimization over N time steps and time horizon $\tau = \sum_{k=1}^N \Delta t$. The set of states over the horizon is denoted as $\mathbf{x}^{0:N}$, and the set of inputs is $\mathbf{u}^{0:N-1}$. To calculate the control policy, we formulate a utility function for each agent, then find the utility-maximizing control actions. The utility function is defined as a combination of reward functions $r_i(\cdot)$, as described in (2.1), and calculated from weighted features of the current state, controls, the environment, and social preference φ_i. We generalize the utility function from (2.1) to m agents as

$$g_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i) = \frac{1}{m-1} \sum_{j \neq i} \left[\cos(\varphi_i) r_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_j) + \sin(\varphi_i) r_j(\mathbf{x}, \mathbf{u}_j, \mathbf{u}_i)\right], \tag{2.3}$$

where $r_i(\cdot)$ and $r_j(\cdot)$ are the reward functions of agent i and agent j respectively. The generalized utility function essentially weights own rewards r_i with $\cos(\varphi_i)$ against the sum of all other agents rewards $r_{\neg i}$ scaled by $1/(m-1)\sin(\varphi_i)$. In simpler terms this describes how the agent i weighs own rewards against rewards of the other agents.

At a given time k, each agent i's utility function is given by $g_i\left(\mathbf{x}^k, \mathbf{u}^k, \varphi_i\right)$, and the utility $g_i^N\left(\mathbf{x}^N, \varphi_i\right)$ at the end of the horizon. The utility over the time horizon τ

is denoted $G_i(\cdot)$, written

$$G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i) = \sum_{k=0}^{N-1} g_i\left(\mathbf{x}^k, \mathbf{u}^k, \varphi_i\right) + g_i^N\left(\mathbf{x}^N, \varphi_i\right). \tag{2.4}$$

In this work, we learn the reward functions $r_i(\cdot)$ from the NGSIM driving data to approximate real human behavior, see Section 2.8.2 for more details on this approach.

2.6.1 Human Decision-Making Model

From psychology literature, we find that people are heterogeneous in their evaluation of joint rewards [202], and we can model preferences for others using utility functions that weight rewards [21, 4, 130]. Murphy and Ackermann [133] model human decision-making as expected utility-maximizing under individual social preferences. Based on these findings from behavioral decision theory, we model human agents in our system as agents that make utility-maximizing decisions. Other robotics literature [164, 98, 223] supports this case. Translating this decision-making into an optimization framework for socially-compliant behavior, we write the utility-maximizing policy

$$\mathbf{u}_i^*\left(\mathbf{x}^0, \varphi_i\right) = \arg\max_{\mathbf{u}_i}\ G_i\left(\mathbf{x}^0, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i\right). \tag{2.5}$$

The solution $\mathbf{u}_i^*$ to (2.5) also corresponds to the actions maximizing the likelihood under the maximum entropy model

$$P(\mathbf{u}_i | \mathbf{x}^0, \mathbf{u}_{\neg i}, \varphi_i) \propto \exp\left(G_i(\mathbf{x}^0, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i)\right), \tag{2.6}$$

used to learn our rewards by IRL [222, 110]. Under this model, the probability of actions $\mathbf{u}$ is proportional to the exponential of the utility encountered along the trajectory. Hence, utility-maximization yields actions most likely imitating human driver behavior, which is important for social compliance.

Although the human driver does not explicitly calculate $\mathbf{u}$, we assume our model and formulation of $\mathbf{u}$ captures the decision-making process of the human driver based on their observations, control actions, and underlying reward function $r_i(\cdot)$ of the environment. Later, we validate on the NGSIM data set that our learned model successfully predicts the actual trajectories driven by the human drivers.

2.6.2 Game-Theoretic Autonomous Control Policy with SVO

To design the control policy for the AV, note that (2.5) formulated for all m agents simultaneously defines a dynamic game [30]. Given SVO estimates for all agents and a set of constraints on the system, we solve for the optimal control policy of a vehicle, u_i^*, assuming the other agents in the system also choose an optimal policy, $u_{\neg i}^*$. For an intuition on how these dynamic games work, we first start with a Stackelberg game. An example traffic scenario that can be modeled as a Stackelberg game is cars arriving at a four-way stop, where they must traverse the intersection based on the first arrival. In the traditional two-agent Stackelberg game [210], the leader ($i = 1$) makes its choice of policy, $\mathbf{u}_1$, and the follower ($i = 2$) maximizes their control given the leader policy, $\mathbf{u}_2^*(\mathbf{u}_1)$.

While the Stackelberg game can model some intersections, in many traffic scenarios, it is unclear who should be the leader and the follower, thus necessitating a more symmetric and simultaneous choice game, which is the approach we use in this work. In the two-agent case, the follower chooses $\mathbf{u}_2(\mathbf{u}_1)$, but the leader re-adjusts based on the follower, or $\mathbf{u}_1(\mathbf{u}_2)$. This back-and-forth creates more levels of tacit negotiation and best response, such that $\mathbf{u}_2(\mathbf{u}_1(\mathbf{u}_2(\mathbf{u}_1(...))))$. This strategy removes the leader-follower dynamics, as well as any asymmetric indirect control, yielding a simultaneous-choice game.

The iterative process of exchanging and optimizing policies is also called Iterated Best Response (IBR), a numerical method to compute a Nash equilibrium [30] of the game defined by (2.5). The generalized procedure for multi-agent IBR is described in Algorithm 1.

Algorithm 1 Iterated Best Response

```
1: Initialize $\mathbf{u}, \mathbf{u}_{\text{prev}}$
2: while $||\mathbf{u} - \mathbf{u}_{\text{prev}}|| > \epsilon$ do
3:     for $i \leftarrow 1$ to $m$ do
4:         $\mathbf{u}_i^* = \arg\max_{\mathbf{u}_i} \; G_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}, \varphi_i)$
5:     end for
6:     $\mathbf{u}_{\text{prev}} = \mathbf{u}, \mathbf{u} = \mathbf{u}^*$
7: end while
8: Return $\mathbf{u}$
```

2.6.3 Constrained Multi-Agent Nash Equilibrium

A limitation of IBR is its iterative nature such that optimizing may take an unacceptable amount of steps. To make solving for the Nash equilibrium computationally tractable, we reformulate the m interdependent optimization problems as a local single-level optimization using the Karush-Kuhn-Tucker (KKT) conditions [30, 150]. We solve the locally-equivalent formulation, including all constraints, with state-of-the-art nonlinear optimizers. This preserves all safety constraints in the optimization, critical for guaranteeing safe operation and performance.

Recall that (2.5) defines a game and explicitly stating all constraints yields

$$\begin{aligned} \mathbf{u}_i^*(\mathbf{u}_{\neg i}) = \arg\max_{\mathbf{u}_i} \; & G_i(\mathbf{x}^0, \mathbf{u}_{\neg i}^*(\mathbf{u}_i), \mathbf{u}_i, \varphi_i), \quad \forall i \in m, \\ \text{s.t. } & \mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k), \\ & c_i(\mathbf{x}, \mathbf{u}_i, \mathbf{u}_{\neg i}^*(\mathbf{u}_i)) \leq 0, \end{aligned} \tag{2.7}$$

which contains m interdependent optimizations. Instead of solving for the Nash equilibrium in the iterative fashion described in Algorithm 1, we can reformulate the optimization with KKT conditions to

$$\mathbf{u}^* = \arg\max_{\mathbf{u},\lambda} \sum_{i=1}^{m} G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i), \tag{2.8}$$

$$\text{s.t. } \mathbf{x}^{k+1} = \mathcal{F}(\mathbf{x}^k, \mathbf{u}^k),$$

$$c_j(\mathbf{x}, \mathbf{u}) \leq 0, \tag{2.9}$$

$$\nabla_{\mathbf{u}_j} G_j(\mathbf{x}^0, \mathbf{u}, \varphi_j) + \lambda_j^\top \nabla_{\mathbf{u}_j} c_j(\mathbf{x}, \mathbf{u}) = 0, \tag{2.10}$$

$$\lambda_j^\top c_j(\mathbf{x}, \mathbf{u}) = 0 \ , \tag{2.11}$$

$$\lambda_j \geq 0, \qquad \forall j \in m \tag{2.12}$$

where (2.10) defines the stationarity condition, (2.11) the complementary slackness, (2.12) the dual and (2.9) the primal feasibility constraints, and λ is the vector of dual variables. The sum over G_i in the objective ensures solving for a maximum. We, therefore, avoid adding an additional constraint to enforce the negative definiteness of the Hessians to ensure the solver to yield maxima. The reformulation of the optimization enables solving with state-of-the-art nonlinear optimizers.

The Nash equilibrium yields a control law for the AV $\mathbf{u}_i^*$ as well as predicted actions $\mathbf{u}_{\neg i}^*$ for all other $m-1$ agents N time steps into the future. Based on learned reward functions and the maximum entropy model, (2.6), $\mathbf{u}_{\neg i}^*$ are also maximum-likelihood predictions. The Nash equilibrium is the predicted outcome of the driving social dilemma and mimics the negotiation process between agents.

Algorithm 2 Socially-Compliant Autonomous Driving

```
1: $\mathbf{x}^0 \leftarrow$ Update state observations of all agents
2: $\boldsymbol{\varphi}_{\neg 1} \leftarrow$ Update SVO estimation of all agents
3: $\varphi_1 \leftarrow$ Choose AV SVO
4: $\mathbf{u}^* \leftarrow$ Plan and predict for all agents (2.5) (Multi-Agent Nash Equilibrium)
5: Execute AV's optimal control $\mathbf{u}_1^*$
```

Algorithm 2 describes the integration of the Nash equilibrium computation into the overall pipeline for socially-compliant autonomous driving.

2.7 Estimating SVO from Driving Observations

In the previous sections, we established how to interactively plan for the ego agent and predict trajectories for other agents given their SVOs φ. If the autonomous vehicle does not know the other agents' SVOs, it will need to estimate this quantity.

We present two probabilistic measurement functions to estimate the likelihood of an agent's SVO and integrate both into recursive filters to achieve good estimation results. The general intuition behind the two measurement likelihood functions are:

1. Given SVOs for all agents, we can predict their trajectories by solving for Nash equilibria as described in Section 6. Proposed SVOs closer to the true SVOs will yield **trajectories closer matching to the observed trajectories**. (Section 2.7.1)

2. Using a maximum Entropy model, frequently employed in Inverse Reinforcement Learning (IRL) we can score how **close to optimality observed trajectories** are for a given SVO. A SVO closer to the true SVO will yield trajectories closer to optimalty. (Section 2.7.2)

We will now formulate the SVO dynamics and general update equations for both approaches and then discuss specific details for the prediction-based approach in Section 2.7.1 and the maximum-entropy-based approach in Section 2.7.2.

To integrate both likelihood functions into a recursive filtering framework we need to first formulate the SVO dynamics. We define the SVO transition probability as a Gaussian distribution on a circle, or more precisely, according to the von Mises distribution

$$p(\varphi_i^k|\varphi_i^{k-1}) \propto \mathcal{M}(\varphi_i^k|\varphi_i^{k-1}, \sigma_{\text{SVO}}^2). \tag{2.13}$$

The von Mises distribution is the circular analog to the normal distribution and a close approximation to the wrapped normal distribution.

We start from the classical filtering problem and formulate the nonlinear filtering equations over r state measurements instead of a single state measurement. Thus, we are interested in finding the likelihood of the SVO φ^{k-r} at time $k-r$ based on the experienced state observations $\mathbf{x}^{0:k-1}$ until time k. In the following, we formulate the predict and update step. To update our predictions about the SVO the prediction step can be defined as

$$p(\varphi^{k-r}|\mathbf{x}^{0:k-1}) = \int p(\varphi^{k-r}|\varphi^{k-r-1})p(\varphi^{k-r-1}|\mathbf{x}^{0:k-1})\, \mathrm{d}\varphi^{k-r-1}. \tag{2.14}$$

The update equation to obtain the new distribution $p(\varphi^{k-r}|\mathbf{x}^{0:k})$ by updating $p(\varphi^{k-r}|\mathbf{x}^{0:k-1})$ based on a new measurement $\mathbf{x}^{k}$ can be constructed as

$$p(\varphi^{k-r}|\mathbf{x}^{0:k}) = \frac{p(\mathbf{x}^{k-r:k}|\varphi^{k-r})p(\varphi^{k-r}|\mathbf{x}^{0:k-1})}{\int_{-\pi}^{\pi} p(\mathbf{x}^{k-r}|\varphi^{k-r})p(\varphi^{k-r}|\mathbf{x}^{0:k-1})\mathrm{d}\varphi^{k-r}}, \tag{2.15}$$

where the measurement function $p(\mathbf{x}^{k-r:k}|\varphi^{k-r})$ is evaluated over the last r state measurements $\mathbf{x}^{k-r:k}$ instead of a single state measurement $\mathbf{x}^{k-r}$ to generate a likelihood of the SVO φ^{k-r}. We have found this modification to be necessary to generate accurate SVO estimates. The measurement function istelf can be approximated as

$$\begin{aligned} p(\mathbf{x}^{k-r:k}|\varphi^{k-r}) &= \int p(\mathbf{x}^{k-r:k}, \varphi^{k-r+1:k}|\varphi^{k-r})\mathrm{d}\varphi^{k-r+1:k} \\ &= \int p(\mathbf{x}^{k-r:k}|\varphi^{k-r:k})p(\varphi^{k-r+1:k}|\varphi^{k-r})\mathrm{d}\varphi^{k-r+1:k} \\ &= \int p(\mathbf{x}^{k-r:k}|\varphi^{k-r:k}) \prod_{j=k-r}^{k-1} p(\varphi^{j+1}|\varphi^{j})\mathrm{d}\varphi^{k-r+1:k} \\ &\approx p(\mathbf{x}^{k-r:k}|\varphi^{k-r+1:k} = \varphi^{k-r}, \varphi^{k-r}) \\ &\approx p(\mathbf{x}^{k-r:k}|\varphi^{k-r}), \end{aligned} \tag{2.16}$$

where, for the sake of computational tractability, we assume only a small change in SVO over the observed horizon r, such that $\varphi_{k-r+1:k} \approx \varphi_{k-r}$. Note that the assumption of a static SVO is only made to evaluate the measurement function. In general, the SVO dynamics are modeled to change over time as $p(\varphi_i^k|\varphi_i^{k-1})$.

Next, we present our prediction-based SVO estimation in Section 2.7.1, then present our maximum-entropy-based SVO estimation in Section 2.7.2. To differentiate stochastic variables from observations, we will from here on use the notation $\hat{\mathbf{x}}$ to denote observations, and $\check{\mathbf{x}}$ to denote predicted states.

2.7.1 Prediction-Based SVO Estimation

The general idea is to observe the states $\mathbf{x}^{k-r:k}$ for r time steps in the past as measurements $\hat{\mathbf{x}}^{k-r:k}$, to find SVO estimates $\boldsymbol{\varphi} = [\check{\varphi}_1, \ldots, \check{\varphi}_m]$ for all cars that can best explain the observations. We define the measurement function as the Gaussian

$$\begin{aligned} p(\mathbf{x}^{k-r:k}|\boldsymbol{\varphi}^{k-r}) &\propto p(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}))) \qquad (2.17)\\ &\propto \mathcal{N}(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}), \Sigma_{\text{trj}}) \qquad (2.18) \end{aligned}$$

with variance Σ_{trj} as the measurement uncertainty. We predict trajectories $\check{\mathbf{x}}^{k-r:k}(\boldsymbol{\varphi}^{k-r})$, following from SVOs $\boldsymbol{\varphi}^{k-r}$ based on solving the multi-agent game-theoretic problem formulated in (2.8). Starting from an initial state $\mathbf{x}^{k-r}$ and SVOs $\boldsymbol{\varphi}^{k-r}$, solving the multi-agent game-theoretic problem yields predicted control trajectories $\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k})$ which we roll out to arrive at the predicted state trajectories $\check{\mathbf{x}}^{k-r:k+q}(\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k-r}))$ from time $k-r$ until $k+q$. The process is detailed in Algorithm 3. Inserting the predicted states from time $k-r$ to k of $\check{\mathbf{x}}^{k-r:k+q}(\check{\mathbf{u}}^{k-r:k+q}(\boldsymbol{\varphi}^{k-r}))$, i.e. $\check{\mathbf{x}}^{k-r:k}(\check{\mathbf{u}}^{k-r:k}(\boldsymbol{\varphi}^{k-r}))$, into (2.17) yields

$$p(\mathbf{x}^{k-r:k}|\boldsymbol{\varphi}^{k-r}) \propto \mathcal{N}(\mathbf{x}^{k-r:k}|\check{\mathbf{x}}^{k-r:k}(\check{\mathbf{u}}^{k-r:k}(\boldsymbol{\varphi}^{k-r})), \Sigma_{\text{trj}}). \qquad (2.19)$$

Note that the state $\mathbf{x}^{k-r}$ of the system r time steps in the past from the current time k indicates the initial state of the optimization (2.8); $\check{\mathbf{u}}^{k-r:k+q}$ therefore denotes the control trajectory propagated $r+q$ steps forward from this initial state. We predict trajectories q steps into the future beyond the current time k. This reflects the idea that human drivers plan ahead and take future accumulated reward into account. Naturally, future plans already impact the actions and states we observe in the short term.

We can now integrate the formulated measurement function into a recursive filter to estimate SVOs of drivers over time. We chose a particle filter, as described in

Algorithm 3 Prediction-Based SVO Measurement

```
1: Input: Observed states $\hat{\mathbf{x}}^{k-r:k}$, proposed SVO $\check{\boldsymbol{\varphi}}^{k-r}$
2: Output: $p(\mathbf{x}^{k-r:k} = \hat{\mathbf{x}}^{k-r:k}|\check{\boldsymbol{\varphi}}^{k-r})$
3: $\check{\mathbf{u}}^{k-r:k+q} \leftarrow$ Predict input based on $\check{\boldsymbol{\varphi}}^{k-r}$ and $\hat{\mathbf{x}}^{k-r}$, (2.8)
4: $\check{\mathbf{x}}^{k-r:k} \leftarrow$ Forward propagate $\check{\mathbf{u}}^{k-r:k}$ from initial state $\hat{\mathbf{x}}^{k-r}$ based on dynamics
5: $p(\mathbf{x}^{k-r:k} = \hat{\mathbf{x}}^{k-r:k}|\check{\boldsymbol{\varphi}}^{k-r})$  Evaluate likelihood (2.19)
```

Algorithm 4, to make the least assumptions about the posterior distribution. We expect the posterior to be multimodal, since actions may not always be interpreted unambiguously. Note that the prediction-based measurement function evaluates the likelihood for all SVOs of all drivers at once, such that a histogram filter would be too high-dimensional and cannot be applied. Future work may explore other, potentially more efficient filtering methods of estimating the SVO of other drivers.

Algorithm 4 SVO Particle Filter Update

```
1: Input: $m$ particles $\check{\boldsymbol{\varphi}}^{k-r-1}$, corresponding weights $w^{k-r-1}$,
   and observations $\hat{\mathbf{x}}^{k-r:k}$
2: Output: $\mu_\varphi, \sigma^2_{\text{SVO}}, \check{\boldsymbol{\varphi}}^{k-r}, w^{k-r}$
3: for all m particles do
4:     Sample $\check{\boldsymbol{\varphi}}^{k-r}_{[i]} \leftarrow \mathcal{M}(\check{\boldsymbol{\varphi}}^{k-r}_{[i]}|\check{\boldsymbol{\varphi}}^{k-r-1}_{[i]}, \sigma^2_{\text{SVO}})$
5:     Update $w^{k-r}_{[i]} \leftarrow w^{k-r-1}_{[i]} \times p(\hat{\mathbf{x}}^{k-r:k}_{[i]}|\check{\boldsymbol{\varphi}}^{k-r}_{[i]})$, (2.19)
6: end for
7: Normalize $w^{k-r} \leftarrow w^{k-r} / \sum_{i=1}^{N} w^{k-r}_{[i]}$
8: if $1/\sum_{i=1}^{N}(w^{k-r}_{[i]})^2 < 0.5N$ (Sample impoverishment) then
9:     Resample($\check{\boldsymbol{\varphi}}^{k-r}$, $w^{k-r}$)
10: end if
11: Compute $\mu_\varphi \leftarrow \sum_{i=1}^{N} w^{k-r}_{[i]} \check{\boldsymbol{\varphi}}^{k-r}_{[i]}$
12: Compute $\sigma^2_\varphi \leftarrow \sum_{i=1}^{N} w^{k-r}_{[i]} (\check{\boldsymbol{\varphi}}^{k-r}_{[i]} - \mu_\varphi)^2$
```

We first initialize the particles $\check{\boldsymbol{\varphi}}^0$ with random weights w^0. During the particle filter update step at time k, the particles are then perturbed in Step 3 according to the dynamics $\mathcal{M}(\varphi^k|\varphi^{k-1}, \sigma^2_{\text{SVO}})$ and scored with the measurement function $p(\mathbf{x}^{k-r:k}|\boldsymbol{\varphi}^{k-r})$ detailed above. Step 6 triggers resampling if the effective number of particles $1/\sum_{i=1}^{N}(w_{[i]}^{k})^2$ is lower than half of the total number of particles, to avoid sample degeneracy and impoverishment, a common problem in particle filters. Subsequently, we compute the particle filter statistics as the weighted mean μ_φ and weighted standard deviation σ_φ^2 of the posterior distribution in Steps 8 and 9 respectively.

2.7.2 Maximum-Entropy-Based SVO Estimation

Inspired by the maximum entropy model popular in the inverse reinforcement learning literature described in Section 2.8.2, we can treat the SVO as a parameter to be estimated and pose the measurement likelihood function as

$$\begin{aligned} p(\mathbf{x}^{k-r:k}|\varphi_i^{k-r}) &\propto p(\mathbf{u}^{k-r:k}(\mathbf{x}^{k-r:k})|\varphi_i^{k-r}) \\ &\propto \exp\big(G^i(\mathbf{u}^{k-r:k},\varphi_i^{k-r})\big)\left[\int \exp\left(G_i(\tilde{\mathbf{u}}_i,\varphi_i^{k-r})\right)\mathrm{d}\tilde{\mathbf{u}}_i\right]^{-1} \\ &\propto \exp\left(\frac{1}{2}\mathbf{b}_i^\top \mathbf{H}_i^{-1}\mathbf{b}_i\right)|-\mathbf{H}_i|^{\frac{1}{2}}(2\pi)^{-\frac{\dim(\mathbf{u}_i)}{2}}. \end{aligned} \tag{2.20}$$

Recall that G_i is agent i's utility function. We denote the Hessian as $\mathbf{H}_i$ and the gradient as $\mathbf{b}_i$, both with respect to $\mathbf{u}_i$. Since the observed controls $\hat{\mathbf{u}}^{k-r:k}$, consisting of steering and acceleration inputs are not directly observable for other cars they have to be inferred from the state trajectory $\hat{\mathbf{x}}^{k-r:k}$ and the dynamics through nonlinear optimization. The inverse of the Hessian can be computed in linear time [110] with respect to the length of the time horizon r. Nonetheless, care needs to be taken since the second order Taylor expansion, employed to make the evaluation of the partition function of the likelihood tractable, is only valid close to the true value.

In this framework, SVO likelihoods can be evaluated independently from other agents' SVOs as the trajectories in the past are fixed and no predictions into the future based on SVO candidates have to be made. Thus, we can rely on a histogram filter to fully capture multiple hypothesis over the full SVO ring without the risk of sample impoverishment of the particle filter. The filtering update process is outlined in Algorithm 5. To simplify notation, we have dropped the agent's index i. In the prediction step, Line 3 propagates the dynamics forward, distributing probability mass from each of the N histogram bins to all other histogram bins according to the dynamics $p(\varphi^k|\varphi^{k-1}, \sigma^2_{\text{SVO}})$. In the update step, line 4 distributes probability mass from each histogram bin to all other bins according to the measurement function (2.20). Finally, we summarize the posterior statistics in the mean μ_φ and variance σ_φ

Algorithm 5 SVO Histogram Filter Update

```
1: Input: $N$ discretizations $\check{\varphi}^{k-r-1}$, corresponding weights $w^{k-r-1}$,
   and observed states $\hat{\mathbf{x}}^{k-r:k}$
2: Output: $\mu_\varphi, \sigma^2_\varphi, \check{\varphi}^{k-r}, w^{k-r}$
3: Predict
4: Dynamics update $w^{k-r} \leftarrow w^{k-r-1} \times p(\check{\varphi}^{k-r}|\check{\varphi}^{k-r-1}, \sigma^2)$, (2.13)
5: Update
6: Measurement update $w^{k-r} \leftarrow w^{k-r} \times p(\hat{\mathbf{x}}^{k-r:k}|\check{\varphi}^{k-r})$, (2.20)
7: Normalize $w^{k-r} \leftarrow w^{k-r} / \sum_{j=1}^{N} w_j^{k-r}$
8: Compute $\mu_\varphi \leftarrow \sum_{j=1}^{N} w_j^{k-r} \check{\varphi}_j^{k-r}$
9: Compute $\sigma^2_\varphi \leftarrow \sum_{j=1}^{N} w_j^{k-r} (\check{\varphi}_j^{k-r} - \mu_\varphi)^2$
```

2.7.3 Discussion of SVO Likelihoods

The prediction-based method takes into account both past and *future* trajectories. The motivation is intuitive: Human drivers simultaneously plan their actions into the future and predict the actions of other agents. Therefore, they decide how to allocate resources among themselves and others beyond the current time into the future. It is reasonable to assume that it is necessary to consider the effect of imagined future plans and predictions when estimating a driver's SVO. The advantage comes at the cost of an increased computational burden since each prediction-based likelihood evaluation requires solving the game-theoretic optimization outlined in Section 3.3.5. The maximum-entropy likelihood function on the other hand is computationally efficient since no optimization is necessary. Additionally, SVO likelihoods can be evaluated independently from other vehicles, therefore scaling linearly with the number of vehicles. In the prediction-based approach, likelihoods can only be evaluated jointly over all agents' SVOs. Solving the game-theoretic formulation to predict driver trajectories scales worse than linear in the number of agents.

2.8 Learning Reward Functions for Human Behavior and Decision Making

In Section 2.6, we introduced our model of human drivers' decision making using a utility-maximizing policy. Here, we elaborate in more detail on the specifics of our model. To compute the utility function, we need an underlying reward function r_i for the human driver. In general, one can find r_i through Inverse Reinforcement Learning (IRL) [2, 110, 137, 222] by learning from human demonstrations. We briefly describe our approach of learning the reward and utility function in Section 2.8.2. While we define the reward functions as linear combinations of weighted features in our application, the concept of SVO is general enough to utilize reward functions of any form.

2.8.1 Reward Function Features in Human Driving

The agents define their utility function based on a reward function. Consistent with reinforcement learning literature [110], we define the reward functions $r_i(\cdot)$ as linear combinations of weighted features of the environment,

$$r_i(\mathbf{x}, \mathbf{u}) = \theta^\top \psi(\mathbf{x}, \mathbf{u}), \tag{2.21}$$

the weights θ scale the features $\psi_i(\cdot)$. We employ features that allow us to quantify

- **road progress**, found by projecting the driven velocity onto the road's tangent;
- **comfort**, defined by quadratically penalizing high steering and acceleration controls;
- **desired velocities** within speed limits;
- **penalizing tailgating** of other vehicles, in the form of orientation aligned Gaussians;
- **collision avoidance**, as in avoiding close lateral and longitudinal distances to other vehicles;

- **centered positions** within the lanes; and
- **road departures** for leaving the drivable space.

Figure 2-4 visualizes the cost function encoding, i.e. negative reward, of the road in the six-lane NGSIM merge scenario. The merge lane angles into the adjacent lane. This costmap has lowest values within the lanes, with higher values indicating features to avoid.

2.8.2 Maximum Entropy Inverse Reinforcement Learning

Inverse Reinforcement Learning, also referred to as inverse optimal control, is the problem of recovering an unknown reward or utility function from a Markov decision process. As discussed in Section 2.6.1, human decision-makers are reasonably modeled as utility maximizing agents. Following this direction, the maximum entropy inverse reinforcement learning model [222] models the probability of actions or controls $\mathbf{u}$ to be proportional to the exponential of the rewards encountered along the trajectory:

$$P(\mathbf{u}_i|\mathbf{x}^0, \mathbf{u}_{\neg i}, \varphi_i, \theta) = \frac{1}{Z} \exp\left(G_i(\mathbf{x}^0, \mathbf{u}, \varphi_i)\right), \tag{2.22}$$

Therefore, less rewarding actions are exponentially less likely. Here, Z is the normalization function which can be evaluated by dynamic programming [222] which poses a practical challenge due to high computational complexity. This is particularly true for long time horizons and high dimensional systems with continuous control inputs.

2.8.3 Learning Human Reward Functions from Driving Data

The following outlines how we learn the human reward function from the utility function. We use the notation $G(\mathbf{u})$ to refer to the sum of utilities g_i along the trajectory defined by $(\mathbf{x}^0, \mathbf{u})$. For the purpose of this section, we use G instead of G_i, as the process is general to all agents. Consider

$$P(\mathbf{u}|\mathbf{x}^0) = \exp\left(G(\mathbf{u})\right) \left[\int \exp\left(G(\tilde{\mathbf{u}})\right) d\tilde{\mathbf{u}}\right]^{-1}. \tag{2.23}$$

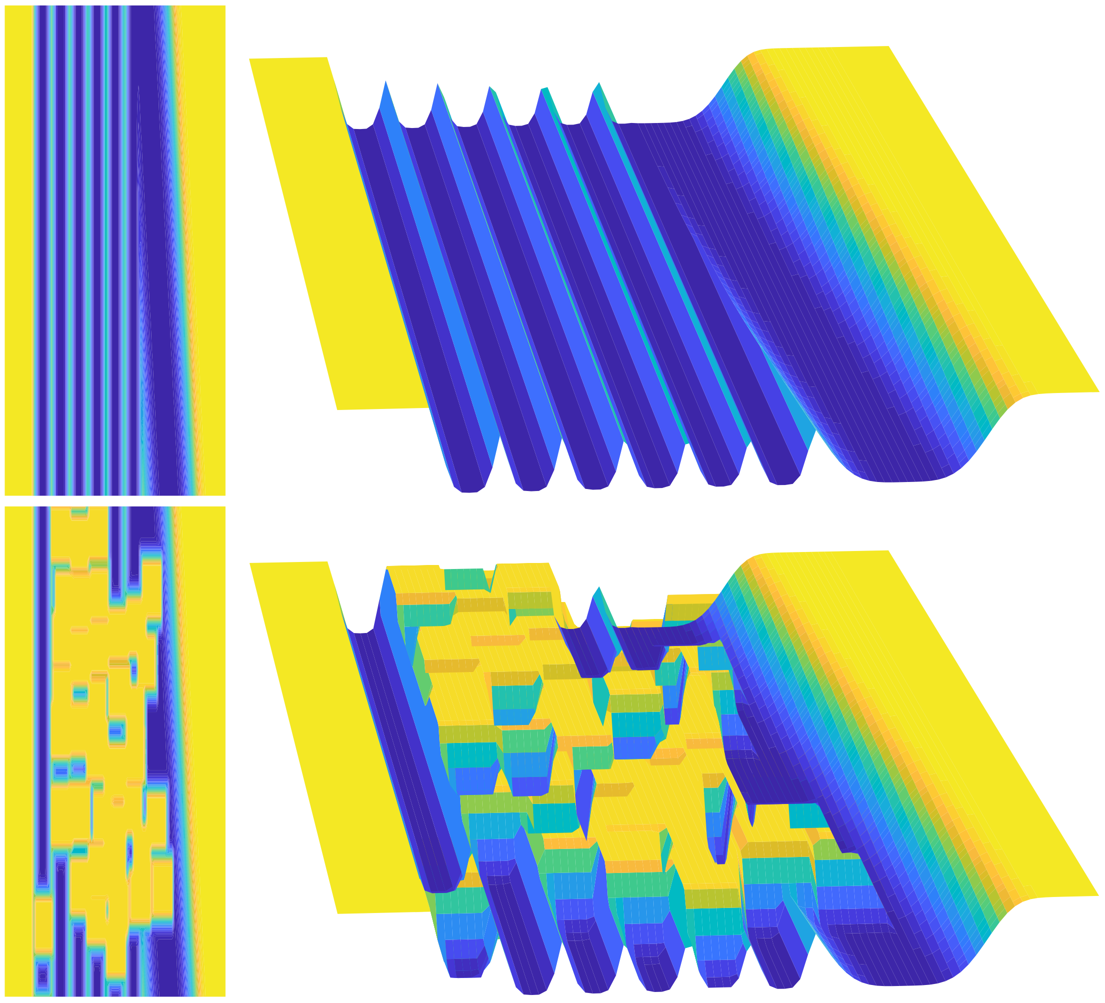


Figure 2-4: **Cost map:** An illustration of the cost map encoding the reward function (2.21) for a six-lane highway and an adjacent merge lane from the NGSIM data set. Less-desirable states incur a higher cost. **Top:** Illustration of the cost map without vehicles. Lanes are naturally encoded with a lower cost, which rewards lane-keeping. The boundaries of the highway have a significantly higher cost. The merge lane is wider and therefore the cost basin is wider as well. **Bottom:** Cost map illustrated with vehicles and obstacles. For an autonomous ego vehicle in congestion, it will avoid collisions with other vehicles by keeping to low-cost areas of the map.

To approximate the intractable normalizer, the authors in [110] apply the Laplace transform, which corresponds to assuming that the demonstration performs a local optimization when choosing the actions $\mathbf{u}$. A local approximation of the utility function as a second order Taylor expansion of $G(\mathbf{u})$ around $\mathbf{u}$ yields

$$G(\tilde{\mathbf{u}}) = G(\mathbf{u}) + (\tilde{\mathbf{u}} - \mathbf{u})^\top \frac{\partial G}{\partial \mathbf{u}} + \frac{1}{2}(\tilde{\mathbf{u}} - \mathbf{u})^\top \frac{\partial^2 G}{\partial \mathbf{u}^2}(\tilde{\mathbf{u}} - \mathbf{u}). \tag{2.24}$$

We refer to the gradient $\frac{\partial G}{\partial \mathbf{u}}$ as $\mathbf{b}$ and the Hessian $\frac{\partial^2 G}{\partial \mathbf{u}^2}$ as $\mathbf{H}$. Inserting the approximation in (2.24) into the exponent in (2.23) allows us to evaluate the integral of the normalization factor in closed form. This yields a tractable way of evaluating the likelihood including the normalization factor,

$$\begin{aligned} P(\mathbf{u}|\mathbf{x}^0) &= \exp(G(\mathbf{u})) \left[\int \exp\left(G(\tilde{\mathbf{u}})\right) \mathrm{d}\tilde{\mathbf{u}} \right]^{-1} \\ &\approx \exp(G(\mathbf{u})) \left[\int \exp\left(G(\mathbf{u}) + (\tilde{\mathbf{u}} - \mathbf{u})^\top \mathbf{b} + \frac{1}{2}(\tilde{\mathbf{u}} - \mathbf{u})^\top \mathbf{H}(\tilde{\mathbf{u}} - \mathbf{u}) \right) \mathrm{d}\tilde{\mathbf{u}} \right]^{-1} \\ &= \exp\left(\frac{1}{2}\mathbf{b}^\top \mathbf{H}^{-1}\mathbf{b}\right) |-\mathbf{H}|^{\frac{1}{2}} (2\pi)^{-\frac{\dim(\mathbf{u})}{2}}. \end{aligned} \tag{2.25}$$

The assumption of local optimality is strictly less restrictive than the assumption of global optimality. In contrast to global methods, the local method described in [110] scales well with task dimensionality and long time horizons. Furthermore, by only updating the utility function locally, only locally optimal demonstrations are sufficient. The log-likelihood of (2.25) is

$$\mathcal{L} = \frac{1}{2}\mathbf{b}^\top \mathbf{H}^{-1}\mathbf{b} + \frac{1}{2}\log|-\mathbf{H}| - \frac{\dim(\mathbf{u})}{2}\log 2\pi. \tag{2.26}$$

The learning process consists of maximizing the likelihood (2.25) for parameters θ in G and therefore $\mathcal{L}(\theta)$,

$$\theta^* = \arg\max_\theta \mathcal{L}(\theta). \tag{2.27}$$

This can be done with standard gradient and non-gradient-based optimization techniques. The resulting θ^* are the maximum entropy fit parameters best explaining the observed trajectories in the given dataset. Employing the learned parameters allows not only to observe the utility best explaining the observed behavior but also to predict and replicate human driver trajectories by optimizing their utility over their future actions. The result is a learned utility-maximizing prediction of human behavior.

In practice, additional regularization schemes to ensure convergence and invertibility of the Hessian $\mathbf{H}$ are needed. A method of solving for the computationally challenging Hessian inversion in linear-time under the restriction of linearized dynamics is described in [110].

Lastly, we need to learn the reward parameters, as described above, and iteratively estimate SVO. This is needed as SVO estimation is dependent on the rewards, and the reward learning is dependent on the SVO weighted utility. In practice, we train the rewards with fixed egoistic SVO until convergence and then continue by iterating between the two mechanisms.

2.9 Methods and Results

We implement our socially-compliant driving algorithm in two ways: first to predict human driver behavior in highway merges, then in simulations of autonomous merging and turning scenarios. We evaluate human driver predictions on the NGSIM data set and examine highway on-ramp merges into congestion. We analyze a total of 92 unique merges from the data set and discuss key results on a representative example. Incorporating SVO reduces errors in trajectory predictions of human drivers by up to 25%. For the AV simulations, we replicate this merging scenario, and also present an unprotected left turn. Our simulations demonstrate how utilizing SVO preferences assists the AV in choosing safe actions, adding nuanced behavior and cooperation with a single parameter.

2.9.1 NGSIM Data-Set Analysis and Validation

To validate our model with human driving data, we used the Next Generation Simulation (NGSIM) data set, provided by the US Department of Transportation and Federal Highway Administration[1]. The NGSIM data set comprises four highway and city traffic scenarios from California and Atlanta.

Our results are computed on the Interstate 80 Freeway Dataset[2], which captures the eastbound traffic in April 2005 during rush hour. Vehicle tracker data is provided for 500 meters of the freeway. The freeway has six traffic lanes, with the leftmost lane being a high-occupancy vehicle (HOV), and an on-ramp for merging traffic. In total, there is approximately 45 minutes of trajectory data, with a resolution of 10 frames per second. We focus on this sample due to the number of interactions that occur during highway driving and merging.

Due to errors and noise in the data set, we pre-processed the vehicle trajectories before performing the estimations and predictions detailed in this chapter. For each vehicle, we filter for noise in local frames, smoothing the trajectories. We check for errors in the data set that arise from tracking errors, or in some cases, mis-attribution of vehicle id that results in duplicate or deleted cars.

The heading of each vehicle is not included in the raw data, and is instead extrapolated from the filtered trajectories. We also recalculate all velocities and accelerations from the filtered quantities. In comparisons between the "predicted" and "actual" trajectory, the actual trajectory is this processed data, not the raw data provided in the data set. Quantities such as vehicle class and size are taken directly from the data set.

To generate our trajectory predictions, the ego car computes a reward function that includes as a component the road network geometry. Lane geometry is not explicitly given in the raw data, thus we have reconstructed the lanes based on trajectories. When generating the reward function for our trajectory predictions, we re-generate lanes matched to the road network of the data set. An exemplary cost map is shown

[1]Available online at: https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm
[2]https://www.fhwa.dot.gov/publications/research/operations/06137/index.cfm

in Figure 2-4.

2.9.2 Vehicle Model

We use a simplified car model for the vehicle dynamics, with state $\mathbf{x}_i = [x_i, y_i, \phi_i, \delta_i, v_i]^T$ consisting of position x_i and y_i, orientation ϕ_i, steering angle δ_i and speed v_i. The control inputs are acceleration $u_{i,\text{acc}}$ and steering angle velocity $u_{i,\text{steer}}$. The continuous-time dynamics are given by

$$\underbrace{\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \\ \dot{\delta}_i \\ \dot{v}_i \end{bmatrix}}_{\dot{\mathbf{x}}_i} = \begin{bmatrix} v_i \cos(\phi_i) \\ v_i \sin(\phi_i) \\ \frac{v_i}{L_i} \tan(\delta_i) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} u_{i,\text{steer}} \\ u_{i,\text{acc}} \end{bmatrix}}_{\mathbf{u}_i}, \tag{2.28}$$

and define a discrete time model by integration $\mathbf{x}_{k+1} = \mathbf{x}_k + \int_k^{k+\Delta t_k} \dot{\mathbf{x}}\, \mathrm{d}t = \mathcal{F}(\mathbf{x}_k, \mathbf{u}_k)$. A fourth order Runge-Kutta scheme ensures integration between timesteps k to sufficient accuracy.

2.9.3 Predicting Human Driving Behavior

To validate our algorithm, we test its ability to predict human trajectories on highway on-ramp merges in the NGSIM data set against several baselines. We implement a non-interactive baseline algorithm, where each agent computes their optimal policy while modeling other agents as lane-keeping dynamic obstacles. This baseline algorithm is analogous to current approaches in modeling multi-agent behavior without game-theoretic interactions. Using the dataset and trajectory history, we compare the baseline prediction's performance to the multi-agent game-theoretic models with (i) static egoistic SVO, equal to neglecting the SVO model, (ii) best static SVO, and (iii) estimated dynamic SVOs. The best static SVO corresponds to the best SVO estimate when holding the SVO constant throughout the interaction. For different interactions, this may yield a different static SVOs for different scenarios. The estimated SVO uses our proposed online algorithm, and a key difference from the static SVO is that the cars' SVO preference is allowed to change throughout the interaction. From the NGSIM data set, we examined 92 merge scenarios and compared the performance across all scenarios. Here, we predict the trajectories of the cars throughout the merge and compute the mean squared error (MSE) along the 3s prediction horizon. Table 2.3 examines the relative position error between the true vehicle trajectory and our predictions. Table 2.2 details absolute errors. We find that incorporating the multi-agent game-theoretic framework, but remaining egoistic, alone improves performance by 5%. Highlighting the importance of SVO, we see an 18% improvement over the baseline with static SVO and 25% with estimated dynamic SVO with prediction-based measurement function. The maximum-entropy-based measurement function achieves an error reduction of 20% over the baseline. We observe that the prediction-based measurement performs 5% better than the maximum-entropy-based measurement which supports our argument that people's future plans and predictions influence their current actions. We can conclude that while the multi-agent game-theoretic approach improves prediction results, the combination of multi-agent game-theoretic and dynamically estimated SVO results yields the largest improvement. The fact that the static best SVO is nearly 13% better than the egoistic SVO highlights the general impact of SVO preferences in human decision-making.

Prediction	baseline	multi-agent game-theoretic			
SVO	-	egoistic	static best	estimated	
meas. likelihood	-	-	-	prediction	max-entropy
MSE position	1.0	0.947	0.821	**0.753**	0.793

Table 2.3: Relative position MSE between predicted and actual trajectories, compared to a single-agent baseline. Our multi-agent game-theoretic model reduces error, and the dynamic, estimated SVO performs best.

Prediction	baseline	multi-agent game-theoretic			
SVO	-	egoistic	static best	estimated	
meas. likelihood	-	-	-	prediction	max-entropy
MSE position $[m^2]$	1.559	1.476	1.279	**1.174**	1.236
MSE longitudinal $[m^2]$	1.451	1.370	1.170	**1.074**	1.135
MSE angle $[rad^2]$	0.149	0.139	0.139	**0.136**	0.138
MSE speed $[(m/s)^2]$	0.988	0.943	0.827	**0.803**	0.813

Table 2.2: Absolute mean squared error (MSE) over the prediction horizon during 92 interactive merges on the NGSIM dataset.

To visually illustrate the accuracy of the trajectory predictions, Figure 2-5 compares ground truth trajectories (blue) with the predicted trajectories using our dynamically estimated SVO (red dashed). We estimate the SVOs with the prediction-based measurement function. We also show predicted trajectories for varying static SVOs (black dotted) over several merges. Each subfigure in these figures represents a single merge example taken from the data set. The estimated SVO trajectory closely follows the ground truth trajectory. By only changing the SVO, a single parameter, we can account for a wide variety of trajectories and behaviors. For example, in the first row predictions flare out to the left and right side, depending on the SVO value. In contrast, the predictions based on a dynamically estimated SVO, displayed in red, follow the ground truth trajectories very accurately.

Figure 2-6 shows a two-agent merge with car 1 (purple) merging into the lane with

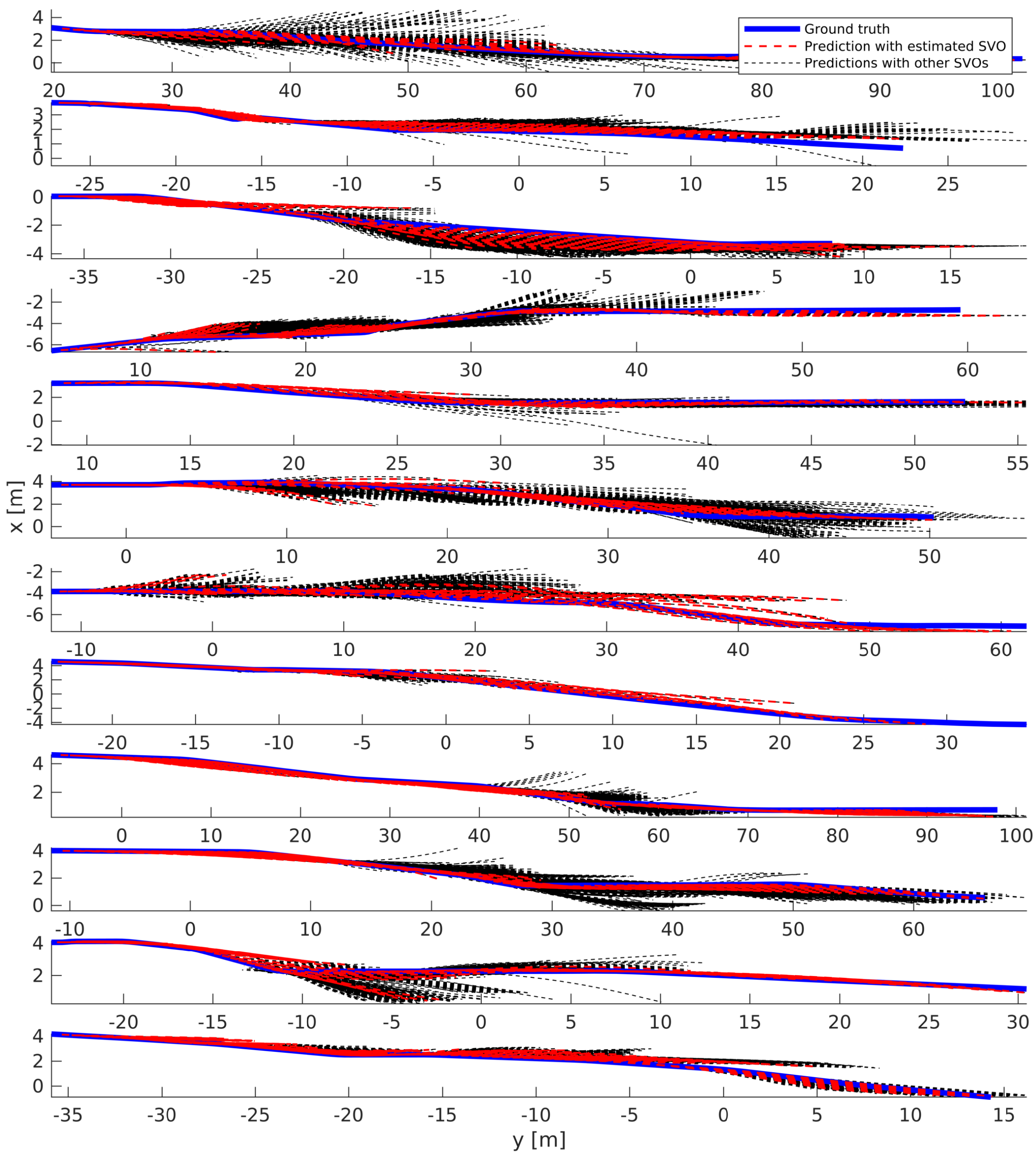


Figure 2-5: **Predicted and ground truth trajectories during lane-merging:** Displayed are ground truth trajectories (blue), the predicted trajectories using our estimated SVO (red dashed) with the prediction-based measurement function, and the predicted trajectories with varying static SVOs (black dotted) for a number of merge scenarios. Deviation of ground truth and varying static SVO predictions are largest during lateral motion of the merge, i.e. when the actual merge is executed. The predictions based on the estimated SVO are able to follow the ground truth trajectories very closely.

car 2 (green). We model the other cars in the data set as obstacles for the planner. For a dynamic SVO prediction, we estimate SVO online from observed trajectories of the vehicles with the prediction, then leverage SVO in predicting the trajectory. Figure 2-6 shows SVO predictions and confidence bounds for both cars through the merge for both prediction and maximum-entropy-based measurement functions. Our SVO estimates help explain the interactions occurring: At $t = 2$, the first car's SVO is egoistic while attempting to merge, but the second car is also egoistic and does not provide a sufficient gap to merge. At $t = 5$, the second car drops back and increases the gap for merging, corresponding to a more prosocial estimated SVO. Once the first car has merged, the second car closes the gap, returning to an egoistic SVO.

