TU Delft

AUTONOMOUS MULTI-ROBOTS LAB

Predictability Awareness for Efficient and Robust Multi-Agent Coordination

MSc Thesis

Roman Chiva Gil

Master thesis project· 42 ECTS
Delft University of Technology
Faculty of Aerospace Engineering
Control & Operations
Delft, Netherlands, 2024

Predictability Awareness For Efficient and Robust Multi-Agent Coordination

MSc Thesis

by

Roman Chiva Gil

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on January 14, 2025 at 14:30

Thesis committee:

Chair:	Dr. C. de Wagter
Supervisor:	Dr. G.C.H.E de Croon
Examiner:	Dr. J. Alonso Mora
Additional Members:	Dr. D. Jarne Ornia
	K. Mustafa
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	January, 2024 - January, 2025
Student number:	4811313

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

Faculty of Aerospace Engineering · Delft University of Technology

Preface

This MSc project has been a valuable learning experience and has motivated me to pursue a career in research. I thoroughly enjoyed working on this topic and take pride in the contributions we achieved together with my supervisors.

I am grateful to my supervisors, Professor Javier Alonso Mora and Professor Guido de Croon, for giving me the opportunity to work on this project.

A special and heartfelt thank you goes to my daily supervisors, Daniel Jarne Ornia and Khaled Mustafa. Thank you for your support throughout the project, for all the weekly meetings, and for helping me navigate the tough spots and think through challenges. Working with you has been both a pleasant and inspiring experience, and I look forward to potential collaborations in the future.

Contents

List of Figures

List of Tables

Part I

Literature Study

1 Introduction

1.1. Background

The ability of robots to effectively navigate unstructured environments, populated by other robots and human agents, without relying on explicit communication or centralized control, represents a significant technological advancement with far-reaching applications and societal impact [1]. In pedestrian navigation, for instance, autonomous agents could improve the safety and efficiency of urban mobility, enabling robots to assist in crowded spaces like shopping malls, airports, or public squares [2]. In the domain of autonomous vehicles, these systems have the potential to revolutionize transportation by reducing human error, optimizing traffic flow, and improving road safety. From a logistics point of view, especially autonomous trucking, could lead to significant cost savings and increased productivity across the sector. Furthermore, in warehouses and industrial settings, robots that navigate and coordinate with other autonomous systems and human workers could streamline logistics, increase productivity, and reduce operational costs. However, for autonomous agents to realize this potential, they must be highly reliable, which presents a substantial challenge. By introducing communication or a form of centralized control, it is possible to reduce the complexity of the task by minimizing uncertainty about other agents' behaviors and intentions. For instance, systems that rely on central coordination or communication protocols have seen successful applications in warehouses. However, the infrastructure required to support communication or centralized control is not always reliable, readily available or affordable making it difficult to implement such solutions on a large scale. Scaling requires decentralized autonomous agents capable of reasoning about their environment, other agents and the uncertainty inherent in multi-agent interaction.

While humans navigate complex environments seemingly effortlessly, we often underestimate the intricacies involved. These scenarios often combine structured rules with a layer of unpredictability, driven by the stochastic nature of agents and the interactions between them. Many modern decentralized systems achieve safe and efficient navigation in a majority of scenarios encountered during operation. However, there exists a small set of edge cases that are inherently unpredictable and challenging to account for during system design. Developing systems that are robust to these edge cases and ensure consistently flawless performance is an exceedingly difficult task[3].

1.2. Current Practical Applications

Although there are many challenges to overcome, some practical applications in controlled environments as well as some more experimental applications in general environments are in use today. In this section, we will explore the current applications of autonomous navigation in three key multi-agent environments: autonomous vehicles, pedestrian environments, and industrial settings.

Autonomous Road Vehicles

Autonomous Road vehicles (AVs) are one of the most promising applications, ranging from private personal transportation to delivery trucks and public transport. These vehicles are classified in a spectrum of autonomy, as defined by the SAE International classification[4], from Level 0 (no automation) to Level 5 (full automation). Most consumer available vehicles, as of today, fall within Level 2 and Level 3. They can perform certain tasks like lane-keeping or adaptive cruise control but still require human intervention in complex scenarios (Often auto-pilot is only available for structured and predictable highway scenarios

Figure 1.1: Waymo autonomous vehicle

Figure 1.2: SoftBank Pepper airport guide robot

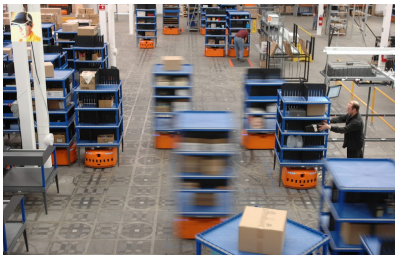


Figure 1.3: Amazon Kiva warehouse automation robot

Figure 1.4: Overall caption for the three figures

or sub-urban neighborhoods). Examples of this type of autonomy include Tesla's Autopilot and General Motors Super Cruise systems. More advanced, Level 4 AVs are being tested in controlled environments, such as urban centers constrained by geo-fenced areas, with companies like Waymo in the US and WeRide in China offering autonomous taxi services in some cities. While these vehicles can operate without human input in complex urban environments and interact with human drivers, they still need human intervention when encountering edge cases. The ultimate goal is Level 5 autonomy, where vehicles can fully navigate any environment without human assistance, though this remains in the research and testing phase, and requires autonomy stacks capable of reasoning beyond replicating learned behavior. Autonomous delivery robots and long-haul trucks are also advancing, with companies like Nuro and TuSimple deploying Level 4 already performing preliminary experiments. While full autonomy still faces a plethora of regulatory and technical challenges, the steady deployment of Level 4 AVs is a promising step forward that can facilitate level 5 in the long run.

Navigating among Pedestrians

Autonomous ground robots can be increasingly found in pedestrian environments such as shopping malls, airports, or hospitals for example. SoftBank's Pepper and Nao serve as guides and assistants, providing customer service and navigating through crowds without relying on explicit communication or coordination from pedestrians. These robots have been used in places like airports to guide tourists in need of assistance *(*link). Autonomous security robots, such as Knightscope K5, patrol public spaces, using sensors to avoid obstacles while monitoring for security threats. Delivery robots, such as those developed by Starship Technologies and Amazon Scout, operate on sidewalks to deliver goods, navigating through urban environments filled with pedestrians. In healthcare settings, Aethon's TUG robots autonomously deliver medications and supplies through hospital hallways, avoiding patients and staff. Although they have found some success in practice, it can be argued that their navigation approach is not interactive with pedestrians, but merely reactive and treat pedestrians as dynamic obstacles rather than interactive agents, in occasion leading to clumsy coordination, inefficient paths or safety risks. Although there is still much progress to be made, these applications are a promising step and lay the groundwork and practical experience for more advanced future applications.

An aspect that is often overlooked, is that humans also need to get used to having robots in their environment, as otherwise humans will be curious or surprised and might react differently than expected (for example, stopping to look or following the robot [5]). This is very challenging to account for when designing autonomy stacks for these agents[2].

Industrial Applications

In industrial environments, autonomous robots already play a key role in streamlining operations. These environments, by virtue of being more controlled, offer improved predictability, making it easier to design autonomous agents to operate in them. Amazon's fulfillment centers utilize Kiva robots, which autonomously organize goods across the warehouse while coordinating with human workers. Similarly, Ocado's (Online supermarket in the UK) automated warehouses use hundreds of robots to pick and pack groceries for delivery, with each robot locally sensing its environment and reasoning to avoid collisions and efficiently complete its tasks. Such autonomous systems enhance productivity and scalability for warehouse logistics,

however this is only made possible due to the low-uncertainty environment they operate in, which allows for the development of a highly fine-tuned autonomy stack. In fact it is not uncommon to redesign a warehouse to accommodate such decentralized autonomous robots. While productive already, in the future it would be beneficial to design systems that can generalize better to different operational conditions.

1.3. Overview of Challenges

As discussed previously, it is becoming increasingly common for autonomous agents to navigate multi-agent environments where they are required to interact with both humans and other robots without full knowledge or extensive communication capabilities, which presents significant challenges. Such environments are complex systems characterized by numerous interacting entities, each with distinct and sometimes conflicting objectives, and often influenced by stochastic or unpredictable internal and external influences. This complexity makes it difficult for autonomous agents to gather the necessary information for effective coordination, as agents may have limited access to others' objectives, intentions, or even the full range of their possible behaviors. With this limited information and the inherent uncertainty an agent must reason about others to navigate. While this comes second nature to humans, in the form of Theory of Mind [3], designing an autonomy stack capable of replicating this reasoning is a formidable technical challenge.

Additionally, this resulting uncertainty induced by partial observability and the agents' varying levels of rationality, makes it challenging for any single agent to accurately anticipate how others might respond to its actions. The key characteristic of multi-agent systems is the fact that agents' trajectories are highly interdependent; the movement or decision of one agent can directly influence others, creating a web of dependencies that grows exponentially with the number of agents. Consequently, planning optimal trajectories in such an environment becomes computationally expensive, as each agent must consider not only its own goals but also reason about the potential actions and reactions of all others with the complexity being further exacerbated by the large uncertainty. This need for extensive reasoning over multiple potential uncertain future scenarios makes real-time, efficient planning a formidable modeling and computational task.

Challenges not addressed in this survey:
Autonomous agents encounter additional operational challenges critical for reliable deployment, but not specific to multi-agent interactions. One such area is perception, since a robust perception forms the foundation for understanding the environment and gathering information on surrounding agents. Accurate and real-time perception is essential to allow agents to observe and assess the actions of others. This area has made significant progress in recent years and often relies on a mix of LIDAR and cameras. With LIDAR becoming more affordable it has become an increasingly popular choice. Often perception data generated by these sensors is returned as a point-cloud. Neural networks are commonly used to generate representations that can be useful for prediction and planning. It is not uncommon for these representations to include uncertainty, however for the remainder of this survey, the works discussed assume a 'perfect' perception system, a common assumption in many of the publications discussed [6].

Another important challenge is System Identification and Control: successfully executing planned trajectories requires an accurate model of the robot and environment and precise control, yet uncertainties in environment dynamics can complicate robustly tracking the planned trajectory. Often, simplified dynamic models are used to compute feasible paths in the planning stage, but real-world dynamics are far more complex. This gap between model assumptions and real behavior can hinder precise trajectory tracking, making robust control another key area for reliable autonomous operation. Similar to the above assumption, the rest of the works in the survey assume a perfect control system. These assumptions are made to allow focusing on the prediction and planning challenges discussed above.

1.4. Autonomy Stack Architecture Overview

For autonomous driving systems, End-to-End (E2E) architectures rely on a single neural network for processing raw perception inputs to directly generate driving commands [7]. This architecture minimizes data loss across stages and can learn complex abstracted representations that account for many environmental factors that would be impossible to explicitly account for otherwise. Potentially, this approach could mimic human driver data perfectly given enough training data and a good enough network design [6]. By avoiding hand-crafted intermediate representations and introducing very minimal prior knowledge,

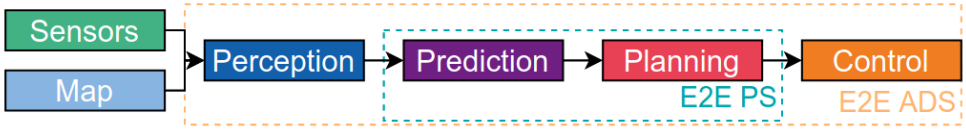


Figure 1.5: Figure from [6] illustrating a modular architecture, an End-To-End (E2E) and End-To-End Planning System (E2E-PS).

it makes no assumptions about how the available information is weighted in planning. For example, it has been reported that systems training this way are able to make use of reflections on mirrors of other vehicles or on the road to anticipate pedestrians or other vehicles not in line of sight [7]. While holding great potential, this approach also presents significant challenges in interpretability and sample efficiency. The lack of clear, interpretable intermediate stages makes it difficult to understand or control specific behaviors, limiting insight into the network's decision-making processes making the system hard to verify for safety in practical applications. Additionally training the system this way is extremely data intensive and requires vast datasets for effective training, thus limiting what applications such an approach can be leveraged for (Currently only Autonomous Road Vehicles, as it is the only application for which there is remotely enough training data). Some works have also investigated a simpler E2E planning system E2E-PS, that only combines prediction and planning into one layer. These approaches are significantly less data intensive while still allowing for flexible interaction modeling without assumptions, however similarly they still face challenges in interpretability, making their application in practical scenarios impractical as they cannot be explicitly verified for the required robustness.

Interpretable End-to-End models aim to bridge the gap between direct E2E approaches and traditional modular systems by including intermediate, interpretable representations within the network. These models allow developers to analyze internal decision points, improving both training stability and generalization. While these representations are not used by subsequent network layers, they act as auxiliary tasks, providing additional structure to stabilize training (the extra losses at the intermediate stages help in stabilizing training and converging faster) and insight into the model's functionality. However, these intermediate outputs introduce complexity in balancing performance and interpretability, as the network must trade off between preserving information for deeper learning layers and maintaining transparency at each step.

In contrast, Modular architectures separate prediction and planning into sequential tasks, where predictions about surrounding agents are computed first and then used to inform the ego-vehicle's trajectory planning. This architecture is intuitive and straightforward to implement, as it allows for development of the modules in parallel and simplifies the decision-making process by treating the surrounding agents' actions as fixed inputs for planning. However, it results in a reactive system that does not account for how the ego-vehicle's actions may influence others. The most significant difference with E2E approaches is that the behavior of the agent is usually the product of an optimization, with the desired behavior expressed through a cost function. This means the agent is less adaptive and cannot reason about the environment as flexibly. As a result, these systems may exhibit overly cautious or uncoordinated behaviors, especially in dynamic multi-agent environments where modeling mutual influence is essential.

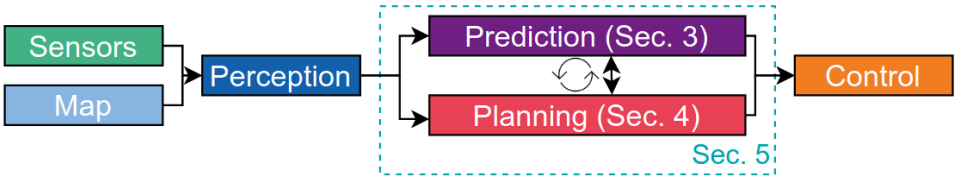


Figure 1.6: Sequential Architecture with integrated prediction and planning

To address this issue, modular architectures with integrated prediction and planning combine these stages into a mutually informed process, allowing the two stages to influence each other, as seen in figure 1.6. This approach allows the system to model interdependencies between agents, capturing how the ego-vehicle's

actions can influence, and be influenced by other agents. In this case, some form of behavioral model is used for other agents, and the ego-agent solves a coupled optimization problem over a dynamical system of interacting agents to plan its trajectory. A drawback of this approach however is that it is very sensitive to the agent models used as well as having exponential computational complexity with the number of agents.

The choice of architecture is a trade-off where choosing the most suitable architecture depends on the application's requirements: balancing for real-time performance, interpretability. safety and the degree of complexity of interactions with surrounding agents. In structured and predictable environments, a sequential modular approach is suitable. The challenge however, comes when aiming to navigate unstructured, unpredictable and densely populated environments, where it is necessary to directly account for interactions among agents. In this case it is necessary to resort to E2E and integrated architectures and address the challenges in safety and computational complexity that come with these approaches.

1.5. Scope and Structure of the Survey:

The sections above gave an overview of the current state and a diversity of approaches relating to autonomous navigation in multi-agent environments. Several architectural approaches have been presented, ranging from full E2E to modular architectures where each sub-task is treated as a separate module. Within modular architectures, an open challenge remains: effectively modeling interactions between agents during planning. The sequential predict-and-plan approach separates these two tasks, with agents planning under the assumption that their predictions about other agents remain fixed. In contrast, integrated approaches merge prediction and planning into a joint task, allowing for the direct modeling of agent interactions. However, as discussed, this comes with challenges related to scalability and accurately modeling the behaviors of other agents in response to the ego.

This survey will focus on modular approaches, discussing the state-of-the-art methods for both sequential predict-and-plan and joint prediction and planning, exploring their advantages and disadvantages, and the scenarios for which they are most suitable. Additionally, it will cover related topics such as observer-aware planning, with considerations for the legibility and predictability of an agent's trajectories and prediction models used to model surrounding agent's behaviors. The following bullet-point list provides a brief summary describing what topics are covered in each section of the survey:

- **Chapter 2: Receding Horizon Trajectory Optimization for Mobile Robots,** This chapter introduces Receding Horizon Trajectory Optimization (RHTO) as one of the most popular planning paradigms used in the planning module of autonomy stacks. Trajectories are generated in a Model Predictive manner, optimizing a sequence of actions up to a defined horizon and executing the first action in a repeated manner. The chapter identifies Gradient-Based Methods and Sampling-Based Methods as a core taxonomic distinction and discusses relevant literature and the advantages and disadvantages of each approach.
- **Chapter 3: Interactive Planning,** This chapter discusses several methodologies presented in literature to account for multi-agent interaction in planning. The section is structured around the taxonomic division between joint methods and sequential predict+plan methods. Joint methods solve a multi-agent problem in which other agents are explicitly modeled, whereas sequential predict+plan solve a single agent problem and implicitly account for interaction. The state of the art literature for both approaches is discussed together with their advantages and disadvantages.
- **Chapter 4: Legibility and Predictability of Robot Motion,** The chapter discusses works in observer-aware planning particularly focusing on the concepts of legibility and predictability introduced by [8]. The original definitions provided have been seminal in this field and have become the standard terminology used. These definitions are discussed in detail together with their advantages and disadvantages. Extensions of these concepts for interactive and receding horizon approaches are additionally discussed.
- **Chapter 5: Multi-Agent Trajectory Prediction,** Vast amounts of literature on multi-agent motion prediction have been published in recent years. Following the taxonomy presented by [9] the section discusses the different approaches and advances in the field in recent years as well as the advantages and disadvantages of each approach.
- **Chapter 6: Research Objective,** The report concludes by giving motivation for the research project this survey is a part of as well as identifying the key research questions this project aims to address.

2

Receding Horizon Trajectory Optimization for Mobile Robots

2.1. Background and Problem Definition

Model Predictive Control(MPC) is a widely-used control framework for mobile robots. It can handle complex dynamics, uncertainty and respect explicitly defined constraints. At each time step, the control actions are computed by solving an optimization problem over a prediction horizon N. The state of the system is propagated with a dynamics model of the robot, and the optimal control inputs are selected to minimize a cost function subject to constraints. The first control action is applied, and the optimization process is repeated at the next time-step. This allows for flexible trajectories even in complex environments[10].

The core components of the framework are a dynamics model, a cost-function J and a set of constraints. The objective is to minimize the cost function J over the prediction horizon N. In general the cost function is a function of both states and actions. A common cost function definition can be seen in equation 2.1:

$$J(x(t), u(t)) = \sum_{k=0}^{N-1} J_S(x_k, u_k) + J_N(x_N) \tag{2.1}$$

In this equation, x_k and u_k are the predicted state of the system and control action at horizon timestep k. $J_S(x_k, u_k)$ is a stage cost that defines control objectives and penalizes excessive control effort. $J_N(x_N)$ is the terminal cost d to avoid undesirable end states. For example, in the context of navigation, this could be used to penalize high velocities toward the end of the plan, to avoid the robot not having time to break if a wall is encountered at the next time-step.

The states visited by the robot are predicted with a dynamics model, which describes how the state evolves given the control inputs. The dynamics are usually expressed in shorthand by a function $f(x, u)$ as seen in equation 2.2:

$$x_{k+1} = f(x_k, u_k) \tag{2.2}$$

These dynamics are often non-linear for mobile robots [11]. Common examples include the popular ”bicycle model” and ”unicycle model” used to approximate the behavior of ground vehicles. This introduces additional complexities into the optimization problem.

A necessary part of the framework is to introduce a set of constraints in the optimization that represent the physical limitations of the system and collision avoidance.