2.9.4 Altruistic Driving Scenario

Unfortunately it is not possible to obtain ground truth labels for the SVO preferences of human drivers. The reason is that SVO preferences in traffic significantly change over time and even when consistent are hard to interpret by the eye. In Figure 2-7 we illustrate the trajectory predictions of a merging vehicle overlaid onto the actual vehicle trajectory for a single merge in which the yielding vehicle behaves altruistic. They decelerate and open a gap for the ego vehicle. We compare the baseline, and multi-agent game-theoretic prediction results with different static SVOs with the estimated SVO approach. As expected, the multi-agent game-theoretic approaches track the actually executed trajectory closer than the baseline approach. The altruistic and estimated SVO trajectory predictions perform the best by closely following the ground truth trajectory.

2.9.5 Merging Drivers are More Competitive

The capability of estimating SVOs of humans by observing their motions allows us to investigate how SVO distributions in natural populations differ. Separating merging and non-merging vehicles in the dataset, we find that merging cars are more likely to be competitive than non-merging cars, shown in the histogram in Figure 2-8 and

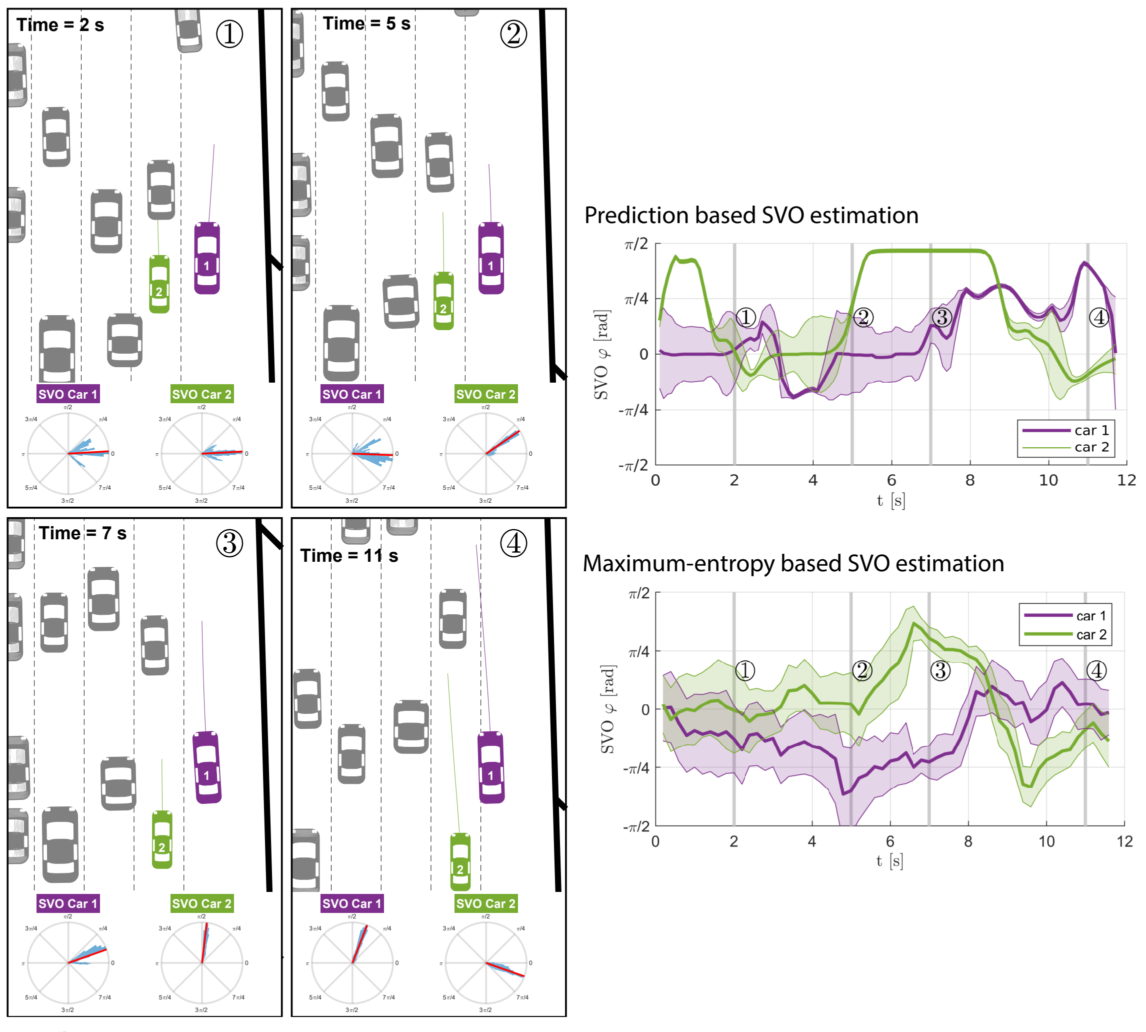


Figure 2-6: **Online SVO estimation during on-ramp merge: Top:** Snapshot of NGSIM dataset with $n = 2$ active cars (purple and green), and $n = 50$ obstacle cars (grey). Here, car 1 (purple) is attempting a merge, and must interact with car 2 (green). The solid lines indicate the predicted trajectory from our algorithm. For SVO estimates at each frame, the blue represents the distribution, while the red line indicates our estimate. **Bottom:** The solid lines display SVO estimates over time, with the shaded region representing the confidence bounds. Initially, car 2 does not cooperate with car 1, and does not allow it to merge. After a few seconds, car 2 becomes more prosocial, which corresponds to it "dropping back" and allowing the first car to merge.

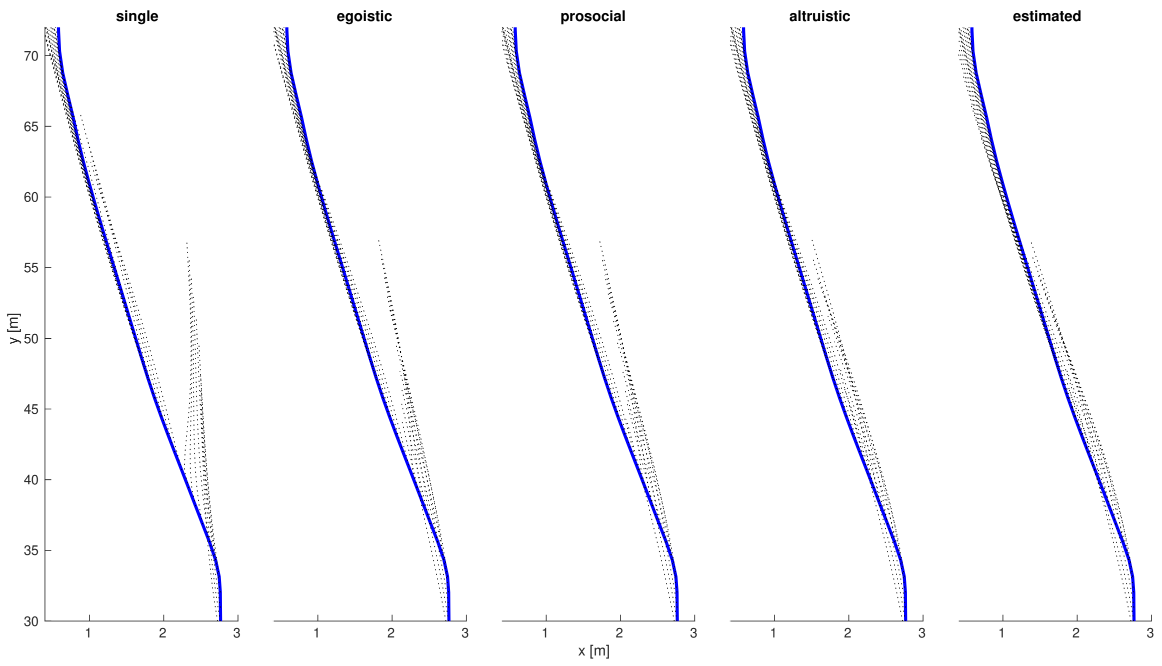


Figure 2-7: **Trajectory predictions with varying SVOs:** Trajectory predictions of a merging vehicle. The vehicle leaves a highway merge lane and moves into the left adjacent lane where another vehicle yields to allow the lane change to succeed. The merging vehicle's actual trajectory is displayed in blue, whereas the predicted trajectories over time are overlaid as dotted lines. In the leftmost plot all other vehicles are predicted with constant velocity while the merging car is predicted by a single car optimization, i.e. the baseline algorithm. All other plots to the right rely on the multi-agent game-theoretic Nash equilibrium formulation. The labels egoistic, prosocial, and altruistic refer to the SVO of the car that allows the predicted car to merge into its lane, whereas all other cars are assumed to be egoistic. Of these, the best fit is achieved if the yielding vehicle's SVO is estimated as altruistic. On the right, all SVOs of all vehicles are estimated dynamically with the prediction-based model resulting in an even closer fit. The combination of game-theoretic formulation and estimating the SVOs of other vehicles allows the vehicle to merge. Otherwise the vehicle would have acted too conservatively and avoided the merge, which we see in the "single" figure where the estimated trajectories point straight ahead when the car is actually moving laterally to the elft to merge.

the SVO ring in Figure 2-8. We employed the prediction-based measurement function during this analysis. This observation also withstands hypothesis testing with statistical significance ($p < 0.002$).

Figure 2-8 illustrates the distribution of all estimated SVO values for all measurements at each time step in all merges. Figure 2-9 plots the mean SVO estimates for each merge onto the SVO ring, with merging and non-merging vehicles indicated in red and blue, respectively. The radius of each observation shows the consistency of each mean SVO estimate during a merge, computed from the variance of SVO measurements during the respective merge. We find from these distributions that merging vehicles show more competitive behavior than the non-merging vehicles, which exhibit more cooperative and prosocial SVO preferences.

We can test for the statistical significance of this observation under the null-hypothesis that the mean SVO of merging cars is higher than that of non-merging cars. The one-sided paired t-test rejects the null-hypothesis with $p = 6.5932e - 04$. Dropping the t-test's assumption that the variables in question are normally distributed in the two groups, the non-parametric Wilcoxon signed-rank test can be applied. The non-parametric Wilcoxon signed-rank test as an alternative to the t-test also rejects the null-hypothesis with $p = 0.0018$. Since all p values are below the significance threshold $p < 0.005$ the null hypothesis can be rejected with high confidence and the alternative hypothesis holds. We conclude that merging drivers exhibit a lower SVO than non-merging drivers. Therefore, merging drivers are more competitive than non-merging drivers. While this statement is intuitive for day-to-day drivers, we can ground the intuition in objective observations of the SVO metric.

2.9.6 Autonomous Merging with SVO

Employing the estimation techniques described in Section 2.7, we can measure the SVO preferences of another agent in a simulated highway merging scenario. Figure 2-10 shows the AV's (red) SVO estimates of another vehicle (blue) over time. At first, the vehicles have little interaction, and the observations of the driver's SVO remain ambiguous, such that the estimate is inaccurate with high variance. As the

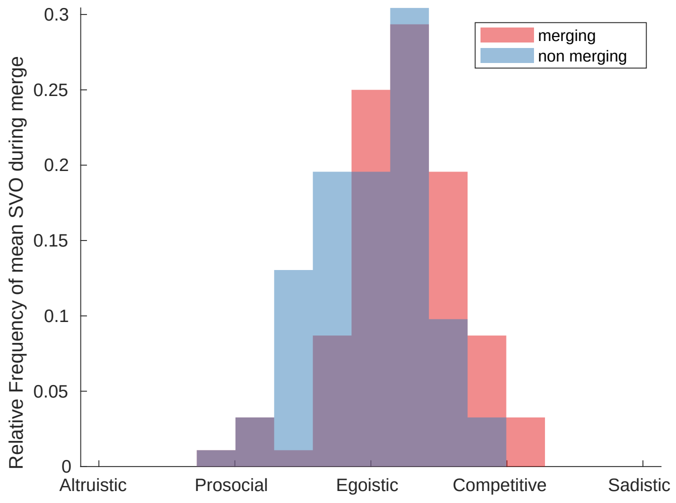


Figure 2-8: **SVO histogram in NGSIM merges:** The distribution of estimated mean SVO values for all evaluated merges. We take the mean over all measurements captured during the full length of each merge interaction. Merging vehicles show more competitive behavior, while the non-merging vehicles exhibit more prosocial or even cooperative behavior. The histogram of merging vehicles is displayed in red. The histogram of non-merging vehicles is shown in blue. We show both with 50% opacity such that the overlap appears in purple.

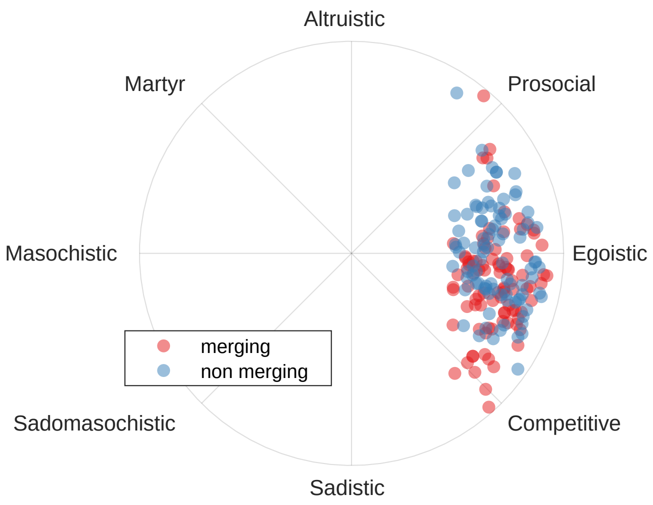


Figure 2-9: **SVO distribution on the ring in NGSIM merges:** Mean SVO estimates for each scenario shown on the SVO circle. Merging (red) and non-merging (blue) data points are shown separately. The radius of each measurement corresponds to the consistency of the SVO measurements during the respective merge. We observe that merging drivers show more competitive behavior.

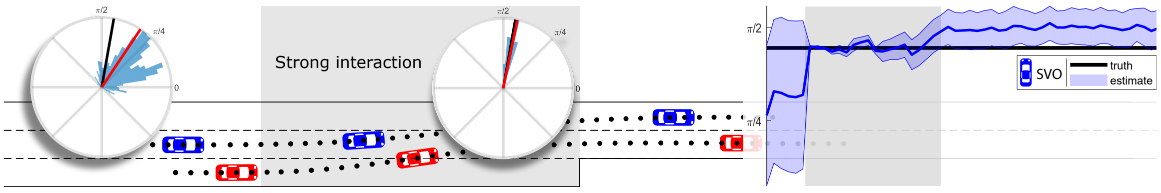


Figure 2-10: **Estimating SVO in simulation: Left:** Estimated distribution of SVO preference of blue car shown as polar histograms in SVO circles for pre, and during merge. **Right:** The mean estimate is shown as red, the ground truth (80°, altruistic) in black. SVO estimates with 1-σ uncertainty bounds shown on the right. Area of strong interaction corresponds to gray area on both sides.

AV approaches the end of its lane, both vehicles begin to interact, indicated in gray in the figure. During this time, the SVO estimate quickly converges to the true value, with high confidence. After the merge, the vehicles no longer interact, and the variance of the SVO estimate increases and the estimate drifts away from the true value. Note that estimating the characteristics of an interaction (e.g. SVO) is only possible if the interaction between agents is impactful; see Section 2.9.10 for a Hessian-based analysis.

2.9.7 Unprotected Left Turns

In this scenario, the AV must make an unprotected left turn against numerous cars traveling in the oncoming direction. If the AV were in light traffic, it could be feasible for it to wait for all other oncoming cars to pass. However, in congested traffic, the intersection might never fully clear. Instead, the AV must predict when an oncoming car will yield, allowing the vehicle to safely make the turn. Figure 2-11 shows our simulation, where the AV (red, $i = 1$) attempts to turn across traffic. Two egoistic cars (blue, $i = 2, 3$) approach the intersection and do not yield for the AV, as predicted. An altruistic third car (magenta, $i = 4$) yields for the AV by slowing down, such that the gap between itself and the other blue car increases. With this increased gap, the AV is able to safely make the turn, and the magenta car continues forward.

2.9.8 Multi-Vehicle Highway Merging

While we have showcased scenarios for interactive predictions with two cars before, in the following experiments we provide examples of highway merging scenarios with interactive predictions for four vehicles. Here, an autonomous vehicle must merge into the adjacent lane of traffic before their lane ends. Figures 2-12 and 2-13 illustrate the difference between egoistic and prosocial behavior, respectively. For each scenario, the autonomous vehicle is car ($i = 1$), shown in red, and must attempt to merge onto the highway around three other vehicles. In Figure 2-12, the magenta car ($i = 4$) is an egoistic agent and we thus do not expect them to accommodate the autonomous

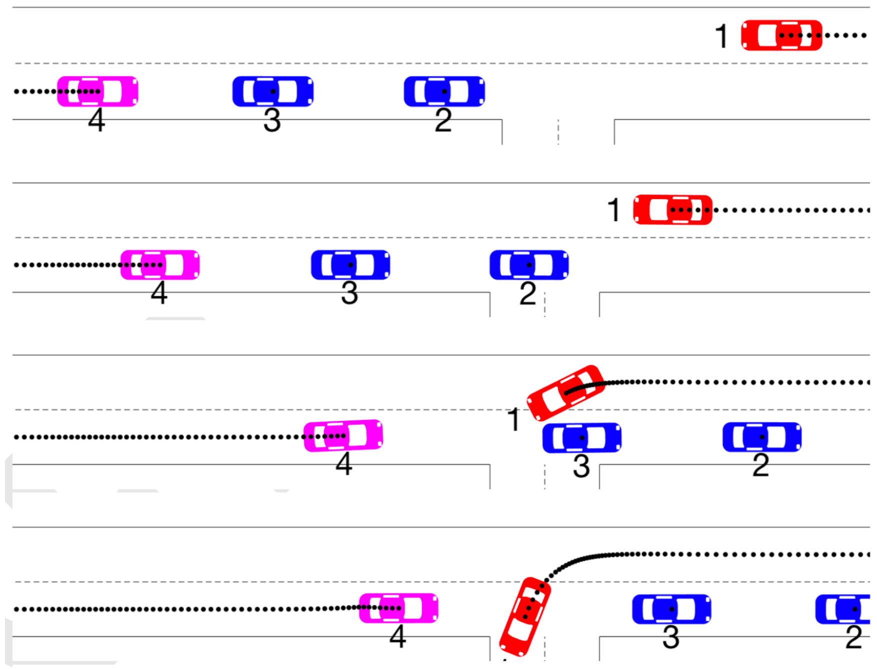


Figure 2-11: **Unprotected left turn across traffic with varying SVO:** Unprotected left turn of an AV (red, $i = 1$) with oncoming traffic. As the AV approaches the intersection, two egoistic cars (blue, $i = 2, 3$) continue and do not yield. A third altruistic car (magenta, $i = 4$) yields by slowing down, allowing the AV to complete the turn in the gap.

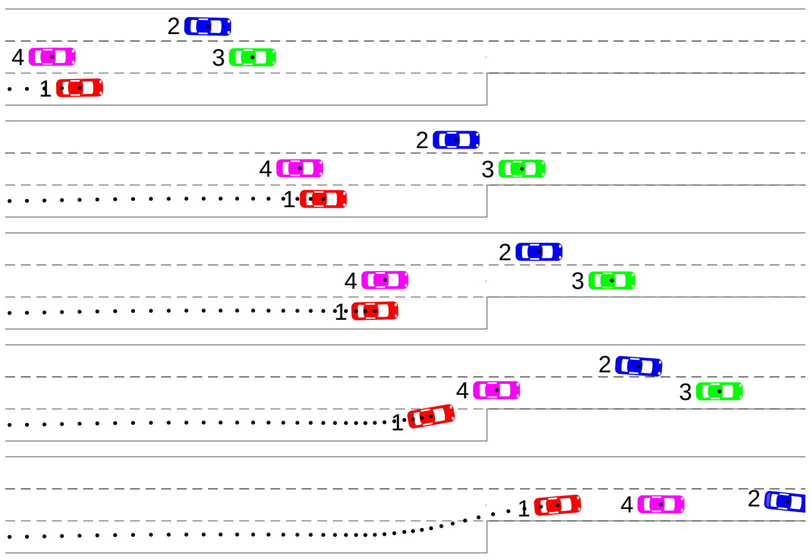


Figure 2-12: **Multi-vehicle merge among egoistic drivers:** Simulation of autonomous ego vehicle (red, $i = 1$) merging onto highway with three egoistic vehicles. The other cars do not allow the red vehicle to merge, and it must brake and wait for the other cars to pass to make the merge. In contrast, Figure 2-13, illustrates a prosocial merge.

vehicle. The egoistic agent does not make room for the autonomous vehicle, and the autonomous vehicle must slow down in order to merge after the cars have passed. Figure 2-14(a) shows the velocity profiles of the cars over time. We notice that the red vehicle must stop, whereas the egoistic vehicle actually accelerates slightly to prevent the autonomous vehicle from merging.

Conversely, Figure 2-13 illustrates the case with three other prosocial cars. In this scenario, we see the red autonomous car merge into a gap between the green ($i = 3$) and magenta ($i = 4$) cars. Examining the velocity profiles in Figure 2-14, we see that for this cooperative merge the magenta vehicle slows down. Additionally, the green vehicle also speeds up in order to increase the gap allowing the red car to merge.

2.9.9 Nash Equilibrium Solver Performance

Computation Times in Experiments

We solve the optimization problem of (2.7) in a receding horizon fashion, i.e., at any timestep k we find the optimal control policy for the autonomous vehicle incorporating

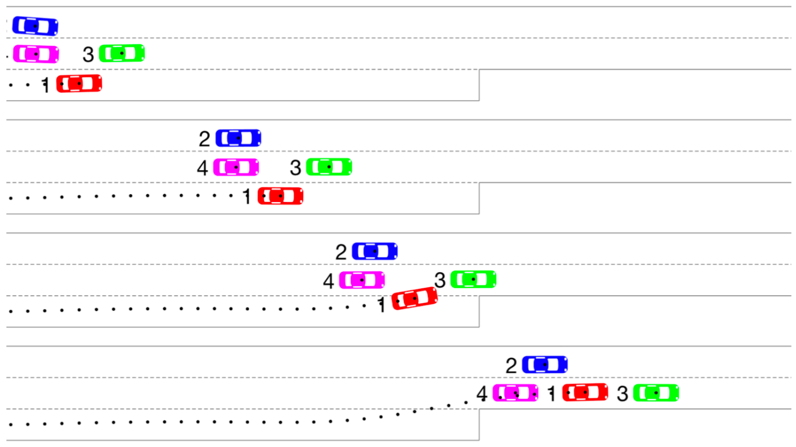
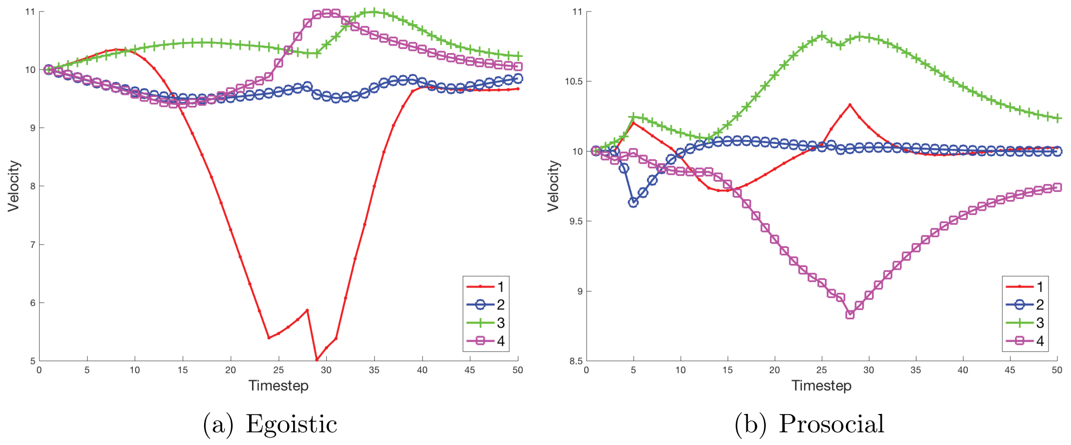


Figure 2-13: **Multi-vehicle merge among prosocial drivers:** Simulation of autonomous vehicle (red, $i = 1$) merging onto highway with three other prosocial vehicles. To allow the red vehicle to merge, the green vehicle ($i = 3$) accelerates and the magenta vehicle ($i = 4$) decelerates, increasing the gap. Here, each agent's individual reward function has penalties for acceleration and braking. Unlike Figure 2-12, because the agents are prosocial, they will slightly modify their actions in order to reduce the braking effort merging car 1.



(a) Egoistic

(b) Prosocial

Figure 2-14: **Multi-vehicle merge speed profiles:** Comparison of the velocity profiles for the (a) egoistic merge in Figure 2-12 and (b) prosocial merge in Figure 2-13. In (a), the autonomous vehicle (red) must brake and wait for the other cars to pass. In (b), the cars cooperatively increase the gap, allowing the red car to merge between them. The decelerations in (b) are smaller than the decelerations in (a), showing the flow of traffic is more smooth with the prosocial group.

the expected and observed controls of the other vehicles. The vehicle then executes the control action $\mathbf{u}^{k,*}$. We set up the optimization problem with CasADi [20], a software framework for nonlinear optimization and optimal control including automatic differentiation. We employed IPOPT [211], a widely used interior point solver, to solve the resulting nonlinear optimization problem. All experiments were conducted on a single core of an AMD Ryzen 7 1700X @3.4Ghz.

For experiments in simulation (left-turn across traffic and 3- to 2-lane highway merge), the interior point solver was capable of solving the optimization problem in less than $100ms$. The planning horizon consists of 20 steps of $\Delta t = 0.2s$, for a total horizon time of $4s$. In our experiments on congested highway driving based on the NGSIM dataset the interior point solver was capable of solving the optimization problem in less than $100ms$ for up to 4 controlled vehicles and up to 10 dynamic obstacles. A sensitivity analysis on total number of vehicles versus ego vehicle solver improvement showed that adding more cars did not have a major influence on the controlled ego vehicles. Thus, we account for the closest neighboring vehicles to the ego vehicle, which are the vehicles involved in the interaction. Furthermore, adding additional vehicles will increase the solution computation time, and may compromise real-time solutions without contributing a performance increase. We can quantify the level of interaction between vehicles by observing the norms of the Hessian blocks corresponding to the pairs of cars in the experiment. Details of this Hessian-based analysis are presented in Section 2.9.10. We observe that the norms of the Hessian blocks corresponding to cars very far away from the ego vehicle are very small, such that their level of interaction is low and they can be approximately treated as independent, which supports our argument.

KKT vs Iterative Best Response Performance Comparison

To compare the performance of the KKT-based approach (see Section 2.6.3) to the Iterated Best Response (IBR) approach to compute the multi-agent Nash equilibrium, we created a series of benchmark problems. An exemplary problem is shown in Figure 2-17. A variable number of vehicles start from randomized initial conditions (position, heading, speed) with the goal to reach randomized goal positions. The cost function consists of control costs, collision avoidance, and distance to the goal position at the final time. These examples are used to illustrate the performance variations between our two methods of solving for the optimal control policies.

Both approaches were started with an initial guess computed based on independently optimized trajectories where all other agents' inputs were set to zero, which results in constant velocity. The computation time for the initial guess was included in the reported total solve time. Both approaches were run until stationarity conditions were met sufficiently (i.e., the magnitude of the gradients of the agents' cost functions were below a given threshold). Solutions of both solvers were checked for consistency and yielded the same trajectories up to some tolerance if initialized in the region of attraction of the same homotopy class. This verifies the correctness of our KKT-based solution approach. All experiments were repeated 500 times.

The KKT-based approach was consistently faster than the iterative approach. While the solution time for the KKT-based approach was more than one order of magnitude faster on average, the difference was more prominent for shorter time horizons (10-20 times for $N < 30$) and lower for longer time horizons ($N > 40$), where N is the number of steps in the horizon. Figure 2-15 shows the relative speedup of the KKT-based approach for $m = 2$ and $m = 6$ agents, respectively. Figure 2-16 shows the solver times in seconds for both approaches across varying time horizons and number of agents.

While the IBR method approaches the Nash equilibrium without the guidance of any gradients, the KKT-based method is able to use gradients to move towards the Nash equilibrium faster. We find that as the number of interactions increases in a scenario, the greater is the performance advantage of the KKT-based approach. An explanation is that if all agents' solution trajectories are independent, then our method of computing an initial guess already yields the Nash equilibrium solution. Thus, both approaches terminate immediately, the overall solution time is dominated by computing the initial guess, and both methods are equally fast. The amount of interaction of agents can be quantified by observing the norm of the non-diagonal

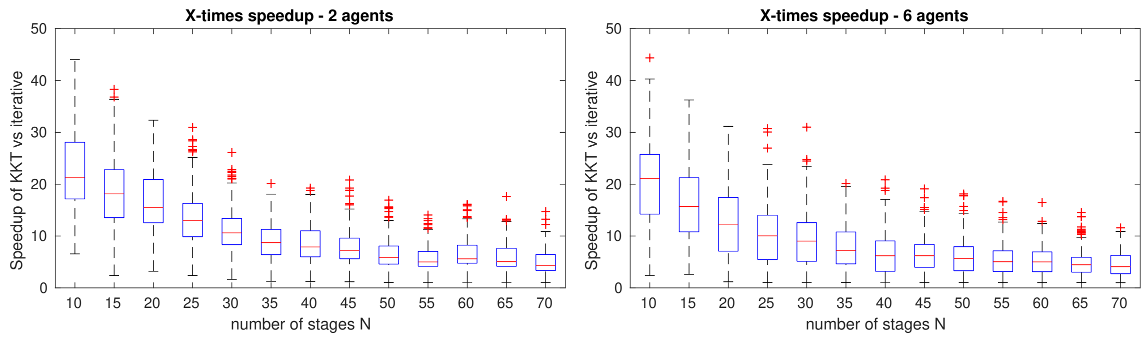


Figure 2-15: **Nash equilibrium solver speedup:** Solver performance speedup of the KKT approach over the IBR approach to solving the multi-agent Nash equilibrium problem. **TOP:** Performance speedup for $m = 2$ agents, with speedup indicated as an $X-$times factor for the KKT approach over the IBR approach across varying time horizons of N stages. **BOTTOM:** Performance speedups for $m = 6$ agents.

blocks of every agents' Hessian, described in further detail in Section 2.9.10.

2.9.10 Quantifying Interactions Between Agents

Here, we present a method of quantifying interactions between vehicles by computing and observing the properties of an agent's Hessian. The amount of interaction between agents may vary from scenario to scenario. In traffic situations, vehicles are more likely to interact when they are closer together, but a heuristic on distance alone doesn't explain all interactions. Two vehicles driving in parallel in adjacent lanes with similar speeds are close in proximity but don't necessarily need to interact if they choose to stay in their respective lanes. However, as soon as the cars need to change lanes or merge into a common lane, the interaction between the vehicles becomes much more significant.

We have presented a game-theoretic model of interactions wherein we model each agent as maximizing a utility function G_i. Here, we show that by observing an agent's Hessian, $\mathbf{H}(G_i)$, we can quickly assess which agents are interacting. The Hessian is computed as $\partial^2 G_i/\partial \mathbf{u}_i \partial \mathbf{u}_j$, which naturally encodes how one agent's utility gradient $\partial G_i/\partial \mathbf{u}_i$ (with respect to its own controls $\mathbf{u}_i$) depends on the inputs $\mathbf{u}_j$ of another

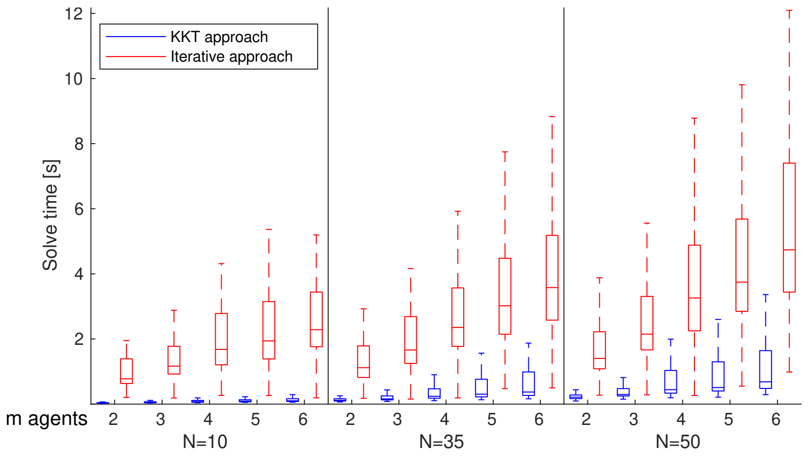


Figure 2-16: **Nash equilibrium solver performance:** Solver runtimes for both the KKT and IBR approaches for varying N-stage time horizon and $m = \{2, .., 6\}$ agents. Each approach is plotted with its median solver time and edges of the error bars indicating the 25th and 75th percentiles over the 500 trials. The KKT approach for solving the multi-agent Nash equilibrium problem is significantly faster than using an IBR solver.

agent j. Agent i's Hessian is written

$$\mathrm{H}(G_i) = \begin{bmatrix} \frac{\partial^2 G_i}{\partial u_1^2} & \frac{\partial^2 G_i}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 G_i}{\partial u_1 \partial u_m} \\ \frac{\partial^2 G_i}{\partial u_2 \partial u_1} & \frac{\partial^2 G_i}{\partial u_2^2} & \cdots & \frac{\partial^2 G_i}{\partial u_2 \partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 G_i}{\partial u_m \partial u_1} & \frac{\partial^2 G_i}{\partial u_m \partial u_2} & \cdots & \frac{\partial^2 G_i}{\partial u_m^2}, \end{bmatrix} \tag{2.29}$$

for a multi-agent game comprising m agents. Consider the multi-agent example shown in Figure 2-17, where agents move from initial locations (circles), to goal locations (crosses). Their cost function consists of control costs, collision avoidance costs, and a cost of the distance from the goal at the final time step. Figure 2-18 displays the Hessian (top) and its block-norms (bottom) for this scenario, with the color intensity encoding the magnitude of the norm. The larger the norm of the corresponding blocks, the stronger the interaction. Investigating agent 1's Hessian, we can see that the norm of $\frac{\partial^2 G_1}{\partial \mathbf{u}_1 \partial \mathbf{u}_3}$ is large (white) and $\frac{\partial^2 G_1}{\partial \mathbf{u}_1 \partial \mathbf{u}_2}$ is somewhat noticeable (gray). Since the Hessian is symmetric, their symmetric counterparts show the same values. We can deduce that agent 1 strongly interacts with agent 3, slightly with agent 2, and

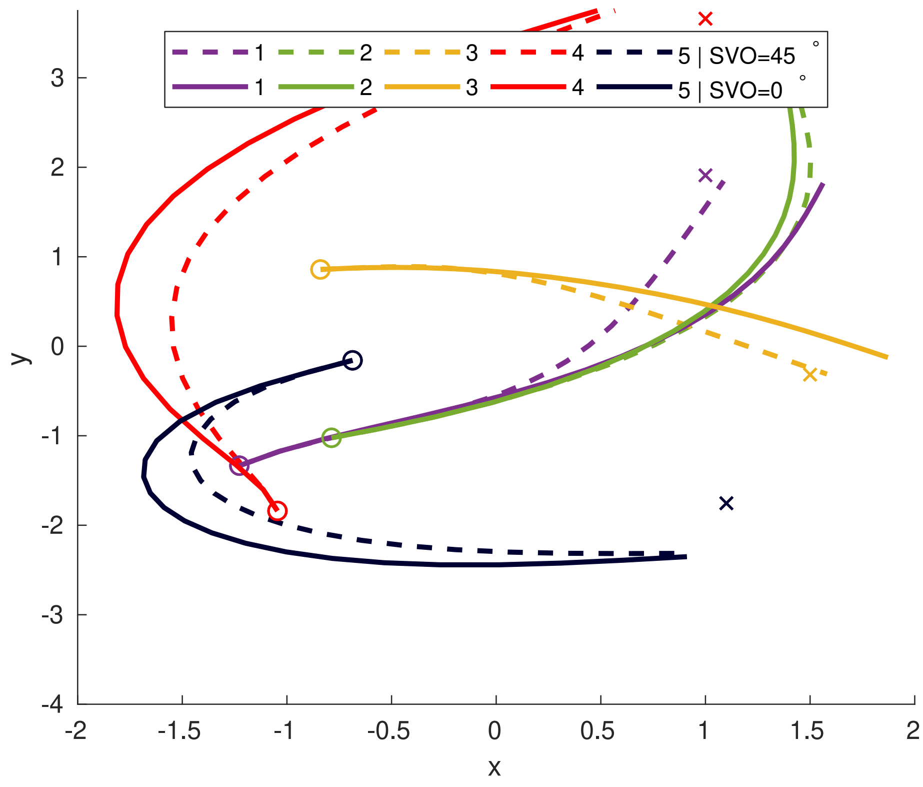


Figure 2-17: **Nash equilibrium visualization:** 5 vehicles start from their start positions and initial headings. We indicate start positions as circles and the respective goal locations as crosses. Visualized are the planned trajectories over the planning horizon. The objective function consists of control costs (acceleration and steering), collision avoidance, and distance-to-goal at the end of the planning horizon. The agents may not always reach the goal location at the end of the planning horizon due to the trade-off of control costs and distance to a goal location, as well as dynamic constraints on maximum steering angles and acceleration. We compare the influence of changing the SVO of all vehicles from egoistic ($\varphi = 0$) to prosocial ($\varphi = \frac{\pi}{4}$). We find that prosocial agents execute cooperative trajectories where some agents modify their actions to allow others to improve their performance. The overall reward increases by 24%. Interestingly, all agents improve, especially 1 and 2 since they switch sides. All other agents receive cascading improvements: 3 can move directly to its goal location and does not have to wait until 1 and 2 have passed. 4 can take a more direct path since 1 and 3 are already out of its way. Agent 5 is only very weakly influenced by the interaction and does not change its trajectory and reward.

not to a significant amount with any other agents. We can verify this observation by seeing that agent 1 and 3 are on a collision path, compare Figure 2-17, and agent 1 closely follows agent 2. If agent 1 would change its actions, agent 2 and 3 would also change their actions. Note that there is a difference between occurring costs because of their proximity and interactions (in the sense of influencing each others' actions): Although agent 1 and 4's paths are very close in the beginning, they are not actively choosing to interact. They start with an initial velocity and heading, which can not be changed too quickly, such that they do not affect each others' actions, although both agents are subject to high collision avoidance costs. Conversely, we find that agents 2 and 4 have strong interactions, as they have similar goal locations. The interaction between this pair agrees with our expectation when observing Figure 2-17. We also see that agent 5 remains independent of the other agents, which moves in the opposite direction of the other agents.

We can also re-run the scenario in Figure 2-17, except imposing that all agents exhibit a prosocial SVO value. Figure 2-19 shows the resulting Hessian and norms for this scenario. Here, we notice that there are more significant interactions across the group, indicated by the increase in brightness values across the images. Therefore, moving from a purely selfish utility model to more prosocial preferences may increase coordination among the group.

In analyzing our traffic data from the NGSIM data set, we make similar observations to the ones in this case study. Two cars driving in close proximity do not necessarily yield a strong interaction. An example of this are two cars in adjacent lanes. Only if their actions influence each other, such as if one car chooses to merge into the lane of the other car, do the interactions become apparent.

2.10 Discussion

We propose the use of Social Value Orientation to measure, quantify, and predict the behavior of human drivers. We model the interactions between drivers as a dynamic game and present a computationally-tractable way of finding its Nash equilibrium.

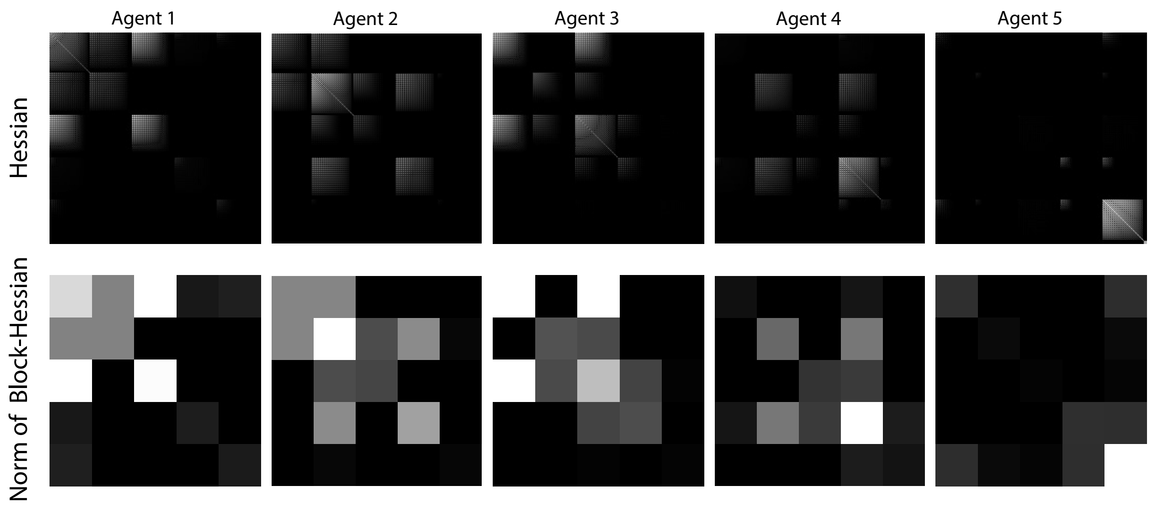


Figure 2-18: **Hessians in egoisitic scenario:** Greyscale representation of the Hessian of scenario of Figure 2-17 at the top row, and norm of block-Hessians at the bottom. Here, each pixel block corresponds to the values of the Hessian in (2.29), with brighter values indicating a higher value. Here, all agents have egoistic SVO preferences.

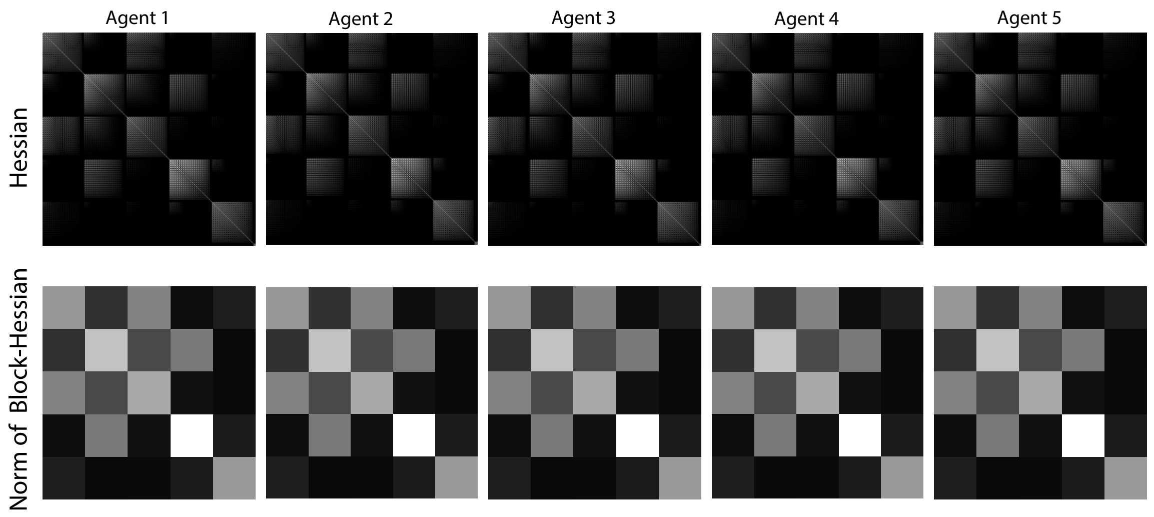


Figure 2-19: **Hessians in egoisitic scenario:** Greyscale representations of the Hessians and norm of block-Hessians of the same scenario as in Figure 2-17, but all agents exhibit prosocial SVO preferences. The prosocial SVO makes all agents' utility functions equal and the Hessians become the same.

Using SVO as our key factor in predicting human behavior, we present two likelihood functions to estimate SVO of other drivers from observed trajectories. We validate our findings in simulation and on the NGSIM data set incorporating the human behavior into the AV planner, resulting in intelligent, socially-aware maneuvers. We find that the multi-agent Nash equilibrium, SVO, as well as its estimation improve predictions and prove essential assets for interactive driving. Our unified algorithm improves on human driver trajectory prediction by 25% over baseline models. For highway merges in the NGSIM data, we also find that the human drivers merging into traffic are consistently more competitive than the drivers yielding to the merging car. These insights can better inform AVs that currently struggle to make these maneuvers. The ability to estimate SVO distributions directly from observed motion instead of in laboratory conditions will prove impactful beyond autonomous driving. Overall, robotic and AI applications where an autonomous system acts among humans will benefit from integrating SVO in their prediction and decision-making algorithms [41, 148].

Social preferences in the form of SVO, in combination with game-theoretic planning, allow us to instantaneously estimate driver personalities and open opportunities for autonomous cars that can understand human behavior and act predictably. To increase the level of reasoning about another agent's state of mind we next look at how we can incorporate uncertainty-aware and information-seeking behavior through game-theoretic belief-space planning.

Chapter 3

Reasoning over Beliefs: Stochastic Dynamic Games in Belief Space

3.1 Introduction

We aim to develop planners for multi-agent systems that are robust under uncertainty and combine information-seeking behavior with game-theoretic reasoning. While game theory can model the interaction and dependency among agents, it does not address the quality of information available to the agent for decision making. Agents must plan and act within a game, remain robust to uncertainy, gain information, and leverage the information gain to improve their control policies. We propose an approach that combines game-theoretic planning with belief-space planning, leveraging the interaction models from game theory while incorporating uncertainties in the modeled dynamics and perception. In multi-agent systems, we find that agents gather information to reduce uncertainty while maintaining decision-making strategies that support complex interactions. Applications include assistive robotics, surveillance, pursuer-evader games, and racing.

While each agent operates independently and does not reveal plans or intentions through communication, agents have approximate models of the other agents. Models of the dynamics allow to infer the ability of other agents to move in the environment, approximate cost models encode the agents' objectives, and models of how

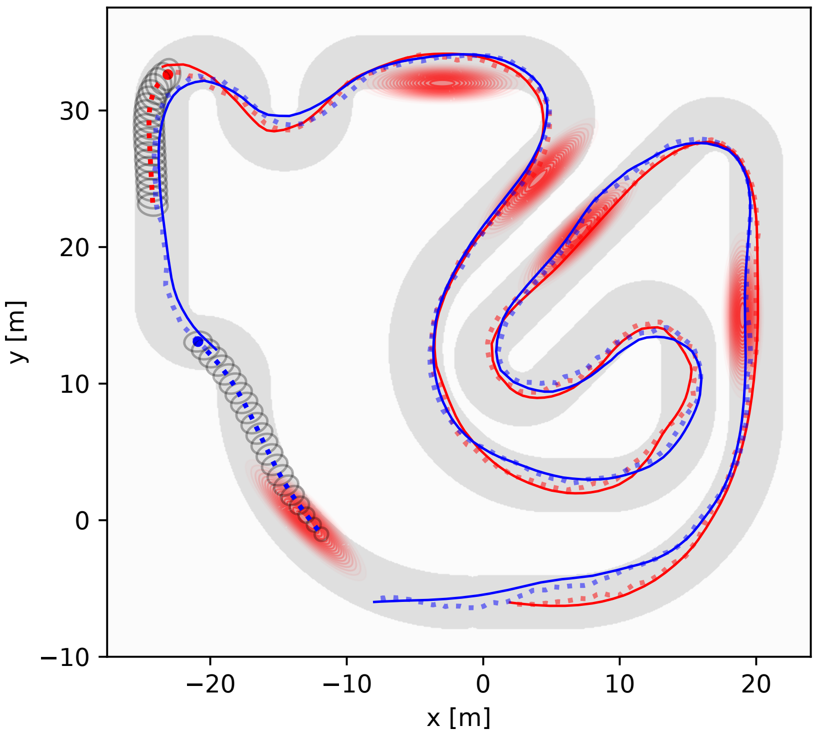


Figure 3-1: **Autonomous racing under uncertainty:** One application we present is dynamic racing. Here, the blue agent starts with a disadvantage, but is equipped with better acceleration and capable of moving faster through corners than the red agent. Our approach allows the blue agent to overtake and win the race. Planned trajectories and chance constraints are shown in dashed lines and ellipses. The traces correspond to the true state (solid) and the noisy EKF-estimate (dashed) available to each agent during the race. The red areas are zones with low noise observations and reduce uncertainty.

other agents perceive the world from observations allows estimating what other agents know and with what certainty. Our work is related to model predictive receding horizon planners: Agents start from an initial belief about themselves, others, and the environment and imagine how the future will evolve if they and other agents were to execute certain actions. These models can be prescribed, observed, sensed, or communicated. As plans are not shared among agents, we propose a game-theoretic framework that predicts the interactive policies of other agents while optimizing for our own policy.

Within the game-theoretic framework, agents take actions that increase their information gain, which in turn results in the ability to improve their control policies with reduced uncertainty. For example, an assistive robot tasked with guiding a human may explore the environment to reduce uncertainty and better navigate. Conversely, a game-theoretic setting can model adversarial agents. Here, agents may choose to "hide" to prevent others from gathering information about themselves, which is relevant to surveillance applications. In the context of racing, agents may force others to increase their uncertainty, such as by pressuring them to drive too fast in a corner which increases uncertainty in their state, or by simply pushing them into the dark. It is therefore not only important to reason about a robot's own uncertainty but also the uncertainty of other agents in the environment, and even more so how one's actions impact the change in uncertainty of others.

Game-theoretic models have not only proven useful to model interactions between autonomous systems, but also in integrating interactive human predictions into autonomous decision-making and planning. We can model the actions of humans as expected cost-minimizing and estimate human cost functions from past observed trajectories with Inverse Reinforcement Learning (IRL) [222]. Consequently, computing expected cost-minimizing actions based on the learned cost functions generates human predictions. The expected cost-minimizing behavior can also be interpreted as a best response to an autonomous agent's actions. This best response setting allows us to estimate how the autonomous system's actions influence human actions. The autonomous system can therefore implicitly control the human's actions to a certain degree. This technique has been applied to predict interactive human behavior for autonomous vehicles [163, 99, 173], and to predict pedestrians [98]. The combination of game-theoretic modeling of human behavior and information seeking planning are therefore even more promising.

For instance, home service robots can provide assistance and support to humans, particularly the elderly population. These robots need to work in close proximity of humans, gauge the human's intent, and understand the state of mind of others to better perform tasks. They have to avoid confusion and misunderstandings and will need to seek information about both their environment and surrounding humans. Additionally, these autonomous systems need to also reason about the amount of information and understanding the human has about the robot. The robot can aid the human's understanding through explicit communication, as well as implicitly through behavior, such as moving to visible locations or clearly indicating intent by unambiguously moving in a desired direction.

We propose a solution that combines multi-agent game-theoretic decision-making under uncertainty and belief-space planning (BSP). Our approach supports robust solutions to a wide range of multi-robot applications where dealing with uncertainty, the need to gather information, and game-theoretic decision-making are fundamental. We build on important advancements in two areas: game-theoretic planning and belief-space planning. Game-theoretic planning successfully solves problems where an agent's objective is at odds with the objective of other agents, such as in modeling human behavior in traffic [111, 173, 172], and leveraging the effects on humans by autonomous cars [163]. [125] gives a recent review on game theory and control. In game theory, the Nash equilibrium is a proposed solution of a non-cooperative game involving two or more players. Each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only their own strategy. Solving for Nash equilibria has been applied to competitive racing [217, 119, 184] and guiding vehicles through intersections [54]. Solution methods in racing include Iterated Best Response [217, 184, 213, 214], using discrete payoff matrices [119], or solving the necessary conditions. We will solve the necessary condition of a static quadratic game at each stage in the backward-pass of Iterative Linear-Quadratic Gaussian control (iLQG) to solve for the Nash equilibrium of the dynamic game.

While game-theoretic planning models the interaction and dependency among agents, it does not address the quality of information available to the agent for decision making. Belief-space planning [94] uses beliefs, which are the distribution of the robot's state estimate, to represent the uncertainties in the perception of the robot. The problem of computing a control policy over the space of belief states is formally described as a Partially Observable Markov Decision Process (POMDP), and has been studied extensively. Solutions to POMDPs are known to be very complex. Solving a POMDP to global optimality is NP-hard: solutions such as point-based algorithms [24, 102, 151, 78] in discrete space are bound to the curse of history, as well as sampling based solvers [40, 143, 156]. Optimization-based approaches have been developed for planning in continuous belief space [106, 142, 152, 201, 190, 157], by approximating beliefs as Gaussian distributions and computing a value function valid in local regions of the belief space. Similarly to [201, 190], we avoid the common maximum-likelihood observation assumption [152, 142]. In comparison to point-based algorithms which scale exponentially in the planning horizon l, optimization-based methods scale linearly, $\mathcal{O}(l)$.

3.1.1 Main Assumptions

To generate reasonable predictions about other agents, they build on approximate prior information. Consider the analogy of a race car driver. A driver knows that other race cars will have comparable driving characteristics, while different classes of vehicles, like trucks, will have different handling dynamics. They also know that the other racing drivers desire to go as fast as possible around the track to win the race, without crashing into other vehicles sliding off the road, similar to a cost model. Lastly, they have experience in how other drivers observe the track and know for example that the quality of perception decreases in the dark. For our autonomous systems, Assumption 1 describes that we assume *common knowledge* of models about the world.

Assumption 1 (Common Knowledge). *Agents have models for cost, dynamics, and observations of other agents.*

Related game-theoretic works, with applications ranging from pedestrian-robot interactions to autonomous racing, make similar assumptions by either prescribing dynamics and cost models [111, 217, 119, 184, 183] or learning cost models through IRL [163, 98, 173]. Assumption 1 allows agents to imagine how the future will evolve: If the robot and other agents were to execute given policies, how would model based predictions of motions and observations in the world impact the beliefs over time. We do not assume any form of direct communication between agents and therefore do not have access to the policies of other agents. Instead, we predict interactive policies of other agents through game-theory and employ the models of costs, dynamics, and observations. The robot, as part of the game, can then leverage the influence of its actions on the predicted actions of other agents to their advantage. We will show in a competitive racing example that Assumption 1 can be relaxed in practice, and that approximate models of other agents prove sufficient to improve performance.

We refer to beliefs as distributions over states, and draw inspiration over how we design our system from the cognitive theory of mind. The cognitive theory of mind [165] defines the ability to attribute mental states, such as beliefs, intents, or desires to oneself and to others. It is integral to understanding that others have beliefs that are different from one's own. Single agent planning in belief space, reasoning about the uncertainty of only the own state, is limited to zero order beliefs (e.g. I think...). In contrast, we will also reason about the uncertainty of other agents. The theory of mind refers to this as first-order belief spaces (e.g., I think they think...). Higher order beliefs such as second-order belief spaces (e.g., I think they think that I think...) are beyond the scope of this work as they quickly become computationally intractable by essentially defining beliefs over beliefs. We find that parametrizing belief spaces efficiently is crucial to generating real-time capable algorithms. Assumption 2 keeps computation complexity at a reasonable level and avoids an explosion in parameters in the recursive beliefs over beliefs.

Assumption 2 (First-Order Beliefs)**.** *Planning and prediction are limited to first-order beliefs: any robot i's belief over another agent j is the same as that agent j's belief about themselves.*

In Section 3.4 we evaluate cases, such as competitive racing, where this assumption is a simplification of the true system dynamics. In the racing scenario, all agents execute separate instances of our algorithm, and therefore maintain separate beliefs. Thus, an agent's belief about themselves does not necessarily match the beliefs that others have about them. However, while these belief mismatches may occur, we see performance improvements over a game-theoretic baseline without belief-space planning, see Sec. 3.4.3, which highlights the importance of accounting for uncertainty and information gain in competitive racing and other applications.

The purpose of Assumptions 1 and 2 is to enable interactive predictions of other agents in belief space while maintaining computational tractability. Since our approach is executed continuously in a receding horizon fashion, and we compute policies that are reactive to deviations from the predicted beliefs, the proposed method can adapt if the observed behavior differs from the predicted ones. Our approach continues to successfully control the agent under reasonable violations of the presented assumptions, such as if the dynamics, observation, or cost models are inaccurate, if their own beliefs do not exactly match the beliefs of others, or if the other agent's optimization is sub-optimal.

3.1.2 Contributions

We present a computationally-tractable solution to multi-agent planning that combines game-theoretic planning and belief-space planning to interact within a problem formulated as a game, gain information, and leverage the information gain to improve the agents' control policies. The main limiting factor in applying either game theory or belief-space planning, and even more so the combination of both to robotic control problems lies in the associated computational complexity. To the best of our knowledge this is the first work to combine general dynamic games and planning in belief space into an efficient real-time algorithm. The main contributions of this chapter are:

1. A method for computing Nash equilibria for dynamic games in belief space;
2. A linear feedback policy, similar to linear-quadratic Gaussian control (LQG), for the robot resulting from the solution, and also a predicted linear feedback policy for all other agents;
3. State and control trajectory based regularization to ensure convergence;
4. Evaluation of the proposed method in three stochastic dynamic games: racing with autonomous vehicles, active surveillance, and guiding eyes for a blind agent.

We organize the remainder of the chapter as follows: Section 3.2 introduces dynamic games in belief space, including a general definition of best response POMDPs and a Nash equilibrium formulation of the non-cooperative dynamic game. We give the resulting problem definition in Section 3.2.1, and assuming beliefs can be represented in the form of Gaussian distributions, approximate the belief dynamics based on an Extended Kalman Filter (EKF) detailed in Section 3.3.1. Our method computes a locally-optimal solution to the best response POMDP problem with continuous state and action spaces and non-linear dynamics and observation models by iteratively solving for a local Nash equilibrium, outlined in Section 3.3. We utilize a belief-space variant of iLQG to compute the Nash equilibrium, Section 3.3.3, by solving for a local Nash equilibrium at each stage of the backward pass, see Section 3.3.2. At every iteration, each agent's value function is approximated based on a quadratization around a nominal trajectory, and the belief dynamics are approximated with an extended Kalman filter. We describe regularization techniques in Section 3.3.4 to ensure that the algorithm converges regardless of initial conditions. Based on these findings, we introduce Algorithm 6 in Section 3.3.5 describing the full belief-space Nash equilibrium computation.

We show the potential of our approach in Section 3.4 by presenting three multi-agent problems that combine the information-seeking behavior with our game-theoretic

Table 3.1: Stochastic Dynamic Games in Belief Space: Main Symbols and Notation

$\mathbf{x}, \mathbf{u}, \mathbf{z}$	State, control input, and measurement
$\mathbf{b},\ \mathbf{s} = [\mathbf{b}^\top, \mathbf{u}^\top]^\top$	Belief, short for belief and controls
Q^i, V^i	Action-value and value function of agent i
π^i	Optimal control policy of agent i
$j_k,\ K_k$	Feedforward and feedback gains at time k
$c^i_k(\mathbf{b}_k, \mathbf{u}_k),\ c^i_l(\mathbf{b}_l)$	Cost of agent i at time k, and terminal cost
$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k)$	State transition with process noise $\mathbf{m}_k$
$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{n}_k)$	Measurement function with meas. noise $\mathbf{n}_k$
$\mathbf{b}_{k+1} = \beta(\mathbf{b}_k, \mathbf{u}_k, \mathbf{z}_{k+1})$	Belief transition
$\mathbf{b} = \bar{\mathbf{b}} + \delta\mathbf{b}$	Nominal + perturbation, similar for $\mathbf{u}, \mathbf{s}$
$c^i_{\mathbf{s},k},\ c^i_{\mathbf{ss},k}$	Gradiant and Hessian of c^i evaluated at $\bar{\mathbf{s}}_k$
$g_{\mathbf{s},k},\ W_{\mathbf{s},k}$	Jacobians of g and W evaluated at $\bar{\mathbf{s}}_k$
$V_{\mathbf{b},k},\ V_{\mathbf{bb},k}$	Gradient and Hessian of value at time k
$Q_{\mathbf{s},k},\ Q_{\mathbf{ss},k}$	Gradient and Hessian of action-value at k
l, N	planning horizon, number of agents

formulation: active surveillance, guiding blind agents, and racing with autonomous vehicles.

3.2 Dynamic Games in Belief Space

We first define POMDPs in their most general form (following notation of [196, 201]), then formulate the resulting game and derive the Nash Equilibrium to be solved for.

We write the belief-space planning problem as a stochastic optimal control problem. Consider a system of N agents $i \in \{1, ..., N\}$, with agent i's state at time k denoted $\mathbf{x}^i_k \in \mathrm{R}^{n_{\mathbf{x}^i}}$, measurement as $\mathbf{z}^i_k \in \mathrm{R}^{n_{\mathbf{z}^i}}$, and control input $\mathbf{u}^i_k \in \mathrm{R}^{n_{\mathbf{u}^i}}$. Here, $n_{\mathbf{x}^i}$, $n_{\mathbf{z}^i}$, $n_{\mathbf{u}^i}$ define the dimensionality of agent i's state, measurement, and control.

For brevity we refer to $\mathbf{x}_k = [\mathbf{x}^{1,\top}_k, \dots, \mathbf{x}^{N,\top}_k]^\top \in \mathrm{R}^{n_\mathbf{x}}$ as the joint state, $\mathbf{z}_k = [\mathbf{z}^{1,\top}_k, \dots, \mathbf{z}^{N,\top}_k]^\top \in \mathrm{R}^{n_\mathbf{z}}$ as the joint measurement, and $\mathbf{u}_k = [\mathbf{u}^{1,\top}_k, \dots, \mathbf{u}^{N,\top}_k]^\top \in \mathrm{R}^{n_\mathbf{u}}$ as the joint control, consisting of all agents. We refer to the joint dimensions as $n_\mathbf{x} = \sum_i n_{\mathbf{x}^i}$, $n_\mathbf{z} = \sum_i n_{\mathbf{z}^i}$, and $n_\mathbf{u} = \sum_i n_{\mathbf{u}^i}$. The notation $\neg i$ indicates all agents except i, e.g. $\mathbf{u}^{\neg i}_k$ relates to the controls of all other agents except i.

We will refer to $\mathbf{u} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{l-1}]$ as the control trajectory until time l. The joint *belief* $\mathbf{b}(\mathbf{x}_k)$ is defined as the distribution of the state $\mathbf{x}_k$ given all past control inputs and sensor measurements, and consists of individual beliefs $\mathbf{b}^i$. For brevity, we define $\mathbf{s} = [\mathbf{b}^\top, \mathbf{u}^\top]^\top$.

Following [196, 201], we compute the belief by

$$\mathbf{b}(\mathbf{x}_k) = p(\mathbf{x}_k | \mathbf{u}_0, \ldots, \mathbf{u}_{k-1}, \mathbf{z}_1, \ldots, \mathbf{z}_k), \tag{3.1}$$

from all past control inputs and sensor measurements. The stochastic dynamics and observation model, here formulated in probabilistic notation as

$$\mathbf{x}_{k+1} \sim p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{z}_k \sim p(\mathbf{z}_k | \mathbf{x}_k), \tag{3.2}$$

allow to forward propagate the belief given a control input $\mathbf{u}_k$ and a measurement $\mathbf{z}_{k+1}$ through Bayesian filtering:

$$\mathbf{b}(\mathbf{x}_{k+1}) = \eta p(\mathbf{z}_{k+1} | \mathbf{x}_{k+1}) \int p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) \mathbf{b}(\mathbf{x}_k) \mathrm{d}\mathbf{x}_k. \tag{3.3}$$

In (3.3), η is a normalizer independent of $\mathbf{x}_{k+1}$ and $\mathbf{b}(\mathbf{x}_{k+1})$ and contains the uncertainty originating from the stochastic dynamics, the uncertain measurement and the uncertainty in the belief at the previous time step. We employ the shorthand $\mathbf{b}_k$ to refer to $\mathbf{b}(\mathbf{x}_k)$. The stochastic *belief dynamics* are defined by (3.3) and are written as

$$\mathbf{b}_{k+1} = \beta(\mathbf{b}_k, \mathbf{u}_k, \mathbf{z}_{k+1}). \tag{3.4}$$

The expected return of each individual agent i under a control trajectory of all agents $\mathbf{u}$, including its own control trajectory $\mathbf{u}^i$, subject to uncertainty on the observed measurements $\mathbf{z}$ over the horizon l is determined by the action-value function

$$Q^i(\mathbf{b}_0, \mathbf{u}) = \mathop{\mathbb{E}}_{\mathbf{z}} \left[c_l^i(\mathbf{b}_l) + \sum_{k=0}^{l-1} c_k^i(\mathbf{b}_k, \mathbf{u}_k) \right]. \tag{3.5}$$

Here $c_k^i(\cdot)$ denotes the cost at time k and $c_l^i(\cdot)$ denotes the terminal cost of agent i. Since there exists an action-value function for each agent, there are N distinct action-value functions Q^i for $i \in \{1, ..., N\}$.

We will first formulate the two problems of (1) solving the general POMDP best response game, and then (2) finding the Nash equilibrium of this game.

Problem 1. ***POMDP Best Response Game:*** *Given an initial belief* $\mathbf{b}_0$*, for agents* $i \in \{1, ..., N\}$*, we need to solve the stochastic optimal control problem*

$$\pi^i = \underset{\mathbf{u}^i}{\arg\min}\ Q^i(\mathbf{b}_0, \mathbf{u}) \quad \forall i \in \{1, ..., N\} \tag{3.6}$$

$$s.t.\ \mathbf{b}_{k+1} = \beta(\mathbf{b}_k, \mathbf{u}_k, \mathbf{z}_{k+1}), \tag{3.7}$$

for each agent by minimizing each agent's expected cost with respect to their own controls $\mathbf{u}^i$*, where* $Q^i(\mathbf{b}_0, \mathbf{u})$ *is the action-value function of agent* i*.*

Note that all agents' optimal policies π^i depend on the actions of all other agents because each agent i minimizes their own action-value function $Q^i(\mathbf{b}_0, \mathbf{u})$. The result is a non-cooperative game [30] in which all agents' policies depend on the optimal policies of all other agents $\pi^i(\pi^{\neg i})$. Since all policies are optimized jointly and severally, the dependence of agent i's policy π^i on other agents' controls $\mathbf{u}^{\neg i}$ is resolved by inserting their optimal policy $\pi^{\neg i}$. We therefore denote π^i instead of $\pi^i(\mathbf{u}^{\neg i})$.

A general solution to (3.6) can be defined recursively by the Bellman equation:

$$\begin{aligned} V_l^i(\mathbf{b}_l) &= c_l(\mathbf{b}_l), \\ Q_k^i(\mathbf{b}_k, \mathbf{u}_k) &= c_k^i(\mathbf{b}_k, \mathbf{u}_k) + \underset{\mathbf{z}_{k+1}}{\mathbb{E}} \left[V_{k+1}^i(\beta(\mathbf{b}_k, \mathbf{u}_k, \mathbf{z}_{k+1})) \right], \\ V_k^i(\mathbf{b}_k) &= \min_{\mathbf{u}_k^i} Q_k^i(\mathbf{b}_k, \mathbf{u}_k), \\ \pi_k^i(\mathbf{b}_k) &= \underset{\mathbf{u}_k^i}{\arg\min}\, Q_k^i(\mathbf{b}_k, \mathbf{u}_k), \end{aligned} \tag{3.8}$$

where $V_k^i(\mathbf{b}_k)$ is the value function and $\pi_k^i(\mathbf{b}_k)$ the optimal policy at time k. Note that in (3.8) the cost $c_k^i(\mathbf{b}_k, \mathbf{u}_k)$, the reached value function $V_{k+1}^i(\beta(\mathbf{b}_k, \mathbf{u}_k, \mathbf{z}_{k+1}))$, and therefore the action-value function $Q_k^i(\mathbf{b}_k, \mathbf{u}_k)$ of agent i depends not only on its own action but also on all other players' actions. This interdependence is analogous to (3.6) but formulated recursively over time.

To better capture how an agent's action-value function depends on the controls of all other actions, we can equivalently write $Q^i(\mathbf{b}_0, \mathbf{u}) = Q^i(\mathbf{b}_0, \mathbf{u}^i, \mathbf{u}^{\neg i})$. More precisely, the interdependence of all players optimal policies is captured in the Nash equilibrium of Problem 1, defined in Problem 2. Problem 2 formulates a sufficient condition for Nash equilibria [30, 141] in belief space.

Problem 2. ***Nash Equilibrium:*** *Find the optimal control policy $\pi = [\pi^{1,\top}, \ldots, \pi^{N,\top}]^\top$ that yields a local Nash equilibrium of the POMDP Best Response Game in Problem 1, such that it satisfies*

$$Q^i(\mathbf{b}_0, \mathbf{u}^i, \pi^{\neg i}) \geq Q^i(\mathbf{b}_0, \pi^i, \pi^{\neg i}), \forall i \in \{1, 2, \ldots, N\}, \tag{3.9}$$

for all $\mathbf{u}^i$ in the neighborhood of π^i.

More intuitively, in the Nash equilibrium no player has anything to gain by changing only their own strategy. Based on the necessary condition of Problem 2, we will derive a local necessary condition for each sub-problem in the backward pass of our game-theoretic variant of belief iLQG.

3.2.1 Problem Formulation

The difficulty in solving POMDPs stems from the infinite-dimensional space of all beliefs, and that in general the value function cannot be expressed in parametric form. To overcome these challenges we describe beliefs by Gaussian distributions, approximating the belief dynamics using an EKF, and make a quadratic approximation of the value function about a nominal trajectory through the belief space. We iteratively compute a local Nash equilibrium over all agents in the proximity of the nominal trajectory by solving the necessary condition (3.9) of Problem 2 at each timestep during a belief-space variant of iLQG to perform the Bellman backward recursion in (3.8). Due to its similarity to iLQG we benefit from linear scaling $\mathcal{O}(l)$ in the planning horizon l, in contrast to point-based POMDP algorithms which scale exponentially.

We are given non-linear stochastic dynamics and observation models in state-transition notation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k), \qquad \mathbf{m}_k \sim \mathcal{N}(0, I), \tag{3.10}$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{n}_k) \qquad \mathbf{n}_k \sim \mathcal{N}(0, I), \tag{3.11}$$

where $\mathbf{m}_k$ and $\mathbf{n}_k$ are the motion and measurement noise, respectively. Without loss of generality, we draw both the motion and measurement noise from independent Gaussian distributions with zero mean and unit variance since the noise can be arbitrarily transformed inside these functions. Depending on the system, motion and sensing noise may be state and control dependent.

Note that formulating the general dynamics and measurement functions jointly of all agents includes, but is not limited to, the special case of independent functions for each agent i as in

$$f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k) = [f^1(\mathbf{x}_k^1, \mathbf{u}_k^1, \mathbf{m}_k^1)^\top, \ldots, f^N(\mathbf{x}_k^N, \mathbf{u}_k^N, \mathbf{m}_k^N)^\top]^\top, \tag{3.12}$$

$$h(\mathbf{x}_k, \mathbf{n}_k) = [h^1(\mathbf{x}_k^1, \mathbf{n}_k^1)^\top, \ldots, h^N(\mathbf{x}_k^N, \mathbf{n}_k^N)^\top]^\top. \tag{3.13}$$

We define the Gaussian belief as $\mathbf{b}_k = (\hat{\mathbf{x}}_k^\top, \Sigma_k)$, by the mean state $\hat{\mathbf{x}}_k$ and the variance Σ_k of the Normal distribution describing the stochastic state $\mathbf{x}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \Sigma_k)$.

3.3 Technical Approach

Before detailing the value iteration method for the Nash equilibrium solution based on a game-theoretic belief-space variant of iLQG in Section 3.3.3, we need to derive two important components. First, we describe the approximation of the general Bayesian filter update (3.4) by an EKF in Section 3.3.1 to formulate the Gaussian belief dynamics. This allows us to forward propagate Gaussian beliefs given an initial belief and a control trajectory which we utilitze in the game-theoretic variant of belief-space iLQG. Second, we show that the necessary condition of Problem 2, the

Nash equilibrium, is equivalent to a local necessary condition at each timestep in the Bellman recursion in Section 3.3.2. The full algorithm is detailed in Section 3.3.5.

3.3.1 Bayesian Filter and Belief Dynamics

The Bayesian filter in (3.4) defines the general belief dynamics of a current belief $\mathbf{b}_k$ and measurement $\mathbf{z}_{k+1}$. To make the belief propagation tractable we follow [201] and approximate the Bayesian filter by an EKF, suitable for non-linear Gaussian beliefs as well as non-linear dynamics and measurement models. For well-defined transition models, the EKF is the standard for nonlinear state estimation [212, 93]. The EKF makes a first-order approximation of f with respect to the stochastic variable $\mathbf{x}$, such that for a given belief $\mathbf{b}_k = (\hat{\mathbf{x}}_k, \Sigma_k)$ we have the standard EKF update equations [201, 196]

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1} &= f(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0) + K_k(\mathbf{z}_{k+1} - h(f(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0), 0)),\\
\Sigma_{k+1} &= \Gamma_{k+1} - K_k H_k \Gamma_{k+1}, \qquad (3.14)
\end{aligned}$$

with corresponding matrices defined by

$$\begin{aligned}
\Gamma_{k+1} &= A_k \Sigma_k A_k^T + M_k M_k^T, \qquad (3.15)\\
K_k &= \Gamma_{k+1} H_k^\top (H_k \Gamma_{k+1} H_k^\top + N_k N_k^\top)^{-1},\\
A_k &= \frac{\partial f}{\partial \mathbf{x}}(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0), \quad M_k = \frac{\partial f}{\partial \mathbf{m}}(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0),\\
H_k &= \frac{\partial h}{\partial \mathbf{x}}(f(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0), 0), \quad N_k = \frac{\partial h}{\partial \mathbf{n}}(f(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0), 0).
\end{aligned}$$

The noisy measurement $\mathbf{z}_k$ in the belief update makes the belief dynamics stochastic. We define $\mathbf{b}_k = [\hat{\mathbf{x}}_k^\top, \mathrm{vec}(\Sigma_k)^\top]^\top$, where $\mathrm{vec}(\Sigma_k)$ is the matrix Σ_k reshaped into vector form and formulate the stochastic belief dynamics

$$\mathbf{b}_{k+1} = g(\mathbf{b}_k, \mathbf{u}_k) + W(\mathbf{b}_k, \mathbf{u}_k)\xi_k, \quad \xi_k \sim \mathcal{N}(0, I), \qquad (3.16)$$

with

$$g_k(\mathbf{b}_k, \mathbf{u}_k) = \begin{bmatrix} f(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0) \\ \mathrm{vec}(\Gamma_{k+1} - K_k H_k \Gamma_{k+1}) \end{bmatrix}, \tag{3.17}$$

$$W_k(\mathbf{b}_k, \mathbf{u}_k) = \begin{bmatrix} \sqrt{K_k H_k \Gamma_{k+1}} \\ \mathbf{0} \end{bmatrix}. \tag{3.18}$$

Here, ξ_k is a Gaussian with dimension $n_{\mathbf{x}}$ that is applied to the stochastic part of $\mathbf{b}_k$, i.e. the stochastic state variable $\mathbf{x}_k$. In this form ξ_k represents both measurement noise $\mathbf{n}_k$ and motion noise $\mathbf{m}_k$ mapped onto the belief transition. The stochastic Gaussian belief dynamics allow us to propagate beliefs efficiently during the forward pass of the game-theoretic variant of belief-space iLQG.

3.3.2 Nash Equilibrium Necessary Condition

While formulating how to propagate uncertainty for the continous POMDP, we also need to define a tractable procedure to solve for Nash equilibria. One common method to solve for Nash equilibria is Iterated Best Response [217, 184, 183], where control policies are exchanged after each agent's separate and independent optimization iteration. In contrast, we directly integrate the necessary condition of the Nash equilibrium into the backward pass of a belief-space variant of iLQG. Specifically, we solve a quadratic game at each stage of the backward pass with a unique solution. First, we formulate the necessary condition of Problem 2 as

$$\frac{\partial Q^i(\mathbf{b}_0, \mathbf{u})}{\partial \mathbf{u}^i} = 0, \ \ \forall i \in \{1, 2, \ldots, N\}, \tag{3.19}$$

which allows us to compute local Nash equilibria by solving (3.19). Theorem 1 states an equivalent condition for $Q^i_k(\mathbf{b}_k, \mathbf{u}_k)$, the value function from time k to l, defined in the Bellman recursion (3.8).

Theorem 1. *The necessary condition of the local Nash equilibrium* (3.19) *is equiva-*

lent to

$$\frac{\partial Q_k^i(\mathbf{b}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k^i} = 0, \tag{3.20}$$

for all $i \in \{1, 2, \ldots, N\}$*, and* $k \in \{0, 1, \ldots, l-1\}$*.*

Proof. Recall that the conventional POMDP formulation (3.6) in Problem 1 is equivalent to the recursive Bellman equation (3.8). Maximizing $Q_k^i(\mathbf{b}_k, \mathbf{u}_k)$ with respect to $\mathbf{u}_k^i$ in (3.8) yields the corresponding necessary optimality condition $\frac{\partial Q_k^i(\mathbf{b}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k^i} = 0$, the same as (3.20). Therefore, the necessary optimality condition (3.19) of Problem 1 is equivalent to the recursive Bellman necessary optimality condition (3.20) in Theorem 1. □

Alternatively, we can split the action-value from time 0 into the action-value from k and the cost accumulated until k,

$$Q^i(\mathbf{b}_0, \mathbf{u}) = Q_k^i(\mathbf{b}_k, \mathbf{u}_k) + \mathop{\mathbb{E}}_{\mathbf{z}}\left[\sum_{t=0}^{k-1} c_t^i(\mathbf{b}_t, \mathbf{u}_t)\right]. \tag{3.21}$$

Taking the derivative of both sides with respect to $\mathbf{u}_k^i$ directly implies that $\frac{\partial Q_k^i(\mathbf{b}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k^i} = \frac{\partial Q^i(\mathbf{b}_0, \mathbf{u})}{\partial \mathbf{u}_k^i}$, since the cost accumulated until k, the second term on the right hand side, does not depend on $\mathbf{u}_k^i$. Intuitively, current actions cannot affect costs accumulated in the past.

Concluding, if each agent i finds an optimizing policy π_k^i to the Bellman recursion, all $\frac{\partial Q_k^i(\mathbf{b}_k, \mathbf{u}_k)}{\partial \mathbf{u}_k^i} = 0$ necessary conditions are fulfilled at time k. Note that each agents' policy $\pi^i(\mathbf{u}^{\neg i})$ depends on the other agents' inputs $\mathbf{u}^{\neg i}$, where $\neg i$ indicates all other agents. Therefore, solving the Bellman recursion simultaneously for all agents defines a static game [30], but more importantly a game at each stage k of the backward-pass.

In the next section, we describe our solution for integrating the Nash equilibrium necessary condition at every time k into the backward pass of a belief-space variant of iLQG.

3.3.3 Iterative Dynamic Programming

In this section we describe our belief-space variant of iLQG for computing local Nash equilibria by solving the Bellman recursion defined in (3.8). We denote the nominal belief as $\bar{\mathbf{b}} = \mathbf{b} - \delta\mathbf{b}$, the nominal controls $\bar{\mathbf{u}} = \mathbf{u} - \delta\mathbf{u}$, and $\bar{\mathbf{s}} = \mathbf{s} - \delta\mathbf{s}$, with $\bar{\mathbf{s}} = [\bar{\mathbf{b}}^\top, \bar{\mathbf{u}}^\top]^\top$ and local perturbations $\delta\mathbf{u}$, $\delta\mathbf{b}$, $\delta\mathbf{s}$. At each iteration, the algorithm performs a backward pass and a forward pass on the current estimate of the belief $\bar{\mathbf{b}} = [\bar{\mathbf{b}}_0, \bar{\mathbf{b}}_1, \ldots, \bar{\mathbf{b}}_l]$ and control trajectory $\bar{\mathbf{u}} = [\bar{\mathbf{u}}_0, \bar{\mathbf{u}}_1, \ldots, \bar{\mathbf{u}}_{l-1}]$, i.e. the nominal trajectories. In the backward pass, the algorithm approximates the value functions for each agent as a quadratic function

$$V_k^i(\bar{\mathbf{b}}_k + \delta\mathbf{b}_k) \approx V_k^i + V_{\mathbf{b},k}^{i,\top}\delta\mathbf{b}_k + \frac{1}{2}\delta\mathbf{b}_{k+1}^\top V_{\mathbf{bb},k}^i \delta\mathbf{b}_k, \tag{3.22}$$

along the nominal trajectory, and computes a linear feedback policy π^1 for the robot and predicted linear feedback policies $\pi^{\neg 1}$ for all other agents. The value function is propagated backwards in time. In the forward pass we produce a new nominal trajectory based on the value function computed in the backward pass and apply the associated feedback policy. This iterative process continues towards a locally optimal solution to the Nash equilibrium in belief space. The key idea is to maintain a quadratic approximation of $Q_k^i(\mathbf{b}_k, \mathbf{u}_k)$ and the value functions $V_k^i(\mathbf{b}_k)$.

We first derive the quadratic form of $Q_k^i(\mathbf{b}_k, \mathbf{u}_k)$ in Theorem 2 by a Taylor expansion of the dynamics and costs, then find the minimizing control policy $\pi_k = [\pi_k^{1,\top}, \ldots, \pi_k^{N,\top}]^\top$ by solving the static game and computing the Nash equilibrium over all agents. From this result we compute the value functions $V_k^i(\mathbf{b}_k) = Q_k^i(\mathbf{b}_k, \pi_k)$ and derive an update law for V backwards in time.

Theorem 2. *By linear expansion of the belief dynamics and quadratic expansion of the cost and value function, $Q_k^i(\mathbf{s}_k)$ is a quadratic of the form*

$$Q_k^i(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k) \approx Q_k^i + Q_{\mathbf{s},k}^{i,\top}\delta\mathbf{s}_k + \frac{1}{2}\delta\mathbf{s}_k^\top Q_{\mathbf{ss},k}^i \delta\mathbf{s}_k, \tag{3.23}$$

where

$$Q^i_k = c^i_k + V^i_{k+1} + \frac{1}{2}\sum_{j=1}^{n_x} W_k^{(j),\top} V^i_{\mathbf{bb},k+1} W_k^{(j)}, \tag{3.24}$$

$$Q^i_{\mathbf{s},k} = c^i_{\mathbf{s},k} + g_{\mathbf{s},k}^\top V^i_{\mathbf{b},k+1} + \sum_{j=1}^{n_x} W_{\mathbf{s},k}^{(j),\top} V^i_{\mathbf{bb},k+1} W_k^{(j)}, \tag{3.25}$$

$$Q^i_{\mathbf{ss},k} = c^i_{\mathbf{ss},k} + g_{\mathbf{s},k}^\top V^i_{\mathbf{bb},k+1} g_{\mathbf{s},k} + \sum_{j=1}^{n_x} W_{\mathbf{s},k}^{(j),\top} V^i_{\mathbf{bb},k+1} W_{\mathbf{s},k}^{(j)}. \tag{3.26}$$

Similar derivations in iLQG [198], and belief-space iLQG [201] showed an agent's action-value function Q to be quadratic with respect to the agent's controls and state or belief. In contrast, we show that agent i's action-value function Q^i is also a quadratic with respect to the *joint* state and controls which is critical to formulate the static quadratic game in the backward pass.

Proof. We start by expanding the terms of the action-value function of the Bellman recursion, cf. (3.8),

$$Q^i_k(\mathbf{b}_k, \mathbf{u}_k) = c^i_k(\mathbf{b}_k, \mathbf{u}_k) + \mathop{\mathbb{E}}_{\xi_k}\left[V^i_{k+1}(g_k(\mathbf{b}_k, \mathbf{u}_k) + W_k(\mathbf{b}_k, \mathbf{u}_k)\xi_k)\right], \tag{3.27}$$

to second order around the nominal control and belief $\bar{\mathbf{s}}_k = [\bar{\mathbf{b}}_k^\top, \bar{\mathbf{u}}_k^\top]^\top$. The term $c^i_k(\mathbf{b}_k, \mathbf{u}_k)$ becomes

$$c^i_k(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k) \approx c^i_k + c^{i,\top}_{\mathbf{s},k}\delta\mathbf{s}_k + \frac{1}{2}\delta\mathbf{s}_k^\top c^i_{\mathbf{ss},k}\delta\mathbf{s}_k, \tag{3.28}$$

with $c^i_k = c^i_k(\bar{\mathbf{s}})$, where $c^i_{\mathbf{s},k}$ and $c^i_{\mathbf{ss},k}$ are the Jacobian and Hessian of c^i_k evaluated at $\bar{\mathbf{s}}_k$. To expand the second term on the right hands side of (3.27) we first expand the stochastic joint belief dynamics to

$$g_k(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k) \approx g_k + g_{\mathbf{s},k}\delta\mathbf{s}_k, \tag{3.29}$$

$$W_k^{(j)}(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k) \approx W_k^{(j)} + W_{\mathbf{s},k}^{(j)}\delta\mathbf{s}_k, \tag{3.30}$$

with terms $g_k = g_k(\bar{\mathbf{s}}_k)$, $W_k^{(j)} = W_k^{(j)}(\bar{\mathbf{s}}_k)$, and $g_{\mathbf{s},k}$, $W_{\mathbf{s},k}^{(j)}$ the respective Jacobians evaluated at $\bar{\mathbf{s}}_k$. $W_k^{(j)}$ denotes the j-th column of matrix W_k.

We now formulate the second term of (3.27). We define the value function as a quadratic around $\bar{\mathbf{b}}_{k+1}$

$$
\begin{aligned}
& V^i_{k+1}(\bar{\mathbf{b}}_{k+1} + \delta \mathbf{b}_{k+1}) && (3.31)\\
&\quad \approx V^i_{k+1} + V^{i,\top}_{\mathbf{b},k+1}\delta\mathbf{b}_{k+1} + \frac{1}{2}\delta\mathbf{b}^\top_{k+1} V^i_{\mathbf{bb},k+1}\delta\mathbf{b}_{k+1} \\
&\quad = V^i_{k+1} + V^{i,\top}_{\mathbf{b},k+1}(\mathbf{b}_{k+1} - \bar{\mathbf{b}}_{k+1}) && (3.32)\\
&\qquad + \frac{1}{2}(\mathbf{b}_{k+1} - \bar{\mathbf{b}}_{k+1})^\top V^i_{\mathbf{bb},k+1}(\mathbf{b}_{k+1} - \bar{\mathbf{b}}_{k+1}),
\end{aligned}
$$

with $\delta\mathbf{b}_{k+1} = \mathbf{b}_{k+1} - \bar{\mathbf{b}}_{k+1}$ for convenience. Inserting the expanded dynamics (3.29), (3.30) into the second term of (3.27), i.e. (3.32), and evaluating the expectation over ξ_k yields

$$
\begin{aligned}
& \mathop{\mathbb{E}}_{\xi_k}\left[V^i_{k+1}(g_k(\mathbf{s}_k) + W_k(\mathbf{s}_k)\xi_k)\right] && (3.33)\\
& \approx \mathop{\mathbb{E}}_{\xi_k}\Bigg[V^i_{k+1} + V^{i,\top}_{\mathbf{b},k+1}\big(g_k(\mathbf{s}_k) + W_k(\mathbf{s}_k)\xi_k - \bar{\mathbf{b}}_{k+1}\big) && (3.34)\\
&\quad + \frac{1}{2}\big(g_k(\mathbf{s}_k) + W_k(\mathbf{s}_k)\xi_k - \bar{\mathbf{b}}_{k+1}\big)^\top V^i_{\mathbf{bb},k+1}\big(g_k(\mathbf{s}_k) + W_k(\mathbf{s}_k)\xi_k - \bar{\mathbf{b}}_{k+1}\big)\Bigg] \\
& = V^i_{k+1} + V^{i,\top}_{\mathbf{b},k+1}\big(g_k(\mathbf{s}_k) - \bar{\mathbf{b}}_{k+1}\big) && (3.35)\\
&\quad + \frac{1}{2}\big(g_k(\mathbf{s}_k) - \bar{\mathbf{b}}_{k+1}\big)^\top V^i_{\mathbf{bb},k+1}\big(g_k(\mathbf{s}_k) - \bar{\mathbf{b}}_{k+1}\big) \\
&\quad + \frac{1}{2}\mathrm{tr}\left(W_k(\mathbf{s}_k)^\top V^i_{\mathbf{bb},k+1} W_k(\mathbf{s}_k)\right) \\
& = V^i_{k+1} + V^{i,\top}_{\mathbf{b},k+1} g_{\mathbf{s},k}\delta\mathbf{s}_k + \frac{1}{2}\delta\mathbf{s}_k^\top g_{\mathbf{s},k}^\top V^i_{\mathbf{bb},k+1} g_{\mathbf{s},k}\delta\mathbf{s}_k && (3.36)\\
&\quad + \frac{1}{2}\sum_{j=1}^{n_x}(W_k^{(j)} + W_{\mathbf{s},k}^{(j)}\delta\mathbf{s}_k)^\top V^i_{\mathbf{bb},k+1}(W_k^{(j)} + W_{\mathbf{s},k}^{(j)}\delta\mathbf{s}_k)^\top.
\end{aligned}
$$

Here we use the value function expansion (3.32) in (3.34), and the fact that $\bar{\mathbf{b}}_{k+1} = g_k(\bar{\mathbf{s}}_k)$ in (3.35) in the form of

$$
g_k(\mathbf{s}_k) - \bar{\mathbf{b}}_{k+1} = g_k(\mathbf{s}_k) - g_k(\bar{\mathbf{s}}_k) \approx g_{\mathbf{s},k}\delta\mathbf{s}_k. \qquad (3.37)
$$

Collecting and grouping all first and second order terms of (3.36) and (3.28) we have that the resulting $Q^i_k(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k)$ is a quadratic with coefficients given by (3.24 - 3.26). □

For notational convenience we will drop the time index k for the Q matrices. We can also recover other partial derivatives of Q^i from (3.24 - 3.26):

$$Q^i_{\mathbf{s}} = \begin{bmatrix} Q^i_{\mathbf{b}} \\ Q^i_{\mathbf{u}^1} \\ \vdots \\ Q^i_{\mathbf{u}^N} \end{bmatrix}, \; Q^i_{\mathbf{ss}} = \begin{bmatrix} Q^i_{\mathbf{bb}} & Q^i_{\mathbf{bu}^1} & \cdots & Q^i_{\mathbf{bu}^N} \\ Q^i_{\mathbf{u}^1\mathbf{b}} & Q^i_{\mathbf{u}^1\mathbf{u}^1} & \cdots & Q^i_{\mathbf{u}^1\mathbf{u}^N} \\ \vdots & \vdots & \ddots & \vdots \\ Q^i_{\mathbf{u}^N\mathbf{b}} & Q^i_{\mathbf{u}^N\mathbf{u}^1} & \cdots & Q^i_{\mathbf{u}^N\mathbf{u}^N} \end{bmatrix}. \tag{3.38}$$

With $Q^i_k(\bar{\mathbf{s}}_k + \delta\mathbf{s}_k)$ in quadratic form from Theorem 2, at stage k each agent i solves the quadratic problem

$$\delta\mathbf{u}^{i,*}_k = \arg\min_{\delta\mathbf{u}^i_k} \; Q^{i,\top}_{\mathbf{s},k}\delta\mathbf{s}_k + \frac{1}{2}\delta\mathbf{s}^\top_k Q^i_{\mathbf{ss},k}\delta\mathbf{s}_k, \tag{3.39}$$

yielding a quadratic game in the variables $\mathbf{u}_k$. Note that each agent's optimal $\delta\mathbf{u}^{i,*}_k$ depends on all other agents' $\delta\mathbf{u}^{\neg i}_k$ as they are contained in $\delta\mathbf{s}_k$. In comparison to other related methods such as Iterative Linear-Quadratic regulator control (iLQR) [113], iLQG [198], or Differential Dynamic Programming (DDP) [92] that solve a single quadratic optimization in the backward pass, we have to solve N interdependent quadratic optimizations. Nonetheless, we obtain a unique and simple to compute solution to the quadratic game [30] by stacking the N optimality conditions of each interdependent optimization. Solving the resulting system of equations amounts to solving all interdependent quadratic optimizations at once. Theorem 3 presents this result.

Theorem 3. *The solution to the quadratic game* (3.39) *is*

$$\delta\mathbf{u}^*_k = -\hat{Q}^{-1}_{\mathbf{uu}}\left(\hat{Q}_{\mathbf{u}} + \hat{Q}_{\mathbf{ub}}\delta\mathbf{b}_k\right), \tag{3.40}$$

where $\hat{Q}_{\mathbf{uu}}$, $\hat{Q}_{\mathbf{ub}}$, $\hat{Q}_{\mathbf{u}}$, *are populated from* (3.38), *and defined*

$$\hat{Q}_{\mathbf{uu}} = \begin{bmatrix} Q^1_{\mathbf{u}^1\mathbf{u}} \\ Q^2_{\mathbf{u}^2\mathbf{u}} \\ \vdots \\ Q^N_{\mathbf{u}^N\mathbf{u}} \end{bmatrix}, \ \hat{Q}_{\mathbf{ub}} = \begin{bmatrix} Q^1_{\mathbf{u}^1\mathbf{b}} \\ Q^2_{\mathbf{u}^2\mathbf{b}} \\ \vdots \\ Q^N_{\mathbf{u}^N\mathbf{b}} \end{bmatrix}, \ \hat{Q}_{\mathbf{u}} = \begin{bmatrix} Q^1_{\mathbf{u}^1} \\ Q^2_{\mathbf{u}^2} \\ \vdots \\ Q^N_{\mathbf{u}^N} \end{bmatrix}. \tag{3.41}$$

Proof. By taking the derivative of the objective of (3.39) and equating it to zero, the stationarity condition of (3.39) yields

$$\begin{bmatrix} Q^i_{\mathbf{u}^i\mathbf{u}^i} & Q^i_{\mathbf{u}^i\mathbf{u}^{\neg i}} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}^i_k \\ \delta\mathbf{u}^{\neg i}_k \end{bmatrix} + Q^i_{\mathbf{u}^i\mathbf{b}}\delta\mathbf{b}_k + Q^i_{\mathbf{u}^i} = 0. \tag{3.42}$$

Stacking the stationarity conditions of all N agents into a single system of equations we find the joint stationarity condition for all interdependent quadratic optimizations

$$\hat{Q}_{\mathbf{uu}}\delta\mathbf{u}_k + \hat{Q}_{\mathbf{ub}}\delta\mathbf{b}_k + \hat{Q}_{\mathbf{u}} = 0, \tag{3.43}$$

where (3.40) is the solution to this system of equations. □

The local necessary condition of Problem 2, derived in Section 3.3.2 holds as shown below.

Corollary 1. *The solution* (3.40) *fulfills the necessary condition of the local Nash equilibrium* (3.19) *at time* k.

Proof. From (3.42), we see that $\frac{\partial Q^i_k(\mathbf{b}_k,\mathbf{u}_k)}{\partial \mathbf{u}^i_k} = 0$. □

We can immediately derive the linear feedback policy for all agents at planning time k of the form

$$\pi_k = \bar{\mathbf{u}}_k + j_k + K_k\delta\mathbf{b}_k, \tag{3.44}$$

with $j_k = -\hat{Q}^{-1}_{\mathbf{uu}}\hat{Q}_{\mathbf{u}}$ the feed forward term and $K_k = -\hat{Q}^{-1}_{\mathbf{uu}}\hat{Q}_{\mathbf{ub}}$ the feedback term. Note that π_k contains the optimal policy of the robot π^1_k and also the predicted policies for all other (N-1) agents $\pi^{\neg 1}_k$. The interdependence has been resolved by solving

(3.43). The predicted linear policies $\pi_k^{\neg 1}$ depend on the change in *joint* belief $\delta\mathbf{b}_k$. The predicted actions will adapt if the robot, the other agents, or the environment behave differently as expected, causing the estimated belief $\mathbf{b}_k$ at future times k to diverge from the predicted nominal belief $\bar{\mathbf{b}}_k$. Similarly, the robot's linear policy π_k^1 will allow it to adapt if other agents deviate from predicted behavior. In contrast, this flexibility would be impossible with a static optimal control trajectory instead of a policy.

We now formulate the backward equations to propagate the value functions V^i backwards, hence defining the backward pass.

Corollary 2. *The discrete backward differential equations of the value functions V^i are*

$$V_k^i = Q^i + Q_{\mathbf{u}}^{i,\top} j_k + \frac{1}{2} j_k^\top Q_{\mathbf{uu}}^i j_k, \tag{3.45}$$

$$V_{\mathbf{b},k}^i = Q_{\mathbf{b}}^i + K_k^\top Q_{\mathbf{uu}}^i j_k + K_k^\top Q_{\mathbf{u}}^i + Q_{\mathbf{ub}}^{i,\top} j_k, \tag{3.46}$$

$$V_{\mathbf{bb},k}^i = Q_{\mathbf{bb}}^i + K_k^\top Q_{\mathbf{uu}}^i K_k + K_k^\top Q_{\mathbf{ub}}^i + Q_{\mathbf{ub}}^{i,\top} K_k, \tag{3.47}$$

with terminal constraints

$$V_l^i = c_l^i(\bar{\mathbf{b}}_l), \; V_{\mathbf{b},l}^i = \left.\frac{\partial c_l^i(\mathbf{b})}{\partial \mathbf{b}}\right|_{\mathbf{b}=\bar{\mathbf{b}}_l}, \; V_{\mathbf{bb},l}^i = \left.\frac{\partial^2 c_l^i(\mathbf{b})}{\partial \mathbf{b}^2}\right|_{\mathbf{b}=\bar{\mathbf{b}}_l}. \tag{3.48}$$

Proof. Substituting the solution (3.40) and (3.44) back into the quadratic (3.23) yields the value function $V_k^i(\bar{\mathbf{b}}_k + \delta\mathbf{b}_k)$.

$$\begin{aligned}
V_k^i(\bar{\mathbf{b}}_k + \delta\mathbf{b}_k) &= Q_k^i(\bar{\mathbf{b}}_k + \delta\mathbf{b}_k, \pi_k) \\
&= Q^i + Q_{\mathbf{u}}^{i,\top}(j_k + K_k\delta\mathbf{b}_k) + Q_{\mathbf{b}}^{i,\top}\delta\mathbf{b}_k \\
&+ \frac{1}{2}(j_k + K_k\delta\mathbf{b}_k)^\top Q_{\mathbf{uu}}^i (j_k + K_k\delta\mathbf{b}_k) + \frac{1}{2}\delta\mathbf{b}_k^\top Q_{\mathbf{bb}}^i \delta\mathbf{b}_k \\
&+ \frac{1}{2}(j_k + K_k\delta\mathbf{b}_k)^\top Q_{\mathbf{ub}}^i \delta\mathbf{b}_k + \frac{1}{2}\delta\mathbf{b}_k^\top Q_{\mathbf{bu}}^i (j_k + K_k\delta\mathbf{b}_k).
\end{aligned} \tag{3.49}$$

Collecting first and second order terms in $\delta\mathbf{b}_k$ gives the Equations (3.45-3.47) in the form of (3.32). The terminal constraints (3.48) result from a tailor expansion of the final cost c_l^i around the final nominal belief $\bar{\mathbf{b}}_l$. $\square$

Based on results of Theorem 3 and Corollary 2 we can propagate the quadratic value functions backwards in time starting from the terminal constraints at time l.