In summary, each time step the following optimization is solved:

$$
\begin{aligned}
\min_{u_0,\dots,u_{N-1}} \quad & J(x(t),u(t)) = \sum_{k=0}^{N-1} J_S(x_k,u_k) + J_N(x_N) \\
\text{subject to} \quad & x_{k+1} = f(x_k,u_k), \quad k = 0,\dots,N-1 \\
& x_0 = x_{\text{init}} \\
& x_k \in \mathcal{X}, \quad k = 0,\dots,N \\
& u_k \in \mathcal{U}, \quad k = 0,\dots,N-1 \\
& g(x_k,u_k) \leq 0, \quad k = 0,\dots,N-1 \\
& h(x_k,u_k) = 0, \quad k = 0,\dots,N-1
\end{aligned}
\tag{2.3}
$$

The initial condition ensures that the problem starts from the initial state $x_0 = x_{\text{init}}$. All states must belong to the feasible set of states $x_k \in \mathcal{X}$. All control actions must belong to the available set of control actions $u_k \in \mathcal{U}$. Inequality constraints $g(x_k,u_k) \leq 0, k = 0,\dots,N-1$, represent constraints such as obstacle avoidance, input bounds, or state limits. In the definition constraints are often grouped into equality and inequality constraints. Equality constraints, $h(x_k,u_k) = 0, k = 0,\dots,N-1$ are also commonly used to represent the state-transitions and conditions such as path requirements or other problem-specific requirements.

2.1.1. Common Challenges in MPC for Mobile Robots

MPC provides a flexible and robust framework for mobile robot control, however several challenges are commonly encountered its application to mobile robots. This section provides a high-level overview of the most relevant challenges in applications specifically for mobile robots. These are non-linear dynamics and a rapidly changing possibly unpredictable environment.

Nonlinear Dynamics

Many mobile robots have non-holonomic an non-linear dynamics. As a result, solving the optimization problem becomes more computationally expensive, and approximations or linearization techniques may be required to operate in real-time. For example, in a robot described by the bicycle model, the robot's dynamics are governed by equation 2.4:

$$
\begin{aligned}
\dot{x} &= v\cos(\theta), \\
\dot{y} &= v\sin(\theta), \\
\dot{\theta} &= \frac{v}{L}\tan(\delta),
\end{aligned}
\tag{2.4}
$$

(x,y) is the position of the robot, θ is the orientation, v is the velocity, δ is the steering angle, and L is the distance between wheels. It is possible to linearize the dynamics, however, this may not always be possible, as it can severely deteriorate performance in some applications.

Instead of linearizing the dynamics, Non-Linear MPC (NMPC) directly handles the non-linearities in the system. Although it is computationally more intensive, advances in optimization methods and hardware allow for real-time NMPC in many applications [12]. For example, Sequential Quadratic Programming (SQP) can handle non-linear dynamics and constraints efficiently. NMPC is a very common approach for mobile robots.

Uncertainty in the Environment

Mobile robots often navigate an unpredictable and rapidly changing environment. For example, when navigating a corridor with pedestrians the robot faces a rapidly changing environment with significant uncertainty about the future states of surrounding agents. Accounting for uncertainty introduces new challenges for the optimization. The two main approaches to handle uncertainty in MPC are Robust MPC and Stochastic MPC, both of which offer distinct advantages and trade-offs:

Robust MPC is designed to ensure that the system operates safely by assuming worst-case scenario outcomes of the uncertainty. It aims to find a solution where the system is still guaranteed to perform safely even when the most disadvantageous uncertainty outcomes . It is essentially entirely risk averse.

By preparing for the most adverse situations, the Robust MPC approach guarantees constraint satisfaction and safety, however, fully risk averse strategies come at the cost of performance, as the robot may be overly cautious, leading to suboptimal trajectories when the worst-case scenario does not occur. A common problem highlighted in literature with this approach is the freezing robot problem [13]. Due to large uncertainties, the robot deems all possible actions as unsafe an is unable to make progress toward its goal.

Stochastic MPC takes a probabilistic approach to uncertainty, allowing the robot to optimize given a distribution of future scenarios. This approach results in a more flexible and less conservative controller compared to Robust MPC as it allows for modelling how much risk the robot will take. For example for collision avoidance with dynamic agents in the environment, introducing chance-constraints in a common approach [11], however, it introduces additional complexity in the optimization, and often requires the uncertainty distribution to be Gaussian. Stochastic MPC does not provide strict guarantees of safety, instead the level of risk is a designer parameter. Modelling this risk and appropriate probability thresholds is an active research field.

Both approaches offer tools for handling uncertainties in dynamic environments, but the choice between them depends on the specific application requirements. In highly safety-critical systems, where failures are catastrophic, Robust MPC may be preferred. On the other hand, Stochastic MPC can provide better performance in environments where probabilistic outcomes are acceptable, offering a balance between performance and risk. In practice, hybrid methods that combine elements of both Robust and Stochastic MPC can be used to strike a balance between safety and efficiency. Using adaptive risk metrics such as CVaR is also a possibility.

2.2. Sampling vs Gradient Based MPC For Mobile Robot Trajectories

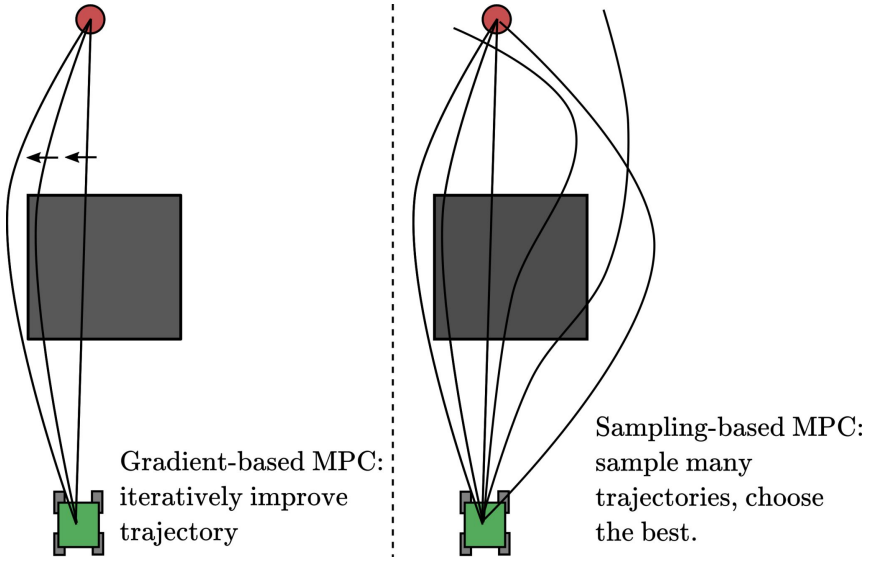


Figure 2.1: Illustration of the different optimization approaches for MPC. The choice between them depends on the specifics of the control problem being considered. (Credit: Petar Velchev TU Delft MSc LinkedIn Post)

In mobile robot navigation, modern optimization methods can broadly be classified into sampling-based and gradient-based approaches, each offering distinct advantages depending on the task at hand. In recent years, sampling based approaches have gained a lot of popularity due to advancements in parallel computing and GPU capabilities. This allows for parallel processing of samples, enabling real-time applications of sampling based optimization for more complex tasks.

Gradient-based methods continuously compute gradients to optimize control inputs, making them highly efficient for smooth, differentiable cost functions and environments with known dynamics. Gradient-based approaches are typically faster for real-time tasks, but they may struggle in non-smooth or obstacle-dense environments. As many works discuss [14], free-space in pedestrian dense environments is highly

non-convex, making it likely for the robot to become stuck in local minima with gradient based methods.

On the other hand, Sampling-based methods optimize by randomly sampling possible trajectories or control inputs, thus they are better able to explore the solution space. This makes them particularly effective in complex, high-dimensional environments where gradients may be difficult to compute or non-differentiable. These methods excel at finding feasible paths in highly cluttered dynamic environments but can be computationally expensive due to the need for extensive exploration and sample evaluation [15].

Both methods offer good solutions each with unique advantages. Gradient-based optimization has better convergence properties and better constraint satisfaction. Sampling-based optimization is able to find better local minima and even find global optima. Practically, sampling-based is easier to implement and can seamlessly handle with discontinuous dynamics. A comparison is provided in table 2.1. There is no silver bullet, and often the choice between both is application dependent. In the following subsections relevant works on gradient and sampling based MPC for mobile robots are discussed.

Aspect	Sampling-Based MPC Approaches	Gradient-Based MPC Approaches
Solution Methodology	Samples a wide range of feasible states/actions and constructs trajectories through simulation. Each sample is evaluated with a cost function and the best sample selected.	Utilizes gradient information for iterative refinement of trajectories, minimizing a cost function. Suitable for differentiable environments with well-defined dynamics.
Convergence	Can guarantee a feasible path if enough samples are used but might not always produce feasible or locally optimal solutions. Relies on sampling density for better convergence to optima.	Converges to a local minimum defined by the initial conditions. Ensures smoother and optimal solutions but risks being trapped in local optima, especially in non-convex spaces such as pedestrian environments.
Handling Non-Convex Constraints	Robust to non-convex constraints as the optimization procedure does not depend on convexity or differentiability of the dynamics or cost function	Struggles with highly non-convex constraints. Efficiency drops, taking longer to converge to optimal solutions. Requires differentiability of system dynamics and convexity of cost function.
Computational Efficiency	Computationally intensive, especially in high-dimensional spaces. Struggles with real-time constraints due to the need for numerous samples. Efficiency can improve with parallel processing.	More computationally efficient under structured, differentiable environments. The complexity rises with non-linear or dynamic interactions but generally faster for smooth, real-time applications.
Exploration vs. Exploitation	Good for exploring a broad state space to find potential global optima but at the cost of higher computation. Can achieve better exploration outcomes.	Tends to exploit known paths for local optimization, which may lead to suboptimal global solutions. Efficient for fine-tuning existing paths rather than exploring new ones.
Use Cases	Ideal for dynamic, high-density environments like busy public spaces where agent interactions are highly varied and unpredictable.	Best suited for structured environments such as indoor navigation or road-following with moderate complexity and predictable pedestrian flow.

Table 2.1: Comparison of Sampling-Based and Gradient-Based MPC Methods for Mobile Robot Planning

2.3. Gradient-Based Methods

The work of [12] is widely cited and provides an illustrative outline of how gradient based methods can be efficiently used in mobile robots in dynamic uncertain environments. Their method is based on Model Predictive Contouring Control (MPCC) and extension of MPC for path following tasks. Their main contribution is to approach the problem as a reference tracking task, as opposed to finding a path towards the goal, the robot is tasked with following a pre-computed trajectory generated by a global planner beforehand,

while locally optimizing the trajectory to minimize tracking error while avoiding obstacles. A illustration of this process can be found in figure 2.2. The reference contains both position and time information, giving the robot a better understanding on whether it should wait behind or overtake a pedestrian depending on its velocity for example. It models static obstacles with convex polyhedra and other agents in the environment as ellipsoids, making the system differentiable and suitable for gradient-based optimization. Other agents future trajectories are predicted with a Constant Velocity (CV) Model. It was tested in both simulation and real environments, with results showing it outperforms other planners like classical MPC in handling complex, dynamic scenarios, demonstrating efficient, real-time performance while avoiding the ”freezing robot problem” discussed by [13].

Figure 2.2: Illustration of MPCC reference tracking methodology. The green line represents the reference trajectory. In order to avoid a pedestrian it deviated from the reference but returns as soon as the pedestrian has been overtaken. This behavior is illustrated by the purple line representing the planned trajectory.

A novel approach by [14] introduces Topology-MPC (T-MPC) a framework designed to improve the exploration capabilities of gradient based planners. Their motivation is that current Non-Convex optimizers used in NMPC methods often converge to locally optimal solutions and frequently switch between different local minima, leading to inefficient and unsafe robot motion. To address this they propose a novel topology-driven trajectory optimization that plans multiple distinct evasive trajectories. A global planner iteratively generates trajectories in distinct homotopy classes. Different homotopy classes are illustrated in figure 3.10. These trajectories are then optimized by local planners working in parallel. An illustration of this process can be found in figure 2.3. While each planner shares the same navigation objectives, they are locally constrained to a specific homotopy class, meaning each local planner attempts a different evasive maneuver. The robot then executes the feasible trajectory with the lowest cost in a receding horizon manner. Results demonstrated that for a mobile robot navigating among pedestrians, this our approach leads to faster and safer trajectories than other state-of-the-art gradient based planners.

2.4. Sampling-Based Methods

There exists a diverse set of sampling-based optimization approaches for RHTO. A core distinction with the methods however has to do with whether they sample in the state-space or in action space. For methods that generate trajectories in state-space, they are usually combined with a lower-level planner tasked with generating an action sequence to track the trajectory (A gradient based MPC may be employed to track this trajectory, however this is not considered planning with a gradient based method as above, since the trajectory is already fixed, its not a planning task but rather a tracking task). Another important distinction has to do with how the samples are constructed. Some methods rely on a fixed set of samples or fixed methodology to generate the samples, such as a set of motion primitives for example. Other approaches follow a more reactive methodology where the geometries of the generated samples are not pre-defined, allowing for more flexible trajectories that react to the environment, such as Model Predictive Path Integral for example.

In [16], the authors propose a semi-reactive approach to sample generation. The proposed method

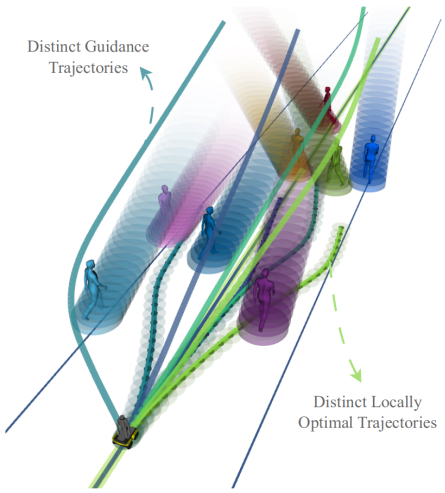
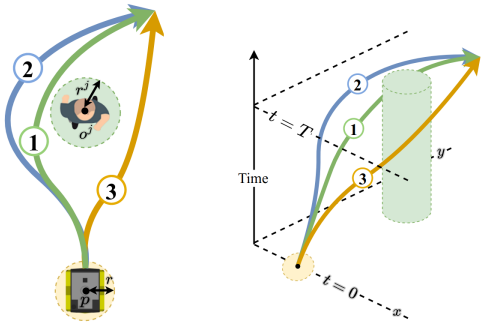


Figure 2.3: In T-MPC, a global planner finds topologically distinct global trajectories. A set of local planners then optimize the trajectories for kinematic feasibility while being constrained to different homotopy classes. This results in a set of diverse potential plans leading to better exploration of the possible solution space.



(a) Planning scene. (b) State space view.

Figure 2.4: Trajectory 1 and 2 are in the same homotopy class. Trajectories 1 and 3 are in different homotopy classes.

operates in Frenet frame, a moving reference frame that follows the curvature of the global reference path. Taking inputs from the global reference path such as target velocity, over-taking decisions and the global path to follow. By operating in Frenet-frame it is possible to then separate between lateral and longitudinal motion, which allows to sample a set of desired longitudinal points and lateral points separately and then combined using a quintic polynomial to fit the end point while assuring the smoothest trajectory possible. The key benefit of this method, is that by sampling trajectories in Frenet frame and assuring smoothness, it is possible to make efficient use of the samples to explore the solution space, resulting in a reliable and lightweight method.

Another common semi-reactive state-based sampling planner is sometimes referred to as a fan planner. This style of planner has been used for planning in differentiable End-to-end sequential autonomous vehicle (AV) stacks [17] [18]. It samples a set of lane centric candidate states and fits a cubic spline. Once a set of candidate trajectories is generated they are fitted with a cubic spline. Dynamically unfeasible trajectories are then discarded. The difference with the previous method is that this approach operates in cartesian space, and has the benefit of being end-to-end differentiable. They make the planner differentiable by relaxing selecting the best sample into sampling from a categorical distribution effectively treating the planner as a classifier during training.

Monte Carlo Tree Search (MCTS) can be used for vehicle trajectory planning by exploring action sequences (e.g., steering, acceleration) in a tree structure, simulating the outcomes of each action through random sampling, and selecting the most promising trajectory based on a cost function and environment simulation [19]. However they require the environment and robot dynamics to be discretized. This approach is particularly well suited for planning while considering the responses of other agents during the forward simulation step. However a disadvantage is that MCTS can become computationally expensive if the action space is very large or the horizon is long. However, with the advancement of parallel GPU computing these limitations become less constraining to the application.

In [20] while employing an approach similar to MCTS the authors deviate from the tree-based sampling approach by using an Exhaustive search methodology instead. They simplify the robot's action space by discretizing and combining several time-steps into each action. They use 8 primitives the robot can craft its trajectories from, picking one at each sampling stage. The response of surrounding drivers is also computed for each selected primitive sequentially. While being constrained to more structured highway scenarios, where the types of trajectories the robot might pursue are less diverse, this allows them to

consider all possible responses, and thereby are able to find the global optimal trajectory in a sampling based manner that considers the responses of other agents.

A very popular sampling based approach is Model Predictive Path Integral Control (MPPI) [21] [15]. With the development of parallel GPU computing MPPI has become a very popular approach with many works developing this approach in recent years. Since MPPI is central to the development of this project the methodology will be explained in more detail as well as relevant works extending the methodology.

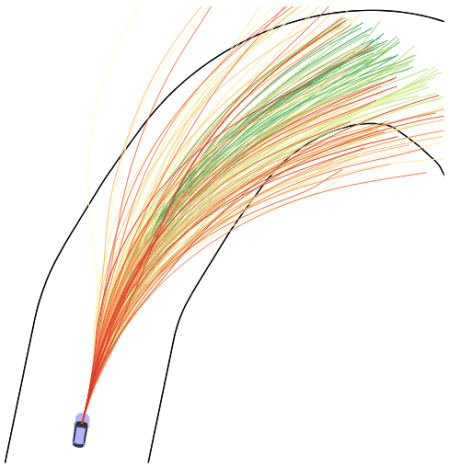


Figure 2.5: MPPI generates samples by applying Gaussian noise to a nominal control sequence. Usually the previously computed control sequence is used as nominal trajectory. The samples are then combined via importance sampling and a weighted average. Figure credit [22]

MPPI generates a set of control trajectories by applying small perturbations to a nominal control input, usually it is bootstrapped, and the previously found optimal sequence of inputs is sampled around to generate new samples. These perturbations are sampled from a Gaussian distribution. In essence the algorithm explores a distribution of control actions around the current best guess. An illustration of this process can be found in figure 2.5. As opposed to simply selecting the best sample like other sampling based methods, the core component of MPPI is the importance sampling step, which allows it to approximate the optimal at each step from a distribution by using a weighted average of the sampled trajectories. For each sample $\mathbf{u}_i$, the corresponding cost J_i is calculated(samples that violate constraints are discarded) to find its weight ω_i as per equation 2.5:

$$\omega_i = e^{-\frac{1}{\lambda}(J_i - J_{min})} \tag{2.5}$$

Where J_{min} is the lowest cost trajectory. The way the weights are crafted resembles a soft-max or Boltzmann distribution, with parameter λ as a temperature parameter. This parameter controls the width of the weight distribution. For lower temperature the distribution of weights becomes more uneven, with the weight less cocnentrated around the lowest cost trajectory. The next control sequence $\mathbf{u}$ is then computed by taking a weighted average of the sampled trajectories. This is computed as per equation 2.6:

$$\mathbf{u} = \sum_{i=1}^{N} \omega_i \mathbf{u}_i \tag{2.6}$$

When updating this way, the best-performing trajectories dominate the update, leading to an improved control action. This iterative refinement process accumulates information over successive timesteps, thus adapts to new conditions robustly by iteratively improving the trajectory with each new iteration while slowly converging closer to the optimal.

The key advantage of MPPI over other sampling based approaches is that the introduction of random noise in the control around the previous best control input makes MPPI highly flexible wile remaining efficient to compute [21]. This randomness allows it to explore a wide range of possible control actions while also focusing on refining the best options. This is a designer selected trade-off, with the level of exploration versus refinement controlled by the variance of the Gaussian noise and the scaling parameter λ, however usually MPPI can balance exploration and exploitation. Due to these advantages, it is widely used for dynamic uncertain environments, where the best control strategy shifts over time and is dependent on uncertainty distributions.

Several recent works have focused on extensions to the MPPI framework. Some works investigated how to make MPPI more robust to modelling errors in the robot model and the environment, to increase robustness in highly uncertain environments. Other extensions focused on improving the sampling distribution to generate more efficient sampling and better exploration.