3.3.4 Regularization

With any Newton-like method, care must be taken when the Hessian $\hat{Q}_{\mathbf{uu}}$ is not positive-definite or when the minimum is not close and the quadratic model inaccurate. To ensure that the algorithm converges regardless of initial conditions, we implement a Levenberg-Marquardt style regularization [108].

Control Regularization

The control regularization is achieved by adding a diagonal term of magnitude $\mu_{\mathbf{u}}$ to the diagonal of $\hat{Q}_{\mathbf{uu}}$, yielding

$$\tilde{Q}^i_{\mathbf{uu}} = \hat{Q}^i_{\mathbf{uu}} + \mu_{\mathbf{u}} I. \tag{3.50}$$

This simple Levenberg-Marquardt style modification results in adding a quadratic cost around the current control sequence, which forces the new optimal control inputs computed by the backward pass to stay closer to the previous iteration.

Belief Regularization

The drawback of the control based regularization scheme is that even small control perturbations can cause large deviations in the state trajectory potentially inhibiting convergence. To ensure that the updated belief trajectory does not deviate too far from the previous iteration, we introduce a scheme that penalizes deviations from beliefs rather than controls with parameter $\mu_{\mathbf{b}}$:

$$\tilde{Q}^i_{\mathbf{ss},k} = c^i_{\mathbf{ss},k} + g^T_{\mathbf{s},k}\left(V^i_{\mathbf{bb},k+1} + \mu_{\mathbf{b}} I\right) g_{\mathbf{s},k} + \sum_{i=1}^{n} W^{(j),T}_{\mathbf{s},k}\left(V^i_{\mathbf{bb},k+1} + \mu_{\mathbf{b}} I\right) W^{(j)}_{\mathbf{s},k}. \tag{3.51}$$

The outcome of the belief-based regularization is placing a quadratic belief-cost around the previous belief trajectory, similar to [195] where a state based regularization was employed. In contrast to the standard control-based regularization, the feedback gains K_k do not go to zero as $\mu_{\mathbf{b}} \to \infty$, but rather force the new trajectory closer to the old one. In practice, we find this improves the robustness of convergence.

3.3.5 Algorithm for Dynamic-Game Belief-Space Planning

We summarize our findings of solving for Nash equilibria of dynamic games in belief space in Algorithm 6. Theorem 2 lays the foundation for the quadratic game solved in the backward pass of Algorithm 6. The solution to the quadratic game presented in Theorem 3 yields a linear feedback policy π_k for all agents. We propagate the value function in the backward pass according to Corollary 2 starting with the terminal conditions from the terminal cost.

Algorithm 6 starts from the current belief estimate $\mathbf{b}_0$, in our experiments provided from an EKF, and an initial control trajectory guess. We found initializing controls to all zeros to work well in practice. We update the nominal control and belief trajectories in the forward pass based on rolling out the belief dynamics model and applying the updated feedback policy π_k. If all agents' action-value functions improved, we accept the updated nominal belief and control trajectories and reduce regularization. Otherwise, we reject the trajectories and increase regularization. The iteration of backward and forward pass continues until each agents' action value function Q^i has converged and changes less than a specified threshold ϵ.

Algorithm 6 Nash Equilibrium of Dynamic Game in Belief Space

```
    Input: Initial belief $\mathbf{b}_0$, control $\bar{\mathbf{u}}$, models $c_k^i$, $c_l^i$, $f$, $h$
    Output: Predicted trajectories $\bar{\mathbf{b}}$, $\bar{\mathbf{u}}$, feedback law $\pi$
 1: $\bar{\mathbf{b}} \leftarrow$ Propagate $\mathbf{b}_0$ with $g$ and $\bar{\mathbf{u}}$
 2: while $|Q^i(\bar{\mathbf{b}}_{\text{new}}, \bar{\mathbf{u}}_{\text{new}}) - Q^i(\bar{\mathbf{b}}, \bar{\mathbf{u}})| > \epsilon$ do
 3:     Backward pass:
 4:     $V^i_{\mathbf{b},l}$, $V^i_{\mathbf{bb},l} \leftarrow$ From terminal boundary conditions (3.48)
 5:     for $k \leftarrow l-1$ to 0 do
 6:         $\pi^i_k$, $j^i_k$, $K^i_k \leftarrow$ Solve quadratic game (3.44)
 7:         $V^i_{\mathbf{b},k}$,$V^i_{\mathbf{bb},k} \leftarrow$ Propagate value function (3.46, 3.47)
 8:     end for
 9:     Forward pass:
10:     $\bar{\mathbf{b}}_{\text{new}}$, $\bar{\mathbf{u}}_{\text{new}} \leftarrow$ Propagate $\mathbf{b}_0$ with $g$ and $\pi$
11:     if $Q^i(\bar{\mathbf{b}}_{\text{new}}, \bar{\mathbf{u}}_{\text{new}}) \le Q^i(\bar{\mathbf{b}}, \bar{\mathbf{u}})$ then
12:         $\bar{\mathbf{b}}, \bar{\mathbf{u}} \leftarrow \bar{\mathbf{b}}_{\text{new}}, \bar{\mathbf{u}}_{\text{new}}$,
13:         lower regularization (3.50, 3.51)
14:     else  increase regularization
15:     end if
16: end while
```

The algorithm yields a linear feedback policy π^1 and a predicted belief trajectory $\mathbf{b}^1$ of the robot, as well as predicted feedback policies $\pi^{\neg 1}$ and predicted belief trajectories $\mathbf{b}^{\neg 1}$ for all other agents over the full time horizon.

3.3.6 Runtime Analysis

The dominant runtime complexity in a single backward step is $\mathcal{O}(N^7 n_{\mathbf{x}}^{i,6})$. A full iteration of Algorithm 6 solves l quadratic games leading the final runtime complexity to $\mathcal{O}(lN^7 n_{\mathbf{x}}^{i,6})$. Scaling linearly in the planning horizon l enables real-time deployment whereas other POMDP algorithms scale exponentially, even without taking any game dynamics into account. The following provides a brief summary of our runtime analysis.

We analyze the runtime by first recalling the dimension of the joint state is $\mathcal{O}(n_{\mathbf{x}})$, and assume for the sake of analysis that the agents' state dimensions are equal such that $\mathcal{O}(n_{\mathbf{x}}) = \mathcal{O}(Nn^i_{\mathbf{x}})$. To simplify the analysis, we also assume the joint input $(n_{\mathbf{u}})$ and the joint measurement dimensions $(n_{\mathbf{z}})$ to be $\mathcal{O}(n_{\mathbf{x}})$. The covariance matrix of the joint state contains $n^2_{\mathbf{x}}/2$ unique elements. Since the joint belief $\mathbf{b}$ contains the covariance of the state additionally to the state, it entails $\mathcal{O}(n^2_{\mathbf{x}})$ elements.

Now consider the matrix multiplicative terms in the iterative dynamic programming procedure. A computational bottleneck occurs when updating the action-value function $Q^i_{\mathbf{ss}}$ in (3.26). Evaluating the product $g_{\mathbf{s}}^\top V^i_{\mathbf{bb}} g_{\mathbf{s}}$ requires the multiplication of matrices with dimensions $\mathcal{O}(n^2_{\mathbf{x}}) \times \mathcal{O}(n^2_{\mathbf{x}})$, resulting in $\mathcal{O}(n^6_{\mathbf{x}}) = \mathcal{O}(N^6 n^{i,6}_{\mathbf{x}})$ complexity. This operation has to be completed for each of the N agents such that the complexity increases to $\mathcal{O}(N^7 n^{i,6}_{\mathbf{x}})$. The term $W^{(j),\top}_{\mathbf{s}} V^i_{\mathbf{bb}} W^{(j)}_{\mathbf{s}}$ in (3.26) must be computed $n_{\mathbf{x}}$ times, but can be evaluated in $\mathcal{O}(n^5_{\mathbf{x}})$, since W only contains $n_{\mathbf{x}}$ non-zero elements. See (3.18) for the definition of W. We solve the quadratic game at each stage by finding the inverse of the $n_{\mathbf{u}} \times n_{\mathbf{u}}$ matrix $\hat{Q}_{\mathbf{uu}}$ in (3.41), which has complexity $\mathcal{O}(n^3_{\mathbf{x}})$. Therefore, $\mathcal{O}(N^7 n^{i,6}_{\mathbf{x}})$ remains the dominant runtime complexity.

Next, we investigate the complexity of evaluating derivatives, Hessians, and Jacobians. The cost Hessian $c^i_{\mathbf{ss}}$ only contains $\mathcal{O}(n^4_{\mathbf{x}})$ elements. Automatic differentiation through source code transformation yields $\mathcal{O}(1)$ complexity for each element, such that the cost Hessian term has no significant impact on the overall runtime complexity. The EKF belief dynamics can be evaluated in $\mathcal{O}(n^3_{\mathbf{x}})$, such that linearizing the belief dynamics to obtain $W_{\mathbf{s}}$ and $g_{\mathbf{s}}$, both with $\mathcal{O}(n^2_{\mathbf{x}})$ entries, results in $\mathcal{O}(n^5_{\mathbf{x}})$.

Thus, we find the dominant runtime complexity in a single backward step is $\mathcal{O}(N^7 n^{i,6}_{\mathbf{x}})$, and a final runtime complexity of $\mathcal{O}(lN^7 n^{i,6}_{\mathbf{x}})$ in a full iteration of Algorithm 6.

3.4 Case Studies

We demonstrate the performance and flexibility of our algorithm in three case studies that combine the information-seeking behavior with our game-theoretic formulation. These case studies examine how the agents interact in the game, gain information, and use the information gain to improve their control policies. We choose these illustrative examples due to their variations in agent interactions and demonstration of broader capabilities. Each of the case studies employs a different dynamics and observation model as well as distinct objectives for the agents. We find the Nash equilibrium to each of these games through Algorithm 6.

3.4.1 Active Surveillance

In this case study, Agents 1 and 2 are in an environment with variable lighting conditions. Agent 1 is tasked with observing Agent 2, but the quality of the observations depends on the available lighting at the location in the environment. Agent 2 has no goal, but is assigned the objective of maintaining a constant velocity while avoiding Agent 1. In the provided examples, the agents do not directly exchange information. Instead, they perceive themselves and the other agent only through observations. At the time of initialization, neither agent has perfect information of the other but only a noisy state estimate defining the initial belief $\mathbf{b}_0$. Using our approach, we show that Agent 1 can successfully herd Agent 2 into the lighted region to achieve its surveillance objective, which would not be possible without incorporating the belief space planning into the dynamic game. Figures 3-2 and 3-3 show the planned trajectories in two environments. Our case study goes beyond the commonly studied multi-robot herding problem [116, 145, 144, 188] which has the goal of herding agents into a specified location. In contrast, our goal is to reduce uncertainty in the final state of another agent, which happens to coincide with pushing the other agent into the light. Algorithms commonly applied to the herding problem are not applicable here as they do not reason about the uncertainty of other agents.

The state of both car-like robots $\mathbf{x}^{(i)} = [x^{(i)}, y^{(i)}, \theta^{(i)}, v^{(i)}]$ consists of their position

(x, y), orientation θ, and speed v. The control inputs $\mathbf{u}^{(i)} = [u^{(i)}_{\text{acc},k}, u^{(i)}_{\text{steer},k}]$ are acceleration $u_{\text{acc},k}$ and steering wheel angle $u_{\text{steer},k}$. The deterministic continuous dynamics of both agents are given by

$$\dot{\mathbf{x}}^{(i)}_k = \left[v^{(i)}_k \cos\theta^{(i)}_k,\ v^{(i)}_k \sin\theta^{(i)}_k, u^{(i)}_{\text{acc},k}, \frac{v^{(i)}_k}{L\tan\left(u^{(i)}_{\text{steer},k}\right)}\right]^\top,$$

where L is the length of the robots. The discrete time dynamics are defined by

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k) = \mathbf{x}_k + \dot{\mathbf{x}}_k\tau + M(\mathbf{u}_k)\cdot\mathbf{m}_k,$$

for timestep τ and $M(\mathbf{u}_k)$ scales the motion noise $\mathbf{m}_k$ proportional to the control input $\mathbf{u}_k$, such that uncertainty increases if excessive controls are executed. We encode the agent's objective and goals in this game by defining the current and terminal costs for Agent 1 and Agent 2 as

$$\begin{aligned}
c^{(1)}_k(\mathbf{b}_k, \mathbf{u}_k) &= \mathbf{u}^{(1),\top}_k R\mathbf{u}^{(1)}_k, \\
c^{(1)}_l(\mathbf{b}_l) &= \det(\Sigma^{(2)}_{x,y,l}), \\
c^{(2)}_k(\mathbf{b}_k, \mathbf{u}_k) &= \mathbf{u}^{(2),\top}_k R\mathbf{u}^{(2)}_k + a_1(v^{(2)}_k - v^{(2)}_{k,\text{des}})^2 + a_2 c_{\text{coll}}(\mathbf{x}_k), \\
c^{(2)}_l(\mathbf{b}_l) &= a_1(v^{(2)}_l - v^{(2)}_{l,\text{des}})^2 + a_2 c_{\text{coll}}(\mathbf{x}_l).
\end{aligned}$$

Agent 1's overall objective is to lower the uncertainty about the position of Agent 2 at the end of the planning horizon, encoded by $c^{(1)}_l(\mathbf{b}_l)$. The term $\det(\Sigma^{(2)}_{x,y})$ is equivalent to the area of the 1σ-threshold ellipse of Agent 2 and representative of the location uncertainty of Agent 2 at the end of the planning horizon. Note that both agents penalize control effort by $\mathbf{u}^{(i),\top}_k R\mathbf{u}^{(i)}_k$, and Agent 2 has additional objectives for maintaining a desired velocity v_{des} and avoiding collisions via an exponential barrier $c_{\text{coll}}(\mathbf{x}_k) = \exp(-d(\mathbf{x}_k))$. Here $d(\mathbf{x}_k)$ is the expected euclidean distance until collision between the two agents, taking their outline into account.

We restrict the robots' sensing abilities to only include noisy position measurements. The observation model varies across the environment based on the available

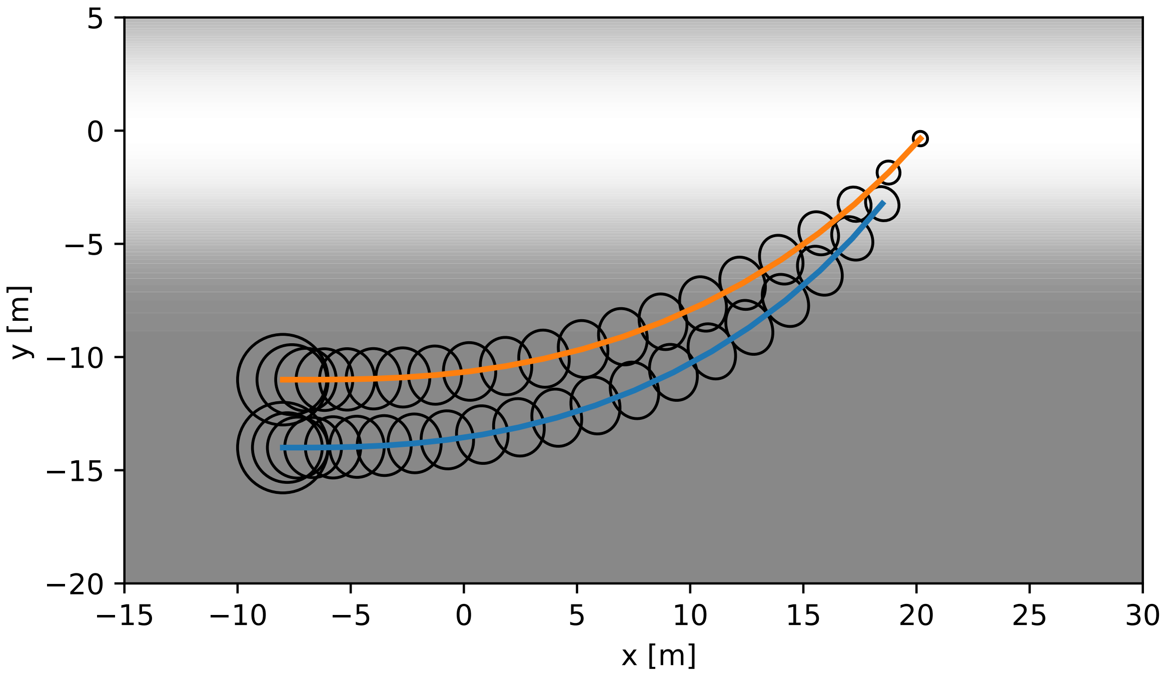


Figure 3-2: **Active surveillance along linear light source:** Agent 1 (blue) is pushing Agent 2 (orange) into the light to reduce the uncertainty over Agent 2 at the end of the planning horizon. Uncertainties are visualized by covariance ellipses. Both agents are initialized with positive velocity in the x-direction.

light at a particular location,

$$\mathbf{z}_k^{(i)} = h(\mathbf{x}_k^{(i)}, \mathbf{n}_k^{(i)}) = [x_k^{(i)}, y_k^{(i)}]^T + N(\mathbf{x}_k^{(i)}) \cdot \mathbf{n}_k^{(i)},$$

where the matrix $N(\mathbf{x}_k^{(i)})$ scales the measurement noise based on the current position (x, y) in the map. We show the nominal trajectories and the associated beliefs of the solution computed using Algorithm 6 in Figures 3-2 and 3-3. In both cases Agent 1 (blue) is able to force Agent 2 into the light to successfully reduce uncertainty. The emergent behavior would not have been possible without belief-space planning, reasoning about another agent's uncertainty, and without the dynamic game, estimating how the own actions influence another agent's actions. We show the resulting behavior without belief-space planning and without any reasoning about Agent 2's uncertainty in the inset of Figure 3-3.

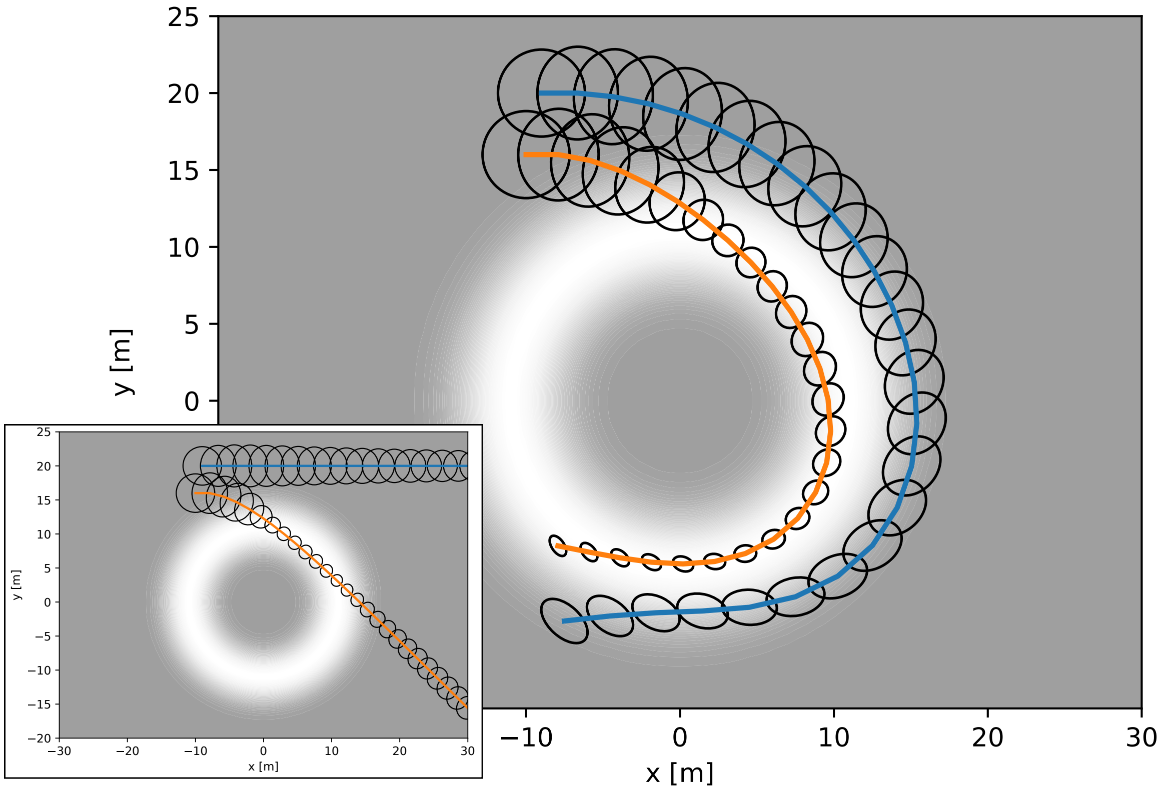


Figure 3-3: **Active surveillance along circular light source:** By nudging Agent 2 onto the circular light source, Agent 1 is able to reduce the uncertainty over Agent 2's state at the end of the planning horizon. The lower left inset shows the same scenario without any information gain. As a result, Agent 1 has no incentive to manipulate Agent 2's behavior since there is no way to influence its uncertainty. Both agents start with positive velocity in the x-direction.

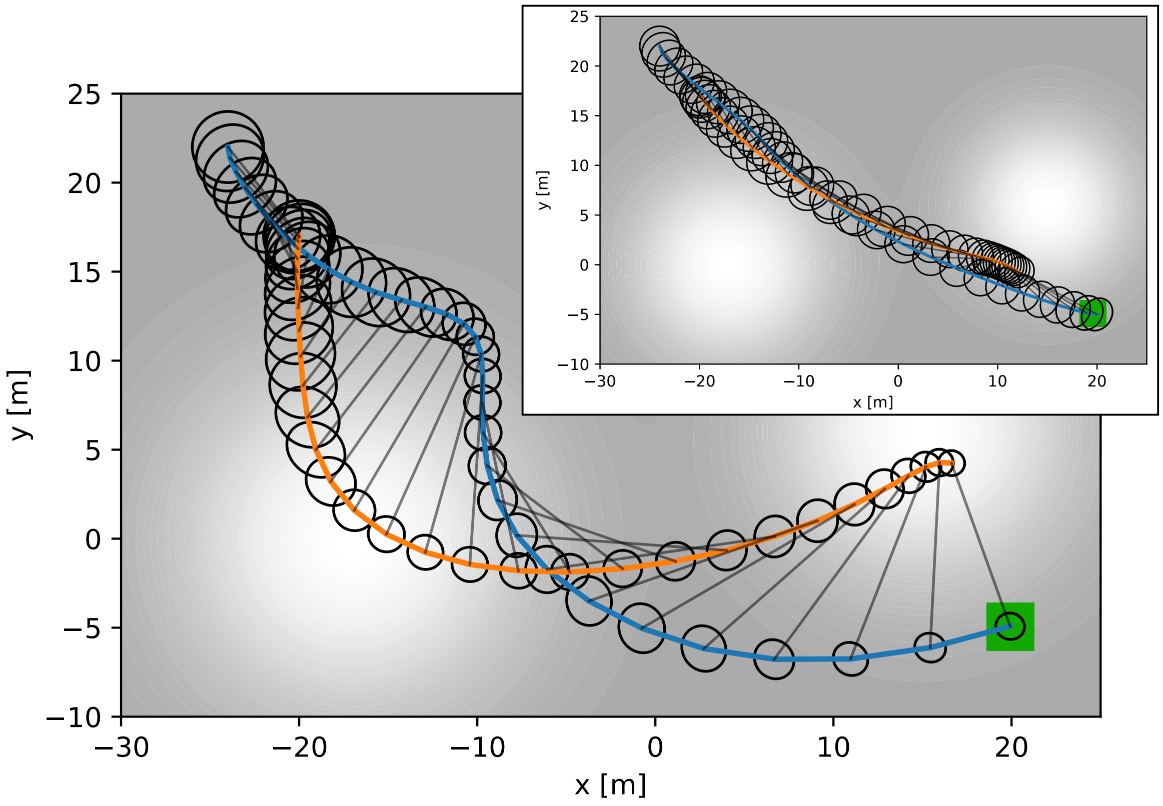


Figure 3-4: **Guide dog guides blind agent:** Guide dog (orange) with leash (black line) guides the blind agent (blue) towards the goal location (green). While doing so it passes by both light sources to reduce the uncertainty of the blind person's position at the goal location. The top right inset shows the case where the guide dog is indifferent about the blind person's uncertainty.

3.4.2 Guide Dog for Blind Agent

In this scenario, Agent 2 guides Agent 1 towards a goal location while choosing a path that reduces the uncertainty in Agent 1's position. The game is won if Agent 1 knows it reaches the goal location with a low uncertainty about its state. However, Agent 1 does not have the ability to navigate itself. We refer to Agent 1 as the "blind" agent. Following this analogy, Agent 2 acts as the "guide dog" for the blind agent. The guide dog can gather information about its own state and the blind agent's state by passing through light sources in the environment which reduces uncertainty. The agents are tethered together, which we model with spring dynamics and refer to the tether as the "leash." If the guide takes the blind agent on the direct path to the goal, the guide would not have sufficient information to know it brought the blind agent to the goal location. Under our approach, the guide dog detours to key areas to reduce the blind agent's uncertainty. We use the analogy of a guide dog leading a blind agent to create an intuitive visual for the reader, however, this system is relevant to many other robotic applications.

We model the system dynamics as two masses on a surface with friction connected by a spring tether. The states of blind agent and guide dog are $\mathbf{x}^{(i)} = [\mathbf{r}^{(i)}, \mathbf{v}^{(i)}]$ the 2D position $\mathbf{r}$ and velocities $\mathbf{v}$. The inputs $\mathbf{u}^{(i)} = F^{(i)}$ are their respective force vectors. The blind agent and guide dog have masses $c_{\text{mass,h}}$ and $c_{\text{mass,d}}$ respectively and are bound to friction coefficients $c_{\text{fric,h}}$ and $c_{\text{fric,d}}$. The accelerations are

$$\begin{aligned}
\mathbf{a}^{(1)} &= 1/c_{\text{mass,h}}(\mathbf{u}^{(1)} - f_{\text{spring}}(\Delta\mathbf{r}) - c_{\text{fric,h}}\mathbf{v}^{(1)}),\\
\mathbf{a}^{(2)} &= 1/c_{\text{mass,d}}(\mathbf{u}^{(2)} + f_{\text{spring}}(\Delta\mathbf{r}) - c_{\text{fric,d}}\mathbf{v}^{(2)}),
\end{aligned}$$

and influenced by the spring force

$$f_{\text{spring}}(\Delta r) = \frac{\Delta\mathbf{r}}{||\Delta\mathbf{r}||} c_{\text{spring}} \max(||\Delta\mathbf{r}|| - c_{\text{leash}}, 0),$$

which is dependent on the distance vector $\Delta(\mathbf{r}) = [\mathbf{r}^{(1)} - \mathbf{r}^{(2)}]$. The dog's leash is flexible with spring constant c_{spring} and has length c_{leash}, such that it only generates a spring force if extended beyond c_{leash} and is otherwise slack. The deterministic continuous dynamics are $\dot{\mathbf{x}}^{(i)} = [\mathbf{v}^{(i),\top}, \mathbf{a}^{(i),\top}]^\top$, and the discrete time dynamics

$$f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k) = \mathbf{x}_k + \dot{\mathbf{x}}_k \tau + M(\mathbf{u}_k) \cdot \mathbf{m}_k,$$

for timestep τ and where $M(\mathbf{u}_k)$ scales the motion noise proportional to the inputs $\mathbf{u}_k$.

We use the cost functions to encode the behaviors and objectives of each agent. Similar to the previous case study, minimizing $\det(\Sigma^{(1)}_{\mathbf{r},l})$ encodes the guiding Agent 2's objective of reducing the uncertainty over the position of the blind Agent 1 at the end of the planning horizon. We define

$$\begin{aligned} c^{(1)}_k(\mathbf{b}_k, \mathbf{u}_k) &= \mathbf{u}^{(1),\top}_k R \mathbf{u}^{(1)}_k + c_{\text{acc,h}} \mathbf{a}^{(1),\top}_k \mathbf{a}^{(1)}_k, \\ c^{(1)}_l(\mathbf{b}_l) &= 0, \\ c^{(2)}_k(\mathbf{b}_k, \mathbf{u}_k) &= \mathbf{u}^{(2),\top}_k R \mathbf{u}^{(2)}_k, \\ c^{(2)}_l(\mathbf{b}_l) &= \det(\Sigma^{(1)}_{\mathbf{r},l}) + ||\mathbf{r}^{(1)}_l - \mathbf{r}_{\text{goal}}||^2. \end{aligned}$$

Here, the term $||\mathbf{r}^{(1)}_l - \mathbf{r}_{\text{goal}}||^2$ drives the guide dog to relocate the blind agent to the goal. We reduce the control efforts of each agent by $\mathbf{u}^{(i),\top}_k R \mathbf{u}^{(i)}_k$, and the blind agent has the additional objective of reducing accelerations with $c_{\text{acc,h}} \mathbf{a}^{(1),\top}_k \mathbf{a}^{(1)}_k$. We use a noisy observation model

$$\mathbf{z}^{(i)}_k = h(\mathbf{x}^{(i)}_k, \mathbf{n}^{(i)}_k) = \mathbf{x}^{(i)}_k + N(\mathbf{x}^{(i)}_k) \cdot \mathbf{n}^{(i)}_k,$$

where the matrix $N(\mathbf{x}^{(i)}_k)$ scales the measurement noise based on the environment shown in Figure 3-4.

We show the resulting behavior in Figure 3-4: The dog (orange) guides the blind agent (blue) from its initial position to the blind person's goal location (green) while reducing the uncertainty of the blind agent's final state by planning a slight detour through the light sources instead of directly towards the goal location. The guide does so while also taking the complex interaction originating from the blind person's forces on the tether into account. The inset on Figure 3-4 is the path taken by the dog with no optimization over the blind agent's uncertainty. While it takes a direct path to the goal, the final uncertainty of the blind agent is large.

3.4.3 Autonomous Racing

Finally, we demonstrate our approach in competitive racing, a common problem in dynamic games. By incorporating belief-space planning into the dynamic game formulation, we show a significant increase in racing performance. This allows the agents to reduce uncertainty and decrease chance constraints, thus maneuvers like overtaking on tight road segments become possible.

In all racing runs each agent maintains a separate instance of Algorithm 6. This means that each agent separately computes their own optimal control actions, the predictions of other respective agents, and their own Nash equilibrium. No other additional information, such as state estimates, beliefs, policies, or initializations are shared among agents. Since each agent executes a separate instance of Algorithm 6, Assumption 2 may not be accurate, i.e. the belief computed by agent j over agent i may only inaccurately resemble the belief of agent i over itself. Nonetheless, we will show that despite a first-order belief assumption, the presented approach yields superior performance to all other baselines.

Each agent's state $\mathbf{x}^{(i)} = [x^{(i)}, y^{(i)}, \theta^{(i)}, v^{(i)}]$ and controls $\mathbf{u}^{(i)} = [u^{(i)}_{\text{acc},k}, u^{(i)}_{\text{steer},k}]$ are the same as in the active surveillance experiment but the different deterministic continuous dynamics are of the from

$$\dot{\mathbf{x}}^{(i)}_k = \left[v^{(i)}_k \cos\left(\theta^{(i)}_k\right), v^{(i)}_k \sin\left(\theta^{(i)}_k\right), u^{(i)}_{\text{acc},k} - c_{\text{drag,i}} v^{(i)}_k - c_{\text{slip,i}} (\dot{\theta}^{(i)})^2, \dot{\theta}^{(i)}\right]^\top,$$

with yaw rate $\dot{\theta}^{(i)} = v^{(i)}_k / L \tan\left(u^{(i)}_{\text{steer},k}\right)$, and drag- $c_{\text{drag,i}}$ and slip coefficient $c_{\text{slip,i}}$. The stochastic discrete time dynamics,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{m}_k) = \mathbf{x}_k + \dot{\mathbf{x}}_k \tau + M(\mathbf{b}_k, \mathbf{u}_k) \cdot \mathbf{m}_k,$$

are subject to noise scaled by $M(\mathbf{b}_k, \mathbf{u}_k)$ proportional to the control input $\mathbf{u}_k$ as well as the squared yaw rate $(\dot{\theta}^{(i)})^2$ of each agent i separately. The observation model

$$\mathbf{z}_k^{(i)} = h(\mathbf{x}_k^{(i)}, \mathbf{n}_k^{(i)}) = \mathbf{x}_k^{(i)} + N(\mathbf{x}_k^{(i)}) \cdot \mathbf{n}_k^{(i)},$$

is subject to noise scaled by $N(\mathbf{x}_k^{(i)})$, depending on the position on the race track map. As shown in Figure 3-1, we indicate zones of low measurement noise as red. It may be beneficial for agents to plan to drive through these low measurement noise regions to increase information gain and to reduce uncertainty.

Each agent's goal is to maximize progress along the race track while staying on the track and not colliding with other agents. We define the progress along the track for any point $\mathbf{p} = (x, y)$ as the arc-length progress $r(\mathbf{p})$ of the closest point on the centerline. Likewise, we define $d(\mathbf{p})$ as the distance of the closest point on the track to $\mathbf{p}$. We visualize both the distance transform as well as the progress transform of the race track shown in Figure 3-1 in Figure 3-5. For competitive racing, each agent tries to maximize the relative progress over other agents $r(\mathbf{p}^{(i)}) - r(\mathbf{p}^{(\neg i)})$. Consequently, agents will engage in competitive blocking and cutting behavior. We design the current and terminal costs of each agent as

$$\begin{aligned} c_k^{(i)}(\mathbf{b}_k, \mathbf{u}_k) &= \mathbf{u}_k^{(i),\top} R \mathbf{u}_k^{(i)} + c_{\text{track}}^{(i)}(\mathbf{b}_k) + c_{\text{coll}}^{(i)}(\mathbf{b}_k), \\ c_l^{(i)}(\mathbf{b}_l) &= -r(p_l^{(i)}) + r(p_l^{(\neg i)}), \end{aligned}$$

penalizing control effort by R, while $c_{\text{track}}^{(i)}(\mathbf{b}_k)$ and $c_{\text{coll}}^{(i)}(\mathbf{b}_k)$ keep the agent on the track and out of collision. We achieve this by finding the upper bound of the 2σ positional uncertainty $\Sigma_{x,y}^{(i)}$ as $\alpha = 2\sqrt{\max(\text{eig}(\Sigma_{x,y}^{(i)}))}$. We can then formulate a chance collision constraint with other agents (limiting $||\mathbf{p}^{(i)} - \mathbf{p}^{(j)}||$) and the boundary of the race track (limiting $d(\mathbf{p})$) by restricting positions in the α vicinity. Finally, to arrive at $c_{\text{track}}^{(i)}(\mathbf{b}_k)$ and $c_{\text{coll}}^{(i)}(\mathbf{b}_k)$ we convert the constraints to soft constraints, penalizing constraint violation exponentially strong, as suggested in [201]. Additionally, we also limit control inputs $\mathbf{u}_k$ by soft constraints.

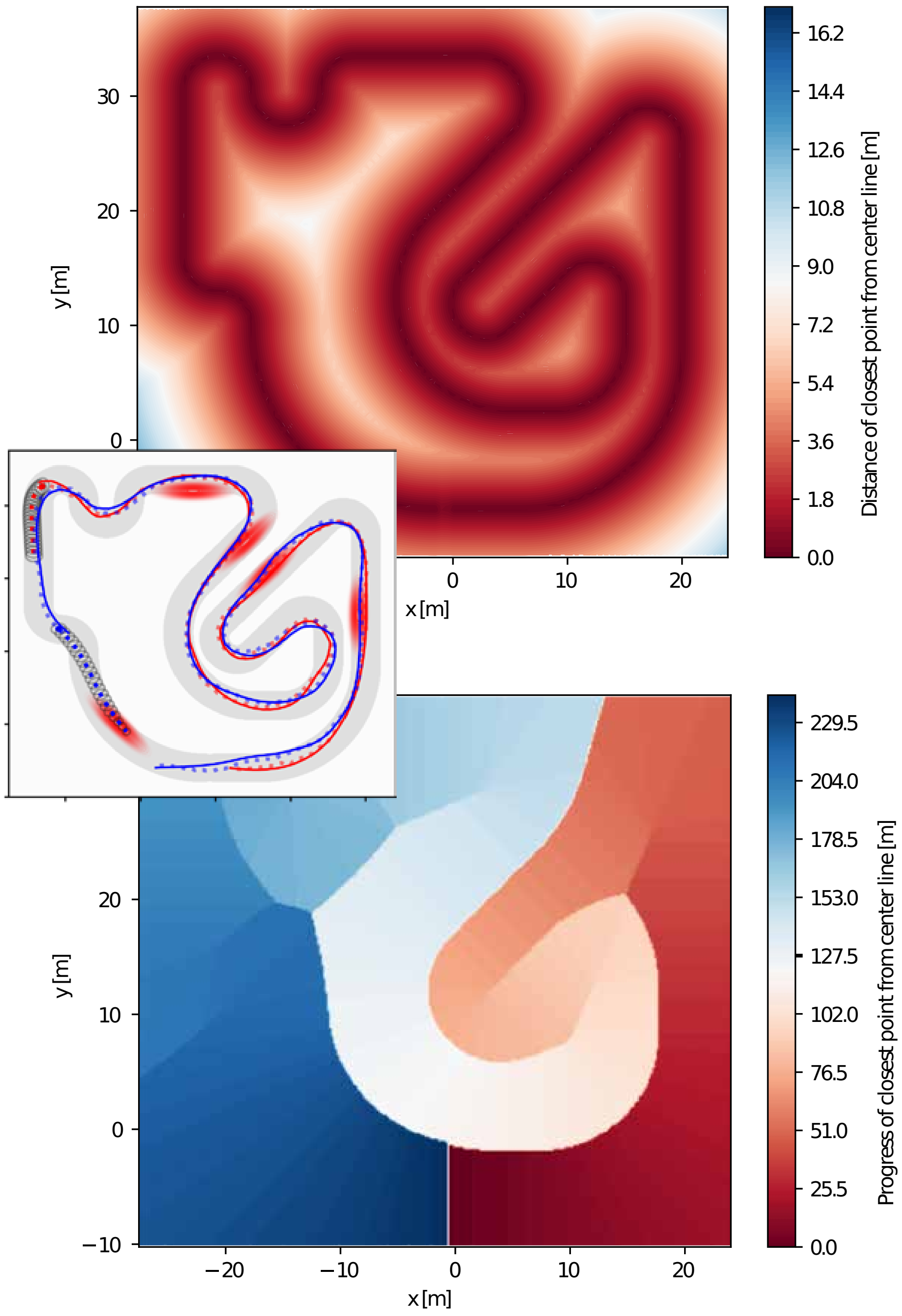
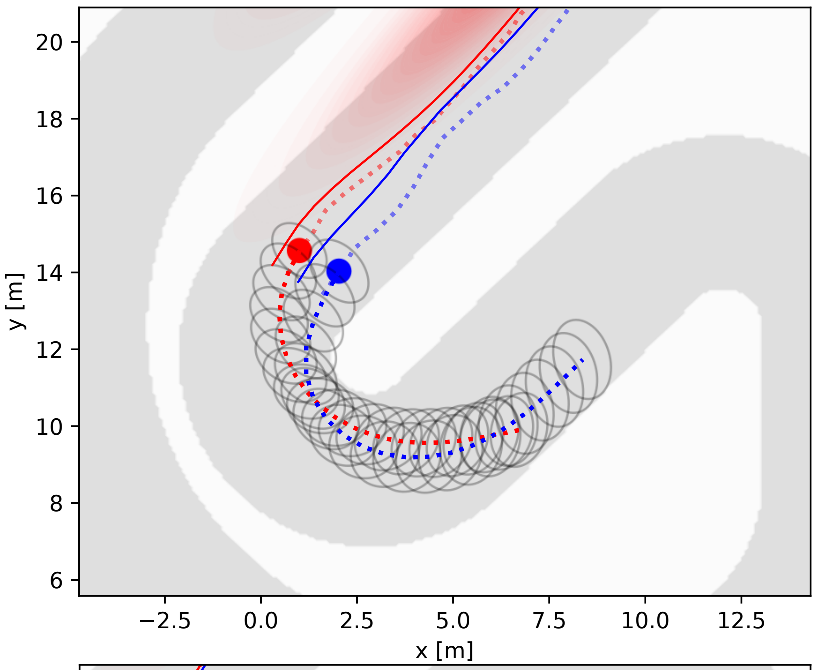


Figure 3-5: **Distance and progress transforms: Top: (Distance Transform)** Map of the distances to the closest point on the center line $d(\mathbf{p})$ of the race track shown in Figure 3-1. **Bottom: (Progress Transform)** Map of the progress $r(\mathbf{p})$ along the race track of the closest point on the center line.



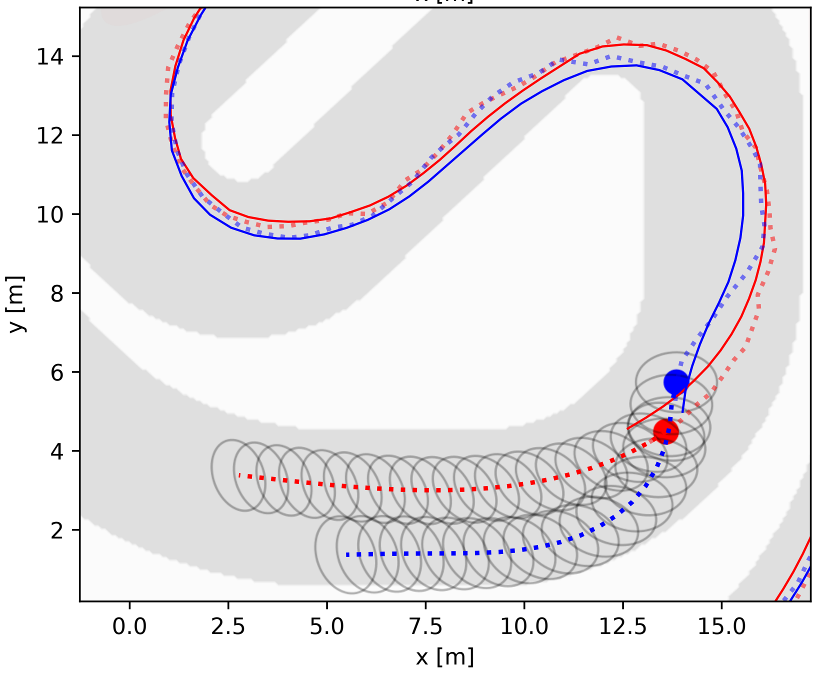


Figure 3-6: **Blocking and cutting maneuvers: Top:** Blue agent cuts in front of the red agent, forcing the red agent to break. As a result, the blue agent can remain in front of the red agent at the end of the turn. **Bottom:** The red agent blocks the blue agent's overtaking maneuver forcing the blue agent to stay behind and take a wider line in the upcoming right turn. Significant amount of noise is simulated visualized by the deviation of the true trajectory (solid lines) and the predicted mean of the belief (dashed lines).

Competitive Racing

In our racing simulation, each car executes the current commanded control computed by their own separate instance of Algorithm 6. The environment's dynamics are propagated forward subject to significant amount of noise. Subsequently, a noisy observation is generated to simulate measurement uncertainty and the current belief is updated by an EKF step. Each agent runs an individual and independent EKF and maintains their own separate belief over themselves and others. Each agent receives noisy measurements with noise drawn independently from other agents. Agents do not share any information, such as policies, measurements, initializations, state estimates, or beliefs, during online operation. To test robustness, we simulate substantial amounts of noise, such that the belief $\mathbf{b}$ may significantly deviate from the true state of the system $\mathbf{x}$, shown in Figure 3-1 and Figure 3-6.

We encourage interaction by starting one agent with lower drag coefficient (and therefore higher speed) behind another *slower* agent. The *faster* agent will eventually catch up to the previous agent and initiate an overtaking maneuver. The better interactions are predicted and integrated into planning, the more successful overtaking maneuvers will occur.

The algorithm described in this chapter is able to synthesize competitive emergent behavior such as blocking of other vehicles and cutting in front of others, illustrated in Figure 3-6. Additionally, although tight racing lines cut corners very closely, the chance constraints are successful in prohibiting collisions under the presence of motion and observation noise.

Benefits of Dynamic Game Planning

We compare the performance of Dynamic Game (DG) planning to conventional methods such as Model Predictive Control (MPC). Both DG and MPC agents plan in belief space. The MPC agent has the exact same cost structure, but observes the other agents' executed actions and predicts agents to continue with the same action. The MPC baseline therefore predicts agents to not react to changes of their own actions

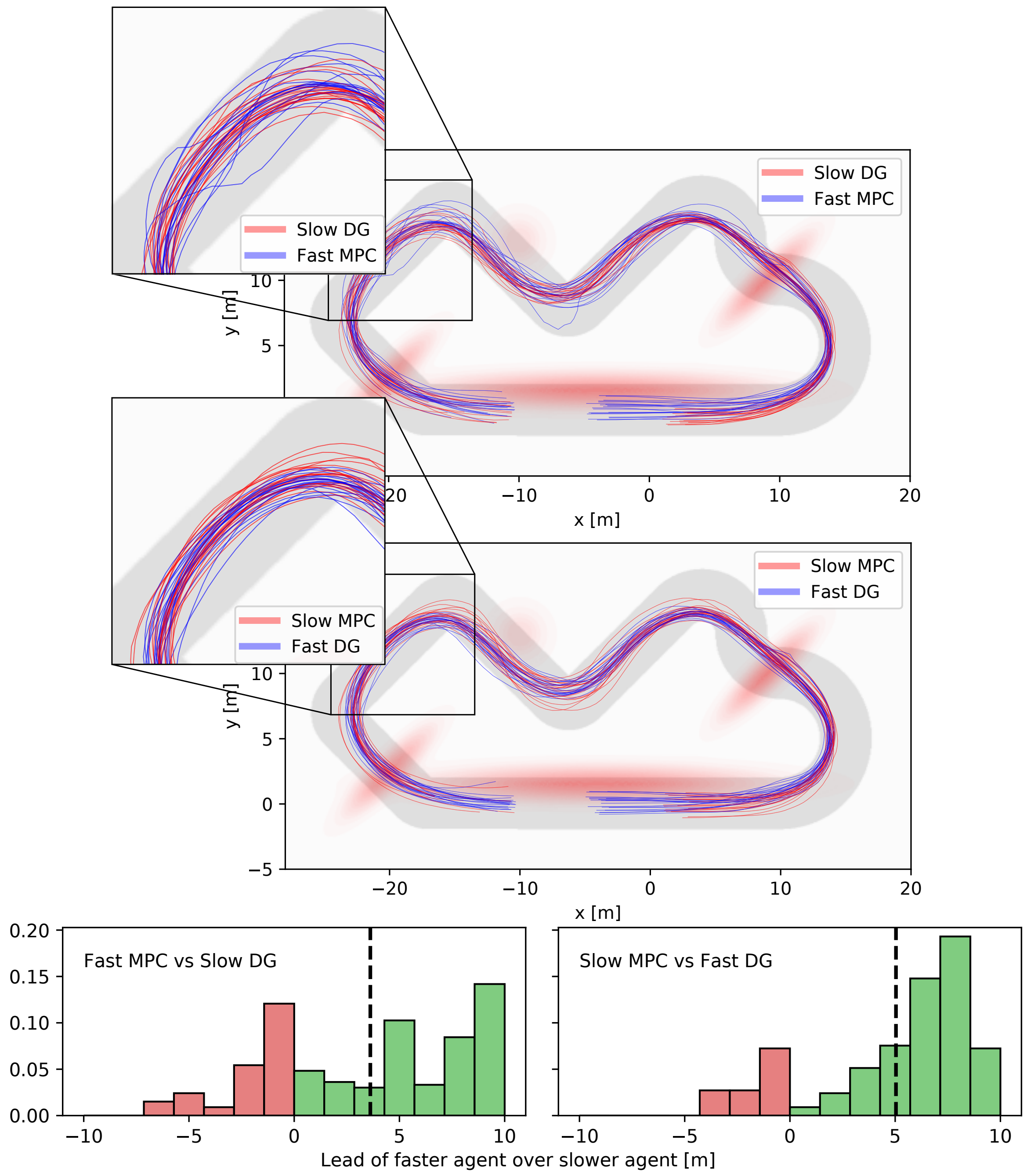


Figure 3-7: **MPC vs DG traces and racing results: Top:** Traces of agents comparing MPC and DG. In both cases the MPC method moves away from the ideal racing line more often due to failed overtaking attempts. It can not foresee it's influence on the DG agent's actions and thus is less efficient. It is also not able to take advantage of estimating is implicit control over the other agent like the DG agent. The agents start from random initialization locations around the origin. **Bottom:** Histograms of the Δarc-length lead of the faster agent over the slower agent. Green indicates that the faster agent won the race against the agent starting in the lead, whereas red indicates the opposite. In comparison, the DG method won more races than the MPC method and had a higher average lead.

Table 3.2: Racing Performance: DG vs MPC and BSP vs non-BSP

Competition Pair	Fraction of Fast winning
Fast DG BSP vs Slow MPC BSP	**82%**
Fast MPC BSP vs Slow DG BSP	64%
Fast DG BSP vs Slow DG non-BSP	**77%**
Fast DG non-BSP vs Slow DG BSP	63%
Fast DG BSP vs Slow DG BSP	**67%**

Table 3.3: Racing Performance: Winning Ratio

Competition Pair	Win ratio
DG BSP vs MPC BSP	**1.44:1**
DG BSP vs DG non-BSP	**1.33:1**

and cannot leverage the effects of their own actions on other agents. The MPC is capable of synthesizing competitive racing trajectories, shown in Figure 3-7, which are identical to the DG trajectories when no other agents are present. The performance of the DG planning distinguishes itself when interactions occur.

We display the results of 200 runs in Figure 3-7, Table 3.2 and Table 3.3. The DG method wins 44% more races relative to the MPC baseline and has a larger lead on average. These results clearly illustrate the competitive advantage of our game-theoretic algorithm from leveraging how others react to one's own actions when planning.

Benefits of Belief-Space Planning

We also compare the performance of DG planning with and without Belief-Space Planning (BSP). In the non-BSP case the current uncertainty Σ_0 of the belief $\mathbf{b}_0$ is held constant over the planning horizon and is not influenced by expected measurements. Note that the current belief is still updated online by an EKF for both agents. Results are reported in Figure 3-9 and Table 3.2. The BSP variant wins 33% more races, has a larger average lead, and the fewest number of collisions. The non-BSP method collides nearly 10 times more often and exhibits behavior inappropriate for observed uncertainty levels. For example, agents are too conservative because low noise regions are not considered in the planning phase, or too aggressive when enter-

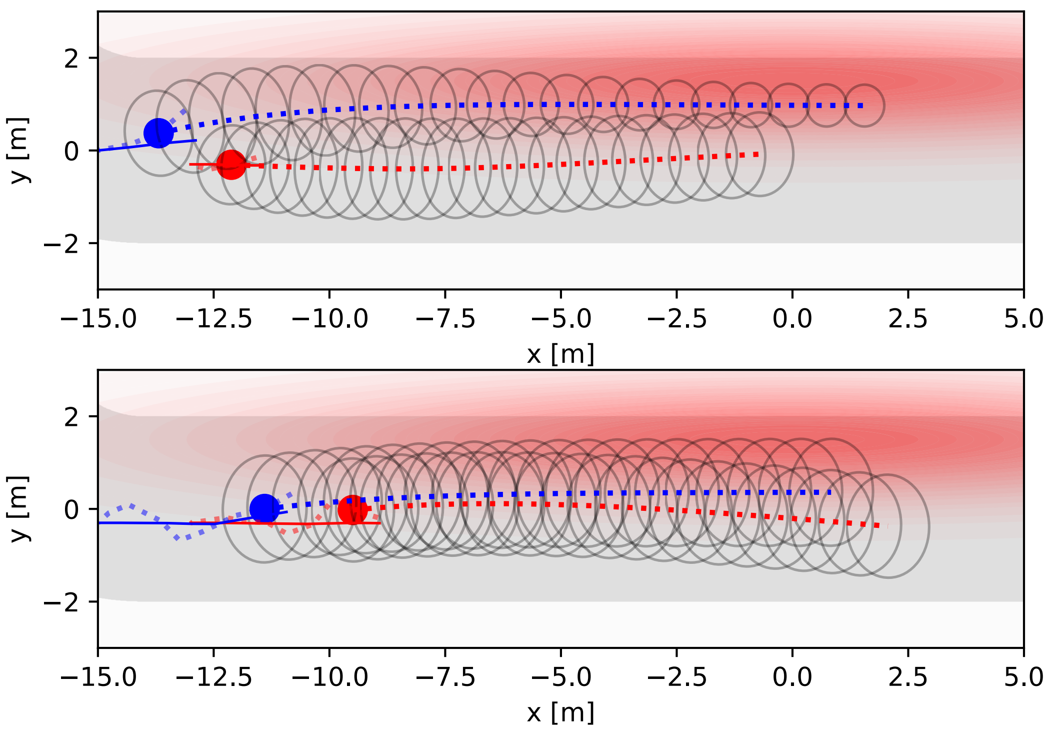


Figure 3-8: **Leveraging BSP for overtaking: Top:** The blue agent overtakes the red agent by decreasing the uncertainty through the low noise region and reducing the chance constraint (ellipses). **Bottom:** The blue agent has the same uncertainty over the planning horizon and fails to overtake since the chance constraints remain large.

ing sharp turns since additional motion noise due to breaking and steering are not accounted for.

Figure 3-8 gives an intuitive explanation for the competitive advantage of planning in belief space. Without information gain, the follower will never be able to overtake due to the large chance constraint. Whereas with information gain, the chance constraint shrinks while moving through a low noise zone, allowing the blue agent to overtake the leading agent. As shown in Figure 3-9, the BSP agent can adapt their trajectories to account for increased noise due to strong actuation, i.e. braking and steering, and gaining information in low noise regions.

Finally, we compare the performance when both agents use DG BSP, Table 3.2, and see that the faster agent wins 67% of the races. Since the performance gain of the faster over the slower agent is smaller than in the previous two comparisons, we assume that the slower agent improves their blocking behavior more than the faster agent improves their ability to overtake. In scenarios where high uncertainty causes the chance constraints to occupy large parts of the track's width, the slower agent can often block the faster agent by proceeding in the middle of the road.

3.4.4 Real-Time Implementation Details

We implement our solver in the CasADi [20] framework leveraging auto-differentiation by source code transformation, automatic problem specific compute graph generation, C-code generation, and sparse operations. Exploiting sparsity is highly important to allow for real-time performance since the belief space, encompassing the mean state and the upper triangle of the covariance matrix can make respective Jacobian and Hessian matrices very large. The average compute times on a Ryzen 7 1700X 3.4 GHz are reported in Table 3.4. Algorithm 1 was run until convergence starting from a cold start for all experiments, i.e. the initial control trajectory $\mathbf{u}$ consists of all zeros. Nonetheless, it is also possible to run the algorithm sequentially by hot starting the optimization with the previous solution. This is common practice in related optimization techniques for controls such as sequential quadratic programming [139] and allows to run Algorithm 6 at 100-200Hz. In these cases it is often enough to run

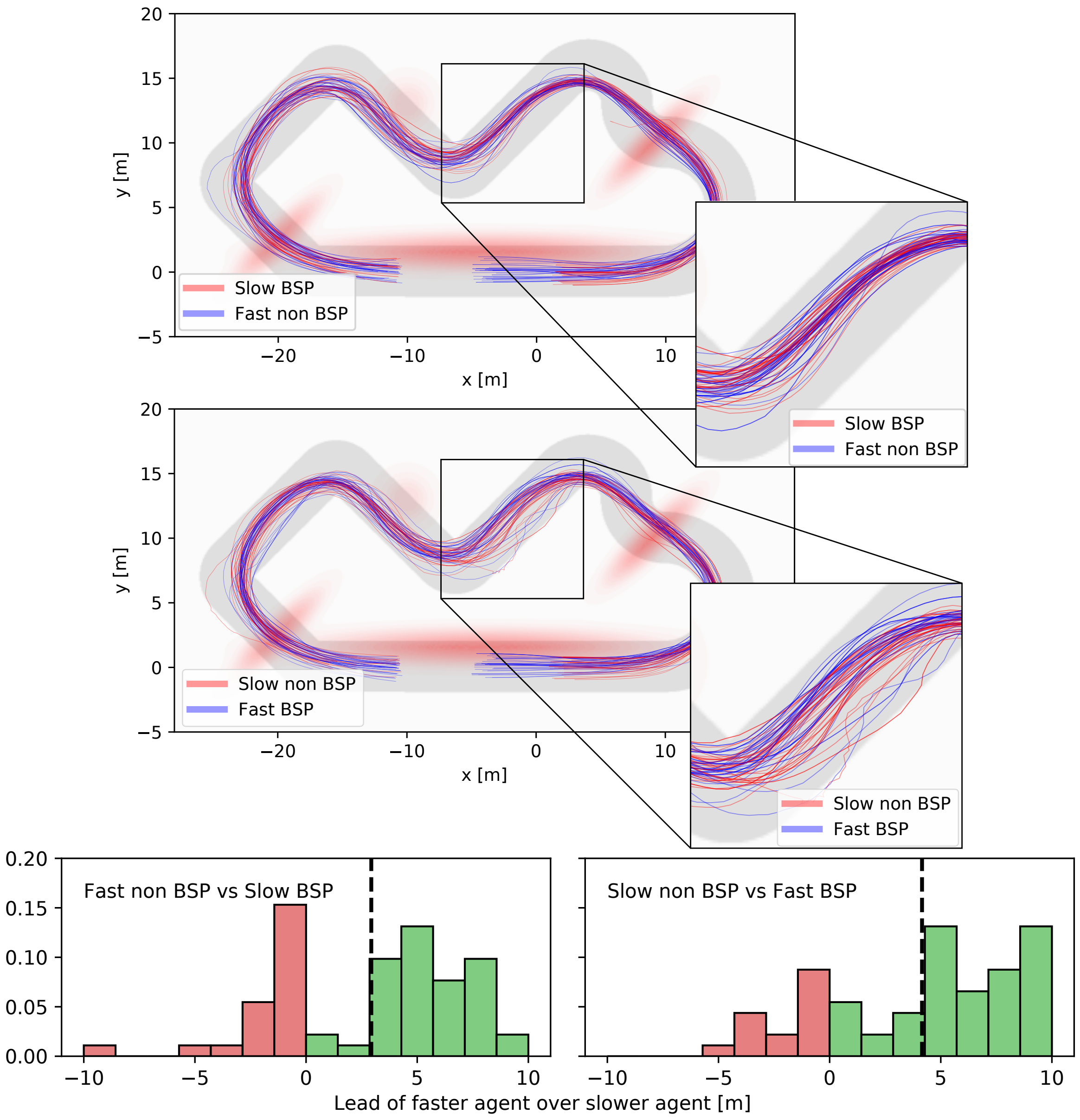


Figure 3-9: **BSP vs non-BSP traces and racing results: Top:** Traces of agents comparing BSP and non-BSP. In both cases the non-BSP method shows more unsafe behavior, leaving the track several times and nearly colliding with the other agent. The BSP agent attempts more aggressive overtaking maneuvers due to the lower uncertainty estimate over itself and the other agent, as shown in the cutout. The agents start from random initialization locations around the origin. **Bottom:** Histograms of the Δarc-length lead of the faster agent over the slower agent. Here, the BSP method won more races than the non-BSP method and had a higher average lead.

only very few iterations to update the previous solution.

Table 3.4: Average computation time

Experiment	Per iteration	Until convergence
Active surveillance	9.3 ms	371.3 ms
Guide dog	11.2 ms	474.9 ms
Racing	5.8 ms	110.5 ms

3.5 Discussion

In this chapter, we propose a formulation for integrating belief-space planning into dynamic games, and present a real-time algorithm for solving the local Nash equilibria of these dynamic games in belief space. We demonstrate its performance of combining game-theoretic planning and information gathering with three case studies: active surveillance, guiding blind agents, and racing with autonomous vehicles.

While game-theoretic planning models the interaction and dependency among agents, it does not address the quality of information available to the agent for decision making. Incorporating belief-space planning in dynamic games allows for new capabilities not possible with other approaches, essential in house service robots or interacting with human agents in traffic. Reasoning about another agent's uncertainty and simultaneously leveraging the effect of own actions on other agents' actions results in complex emergent behavior such as indirectly pushing and guiding others through regions of light, without the use of any form direct of communication.

In competitive use cases such as racing, emergent behavior consists of cutting, blocking, forcing others to break hard with the goal of increasing their uncertainty and slowing them down in turns, as well as the exploitation of high information-gain zones for overtaking. In particular, game-theoretic belief-space planning significantly increased performance in dynamic racing when benchmarked against state-of-the-art planning methods. Game-theoretic belief-space planning wins 44% more races when competing with a non-game-theoretic baseline with belief-space planning and 34% more races than a game-theoretic baseline without belief-space planning.

In this work we limit ourselves to first-order beliefs to avoid the explosion in parameters for recursive beliefs over beliefs. Nonetheless, even in cases where a first-order belief assumption is a simplification of the true belief dynamics, such as racing, we see improved performance to baselines that do not take the belief over other agents into account. In future work we intend to develop extensions beyond first-order belief spaces.

We efficiently solve for Nash equilibria in belief space and achieve real-time performance, operating our algorithm at more than 100Hz. Efficiently solving a quadratic game at each stage of the recursive backward pass of a belief-space variant of iLQG results in an algorithm with runtime complexity $\mathcal{O}(lN^7 n_{\mathbf{x}}^{i,6})$. Linear complexity in the planning horizon allows for online deployment, in comparison to point-based POMDP algorithms with exponential complexity. While our algorithm also achieves polynomial runtime complexity in the number of agents N, future work will investigate lowering the complexity further.

In the presented work so far, we have learned reward functions from human data and leveraged known models for dynamics and observations for game-theoretic planning. We will now drop the assumption of known models and learn to compete from scratch without any prior knowledge about the world. Additionally, while the previous work had access to proprioceptive state estimation, we will present agents that learn competitive control policies directly from raw image observations.

Chapter 4

Learning from Competition: Learning to Race Using Visual Control Policies in Latent Space

4.1 Introduction

Driverless racing is a challenging task promising to push the limits of autonomous navigation as it requires robust operation of the entire driving stack at high speeds. Racing becomes particularly challenging when multiple agents simultaneously compete within the same environment. This requires both, fast processing of observations and reasoning about opponent behavior. By addressing these challenges, racing can provide novel insights for the deployment of autonomous systems. In the context of Multi-Agent Reinforcement Learning (MARL), racing can benchmark competitiveness as it requires reasoning about the interplay between ego actions and opponent behavior. This is particularly challenging when operating under partial observability in high-dimensional input spaces and in the absence of structured priors over environment behavior.

Recent approaches to autonomous racing tend to assume access to a nominal environment model [9, 44, 120]. Some also leverage game-theoretic frameworks such as

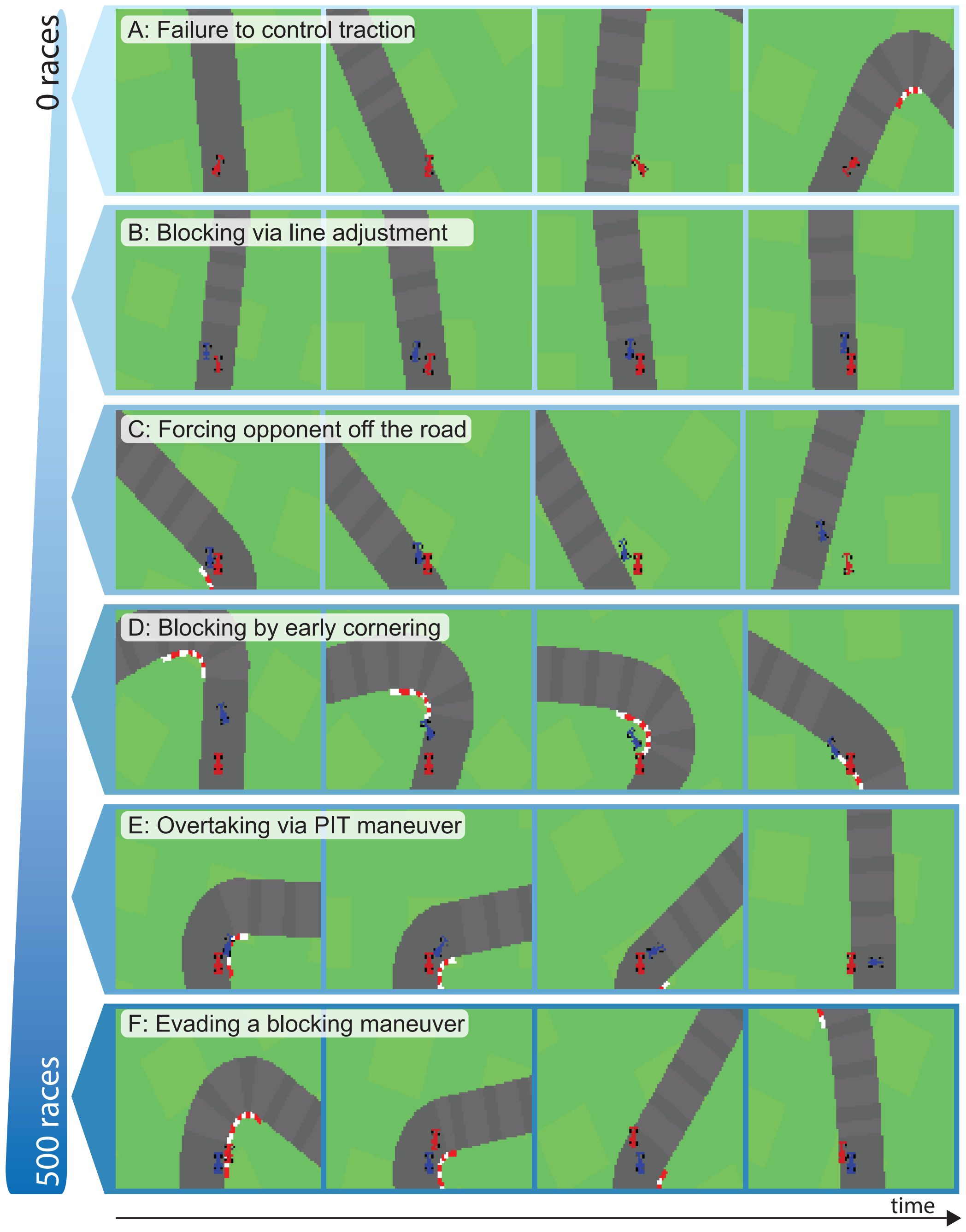

0 races
500 races
A: Failure to control traction
B: Blocking via line adjustment
C: Forcing opponent off the road
D: Blocking by early cornering
E: Overtaking via PIT maneuver
F: Evading a blocking maneuver
time

Figure 4-1: **Skill acquisition from experience:** The agent learns increasingly competitive behaviors as training progresses.

iterated best response [183, 214, 119]. However, in racing and other real-world multi-agent settings, relevant behavioral and environmental features may be too complex to be captured by MPC or purely game-theoretic approaches that assume perfect knowledge of the state and the system dynamics. MARL provides an alternative framework for modeling rich agent interactions. Impressive successes have been demonstrated for large-scale competition in discrete action spaces [208] with access to privileged ground truth information such as categorized entity lists and multi-layer maps. Continuous control applications of MARL in competitive settings that only provide image observations are less well-studied. In particular, combining high-dimensional observations such as images, partial observability of other agents, multi-agent world model learning, and acquiring complex competitive behaviors through self-play is still a challenge.

4.1.1 Main Assumptions

Human Formula 1 drivers may assume that other drivers' cars handle in similar ways to their own vehicle. Similarly, we assume that our vehicle and the vehicles of other agents have comparable performance and handling characteristics. Likewise, we assume that the competitors' reward structure is similar to our own. Drivers seek to make progress along the track and stay in the lead. These assumptions allow us to make rational predictions about the behavior of other agents. Nonetheless, in comparison to the previous chapters, the functions for reward, dynamics, and observation are unknown and have to be inferred from collected experience. In this work, we use top-down views of the vehicle and the track. While the presented approach may extend to more general and less advantageous perspectives, exploring these is beyond the scope of this chapter. We do not assume access to any privileged information such as the locations of the track or other vehicles. Instead, our method operates on raw image observations. During the online phase of our algorithm, we do not assume access to the opponent's view or controls. Instead, we learn to predict the opponent's view in a latent space based on our observations. Nonetheless, during the offline phase of the algorithm, we assume access to both the own and the opponent's observations and controls of past episodes and races.

4.1.2 Contributions

We address the outlined research gap by proposing Deep Latent Competition (DLC), a novel model-based MARL method that operates on raw image inputs. While this method can be applied to a range of problems, we focus on demonstrating it in the context of two-player racing. Our approach learns a world model for imagining competitive behavior in latent-space. This allows for training agents via imagined self-play such that they can predict opponent behavior and incorporate the expected outcomes of action sequences in their policy selection. We further learn to predict the belief of other agents purely based on observations from the ego agent's perspective. We validate this methodology on a novel multi-agent racing benchmark based on OpenAI Gym [37] that requires the application of continuous visual control policies. In summary, this work contains the following contributions:

1. A novel model-based reinforcement learning algorithm for **learning competitive control policies** for multi-agent problems from raw image observations through self-play in latent space.
2. A multi-agent world model structure that allows for **(a) imagination of competing agents' behavior** in a learned latent space and **(b) estimation of the beliefs of others** from own observations.
3. **Extensive evaluations** in a **new multi-agent racing benchmark** demonstrating superiority over approaches that do not reason about other agents in imagination.

4.2 Related Work

4.2.1 Autonomous Racing and Navigation

Most recent approaches considering autonomous racing assume knowledge of the underlying dynamics model and use machine learning based techniques for improving said model. This can be employed in conjunction with a variety of control approaches [9, 44, 120, 172]. Game-theoretic methods additionally explicitly model interactions with other agents for decision making [174, 183, 213, 214, 216, 119, 173]. These approaches usually do not operate on high-dimensional input spaces and often impose assumptions on the type of interactions. On the other hand, learning-based end-to-end navigation approaches [154, 12, 28, 33, 96, 14, 13] can operate directly on high-dimensional sensor data but typically involve no inductive biases for considering interactions.

4.2.2 Multi-Agent Reinforcement Learning

Recently, MARL agents have surpassed human-level performance in many multi-agent environments including complex board and card games such as Go [182], chess, shogi [181], and Poker [131, 39]. MARL agents also reached grandmaster-level performance in the real-time strategy game Star Craft II [208, 209], and showed complex emergent tool use in hide-and-seek [25]. We are motivated by the success of recent algorithms for continuous control tasks in cooperative, competitive, and team competition environments [112, 90, 122, 29]. While these agents leveraged privileged information, such as entity lists, states, and maps, we present an agent that learns competitive strategies directly from raw image observations. A common thread in MARL is self-play auto-curricula [123, 104, 83, 84]. In contrast, we gain competitiveness from imagined self-play in a learned multi-agent latent world model.

4.2.3 Latent Imagination in RL

Use of neural networks, particularly recurrent neural networks, for modeling the evolution of the environment allowing for *"mental imagination"* has been proposed as early as 1990 [166] and recently revisited in [74]. In a similar spirit, variational inference approaches have been combined with the linear-quadratic-regulators for learning to control from raw images [215, 220]. Another line of recent algorithms combines latent (multi-step) imagination with video prediction [95, 76, 176], achieving state-

Table 4.1: Deep Latent Competition: Main Symbols and Notation

S, A^i	set of environment states, continuous action set
$T : S \times A^1 \times \cdots \times A^n \rightarrow \Pi(S)$	transition function
$R^i : S \times A^i \rightarrow \mathbb{R}$	reward function
$O^i : S \times A^i \rightarrow \Pi(\Omega^i)$	observation function
o_t^i, r_t^i	observation and reward of agent i at time t
a_t^i, s_t^i, z_t^i	action, latent state, latent embedding
θ, ψ	model and policy parameters
$\mathcal{D}$	replay buffer
L, H	batch sequence length, trajectory rollout length
K	number of episodes
γ, λ	discount factor, exponential recency factor
n, T	number of agents, time horizon

of-the-art performance on several standard benchmarks. We draw inspiration from these ideas and generalize the concept of multi-step latent imagination to multiplayer settings.

4.3 Representing Multi-Agent World Models

We define competitive visual control in our racing domain as a MARL problem. A collection of agents interacts within the environment and learns to optimize their behavior in an effort to maximize individual cumulative reward. MARL is typically modelled as a Markov game [177, 121, 71], in which each agent solves a partially observable Markov decision process (POMDP) [94, 79, 193]. In the following, we first introduce the general problem definition, outline our approach to imagined self-play and introduce our representation learning approach.

4.3.1 Problem Formulation of Competitive MARL

We define the POMDP of agent $i \in \{1, \ldots, n\}$ as the tuple $M^i = \langle S, A^i, T, \Omega^i, O^i, R^i \rangle$, where S denotes the set of environment states, A^i the continuous action set, and $T : S \times A^1 \times \cdots \times A^n \rightarrow \Pi(S)$ the corresponding transition function with associated probability distribution $\Pi(\cdot)$. For each agent i, we furthermore define a reward function $R^i : S \times A^i \rightarrow \mathbb{R}$, as well as an observation set Ω^i with corresponding observation

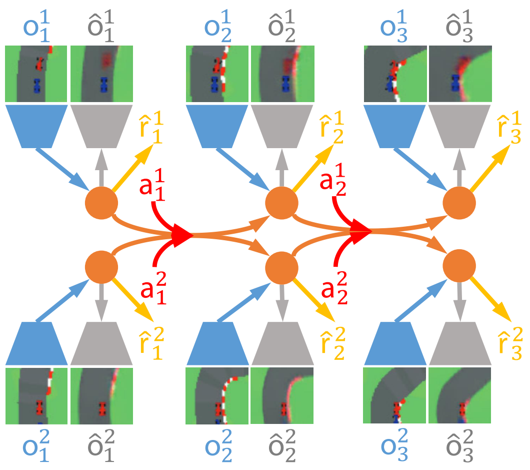


Figure 4-2: **Model learning:** The agent learns to encode observations into separate latent states for each agent based on reconstruction and reward prediction. The learned transition model propagates all agents' latent states jointly.

function $O^i : S \times A^i \rightarrow \Pi(\Omega^i)$. In the following, we consider a homogeneous set of agents with identical action and observation spaces as well as reward functions, such that $X^i = X$ for $X = \{A, \Omega, O, R\}$.

We do not assume prior knowledge about the environment. Thus, the nominal reward function R, state transition function T, and observation function O are unknown. Let $q_\phi(a_t^i | o_{\leq t}^i, a_{<t}^i)$ denote the policy of agent i, conditioned only on its own observation-action history, and define the associated expected return over a race of duration T to be $\mathbb{E} \sum_{t=1}^{T} r_t^i$. The objective is then to develop an agent that maximizes the expected return in the absence of prior knowledge about nominal environment and opponent behavior. This defines an extensive-form game as the agents are competing for reward over multiple timesteps, while only receiving instantaneous environment feedback via high-dimensional observations o_t^i and scalar rewards r_t^i.

4.3.2 Learning through Imagined Self-Play

Model-based reinforcement learning consists of the tasks of (a) model learning, (b) behavior optimization, and (c) environment interaction. Model-based MARL extends model learning to include predictions of other agents' behavior, while behavior optimization needs to account for competitive fitness. The training proceeds central-

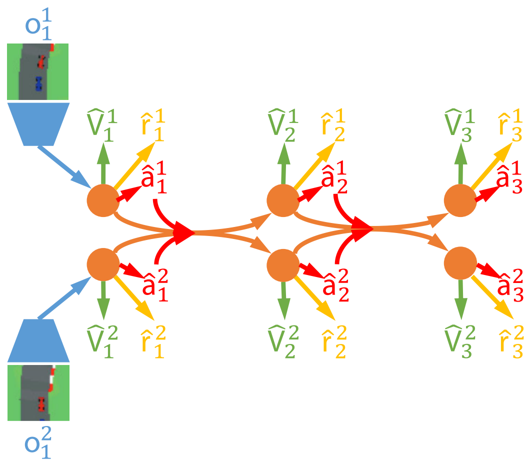


Figure 4-3: **Self-play in imagination:** The agent predicts state values and optimizes actions that maximize future returns by propagating gradients back through imagined game trajectories. The agent's competitiveness continuously improves through self-play.

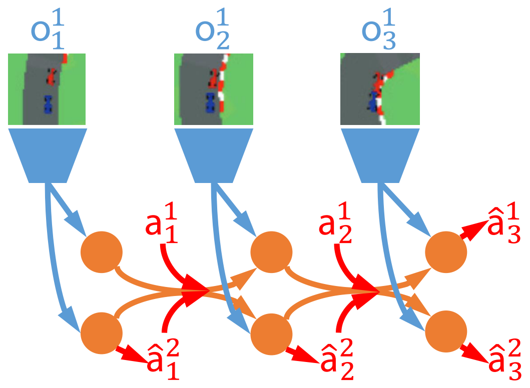


Figure 4-4: **Environment interaction:** The agent estimates the own and other agents' current state based on an encoding of own historic observations only and predicts the actions of the other agent and the own actions to be executed in the environment. True observations and actions of other agents are not available.

ized. While the learning algorithm has access to the observation-action histories of all agents, the deployment is decentralized, providing each agent only with their individual observation-action history. As detailed in Figures 4-2,4-3,4-4 our algorithm iteratively executes the following:

1. Learning a world model consisting of the joint dynamics and reward function based on previous experience of all agents, see Figure 4-2. Learning to predict how the world evolves conditioned on own and expected opponent actions enables each agent to imagine the outcome of games without requiring additional real-world experience.

2. Learning action and value function models through policy iteration on imagined model rollouts. The agents interact with their adversaries through imagined self-play and acquire increasingly competitive behaviors without the necessity for execution in the real world, see Figure 4-3.

3. Competing in the real world to collect novel experience and judge the performance of the current behavior. Each agent only has access to their own observation-action history and performance indirectly depends on how well the states of opponents are being estimated, see Figure 4-4.

Representing Multi-Agent World Models

Leveraging a multi-agent world model accelerates learning through imagined self-play. The agent optimizes its behavior by simulating interactions with its opponents in the environment without execution in the real world. In contrast to single-agent world models [215, 220, 77], this requires explicit representation of the state and action of each agent, as well as a mechanism for an agent to predict behavior of another agent. In the following, we will consider a two-player game ($n = 2$) and extend the representation model formulation provided in [76] to yield:

$$\begin{aligned}
&\text{Representation model:} && p_\theta\left(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2\right), \\
&\text{Transition model:} && q_\theta\left(s_t^1, s_t^2 | s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2\right), \\
&\text{Encoder model:} && q_\theta\left(z_t^i, \tilde{z}_t^{\neg i} \middle| \, o_t^i\right), \\
&\text{Observation model:} && q_\theta\left(o_t^i | s_t^i\right), \\
&\text{Reward model:} && q_\theta\left(r_t^i | s_t^i\right),
\end{aligned} \tag{4.1}$$

where p and q denote distributions in latent space, with θ as their joint parameterization. The representation model encodes observations (o_t^1, o_t^2) into Markovian model states (s_t^1, s_t^2) [76], which are propagated under a joint transition function to predict future model states (s_{t+1}^1, s_{t+1}^2). We explicitly provide the underlying encoder model to emphasize that each agent not only learns an embedding corresponding to its own viewpoint, z_t^i, but additionally learns to predict embeddings of the opponent, $\tilde{z}_t^{\neg i}$. This is crucial during deployment, as ground truth observations and actions of opponents are not available. Instead, we leverage our predicted embeddings $\tilde{z}_t^{\neg i}$ in conjunction with an action model $q_\phi\left(a_t^{\neg i} | s_t^{\neg i}\right)$ and employ a slightly modified representation model $p_\theta\left(s_t^i, s_t^{\neg i} | s_{t-1}^i, s_{t-1}^{\neg i}, a_{t-1}^i, a_{t-1}^{\neg i}, o_t^i\right)$ that is only conditioned on the observation of the ego agent, o_t^1. The encoder model is part of the representation model and embeds observations into embedding states based on which model states are then generated. For each agent, we furthermore define an individual observation model $q_\theta\left(o_t^i | s_t^i\right)$ and reward model $q_\theta\left(r_t^i | s_t^i\right)$. The underlying architectures follow [77, 76], where the transition model is represented by a recurrent state space model (RSSM), the encoder and observation models by a convolutional neural network (CNN) and transposed CNN, respectively, and the reward model by a dense neural network. Training of these models then proceeds centralized, such that the learning algorithm is given access to the interaction histories of each agent.

4.4 Learning to Compete by Imagined Self-Play

Our proposed algorithm learns a world model from ground truth data, based on which behavior is refined through imagined self-play. In the following, we introduce the objectives for the two stages.

4.4.1 Representation Learning

The representation learning objective combines image reconstruction with reward prediction in order to discover latent spaces that not only offer compact representations of environment states but further facilitate prediction of associated trajectory performance. Because agents are actively competing for reward, their states are propagated jointly through the transition model with each agent learning to predict relevant opponent states. The observation model then not only provides reconstruction signals for the ego perspective via the true ego state s_t^i, but also for the opponent perspective via the predicted opponent state $\tilde{s}_t^{\neg i}$. Following [76], the models introduced in Eq. (4.1) are optimized to maximize a reformulation of their variational lower bound objective:

$$\begin{aligned}
J_{M,\hat{s}} &= \mathbb{E}_{\mathcal{D}}(\textstyle\sum_t (J_{O,t} + J_{R,t} + J_{D,t})), \qquad (4.2)\\
J_{O,t} &= \ln q(o_t^1|\hat{s}_t^1) + \ln q(o_t^2|\hat{s}_t^2),\\
J_{R,t} &= \ln q(r_t^1|\hat{s}_t^1) + \ln q(r_t^2|\hat{s}_t^2),\\
J_{D,t} &= -\beta\, \mathrm{KL}(p(\hat{s}_t^1, \hat{s}_t^2|\hat{s}_{t-1}^1, \hat{s}_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2) \parallel q(\hat{s}_t^1, \hat{s}_t^2|\hat{s}_{t-1}^1, \hat{s}_{t-1}^2, a_{t-1}^1, a_{t-1}^2)),
\end{aligned}$$

where the general model state $\hat{s}$ may either originate from ground truth embeddings s or their predictions $\tilde{s}$. In practice, we optimize a linear combination of $J_{M,\hat{s}}$ with $\hat{s} = \{(s_t^1, s_t^2), (s_t^1, \tilde{s}_t^2), (\tilde{s}_t^1, s_t^2)\}$. This enables learning of latent representations conducive to solving the task via the ground truth embeddings, while constraining predicted embeddings to sensible representations within that space.

4.4.2 Behavior Learning

The behavior learning objective optimizes for competitive fitness by maximizing the expected return of the action model $q_\phi(a_t^i|s_t^i)$ over a T-step race. The agent can imagine outcomes of potential interaction sequences by leveraging the learned transition model, therefore bypassing execution in the real world through imagined self-play. Generating entire race sequences can be computationally prohibitive and we follow [76] in complementing finite horizon model rollouts with predictions from a value model $v_\psi(s_t^i)$ in order to approximate returns corresponding to an extensive form race. The action and value model are then trained jointly using policy iteration on the objectives

$$\max_\phi \mathbb{E}_{q_\theta,q_\phi}\Big(\textstyle\sum_{\tau=t}^{t+H} V_\lambda(s_\tau)\Big), \qquad \min_\psi \mathbb{E}_{q_\theta,q_\phi}\Big(\textstyle\sum_{\tau=t}^{t+H} \|v_\psi(s_\tau) - V_\lambda(s_\tau)\|^2\Big), \tag{4.3}$$

where $V_\lambda(s_\tau)$ represents an exponentially (λ) recency-weighted average of the k-step value estimates $V_N^k(s_\tau)$ to stabilize the learning [193]. We provide the corresponding value function definitions as

$$\begin{aligned} V_\lambda(s_\tau) &= (1-\lambda)\textstyle\sum_{n=1}^{H-1} \lambda^{n-1} V_N^n(s_\tau) + \lambda^{H-1} V_N^H(s_\tau), \\ V_N^k(s_\tau) &= \mathbb{E}_{q_\theta,q_\phi}\Big(\textstyle\sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h)\Big), \end{aligned} \tag{4.4}$$

where $h = \min(\tau + k, t + H)$. This process leverages imagined trajectories $\{(s_\tau^1, s_\tau^2, a_\tau^1, a_\tau^2)\}_{\tau=t}^{t+H}$ over a horizon of H starting from each timestep t within the sampled batch sequence, where opponent behavior is estimated by querying the action model with predicted latent viewpoints as $q_\phi(a_t^{\neg i}|\tilde{s}_t^{\neg i})$. The resulting algorithm optimizes policies by back-propagating analytic value gradients of imagined self-play trajectories through the learned multi-agent world model.

4.4.3 Deep Latent Competition

The resulting algorithm is provided as pseudocode in Algorithm 7. It runs for K episodes and proceeds in two phases: during the online phase, data is collected from a T-step race with each agent only having access to their respective observations and relying on predicted opponent behavior. During the offline phase, representation learning and policy iteration propagate information into the world model and the action model based on the interaction histories of all agents. To this end, batch sequences of length L are sampled from replay memory $\mathcal{D}$ to serve as targets for representation learning. The behavior is then refined in simulation based on rollout trajectories of length H starting from the ground truth samples which are used in generating the value estimates according to Eq. 4.4. Here, we set the underlying parameters to $T = 1000$, $L = 50$ and $H = 15$. All models are then optimized on the previous objectives with the Adam optimizer [97].

Algorithm 7 Deep Latent Competition (DLC)

```
1: Initialize: model parameters $\{\theta, \psi, \phi\}$ randomly; memory $\mathcal{D}$ with 5 random episodes
2: for episode $k \leftarrow 1$ to $K$ do
3:     Online:
4:     for timestep $t \leftarrow 1$ to $T$ do
5:         Observe $o_t^1$ and predict embeddings $z_t^1, \tilde{z}_t^2 \sim q_\theta(z_t^1, \tilde{z}_t^2 | o_t^1)$
6:         Propagate states $s_t^1, \tilde{s}_t^2 \sim p_\theta\big(s_t^1, \tilde{s}_t^2 | s_{t-1}^1, \tilde{s}_{t-1}^2, a_{t-1}^1, \tilde{a}_{t-1}^2, z_t^1, \tilde{z}_t^2\big)$
7:         Generate action $a_t^1 \sim q_\phi(a_t^1 | s_t^1)$, predicted response $\tilde{a}_t^2 \sim q_\phi(\tilde{a}_t^2 | \tilde{s}_t^2)$
8:         Execute $a_t^1$ in the environment
9:     end for
10:    Offline:
11:    Add episode transitions $\{(o_t^1, o_t^2, a_t^1, a_t^2, r_t^1, r_t^2)\}_{t=1}^T$ to memory $\mathcal{D}$
12:    for trainstep $s \leftarrow 1$ to $S$ do
13:        Model update:
14:        Sample batch of sequences $\{(o_t^1, o_t^2, a_t^1, a_t^2, r_t^1, r_t^2)\}_{t=b}^{b+L} \sim \mathcal{D}$
15:        Use the encoder model to predict embeddings $z_t^1, \tilde{z}_t^1, z_t^2, \tilde{z}_t^2$
16:        Use the representation model to predict states $s_t^1, \tilde{s}_t^1, s_t^2, \tilde{s}_t^2$
17:        Update $\theta$ via representation learning on $\{(s_t^1, s_t^2), (s_t^1, \tilde{s}_t^2), (\tilde{s}_t^1, s_t^2)\}$
18:        Policy and Value update:
19:        Compute value estimates $V_\lambda^1(s_\tau^1), V_\lambda^2(s_\tau^2) \leftarrow \texttt{rollout}(s_t^1, s_t^2, H)$
20:        Update $\phi$ and $\psi$ based on Eq. (4.3) for all targets $\{V_\lambda^1(s_\tau^1), V_\lambda^2(s_\tau^2)\}$
21:    end for
22: end for
```

4.5 Implementation Details

In this section, we report the network architecture parameters and detail the models used in our approach.

4.5.1 Network Architectures

Table 4.2 reports all network parameters. The transition model is joint in our implementation and the associated input and output variables correspond to joint representations (e.g. the previous action $a_{\tau-1}$ (6) can be interpreted as $a_{\tau-1}$ (2×3)). We note that repeated layers have been condensed with Dense $\times$ i referring to application of the same dense layer architecture i times. The employed parameter abbreviations are referring to: a=activation, k=kernel, p=padding, s=stride.

Based on the general network architectures provided in Table 4.2, we comment on how the two parts of the transition model integrate with each other and provide further details on each of the models.

Transition Model

The joint transition model follows the recurrent state space model (RSSM) architecture presented in [77, 76]. The RSSM is extended to the multi-agent setting and propagates joint model states consisting of a deterministic and a stochastic component, respectively denoted by $s_{t,d}$ and $s_{t,s}$ at time t. The stochastic component $s_{t,s}$ is implemented via a diagonal Gaussian distribution and its derivation is provided in the next section. The transition model then predicts priors for the associated mean and standard deviation based on the previous model state and applied action (*imagine 1-step*). In the presence of observations, prior estimates can be updated to posterior estimates (*observe 1-step*). The transition model may then initialize its states by propagating posteriors based on a context sequence (*imagine 1-step* and *observe 1-step*) from which interactions can be imagined by propagating prior estimates (*imagine 1-step*).

Encoder Model

The encoder parameterization follows the architectural choices presented in [74], where we adapt the convolutional layers to match the dimensionality of our observations. The agent leverages two encoders in parallel, one for generating latent

Table 4.2: General network architectures of the underlying models.

Layer Type	Input (dimensions)	Output (dimensions)	Additional Parameters
Transition model (*imagine 1-step*)			
Dense	$s_{\tau-1,s}$ (60), $a_{\tau-1}$ (6)	$\text{fc}^1_{t,i}$ (600)	a=ELU
GRU	$\text{fc}^1_{t,i}$ (600), $s_{\tau-1,d}$ (400)	rs_τ (400), $s_{\tau,d}$ (400)	a=tanh
Dense	rs_τ (400)	$\text{fc}^2_{t,i}$ (600)	a=ELU
Dense	$\text{fc}^2_{t,i}$ (600)	$\mu^{prior}_{\tau,s}$ (60), $\sigma^{prior}_{\tau,s}$ (60)	a=None
Transition model (*observe 1-step*)			
Dense	$s_{\tau,d}$ (400), z_τ (2048)	$\text{fc}^1_{t,o}$ (600)	a=ELU
Dense	$\text{fc}^1_{t,o}$ (600)	$\mu^{post}_{\tau,s}$ (60), $\sigma^{post}_{\tau,s}$ (60)	a=None
Encoder model			
Conv2D	obs (96, 96, 3)	cv1 (31, 31, 32)	a=ReLU, s=3, k=(4,4)
Conv2D	cv1 (31, 31, 32)	cv2 (14, 14, 64)	a=ReLU, s=2, k=(4,4)
Conv2D	cv2 (14, 14, 64)	cv3 (6, 6, 128)	a=ReLU, s=2, k=(4,4)
Conv2D	cv3 (6, 6, 128)	cv4 (2, 2, 256)	a=ReLU, s=2, k=(4,4)
Observation model			
Dense	$s^i_{\tau,d}$ (200), $s^i_{\tau,s}$ (30)	fc^1_o (1, 1, 1024)	a=None
Deconv2D	fc^1_o (1, 1, 1024)	dc1 (5, 5, 128)	a=ReLU, s=2, k=(5,5)
Deconv2D	dc1 (5, 5, 128)	dc2 (13, 13, 64)	a=ReLU, s=2, k=(5,5)
Deconv2D	dc2 (13, 13, 64)	dc3 (31, 31, 32)	a=ReLU, s=2, k=(6,6), p=1
Deconv2D	dc3 (31, 31, 32)	dc4 (96, 96, 3)	a=ReLU, s=3, k=(6,6)
Reward model			
Dense	$s^i_{\tau,d}$ (200), $s^i_{\tau,s}$ (30)	fc^1_r (400)	a=ELU
Dense × 1	$\text{fc}^{\{1\}}_r$ (400)	$\text{fc}^{\{2\}}_r$ (400)	a=ELU
Dense	fc^2_r (400)	fc^3_r (1)	a=ELU
Value model			
Dense	$s^i_{\tau,d}$ (200), $s^i_{\tau,s}$ (30)	fc^1_v (400)	a=ELU
Dense × 2	$\text{fc}^{\{1,2\}}_v$ (400)	$\text{fc}^{\{2,3\}}_v$ (400)	a=ELU
Dense	fc^3_v (400)	fc^4_v (1)	a=ELU
Action model			
Dense	$s^i_{\tau,d}$ (200), $s^i_{\tau,s}$ (30)	fc^1_a (400)	a=ELU
Dense × 3	$\text{fc}^{\{1,2,3\}}_a$ (400)	$\text{fc}^{\{2,3,4\}}_a$ (400)	a=ELU
Dense	fc^4_a (400)	μ_a (3), σ_a (3)	a=ELU

vectors of the ego perspective and one for predicting latent vectors of the opponent perspective. Inputs are 96×96 RGB image observations.

Observation Model

The observation model follows the decoder architecture presented in [74], where we adapt the transposed convolutional layers to match the dimensionality of our observations. The image observations of agent i are reconstructed from the associated model states $s^i_{\tau,d}$ and $s^i_{\tau,s}$.

Reward and Value Model

Rewards and values of agent i are both predicted as scalar values from fully-connected networks that operate on the associated model states $s^i_{\tau,d}$ and $s^i_{\tau,s}$, similar to [76].

Action Model

The action model follows [76], where the predicted mean μ_a is rescaled and passed through a tanh function to allow for saturated action distributions. It is combined with a softplus standard deviation based on σ_a and the resulting Normal distribution is again squashed using a tanh [76, 75].

Order Independence

We note as an implementation detail that the transition distribution $q_\theta\left(s^1_t, s^2_t | s^1_{t-1}, s^2_{t-1}, a^1_{t-1}, a^2_{t-1}\right)$ should be independent of the order in which agents are provided. Thus, reversing the ordering and considering $q_\theta\left(s^2_t, s^1_t | s^2_{t-1}, s^1_{t-1}, a^2_{t-1}, a^1_{t-1}\right)$ should yield the same distribution. We achieve this by two passes through the learned transition model and subsequent averaging. In the second forward pass the agents' input order to the transition model is flipped.

Training Parameters

Most of the training parameters correspond to the implementation of [76], a state-of-the-art model-based RL algorithm for learning to plan in latent-space from image observations. DLC trains every 1000 environment steps for 200 iterations with the Adam optimizer [97]. The batch size is set to 50. The representation, value and actor model are respectively trained with learning rates 6e-4, 6e-4, and 8e-5. Gradients over the magnitude of 100 are clipped for all models. The prior $\sigma_{\tau,s}^{prior}$ and posterior $\sigma_{\tau,s}^{post}$ variance in the transition model are bounded from below to a minimum value of 0.1. The model loss on true observations J_{M,s_t^1,s_t^2} is weighted twice as much as the model losses on predicted opponent observations $J_{M,s_t^1,\tilde{s}_t^2}$ and $J_{M,\tilde{s}_t^1,s_t^2}$. Throughout, we use $\gamma = 0.99$ and $\lambda = 0.95$. The model learning horizon is $L = 50$ whereas the imagination horizon is $H = 15$. Value and action models are trained on the same trajectory rollouts.

Environment Details

To ensure generalization, we randomized both the color and initial position of all vehicles throughout training. We also evaluated (but did not use) penalization for driving in the backward direction which can occur after returning from a spin on the grass.

Encoder

Our representation model is a latent variable model. That is, it can be written as

$$\begin{aligned} p_\theta(S_t|S_{t-1}, A_{t-1}, O_t) &= p_\theta\left(s_t^1, s_t^2|s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, o_t^1, o_t^2\right) \\ &= \int\int p_\theta\left(s_t^1, s_t^2|s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, z_t^1, z_t^2\right) \qquad (4.5) \\ &\qquad \cdot p(z_t^1|o_t^1) \ \cdot p(z_t^2|o_t^2)\ \mathrm{d}z_t^1\ \mathrm{d}z_t^2\ , \end{aligned}$$

where $S_t := \{s_t^1,\ s_t^2\}$ is the concatenation of both states and A_{t-1}, O_{t-1} are defined analogously. Each agent i maintains its own belief of the world, without having access to both observations. Thus, at test time it computes instead

$$p_\theta(S_t|S_{t-1}, A_{t-1}, o_t^i) = \\ \int\int p_\theta\left(s_t^1, s_t^2|s_{t-1}^1, s_{t-1}^2, a_{t-1}^1, a_{t-1}^2, z_t^i, \tilde{z}_t^{\neg i}\right) \cdot q_\theta(z_t^i, \tilde{z}_t^{\neg i}|o_t^i)\, \mathrm{d}z_t^i\, \mathrm{d}\tilde{z}_t^{\neg i}\ . \quad (4.6)$$

Therefore, we never need to learn $p(z_t^i|o_t^i)$. Instead we directly learn the representation $q_\theta(z_t^i, \tilde{z}_t^{\neg i}|o_t^i)$. This is implemented as a convolutional encoder $f_E(o^i)$ making $q_\theta(z_t^i, \tilde{z}_t^{\neg i}|o_t^i)$ a Dirac distribution

$$q_\theta(z_t^i, \tilde{z}_t^{\neg i}|o_t^i) = \delta\left((z_t^i, \tilde{z}_t^{\neg i}) - f_E(o_t^i)\right)\ . \quad (4.7)$$

4.6 Latent Racing Experiments

We demonstrate the ability of DLC to learn competitive visual control policies in a novel multi-agent racing environment, and compare performance against baselines to highlight the importance of both the joint transition model and the learned observer. We further highlight the learned representation model's capability of predicting opponent viewpoints from ego observations.

4.6.1 Racing Environment

We propose `MultiCarRacing-v0`, a novel multi-agent racing environment for learning competitive visual control policies[1]. The environment extends the Gym task `CarRacing-v0` [37] and provides each agent with top-down 96x96 pixel image observations from their ego perspective based on which continuous control inputs need to be selected. The viewpoint is motivated by recent results in the context of driving [55, 46, 124] and holds the promise of future deployment on physical platforms. The environment allows for differentiating between skillful and competitive driving. While the former is the basis for high-performance racing, learning to beat a skillful opponent is a far greater challenge. Interactions between agents are sparse but

[1]Code available at `https://github.com/igilitschenski/multi_car_racing`

information-rich: only when agents collide, push, or block each other do they directly impact each others state.