Tube-MPPI [23] extends standard MPPI by allowing to define a safe corridor, referred to as a 'tube' the robot will not deviate from. The 'tube' accounts for model uncertainties and disturbances. The method ensures that the system remains within this tube robustly under external perturbations. It combines MPPI's sampling-based optimization with feedback control to keep the system within safe bounds during execution. The feedback controller acts as a corrective layer that pushes the robot back when it begins to drift. Covariance Steering MPPI [24] extends MPPI to control both the trajectory and the state covariance. It optimizes control inputs to not only minimize trajectory costs but also shape the uncertainty distribution of the system over time.

A problem with MPPI is that it is more prone to get stuck in local optima than other sampling based controllers that maintain a constant sample diversity. The better optima found by MPPI come at this cost as per the exploration-exploitation tradeoff when choosing λ. To address this shortcoming [25] proposes Biased-MPPI, where instead of constructing a sampling distribution from only the nominal control distribution like regular MPPI, inputs from ancilliary controllers (e.g., classical, learning-based, or task-specific controllers)are also added to the distribution. In practice, this results in faster convergence to optima and better reactivity to the environment. To illustrate these benefits figure 2.6 is provided. The ancilliary controllers included react faster than regular MPPI in this case and provide samples that avoid the collision. These samples are weighted much heavier in the importance sampling step, pushing the overall distribution in that direction. This manages to react in time and avoid collision with the box being pushed in front of the robot. Regular MPPI cannot react fast enough and thus crashes into the box.

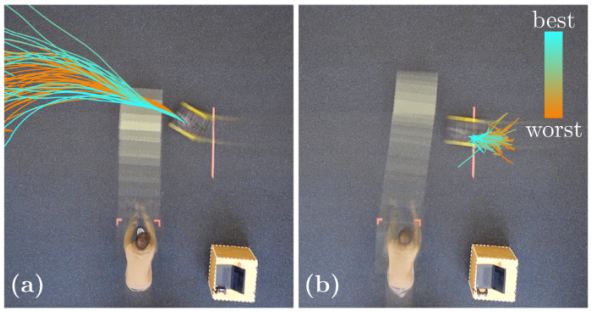


Figure 2.6: Biased-MPPI includes the input from ancilliary controllers to the sampling distribution of regular MPPI. This results in faster convergence to optima allowing for better reaction to the environment. Visualized are the top 50 sampled trajectories, color-graded by their cost. (a) Classic MPPI is about to crash. (b) Biased-MPPI avoids collision. Figure credit from [25]

Another common problem with MPPI based approaches is sample efficiency. MPPI can be quite compute intensive, especially if the robot model is complex or the cost function is expensive to evaluate. Often many of the samples generated are 'wasted' by being uninformative when very closely clustered togetehr, resulting in collisions or other inefficiencies. To address this shortcoming, [22] propose Model Predictive Optimized Path Integral strategies (MPOPI). In MPOPI , the entire control sequence is modeled as a single joint distribution $\mathbf{U} = [u_0, u_1, ..., u_{T-1}]$. The covariance matrix Σ represents both the variance of each individual time step and the covariances between control inputs across different time steps. This

allows for considering the correlations between different horizon time steps. The noise used in this case is a multi-variate Gaussian distribution:$\mathbf{U} \sim \mathcal{N}(\mu, \Sigma)$. μ is the nominal input and Σ is the covariance matrix over the entire time horizon. The covariance matrix Σ is structured as a block matrix of size $mT \times mT$, where m is the dimension of the control action and T is the horizon. This matrix is given by equation 2.7:

$$\Sigma = \begin{bmatrix} \Sigma_0 & 0 & \cdots & 0 \\ 0 & \Sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{T-1} \end{bmatrix} \tag{2.7}$$

In this matrix each element in the diagonal: Σ_t is the covariance for time step t, while the off-diagonal entries model the correlation between actions at different time steps. MPOPI uses Adaptive Importance Sampling (AIS). At each iteration, the distribution is updated to refine both the mean μ and the covariance Σ using the feedback from previous samples. In this manner the more frequently successful samples are generated, the narrower the variance becomes at those time steps, directing the algorithm toward more optimal regions of the control space. This results in fewer samples needed to converge to an optimum, However, it is worth pointing out that this comes at a cost. MPOPI is more likely to get stuck in local optima because it adapts the sampling to focus on regions that have previously yielded good results.

2.5. Handling Risk

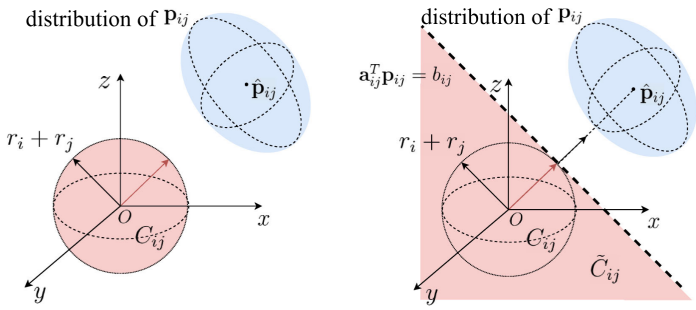
When navigating a dynamic environment there will often be some uncertainty about the future states of the system. For example when navigating around pedestrians or other robots there will be uncertainty about their future positions that needs to be accounted for to maintain safety in planning. As previously discussed there are two approaches to handle this with an MPC trajectory planner, Robust MPC and Stochastic MPC.

Robust MPC provides a safety guarantee in planning by entirely avoiding any potentially dangerous future states. For example, given a distribution about the potential future states of a pedestrian, the entire portion of the state-space covered by this uncertainty distribution would be constrained. Usually the shape of these distributions is approximated with convex polyhedra to assure the constraints remain differentiable, such as in the previously discussed MPCC based planner [12]. While assuring safety is beneficial, Robust MPC based planners tend to be over-conservative and are more susceptible to the 'freezing robot problem', where the robot's trajectory is severely inefficient as large portions of the state-space are constrained. Additionally, it is common to use unbounded distributions such as eg, Gaussians to represent uncertainty, in this case the 'worst case scenario' would be considered to approximate a 'bounded distribution', however this will often lead to avoiding unncessarily large areas [26].

On the other hand, stochastic MPC is better suited for these types of environments. Instead of considering worst case scenarios, it manages uncertainty by setting probabilistic guarantees via Chance Constraints. Instead of ensuring that the system satisfies a strict deterministic constraint at every time step (as in Robust MPC), chance constraints ensure that the constraint is satisfied with a specified probability. Equation 2.8 shows a formulation of a chance constraint:

$$\mathbb{P}(x_t \in \mathcal{X}) \geq 1 - \epsilon \tag{2.8}$$

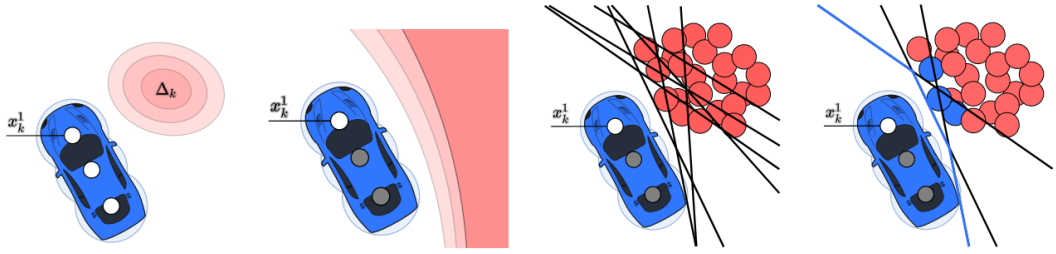
Where x_t is the state of the robot, $\mathcal{X}$ is the feasible set of states that don't violate the constraint and ϵ is the probability threshold or risk tolerance, which represents the probability the constraint is being violated. However, the introduction of stochastic constraints introduces complications in the optimization. Chance constraints cannot be optimized directly because they involve probabilistic conditions which are often non-convex and hard to handle in optimization problems. To address this, chance constraints are often approximated by converting them into representative deterministic constraints using techniques like Gaussian approximations. For example, in [11] the authors follow the Gaussian approximation approach. The constraint is linearized and an error function to bound the probability of constraint satisfaction with a margin based on the mean and covariance of the uncertainty. This linearization makes the problem tractable for real-time applications, even with low-compute such as on-board a Quad-rotor as the authors demonstrate. An illustration of the linearization can be found in figure 2.7.



(a) Spherical region (b) Linearization of (a)

Figure 2.7: Linearization of a multivariate Gaussian distribution. The blue sphere represents the robot-obstacle collision avoidance distance around the position of the robot. The red distribution represents uncertainty about the position of the obstacle before and after linearizartion. The linearized constraint makes the problem tractable for real-time applications. Figure from [11]

Another method is to use scenario-based optimization [27], where multiple possible realizations of the uncertainties are sampled. The probabilistic constraints are then cast as a finite set of deterministic constraints based on the sampled scenarios. Usually several of the scenarios are redundant. These scenarios are pruned with the constraints being crafted using only the most relevant samples. An illustration of this process can be found in figure 2.8 comparing linearizing against a scenario-based approach. This can approximate the chance constraints without relying on assumptions about the uncertainty distribution. This has the advantage of not requiring to make assumptions about the shape of the uncertainty distribution. The authors of [27] apply this methodology for mobile robots in an environment with other agents. They do this with the motivation that the uncertainty distributions representing the motions of agents in a dynamic environment are difficult to capture with a Gaussian approximation as they are often multi-modal for example.



(a) Original CCP (2c) (b) Linearized CCP (7b) (c) Linear SP (8) (d) Pruned SP (14c)

Figure 2.8: Figure comparing linearizing the uncertainty (left) against a scenario-based approach with scenario pruning (right). A shows the 1 σ to 3 σ interval of the uncertainty in red shades. B shows the probabilistic collision region when linearized from the robot disc at the front. C shows the sampled locations in red and boundaries of the constraints in black. D shows the resulting minimal polytope in blue after pruning. Figure from [27]

The work of [26] improves on traditional scenario-based approaches. In this approaches the risk level selected is manifested by the number of samples used to craft the constraints. This may cause an overly conservative constraint to be built if a lower probability scenario is samples, thus the actual risk may be lower than the target risk. To address this, the authors propose generating several trajectories in parallel, each with a different risk threshold. After generating these trajectories, an a posteriori risk assessment is performed to compute the actual risk, allowing the selection of the least conservative yet still compliant trajectory. This approach ensures that safety is maintained without unnecessary conservatism, improving performance in congested environments.

Another popular stochastic approach to handle risk in MPC planning is to use constraints on the Conditional

Value at Risk (CVaR). CVaR optimization minimizes the risk of catastrophic outcomes, such as collisions that result in significant damage or mission failure. For example, while a chance constraint ensures that the collision probability is low, CVaR would ensure that if a collision does occur, the expected severity (e.g., damage, cost) is minimized. Chance constraints handle risk in a binary way, CVaR handles it on a spectrum, assessing the expected damage when the worst happens. In this way the collision avoidance constraints are more adaptive, leaving larger margins at high speeds for example. However, CVaR additionally requires the use of a model to assess the severity of a particular outcome. Aside from the difficulty of designing such a model, it also makes the optimization more challenging to solve. The authors of [28] adapted CVaR for decision making robots in uncertain dynamic environments. Other authors have explored integrating CVaR with an MPPI planner[29].

2.6. Conclusion

Model Predictive Control (MPC) has proven to be a robust and adaptable framework for mobile robot trajectory planning. It is capable of handling complex dynamics and environmental constraints, however despite its advantages, MPC implementation presents several challenges, notably in addressing non-linear dynamics and environmental uncertainty. Gradient-based methods offer efficient real-time optimization and better convergence properties but can struggle in non-convex, cluttered environments. Conversely, sampling-based approaches, like MPPI, excel in complex, dynamic spaces and can achieve global optima, though at a higher computational cost. Recent advancements have leveraged parallel computing and hybrid methods to enhance these approaches, making them more applicable for real-time applications with complex dynamics in rapidly changing dynamic environments.

Risk management is another critical aspect of MPC, particularly for navigating unpredictable environments. Robust MPC provides safety guarantees but is often overly conservative, whereas Stochastic MPC, with methods like chance constraints and CVaR, enables more flexible and adaptive reasoning about environmental uncertainty and risk. The development of hybrid approaches, such as scenario-based risk assessments and adaptive sampling strategies such as MPPI has further pushed the boundaries of MPC based planning capabilities in balancing safety and performance. The choice between various MPC methods ultimately depends on the specific application needs.

3

Interactive Planning

3.1. Background and Problem Definition

3.1.1. Background

Multi-agent environments present significantly more complex planning challenges when compared to single-agent environments. In cases where communication or full information about other agents is available, planning can be simplified, as agents can coordinate with less uncertainty. However, such ideal conditions cannot always be assumed. In many real-world scenarios, agents do not have access to full information or direct communication, which leads to significant uncertainty about the trajectories and intentions of others. Moreover, agent trajectories are very often interdependent; the actions of one agent will influence the behavior of surrounding agents, creating a dynamic coupling between their paths. This phenomenon is especially pronounced in crowded environments, often encountered in urban navigation as illustrated in figure 3.1. Interactive planning is often compared to a form of negotiation, where agents must implicitly or explicitly agree on coordination strategies, such as determining who yields the right of way [3]. Therefore, successful path planning in these environments requires methods that handle both uncertainty and the interaction between agents [13].

Figure 3.1: In crowded multi-agent environments, agent's trajectories are coupled. When planning it is essential to model how surrounding agents will respond to the ego-trajectory.

3.1.2. Problem Formulation

This challenging task can be cast as a receding horizon trajectory optimization problem. In this approach, the multi-agent planning problem is modeled as a coupled dynamical system, where each agent interacts not only with the environment but also with other agents. This representation as a coupled dynamical system allows an agent to plan its trajectory while explicitly accounting for other agents responses by solving a joint trajectory optimization over all agents [3]. In this case other agents behaviors can be approximated via a cost function or a prediction model. In its most general formulation, this optimization

can be mathematically expressed as seen in equation 3.1:

$$
\begin{aligned}
\mathbf{u}^* = & \min_{\{\mathbf{u}_t^i\}_{i=1}^N} \sum_{i=1}^{N} \left(\sum_{t=0}^{T-1} L^i(\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{x}_t^{-i}, \mathbf{u}_t^{-i}) + F^i(\mathbf{x}_T^i, \mathbf{x}_T^{-i}) \right) \\
& \text{subject to} \\
& \mathbf{x}_{t+1}^i = f^i(\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{x}_t^{-i}, \mathbf{u}_t^{-i}), \quad \forall i \in \{1, \ldots, N\}, \\
& g(\mathbf{x}_t^i, \mathbf{x}_t^j, \Sigma_t^i, \Sigma_t^j, \epsilon) \leq 0, \quad \forall i, j \in \{1, \ldots, N\},\ i \neq j, \\
& \mathbf{x}_t^i \in \mathcal{X}^i, \quad \mathbf{u}_t^i \in \mathcal{U}^i, \quad \mathbf{x}_0^i \text{ given}, \forall i \in \{1, \ldots, N\}, \\
& \forall t \in \{0, \ldots, T-1\}
\end{aligned}
\tag{3.1}
$$

The objective is to minimize the combined cost for all agents, where each agent i has a stage cost $L^i(\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{x}_t^{-i}, \mathbf{u}_t^{-i})$ over time steps t, and a terminal cost $F^i(\mathbf{x}_T^i, \mathbf{x}_T^{-i})$ at the final time step T. The system is subject to state dynamics $\mathbf{x}_{t+1}^i = f^i(\mathbf{x}_t^i, \mathbf{u}_t^i, \mathbf{x}_t^{-i}, \mathbf{u}_t^{-i})$, which capture the coupling between agents. Inequality constraints $g(\mathbf{x}_t^i, \mathbf{x}_t^j, \Sigma_t^i, \Sigma_t^j, \epsilon) \leq 0$ can be used to ensure safety (e.g., collision avoidance), while state and control constraints $\mathbf{x}_t^i \in \mathcal{X}^i$, $\mathbf{u}_t^i \in \mathcal{U}^i$ enforce kinematic feasibility.

While this general formulation provides a framework for modeling interactions in multi-agent systems, it faces significant challenges on two key fronts. First, accurately modeling or estimating the behaviors and intentions of other agents is critical and can be approached in two primary ways: by estimating the cost functions governing other agents' decision-making, or by using prediction models to forecast their future trajectories. Estimating cost functions assumes that agents act rationally according to an internal objective, but inferring these objectives from observable actions is often complex and may require extensive prior knowledge and inference during the interaction[30]. Small errors in estimating these cost functions can lead to incorrect predictions about agent behavior, additionally, the rationality assumption may be invalidated when interacting with non-robotic agents, since a relatively simple cost function is unable to capture the complexities of human behavior. On the other hand, prediction models can be used to approximate other agents' trajectories with fewer assumptions about the structure of their reasoning. However, these models also present significant challenges: data-driven models may suffer from generalization issues, while purely physics-based models fail to capture the strategic interactions between agents [30]. Additionally, both prediction methods often introduce significant uncertainty as predictions are often distributional estimates, which complicates the optimization problem.

Second, the computational complexity of the problem increases exponentially with the number of agents. As the number of interacting agents grows, the joint state and action spaces become increasingly large, making real-time trajectory optimization computationally prohibitive [1]. This combinatorial explosion, often called 'curse of dimensionality', requires efficient algorithms and often approximations or assumptions to maintain scalability in practical applications. The following subsection provides a taxonomy of methods in literature for addressing this challenging task.

3.2. Taxonomy

Given the inherent complexity of interactive multi-agent planning, there is no silver-bullet apprproach that can generalize to all instances of the problem. Solutions presented in the literature are often highly tailored to specific applications, leading to a wide array of frameworks, each with different assumptions about the environment and varying strengths. This has given rise to a substantial body of research. Consequently, a wealth of survey papers has emerged, presenting taxonomies that classify methods according to distinct aspects. For example, [3] categorizes approaches based on the agent models used to predict the behavior of other agents, while [1] [6] classify methods according to how the underlying optimization problem is structured.

In this review, we focus on the challenge of scalability in multi-agent planning. To this end, we classify methods based on whether they solve the problem using a joint optimization approach, where the behaviors of all agents are optimized together, or a sequential approach, where prediction and planning are decoupled. In the sequential approach, a critical assumption is made [1]: the future trajectories of other agents can be predicted independently, without direct feedback from the ego-agent's planned actions. This simplifies the problem by first predicting the movements of other agents and then planning the ego-agent's

trajectory based on these fixed or semi-fixed predictions, reducing planning to a single agent problem, thus has invariant computational complexity with the number of agents in the environment. However, this framework relies modeling interactions implicitly, which often leads to poor performance in highly interactive environments. A diagram of the proposed taxonomy is presented in figure 5.3.

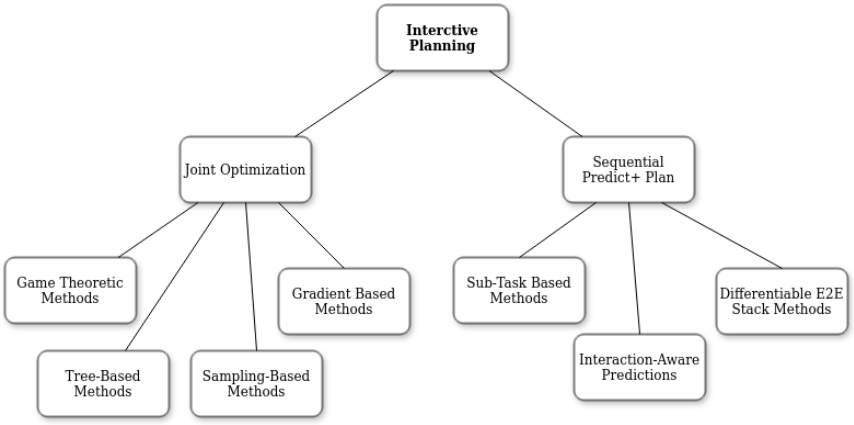


Figure 3.2: Proposed taxonomy for interaction-aware planning methods used in this chapter. Given this review is focused on scalability, the methods are broadly classified based on the complexity of the optimization task solved. With the two root categories being joint-optimization, explicitly considering interaction and sequential predict+plan approaches which remain scalable by opting to solve a single agent problem instead, while implicitly accounting for interactions.