4.6.2 Dynamics and Rewards

The vehicles in the environment exhibit slip and collision dynamics. Breaking or accelerating too hard induces skidding and in combination with steering causes substantial understeering or oversteering (drifting). Similarly, moving off the track lowers available friction. Collision dynamics allow for elaborate interaction strategies during the race: pushing other agents off the track, blocking overtaking attempts, or turning an opponent sideways via a PIT maneuver. We show examples in Figures 4-1, 4-5, and 4-6. The reward mapping follows `CarRacing-v0`, in that each agent incurs a loss of -0.1 per timestep and receives a reward for each visited tile along the track. We incentivize competition by discounting rewards based on visitation order: the first agent to visit a track tile is rewarded with $+1000/N$, while the second agent receives $+500/N$ (on an N-tile track).

4.6.3 Benchmarking Performance

The algorithm presented in this work, DLC, learns a multi-agent world model that enables imagined self-play by combining the underlying transition function with an observer capable of predicting opponent latent states. We refer to this method as *joint transition + observer* and compare performance against two baselines. The first is *joint transition*, which propagates ground truth latent states of both agents and allows for assessing performance of the observer. The second is *individual transition*, which propagates ground truth latent states individually and highlights the added value of a joint transition model. Each method is trained on 500 races. We evaluate the resulting performance in a round-robin tournament, a competition in which each contestant meets all other contestants in turn. The round-robin tournament consists of 100 races for each pairing (300 races per tournament) at multiple stages of training progress. We repeat this for 5 random seeds.

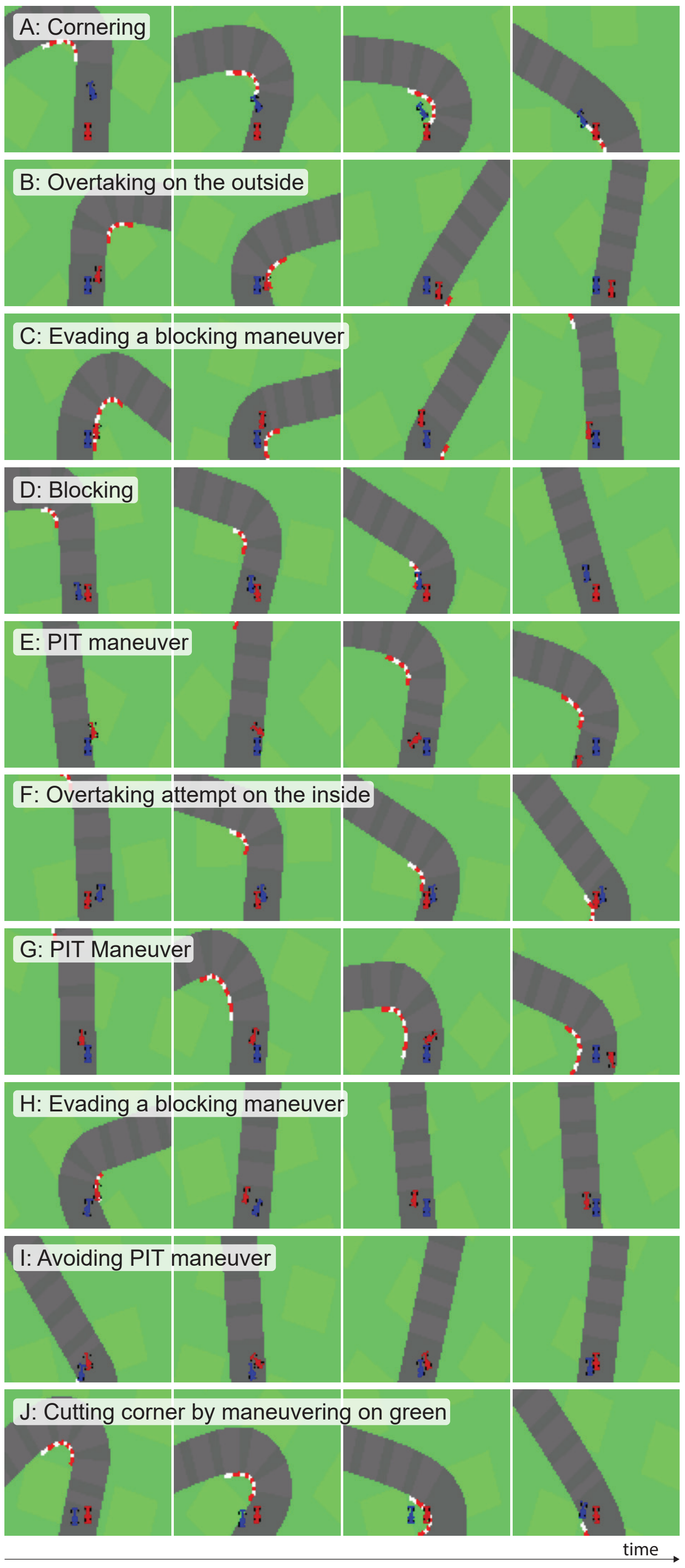

A: Cornering
B: Overtaking on the outside
C: Evading a blocking maneuver
D: Blocking
E: PIT maneuver
F: Overtaking attempt on the inside
G: PIT Maneuver
H: Evading a blocking maneuver
I: Avoiding PIT maneuver
J: Cutting corner by maneuvering on green
time

Figure 4-5: **Learned racing skills:** The agent has learned to leverage a large variety of racing skills towards their competitive advantage. The skills include cornering, blocking, overtaking, and forcing others off the road.

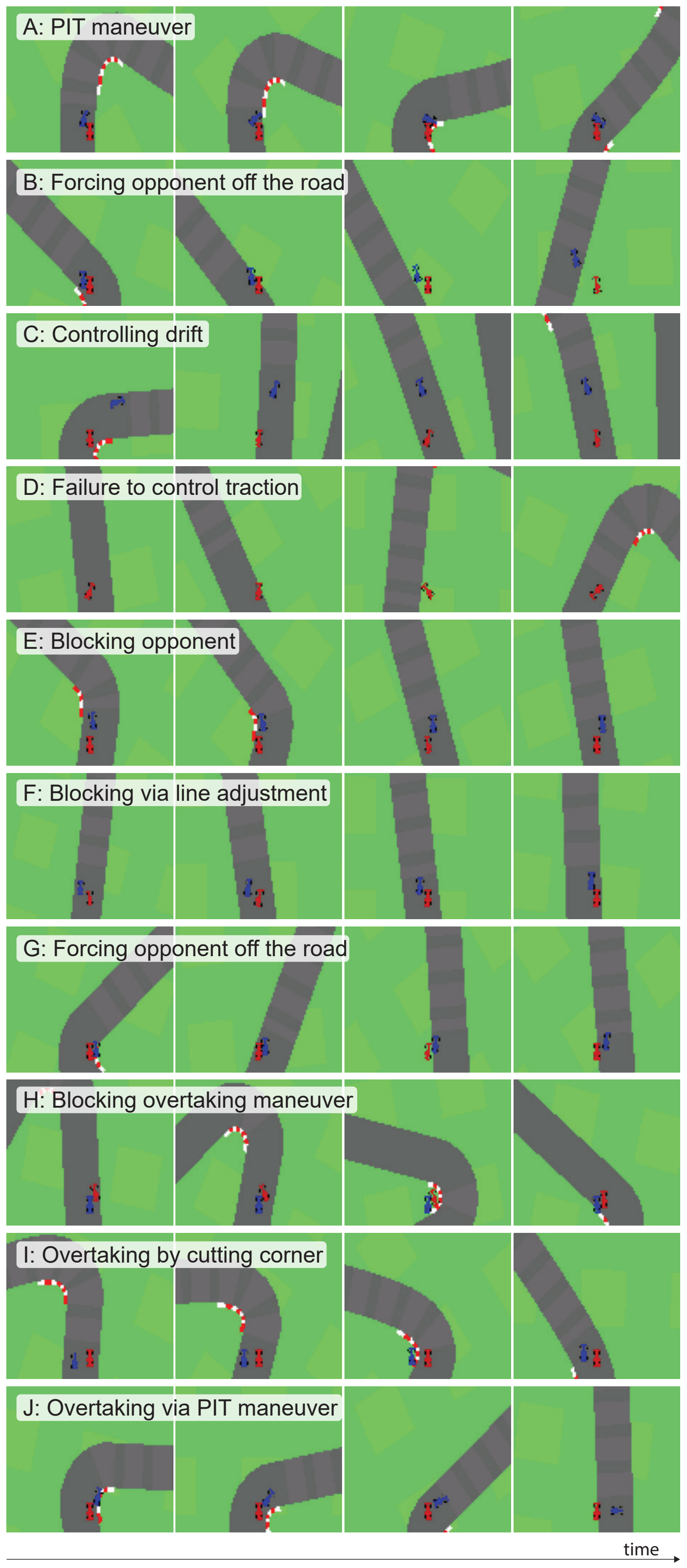

A: PIT maneuver
B: Forcing opponent off the road
C: Controlling drift
D: Failure to control traction
E: Blocking opponent
F: Blocking via line adjustment
G: Forcing opponent off the road
H: Blocking overtaking maneuver
I: Overtaking by cutting corner
J: Overtaking via PIT maneuver
time

Figure 4-6: **Common racing scenarios:** During a race, agents have to drive at the dynamical limits of handling to move as fast as possible along the track. While they don't receive penalties for leaving the road, moving onto the slippery grass increases the risk of spinning out.

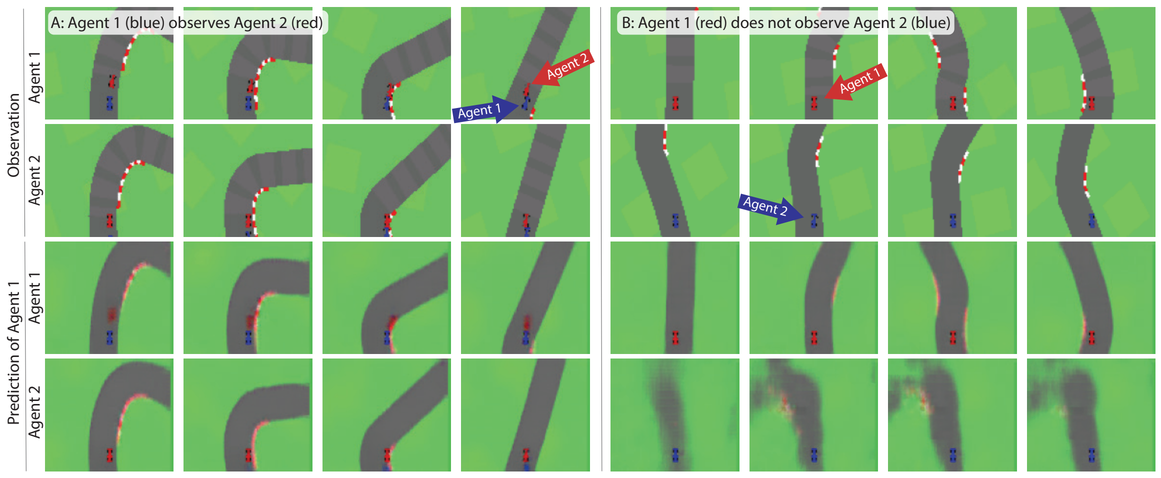

A: Agent 1 (blue) observes Agent 2 (red)
B: Agent 1 (red) does not observe Agent 2 (blue)
Observation
Agent 1
Agent 2
Prediction of Agent 1
Agent 1
Agent 2
Agent 1
Agent 2
Agent 1
Agent 2

Figure 4-7: **Closed loop prediction:** **A**: Agent 1 observes the opponent and is therefore able to reconstruct both views. **B**: 1 is unable to observe 2 and reconstructs their view with high uncertainty.

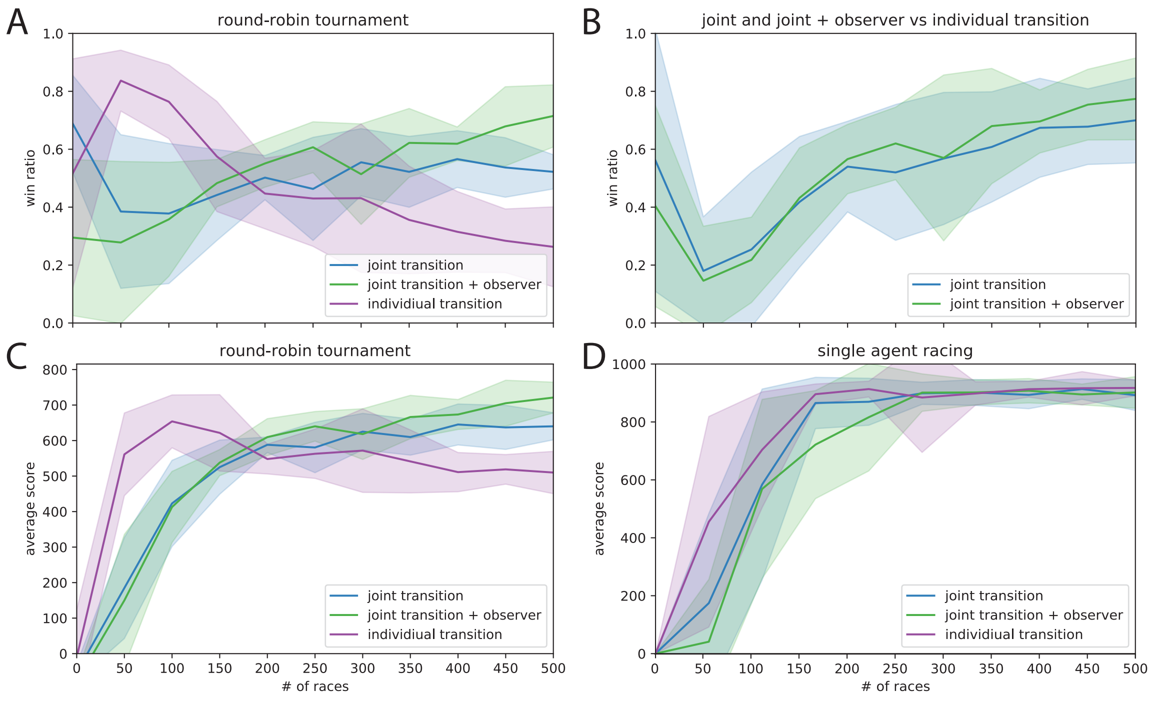


Figure 4-8: **Comparison of racing performance and competitiveness:** **A**: Win ratio of all agents in a round-robin tournament. **B**: Agents with joint and joint + observer transition compete directly against an agent with individual transition function. **C**: Average score of all agents in a round-robin tournament. **D**: Single agent racing performance disentangles general skill learning from learning to compete through interaction.

Figure 4-8A shows the resulting win-ratio and Figure 4-8B the average score for the round-robin tournament. The representation learning problem for *individual transition* is easier as the learned transition function does not need to disentangle how the states and actions of both agents affect the transition. Therefore, the *individual transition* baseline starts off strong. However, after 200 races both joint transition methods have learned to compete more effectively through imagined self-play. The *individual transition* baseline can not leverage this effect, as agent actions do not affect opponents states during imagined rollouts and self-play may only occur in the real world. As visible in Figure 4-8C, after 500 races both joint transition approaches win 70-80% of races against the *individual transition* baseline. Furthermore, both joint transition methods perform similarly, suggesting that the learned observer is capable of predicting opponent latent states in a way that is sufficient in order to induce learning of competitive behaviors.

The single agent racing performance, see Figure 4-8D, further allows us to disentangle general skill learning from learning to compete. It confirms that the *individual transition* baseline is able to acquire general racing skills faster. Similarly, the joint transition methods do not outperform the *individual transition* baseline in the single agent racing case, such that their performance increase for the multi-agent setting in Figure 4-8A can be explained by their increased ability to compete.

4.6.4 Predicting the Opponent's Latent State from Ego Observations

Given an agent's own history of observations and controls, we can estimate the compact latent state and it's associated reconstruction to visualize the representation model's understanding of the world. More interestingly, we can create a reconstruction of the predicted opponent's latent state to visualize how well the agent can infer the opponent view. Given a predicted opponent state we can predict their actions based on the learned policy. Figure 4-7 provides two scenarios with reconstructions based only on observations of agent 1.

In Figure 4-7A, the opponent is within the field of view of agent 1. The reconstructions are focused and close to the ground truth for both the ego and opponent viewpoints, indicating that the latent state is accurate and well understood. In Figure 4-7B, agent 1 never observes its opponent and is unable to accurately predict their view. However, instead of predicting only noise, agent 2 is imagined to drive on a straight road with the possibility of an upcoming turn. This is a sensible prediction given that agent 2 is currently not observable and in turn will not immediately affect the motion plan of agent 1.

4.6.5 Imagining Interaction Sequences

To facilitate efficient self-play in a learned world model, we require all agent states to be jointly propagated in a consistent manner. In Figure 4-9, we provide 5 observations of agent 1 as context and investigate the model's ability to predict forward in time for 25 additional timesteps. Similarly to Figure 4-7, we exclude ground-truth observations of agent 2. While actions of agent 1 are available for the full horizon, the learned policy predicts the actions of agent 2.

Referring to the context frames, we observe the benefit of recursively estimating states in agent 1's reconstruction of agent 2's viewpoint: while the reconstruction has high uncertainty in the first frame, accuracy improves with every new observation. Agent 1's reconstruction of their ego view is detailed from the start with the blue agent remaining blurry. Moving beyond the context frames, the predicted views quickly diverge from the ground truth. This is expected as the agent is unable to see beyond the first turn. In the following imagination, the blue agent overtakes the red agent by forcing them off the track (Figure 4-9 prediction of agent 1, timestep 15). This is reasonable, as the blue agent leaves the left turn on the inside with an opportunity to push the red agent off the road. The imagined sequences feature detailed predictions of the track, including markings on an upcoming left turn (timestep 25).

Most importantly, the predictions of both agents remain consistent. The two agents' relative positions with each other and the track are in correspondence along the full horizon. Likewise, the predicted characteristics of a track including curbs

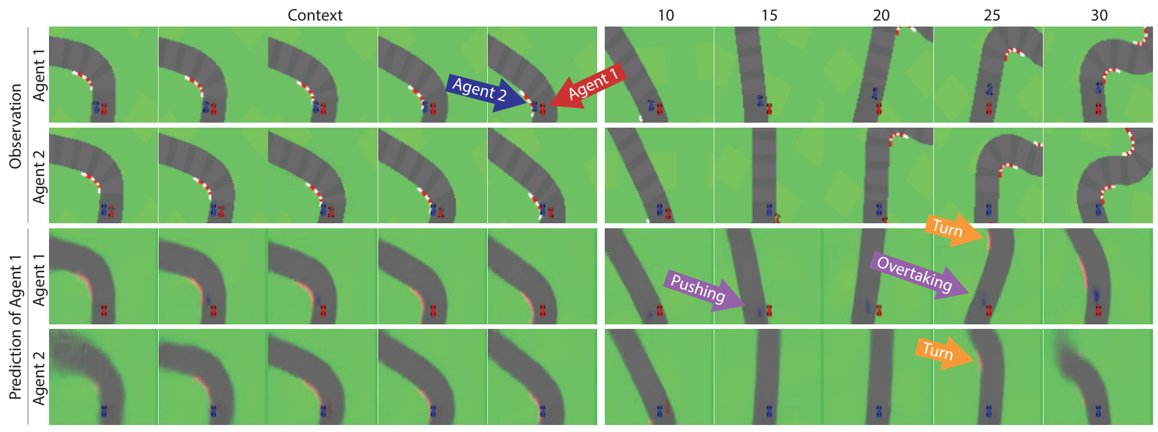


Figure 4-9: **Open loop prediction:** Agent 1 (red) receives 5 observation frames as context and predicts the own view and the view of agent 2 (blue). We employ the learned dynamics to forward propagate 25 additional timesteps into the future without any further observations.

and a left turn are consistent between both agents' predicted views (timestep 25). Consistency is crucial as it allows to learn from imagined self-play in a world model. If the two predictions were to diverge, the impact of the ego actions on another agent's behavior could not be estimated. The outcome of the imagined games would be uncertain. The joint transition model helps keeping predictions consistent, as it allows for information exchange between both agents' latent states during forward propagation.

4.7 Discussion

We present Deep Latent Competition (DLC), a novel reinforcement learning algorithm that learns competitive visual control policies through self-play in imagination. The DLC agent can imagine interaction sequences in the compact latent space based on a multi-agent world model that combines a joint transition function with opponent viewpoint prediction. The behavior is then optimized by back-propagating the analytic value gradients of these imagined game trajectories through the learned world model. Experiments in a novel continuous visual control racing environment demonstrate that the DLC agent learns to make consistent multi-agent forward predictions.

Optimizing competitive behaviors through imagined self-play based on these joint predictions yields an agent that performs superior to an agent that propagates ground truth observations separately. In the future, we aim to deploy the DLC agent on hardware platforms to yield competitive racecar driving in the real world. Extending the framework to include more complex game-theoretic considerations in the forward predictions offers another intriguing avenue for future work.

Chapters 2-4 investigated interactions between agents as individual entities. In the next chapter we will present a guardian angel system that interacts with a human driver by sharing control. Our goal is to gradually bring autonomy capabilities to practice, by using a guardian system that aims to correct the driver's mistakes.

Chapter 5

Guardian Angel: Parallel Autonomy in Automated Vehicles

5.1 Introduction

Globally, over 3,000 human lives are lost *every day* [22] in vehicle-related accidents and over one hundred thousand are injured or disabled on average. Worse still is that we expect this number to continue to increase [136]. In the United States, 11% of accidents are caused by driver distraction (such as cell phone use), 31% involve an impaired driver due to alcohol consumption, 28% involve speeding, and an additional 2.6% are due to fatigue [138]. This troubling trend has resulted in the continued development of advanced safety systems by commercial car manufacturers and suppliers.

For example, systems exist to automatically brake in the case of unexpected obstacles [140], to maintain a car inside a lane at a given speed, and alert users of pedestrians, signage, and other vehicles on the roadway [51]. However, the scenarios that these systems can deal with are relatively simple compared to the diverse and complicated situations that we routinely find ourselves in as human drivers. Human drivers are capable of handling nearly all driving tasks well enough most of the time but are overwhelmed in key moments when quick and precise actions are needed in fast and complex traffic scenarios. A deer crossing the road, a preceding crash on

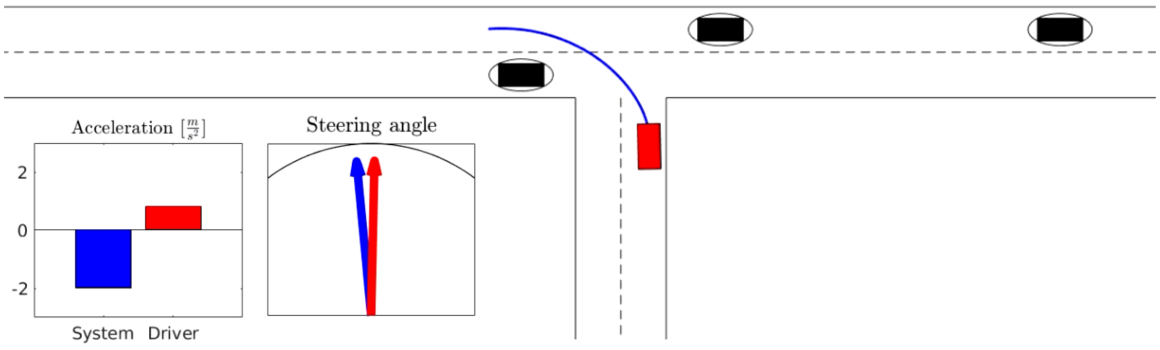


Figure 5-1: **Parallel Autonomy in complex driving scenarios:** A human driver (red) tries to accelerate into an intersection, as shown by the red acceleration bar in the lower left inset. Since this action would result in a collision, the Parallel Autonomy system prevents the vehicle from continuing. The system brakes, indicated by the blue bar and deviates from the driver's commanded controls to ensure safety.

the highway, a missed blind spot, or driver tiredness are only some of the ubiquitous challenges human drivers face in everyday traffic. To handle these situations, an advanced autonomy system must ensure safety when required while a human driver is in control of the car.

In this work we propose a framework for advanced safety in complex scenarios that we refer to as Parallel Autonomy. In this framework we minimize the deviation from the human input while ensuring safety. The design of the system has two main objectives: (a) minimal intervention - we only apply autonomous control when necessary, and (b) guaranteed safety - the collision free state of the vehicle is explicitly enforced through constraints in the optimization.

There are three types of collaborative autonomy:

1. **Series Autonomy**, in which the human driver orders the vehicle to execute a function, similar to most self-driving approaches to date. Currently, we do not yet have systems that work reliably under all driving conditions or even for highway driving. While this direction is promising to save lives in the future, it is not applicable at this point.

2. **Interleaved Autonomy**, in which the human driver and the autonomous system intermittently take turns in operating the vehicle. The human driver will

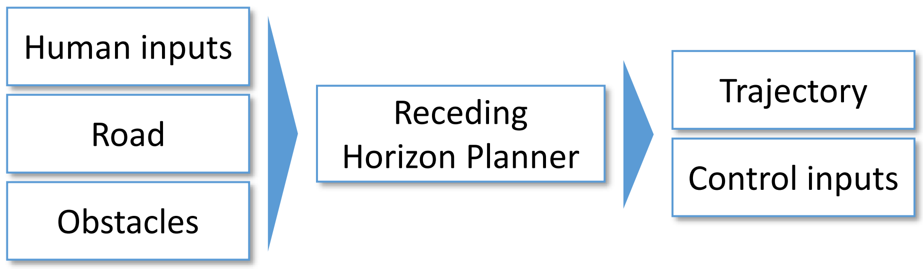


Figure 5-2: **System schematic:** A receding horizon planner computes control inputs based on human inputs and an environment consisting of the road, and static and dynamic obstacles.

take over control of the vehicle if the autonomy system fails or requests intervention. This may result in the handover problem: If the system fails abruptly, the human driver may not be ready to immediately resume control. Since failures may only occur in rare cases, people may become overconfident in the system, lose attentiveness, use their phone [160], sleep [81], or conduct other activities. Systems would have to be able to predict their own failure way ahead into the future to give people enough time to be cognizant of their environment and safely respond.

3. **Parallel Autonomy**, in which the autonomous system functions as a guardian angel in the background to ensure safety while the human driver is operating the vehicle. Ideally, the system will follow the do no harm principle and will only intervene when it is certain to improve safety. The benefit of this approach is that autonomous driving technology can be rolled out incrementally without the strict requirement to work perfectly in all driving conditions. Therefore, this approach can save lives already at this point in time.

Whether drivers are distracted by smartphones, searching in their glove box, operating the navigation system, or are simply overwhelmed by the difficulty of driving in challenging scenarios, the Parallel Autonomy principles offer additional safety due to redundancy.

5.1.1 Main Assumptions

In this work, we assume that a perception system provides the current state of the ego vehicle and the road boundaries, see Figure 5-2. Additionally, we assume that a prediction system produces uncertain future predictions of other vehicles, which we will integrate into our planning framework. Similar neural network prediction models that provide both expectations and estimated uncertainties [12] have become available in recent years. In this work, we assume that we can parametrize the posterior distributions, describing the uncertain current and future states of other vehicles, in the form of Gaussians. In contrast to Chapters 2 and 3, we neglect the influence of our actions on other agents and assume that their predictions remain independent. We model the vehicle through kinematic and dynamic models. Their parameters have to be estimated to ensure the models to remain accurate. In comparison to the stochastic dynamics in Chapter 3, we model these dynamics as deterministic and assume that their impact is negligible to the uncertainty in the predictions of the other drivers. We employ local optimization that makes choosing a suitable initial guess necessary to ensure good performance. While we can compute these through search-based or sampling-based methods, simpler heuristics such as an all-zero initial guess proved sufficient.

5.1.2 Contributions

We provide a formulation and algorithmic solution to Parallel Autonomy based on a Nonlinear Model Predictive Control (NMPC) policy. Specifically, in this chapter we:

- incorporate the time-varying uncertainty of the dynamic obstacle predictions as analytic chance constraints in the optimization;
- introduce a contour tracking approach with additional constraints for the road boundaries to ensure that the vehicle safely follows the road;
- minimize the deviation of the system's steering and acceleration control and the control commanded by the driver subject to not compromising safety; and

- plan online over long time horizons ($\sim 9s$).

We show the basic operation of the controller in Figure 5-1, where the driver attempts to cut into of oncoming traffic to make a left turn. The Parallel Autonomy system prevents a collision with the approaching vehicles by intervening.

This chapter contributes the following:

1. A formulation of Parallel Autonomy as a shared control approach between human drivers and intelligent vehicles, which adheres to the minimal intervention principle and can handle complex driving scenarios.
2. A NMPC formulation for real-time trajectory generation in intelligent and autonomous vehicles, suitable for state of the art solvers.[1]
3. Extensive evaluations in complex traffic scenarios with real human inputs of different driving styles, such as left turns across traffic and driving on a snowy race track subject to slip.

In the following, we will employ two motion models of different complexity:

1. A dynamical nonlinear combined slip vehicle model including load transfer for static environments.
2. A kinematic model for dynamic and complex. environments

We organize the chapter as follows: In Section 5.2 we summarize the related work in the field. Section 5.3 presents the Parallel Autonomy control approach. In Section 5.4 we provide a concrete instantiation of the framework and present the NMPC approach to solving it. Finally, we show detailed simulation results in Section 5.5 and conclusions in Section 5.6.

5.2 Related Works

In this section we provide an overview of the related work in the areas of general safety, shared control for autonomous vehicles, and Model Predictive Control (MPC).

[1]We employ FORCES Pro by Embotech to generate a fast NMPC solver for our problem formulation.

5.2.1 Safety of Autonomous Vehicles

In theory, safety can be guaranteed for deterministic systems by computing the set of the states for which the vehicle will *inevitably* have a collision and then ensuring that the vehicle never enters that set. The set is referred to by different terms in the literature, such as the capture set [63, 7, 35, 87], the inevitable collision states (ICS) [64, 31, 11], the region of inevitable collision (RIC) [45], and the target set [127]. However, without some assumptions or limiting the applicability to relatively simplistic scenarios, this set is difficult to compute analytically. [47] apply a control barrier function to guarantee never entering the infeasible set upon moving into an avoidable set constructed from a polar algorithm in slow speeds to avoid pedestrians. These ICS-inspired methods tend to only intervene when the system is at the boundary of the capture set, which can cause undesirable behavior, toggling between either the autonomous system input or the human input. We follow the idea of [63, 31, 11] and define a set of probabilistic constraints for collision avoidance to produce safe behavior. Yet, our method provides smooth inputs to the vehicle since the Parallel Autonomy safety system smoothly minimizes deviation from user inputs in an optimization framework subject to safety constraints.

5.2.2 Shared Control of Intelligent Vehicles

The most intuitive way of merging the human input with the output of a safety system is by linear combination of the two, possibly by threat measures based on the dynamic limitations of the vehicle [17, 19]. The system could simply override the human input depending on the severity of the threat. While systems may reason about following the human driver's intended homotopy class [17], these presented systems do not take the human input directly into account but rather interpolate between human driver and system controls. In a similar line of research, safety margins can be computed from sampled trajectories clustered into homotopy classes [50]. These approaches do not directly share the control with the human driver.

In contrast, in this work we directly incorporate the human inputs into an optimization framework in a minimally invasive manner and also add a soft *nudging* behavior to guide the driver. Our approach minimizes the deviation of the autonomous system's plan from the driver's intent - current steering and acceleration inputs - and provides small feedback to the driver already shortly before situations are predicted to become critical. This is important to prevent startling the driver and to decrease the likelihood of unnecessary strong intervention, thanks to slight intervention early on.

Similar to our method, several constrained optimization approaches for shared control exist. While some approaches directly minimize the difference of the human predicted control input from the necessary control input to produce safe trajectories [179], others minimize the difference in steering wheel angle [68]. [10] minimized the deviation from human inputs, in this case, orientation and speed, via a convex constrained optimization to generate safe motion. Beyond direct control inputs, other approaches minimize the deviation from the currently desired front wheel lateral force with an additional discount factor decreasing the impact over time [59]. We apply a discount factor in a similar manner to focus on the deviation from human inputs in the short term but give the optimization objective more freedom in the far future. We present a general approach where we jointly minimize deviation from both steering and acceleration inputs, blend in additional trajectory-specific costs to provide a nudging behavior, model the slip-dynamics of the vehicle, and strictly enforce safety constraints.

5.2.3 Receding Horizon Control for Shared Control

We formulate the constrained optimization as a receding horizon control problem, typically referred to as Model Predictive Control (MPC). Related to our work, [68] employed a hierarchical MPC approach for avoidance of static obstacles with motion primitives and path tracking, which switches control to and from the driver as a function of driver attentiveness. Instead of planning paths first and computing velocity profiles separately, we directly plan full trajectories that also avoid dynamic obstacles. Similar approaches include: a constrained pathless MPC that blends human and controller steering commands only, proposed by [17], a shared control MPC for static unstructured robot environments avoiding circular obstacles [187], and robust NMPC [68] to avoid static obstacles while tracking the roads center line over a very short horizon of less than $1.5s$. Alternatively, [59] defined vehicle-stability and environmental envelopes to supply safe steering commands at constant speed in a discretized environment.

In contrast, our approach does not require manual linearization - we solve a Nonlinear MPC (NMPC) problem directly - and can handle complex road scenarios with dynamic maneuvers and dynamic obstacles, with steering *and* acceleration control over long horizons ($\sim 9s$).

Some methods [18, 59] construct corridors consisting of multiple convex regions to describe the area that the vehicle will drive through. Based on the corridors and constant velocity, they apply constraints on the lateral position of the vehicle to avoid colliding with obstacles yielding a convex optimization problem with global optimality. In contrast, operating over both steering and acceleration in a non-convex environment and solving our NMPC formulation inherits the general limitations of non-convex optimization, such as uncertain convergence and runtime, and lack of guarantee of optimality. Nonetheless, combined steering and braking is an essential function in vehicle safety.

We provide all costs and constraints to the solver in closed form without pre-linearization and benefit from recent advances in efficient Interior-Point solvers [53] to directly solve the NMPC. To guide the planner along the road, we build upon Model Predictive Contouring Control (MPCC) [60, 103, 118], which approximates path progress inside a corridor, the road in our application. By tracking the center of the lane and remaining within the limits of the road our planner can be employed for both Parallel Autonomy, where we minimize the deviation from human input, and for fully autonomous vehicles.

5.3 Problem Formulation

The Parallel Autonomy problem is based on two overarching principles:

- *Minimal intervention* with respect to the human driver. That is, the control inputs to the vehicle should be as close as possible to those of the human driver.
- *Safety.* The probability of collision with respect to the environment and other traffic participants is below a given threshold.

5.3.1 Definitions

We use the discrete time shorthand $k \triangleq t_k$, where $t_k = t_0 + \sum_{i=1}^{k} \Delta t_i$, with t_0 the current time and Δt_i the i-th timestep of the planner. Vectors are bold.

Ego Vehicle

We refer to the car operated by the human driver as ego vehicle. At time k, we denote the configuration of the ego vehicle, typically position $\mathbf{p}_k = [x_k, y_k]$, heading ϕ_k, longitudinal and lateral velocity $v_{x,k}$, $v_{y,k}$, yaw rate $\dot{\phi}_k$ and steering angle δ_k, by the state $\mathbf{x}_k = [\mathbf{p}_k, \phi_k, \delta_k, v_{x,k}, v_{y,k}] \in \mathcal{X}$. Its control input, typically steering velocity $\dot{\delta}_k$ and longitudinal acceleration $\dot{v}_{x,k}$, is labeled $\mathbf{u}_k = [\dot{\delta}_k^u, \dot{v}_{x,k}^u] \in \mathcal{U}$.

The evolution of the state of a vehicle is then represented by a general discrete dynamical system

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{5.1}$$

described in Section 5.3.3. The arguably largest source of uncertainty in the system operating over long time horizons lies in the prediction of motions of other vehicles, which dominates all other sources of uncertainty. The comparably low uncertainty in the dynamic motions of the ego vehicle thus motivates the choice for a deterministic motion model. We denote the area occupied by the ego vehicle at state $\mathbf{x}_k$ by $\mathcal{B}(\mathbf{x}_k) \subset \mathbb{R}^2$. In particular, we model it as a union of circles as shown in Section 5.4.5, Figure 5-5.

Other Traffic Participants

Apart from the ego vehicle, other traffic participants, such as vehicles, pedestrians and bikes, are indexed by $i = \{1, \ldots, n\}$. We denote their states as $\mathbf{x}_k^i \in \mathcal{X}_i$. To incorporate uncertainty, we assume that an approximate posterior distribution describing the current and future state of the other vehicles for up to m timesteps is available, e.g. from an inference framework [80]. The distributions are parametrized by their expected mean configuration $[\mathbf{p}_{0:m}^i, \phi_{0:m}^i]$ and positional covariance $\mathbf{\Sigma}_{0:m}^i$.

At a given state, each traffic participant occupies an area $\mathcal{B}^i(\mathbf{x}_k^i, \mathbf{\Sigma}_k^i, p_\epsilon) \subset \mathbb{R}^2$ with probability larger than p_ϵ. Here p_ϵ is the accepted probability of collision. We model them as ellipses that grow in size with uncertainty, as described in the forthcoming Section 5.4.5. This results in an approximate closed-form description of a probabilistic collision constraint and enables computational tractability.

Free Space

We consider the workspace $\mathcal{W} = \mathbb{R}^2$ and an obstacle map $\mathcal{O} \subset \mathcal{W}$ containing the static obstacles, such as the limits of the road and areas not classified as road. We define the environment $\mathcal{E}(k)$ as the state of the world (obstacles, traffic participants) at a time instance k.

5.3.2 Parallel Autonomy Formulation

We formulate a general discrete time constrained optimization with m timesteps and time horizon $\tau = \sum_{k=1}^{m} \Delta t_k$. We use the following notation for a set of states $\mathbf{x}_{0:m} = [\mathbf{x}_0, \ldots, \mathbf{x}_m] \in \mathcal{X}^{m+1}$ and for a set of inputs $\mathbf{u}_{0:m-1} = [\mathbf{u}_0, \ldots, \mathbf{u}_{m-1}] \in \mathcal{U}^m$.

The objective is to compute the optimal inputs $\mathbf{u}_{0:m-1}^*$ for the ego vehicle which minimize a cost function $\hat{J}_h(\mathbf{u}_{0:m-1}, \mathbf{u}_0^h) + \hat{J}_t(\mathbf{x}_{0:m}, \mathbf{u}_{0:m-1})$, where

- $\hat{J}_h(\mathbf{u}_{0:m-1}, \mathbf{u}_0^h)$ is a cost term that minimizes the deviation of the planned controls from the currently observed human input $\mathbf{u}_0^h$.
- $\hat{J}_t(\mathbf{x}_{0:m}, \mathbf{u}_{0:m-1})$ is a cost term that only depends on intrinsic properties of the planned trajectory. It can include various optimization objectives such as energy

minimization, comfort, or following a lane.

The optimization is subject to a set of constraints, which represent

- the transition model of the ego vehicle, $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$,
- no collisions with static obstacles, $\mathcal{B}(\mathbf{x}_k) \cap \mathcal{O} = \emptyset$, and
- no collisions with other traffic participants up to probability p_ϵ described by $\mathcal{B}(\mathbf{x}_k) \cap \bigcup_{i \in \{1,\ldots,n\}} \mathcal{B}^i(\mathbf{x}^i_k, \mathbf{\Sigma}^i_k, p_\epsilon) = \emptyset$.

Given the estimated trajectories $(\mathbf{x}^i_{0:m}, \mathbf{\Sigma}^i_{0:m})$ for all traffic participants $i = 1, \ldots, n$ and the initial state $\mathbf{x}_0$ of the ego vehicle, the optimal trajectory for the ego vehicle is then given by the following receding-horizon optimization,

$$\begin{aligned}
\mathbf{u}^*_{0:m-1} = \underset{\mathbf{u}_{0:m-1}}{\arg\min}\ & \hat{J}_h(\mathbf{u}_{0:m-1}, \mathbf{u}^h_0) + \hat{J}_t(\mathbf{x}_{0:m}, \mathbf{u}_{0:m-1}) \\
\text{s.t.}\ & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \\
& \mathcal{B}(\mathbf{x}_k) \cap \mathcal{O} = \emptyset \\
& \mathcal{B}(\mathbf{x}_k) \cap \bigcup_{i \in \{1,\ldots,n\}} \mathcal{B}^i(\mathbf{x}^i_k, \mathbf{\Sigma}^i_k, p_\epsilon) = \emptyset, \\
& \qquad\qquad \forall k \in \{0, \ldots, m\}.
\end{aligned} \tag{5.2}$$

We describe the method in detail in Section 5.4.

5.3.3 Motion Models

Previous approaches utilized constant longitudinal speed and small angle assumptions in selected static obstacle avoidance scenarios along straight roads [17, 19, 59]. In contrast, we consider the impact of joint speed and steering control for higher safety in dynamic, more general and more complex traffic environments over longer time horizons.

We first introduce a kinematic motion model with constraints to ensure limited slip for complex and dynamic environments in dense traffic, and subsequently present

Table 5.1: Parallel Autonomy: Main Symbols and Notation

$\mathbf{x}, \mathbf{u}$	Vehicle state and control input
$\mathbf{u}^*$	optimal control input
$\tau = \sum_{k=1}^{m} \Delta t_k$	time horizon
$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$	state transition function
$\mathbf{x}_{0:m}^i$, $\mathbf{\Sigma}_{0:m}^i$	state and covariance trajectory of agent i
$\mathbf{u}_0^h$	current human control input
J_h, J_t	Minimal intervention, and trajectory cost
J_{MPCC}	Model predictive contouring cost
$\mathcal{B}^i(\mathbf{x}_k^i, \Sigma_k^i, p_\epsilon)$	Probabilistic footprint of dynamic obstacles
$\mathcal{B}(\mathbf{x}_k)$	Ego vehicle footprint
$\mathcal{O}, \mathcal{E}, \mathcal{W}$	Static obstacles, environment, workspace
v_x, v_y	longitudinal and lateral speed
δ, ϕ	steering wheel angle, yaw angle
L	vehicle length
l_r, l_f	distance to cog from front and rear axle
s_{fx}, s_{fy}	longitudinal and lateral slip on front tire
F_{fx}, F_{fy}	longitudinal and lateral front tire forces
m, I_z	vehicle mass and moment of inertia
$\mu, B_\alpha, C_\alpha, D_\alpha$	Pacejka tire friction coefficients
$\gamma(s)$	longitudinal slip distribution among front and rear tires
$\left(x^P(\theta), y^P(\theta)\right)$	path parametrized by θ
$\psi^P(\theta)$	path heading
$b_l(\theta), b_r(\theta)$	left and right road boundary
r	arc length
$\mathbf{n}(\theta), \mathbf{t}(\theta)$	normal and tangential path vectors
$\tilde{e}^l(\mathbf{x}_k, \theta_k), \tilde{e}^l(\mathbf{x}_k, \theta_k)$	longitudinal and lateral path tracking error
p_ϵ	collision probability threshold

a more complex combined slip dynamic model including load transfer. Our method applies to both models.

Kinematic Model

The kinematic model is a simplified car model with a fixed rear wheel and a steerable front wheel with state $\mathbf{x} = [\mathbf{p}, \phi, \delta, v_x, v_y]$ and controls $\mathbf{u} = [\dot{\delta}^u, \dot{v}_x^u]$ with lateral velocity $v_y = 0$. The rear-wheel driven vehicle with inter-axle distance L and continuous kinematic model

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\delta} \\ \dot{v}_x \end{bmatrix}}_{\dot{\mathbf{x}}} = \begin{bmatrix} v_x \cos(\phi) \\ v_x \sin(\phi) \\ \frac{v_x}{L} \tan(\delta) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} \dot{\delta}^u \\ \dot{v}_x^u \end{bmatrix}}_{\mathbf{u}}, \tag{5.3}$$

is described by a discrete time model by integration $\mathbf{x}_{k+1} = \mathbf{x}_k + \int_k^{k+\Delta t_k} \dot{\mathbf{x}} \, \mathrm{d}t = f(\mathbf{x}_k, \mathbf{u}_k)$. A fourth order Runge-Kutta scheme ensures integration between timesteps k to sufficient accuracy.

We limit the steering angle, $|\delta| \leq \delta_{\max}$, the steering speed, $|\dot{\delta}| \leq \dot{\delta}_{\max}$, the longitudinal speed, $v_x \leq v_{x,\max}$, as well as braking and acceleration $\dot{v}_{x,\min} \leq \dot{v}_x \leq \dot{v}_{x,\max}$ to reasonable values conforming with the vehicle's performance and some rules of the road, e.g. limits on speed.

We account for, and prohibit, unsafe driving modes such as high cornering speeds by limiting the product of longitudinal velocity and yaw-rate

$$|v_x \dot{\phi}| \leq (v_x \dot{\phi})_{\max}, \tag{5.4}$$

which essentially poses a velocity dependent constraint on the vehicle's maximum allowed curvature. This model works well for less aggressive driving behaviors that are not too close to the vehicle's limits of handling.

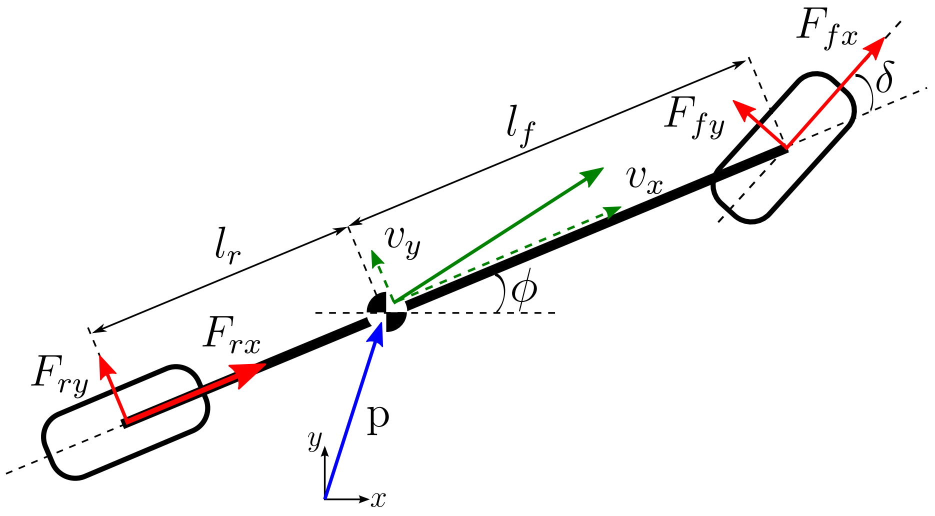


Figure 5-3: Dynamical half-car model

Combined Slip Model with Load Transfer

Combined braking and steering is one of the most essential aspects of vehicle safety and motivates our choice for a combined slip model [89]. Specifically, we allow load transfer between the front and rear tires - a dynamic that is often leveraged by rally racing drivers to control the yaw dynamics [205]. For simplicity, we neglect suspension and wheel dynamics, and assume that only the front wheel is steerable. The motion model is a dynamical half-car model with mass m, and yaw moment of inertia I_z. The distances of front and rear wheel from the center of gravity (**cog**) are l_f and l_r respectively. h is the height of the **cog**. The position of the **cog** in inertial frame is given by $\mathbf{p} = [x, y]$ and the heading by ϕ. v_x, v_y denote the longitudinal and lateral velocities in the car-body fixed axis system, and $F_{\alpha\beta}$, $\alpha \in \{f, r\}$, $\beta \in \{x, y\}$, the tire force components, see Figure 5-3. The equations of motion of the dynamical half-car model are then described by

$$\dot{\mathbf{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}, \tag{5.5}$$

$$\begin{bmatrix} m\dot{v}_x \\ m\dot{v}_y \\ I_z\ddot{\phi} \end{bmatrix} = \begin{bmatrix} F_{fx}\cos\delta - F_{fy}\sin\delta + F_{rx} + mv_y\dot{\phi} \\ F_{fx}\sin\delta + F_{fy}\cos\delta + F_{ry} - mv_x\dot{\phi} \\ l_f\left(F_{fx}\sin\delta + F_{fy}\cos\delta\right) - l_rF_{ry} \end{bmatrix}; \tag{5.6}$$

and define a discrete time model by integration in the same way as for the kinematic model.

We denote the state as $\mathbf{x} = [\mathbf{p}, \phi, \delta, v_x, v_y]$, control the vehicle by steering velocity $\dot{\delta}$, and longitudinal slip s, and write them as $\mathbf{u} = [\dot{\delta}^u, s^u]$. s^u defines the longitudinal slips as $s_{fx} = \gamma(s^u)s^u$, $s_{rx} = (1-\gamma(s^u))\,s^u$. $\gamma(s^u) \in [0,1]$ specifies the longitudinal slip distribution among front and rear tire. This enables us to model both front and rear wheel driven vehicles as well as arbitrary all-wheel drive configurations.

The lateral slips for front and rear tire are

$$s_{fy} = \frac{(v_y + l_f\dot{\phi})\cos\delta - v_x\sin\delta}{v_x\cos\delta + (v_y + l_f\dot{\phi})\sin\delta}, \quad s_{ry} = \frac{v_y - l_r\dot{\phi}}{v_x}. \tag{5.7}$$

The total tire slips are $s_\alpha = \sqrt{s_{\alpha x}^2 + s_{\alpha y}^2}$, $\alpha \in \{f, r\}$. We can compute the tire forces as

$$F_{\alpha\beta} = \mu_{\alpha\beta}F_{\alpha z},\ \alpha \in \{f, r\},\ \beta \in \{x, y\}, \tag{5.8}$$

with normal loads on front and rear tires

$$F_{fz} = \frac{mg(l_r - \mu_{rx}h)}{l_f + l_r + h(\mu_{fx}\cos\delta - \mu_{fy}\sin\delta - \mu_{rx})}, \tag{5.9}$$

$$F_{rz} = mg - F_{fz}. \tag{5.10}$$

Pacejka's simplified magic formula [26] defines the corresponding friction coefficients as

$$\mu_{\alpha\beta} = -(s_{\alpha\beta}/s_\alpha)\mu_\alpha, \ \ \alpha \in \{f, r\}, \ \ \beta \in \{x, y\}, \tag{5.11}$$

with

$$\mu_\alpha = D_\alpha \sin\left(C_\alpha \arctan(B_\alpha s_\alpha)\right), \ \ \alpha \in \{f, r\}. \tag{5.12}$$

$D_\alpha, C_\alpha, B_\alpha$ are tire and road surface specific coefficients.

We limit the total friction forces for both front and rear tires by

$$F_\alpha = \sqrt{F_{\alpha x}^2 + F_{\alpha y}^2} \leq \mu_{\alpha,\max} F_{\alpha z}, \;\; \alpha \in \{f, r\}, \tag{5.13}$$

to ensure operation in a safe driving envelope. $\mu_{\alpha,\max}$ is the maximum allowed friction coefficient. The constraint essentially limits the yaw rate dependent on the state $\mathbf{x}$. The constraint is suitable even for race-car driving under significant amounts of slip [89] and prohibits unsafe driving modes.

The constraints on steering speed $|\dot{\delta}| \leq \dot{\delta}_{\max}$, steering angle $|\delta| \leq \delta_{\max}$, and longitudinal speed, $v_x \leq v_{x,\max}$ remain the unchanged. We introduce an additional constraint on the lateral velocity $|v_y| \leq v_{y,\max}$.

5.4 Method

In this section we describe the method to solve the general problem of (5.2) in a specific setting.

5.4.1 Overview

We formulate a NMPC to compute a safe trajectory for the predefined time horizon. The constrained optimization consists of the following costs and constraints.

Cost

To maintain generality of the problem formulation while easing the understanding of the specifics of the instantiation, we slightly alter the notation $\hat{J}_h$, $\hat{J}_t$, compare (5.2), to J_h, J_t and will include weighting factor between these two losses in (5.42). We define the time-dependent weighted combination of J_h and J_t to avoid the necessity of prediction of future human inputs in Section 5.4.7. The cost term $J_t(\mathbf{x}_{0:m}, \mathbf{u}_{0:m-1})$ defined in Section 5.4.3 consists of terms commonly used in autonomous driving. It is responsible for giving feedback to the driver in the form of slightly *nudging* them back into the correct direction without strong intervention. If diverted too far from a common autonomous driving trajectory, e.g. the road's center (Section 5.4.2), the trajectory cost will guide the driver back to safety and avoid strong future control intervention to maintain safety (Section 5.4.3). It also directly penalizes controls, the yaw rate and other features indicating discomfort or extreme driving conditions. This is to avoid unnecessary future intervention by small intervention early on.

The cost term J_h penalizes the deviation of the system from the current acceleration and steering angle specified by the human driver. This term and its generalization to higher order states is described in Section 5.4.6.

Constraints

The optimization is subject to a set of constraints: (1) to respect the transition model of the system, including kinematic and dynamic constraints, described in Section 5.3.3, (2) to maintain the vehicle within the limits of the road, indicated in Section 5.4.4 and (3) to avoid other traffic participants in the sense of guaranteeing a probability of collision below p_ϵ, as given in Section 5.4.5.

Constrained Optimization

The resulting NMPC, which solves (5.2), is then described in Section 5.4.8.

5.4.2 Lane Tracking

In this section we build on the MPCC method of [60, 103, 118] and apply it to our problem setting. The control framework combines path generation and path tracking into one problem. The MPCC essentially plans a progress-optimal path by (nonlinear) projection of the vehicle's position onto the centerline. This is different from tracking controllers in that the controller has more freedom to determine the state trajectories to follow the given path. For example this allows jointly optimize for a velocity profile which in tracking is already defined by a reference trajectory. The resulting controller can plan and follow a path inside a specified corridor which is similar to time-optimal paths in race tracks when the horizon is chosen long enough.

We can adapt the MPCC control framework to our problem formulation by including additional cost terms into the optimization such as the minimal intervention cost, see Section 5.4.6. We use the center line as a reference path but employ it merely as a measure of progress by selecting appropriate weights in the optimization. The MPCC control framework also allows us formulate analytic road-boundary constraints by constraining the deviation from the centerline.

The MPCC approach is a suitable choice for our Parallel Autonomy formulation since it can (a) ensure safety by staying within the the road defined as a the *contour*, (b) can integrate slight *nudging* by penalizing deviation from the center line, and (c) integrate the driver's goal of achieving progress along the road.

Before posing the adapted MPCC problem and integrating it into our approach, we introduce several preliminaries, including the parametrization of the reference path (for which we will use the center line of the driving lane), and the definition of useful error measures due to an approximated path progress.

Progress on the Reference Path

The vehicle at position $\mathbf{p}_k = (x_k, y_k)$ at time k tracks a continuously differentiable two-dimensional geometric reference path $\big(x^P(\theta), y^P(\theta)\big)$ with path parameter θ and the tangential and normal vectors

$$\mathbf{t}(\theta) = \begin{bmatrix} \frac{\partial x^P(\theta)}{\partial \theta} \\ \frac{\partial y^P(\theta)}{\partial \theta} \end{bmatrix}, \ \mathbf{n}(\theta) = \begin{bmatrix} -\frac{\partial y^P(\theta)}{\partial \theta} \\ \frac{\partial x^P(\theta)}{\partial \theta} \end{bmatrix}. \tag{5.14}$$

The direction of the path is given by

$$\phi^p(\theta) = \arctan\left(\frac{\partial y^p(\theta)}{\partial x^p(\theta)}\right), \tag{5.15}$$

and we will refer to $\Delta\phi = \phi - \phi^p(\theta)$ as the deviation of the vehicle's heading to the path. We parametrize the path by the arc length r, such that $\partial\theta/\partial r = 1$. This allows us to estimate the progress of the vehicle along the reference path with the vehicle's actually driven longitudinal velocity $v_{x,k}$ and distance $r = \int v_x \, \mathrm{d}t$, for small $\Delta\phi$. While parametrization of curves by the arc length is not trivial, cubic spline parametrizatios are close to the arc length if the distance between knots is small relative to the curvature. Since the ego vehicle will follow a given road with sufficiently low deviation from the reference, enforced by the road's boundary, we can assume that

$$\Delta\theta \approx \Delta r = v_x \Delta t, \tag{5.16}$$

holds. This additional assumption yields an approximated progress along the path parameter

$$\theta_{k+1} = \theta_k + v_{x,k}\Delta t_k, \tag{5.17}$$

where $v_{x,k}\Delta t_k$ describes the approximated progress at time step k. Ideally, we want to compute the path parameter $\theta^P(x_k, y_k)$ of the closest point on the reference path to the ego vehicle's current position $\mathbf{p}_k = (x_k, y_k)$ to evaluate the associated progress, distance, and relative heading of the vehicle to the path to evaluate costs and road boundary constraints.

Finding $\theta^P(x_k, y_k)$ exactly from the projection operator involves embedding the optimization

$$\theta^p(x_k, y_k) = \arg\min_{\theta'_k} \left(x_k - x^P(\theta'_k)\right)^2 + \left(y_k - y^P(\theta'_k)\right)^2, \tag{5.18}$$

into the NMPC optimization. The projection optimization is computationally expensive for general curves and renders the direct projection operator unsuitable for fast MPC. We therefore approximate $\theta^P(x_k, y_k)$ by (5.17) for a fast lookup of path costs and constraints.

Longitudinal Error

The approximation of the curvilinear abscissa $\theta^P(x_k, y_k)$ by θ_k introduces two errors (see Figure 5-4) if the vehicle's actual path deviates from the reference path: a longitudinal (lag) error along the path and a lateral (countouring) error normal to the

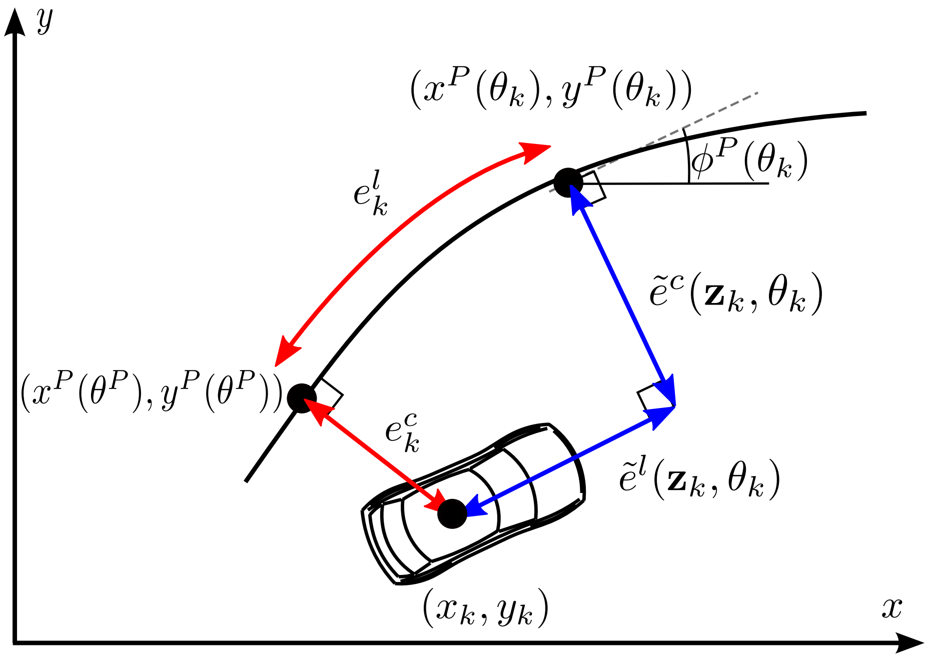

y
$(x^P(\theta_k), y^P(\theta_k))$
$\phi^P(\theta_k)$
e_k^l
$\tilde{e}^c(\mathbf{z}_k, \theta_k)$
$(x^P(\theta^P), y^P(\theta^P))$
e_k^c
$\tilde{e}^l(\mathbf{z}_k, \theta_k)$
(x_k, y_k)
x

Figure 5-4: **Path tracking coordinate system:** Approximation of actual path abscissa θ^P by virtual integrator θ_k, and therefore approximation of the true projection $(x^P(\theta^P), y^P(\theta^P)$ by $(x^P(\theta_k), y^P(\theta_k)$. The path projection estimation causes the longitudinal error e_k^l approximated by $\tilde{e}^l(\mathbf{x}_k, \theta_k)$, and contouring error e_k^c approximated by $\tilde{e}^c(\mathbf{x}_k, \theta_k)$. The contouring error $\tilde{e}^c(\mathbf{x}_k, \theta_k)$ is also used for an approximation of the lateral distance of the vehicle to the reference path.

path. The longitudinal error is

$$\tilde{e}^l(\mathbf{x}_k, \theta_k) = \frac{\mathbf{t}(\theta_k)^\top}{||\mathbf{t}(\theta_k)||} \begin{bmatrix} x_k - x^P(\theta_k) \\ y_k - y^P(\theta_k) \end{bmatrix} \tag{5.19}$$

$$= -\cos\phi^P(\theta_k)\left(x_k - x^P(\theta_k)\right) - \sin\phi^P(\theta_k)\left(y_k - y^P(\theta_k)\right), \tag{5.20}$$

resulting from projecting the position error of the vehicle with respect to the path's abscissa θ_k along the path's tangent $\mathbf{t}(\theta_k)$, see Figure 5-5.

For sufficiently small $\tilde{e}^l(\mathbf{x}_k, \theta_k)$ the approximated path progress is close to the actual curvilinear abscissa ((5.16)), and $\theta_k \approx \theta^P(x_k, y_k)$. The longitudinal error $\tilde{e}^l(\mathbf{x}_k, \theta_k)$ needs to be penalized in the MPCC optimization to keep the error of the approximated evolution θ_k along the path sufficiently small, as suggested in [103].

Contouring Error

The contouring error

$$\tilde{e}^c(\mathbf{x}_k, \theta_k) = \frac{\mathbf{n}(\theta_k)^\top}{||\mathbf{n}(\theta_k)||} \begin{bmatrix} x_k - x^P(\theta_k) \\ y_k - y^P(\theta_k) \end{bmatrix} \tag{5.21}$$

$$= \sin\phi^P(\theta_k)\left(x_k - x^P(\theta_k)\right) - \cos\phi^P(\theta_k)\left(y_k - y^P(\theta_k)\right) \tag{5.22}$$

describes the deviation of the vehicle's actual position from the estimated position projected onto the path's normal. It is thus a measure of how far the vehicle deviates from a given reference path in lateral direction.

The MPCC cost function

$$J_{\text{MPCC}}(\mathbf{x}_k, \theta_k) = \mathbf{e}_k^\top Q \mathbf{e}_k - \rho v_{x,k}, \tag{5.23}$$

with path error vector, formed from approximated longitudinal and contouring error

$$\mathbf{e}_k = \begin{bmatrix} \tilde{e}^l(\mathbf{x}_k, \theta_k) \\ \tilde{e}^c(\mathbf{x}_k, \theta_k) \end{bmatrix}, \tag{5.24}$$

balances the trade-off between contouring error, longitudinal error, and approximated path progress v_k. We can encode the previously discussed nudging behavior for diverting too far from the reference path and the driver's anticipated intention to make progress along the road through the parameters $Q \in \mathbb{S}^2_{++}$ and $\rho \in \mathbb{R}_+$. Since each stage k in the complete cost function will be scaled with Δt_k, see (5.42) in Section 5.4.8, $v_{x,k}$ is not multiplied with Δt_k but represents the approximate progress along the path regardless.

5.4.3 Trajectory Cost

The trajectory cost

$$J_t(\mathbf{x}_k, \mathbf{u}_k, \theta_k) = J_{\text{MPCC}}(\mathbf{x}_k, \theta_k) + \mathbf{u}_k^\top R \mathbf{u}_k + \dot{\phi}_k \alpha \dot{\phi}_k. \tag{5.25}$$

contains the MPCC cost, compare (5.23), and additionally penalizes strong control inputs and yaw rate, a measure of driving comfort. The weights $R \in \mathbb{S}_{++}$ and $\alpha \in \mathbb{R}_+$ allow for different prioritization of these objectives. J_{MPCC} readily encodes the penalization of large deviation from the reference path and simultaneously encourages progress along the road that aligns with the driver's goal of progressing towards their destination.

By not only minimizing the intervention J_h but also including the trajectory costs J_t into the total cost function we achieve the following:

Continuous Increase in Intervention

Potentially startling and confusing the driver by abrupt intervention may result in an increased risk of degenerated driver performance. Instead of suddenly taking over control in a thrashing manner to enforce safety constraints we achieve a continuous increase in intervention by including the trajectory cost. Therefore, it is possible to provide feedback to the driver by inducing a slight *nudging* behavior, potentially increasing driver alertness and preparation while continuously increasing the level of intervention the closer the system moves towards meeting safety constraints.

Preventing Disruptive Interventions by Small Early Intervention

Early and small intervention resulting in *nudging* the driver into beneficial directions also reduces the risk of large intervention in the future. Additionally to this direct effect, some hazardous situations can be avoided altogether due to potentially increased awareness and early feedback to the driver. Therefore J_t also aligns with our goal to minimize the occurrence of extreme intervention.

5.4.4 Road Representation and Static Obstacle Constraints

The ego vehicle's reference path is parametrized by a C^1-continuous clothoid following the road network through pre-specified points. We approximate the clothoid by cubic splines. Closely spaced knots parameterize the spline by the arc length to sufficient accuracy. The evaluation of the clothoid is computationally expensive because it requires solving a Fresnel integral Cubic splines enable fast and analytic evaluation of the reference path, its derivatives, and the road boundaries needed for solving the non-linear optimization.

The signed lateral distance $d(\mathbf{x}_k, \theta)$ of the ego vehicle's position $\mathbf{p}_k$ from the reference path is given by the projection along the normal of the reference path at the current curvilinear abscissa θ^P, approximated by θ_k such that $d(\mathbf{x}_k, \theta_k) = \tilde{e}^c(\mathbf{x}_k, \theta_k)$.

The free and drivable space of the ego vehicle at the path abscissa θ_k is limited by both the left road boundary $b_l(\theta_k)$ and the right road boundary $b_r(\theta_k)$, Figure 5-5. They are parametrized by cubic splines to enable analytic evaluation and derivation. The boundaries may enclose other obstacles $\mathcal{O}$ in addition to the limits of the road such as static obstacles, construction zones and other potentially dangerous objects.

We limit the lateral offset of the vehicle to the path by

$$b_l(\theta_k) + w(\Delta\phi_k) \leq d(\mathbf{x}_k, \theta_k) \leq b_r(\theta_k) - w(\Delta\phi_k), \tag{5.26}$$

to ensure that the ego vehicle stays clear of all static obstacles. We must choose $w(\Delta\phi_k)$ larger than half the vehicle's width to account for the ego vehicle's relative orientation to the path. We project the vehicle's orientation-dependent outline onto the

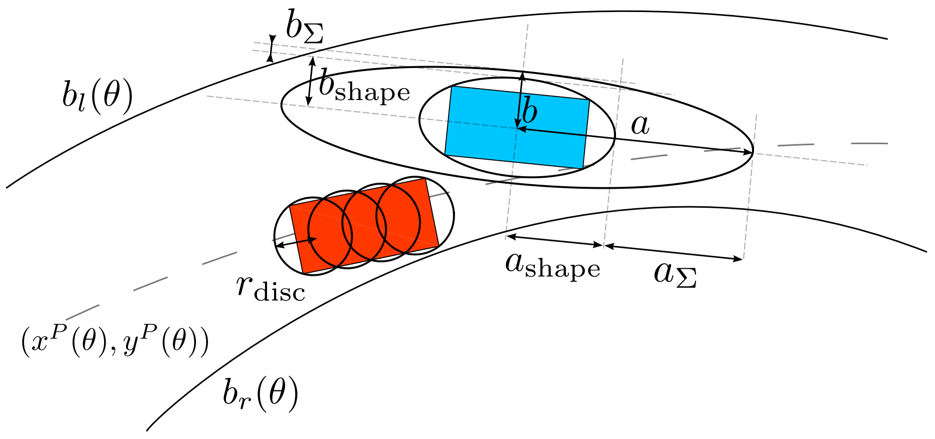


Figure 5-5: **Agent and road representation:** Ego vehicle (red) approximated by circles of radius r_{disc} and other vehicle (blue) with shape- and uncertainty-ellipse corresponding to minimum occupancy probability p_ϵ. Road boundaries $b_l(\theta)$ on the left and right $b_r(\theta)$ of reference path $(x^P(\theta), y^P(\theta))$.

reference path's normal and arrive at $w(\Delta\phi_k) = w/2\cos(\Delta\phi_k) + \max(l_f, l_r)\sin(|\Delta\phi_k|)$ as a suitable bound. Simply taking the vehicle's length as an upper bound turns out to be too conservative for navigating in narrow road segments. It would also severely limit the host vehicle's ability to utilize the road to its full extent to avoid other vehicles and road hazards. We additionally constrain the difference between the ego vehicle's heading ϕ_k and the path's heading $\phi^P(\theta_k)$ as

$$|\phi_k - \phi^P(\theta_k)| \leq \Delta\phi_{\max}. \tag{5.27}$$

5.4.5 Avoidance of Dynamic Traffic Participants

The ego vehicle needs to plan its motion in dynamic, uncertain environments. Successful and efficient operation of autonomous systems in such environments requires reasoning about the uncertain future evolution of the states of the moving agents and obstacles. The evolution of trajectories of other vehicles can be reasonably predicted for a short time (typically less than a second) by considering physical quantities such as the vehicles' estimated velocities, directions, and yaw rates. We can analytically propagate uncertain states with linear dynamics subject to Gaussian noise. The prediction of other traffic participants within the driving environment over longer horizons is much stronger influenced by their intentions, motivations, and goals. Thus, a different representation is more suitable. In our experiments, we choose to *model* the mean and uncertainty propagation for vehicles following their lanes. Regardless, it is straightforward to integrate more elaborate predictions of other agents that are parametrized by trajectories of Gaussians. Nonetheless, accurate predictions of other agents are beyond the scope of this work.

In the following, we will derive chance constraints that ensure safety despite uncertain dynamic obstacle predictions. We will also detail the approximations necessary to make the computationally challenging problem tractable for real-time applications.

We formulate collision chance constraints

$$P(C) < p_\epsilon, \tag{5.28}$$

as a bound on the probability of collision

$$P(C) = \int\int I_C(\mathbf{x}_k, \mathbf{x}_k^i) p(\mathbf{x}_k, \mathbf{x}_k^i)\, d\mathbf{x}_k\, d\mathbf{x}_k^i. \tag{5.29}$$

between the ego vehicle's footprint $\mathcal{B}(\mathbf{x}_k)$ and vehicle i's footprint $\mathcal{B}(\mathbf{x}_k^i)$. We integrate over the distributions of their states $\mathbf{x}_k$, $\mathbf{x}_k^i$ and indicate a collision with

$$I_C(\mathbf{x}_k, \mathbf{x}_k^i) := \begin{cases} 0, & \text{if } \mathcal{B}(\mathbf{x}_k) \cap \mathcal{B}^i(\mathbf{x}_k^i) = \emptyset \\ 1, & \text{otherwise.} \end{cases} \tag{5.30}$$

For the general case, including rectangular objects and arbitrary distributions, the probability of collision can only be estimated by Monte Carlo sampling that makes it unsuitable for use in real-time optimization.

Assuming conditional independence between vehicles, no uncertainty over the host vehicle's state $\mathbf{x}_k$, and uncertainty only over the other vehicles state $\mathbf{x}_k^i$, simplifies (5.29). The chance constraint for encoding (5.28) simplifies to

$$\mathcal{B}(\mathbf{x}_k) \cap \mathcal{B}^i(\mathbf{x}_k^i, \mathbf{\Sigma}_k^i, p_\epsilon) = \emptyset. \tag{5.31}$$

Here, $\mathcal{B}(\mathbf{x}_k)$ is the host vehicle's footprint and $\mathcal{B}^i(\mathbf{x}_k^i, \mathbf{\Sigma}_k^i, p_\epsilon)$ the other vehicle's footprint with probability of more than p_ϵ. We will derive a closed form approximation for this constraint in the following.

For brevity, index i will be omitted in this part of this section. The shapes of other traffic participants, such as vehicles, pedestrians, and bikes are approximated by a footprint encompassing ellipse $\mathcal{E}(a_{\text{shape}}, b_{\text{shape}})$. The ellipse has orientation ϕ with semi-major axes a_{shape} and b_{shape} in longitudinal and lateral direction of the obstacle respectively, see Figure 5-5. We take advantage of the analytic description of their occupied area to derive approximate collision states that can be described in closed analytic form. We assume to know the evolution of the obstacles' future trajectories up to some uncertainty in the form of a posterior distributions parameterized by mean $\mathbf{x}_{0:m}$ and uncertainty $\mathbf{\Sigma}_{0:m}$ trajectories. For our experiments, we supply a model of the growth of uncertainty

$$\mathbf{\Sigma}_{k+1} = \min\left(\mathbf{\Sigma}_k + \mathbf{\Sigma}\Delta t_k, \mathbf{\Sigma}_{\max}\right). \tag{5.32}$$

We describe the vehicle's position uncertainty with the covariance matrix $\mathbf{\Sigma}_k = \operatorname{diag}[(\sigma_k^a)^2, (\sigma_k^b)^2]$ at time k. We approximate the covariance in the vehicle's position to be aligned with the vehicle's heading. The principal axis of the vehicle's Gaussian and the principal axis of the encompassing ellipse are therefore aligned, see Figure 5-5. The growth of uncertainty is determined by $\mathbf{\Sigma} = \operatorname{diag}[(\sigma^a)^2, (\sigma^b)^2]$. We model the uncertainty growth as bounded by $\mathbf{\Sigma}_{\max}$ to account for the high likelihood of vehicles to stay in their current lanes. While we have chosen this model of propagation of uncertainty for simplicity in our experiments, other more complex models or learned predictions are feasible in this framework as well.

The p_ϵ level-set lines of the Gaussian $\mathcal{N}(0, \mathbf{\Sigma}_k)$ describing the position uncertainty of the other traffic participants form ellipses $\mathcal{L}(a_{\Sigma_k}, b_{\Sigma_k})$ with coefficients

$$\begin{bmatrix} a_{\Sigma_k} \\ b_{\Sigma_k} \end{bmatrix} = \begin{bmatrix} \sigma_k^a \\ \sigma_k^b \end{bmatrix} \left(-2\log\left(p_\epsilon 2\pi\sigma_k^a\sigma_k^b\right)\right)^{1/2}. \tag{5.33}$$

The Minkowski-sum $\oplus$ of the axis aligned uncertainty ellipse $\mathcal{L}(a_{\Sigma_k}, b_{\Sigma_k})$ and the shape ellipse $\mathcal{L}(a_{\text{shape}}, b_{\text{shape}})$ is approximately an ellipse

$$\begin{aligned}\mathcal{B}^i(\mathbf{x}_k^i, \boldsymbol{\Sigma}_k^i, p_\epsilon) &\subset \mathcal{L}(a_{\Sigma_k}, b_{\Sigma_k}) \oplus \mathcal{L}(a_{\text{shape}}, b_{\text{shape}}) \\ &\subset \mathcal{L}(a_{\Sigma_k} + a_{\text{shape}}, b_{\Sigma_k} + b_{\text{shape}}),\end{aligned} \tag{5.34}$$

inscribing the vehicle's footprint up to an uncertainty threshold p_ϵ. A similar, but conservative outer approximation is defined in [36]. The axis alignment of the shape and uncertainty ellipses enables us to directly add the coefficients of the semi-major axes to find the obstacle's approximate ellipse $\mathcal{B}(\mathbf{x}_k^i, \Sigma_k^i, p_\epsilon)$ with occupancy probability above the p_ϵ threshold.

We approximate the rectangular shape of the ego vehicle by a set of 4 discs $\mathcal{R}^j(\mathbf{x}_k)$, $j \in \{1, \dots, 4\}$, of radius r_{ego} chosen to conservatively enclose the vehicle, see Figure 5-5.

$$\mathcal{B}(\mathbf{x}_k) \subset \left(\bigcup_{j \in \{1,\dots,4\}} \mathcal{R}^j(\mathbf{x}_k) \right), \tag{5.35}$$

It is necessary to employ discs instead of ellipses for the ego vehicle, since the ego vehicle and the other vehicles are not necessarily axis aligned. The approximate Minkowski sum can not be derived easily for non-axis aligned ellipses. The approximate Minkowski sum of the ego-car's discs and an ellipse on the other hand has a closed form description. We can therefore transform the collision constraint (5.31) to

$$\left(\bigcup_{j \in \{1,\dots,4\}} \mathcal{R}^j(\mathbf{x}_k) \right) \bigcap \mathcal{L}(a_{\Sigma_k} + a_{\text{shape}}, b_{\Sigma_k} + b_{\text{shape}}) = \emptyset, \tag{5.36}$$

and we can apply the approximate Minkowski sum on the ego car's j-th discs $\mathcal{R}^j(\mathbf{x}_k)$

and the previously derived occupancy ellipse to form analytic collision constraints

$$c_k^{\text{obst.,i}}(\mathbf{x}_k) = \begin{bmatrix} \Delta x_j \\ \Delta y_j \end{bmatrix}^\top R(\phi)^\top \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix} R(\phi) \begin{bmatrix} \Delta x_j \\ \Delta y_j \end{bmatrix} \Bigg|_{k,i} > 1, \quad \forall j \in \{1, \ldots, 4\} \tag{5.37}$$

where Δx_j, Δy_j are the distance of the ego vehicle's j-th disc to the center of the obstacle i at time k. $R(\phi)$ is the rotation matrix corresponding to the obstacle's heading, and

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a_{\text{shape}} + a_{\Sigma_k} + r_{\text{disc}} \\ b_{\text{shape}} + b_{\Sigma_k} + r_{\text{disc}} \end{bmatrix}, \tag{5.38}$$

describe the semi-major axis of the resulting constraint-ellipse. As a result we have an analytic closed-form constraint prohibiting collisions with probability higher than p_ϵ with other vehicles.

For more elaborate methods of uncertainty propagation, such as sigma point transformations or Monte Carlo sampling, we can compute a conservative approximation of a Gaussian with the principal axis aligned to the expected vehicle orientation. The conservative approximation would define new uncertainty ellipse coefficients $a_{\Sigma_k}, b_{\Sigma_k}$. The uncertainty propagation does not affect optimization run-time as much since it only needs to be completed once before the optimization. More generally, if we can describe the p_ϵ occupancy of other vehicles by an ellipse, the remainder of the method remains applicable without the need for the axis alignment assumption.

5.4.6 Minimal Intervention

It is our goal to follow the human input very closely and intervene only when deemed necessary. The intervention cost

$$J_{\text{h}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_k^h) = \Big[\{\mathbf{u}_k, \mathbf{x}_k\} - \mathbf{u}_k^h \Big]^\top K \Big[\{\mathbf{u}_k, \mathbf{x}_k\} - \mathbf{u}_k^h \Big], \tag{5.39}$$

with weights $K \in \mathbb{S}_{++}$ penalizes the deviation of the system's state or input from the human driver's predicted desired input $\mathbf{u}_k^h$. We may adapt the function depending on the which human inputs $\mathbf{u}_k^h$ are observable. We penalize the deviation of the corresponding states $\mathbf{x}_k$ and control inputs $\mathbf{u}_k$ from human inputs $\mathbf{u}_k^h$ accordingly.

Instead of predicting future driver inputs $\mathbf{u}_0^h$ from a driver model [179], we hold the human input constant $\mathbf{u}_k^h = \mathbf{u}_0^h \; \forall k$ and penalize the difference of planned controls and the current human input. As we will see in the forthcoming Section 5.4.7, we will introduce a time-dependent weighting such that the impact of $J_{\mathrm{h}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_0^h)$ dominates the optimization in the short term but drops off quickly. Hence, we do not need to rely on accurate driver predictions. The optimization plans trajectories sufficiently close to the short term intentions of the human driver and subsequently follows the long-term goals with increasing time.

In our experimental setup we can only observe the driver's desired steering angle δ^h and acceleration $\dot{v}_x^h$, but not the steering speed $\dot{\delta}^h$. The minimal intervention cost accounting only for the deviation from the *current* human driver input $\mathbf{u}_0^h = [\delta_0^h, \dot{v}_{x,0}^h]^\top$ becomes

$$J_{\mathrm{h}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_0^h) = \begin{bmatrix} \dot{v}_{x,k}^u - \dot{v}_{x,0}^h \\ \delta_k - \delta_0^h \end{bmatrix}^\top K \begin{bmatrix} \dot{v}_{x,k}^u - \dot{v}_{x,0}^h \\ \delta_k - \delta_0^h \end{bmatrix}. \tag{5.40}$$

Nonetheless, if available, the framework is general enough to incorporate predictions of desired human inputs. If observable, we may also use other quantities as the driver's desired input, such as steering velocity, acceleration, or torque.

5.4.7 Merging of Minimal Intervention and Trajectory Costs

We propose a linear, time-dependent combination of the cost of intervention J_h (5.40) and trajectory cost J_t (5.25)

$$J(\mathbf{x}_k, \mathbf{u}_k, \theta_k, \mathbf{u}_0^h) = \beta\omega(t_k) J_{\mathrm{h}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{u}_0^h) + (1 - \omega(t_k))\, J_{\mathrm{t}}(\mathbf{x}_k, \mathbf{u}_k, \theta_k), \tag{5.41}$$

with the goal of decreasing dependence on accurate prediction of future driver commands $\mathbf{u}_{0:m}^h$ and instead only relying on the current human input $\mathbf{u}_0^h$.

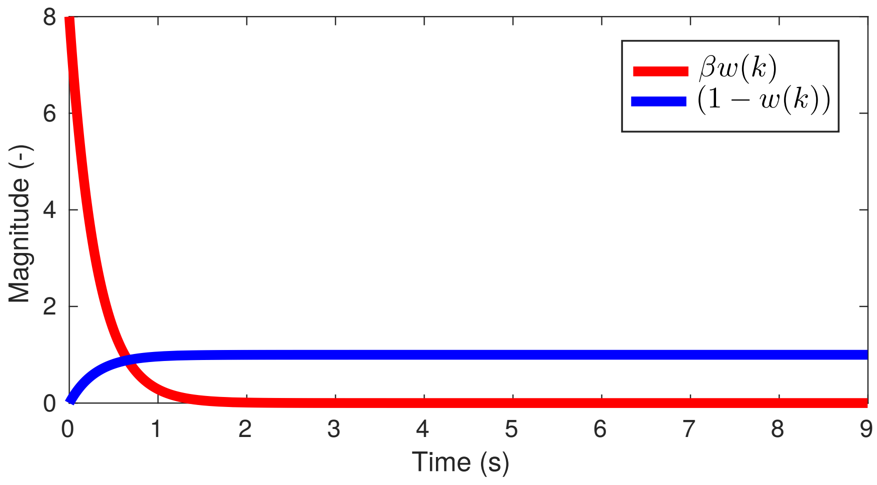


Figure 5-6: **Minimal intervention and trajectory cost trade off:** Comparison of magnitude of minimal intervention cost J_h weighting ($\beta w(t_k)$), red, and trajectory cost J_h weighting $(1 - w(t_k))$, blue, over the planning horizon.

We chose weights $\beta \in \mathbb{R}_+$ and an exponential decay function $w(t_k) = \exp(-\alpha t_k)$, $\alpha \in \mathbb{R}_+$ to increase the impact of the human input in the short-term, see Figure 5-6. We set $w(t_k)$, the relative weight of J_h to drop off to 10% of its initial value after only 0.5s. Therefore, and because of a large β, the minimal intervention cost J_h dominates the optimization in the short term to make the system very responsive to current human inputs. With increasing time it will more and more rely on J_t, while J_h loses relevance. The system's planned output trajectory may not coincide with the human driver's anticipated trajectory for timesteps far into the future. Regardless, the output trajectory will eventually *snap* into the correct human long term intention as time progresses. A deviation of the system will only become noticeable if safety constraints become active. The human driver will perceive the system as inactive otherwise. A study on driving in shared safety systems [18] underlines the importance: Human drivers approved of a surprisingly high amount of intervention if their own high-level goals were achieved. Intervention may not be perceived as such if it does not follow adversary goals.

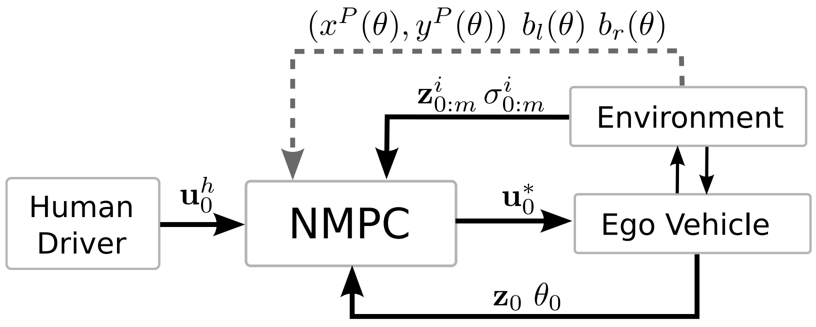


Figure 5-7: NMPC control scheme

5.4.8 Optimization

We collect the aforementioned state-, dynamics-, path-, and obstacle constraints and form the final constrained nonlinear optimization problem

$$\mathbf{u}^*_{0:m-1} = \underset{\mathbf{u}_{0:m-1}}{\arg\min} \sum_{k=0}^{m} J(\mathbf{x}_k, \mathbf{u}_k, \theta_k, \mathbf{u}^h_0)\Delta t_k \tag{5.42}$$

$$\text{s.t. } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \tag{5.43}$$

$$\theta_{k+1} = \theta_k + v_k \Delta t_k, , \tag{5.17}$$

$$\mathbf{x}_{\min} < \mathbf{x}_k < \mathbf{x}_{\max}, \tag{5.44}$$

$$\mathbf{u}_{\min} < \mathbf{u}_k < \mathbf{u}_{\max}, \tag{5.45}$$

$$|\phi_k - \phi^P(\theta_k)| < \Delta\phi_{\max}, \tag{5.27}$$

$$|v_x \dot{\phi}_k| < (v_x \dot{\phi})_{\max}, \tag{5.4}$$

$$F_\alpha = \sqrt{F^2_{\alpha x} + F^2_{\alpha y}} \leq \mu_{\alpha,\max} F_{\alpha z}, \ \ \alpha \in \{f, r\}, \tag{5.13}$$

$$b_l(\theta_k) + w(\Delta\phi_k) \leq d(\mathbf{x}_k, \theta_k) \leq b_r(\theta_k) - w(\Delta\phi_k), \tag{5.26}$$

$$c_k^{\text{obstacle,i}}(\mathbf{x}_k) > 1, \ i = \{1, \dots, n\}, \tag{5.37}$$

$$\forall k \in \{0, \dots, m\}.$$

Constraint (5.4) is active for the kinematic model only, while (5.13) is only active for the dynamical model. At initialization the path $(x^P(\theta), y^P(\theta))$ and boundaries $b_l(\theta)$ and $b_r(\theta)$ are given by the road and static obstacles, see Figure 5-7 and Algorithm 8. The boundaries are updated once new information about roads or static obstacles become available. At the beginning of each control loop the initial states $\mathbf{x}_0$, θ_0, human control input $\mathbf{u}_0^h$, and predictions of other traffic participants mean $\mathbf{x}_{0:m}^i$ and covariances $\Sigma_{0:m}^i$ are provided to the NMPC and the corresponding constraint ellipses are computed. Subsequently, we solve (5.42) and the system executes the resulting optimal control $\mathbf{u}_0^*$. The system consequently returns to the beginning of the loop, see Figure 5-7 and Algorithm 8. We solve the optimization problem (5.42) with a Primal-Dual Interior Point solver generated by FORCES Pro [53].

Algorithm 8 Summary of NMPC control flow

```
 1: Sense environment $\mathcal{E}$ including static obstacles $\mathcal{O}$ and dynamic obstacles $\mathbf{x}_0^i$,
    $a_{\text{shape}}^i, b_{\text{shape}}^i$, $i \in \{1, \dots, n\}$;
 2: Initialize $(x^P(\theta), y^P(\theta))$, the path spline of the ego vehicle from the road network;
 3: Compute $b_l(\theta)$, $b_r(\theta)$, the path-boundary splines including road constraints and
    static obstacles $\mathcal{O}$;
 4: loop
 5:     Sense env. $\mathcal{E}$, i.e. $\mathcal{O}$, $\mathbf{x}_0^i$, $a_{\text{shape}}^i, b_{\text{shape}}^i$, $i \in \{1, \dots, n\}$;
 6:     if Static environment changed then
 7:         Update $\big(x^P(\theta), y^P(\theta)\big)$;
 8:         Compute $b_l(\theta)$, $b_r(\theta)$;
 9:     end if
10:     for $i \in \{1, \dots, n\}$ do;
11:         Predict $\mathbf{x}_{1:m}^i$;
12:         Propagate $\Sigma_{0:m}^i$;                                    ▷ (5.32)
13:         Compute $(a_{0:m}^i, b_{0:m}^i)$;                               ▷ (5.38)
14:     end for
15:     Find $\theta_0$, ego vehicle path abscissa;                         ▷ (5.18)
16:     Measure $\mathbf{x}_0$, the current ego state;
17:     Measure $\mathbf{u}_0^h$, the current human driver input;
18:     Compute $\mathbf{u}_0^*$;                                           ▷ (5.42)
19:     Apply $\mathbf{u}_0^*$ to system;
20: end loop
```

5.4.9 Technical Discussion

The described method guarantees dynamic collision avoidance while staying within the limits of the road up to the time horizon. The conditions for this are that (a) the solver can find a solution, and (b) knowledge of the road, static obstacles, dynamic obstacles on the road, and a prediction of the mean states of the dynamic obstacles' motion up to a sufficient uncertainty are available. Since this work does not address recursive feasibility, (a) can not always be guaranteed.

Quality of the Trajectory

The optimization problem is non-convex and, e.g. in the event of deciding to overtake a vehicle on the left or right, can become of combinatorial nature. Although the Primal-Dual Interior Point method solves for a local solution, we have achieved good results even for poor initialization for four reasons:

1. The local solution trajectory will *snap* into the human driver's desired trajectory eventually as time progresses. Only the current control $\mathbf{u}_0$ is executed and since the current human inputs $\mathbf{u}_0^h$ dominate the cost function, as discussed in Section 5.4.7, it is acceptable to currently execute a local solution.

2. Any local solution is safe. While not necessarily optimal in the sense of minimal intervention, any feasible solution will abide by the safety constraints.

3. Increased static and dynamic regularization parameters yield robust optimization in practice.

4. The J_{MPCC} cost term shapes the trajectory optimization process closer to convex by encouraging progress along the path and creating a cost basin by penalizing large lateral offsets from the path.

Executing the NMPC, including solving a nonlinear non-convex optimization problem, has a number of general drawbacks such as uncertain convergence, potentially unbounded runtime, and the lack of guarantees of optimality. In future work a high-level trajectory planner, solving the combinatorial problem, could be applied to yield a meaningful initialization to ensure global optimality.

Number of Variables

Per stage 1 state originates from the path integrator θ_k, 2 inputs, and 5 (7) states $\mathbf{x}_k$ for the kinematic (dynamical) model, thus 8 (10) variables exist. The NMPC plans over 50 stages, including 8 (10) variables per stage, and thus 394 (492) variables in total excluding the initial states for the kinematic (dynamical) model are needed.

Number of Constraints

Including the system dynamics and path progress evolution in total 6 (8) equality constraints need to be respected. 14 (16) inequality constraints result from state, input, velocity-yaw-rate, (friction), heading-deviation, and road boundary constraints. $4n$ additional inequality constraints originate from n dynamic obstacles in the environment and the 4 circle approximation of the ego vehicle. In total $20\,(24) + 4n$ constraints are specified per stage. Depending on the solver only a subset of those are active.

5.5 Results

We evaluate the capabilities of our approach in a variety of simulated scenarios. The human driver controls a physical steering wheel and pedals, see Figure 5-8, which generate the desired inputs $\mathbf{u}_0^h = [\delta_0^h, \dot{v}_{x,0}^h]$, i.e. steering angle δ_0^h and acceleration $\dot{v}_{x,0}^h$. The human inputs are then processed in the NMPC formulation to achieve safe motion. The reference path and the road boundaries b_l and b_r are designed to fit the road network. We adopt a variable step size approach in all scenarios, and for all motion models, to increase the time horizon of the planner without sacrificing computation cost. During the first 10 steps we employ $\Delta t_k = 0.1s$ and $\Delta t_k = 0.2s$ for the remaining 40 steps, resulting in a planning horizon of nearly $9s$. During all experiments the cost function's weights remained unchanged and are displayed in Table 5.2.

5.5.1 Left Turn Across Traffic, Merging, and Overtaking

In this challenging left turn cross traffic scenario, see Figure 5-1 and Figure 5-9, the ego vehicle intends to merge into the oncoming traffic while avoiding collisions with cross traffic. Afterwards, the driver attempts to overtake a vehicle while traffic obstructs

Table 5.2: Cost weights

Function	Parameter	Value
MPCC	$Q = \text{diag}(q_{\text{long}}, q_{\text{lat}})$	$\text{diag}(1.0, 1.0)$
Progress	ρ	5.0
Control	$R = \text{diag}(r_{acc}, r_{\dot{\delta}})$	$\text{diag}(1.0, 1.0)$
Yaw rate	α	0.1
Minimal intervention	$K = \text{diag}(k_{\dot{v}}, k_{\delta})$	$\text{diag}(1.0, 2.0)$
Scaling of intervention	β	500

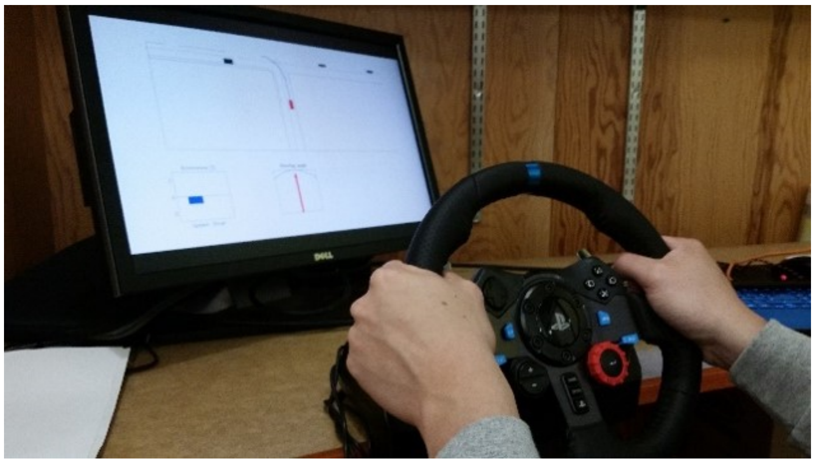


Figure 5-8: Virtual driving setup with steering wheel and pedals

the maneuver. For this challenging scenario with multiple dynamic obstacles and a decision-making component the simpler kinematic model was employed.

We have evaluated the method in a set of 100 randomly generated scenarios. In these scenarios the initial positions, trajectories including acceleration and velocity profiles, of all other traffic participants were randomly generated. The Parallel Autonomy system was able to ensure safety at all times, although we purposefully caused unsafe human driver inputs which would have resulted in crashes without the proposed system.

In the following we will present two representative examples for two different human driving styles, i.e. an aggressive and a calm driver. We define the total intervention as a direct measure to compare the NMPC's behavior subject to different scenarios and driving styles:

$$\text{Intervention} := \frac{100}{2}\left(\frac{|\dot{v}_{x,0} - \dot{v}^h_{x,0}|}{2\dot{v}_{x,\max}} + \frac{|\delta_0 - \delta^h_0|}{2\delta_{\max}}\right) \quad [\%]. \tag{5.46}$$

We measure the amount of total intervention as the sum of deviation from the human input normalized with the maximum possible inputs scaled to % .

Aggressive Driver

In the first case, see Figure 5-9, an aggressive driver nearly collides with the right road boundary even before entering the intersection ①, which is prevented by countersteering of the autonomous system since the right road boundary constraint becomes active. Then, the driver tries to accelerate into the intersection ②, although other vehicles are just passing by, resulting in a near collision. Our system brings the ego vehicle to a full stop, lets the other vehicles pass, and then proceeds by allowing the driver to merge into the traffic when a large enough gap appears. Here the dynamic obstacle constraints prevented a collision and influenced the merging behavior. To achieve this high-level of reasoning and scheduling, both a long planning horizon as well as low-level control on a trajectory basis combined with high replanning frequency to quickly account for changes in the environment are necessary.

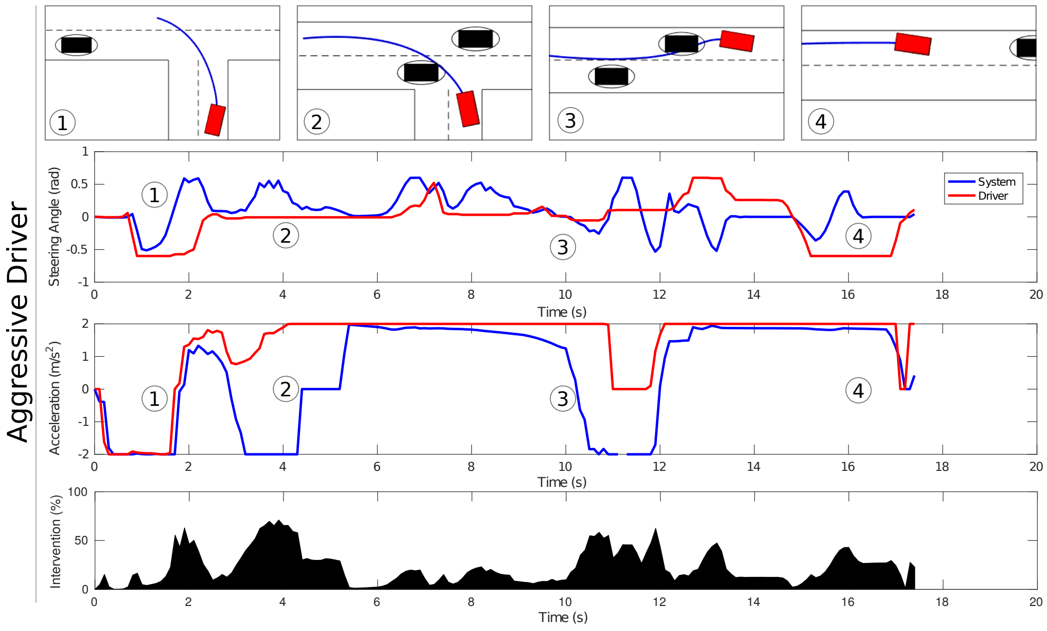


Figure 5-9: **Aggressive left turn across traffic**: The system's steering angle and acceleration are displayed in blue, the human input in red. Snapshots of the current scenes at specific time-stamps are displayed above the acceleration and steering plots: The ego vehicle in red, the MPC planned path in blue. All other vehicles in black. An aggressive driver causes multiple critical situations where the system is forced to intervene to large amounts to keep the vehicle in a safe state. Large deviation from the driver's desired acceleration and steering wheel angle to the actual system output are observable. E.g. collision at time (2) is prohibited by strong braking.