The main divide in the taxonomy distinguishes between joint and sequential approaches. The following sections explore each branch in more detail by discussing relevant literature. Literature in joint approaches is selected to focus on how interaction is modelled and what optimization techniques and assumptions are introduced to make the problem tractable, whereas works on sequential methods are selected to focus implicitly accounting for interactions.

3.3. Joint Optimization Methods

3.3.1. Game-Theoretic Methods

From an interaction perspective, one of the most comprehensive approaches is found in game-theoretic methods [3]. These methods frame the planning problem as a trajectory game, where a constrained dynamic game is solved at each time-step. Instead of simply minimizing a cost function, the objective is to find an equilibrium solution. This is desirable because it leads to better stability: the goal is to find a strategy for all agents where no one has an incentive to deviate, resulting in safer and more coordinated interactions. Earlier approaches often focused on Stackelberg equilibrium solutions, where one agent (the leader) optimizes its strategy first, and the other agents (the followers) respond optimally to the leader's choice. However, more modern approaches have evolved to incorporate Nash equilibrium, where all agents optimize their strategies simultaneously, with each agent considering the others' strategies. Both approaches have different applications: Stackelberg equilibria are useful in hierarchical systems with one of the agents in a clear leadership role (Can be often found for autonomous vehicles for example). In contrast, Nash equilibria are well-suited for a wider variety of environments encompassing a wider variety of multi-agent coordination environments.

The largest challenge game theoretic formulations of this problem face is tractability, as often these methods are too compute intensive for real time deployment, even for problems with only two agents [31]. However recently there has been a lot of reasearch in the development of more efficient solvers. For example [32], uses a quasi-Newton root-finding algorithm to satisfy the first-order optimality conditions for Nash equilibrium, and handles constraints with an augmented Lagrangian formulation. The solver is designed to handle nonlinear state and input constraints, which are common for tasks like collision avoidance. Their method *ALGAMES* outperforms older solvers like iLQGames by finding equilibrium strategies three times faster in some cases. A shortcoming is that it requires good initial guesses, and is likely to fail with poor guesses. On the other hand [31] proposes a faster approach by casting the problem as a constrained

dynamic potential game and exploiting symmetries in the cost structure of the agents. In this manner it is possible to simplify the problem into a single-agent optimization. This is achieved under the assumption that agents have decoupled costs but share common constraints (like collision avoidance) and are rational, cooperative actors aware of each other's constraints. The method, *ALTRO* is up to 20 times faster than ALGAMES due to this simplification, but it applies primarily to scenarios where agent interactions can be framed as potential games with homogeneous constraints. While highly efficient, its applicability is limited to cooperative, structured settings with smooth dynamics and constraints, and it may not generalize well to environments populated with human agents.

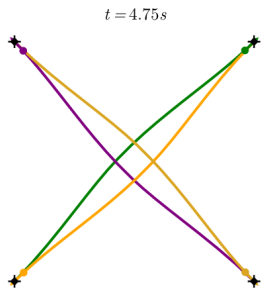


Figure 3.3: Illustration of Swapping Task solved for this experiment

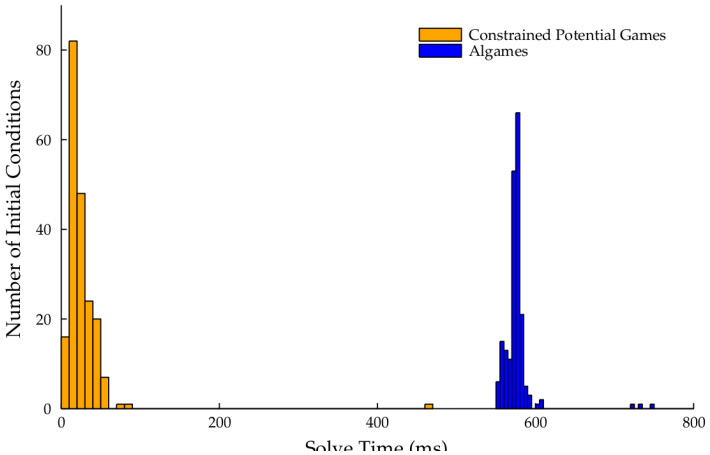


Figure 3.4: Compute times for *ALTRO* in yellow and *ALGAMES* in blue for a different set of initial conditions.

Figure 3.5: Modern Game Theoretic solvers are becoming increasingly faster. Both solvers here are SOTA solvers, however it can be observed that by making additional assumptions about the structure of the game ALTRO is able to achieve significantly faster performance. These assumptions are often needed to make the problems tractable, and a lot of the research on these problems deals with what would be the right assumptions to make and under which conditions they hold best. Figures from [31].

So far, the planning aspect of game-theoretic approaches has been discussed, however another fundamental challenge these methods face is estimating the cost function of other agents, as the solvers described above assume full rationality of the agents and full information about their intentions and constraints. Accurately estimating these is crucial because they directly influence the equilibrium solutions found. If the parameters or intentions of other agents are inaccurate, the resulting strategies can lead to suboptimal or unsafe interactions. Without perfect knowledge, it is possible to solve inverse games using estimation techniques like Bayesian inference or use inverse reinforcement learning (IRL). This allows for estimating other agents' objectives based on observed behaviors. These approaches aim to approximate the unknown parameters and update strategies dynamically, but they add complexity and can introduce uncertainty, making real-time decision-making more challenging. Reliable estimation thus remains a fundamental challenge for deploying game-theoretic methods in dynamic, real-world environments. The work of [33] addresses this challenge by employing a variational autoencoder (VAE) with an embedded differentiable game solver, constructing posterior distributions over unknown game parameters. This allows agents to reason about uncertainty in others' objectives. Instead of traditional Bayesian Maximum Likelihood Estimates (MLE), this approach is able to capture multi-modal distributions and efficiently sample from posteriors without needing to solve additional games at runtime. An illustration of their method can be found in figure 3.6. This is significant as usually inverse games are solved by iterating forward games with different parameters, which can be very computationally expensive. This leads to safer and more robust decision-making and is a notable step forward.

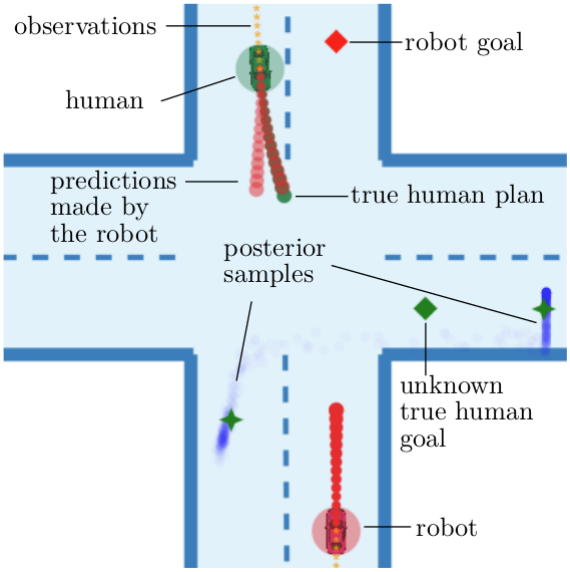


Figure 3.6: As opposed to using MLE estimates, the work of [33] explores introducing a differentiable game solver in a structured variational autoencoder to infer a distribution (possibly multi-modal as seen in the figure) of the other agent's objectives based on prior observation. This allows efficient sampling from the inferred posteriors without the iterative computing of game solutions to estimate the parameters of traditional methods.

3.3.2. Tree-Search Methods

Tree-based approaches usually discretize the actions available to the agent at each time-step and use the finite choices to build a Markov Decision Chain or a similar structure. To find an optimal sequence of decision methods like Monte Carlo Tree Search (MCTS) [20] and Tree Policy Planning (TPP) [34] construct a tree of possible future trajectories for the ego agent and other agents in the environment, allowing the system to explore different interaction scenarios and evaluate their outcomes. This allows for a 'multi-stage' planning process where the agent can explore how other agents will react to the different decisions it makes. Each branch of the tree is evaluated to compute a reward, and eventually the root of the branch with the best reward is executed. While the optimization process is standard and is used in many other MDP problems, it is often difficult to achieve performance in real time as this requires repeated predictions for other agents, thus querying a model a large amount of times. Training a model that remains accurate for later stages of the tree is also challenging [30].

The approach presented by [20] combines a Conditional Variational Autoencoder (CVAE) with an Exhaustive tree search planner. By discretizing the action space further into both discrete actions and longer timesteps (several timesteps grouped into a single action) it is possible to explore all branches of the tree simultaneously by leveraging modern GPU parallel implementations. This massive parallelism allows for excellent search capbilities over the entire state-space and is especially useful for multi-modal scenarios to avoid local minima. For each candidate action sequence, the CVAE predicts a distribution over the human's likely responses, conditioned on both the past interaction history and the robot's planned actions. By sampling from this distribution, the robot evaluates all possible action sequences and selects the one that minimizes an expected cost,. For promising branches more exhaustive sampling of the CVAE is employed. In this manner, they are able to plan highly interactive trajectories with pro-active behaviors from the agent. As their approach is especially suited for highly multi-modal scenarios, they test it on a traffic-weaving environment.

Similarly, [34] presents Differentiable Conditional Prediction and Cost evaluation for Tree Policy Planning (DTPP), a novel approach for both motion prediction and cost evaluation integrated with tree-based policy planning. DTPP integrates both motion prediction and cost evaluation into a single, jointly-trained framework, allowing for more efficient planning and accurate predictions conditioned by the planned trajectory. An illustration of the framework can be found in figure 3.7. In the tree-structured policy planning framework, a "trajectory tree" is built for the ego vehicle, representing potential future paths, while a

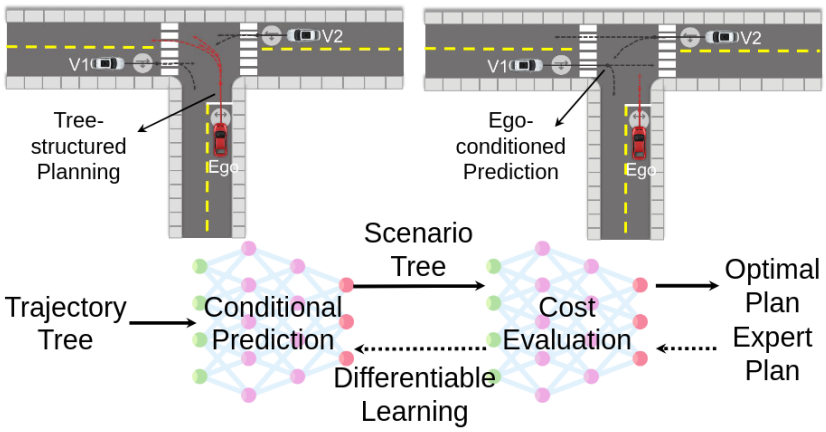


Figure 3.7: An illustration of (Top) ego-conditioned trajectory predictions for surrounding agents. (Bottom) Training pipeline for DTPP model [34], which combines prediction and planning learning to predict both the actions of surrrounding agents conditioned on the ego policy as well as learned cost functions for the ego agent, allowing it to more accurately evaluate its trajectories. This results in more adaptive behavior from the agent.

”scenario tree” is constructed to predict the responses of other agents to these potential ego trajectories. A query-centric Transformer model is used for ego-conditioned predictions based on the ego vehicle’s potential trajectories. Additionally, a learned context-aware cost function evaluates each tree branch, combining learned interaction features and handcrafted components, allowing for flexible and adaptive decision-making. In terms of efficiency, Compared to MCTS and other tree-based methods, DTPP is more efficient because it uses differentiable joint training, allowing the cost model to guide the expansion of the tree. This makes it possible to prune unpromising branches early and thereby reduce the overall computational load while maintaining excellent planning performance.

3.3.3. Sampling Based Methods

Sampling based methods solve the joint optimization problem by generating full trajectory samples for all agents, a ’joint trajectory’, in the interaction and selecting the ego trajectory from the best performing joint trajectory. This is an intuitive process to solve the optimization problem that’s capable of effectively exploring the trajectory space. However it faces challenges with modelling potential trajectories for other agents and keeping the optimization tractable for real time performance.

In [35], MPPI is applied for joint optimization using a two-stage sampling process. In the first stage, control inputs are sampled and evaluated independently for each agent. Each agent’s trajectory is simulated, and any trajectory that results in a collision or violates constraints is discarded early, using an agent-centric cost function that includes penalties for static collisions, deviations from tracking goals, excessive rotation, and speed violations. As for other agents cost function, a constant velocity model is assumed (The agent is modeled as aiming to maintain its current velocity). In the second stage, valid agent-level samples are combined to form system-wide trajectories. These trajectories are evaluated based on interaction costs, which account for dynamic collisions and regulation compliance (e.g., navigation rules). The two-stage process increases sampling efficiency by discarding some invalid samples early, ensuring that computational resources are focused on valid potential interaction-aware trajectories, however the number system level samples to evaluate still grows exponentially with the number of agents. An advantage however, is that samples are straightforward to parallelize, allowing for real-time performance with a moderate amount of agents. The largest advantage of this method is the simplified implementation as well as minimal modeling challenges for other agents. An extension to this model is provided by [36], where a learning based prediction model is incorporated to model other agents desired goal locations.

In [37] the authors present a multi-policy sampling-based framework. Each vehicle, including the ego vehicle, is modeled as executing one of several predefined high-level policies (e.g., lane-following, lane-changing, or turning). Behavioral prediction is achieved using Bayesian change-point detection, which segments the observed trajectories of other vehicles to infer the probability distribution over the potential

policies they might be executing, resulting in a distribution over the policy set. The system samples these policies with the number of policies for each sample determined by the belief for that policy. In this manner a set of samples is constructed for each agent including the ego, and the interactions between different samples can be evaluated, an illustration of this process can be found in figure 3.8. The samples are then forward simulated together in a closed-loop simulation to capture interaction, so that they represent high likelihood joint interactions. They are evaluated using a reward function that accounts for safety, rule compliance, comfort, and progress towards the goal. The ego vehicle then selects the policy that maximizes the expected reward.

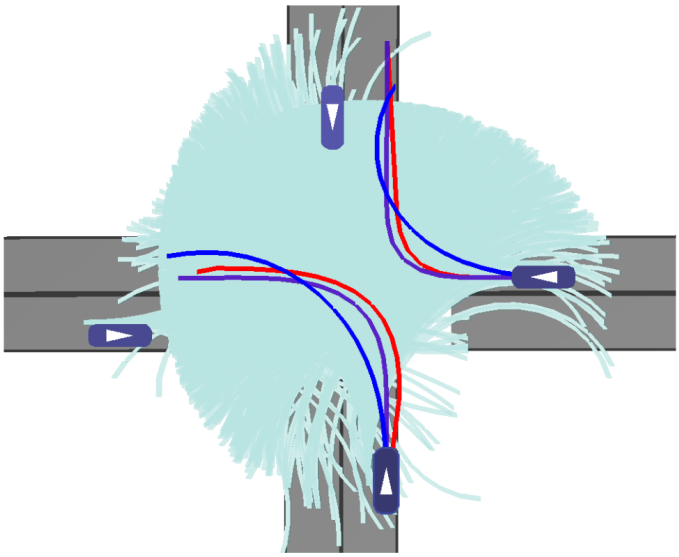


Figure 3.8: Illustration of sampling-based methodologies from [37]. A set of potential sample trajectories is generated for each agent, the interactions between the samples for different agents can then be examined to find the most suitable ego response.

The work of [38] presents a novel approach to sampling based joint optimization, where they use a sampling optimization methodology to find game-theoretic equilibrium solutions, effectively combining the benefits of both approaches. The authors design a model-based imitation learning policy (IMAP), that can learn to predict the behaviors of surrounding agents in reaction to other agents trajectories. In this manner the predictions can be conditioned on potential trajectories of surrounding agents. A novel approach they employ in training the policy, is that it is trained in closed loop, meaning it learns to predict actions one at a time, essentially learning to 'drive' as opposed to learning to predict. However with iterative evaluation it can be used to craft predictions up to a given horizon that remain dynamically feasible and better linked to the environment geometry and collision avoidance. This is a novel approach that blurs the lines between motion prediction and end-to-end driving. In their approach, this novel prediction methodology is then coupled with a derivative free motion planner for the ego agent, in their case, a Cross-Entropy optimization is used to solve the MPC problem for the ego agent. They then point out this policy can be used to solve for game theoretic solutions without the need for modeling other agents rewards or even best responses, and instead sampling from this policy, essentially 'playing a game with a motion prediction model', as this information about the players is already embedded in the model. They use this approach to find iterative equilibrium solutions, with both Stackelberg and Nash equilibrium solutions possible. The first method they propose is an iterative leader-folower MPC solution (ILF-MPC), with the ego-agent as the leader. The ego agent computes a set of candidate trajectories and for each trajectory, the IMAP policy computes a best response to each. The best trajectories are used to initialize the next iteration. This approach however, while converging faster and having less computational load might lead to overconfident ego behavior. The other approach is inspired by Nash equilibrium and computes an Iterative Best Response (IBR-MPC) which gives less influence to the ego agent. While slightly more computationally expensive this approach results in more balanced solutions. An overview of this framework is presented in figure 3.9. The integrated approach outperforms existing methods in environments like lane changes and adversarial

driving where agent interactions play a large role.

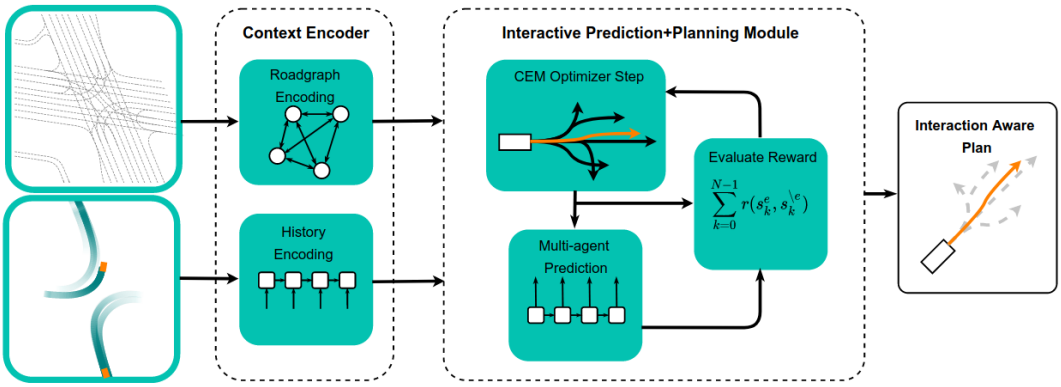


Figure 3.9: IMAP policy presented by [38] can be used to solve game-theoretic interactions via sampling from a learned policy for other agents. This eliminates the need for using simple models for other agents to make the problem tractable, which is one of the main limitations of traditional game theoretic methods.

3.3.4. Gradient-Based Methods

Other approaches instead solve the joint optimization of the coupled dynamical system directly using gradient based methods, similar to single agent MPC approach. This is often a complex optimization that can become intractable and is very prone to converging to local minima. To handle this challenging problem the approaches often explore finding good initial guesses or solve the optimization in parallel with several guesses to better explore the state-space. Often the policies of other agents are simplified to make the problem tractable.

The work of [39] introduces a novel approach they call Interactive Joint Planning (IJP) using gradient based methods for a joint multi-agent optimization. To make the complex problem tractable, IJP uses a data driven model from previous work (an advantage of the method is it is compatible with any available pre-trained model) to initialize the agents' trajectories. This gives a good initial guess for the problem, making it tractable. They model other agents behavior through a cost function penalizing significant deviations from these predicted behaviors. This ensures the agents' actions remain realistic while still allowing for interaction. To explore a diverse range of possible solutions, IJP introduces homotopy classes, which categorize trajectories based on how they navigate obstacles and interact with other agents. An illustration of homotopy classes for this problem can be found in figure 3.10. The planner runs multiple optimization iterations for each homotopy class, ensuring it finds the best solution for each distinct class. It then picks the best available solution from all homotopy classes. This is an effective method as it is able to take advantage of the exceptional predictive properties of model for high level objectives, but also exploiting the advantages of optimization at a local level to capture interaction and converge to optima. Combining the best of both worlds this way and doing so in a tractable manner.

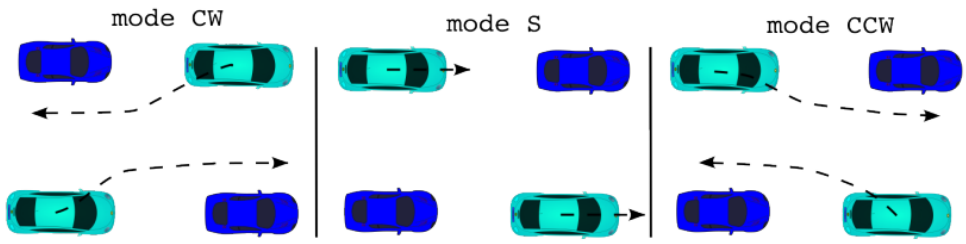
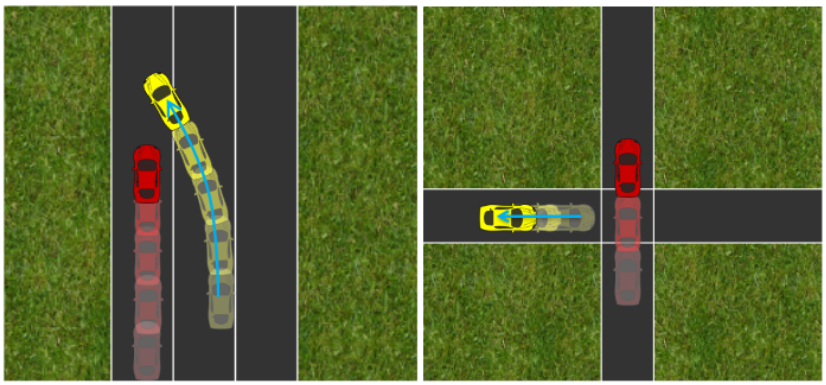


Figure 3.10: Caption

The work of [40] optimizes a coupled dynamical system from the perspective of the ego-agent, modeling how its behavior affects surrounding agents, allowing the planner to leverage its effect on their behavior.

Instead of passively avoiding human-driven vehicles, the method actively anticipates and can achieve more efficient and cooperative interactions. Using Inverse Reinforcement Learning (IRL), the system models human drivers as optimal planners and learns a reward function that represents their behavior. The ego-agent then uses Model Predictive Control (MPC) to plan actions that not only achieve its goals but also influence human drivers, such as encouraging them to slow down, change lanes, or go first at intersections. The optimization uses a gradient based approach with the best response of the human computed as a best response to the action of the robot. In general this is similar to the first step of an IBR appraoch, however by only performing one iteration the system avoids the complexity of a fully simultaneous game-theoretic interaction while retaining some of the interactive properties. User studies confirm that the approach effectively anticipates and influences human behavior as seen in figure 3.11. In this way the approach offers interactive planning properties with low complexity, however this approach is prone to over-confident behaviors.



(a) Car merges *ahead* of human; anticipates human *braking*

(b) Car *backs up* at 4way stop; anticipates human *proceeding*

Figure 3.11: The approach [40] achieves interactive behaviors while keeping computational complexity low. They model the human as an optimal planner and for each robot action they model the human as performing its best response to that action. This is similar to the first step of a game theoretic iterative Leader Follower approach. Even with this simplifying assumption, some interactive behaviors are retained.

A very similar approach is presented by [41], however instead of using a cost function learned via IRL and assuming rational behavior, they employ a learned model, namely a Socially-Aware Generative-Adversarial Network (SGAN) that iteratively predicts the behaviors of the agents one time-step at a time in reaction to the ego's planed trajectory.

3.4. Sequential Methods

Sequential methods offer a more scalable alternative to joint optimization for multi-agent planning by decoupling the prediction and planning tasks. In contrast to joint approaches, where agents plan their trajectories simultaneously while explicitly modeling the responses triggered in surrounding agents, sequential methods first predict the future trajectories of other agents and then plan the ego agent's path based on these fixed predictions. This reduces the problem to a single-agent optimization task, drastically lowering the computational complexity. However, the critical assumption that results in this benefit is that the trajectories of other agents can be predicted independently of the ego agent's planned actions. In environments with low interaction, this approach performs well. However, in highly interactive or densely populated settings, such assumptions can lead to poor coordination, as these models do not account for the coupled dynamics of agent interactions. To overcome these limitations, various strategies have been proposed to implicitly account for interactions in the prediction phase. The following sections explore key methodologies within the sequential approach to implicitly model or account for interaction.

3.4.1. Sub-Task Based Methods

The work of [42] proposes a framework that allows sequential planning robots to navigate complex interactive environment by learning to modify their path to reach their goal while avoiding complex interactive situations.

The core idea is to learn a policy that recommends sub-goals to guide a robot through interactive and dynamic environments. The policy is trained using deep reinforcement learning and is designed to suggest intermediate sub-goals that help the robot avoid dangerous situations, such as clusters of pedestrians or other obstacles. By doing so, the robot can navigate safely without directly following a fixed reference path that could lead to unsafe interactions. Instead, the sub-goal recommendations allow the robot to adapt its trajectory in real-time, ensuring safe and efficient movement toward the final destination. A new sub-goal is computed at each iteration, and serves as an effective surrogate for handling interactions implicitly. The framework is illustrated in figure 3.12. This approach improves collision avoidance and ensures the robot can handle complex scenarios it might otherwise struggle with. Although this might result in more costly trajectories, introducing this flexibility in the reference path it is an effective way to achieve safe navigation in complex environments while keeping the planner simple.

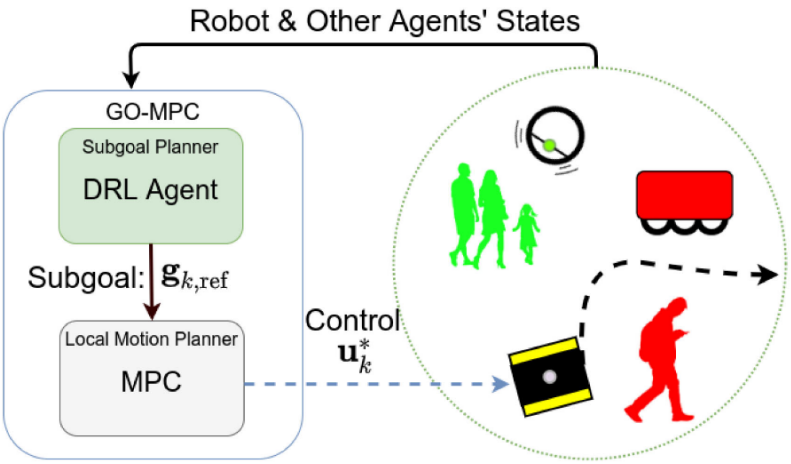


Figure 3.12: The approach presented by [42] learns to predict intermediate goal locations as an auxiliary task. These goal locations allow the robot to navigate toward the goal in a more flexible manner by learning to predict intermediate goals that allow the robot to avoid complex interactions, or guiding the robot through interactions. This serves as an effective way to implicitly handle interaction.

Similarly, in a second work [43], the same authors tackle the issue of navigating dense traffic via an auxiliary task. This time, a DRL policy recommends velocity references that take into account the behavior of surrounding vehicles, enabling the autonomous vehicle (AV) to negotiate complex interactions, such as merging into traffic or handling uncooperative drivers. By learning to adjust the AV's behavior based on the anticipated cooperation of other vehicles, this method can implicitly account for interaction and is shown to improve the vehicle's ability to navigate dense traffic safely and efficiently. The framework stays the same, but they substitute the MPC module for the one presented in figure 3.13.

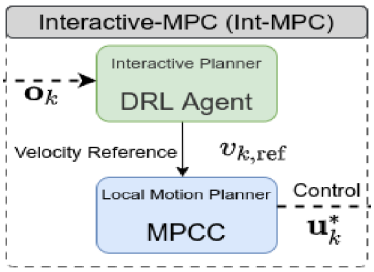


Figure 3.13: Velocity references are learned via DRL as an auxiliary task. This can serve as an effective method to handle interaction implicitly in dense traffic situations.

In both works, the key idea is to handle interactions implicitly through learned auxiliary tasks that guide the primary planning process. In the first paper, sub-goal recommendations guide navigation by suggesting safer paths, while in the second, velocity references guide the AV's motion planner to better handle

interactions. Integrating auxiliary tasks into sequential planning allows the robot or AV to show interactive behavior while avoiding the computational complexity and modeling challenges associated with joint optimization.

3.4.2. Interaction-Aware Predictions

An alternative to joint optimization is to account for interaction through a prediction model capable of of anticipating interaction and making predictions for other agents behavior in reaction to the trajectory planned by the ego agent upstream. However, training such a model to generalize across different environments is challenging. The accuracy of the predictions heavily depends on the planner being used during deployment matching the behavior seen during training. If a different planner is used, the prediction model may go out of distribution, as it would struggle to anticipate the ego agent's upstream decisions correctly, leading to inaccurate predictions of how other agents will react. This misalignment can significantly degrade the performance of the model [6], highlighting the importance of coupling the prediction model with a planner that replicates the behavior observed in the expert demonstrations used during training.

The work of [44] presents an interaction-aware prediction model based on recurrent neural networks (RNN) trained on data generated from a centralized motion planning system finding optimal solutions in a variety of scenarios. By learning from these demonstrated trajectories, the RNN model is able to capture the interaction dynamics between multiple robots and obstacles, predicting how the robots will move in response to one another. This prediction model is incorporated into an (MPC) framework. Each robot uses the predicted trajectories of its neighbors to plan its own motion. In this manner the robots are able to successfully coordinate around the learned optimal coordination strategies encoded in the model without having to solve a joint optimization. This allows scaling to a large number of robots, however it is constrained to a specific type of planner and cost function structures, such that the upstream ego plan does not invalidate the predictions.

3.4.3. Differentiable E2E Stack Methods

Similar to the section above Differentiable E2E Stack methods account for interactions implicitly via the prediction model, however a key difference with the above methods is that the prediction model is trained in closed loop with a planner. In this case, the planner can also have trainable parameters. These methods work by training both prediction and planner together via imitation learning or reinforcement learning to learn to replicate the expert behaviors in the training data. In this manner, both modules couple well together, and compliment each other to learn to interact like the experts n the training data.

Figure 3.14: The authors of [17] present DiffStack a differentiable end to end modular architecture. By learning both a prediction model and planner parameters together, this leads to tight integration between both modules. This addresses problems often encountered when using learned prediction models with planners. In this manner the vehicle is capable of interactive behaviors while maintaining a sequential planning approach.

The work of [17] presents a control stack that includes differentiable modules for prediction, planning, and control. The prediction module employs a neural network, specifically Trajectron++, to predict multi-modal trajectories of other agents, allowing it to capture interactions between agents. The planning module uses a sampling-based approach to generate dynamically-feasible candidate trajectories, and it selects the best trajectory based on a cost function. This cost function penalizes collisions, distance to the goal, lane deviation, and control effort. Both planner and control module have learnable parameters. The planner is made differentiable by relaxing the argmin operator, treating trajectory selection as a classification task and thus enabling back-propagation of gradients to the prediction model through the planner. An overview of the resulting stack is presented in figure 3.14. The control module applies an MPC approach,

using differentiable Linear Quadratic Regulator (LQR) to optimize the control sequence iteratively to track the planned trajectory. The differentiable stack allows for optimization of upstream prediction models with respect to the final control objectives, this influences predictions to be more relevant to the planning task. Traditional AV stacks, though modular and interpretable, face integration challenges, while DiffStack maintains modularity and interpretability with improved performance through end-to-end training that integrates the different modules more effectively.

Another very prominent work in this space explores Planning Oriented autonomous driving, where the entire stack is designed again to be differentiable, and tailored to aid the prediction task downstream [18]. Their critical insight is that by assigning a higher loss to the planning task downstream than the prediction loss upstream, this frees up some of the representational power of the prediction model to make predictions that not necessarily represent the ground truth extremely accurately, but instead guide the agent to behave pro-socially and plan efficiently. The framework uses query-based designs for communication between tasks, leveraging transformer-based architectures to capture both agent-agent and agent-environment interactions. The intermediate representations are not as hand crafted, which results in some better learned representations that convey information more effectively. However this results in lower interpetability. Traditional approaches to the task, where each module is trained separately often lead to error accumulation through the stack, whereas training the modules together with high focus placed on the final planning task results in a more robust pipeline. An overview of their framework is presented in figure 3.15

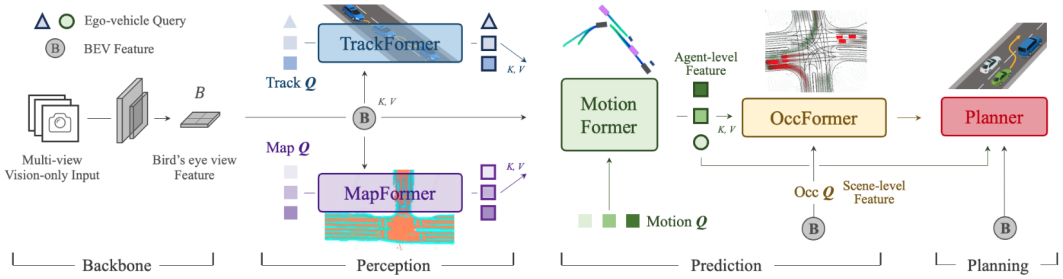


Figure 3.15: This figure illustrates the stack architecture proposed in [18], with an entirely differentiable stack optimized with a higher loss for the final planning task, allowing the prediction module to focus on guiding efficient, pro-social agent behavior rather than strict accuracy, and leveraging query-based, transformer-driven representations for seamless task communication.

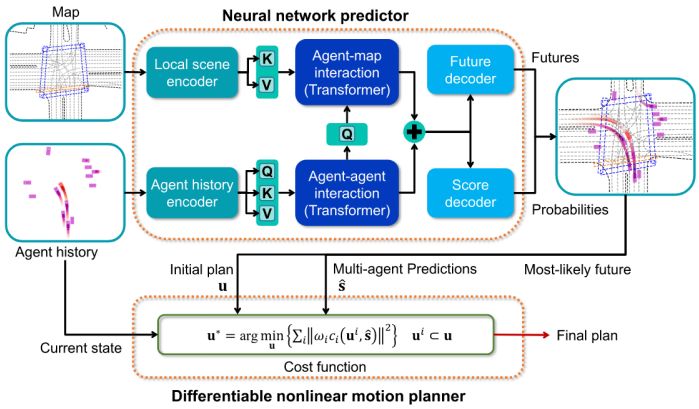


Figure 3.16: Architecture of DIPP by [45]. A transformer network is used to model joint futures for the agents, thus effectively capturing the most likely interactions. However, the prediction for the ego is sent to an action decoder. The decoded action sequence is used to warm-start an MPC planner. In this manner the planner can converge to better local minima.

Another similar approach, 'Deep interactive prediction and planning' (DIPP) a is presented by [45], however the difference is how interactions among the agents are modeled. This approach employs a transformer network that makes scene centric predictions for all agents conditioned on past interaction history and

map geometry. The interaction is computed in latent space, and the latent space representation is sent to a trajectory decoder for surrounding agents, whereas the the representation for the ego-agent is sent to an action decoder. The decoded action sequence is then used to warm-start an MPC planner with learned parameters. The pipeline is trained with a prediction error loss for other agents and a tracking loss for the ego agent with respect to the expert trajectory in the data. Approaching interaction this way results in more flexible trajectories for the ego agent as the planner is more expressive and can converge to better local minima, however this minimum is dictated by the outputs of the model, and thus the trajectory will still be interaction aware while maintaining a sequential approach. An overview of this architecture is shown in figure 3.16.

While these approaches offer strong integration between the planner and prediction model, they often lack flexibility and are not suitable for all environments. The planner and prediction model function well together, but they are tightly coupled, making it difficult to swap out one component without affecting the other. Another critique is the lack of explainability for the learned planner parameters, which is a safety concern since understanding these parameters is crucial in preventing malfunctions, the functioning of the network that generates them dynamically is often a black box. Additionally, coupling the layers in this way is highly data-intensive, and while it performs well in data-rich applications like autonomous vehicles with abundant datasets, it becomes impractical for more niche multi-agent scenarios where large datasets are unavailable.

3.5. Conclusion

In this chapter, various approaches to solving multi-agent path planning problems are discussed, focusing on the comparison between joint optimization methods and sequential methods. Both categories have their strengths and weaknesses, and their applicability depends heavily on the specific requirements of the task at hand.

Joint optimization methods, particularly game-theoretic approaches, offer comprehensive models that explicitly account for the interaction between agents, leading to more coordinated and stable behaviors. However, they face significant challenges related to scalability and computational complexity, making real-time deployment difficult in many applications. Modern methods attempt to mitigate these by making application specific trade-offs in terms of performance or assumptions about the environment or agent behavior. Although computationally expensive, these methods are often the best choice for highly interactive environments.

On the other hand, sequential methods, which decouple the prediction and planning processes, offer scalability by making assumptions that reduce the problem to a single-agent task. These methods perform well in low-interaction or structured environments. However, they often struggle in highly interactive scenarios, as they do not fully capture the interdependent dynamics of agents and instead rely on implicitly accounting for these couplings, which can lead to suboptimal coordination. Recent advances in interaction-aware prediction models and differentiable end-to-end architectures overcome these limitations by coupling the prediction and planning processes more tightly via joint learning of prediction models and planner parameters.

The choice of method depends on the environment requirements and available computational resources. While joint methods provide rich interaction modeling, sequential methods remain the preferred choice for scalability and real-time applications in most practical applications.

4

Legibility and Predictability of Robot Motion

4.1. Background

In multi-agent environments, agents often influence each others actions. This dynamic is particularly crucial in cooperative settings to achieve a desired shared objective such as collision avoidance for example. Humans effortlessly display cues such as gaze angle, body orientation or hand movements. While subtle, these signals provide essential information about a person's upcoming actions or goals, which greatly facilitate social interactions and cooperative behaviors. For example, a human agent may turn their body towards an object before reaching for it, or tilt their gaze to a particular location before moving in that direction[46]. This gives others the ability to predict their next move with relative ease. In contrast, robots often lack these cues, making it more challenging for other agents to understand their intentions. As a result, robots may fail to communicate their goals and plans effectively, which can compromise interactions in shared environments.

To this end, the field of legibility and predictability in robot motion focuses on how a robot can plan trajectories to implicitly communicate its intentions to observers, thereby facilitating smoother interactions[46][47]. The robot not only conveys its goal but also acts in a manner that continually reinforces the observer's confidence in their inference of the robot's objectives. By planning movements that are more predictable and easier to interpret, robots can enhance the overall efficiency and effectiveness of cooperative interactions with other agents. Research in this domain has traditionally focused on human-robot interaction (HRI) for finite tasks in a static environment where the human is not part of the environment, but only an observer. Recent works have explored the adaptation to multi-agent dynamic environments and receding horizon tasks. The section begins with introducing the definitions of legibility and predictability and the problem definition. In the next sub-section recent works adapting these concepts to dynamic environments and receding horizon applications are discussed.

4.2. Problem Definition

Earlier works in the field explored related concepts and defined various heuristics for intent communication and predictability of robot motion[46]. However, the most influential and cited approach in later works builds on the formalization by [48][8]. Their work provided precise definitions for legibility and predictability based in psychology. A frequently cited example in their research involves robotic arm clearing a table and a human observer attempting to infer its intention, about which objects it is most likely reaching for. In informal language legibility and predictability are often used synonymously, however one of the core insights of [48] is that these are fundamentally different properties of motion stemming from inferences in opposing directions. The differences between the properties and how this relates to mot ion is clearly illustrated by figure 4.1. The formal definitions for Legibility and Predictability are summarized in the following bullet points:

- **Legibility**: The degree to which an observer can quickly and confidently infer the correct end-goal G of the robot's trajectory.
- **Predictability**: The extent to which the robot's trajectory matches an observer's expectations, given a known end-goal G.