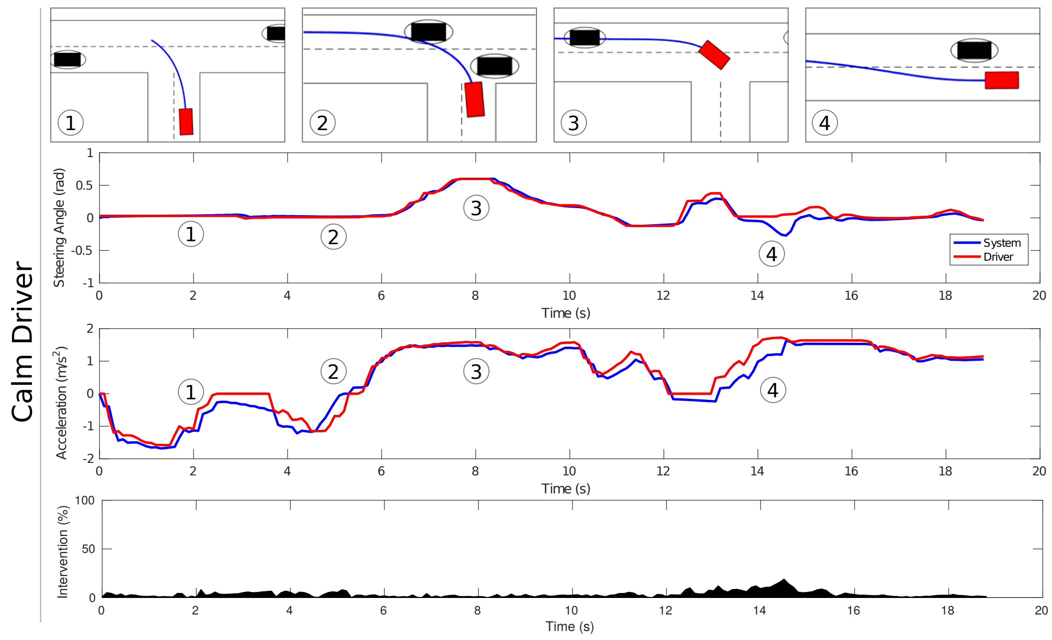


Figure 5-10: **Calm left turn with traffic**: System output stays close to the desired human acceleration and steering wheel angle. An exception appears at (4) where the driver is not counter steering enough to prohibit a predicted collision with the left road boundary.

At ③ the driver approaches a preceding vehicle with high relative speed and tries to collide by accelerating even further. Our system brakes the ego vehicle, first with gradually increasing effort and then with maximum acceleration $a_{\max}$, and allows an overtaking maneuver once the oncoming traffic has passed. At ④ the driver erratically tries to break through the right road boundary, which is prohibited by our system, and the ego vehicle can continue driving on the road safely. In all these cases, the system can guard the human driver from actually causing any harm to himself and others.

Due to the severity of the scenario a maximum intervention of 71% and 23% on average was necessary to assure safety.

Calm Driver

We show the opposite spectrum of how our method reacts in Figure 5-10: A calm driver experiences the same previous scenario. We observe that if the inputs from the human driver are deemed safe, barely any difference between human and system

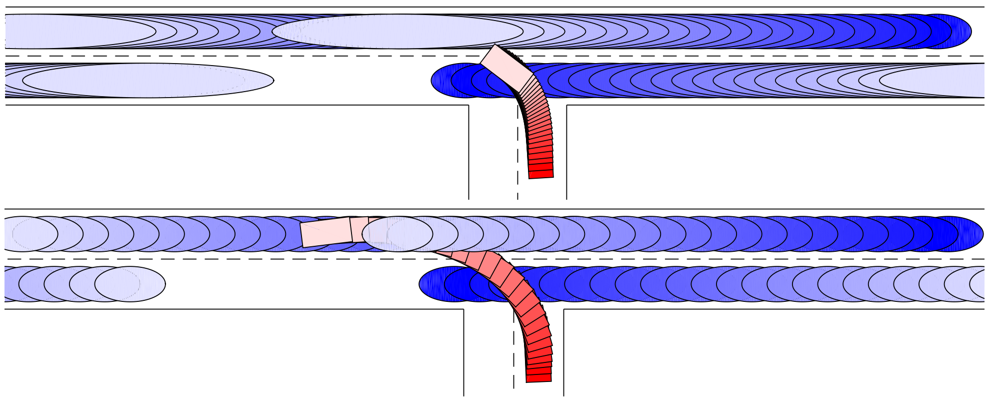


Figure 5-11: **Impact of prediction uncertainty:** Comparison of NMPC *plans* with uncertainty estimate (top) and without (bottom) shown by the ellipses representing their occupancy probability threshold. Predicted future states are shown in fading colors $0.4s$ apart over a horizon of $9s$.

inputs occurs. The system thus minimizes intervention if no critical situations arise. Since steering the vehicle with steering wheel and pedals in simulation is not an easy task, due to the lack of feedback, the human driver did not brake sufficiently at ② and misses to counter-steer during a lane change maneuver ④. Here the system applied the slight *nudging* behavior to carefully guide the driver away from the imminent collision with the left road boundary. Even in this situation, the maximum intervention did not exceed 16% and was only 3% on average for the whole scenario, which underlines the functionality of the minimal intervention principle.

5.5.2 Impact of Uncertainty

Taking the uncertainty in the prediction of other vehicles into account is significant since future states can deviate substantially from the expectation. In the case of neglecting uncertainties, the planned behavior can be more aggressive and is given more leeway in the constraints. See Figure 5-11-bottom, where the vehicle is allowed to merge into the lane in front of a second vehicle. Taking future obstacles' uncertainty growth into account, see Figure 5-11-top, results in more conservative behavior, and the ego vehicle is prohibited from merging in front of the oncoming driver. Here, we discussed the plan of the NMPC and not the actual observed behavior. Since the control loop runs at more than $10Hz$, frequently updating the planned trajectory, the executed controls will be less conservative as they adapt to new observations. It will be possible to actually observe the true positions and velocities of the other vehicles over time and thus replan with lower uncertainty. Nonetheless, we have shown the impact of uncertainty aware planning by including chance constraints into the optimization.

5.5.3 Snowy Race Track

Table 5.3: Vehicle specifications

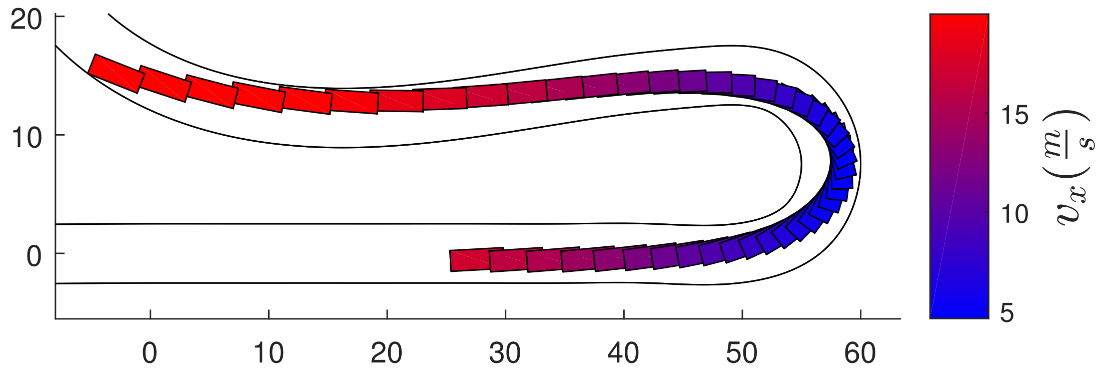
Parameter	Value
l_f, l_r, w, h	$1.2, 1.5, 1.76, 0.9$ (m)
m, I_z	$1400kg$, $3000kg/m^2$
$B_\alpha, C_\alpha, D_\alpha$ (snow)	$5, 2, 0.3$

Sharp Turn

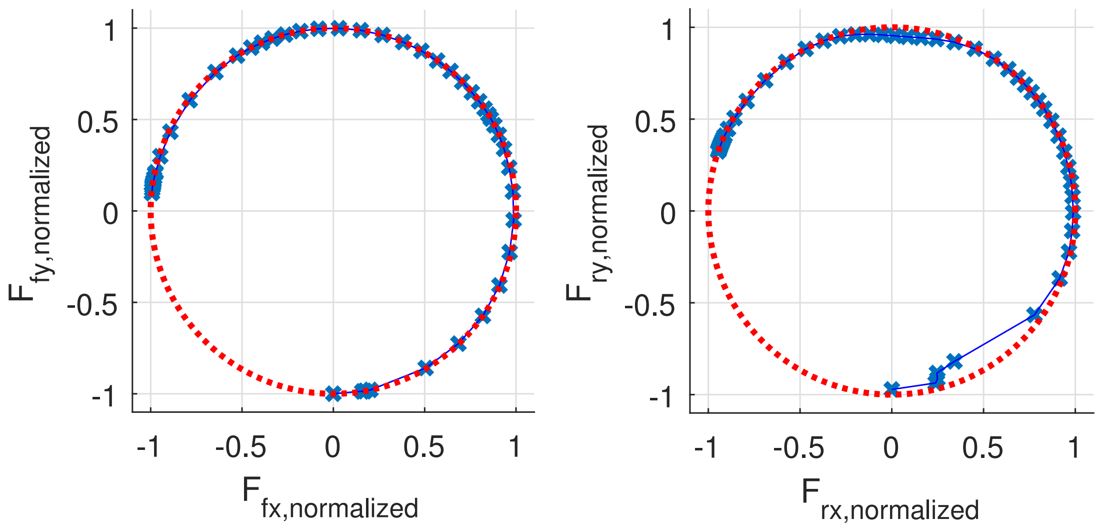
In this scenario, see Figure 5-12(a), the vehicle enters a sharp left turn on a race track. The current human inputs would cause the vehicle to spin off the road at high speed. The controller brakes the vehicle to a safe speed complying with the friction constraints, see Figure 5-12(b), then accelerates at the exit of the turn to maximize progress while always respecting the road's limits. The planned trajectory shows similarities to a racing line during high-speed cornering. This behavior shows the advantage of longitudinal and lateral control. Without deceleration, the vehicle would not have been able to complete the turn, see Figure 5-12(a). The plan maintains a smooth acceleration profile during the turn and maximizes the use of available forces, see Figure 5-12(b), while abiding by the friction constraint.

Sudden Appearance of Obstacles

A suddenly appearing static obstacle represented by an ellipse in the driving-path of the vehicle, e.g. a stationary deer, needs to be avoided, see Figure 5-13(a). After replanning a new trajectory, the vehicle can swerve to the left side of the road to avoid the static obstacle in its previous driving path. The optimization does not have any

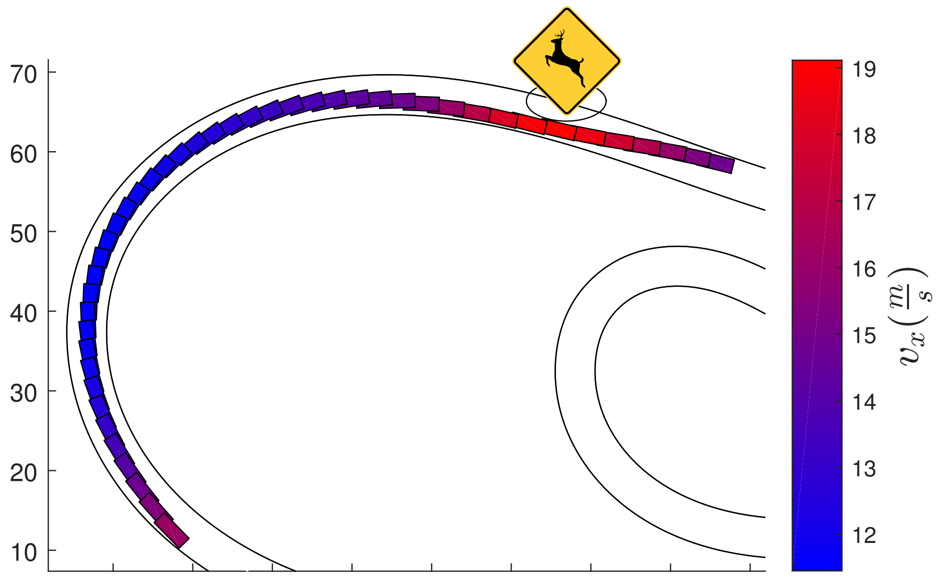


(a) NMPC *plan* of vehicle, where vehicle poses are $0.2s$ apart for a horizon of $9s$.

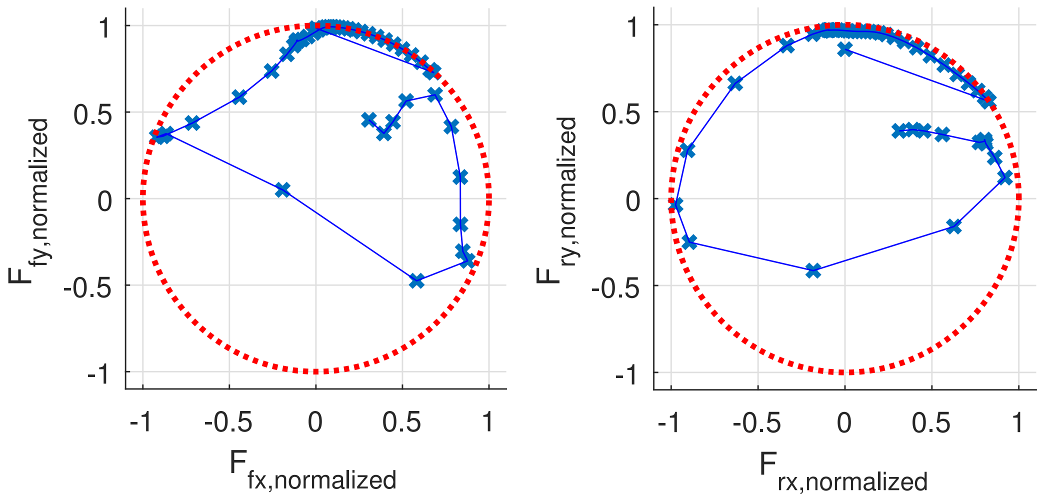


(b) Normalized friction for both tires, $F_{\alpha\beta,\text{normalized}} = F_{\alpha\beta}/(\mu_{\alpha,\max}F_{\alpha z})$.

Figure 5-12: **NMPC plan in tight turn**: The planner decelerates into the sharp corner to comply with friction constraints.



(a) NMPC *plan* of vehicle avoiding unexpected obstacle, where vehicle poses are $0.2s$ apart for a horizon of $9s$.



(b) The friction constraints remain valid during the avoidance maneuver.

Figure 5-13: **Avoiding unexpected obstacle**: The system successfully avoids the collision.

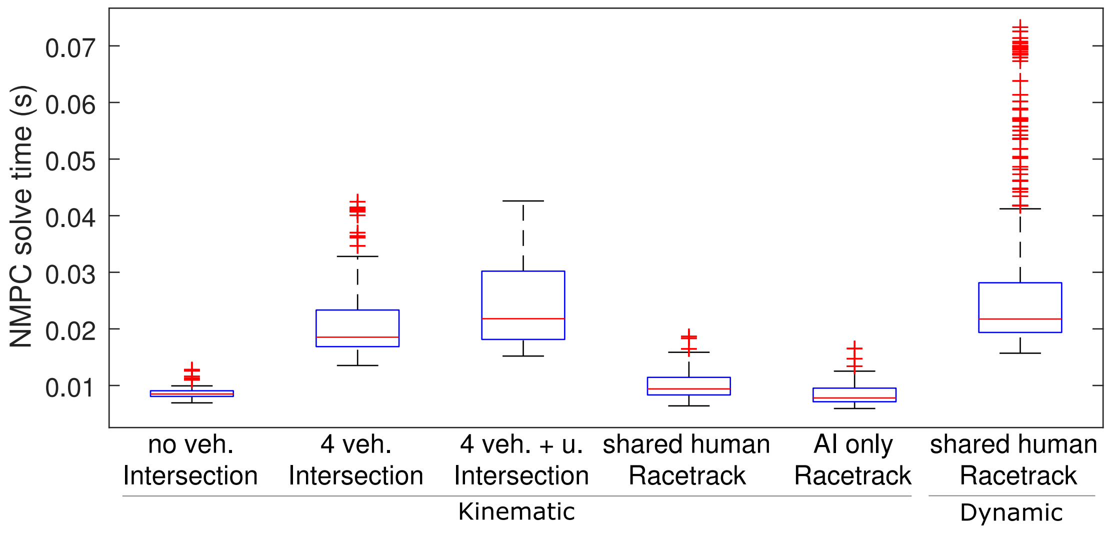


Figure 5-14: **NMPC computation times:** Computation times for the different human in the loop scenarios and varying number of other traffic participants, as well as scenarios with high uncertainty. Finally the case of AI only, without the human in the loop. On the far right is the dynamic vehicle model in a static environment. Results were computed on a single core of an AMD Ryzen 7 1700X @3.4Ghz.

incentive to slow down in the vicinity of obstacles and thus continues to accelerate to increase progress along the road until it brakes for the imminent left corner.

5.5.4 Computation Time

We display the NMPC solve times collected during several runs on a single-core CPU in Figure 5-14. For the kinematic model, we observed a strong influence of the complexity of the scenario on the computation time. In the case of no dynamic obstacles, we observed solve-times of less than $20ms$ even for a challenging race track with many tight turns, forcing the NMPC to intervene and decelerate due to velocity-yaw-rate constraints. In cases where the system needs to nudge into tight gaps while simultaneously deciding whether a subsequent overtaking maneuver is feasible, computation times can reach up to $50ms$ in exceptional cases.

The dynamical vehicle model is surprisingly fast on average but exhibits worst-case run-times of nearly $75ms$. While we achieved robust performance and convergence for all tests in static environments, for some cases in dynamic environments no feasible solution was found and we, therefore, exclude the dynamical model from those use cases.

In summary, our system was able to reach the goal replanning frequency of $10Hz$ at all times for the kinematic vehicle model in complex dynamic environments, and the dynamical model in simpler and static environments.

5.6 Discussion

In this work, we presented a NMPC that minimizes deviation from the human input while ensuring safety according to our proposed general Parallel Autonomy control framework. We have shown the increased functionality compared to other approaches in complex and more realistic driving scenarios. The approach is capable of reasoning over long time horizons of more than $9s$ in real-time, i.e. more than $10Hz$, while maintaining close to the human input without a necessary prediction layer for human intention.

We have shown our method to work with a kinematic model in challenging and highly complex dynamic environments, and a dynamical model in static environments.

Future work will try to reduce general limitations of NMPC, such as uncertain convergence and lack of a guarantee of optimality by the initialization in a correct homotopy class, or by exploring strategies on how to deal with failure cases where no solution can be found, or to extend the presented method to achieve provable safety beyond the planning horizon by ensuring recursive feasibility. Additional tests on our experimental platform [135], will also enclose an inference framework to gain more elaborate predictions of other traffic participants, future trajectories, and their uncertainty. More work is also needed to investigate the importance of visibility to other drivers [16].

Instead of nudging the driver in the correct direction, it is possible to not intervene at all if the driver is doing well. The impact of the different methodologies on human drivers will be studied in the future. Lastly, the proposed receding horizon planner also applies to fully autonomous vehicles if the minimal intervention cost is excluded and future experiments will show the functionality.

Chapter 6

Conclusions and Closing Remarks

This thesis presents algorithms for learning and control for interactions in mixed human-robot environments. We contribute methods that enable complex interactions towards robots that work with and around people for a variety of applications. We enable *(I) social human-robot interactions* by estimating and reasoning about the social preferences of people and leverage how actions can affect the beliefs and plans of other people, even under *(II) uncertainty and with partial observations.* We demonstrate agents that *(III) learn through competition* based on raw unprivileged visual observations. Toward a future where robots elevate and protect people, we introduce a *guardian system* in the form of *(IV) Parallel Autonomy.*

6.1 Final Thoughts and Lessons Learned

In the following, we will discuss valuable insights we have gained from working on the topics in this thesis.

6.1.1 Local Solutions in Optimization and Nash Equilibria

Many of the approaches presented in this thesis rely on first-order (gradient descent) or second-order (Newton's method) optimization methods. Regardless of whether single-agent real-time optimal control and motion planning or multi-agent game-theoretic planning in the form of solving for Nash equilibria, most approaches discussed employ local optimization. The benefits of fast solve times and real-time operation come at the cost of local optimality and relying on smooth and well defined objective functions. The optimization algorithms are limited to converge to a local minimum of a single homotopy class that does not necessarily coincide with the global optimum. The homotopy class is highly dependent on the initial solution guess but is a detrimental factor for actual performance. Unfortunately, whether an algorithm avoids an obstacle on the left or the right may be determined by the randomness of an initial guess instead of a deliberate choice. Making the correct global decision, i.e. correctly deciding on a homotopy class, can be more important than local optimality. Regardless, there are ways to overcome some of the issues of these methods towards arriving at a globally optimal solution. Convexifying the problem simplifies the solution process as it is straight forward to find the only existing globally optimal solution. While this is not always possible, one can also convert the original problem into a mixed-integer combination of convex problems, essentially evaluating all possible homotopy classes. Finally, another option is to supply an initial guess inside the globally optimal homotopy class and to additionally restrict the solution to never leave the homotopy class during convergence. However, all these improvements require sophisticated domain knowledge that may not always be available. They also open other combinatorial problems when evaluating all homotopy classes. This can pose issues for real-time operation. Additionally, the procedures are less well defined in game-theoretic settings. While there are also different classes of algorithms relying on sampling or searching, these usually do not scale well to high-dimensional and complex problems for online operation. Regardless, the success of learning highly skilled policies in this thesis as well as in the field of RL in general, suggests that combining offline policy-learning for an initial guess and refining the solution online through higher-order methods, such as in MPC, promises scalable and multi-purpose motion-planning for problems where models of the environment are either learned, or known a priori. This approach is especially promising for game-theoretic problems, where policies can be first learned through self-play offline and subsequently solutions refined online through state-of-the-art Nash equilibrium solvers presented in this thesis.

6.1.2 Scalable and General Solutions vs Domain-Specific Engineering

The recent decade of progress in AI research has shown that general solutions which scale well will eventually surpass engineered domain-specific solutions in the long-term [192]. While manually engineered methods are mostly superior in the short term through faster computation time and better interpretability of outcomes they become increasingly difficult to scale to complex systems. As (parallel) compute capabilities increase over time, more general solution techniques that can be applied across domains become more successful. This observation shall by no means devalue contributions that do make domain-specific progress. The progress is essential in the pursuit of scientific understanding of the specialized fields and enables building frameworks that keep future more general solution methods in check. While there is excitement about the novel and unprecedented capabilities of general and scalable algorithms in AI, accelerated by the increase in compute, these advances also came at the cost of a lack in interpretability, understanding, and provable correctness of the solutions. While some engineering disciplines such as aviation adhere to a strict set of rules and safety standards, newer disciplines such as autonomous driving do not yet have a rigid body of requirements. For some applications which rely on learned perception and learned direct control from perception, it may never be possible to define a strict set of rules that prove safety in all cases. Instead, safety requirements may have to be more statistical in nature. In these cases, domain-specific knowledge is essential in defining the statistical tests and requirements that the more general solution methods have to adhere to. They may also give feedback and help improve the more general systems online and help to keep them in check. One may pose the question of whether these domain-specific fields should focus their efforts on building systems that ensure the general solution methods safety or whether they should continue developing solution methods themselves. Regardless, the advent of new capabilities in autonomy will yield many novel application areas that contribute positively to society and do not have such strict requirements. It is especially promising to move on from replicating human capabilities to reduce labor costs to inventing new products and applications that create more value instead of redistributing it.

6.1.3 Emergent Behavior

As we have seen in this thesis, emergent behavior may result from many different solution approaches ranging from optimal control, through solving for Nash equilibria, to self-play in Multi-Agent Reinforcement Learning (MARL). Nonetheless, a suitable problem formulation is equally critical. Problem formulations include specifying cost objectives, dynamics, and observation models. We also saw problems in which agents didn't have access to any of these models and had to learn them from scratch. On the one hand, policy optimization can leverage the learned models to accelerate learning such that we can interpret learning a model of the world as a part of the solution. On the other hand, we can also see model learning as part of the problem formulation since we may reuse learned models to learn policies for different tasks in similar environments. Problem formulations also include game or group dynamics in the case of multi-agent environments. While we usually see problem formulations as a prerequisite to propose potential solutions, we often make simplifications to the problems at hand to make solutions feasible. We replace sparse objectives with dense cost functions to guide optimizers along gradients (contouring cost in Chapter 5), impose artificial structure to reduce search spaces (Gaussian beliefs in Chapter 3), and decouple problems into hierarchies of subproblems (SVO estimation and planning in Chapter 2). These directions usually require some form of domain knowledge to transform the original sparse reward, infinite horizon problems into more amenable formulations. In many cases, such as in tying a shoe-lace or giving people emotional support, it is not clear how to concisely formulate a problem at all. It is debatable what optimization objective to choose. Ideally, we would advise robots to *make the world a better place*, which is a very high-level, fuzzy, long-term, mixed-human robot, and multi-agent goal. Miss-specifying this goal could have detrimental outcomes and has been a source of inspiration for movies starring robots that decide the world to be better off without humans. Getting stuck in local optima could have equally destructive outcomes. Aside from these concerns, game-theoretic formulations may be a way towards more general formulations. In this thesis we saw great promise in training agents in progressively more challenging environments in the form of self-play. Self-play as an auto-curriculum automatically regulates the difficulty of the environment: The competitor's performance increases whenever the own agent's performance increases. Other successful game-theoretic auto-curricula like the min-max formulation in GANs rely on similar principles but can be harder to train due to mode collapse. Regulating the right amount of competitiveness between generator and adversary can be difficult, especially in zero-sum games. Self-play in non-zero-sum formulations on the other hand has the potential to also discover cooperative strategies while still improving single agent performance. Generalizing this procedure of ever improving agents and environments could be the next step towards intelligent agents.

6.2 Closing Remarks

With systems capable of increasingly complex interactive behavior, robots are on their way into consumer products and our daily lives. By incorporating knowledge from disciplines that study human social behavior, game-theory as well as learning from observed human behavior, robots will learn to seamlessly work with and around us, and elevate and protect people. Intelligent agents that are cognizant of how their actions affect the environment, other agents, and their beliefs allow them to learn complex skills from interacting with others and themselves in imagination. The interplay of learning agents and imagined multi-agent environments that become more challenging as performance increases will bring us closer to agents that exhibit the generality and flexibility of human intelligence. Perhaps this will also bring us closer to understanding the nature of general intelligence itself.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] Pieter Abbeel and Andrew Y Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 1–8. ACM, 2005.

[3] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[4] Kurt A. Ackermann, Jürgen Fleiß, and Ryan O. Murphy. Reciprocity as an individual difference. *Journal of Conflict Resolution*, 60(2):340–367, 2016.

[5] Kurt A Ackermann and Ryan O Murphy. Explaining cooperative behavior in public goods games: How preferences and beliefs affect contribution levels. *Games*, 10(1):15, 2019.

[6] National Highway Traffic Safety Administration et al. 2015 motor vehicle crashes: overview. *Traffic safety facts research note*, 2016:1–9, 2016.

[7] Heejin Ahn, Andrea Rizzi, Alessandro Colombo, and Domitilla Del Vecchio. Robust supervisors for intersection collision avoidance in the presence of uncontrolled vehicles. *arXiv preprint arXiv:1603.03916*, 2016.

[8] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[9] Eugenio Alcalá, Vicenç Puig, Joseba Quevedo, and Ugo Rosolia. Autonomous racing using Linear Parameter Varying-Model Predictive Control (LPV-MPC). *Control Engineering Practice*, 95, 2020.

[10] J. Alonso-Mora, P. Gohl, S. Watson, R. Siegwart, and P. Beardsley. Shared control of autonomous vehicles based on velocity space optimization. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1639–1645, May 2014.

[11] D. Althoff, M. Althoff, D. Wollherr, and M. Buss. Probabilistic collision state checker for crowded environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 1492–1498, May 2010.

[12] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation. *Robotics and Automation Letters*, 5(2), 2020.

[13] Alexander Amini, Wilko Schwarting, Guy Rosman, Brandon Araki, Sertac Karaman, and Daniela Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 568–575. IEEE, 2018.

[14] Alexander Amini, Ava P Soleimany, Wilko Schwarting, Sangeeta N Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019.

[15] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[16] Hans Andersen, Wilko Schwarting, Felix Naser, You Hong Eng, Marcelo H Ang, Daniela Rus, and Javier Alonso-Mora. Trajectory optimization for autonomous overtaking with visibility maximization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2017.

[17] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma. Constraint-based planning and control for safe, semi-autonomous operation of vehicles. In *2012 IEEE Intelligent Vehicles Symposium*, pages 383–388, June 2012.

[18] Sterling J Anderson, Sisir B Karumanchi, Karl Iagnemma, and James M Walker. The intelligent copilot: A constraint-based approach to shared-adaptive control of ground vehicles. *Intelligent Transportation Systems Magazine, IEEE*, 5(2):45–54, 2013.

[19] Sterling J Anderson, Steven C Peters, Tom E Pilutti, and Karl Iagnemma. An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. *International Journal of Vehicle Autonomous Systems*, 8(2-4):190–216, 2010.

[20] 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*, In Press, 2018.

[21] James Andreoni and John Miller. Giving according to garp: An experimental test of the consistency of preferences for altruism. *Econometrica*, 70(2):737–753, 2002.

[22] ASIRT. Association for Safe International Road Travel 2016.

[23] David Autor, DA Mindell, and Elisabeth B Reynolds. The work of the future: Shaping technology and institutions, 2019.

[24] Haoyu Bai, David Hsu, and Wee Sun Lee. Integrated perception and planning in the continuous space: A pomdp approach. *The International Journal of Robotics Research*, 33(9):1288–1302, 2014.

[25] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2019.

[26] Egbert Bakker, Lars Nyborg, and Hans B. Pacejka. Tyre modelling for use in vehicle dynamics studies. In *SAE Technical Paper*. SAE International, 02 1987.

[27] Daniel Balliet, Craig Parks, and Jeff Joireman. Social value orientation and cooperation in social dilemmas: A meta-analysis. *Group Processes & Intergroup Relations*, 12(4):533–547, 2009.

[28] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[29] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *International Conference on Learning Representations*, 2018.

[30] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.

[31] A. Bautin, L. Martinez-Gomez, and T. Fraichard. Inevitable collision states: A probabilistic perspective. In *2010 IEEE International Conference on Robotics and Automation*, pages 4022–4027, May 2010.

[32] David Berreby. The robot revolution has arrived. National Geographic, 2020.

[33] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint:1604.07316*, 2016.

[34] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

[35] Delphine Bresch-Pietri and Domitilla Del Vecchio. Estimation for decentralized safety control under communication delay and measurement uncertainty. *Automatica*, 62:292 – 303, 2015.

[36] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[37] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[38] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[39] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[40] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[41] Noam Buckman, Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Sharing is caring: Socially-compliant autonomous intersection negotiation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6136–6143. IEEE, 2019.

[42] Colin F Camerer and Ernst Fehr. When does "economic man" dominate social behavior? *Science*, 311(5757):47–52, 2006.

[43] Jeffrey P Carpenter. Is fairness used instrumentally? evidence from sequential bargaining. *Journal of Economic Psychology*, 24(4):467–489, 2003.

[44] Jesus Velasco Carrau, Alexander Liniger, Xiaojing Zhang, and John Lygeros. Efficient implementation of Randomized MPC for miniature race cars. In *European Control Conference (ECC)*, 2016.

[45] Nicholas Chan, James Kuffner, and Matthew Zucker. Improved motion planning speed and safety using regions of inevitable collision. In *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control (RoManSy'08)*, July 2008.

[46] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by Cheating. In *Conference on Robot Learning (CoRL)*, 2020.

[47] Y. Chen, H. Peng, and J. Grizzle. Obstacle avoidance for low-speed autonomous vehicles with barrier function. *IEEE Transactions on Control Systems Technology*, PP(99):1–13, 2017.

[48] Changhyun Choi, Wilko Schwarting, Joseph DelPreto, and Daniela Rus. Learning object grasping for soft robot hands. *IEEE Robotics and Automation Letters*, 3(3):2370–2377, 2018.

[49] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. Algames: A fast solver for constrained dynamic games. *arXiv preprint arXiv:1910.09713*, 2019.

[50] Alexandre Constantin, Junghee Park, and Karl Iagnemma. A margin–based approach to vehicle threat assessment. *International Journal of Vehicle Autonomous Systems*, 12(4):384–411, 2014.

[51] E. Dagan, O. Mano, G. P. Stein, and A. Shashua. Forward collision warning with a single camera. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 37–42, June 2004.

[52] Carsten KW De Dreu and Paul AM Van Lange. The impact of social value orientations on negotiator cognition and behavior. *Personality and Social Psychology Bulletin*, 21(11):1178–1188, 1995.

[53] Alexander Domahidi and Juan Jerez. FORCES Pro. embotech GmbH (`http://embotech.com/FORCES-Pro`), July 2014. Available at `http://embotech.com/FORCES-Pro`.

[54] Axel Dreves and Matthias Gerdts. A generalized nash equilibrium approach for optimal control problems of autonomous cars. *Optimal Control Applications and Methods*, 39(1):326–342, 2018.

[55] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A Theodorou, and James M Rehg. Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control. In *Conference on Robot Learning (CoRL)*, 2017.

[56] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[57] John W D'Attoma, Clara Volintiru, and Antoine Malézieux. Gender, social value orientation, and tax compliance. *CESifo Economic Studies*, 66(3):265–284, 2020.

[58] Amir Efrati. Waymo's Big Ambitions Slowed by Tech Trouble. *The Information*, 2018. Available at `https://www.theinformation.com/articles/waymos-big-ambitions-slowed-by-tech-trouble`.

[59] S. M. Erlien, S. Fujita, and J. C. Gerdes. Shared steering control using safe envelopes for obstacle avoidance and vehicle stability. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):441–451, Feb 2016.

[60] T. Faulwasser, B. Kern, and R. Findeisen. Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8642–8647, Dec 2009.

[61] Susann Fiedler, Andreas Glöckner, Andreas Nicklisch, and Stephan Dickert. Social value orientation and information search in social dilemmas: An eye-tracking analysis. *Organizational behavior and human decision processes*, 120(2):272–284, 2013.

[62] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[63] M. Forghani, J. M. McNew, D. Hoehener, and D. Del Vecchio. Design of driver-assist systems under probabilistic safety specifications near stop signs. *IEEE Transactions on Automation Science and Engineering*, 13(1):43–53, Jan 2016.

[64] Thierry Fraichard and Hajime Asama. Inevitable collision states. a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[65] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481. IEEE, 2020.

[66] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[67] FWHWA and US Department of Transportation. Ngsim-next generation simulation, 2017.

[68] Y. Gao, A. Gray, A. Carvalho, H. E. Tseng, and F. Borrelli. Robust nonlinear predictive control for semiautonomous ground vehicles. In *2014 American Control Conference*, pages 4913–4918, June 2014.

[69] Alexis Garapin, Laurent Muller, and Bilel Rahali. Does trust mean giving and not risking? experimental evidence from the trust game. *Revue d'économie politique*, 125(5):701–716, 2015.

[70] Tommy Gärling, Satoshi Fujii, Anita Gärling, and Cecilia Jakobsson. Moderating effects of social value orientation on determinants of proenvironmental behavior intention. *Journal of environmental psychology*, 23(1):1–9, 2003.

[71] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[72] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[73] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[74] David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[75] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[76] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[77] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, 2019.

[78] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[79] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[80] F. Havlak and M. Campbell. Discrete and continuous, probabilistic anticipation for autonomous robots in urban environments. *IEEE Transactions on Robotics*, 30(2):461–474, April 2014.

[81] Andrew J. Hawkins. Tesla owner in canada charged with 'sleeping' while driving over 90 mph. *The Verge*, 2020. Available at `https://www.theverge.com/2020/9/18/21445168/tesla-driver-sleeping-police-charged-canada-autopilot`.

[82] John Hawksworth, Richard Berriman, and Saloni Goel. Will robots really steal our jobs? an international analysis of the potential long term impact of automation. *PricewaterhouseCoopers, http://pwc. co. uk/economics, access*, 13, 2018.

[83] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[84] Johannes Heinrich and David Silver. Deep Reinforcement Learning from Self-Play in Imperfect-Information Games. *arXiv preprint: 1603.01121*, 2016.

[85] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[86] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[87] D. Hoehener, G. Huang, and D. Del Vecchio. Design of a lane departure driver-assist system under safety specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 2468–2474, Dec 2016.

[88] Zefan Huang, Wilko Schwarting, Alyssa Pierson, Hongliang Guo, Marcelo Ang Jr, and Daniela Rus. Safe path planning with multi-model risk level sets.

[89] Jeong hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *2013 American Control Conference*, pages 188–193, June 2013.

[90] Shariq Iqbal and Fei Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2019.

[91] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2375–2384, 2019.

[92] David H Jacobson and David Q Mayne. Differential dynamic programming. 1970.

[93] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[94] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[95] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations (ICLR)*, 2020.

[96] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *arXiv preprint 1807.00412*, 2018.

[97] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014.

[98] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016.

[99] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. IEEE, 2015.

[100] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*, 2012.

[101] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE, 2017.

[102] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

[103] D. Lam, C. Manzie, and M. C. Good. Model predictive contouring control for biaxial systems. *IEEE Transactions on Control Systems Technology*, 21(2):552–559, March 2013.

[104] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[105] John Launchbury. A DARPA perspective on AI, 2017.

[106] Alex Lee, Yan Duan, Sachin Patil, John Schulman, Zoe McCarthy, Jur van den Berg, Ken Goldberg, and Pieter Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5660–5667. IEEE, 2013.

[107] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017.

[108] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[109] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[110] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

[111] Nan Li, Dave W Oyler, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck R Girard. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on control systems technology*, 26(5):1782–1797, 2017.

[112] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[113] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[114] Lucas Liebenwein, Wilko Schwarting, Cristian-Ioan Vasile, Jonathan DeCastro, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Compositional and contract-based verification for autonomous driving on road networks. In *Robotics Research*, pages 163–181. Springer, 2020.

[115] Wim B. G. Liebrand and Charles G. McClintock. The ring measure of social values: A computerized procedure for assessing individual differences in information processing and social value orientation. *European Journal of Personality*, 2(3):217–230, 1988.

[116] Jyh-Ming Lien, S. Rodriguez, J. Malric, and N.M. Amato. Shepherding behaviors with multiple shepherds. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3402–3407, April 2005.

[117] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[118] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[119] Alexander Liniger and John Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2019.

[120] Alexander Liniger and John Lygeros. Real-Time Control for Autonomous Racing Based on Viability Theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, 2019.

[121] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 1994.

[122] Siqi Liu, Guy Lever, Nicholas Heess, Josh Merel, Saran Tunyasuvunakool, and Thore Graepel. Emergent coordination through competition. In *International Conference on Learning Representations (ICLR)*, 2019.

[123] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[124] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar Narasimhan, Madhava Krishna, and Krishna Murthy Jatavallabhula. MonoLayout: Amodal scene layout from a single image. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[125] Jason R Marden and Jeff S Shamma. Game theory and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[126] Charles G McClintock and Scott T Allison. Social value orientation and helping behavior. *Journal of Applied Social Psychology*, 19(4):353–362, 1989.

[127] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, July 2005.

[128] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[129] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[130] Frédéric Moisan, Robert ten Brincke, Ryan Murphy, and Cleotilde Gonzalez. Not all prisoner's dilemma games are equal: Incentives, social preferences, and cooperation. *Decision*, 5, 03 2017.

[131] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[132] Ryan O. Murphy and Kurt A. Ackermann. Social value orientation: Theoretical and measurement issues in the study of social preferences. *Personality and Social Psychology Review*, 18(1):13–41, 2014. PMID: 24065346.

[133] Ryan O. Murphy and Kurt A. Ackermann. Social preferences, positive expectations, and trust based cooperation. *Journal of Mathematical Psychology*, 67:45 – 50, 2015.

[134] Ryan O. Murphy, Kurt A. Ackermann, and Michel Handgraaf. Measuring social value orientation. *Judgment and Decision Making*, 6(8):771–781, 2011.

[135] Felix Naser, David Dorhout, Stephen Proulx, Scott Drew Pendleton, Hans Andersen, Wilko Schwarting, Liam Paull, Javier Alonso-Mora, Marcelo H Ang, Sertac Karaman, et al. A parallel autonomy research platform. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 933–940. IEEE, 2017.

[136] National Safety Council (NSC). Motor Vehicle Fatality Estimates 2012-2015.

[137] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[138] NHTSA. Traffic Safety Facts 2015.

[139] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[140] M. Nohmi, N. Fujikura, C. Ueda, and E. Toyota. Automatic braking or acceleration control system for a vehicle, January 3 1978. US Patent 4,066,230.

[141] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[142] Sachin Patil, Gregory Kahn, Michael Laskey, John Schulman, Ken Goldberg, and Pieter Abbeel. Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Algorithmic foundations of robotics XI*, pages 515–533. Springer, 2015.

[143] Sachin Patil, Jur Van Den Berg, and Ron Alterovitz. Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3238–3244. IEEE, 2012.

[144] A. Pierson and M. Schwager. Controlling noncooperative herds with robotic herders. *IEEE Transactions on Robotics*, 34(2):517–525, April 2018.

[145] Alyssa Pierson and Mac Schwager. Bio-inspired non-cooperative multi-robot herding. In *ICRA*, pages 1843–1849. Citeseer, 2015.

[146] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Navigating congested environments with risk level sets. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[147] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning risk level set parameters from data sets for safer driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 273–280. IEEE, 2019.

[148] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Weighted buffered voronoi cells for distributed semi-cooperative behavior. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5611–5617. IEEE, 2020.

[149] Alyssa Pierson, Cristian-Ioan Vasile, Anshula Gandhi, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Dynamic risk density for autonomous navigation in cluttered environments without object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5807–5814. IEEE, 2019.

[150] Maria Pilecka et al. Combined reformulation of bilevel programming problems. *Schedae Informaticae*, 21:65–79, 2012.

[151] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[152] Robert Platt, Russell Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Robotics Science and Systems Conference (RSS)*, 2010.

[153] Jan Luca Pletzer, Daniel Balliet, Jeff Joireman, D Michael Kuhlman, Sven C Voelpel, and Paul AM Van Lange. Social value orientation, expectations, and cooperation in social dilemmas: A meta-analysis. *European Journal of Personality*, 32(1):62–83, 2018.

[154] Dean A Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

[155] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

[156] Samuel Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

[157] Dicong Qiu, Yibiao Zhao, and Chris L Baker. Latent belief space motion planning under cost, dynamics, and intent uncertainty. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[158] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[159] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[160] Dominic Rushe. Tesla driver who died in 'autopilot' crash was playing on phone, inquiry finds. *The Guardian*, 2020. Available at `www.theguardian.com/technology/2020/feb/25/tesla-driver-autopilot-crash`.

[161] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019.

[162] Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. Information gathering actions over human internal state. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73. IEEE, 2016.

[163] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, volume 2. Ann Arbor, MI, USA, 2016.

[164] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverages effects on human actions. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, June 2016.

[165] Brian Scassellati. Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1):13–24, 2002.

[166] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *International Joint Conference on Neural Networks (IJCNN)*, 1990.

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

[168] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[169] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[170] Wilko Schwarting, Javier Alonso-Mora, Liam Pauli, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935. IEEE, 2017.

[171] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2994–3008, 2017.

[172] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[173] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019.

[174] Wilko Schwarting, Alyssa Pierson, Sertac Karaman, and Daniela Rus. Stochastic dynamic games in belief space. *arXiv preprint arXiv:1909.06963*, 2019.

[175] 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 *Conference on Robot Learning (CoRL)*, 2020. under review.

[176] Tim Seyde, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Learning to plan optimistically: Uncertainty-guided deep exploration via latent model ensembles. *arXiv preprint:2010.14641*, 2020.

[177] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[178] Gregory P Shelley, MJ Page, Peter Rives, Erin Yeagley, and D Michael Kuhlman. Nonverbal communication and detection of individual differences in social value orientation. *Social decision making: Social dilemmas, social values, and ethical judgments*, pages 147–169, 2009.

[179] V. A. Shia, Y. Gao, R. Vasudevan, K. D. Campbell, T. Lin, F. Borrelli, and R. Bajcsy. Semiautonomous vehicular control using driver modeling. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2696–2709, Dec 2014.

[180] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[181] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[182] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[183] Riccardo Spica, Eric Cristofalo, Zijian Wang, Eduardo Montijano, and Mac Schwager. A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Transactions on Robotics*, 2020.

[184] Riccardo Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. A game theoretic approach to autonomous two-player drone racing. *arXiv preprint arXiv:1801.02302*, 2018.

[185] Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Perolat, Karl Tuyls, Remi Munos, and Michael Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[186] Jack Stewart. Why people keep rear-ending self-driving cars. *Wired*, 2018. Available at `https://www.wired.com/story/self-driving-car-crashes-rear-endings-why-charts-statistics/`.

[187] Justin Storms, Kevin Chen, and Dawn Tilbury. A shared control method for obstacle avoidance with mobile robots and its interaction with communication delay. *The International Journal of Robotics Research*, 2017.

[188] Daniel Strömbom, Richard P Mann, Alan M Wilson, Stephen Hailes, A Jennifer Morton, David JT Sumpter, and Andrew J King. Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of The Royal Society Interface*, 11(100):20140719, 2014.

[189] Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Anca D Dragan. Courteous autonomous cars. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 663–670. IEEE, 2018.

[190] Wen Sun, Jur van den Berg, and Ron Alterovitz. Stochastic extended lqr for optimization-based motion planning under uncertainty. *IEEE Transactions on Automation Science and Engineering*, 13(2):437–447, 2016.

[191] Zachary N Sunberg, Christopher J Ho, and Mykel J Kochenderfer. The value of inferring the internal state of traffic participants for autonomous freeway driving. In *2017 American Control Conference (ACC)*, pages 3004–3010. IEEE, 2017.

[192] Richard Sutton. The bitter lesson. *Incomplete Ideas (blog), March*, 13:12, 2019.

[193] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[194] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

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

[196] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics.* MIT press, 2005.

[197] Jonathan Tilley. Automation, robotics, and the factory of the future. McKinsey & Company, 2017.

[198] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306. IEEE, 2005.

[199] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[200] Annemarie Turnwald and Dirk Wollherr. Human-like motion planning based on game theoretic decision making. *International Journal of Social Robotics*, 11(1):151–170, 2019.

[201] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[202] Paul AM Van Lange. The pursuit of joint outcomes and equality in outcomes: An integrative model of social value orientation. *Journal of personality and social psychology*, 77(2):337, 1999.

[203] Paul AM Van Lange, René Bekkers, Theo NM Schuyt, and Mark Van Vugt. From games to giving: Social value orientation predicts donations to noble causes. *Basic and applied social psychology*, 29(4):375–384, 2007.

[204] Paul AM Van Lange, Ellen De Bruin, Wilma Otten, and Jeffrey A Joireman. Development of prosocial, individualistic, and competitive orientations: theory and preliminary evidence. *Journal of personality and social psychology*, 73(4):733, 1997.

[205] Efstathios Velenis, Panagiotis Tsiotras, and Jianbo Lu. Optimality properties and driver input parameterization for trail-braking cornering. *European Journal of Control*, 14(4):308 – 320, 2008.

[206] Anirudh Vemula, Katharina Muelling, and Jean Oh. Modeling cooperative navigation in dense human crowds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1685–1692. IEEE, 2017.

[207] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[208] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[209] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint: 1708.04782*, 2017.

[210] Heinrich Von Stackelberg. *Market structure and equilibrium.* Springer Science & Business Media, 2010.

[211] 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, Mar 2006.

[212] Eric Wan. Sigma-point filters: An overview with applications to integrated navigation and vision assisted control. In *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, pages 201–202. IEEE, 2006.

[213] Zijian Wang, Riccardo Spica, and Mac Schwager. Game Theoretic Motion Planning for Multi-robot Racing. In *International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2019.

[214] Zijian Wang, Tim Taubner, and Mac Schwager. Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments. *Robotics and Autonomous Systems*, 125, 2020.

[215] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[216] Grady Williams, Brian Goldfain, Paul Drews, James M. Rehg, and Evangelos A. Theodorou. Autonomous Racing with AutoRally Vehicles and Differential Games. *arXiv preprint: 1707.04540*, jul 2017.

[217] Grady Williams, Brian Goldfain, Paul Drews, James M Rehg, and Evangelos A Theodorou. Best response model predictive control for agile interactions between autonomous ground vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2403–2410. IEEE, 2018.

[218] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[219] Jeremy Wyatt. Robotics: the next big thing? *Aspenia*, 68-69-70, 2015.

[220] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *International Conference on Machine Learning (ICML)*, 2019.

[221] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

[222] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[223] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009.