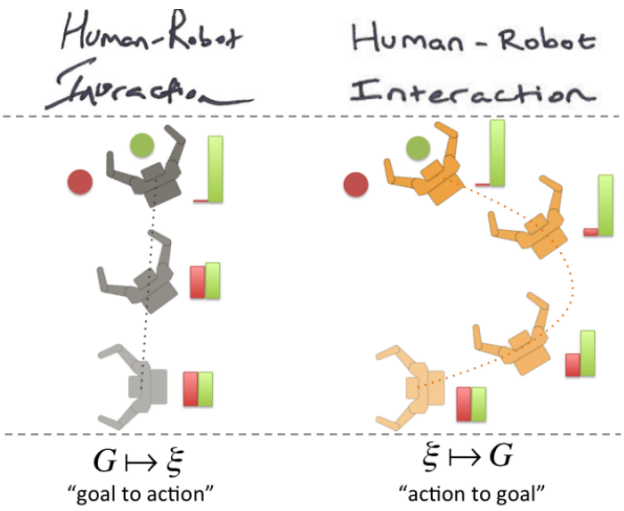


Figure 4.1: Top: Predictable, day-to-day, expected handwriting vs. legible handwriting. Center: A predictable and a legible trajectory of a robot's hand for the same task of grasping the green object. Bottom: Predictability and legibility stem from inferences in opposing directions.

Although these concepts are often correlated—for instance, a direct path toward a goal tends to be both legible and predictable—they can diverge under certain conditions. In environments with complex or ambiguous goal configurations, a highly **legible** trajectory may not be the most **predictable** one, and vice versa. This is especially true for environments where progress toward a possible goal is very highly correlated with progress toward another possible goal[49]. This will become clearer when the mathematical formulations for optimizing preidctability and legibility are discussed. Both properties are essential in designing motion planners for social and interactive tasks, balancing legibility and predictability requires careful consideration of the specific task and environment.

Mathematically, optimizing a trajectory ξ for predictability given a goal $G \in \mathcal{G}$ can be expressed by the following objective:

$$\xi^*_{S \to G} = \arg \min_{\xi \in \Xi_{S \to G}} C(\xi) \tag{4.1a}$$

$$\text{predictability}(\xi) = \exp\left(-C(\xi)\right) \tag{4.1b}$$

The observer is modelled as expecting the robot to be rational. It can be expressed as the optimization of a cost function $C(\xi)$ that the observer uses to estimate efficient and smooth motion 4.1a. Predictability is measured by how closely the actual trajectory matches this optimal inference 4.1b. On the other hand, legibility is concerned with how fast an observer can infer the correct goal G from an incomplete trajectory$\xi_{S \to Q}$ (From start S to mid-point Q). Legibility can be formalized through an inference function $I_L(\xi)$, which maps a trajectory (or a snippet of it) to the most probable goal as seen in equation 4.2a. $P(G \mid \xi_{S \to Q})$ is computed using Bayesian inference, with the likelihood of the trajectory given one of the possible goals. A legible trajectory is one where the correct goal G^* becomes apparent early. The legibility score is defined by equation 4.2b. Here $f(t)$ is a decreasing function used to assign more importance to early time-steps.

$$I_L(\xi_{S \to Q}) = \arg\max\ G \in \mathcal{G} P(G \mid \xi_{S \to Q}) \tag{4.2a}$$

$$\text{legibility}(\xi) = \int P(G^* \mid \xi_{S \to \xi(t)}) f(t)\, dt \tag{4.2b}$$

As mentioned above, a predictable trajectory follows the most efficient path toward the goal from the observers perspective, but this path may not make the goal immediately obvious (thus reducing legibility)

if the observer has uncertainty about the goal. In the opposite case, an overly legible trajectory could involve exaggerated motions that make the goal clear but deviate from the expected optimal path, reducing predictability and resulting in overly costly trajectories unnecessarily. To balance these two objectives a trade-off can be defined. This is done by combining both objectives in one equation and introducing a weight λ to control the emphasis on legibility versus predictability as seen in equation 4.3:

$$L(\xi) = \text{legibility}(\xi) - \lambda C(\xi) \tag{4.3}$$

The $\lambda C(\xi)$ term pushes the trajectory to not deviate too far from what is expected, while still aiming to be legible. Higher values of λ prioritize predictable, efficient motion, and lower values favor more legible, intent-expressive trajectories.

Shortcomings of original formulation
Traditional formulations of this problem are not well suited for receding horizon applications as they optimize over complete trajectories and rely on utility-based analytical models of observer expectations [49]. Additionally, the observer is modeled as inactive, thus having no influence on the planning agent. This assumption breaks down in multi-agent navigation where the observer and the agent share the workspace and influence each other. Furthermore, the traditional formulation becomes intractable for non-holonomic dynamics such as Unicycle or Bicycle model's commonly used for mobile robots.

4.3. Adaptation of Legibility and Predictability to Receding Horizon Applications

The concepts of legibility and predictability were initially formalized in the context of finite horizon tasks, however recent works have explored how they can be effectively adapted to dynamic, receding horizon control scenarios. In receding horizon control, a robot continuously re-plans its trajectory over a short, moving time horizon as it navigates its environment. This dynamic process introduces a need to balance legibility and predictability throughout the evolving motion plan.

In [49], the authors address this challenge, particularly focusing on improving the traditional legibility optimization to allow it to be used in receding-horizon in an environment with other agents even if the ego agent has non-holonomic dynamics. They propose a re-formulation that simplifies the legibility objective to Linear-Quadratic Regulator(LQR) form, allowing real-time deployment in dynamic systems. They do this by optimizing for a surrogate objective minimizing the difference between the costs associated with two hypotheses—one representing the null (uninformed) hypothesis and the other representing the intended task as per equation 4.4. This formulation assumes an environment with a discrete set of possible agent objectives, and that the observer has access to the set of respective cost functions for each objective.

$$L(u(\cdot)) = \alpha(t) \cdot H_{J1}(u(\cdot)) - (1 - \alpha(t)) \cdot H_{J0}(u(\cdot)) \tag{4.4}$$

In this equation $\alpha(t)$ is a decreasing function that prioritizes legibility at the earlier horizon timesteps. This objective allows robots to optimize their trajectories for both efficiency and communication of intent in a tractable manner in receding-horizon in an environment where they interact with the observers however it assumes observers with only two possible hypotheses (2 goals) and a utility-based observer expectation model. Additionally the trade-off depends on $\alpha(t)$ which has to be set before-hand making it less flexible to unpredictable environments.

A related approach by [50] proposes a navigation framework that adapts legibility and predictability to multi-agent contexts, focusing on the challenge of enhancing the communication of a robot's intended passing side during dynamic interactions. As opposed to optimizing for a global goal this approach accounts for *dynamic goal regions*, which represent potential passing and collision strategies in multi-agent environments, as illustrated in figure 4.2.

By optimizing for legibility in ambiguous scenarios and predictability in unambiguous ones, the framework ensures both safe and efficient collision avoidance during interactions. The cost function representing this objective takes a very similar structure to the objective developed by [49], optimizing to minimize the difference between the costs for both regions.

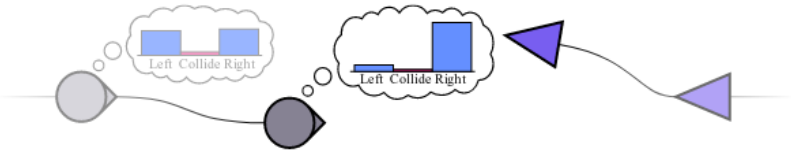


Figure 4.2: [50] focuses on optimizing for legibility and predictability with respect to passing side as opposed to a global goal in the environment. This simplifies the process of collision avoidance for interacting agents.

A similar results is achieved by [51]. Their method improves coordination in collision avoidance by allowing other agents to more quickly infer the ego-agent's intended avoidance strategy. The key contribution is the use of a markup factor μ^t, which heavily weights control costs later in the planning horizon, incentivizing the robot to react faster and thus make its intent more legible to observers. The improved proactive behavior enhances both safety and cooperation in shared environments. This promotes prosocial interaction where the responsibility for collision avoidance is shared equitably between agents by giving other agents more time to react and adapt their strategy. The performance of this method (figure 4.4) interacting with a human in an experiment (figure 4.5) is shown compared to a social forces model (figure 4.3). A constant velocity prediction was used by the robot.

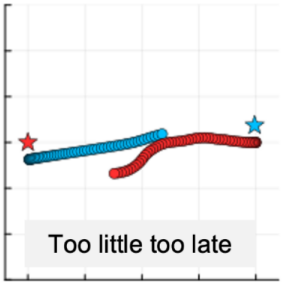


Figure 4.3: Social Forces Avoidance

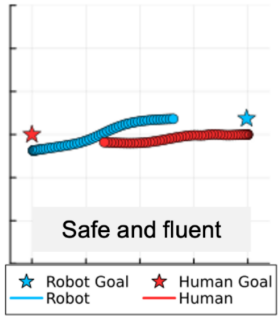


Figure 4.4: Legible method presented by [51]

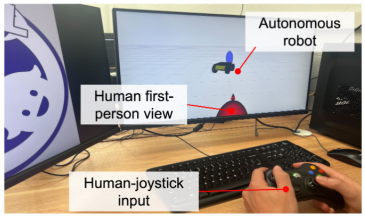


Figure 4.5: Experimental Setup

An alternative approach is to design a hand-crafted heuristic for legibility in a specific environment that can be used as an additional cost in planning. [52] propose a Legible Model Predictive Control (MPC) method for autonomous driving, specifically for lane changes in highway environments where safety and efficiency depend on the ability of surrounding vehicles to correctly anticipate the planned maneuver. The cost function is composed of a general term J_{gen} for objectives like energy efficiency and comfort, and a legibility term J^M_{leg} that optimizes for maneuver inference by other vehicles for a given maneuver M in trajectory ξ.

$$J = J_{\text{gen}} + w_{\text{leg}} J^M_{\text{leg}} \tag{4.5}$$

$$J^M_{\text{leg}} = \sum_{j=0}^{N} \frac{1}{c + P(M \mid \xi_{k+j})} \tag{4.6}$$

In this equation, $P(M \mid \xi_{k+j})$ represents the probability that the the observer infers the ego vehicle's planned maneuver M at each time step $k+j$, with N is the horizon length. The constant c is introduced to prevent division by zero and avoid explosion of the term for $P \to 0$. By penalizing low values of $P(M \mid \xi_{k+j})$, the cost function encourages the ego vehicle to choose trajectories that increase the likelihood of correct maneuver inference. This approach has the advantage that is integrates easily with a pre-existing planner, however it requires explicitly designing a function P to model observer's expectation. Additionally by being

maneuver specific it requires doing so for every possible maneuver where the agent interacts with other agents, making it very work intensive to design a general planner.

A variety of other approaches that achieve similar results can be found in literature [53][47], however while inspired by legibility and predictability their approaches differ more significantly from the original definitions presented. For example, [54] explores the problem of intent demonstration in general-sum dynamic games where a player with full information attempts to minimize the uncertainty about its utility function to players with partial information. The agent with full information does this by trading off its own task performance with the difference between the uncertain agents' estimates of its intent and the true intent. The work of [55] presents a novel framework for communicating intention that focuses on collision avoidance behavior. The framework uses a topological method based on "braid groups" to evaluate the complexity of multi-agent trajectories, aiming to minimize trajectory entanglement. By estimating others agents' avoidance behavior and reducing the complexity of interactions with the ego, the system achieves similar results to legibility and predictability. Another novel approach developed by [5] presents a data-driven approach for legible and predictable robot navigation in dynamic environments by learning from human pedestrian data. The approach uses maximum-entropy models to predict the behaviors of other agents and jointly plan the robot's actions in a way that is both interaction-aware and socially compliant. An illustration of this method from the robot's point of view (POV) can be found in figure 4.6.

Figure 4.6: POV of the robot jointly predicting others and planning its own trajectory by learning from human expert demonstrations. Figure from [5]

Figure 4.7: Since the robot is not human other agents react differently to it, pushing the learned model to face out of distribution cases

By learning from human data, the robot can plan its trajectories so that it mimics human behaviors and thereby complies with the observers expectation avoiding surprising behavior. The key innovation of this method with respect to previous legibility and predictability work is that it implicitly uses an observer expectation model learned from data. This model can more accurately capture the observers true expectation model, as opposed to hand-crafted heuristics used in the works discusses above. However, this method encountered significant challenges in test time with humans as the humans reacted to the robot differently than to other humans due to the 'novelty factor' of seeing a robot navigating among humans. They often stopped to look at it, thereby throwing the model out-of-distribution as seen in figure 4.7.

4.4. Conclusion

The adaptation of legibility and predictability concepts to receding horizon control in dynamic multi-agent environments can offer valuable insights for improving coordination during interactions between the agents. Early works took inspiration from human psychology to define mathematical formulations to capture these properties of motion. Recent innovations have refined these definitions and integrated them into more practical, real-time frameworks suitable for interactive receding horizon applications, enabling better coordination and safer interactions for autonomous agents. Approaches have evolved and range from from heruristic driven analytical observer models to data-driven methods that learn expectations directly from human demonstrations. Despite these advances, challenges remain, where unpredictable robot or human actions can push robots into unfamiliar, out-of-distribution scenarios.

5

Multi-Agent Trajectory Prediction

5.1. Background

As robots enter society, scenarios in which robotic agents interact with humans will become widespread. To ensure smooth and safe interactions, reasoning about the future is a crucial capability for agents. Humans can easily make decisions by implicitly reasoning about the future of interactions with other human agents. They do this using a mental model of others, often referred to as *Theory of Mind* in psychology [56]. On the other hand, prediction models capable of multi-agent trajectory forecasting need to be developed to endow robots with such capabilities. The development of such models has attracted increasing attention in recent years across several communities, as evidenced by figure 5.1:

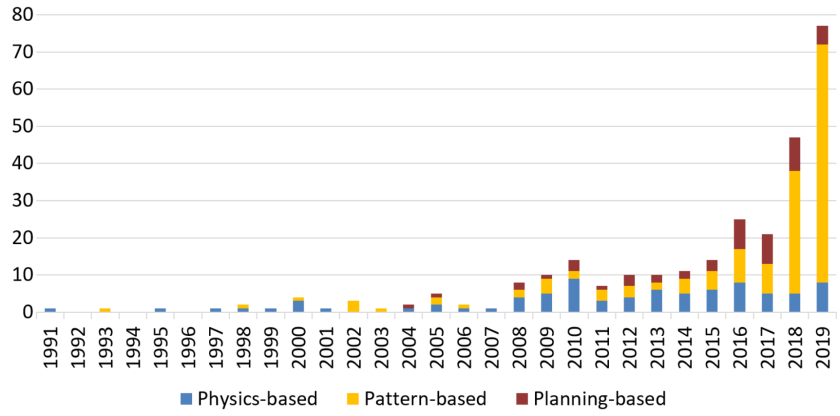


Figure 5.1: Publication trends in trajectory prediction from 1991-2019. Image from [57]

It is extremely challenging to develop a prediction model capable of accurately generalizing in practice. The distribution of multi-agent futures is often highly complex and can be influenced a range of both internal and external stimuli [57]. Aside from its goal intention, an agents future behavior can be influenced by interactions with surrounding agents, social rules or norms and the geometry and semantics of the environment. Some of these factors are not directly observable, thus goal inference and interaction modeling are necessary intermediate steps for trajectory prediction. Additionally, prediction models are required to operate in real-time.

5.2. Problem Formulation

Trajectory prediction tries to solve the problem of mapping from historical state observations to an estimate of future states(or distribution thereof). Aside from state observations, predictions are often conditioned on other context information such as environment geometry and the state histories of surrounding agents.

Following the breakdown presented in [6], the actors in the environment are partitioned into the *ego agent* (EA) and *surrounding agents* (SA). The state history of an agent i is defined over a time interval $t - t_{obs}, ..., t$ as:

$$X_i = \{x_{t-t_{obs}}, \ldots, x_t\} \tag{5.1}$$

Each state x may contain positional information, heading angle, velocity and further optional information such as goal information or other static attributes. The state history for the ego agent can be represented by X_{EA} as per 5.1. The state histories of surrounding vehicles can be concatenated and represented by:

$$\mathbf{X}_{SA} = \{X_{t-t_{obs}} \ldots, X_n\} \tag{5.2}$$

Analogously, the future states predicted by an agent i over a horizon t_{pred} can be written as:

$$Y_i = \{y_{t+1}, y_{t+2}, \ldots, y_{t+t_{pred}}\} \tag{5.3}$$

The concatenation of all surrounding agents predictions can be written as $\mathbf{Y_{SA}}$. Additional context information is represented by $\mathcal{C}$. The task of trajectory prediction is to construct a model P that maps from state histories **X** and context information $\mathcal{C}$ to a probability distribution P_{pred} over future trajectories **Y**:

$$P_{pred} = P(\mathbf{Y}|\mathbf{X}, \mathcal{C}) \tag{5.4}$$

The distribution accounts for the inherent uncertainties in the prediction task. It is common to represent this distribution with multiple samples with associated probabilities (e.g., a multi-modal prediction). Usually, the input **X** to the model contains the state histories of all agents in the scene. There exist several possibilities for model output, as seen in Table 5.1. The simplest task to learn is to estimate the future trajectory of each agent individually Y_{SA}; however, this approach often fails to predict the interactive behaviors of the agents. Predicting joint distributions for multiple agents can better capture interactive behavior and make consistent predictions. Although more accurate, the joint prediction task is difficult to learn, as the number of possibilities grows exponentially with the number of agents [30]. For this reason, some literature has explored predicting joint distributions for highly interactive cliques of agents $\mathbf{Y_{CA}}$ [58] as opposed to all agents in the scene **Y**.

Y	**X**	Task
Y_{SA}	$\mathbf{X}_{SA}, X_{EA}$	Single-Agent Prediction
$\mathbf{Y}_{CA}$	$\mathbf{X}_{SA}, X_{EA}$	Clique Prediction
$\mathbf{Y}_{SA}$	$\mathbf{X}_{SA}, X_{EA}$	Joint Prediction

Table 5.1: Different variants on the prediction task

An additional distinction in the prediction task has to do with how information is fed into the model with two main types of input representation, namely rasterized and sparse as seen in figure 5.2. Rasterized approaches rely on dense, fixed resolution grids with often multiple channels each representing different information. These approaches provide the model with all available information about the scene, making no assumptions about what input is relevant or how different parts of the input relate to each other. Much of the input is oftentimes redundant, as the future decisions of the agents in the scene are only dependent of a few key pieces of information. The idea of sparse input representation is to introduce some prior knowledge into the learning process to help discriminate what parts of the input are more important, as well as how different parts of the input relate to each other. Graphs are commonly used for this representation. The graphs are crafted from a set of vectors, polylines and polygons approximating the different features in the scene. These can then easily be encoded into fixed sized latent features. This approach has become increasingly popular in recent years as it can easily integrate with Transformer Networks which have become the state of the art for sequence modeling [59].

5.3. Taxonomy

The field has evolved significantly since its inception, and a variety of approaches have emerged. These range from early hand-crafted deterministic approaches such as Social Forces [60] or RVO model [61] to modern data-driven approaches. A variety of survey papers present different taxonomies of the field. A

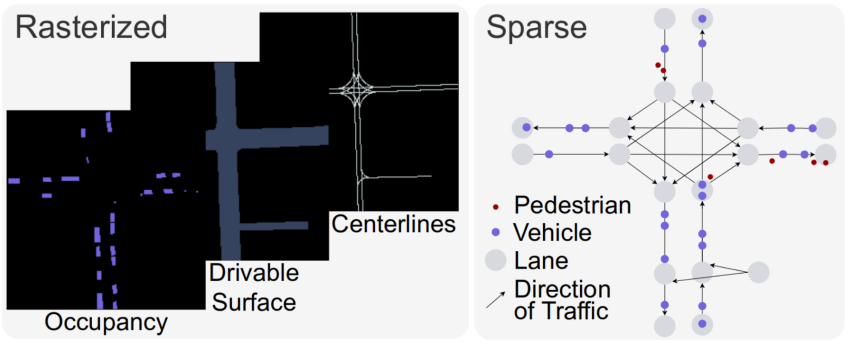


Figure 5.2: Rasterized and Sparse input representations. Figure from [6]

very clear taxonomy is presented by [9] which first groups methods by the assumptions they make about agents followed by the technical approaches they employ. An illustration of this taxonomy can be seen in figure 5.3.

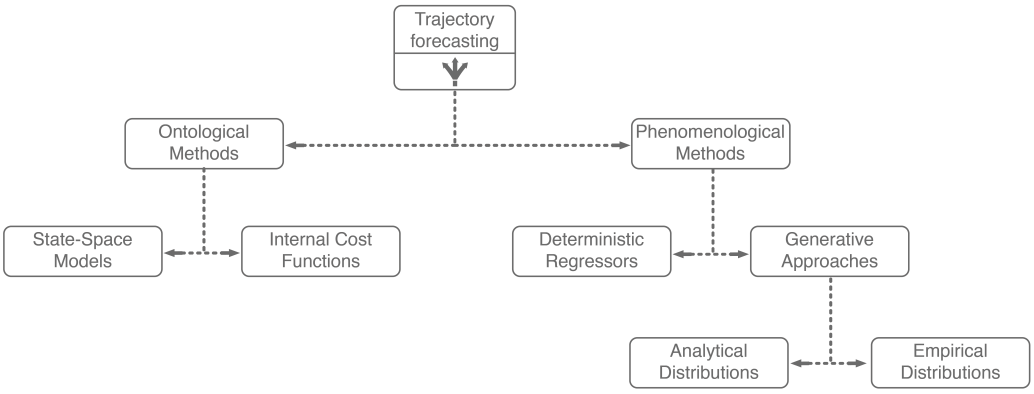


Figure 5.3: Multi-Agent Trajectory Prediction Taxonomy presented by [9]

The first distinction in the taxonomy distinguishes methods by the assumptions about the structure of the problem. Ontological approaches assume an underlying logic or model of the agents, whether that be a set of rules, dynamics model or cost function an agent is trying to minimize. In other taxonomies these approaches are referred to as physics and planning based approaches. On the other hand, phenomenological approaches make no assumptions about the structure of the problem an instead learn to forecast trajectories from large amounts of demonstration data. In other taxonomies these approaches are referred to as pattern based and deep-learning based approaches.

5.4. Ontological Methods

5.4.1. State-Space Models

A simple and often effective method for prediction is based on assuming a physics model describing the kinematic behavior of an agent. Common methods include Constant Velocity, Constant Acceleration, Constant Steering... Despite its simplicity, this approach can be remarkably effective, at times surpassing state-of-the-art models in pedestrian trajectory prediction for example [62]. However, using only kinematics does not account for critical factors such as obstacles in the environment and interactions between agents.

To address these limitations, numerous heuristic models have been developed to approximate agent interactions. For example, a common method for traffic prediction is the Intelligent Driver Model (IDM) [63]. It is designed to describe how a driver adjusts its acceleration based on the behavior of the car ahead,

maintaining safe distances while attempting to remain at desired speeds. While IDM is mainly suited for longitudinal driving, MOBIL (Minimizing Overall Braking Induced by Lane changes) extends the method by additionally modeling lateral behaviors [63]. The Social Forces Model can be used to model pedestrian interactions [60]. In this model pedestrians are treated as a system of particles governed by attractive and repulsive forces. These represent social behaviors, such as the desire to reach a destination, avoid collisions, and maintain personal space. AN illustration of the basic principle of this method can be seen in in Figure 5.4:

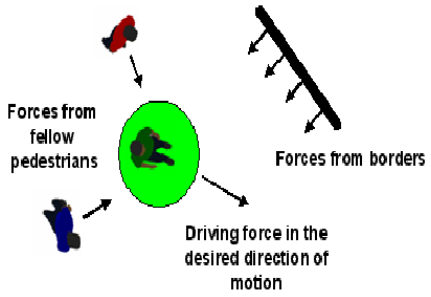


Figure 5.4: Illustration of basic Social Forces Model from [60]. An agent is influenced by an attractive force toward its goal while being pushed by repulsive forces from other agents and obstacles in the environment.

Another popular method is the Reciprocal Velocity Obstacle (RVO) [61]. This method is useful in dense dynamic environments. RVO makes predictions by assuming that all agents cooperate by making reciprocal adjustments to their velocity to avoid collision. Optimal Reciprocal Collision Avoidance (ORCA) [64] extends RVO by solving a constrained optimization problem where each agent computes the best possible velocity that avoids collisions while minimizing unnecessary changes in direction or speed, resulting in smoother more desirable trajectories. Reciprocal approaches such as RVO and ORCA are especially popular for modeling robot-robot interactions. They are efficient and can scale to dense scenes for many agents, however by assuming reciprocity they may not generalize well to agents that don't follow the same 'rules' such as human agents.

5.4.2. Internal Cost Functions

More recently, Inverse Reinforcement Learning (IRL) has become a popular ontological approach for trajectory prediction [3]. Given a dataset of agent trajectories ξ, this approach assumes that agents are rational actors trying to maximize some reward function, thus the goal in training is to find an approximate of that cost function. Then in test time, predictions are formed by crafting a distribution over trajectories based on the reward. The cost function is usually structured as seen in equation 5.5:

$$R(s) = w^T \phi(s), \tag{5.5}$$

$R(s)$ represents the reward at state s, w denotes the weights to be learned, and $\phi(s)$ is a function that extracts the relevant features from the state. The goal is to find the set of weights w that maximizes the rewards of the the observed trajectories ξ. However, a common issue is that there may be multiple reward functions that maximize the reward for ξ . To resolve this ambiguity, the principle of maximum entropy is often employed, selecting the reward function that maximizes entropy, thus reducing overfitting to ξ. This approach, known as Maximum Entropy (MaxEnt) IRL, is widely used in modeling real-world navigation and driving behaviors. In MaxEnt IRL prediction distributions, trajectories with higher reward are exponentially more likely, leading to the formulation in Equation 5.6. This probabilistic framework makes sampling from the policy a simple and efficient method to craft the distribution.

$$p(\xi|w) \propto \text{exp}\left(\sum_{s\in\xi} R(s)\right) = \sum_{s\in\xi} w^T \phi(s) \tag{5.6}$$

Using learned or inferred in real-time cost functions for prediction couples very well with interactive decision-making processes, such as the joint planning approaches found in game theoretic planning. This is especially useful for applications in robot-robot interactions. In these scenarios, the rational decision-making assumption predictions depend on holds strongly [3].

In conclusion, ontological approaches allow for the incorporation of prior knowledge to provide a structured and sample-efficient framework for modeling decision-making agents. However, the assumptions introduced by the prior knowledge also impose limitations, particularly when the true reward function is non-linear, non-Markovian, or differs significantly from the assumed model [30]. As the availability of training data continues to grow, the exploration of alternative methodologies, such as phenomenological approaches, becomes increasingly more relevant.

5.5. Phenomenological Methods

Phenomenological approaches for trajectory forecasting focus on minimizing assumptions about the internal decision-making processes of agents. Also referred to as pattern based methods, they leverage the universal representational power of Deep Neural Networks (DNNs) to capture the complexity of environments involving multiple decision-making agents. As the availability of larger datasets continues to grow these methods have become increasingly popular [57].

5.5.1. Deterministic Regressors

Methods based on regression models such as *Gaussian Process Regression (GPR)* [65] and deep learning architectures like *Recurrent Neural Networks (RNNs)*, *Long Short-Term Memory (LSTM) networks* and *Convolutional Neural Networks (CNNs)*, have been widely employed for trajectory forecasting. Among these, LSTMs often outperform other architectures and are computationally efficient for online evaluations. Consequently, LSTMs have become a core component of many trajectory models, given their suitability for modeling temporal sequences. Trajectory forecasting is typically framed as a time series prediction problem, making LSTMs a natural choice for this task [66]. Despite their successes, a limitation of these methods arises in safety-critical contexts, such as autonomous road vehicles, where given the inherent multi-modality of trajectory predictions makes single deterministic predictions insufficient to account for risk at the required level. In such scenarios, it is essential to consider multiple possible future outcomes and their associated likelihoods. Recent advancements in deep generative models have shifted focus from predicting a single trajectory to producing distributions of potential future trajectories [67]. This shift is particularly advantageous for downstream tasks such as motion planning, where knowledge of distributional properties, including variance, can inform safer decision-making.

Figure 5.5: Figure from Social-LSTM publication comparing model outputs of CV, Social Forces and Social-LSTM [66]. The figure illustrates how the models can capture increasingly complex trajectory features respectively. Additionally note how the outputs are uni-modal.

5.5.2. Generative Approaches

Generative models utilizing deep recurrent architectures like *Conditional Variational Autoencoders (CVAEs)* or *Generative Adversarial Networks (GANs)*, emerged as effective multi-modal predictors. These models can explicitly or implicitly encode multi-modality, producing position distributions that reflect the inherent uncertainty in future agent behavior. GAN-based models generate trajectory distributions directly, while CVAE-based approaches often rely on a bivariate Gaussian or *Gaussian Mixture Model (GMM)* to model

output distributions [30]. The main difference between GAN and CVAE based methods lies in their output distributions. GANs are designed to learn a mapping from samples drawn from a known distribution $p(x)$ to samples from an unknown target distribution $p(y)$ approximated from the training data. Effectively what this allows is to sample from a known distribution (x typically represents the agent's trajectory history and context), to an unknown 'predicted' distribution (y represents the predicted future trajectory). In this manner, GANs directly produce predicted trajectory samples. Crafting a distribution required repeated sampling which could become inefficient for real-time prediction. Instead, CVAEs represent $p(y \mid x)$ by decomposing it into components dependent on a latent variable z, as follows:

$$p(y \mid x) = \sum_z p(y \mid x, z)p(z \mid x),$$

where z is typically discrete. This decomposition allows for the generation of an analytic output distribution, similar to Gaussian Mixtures. Discrete latent spaces have been shown to provide superior performance for trajectory forecasting tasks especially when coupled with a planning task downstream that accounts for safety [58].

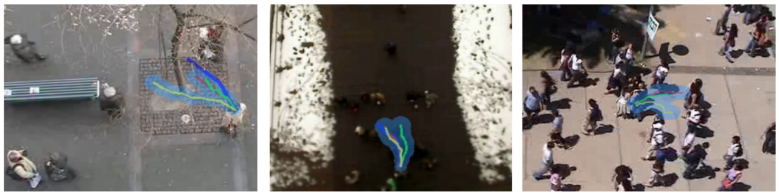


Figure 5.6: Figure from Social-VRNN publication [68], showcasing the multi-modal outputs of a variational approach. In this case a Vartiational Recurrent Neural Network (VRNN). The VRNN offers a variational approach to modelling time-series data, allowing for multi-modal outputs.

Finally, for environments where a wealth of training data is available, such as autonomous driving, Transformer based architectures have dominated the state-of-the-art in recent years [69] [70]. In the same way Transformers supplanted RNNs and LSTMs for Natural Language Processing (NLP), their exceptional sequence modeling capabilities extend to behavior prediction. Transformers excel at scalability, capacity, and parallelism. Training them is data intensive as they often contain many learnable parameters, however these architectures are very general and can very effectively leverage their parameters. Often times the performance of these networks keeps increasing with more training data. It is for these reasons they excel at complex temporal sequence modeling tasks and dominate multi-agent prediction for interactive traffic scenarios where a wealth of data is available. A popular benchmark in which this trend becomes apparent is the annual Waymo Prediction Challenge, where applicants compete to develop SOTA models. The leader-board for the 2023 edition is presented in figure 5.7. This figure is provided to illustrate the dominance of the transformer architecture in SOTA models. All entries with 'Tr' in the name explicitly hint at the use of this architecture, however for all entries the transformer is at the heart of the model and often takes the role of capturing the interactions of agents with the environment and each other. In figure 5.8 the architecture diagram for the Wayformer model is presented as an illustration of a transformer based network [71]. Usually the networks have several input modalities including state histories, road graphs, and other optional context information. Each input type often has its own encoder block. The encoded inputs are fused in a transformer block which serves as the core of the network, computing the interactions of agents with each other and the map in latent space. To obtain trajectories the decoder can be a second transformer as seen in Wayformer, however, it is also possible to use a feedforward block. The decoders are often designed to decode multi-modal predictions.

5.6. Conclusion

Multi-agent trajectory prediction is a crucial capability for autonomous agents navigating among humans and other robots, where understanding the future behaviors of surrounding agents is necessary for safe and efficient interaction. The field has evolved significantly, transitioning from deterministic, heuristic models to data-driven approaches. Ontological methods, integrate domain knowledge to create structured predictions but face limitations when true agent behaviors deviate from assumed models. Phenomenological methods, empowered by the increasing availability of large datasets, have demonstrated superior predictive power

Method Name	Lidar data for training	Object Type	Evaluation Time	Soft mAP	mAP	minADE	minFDE	Miss Rate	Overlap Rate	Date (Pacific Daylight Time)
		All	Avg	Show rest						
MGTR_ens		All	Avg	0.4764	0.4658	0.5825	1.2009	0.1258	0.1270	2023-09-15 19:06
MTR++_Ens		All	Avg	0.4738	0.4634	0.5581	1.1166	0.1122	0.1276	2023-05-23 15:37
MGTR		All	Avg	0.4599	0.4505	0.5918	1.2135	0.1298	0.1275	2023-09-14 21:18
GTR_ens		All	Avg	0.4518	0.4428	0.5855	1.2056	0.1296	0.1277	2023-05-25 02:58
EDA_single		All	Avg	0.4510	0.4401	0.5718	1.1702	0.1169	0.1266	2023-08-07 07:23
IAIR+		All	Avg	0.4480	0.4347	0.5783	1.1679	0.1238	0.1263	2023-05-23 23:56
AMP		All	Avg	0.4451	0.4334	0.6021	1.1918	0.1311	0.1388	2023-12-09 23:27
MTR++		All	Avg	0.4414	0.4329	0.5906	1.1939	0.1298	0.1281	2023-05-23 12:31
IQE_2		All	Avg	0.4409	0.4286	0.5798	1.1681	0.1250	0.1270	2023-11-28 06:43
GTR-R36		All	Avg	0.4384	0.4255	0.6005	1.2225	0.1330	0.1279	2023-05-23 20:39

Page 1 of 3 10 Rows

Figure 5.7: Due to the availability of a wealth of training data, the popular Waymo motion prediction benchmark has become dominated by transformer based architectures.

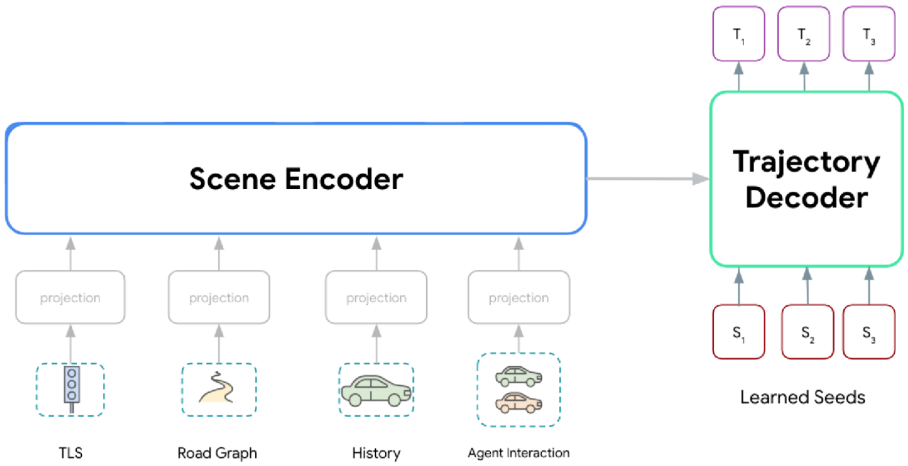


Figure 5.8: Architecture diagram of Wayformer model as an illustration of a typical transformer based architecture [71]. In this case two transformers are used. In the first encoder block the different input modalities are initially encoded by separate encoder blocks and then fused in a transformer block capturing the interactions in latent space. The output from this block is further decoded by a transformer decoder block.

and make no assumptions on the internal reasoning of surrounding agents. however they face challenges generalizing to scenarios not seen in the training data.

Despite the large advancements advances, challenges remain in crafting prediction models that are both computationally efficient for real-time applications and capable of generalizing effectively across diverse environments. Although transformer-based models excel due to their scalability and capacity, their reliance on extensive training data can limit applicability in data-scarce domains. [3] points out diminishing returns from higher accuracy prediction models, and instead proposes that further improvements in downstream performance will instead come from better integration of the prediction models developed with the planning task downstream.

6

Conclusions and Research Objective

This chapter builds on the discussions of previous chapters to outline the research objective and its motivations. The relevant takeaways from the previous sections are summarized in a conclusions section. Next, the motivation section discusses why integrating prediction models with sequential planning remains challenging and identifies opportunities for improvement. A Proposed Approach section outlines a preliminary methodology to achieve this objective. Finally, the chapter closes by providing Research Questions including the main research objective and supporting questions to guide the investigation.

6.1. Conclusions

Significant advances have been made in recent years in the literature on autonomous navigation of ground robots in multi-agent environments. These environments are notably characterized by being dynamic with high levels of uncertainty, introducing considerable challenges in prediction of the environment and planning of safe and efficient trajectories. A range of architectural approaches and planning methodologies have been proposed, spanning from fully end-to-end systems that use neural networks to map sensor inputs directly to control commands, to modular architectures where individual tasks are addressed by separate modules. Currently, most practical applications rely on sequential architectures, as end-to-end methods remain experimental. Within the planning module of sequential architectures, receding horizon trajectory optimization has emerged as a widely adopted paradigm.

Given agents that plan trajectories in a receding horizon manner, this raises the question of how to account for interaction in the planning stage. Research approaches in the literature can be broadly divided into two main methodologies as discussed in chapter 3: joint optimization and sequential predict-and-plan. Joint optimization methods explicitly model interactions between agents by solving a coupled optimization problem that considers all agents paths simultaneously. This approach allows for strong performance in environments where interactions are critical and decisions need to be coordinated. However, a notable challenge is that these methods tend to scale poorly as the number of agents increases, having exponential computational complexity with the number of interacting agents. This makes real-time performance in dense environments intractable.

On the other hand, sequential predict-and-plan methods address the issue by decoupling the prediction of other agents' trajectories from the ego agent's plan. This approach instead implicitly accounts for interactions while solving a single agent problem. The benefit is that it scales well with the number of agents, making real-time performance in dense complex environments tractable. However, sequential methods often struggle in highly interactive uncertain settings. The degree to which agents can interact and coordinate with others is often determined by the prediction model used. Thus, accurate prediction models are essential, however such models can be challenging to develop due to the complexity and inherent uncertainty of multi-agent interactions.

6.2. Motivation

Given the tremendous advancements in multi-agent trajectory forecasting provided by advancements in sequence modeling, predictions are able to account for interaction and provide consistent predictions at a scene level. This presents a good opportunity for the improvement of sequential predict+plan approaches, as the extent to which they are able to coordinate and interact with others is determined in large part by

the prediction model. Thus leveraging the improved ability of state-of-the-art prediction models to forecast interactive trajectories these methods could be improved to achieve similar performance to joint methods in highly interactive environments, while avoiding the exploding computational complexity.

However, despite the advancements in prediction models, integrating these models with planners has yielded mixed results. This is often due to the tendency for algorithmic planners to induce distribution shift on the model outputs (often trained on expert data collected from human drivers or pedestrians, as this is the richest data-source and they types of environment that result in most complexity) unless the models are trained or fine-tuned using data generated by the planner itself. This dependency limits the practical applicability of prediction models in autonomous navigation, particularly in highly dynamic and interactive environments where agents trajectories are heavily coupled.

For this reason it is interesting to investigate in what manner prediction models could be used in sequential planning more effectively. To this end, it is possible to identify that the main factor causing this shortcoming in the first place is that in the planning stage no constraints or conditions are placed on the plan being computed other than to avoid collision with others given their foretasted trajectory. However, since the trajectories of agents in this context are heavily coupled, any prediction about surrounding agents also implicitly places an expectation on the ego-agent's plan. If the agent's plan significantly deviates from this expectation, it can negatively influence the accuracy of the predictions. Additionally, since modern prediction models consider all information in the scene when making trajectory forecasts, if the ego agent's behavior significantly deviates from any expert demonstrations used in training, the model with result in low quality predictions for other agents induced by the ego behavior. Given these two considerations, this research proposes to investigate how some form of feedback from the prediction model can be used in planning to improve the quality of predictions about others and enable smoother interaction.

6.3. Proposed Approach

Drawing from the literature on observer-aware planning discussed in Chapter 4, we can explore how these properties of motion can be adapted to sequential predict-and-plan frameworks. In this context, agents do not reason about these properties in relation to potential end goals but instead utilize a prediction model to represent the observer's expectations. When employing a prediction model, an agent exhibits predictable behavior if it follows the most probable trajectory as output by the model. On the other hand, an agent optimizes its trajectory for legibility by planning in a way that minimizes the complexity of the prediction distribution and pushes it to align with its own goal.

By assuming surrounding agents perform planning in a sequential predict+plan manner and use a prediction model to approximate the ego-behavior, this prediction model queried on the ego agent can be used to approximate the expectations of these surrounding agents. This approach provides the ego agent with insight into what neighboring agents anticipate it will do and the future scenarios they are considering. Significant deviations from this predicted distribution would be surprising to other agents, potentially prompting them to re-plan their own trajectories. This could invalidate the predictions of the ego-agent, potentially leading to a clumsy and uncoordinated interaction. By querying a prediction model using the ego-state history, an agent can gain valuable insight into what other agents expect its future behavior to be. To act predictably, it should aim to stay as close as possible to these expectations and thereby avoid surprising others, making them easier to predict. However, this model might not always accurately represent the intentions of the ego agent. Thus, the agent must trade off predictability with legibility. To achieve legibility, the agent should strive to influence the prediction distribution to align with its optimal path and reduce the complexity of the prediction distribution.

The project aims to investigate how to utilize a prediction model in this manner and how to balance this trade-off between legibility and predictability. It is hypothesized that this would enable agents to act in an interactive and socially compliant manner while adhering to a sequential predict-and-plan approach by indirectly boosting the effectiveness of the prediction model used to coordinate with others. Essentially, this would involve adding a 'predictability' term to the cost function, to bound the plans of the agent to regions of the state-space that don't invalidate the prediction distribution used in planning.

6.4. Research Questions

In this subsection the research objective of the thesis is provided, followed by a supporting set of sub-questions to serve as stepping stones in answering the main research objective:

Research Objective

To what extent can feedback from a prediction model, queried on the ego state-history, be leveraged to enhance coordination during interactions in sequential predict-and-plan architectures within multi-agent environments?

Research Question 1

Drawing from the literature on observer-aware planning, how can a trade-off between legibility and predictability, relative to the output of a prediction model, be incorporated as a term in an agent's navigation cost function?

Research Question 2

What types of prediction models are most suitable for this approach? Can the method generalize across various prediction models, including discrete and continuous distributions? How does it perform when applied to agent-centric versus scene-centric prediction models?

Research Question 3

How does the method perform when there is no access to the exact prediction model utilized by other agents? How robust is the approach under conditions where an approximate or degraded version of the model is used? What are the implications for interactions involving human agents?

Research Question 4

What are the effects of tuning the cost function parameters? How do variations in the weight allocation impact performance, and is there an optimal configuration? Can adaptive weight adjustments based on context improve outcomes?

Research Question 5

What benchmarks and test scenarios should be selected to effectively evaluate this method? How should it be compared to existing approaches, and what representative scenarios would best highlight the distinct effects of the proposed method?

References

[1] Wilko Schwarting et al. “Planning and Decision-Making for Autonomous Vehicles”. en. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (May 2018), pp. 187–210. DOI: 10.1146/annurev-control-060117-105157. URL: https://www.annualreviews.org/doi/10.1146/annurev-control-060117-105157 (visited on 12/30/2023).

[2] Christoforos Mavrogiannis et al. “Core Challenges of Social Robot Navigation: A Survey”. en. In: *ACM Transactions on Human-Robot Interaction* 12.3 (Sept. 2023), pp. 1–39. DOI: 10.1145/3583741. URL: https://dl.acm.org/doi/10.1145/3583741 (visited on 02/18/2024).

[3] Wenshuo Wang et al. “Social Interactions for Autonomous Driving: A Review and Perspectives”. en. In: *Foundations and Trends® in Robotics* 10.3-4 (2022). arXiv:2208.07541 [cs], pp. 198–376. DOI: 10.1561/2300000078. URL: http://arxiv.org/abs/2208.07541 (visited on 11/23/2023).

[4] *SAE Levels of Driving Automation™ Refined for Clarity and International Audience*. en. URL: https://www.sae.org/site/blog/sae-j3016-update (visited on 11/15/2024).

[5] Mark Pfeiffer et al. “Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models”. en. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea: IEEE, Oct. 2016, pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329. URL: http://ieeexplore.ieee.org/document/7759329/ (visited on 03/07/2024).

[6] Steffen Hagedorn et al. *Rethinking Integration of Prediction and Planning in Deep Learning-Based Automated Driving Systems: A Review*. en. arXiv:2308.05731 [cs]. Aug. 2023. URL: http://arxiv.org/abs/2308.05731 (visited on 12/30/2023).

[7] Li Chen et al. “End-to-end Autonomous Driving: Challenges and Frontiers”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–20. DOI: 10.1109/TPAMI.2024.3435937. URL: https://ieeexplore.ieee.org/abstract/document/10614862 (visited on 09/08/2024).

[8] Anca D. Dragan et al. “Legibility and predictability of robot motion”. en. In: *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. Tokyo, Japan: IEEE, Mar. 2013, pp. 301–308. DOI: 10.1109/HRI.2013.6483603. URL: http://ieeexplore.ieee.org/document/6483603/ (visited on 01/18/2024).

[9] *Back to the Future: Planning-Aware Trajectory Forecasting for Autonomous Driving | SAIL Blog*. URL: http://ai.stanford.edu/blog/trajectory-forecasting/ (visited on 11/14/2024).

[10] Shuyou Yu et al. “Model predictive control for autonomous ground vehicles: a review”. en. In: *Autonomous Intelligent Systems* 1.1 (Aug. 2021), p. 4. DOI: 10.1007/s43684-021-00005-z. URL: https://link.springer.com/10.1007/s43684-021-00005-z (visited on 04/05/2024).

[11] Hai Zhu et al. “Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments”. en. In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 776–783. DOI: 10.1109/LRA.2019.2893494. URL: https://ieeexplore.ieee.org/document/8613928/ (visited on 10/06/2024).

[12] Bruno Brito et al. *Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments*. en. arXiv:2010.10190 [cs]. Oct. 2020. URL: http://arxiv.org/abs/2010.10190 (visited on 01/14/2024).

[13] Peter Trautman et al. “Unfreezing the robot: Navigation in dense, interacting crowds”. en. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei: IEEE, Oct. 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369. URL: http://ieeexplore.ieee.org/document/5654369/ (visited on 03/07/2024).

[14] Oscar de Groot et al. *Topology-Driven Parallel Trajectory Optimization in Dynamic Environments*. en. arXiv:2401.06021 [cs]. Jan. 2024. URL: http://arxiv.org/abs/2401.06021 (visited on 06/18/2024).

[15] Grady Williams et al. “Model Predictive Path Integral Control: From Theory to Parallel Computation”. en. In: *Journal of Guidance, Control, and Dynamics* 40.2 (Feb. 2017), pp. 344–357. DOI: 10.2514/1.G001921. URL: https://arc.aiaa.org/doi/10.2514/1.G001921 (visited on 03/12/2024).

[16] Moritz Werling et al. “Optimal trajectory generation for dynamic street scenarios in a Frenét Frame”. en. In: *2010 IEEE International Conference on Robotics and Automation*. Anchorage, AK: IEEE, May 2010, pp. 987–993. DOI: 10.1109/ROBOT.2010.5509799. URL: http://ieeexplore.ieee.org/document/5509799/ (visited on 10/05/2024).

[17] Peter Karkus et al. *DiffStack: A Differentiable and Modular Control Stack for Autonomous Vehicles*. en. arXiv:2212.06437 [cs]. Dec. 2022. URL: http://arxiv.org/abs/2212.06437 (visited on 08/29/2024).

[18] Yihan Hu et al. *Planning-oriented Autonomous Driving*. en. arXiv:2212.10156 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2212.10156 (visited on 08/29/2024).

[19] David Gonzalez et al. “A Review of Motion Planning Techniques for Automated Vehicles”. en. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (Apr. 2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841. URL: http://ieeexplore.ieee.org/document/7339478/ (visited on 12/30/2023).

[20] Edward Schmerling et al. *Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction*. en. arXiv:1710.09483 [cs]. Oct. 2017. URL: http://arxiv.org/abs/1710.09483 (visited on 01/05/2024).

[21] Grady Williams et al. *Information Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving*. en. arXiv:1707.02342 [cs]. July 2017. URL: http://arxiv.org/abs/1707.02342 (visited on 02/01/2024).

[22] Dylan M. Asmar et al. *Model Predictive Optimized Path Integral Strategies*. en. arXiv:2203.16633 [cs, eess]. Mar. 2023. URL: http://arxiv.org/abs/2203.16633 (visited on 10/05/2024).

[23] Manan S. Gandhi et al. “Robust Model Predictive Path Integral Control: Analysis and Performance Guarantees”. en. In: *IEEE Robotics and Automation Letters* 6.2 (Apr. 2021), pp. 1423–1430. DOI: 10.1109/LRA.2021.3057563. URL: https://ieeexplore.ieee.org/document/9349120/ (visited on 11/14/2024).

[24] Ziyi Wang et al. *Sampling-Based Optimization for Multi-Agent Model Predictive Control*. en. arXiv:2211.11878 [math]. Nov. 2022. URL: http://arxiv.org/abs/2211.11878 (visited on 02/18/2024).

[25] Elia Trevisan et al. “Biased-MPPI: Informing Sampling-Based Model Predictive Control by Fusing Ancillary Controllers”. en. In: *IEEE Robotics and Automation Letters* 9.6 (June 2024). arXiv:2401.09241 [cs], pp. 5871–5878. DOI: 10.1109/LRA.2024.3397083. URL: http://arxiv.org/abs/2401.09241 (visited on 06/21/2024).

[26] Khaled A. Mustafa et al. *Probabilistic Risk Assessment for Chance-Constrained Collision Avoidance in Uncertain Dynamic Environments*. en. arXiv:2302.10846 [cs]. Feb. 2023. URL: http://arxiv.org/abs/2302.10846 (visited on 01/05/2024).

[27] O. de Groot et al. *Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments*. en. arXiv:2103.12517 [cs]. Mar. 2021. URL: http://arxiv.org/abs/2103.12517 (visited on 01/05/2024).

[28] Yinlam Chow et al. “Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach”. en. In: ().

[29] Ji Yin et al. “Risk-Aware Model Predictive Path Integral Control Using Conditional Value-at-Risk”. en. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United

Kingdom: IEEE, May 2023, pp. 7937–7943. DOI: 10.1109/ICRA48891.2023.10161100. URL: https://ieeexplore.ieee.org/document/10161100/ (visited on 03/07/2024).

[30] Yanjun Huang et al. “A Survey on Trajectory-Prediction Methods for Autonomous Driving”. en. In: *IEEE Transactions on Intelligent Vehicles* 7.3 (Sept. 2022), pp. 652–674. DOI: 10.1109/TIV.2022.3167103. URL: https://ieeexplore.ieee.org/document/9756903/ (visited on 02/28/2024).

[31] Maulik Bhatt et al. “Efficient Constrained Multi-Agent Trajectory Optimization Using Dynamic Potential Games”. In: Oct. 2023, pp. 7303–7310. DOI: 10.1109/IROS55552.2023.10342328.

[32] Simon Le Cleac’h et al. “ALGAMES: A Fast Solver for Constrained Dynamic Games”. en. In: *Robotics: Science and Systems XVI*. arXiv:1910.09713 [cs]. July 2020. DOI: 10.15607/RSS.2020.XVI.091. URL: http://arxiv.org/abs/1910.09713 (visited on 08/29/2024).

[33] Xinjie Liu et al. *Auto-Encoding Bayesian Inverse Games*. en. arXiv:2402.08902 [cs, eess]. Feb. 2024. URL: http://arxiv.org/abs/2402.08902 (visited on 03/07/2024).

[34] Zhiyu Huang et al. *DTPP: Differentiable Joint Conditional Prediction and Cost Evaluation for Tree Policy Planning in Autonomous Driving*. en. arXiv:2310.05885 [cs]. Feb. 2024. URL: http://arxiv.org/abs/2310.05885 (visited on 10/11/2024).

[35] Lucas Streichenberg et al. “Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals”. en. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. arXiv:2302.06547 [cs]. May 2023, pp. 1379–1385. DOI: 10.1109/ICRA48891.2023.10161511. URL: http://arxiv.org/abs/2302.06547 (visited on 02/06/2024).

[36] Walter Jansma et al. *Interaction-Aware Sampling-Based MPC with Learned Local Goal Predictions*. en. arXiv:2309.14931 [cs]. Oct. 2023. URL: http://arxiv.org/abs/2309.14931 (visited on 02/22/2024).

[37] Enric Galceran et al. “Multipolicy Decision-Making for Autonomous Driving via Changepoint-based Behavior Prediction”. en. In: *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, July 2015. DOI: 10.15607/RSS.2015.XI.043. URL: http://www.roboticsproceedings.org/rss11/p43.pdf (visited on 01/05/2024).

[38] Jose L Vazquez et al. “Deep Interactive Motion Prediction and Planning: Playing Games with Motion Prediction Models”. en. In: ().

[39] Yuxiao Chen et al. *Interactive Joint Planning for Autonomous Vehicles*. en. arXiv:2310.18301 [cs]. Nov. 2023. URL: http://arxiv.org/abs/2310.18301 (visited on 10/11/2024).

[40] Dorsa Sadigh et al. “Planning for Autonomous Cars that Leverage Effects on Human Actions”. en. In: *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016. DOI: 10.15607/RSS.2016.XII.029. URL: http://www.roboticsproceedings.org/rss12/p29.pdf (visited on 03/08/2024).

[41] Piyush Gupta et al. “Interaction-Aware Trajectory Planning for Autonomous Vehicles with Analytic Integration of Neural Networks into Model Predictive Control”. en. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 7794–7800. DOI: 10.1109/ICRA48891.2023.10160890. URL: https://ieeexplore.ieee.org/document/10160890/ (visited on 11/14/2024).

[42] Bruno Brito et al. *Learning Interaction-aware Guidance Policies for Motion Planning in Dense Traffic Scenarios*. en. arXiv:2107.04538 [cs]. July 2021. URL: http://arxiv.org/abs/2107.04538 (visited on 01/14/2024).

[43] Bruno Brito et al. “Where to go Next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments”. en. In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 4616–4623. DOI: 10.1109/LRA.2021.3068662. URL: https://ieeexplore.ieee.org/document/9385847/ (visited on 01/14/2024).

[44] Hai Zhu et al. *Learning Interaction-Aware Trajectory Predictions for Decentralized Multi-Robot Motion Planning in Dynamic Environments*. en. arXiv:2102.05382 [cs]. Feb. 2021. URL: http://arxiv.org/abs/2102.05382 (visited on 01/14/2024).

[45] Zhiyu Huang et al. “Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving”. en. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–15. DOI: 10.1109/TNNLS.2023.3283542. URL: https://ieeexplore.ieee.org/document/10154577/ (visited on 12/22/2023).

[46] Tathagata Chakraborti et al. *Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior*. en. arXiv:1811.09722 [cs]. Nov. 2018. URL: http://arxiv.org/abs/1811.09722 (visited on 03/13/2024).

[47] Phani Teja Singamaneni et al. “A survey on socially aware robot navigation: Taxonomy and future challenges”. en. In: *The International Journal of Robotics Research* (Feb. 2024), p. 02783649241230562. DOI: 10.1177/02783649241230562. URL: http://journals.sagepub.com/doi/10.1177/02783649241230562 (visited on 03/07/2024).

[48] Anca Dragan et al. “Integrating human observer inferences into robot motion planning”. en. In: *Autonomous Robots* 37.4 (Dec. 2014), pp. 351–368. DOI: 10.1007/s10514-014-9408-x. URL: http://link.springer.com/10.1007/s10514-014-9408-x (visited on 03/07/2024).

[49] D. Livingston McPherson et al. “An Efficient Understandability Objective for Dynamic Optimal Control”. en. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sept. 2021, pp. 986–992. DOI: 10.1109/IROS51168.2021.9636007. URL: https://ieeexplore.ieee.org/document/9636007/ (visited on 01/19/2024).

[50] Jean-Luc Bastarache et al. “On Legible and Predictable Robot Navigation in Multi-Agent Environments”. en. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 5508–5514. DOI: 10.1109/ICRA48891.2023.10160572. URL: https://ieeexplore.ieee.org/document/10160572/ (visited on 01/19/2024).

[51] Jasper Geldenbott et al. *Legible and Proactive Robot Planning for Prosocial Human-Robot Interactions*. en. arXiv:2404.03734 [cs, eess]. Apr. 2024. URL: http://arxiv.org/abs/2404.03734 (visited on 08/29/2024).

[52] Tim Brüdigam et al. “Legible Model Predictive Control for Autonomous Driving on Highways”. en. In: *IFAC-PapersOnLine* 51.20 (2018), pp. 215–221. DOI: 10.1016/j.ifacol.2018.11.016. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896318326703 (visited on 01/18/2024).

[53] Noriaki Hirose et al. “SACSoN: Scalable Autonomous Control for Social Navigation”. en. In: *IEEE Robotics and Automation Letters* 9.1 (Jan. 2024), pp. 49–56. DOI: 10.1109/LRA.2023.3329626. URL: https://ieeexplore.ieee.org/document/10305270/ (visited on 03/07/2024).

[54] Jingqi Li et al. *Intent Demonstration in General-Sum Dynamic Games via Iterative Linear-Quadratic Approximations*. en. arXiv:2402.10182 [cs, eess]. Feb. 2024. URL: http://arxiv.org/abs/2402.10182 (visited on 03/08/2024).

[55] Christoforos I. Mavrogiannis et al. “Social Momentum: A Framework for Legible Navigation in Dynamic Multi-Agent Environments”. en. In: *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*. Chicago IL USA: ACM, Feb. 2018, pp. 361–369. DOI: 10.1145/3171221.3171255. URL: https://dl.acm.org/doi/10.1145/3171221.3171255 (visited on 03/07/2024).

[56] Peter Bossaerts et al. “Computational Complexity and Human Decision-Making”. English. In: *Trends in Cognitive Sciences* 21.12 (Dec. 2017). Publisher: Elsevier, pp. 917–929. DOI: 10.1016/j.tics.2017.09.005. URL: https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(17)30193-6 (visited on 09/08/2024).

[57] Andrey Rudenko et al. “Human Motion Trajectory Prediction: A Survey”. en. In: *The International Journal of Robotics Research* 39.8 (July 2020). arXiv:1905.06113 [cs], pp. 895–935. DOI: 10.1177/0278364920917446. URL: http://arxiv.org/abs/1905.06113 (visited on 09/17/2024).

[58] Tim Salzmann et al. *Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data*. en. arXiv:2001.03093 [cs]. Jan. 2021. URL: http://arxiv.org/abs/2001.03093 (visited on 02/18/2024).

[59] Christopher M. Bishop et al. *Deep Learning: Foundations and Concepts*. en. Cham: Springer International Publishing, 2024. DOI: 10.1007/978-3-031-45468-4. URL: https://link.springer.com/10.1007/978-3-031-45468-4 (visited on 04/12/2024).

[60] Dirk Helbing et al. *Social Force Model for Pedestrian Dynamics*. arXiv:cond-mat/9805244. May 1998. DOI: 10.48550/arXiv.cond-mat/9805244. URL: http://arxiv.org/abs/cond-mat/9805244 (visited on 11/14/2024).

[61] Jur Van Den Berg et al. "Reciprocal Velocity Obstacles for real-time multi-agent navigation". en. In: *2008 IEEE International Conference on Robotics and Automation*. Pasadena, CA, USA: IEEE, May 2008, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489. URL: http://ieeexplore.ieee.org/document/4543489/ (visited on 11/14/2024).

[62] Christoph Schöller et al. *What the Constant Velocity Model Can Teach Us About Pedestrian Motion Prediction*. en. arXiv:1903.07933 [cs]. Jan. 2020. URL: http://arxiv.org/abs/1903.07933 (visited on 04/05/2024).

[63] Saleh Albeaik et al. *Limitations and Improvements of the Intelligent Driver Model (IDM)*. en. arXiv:2104.02583 [eess]. Apr. 2022. URL: http://arxiv.org/abs/2104.02583 (visited on 11/14/2024).

[64] Jur Van Den Berg et al. "Reciprocal n-Body Collision Avoidance". en. In: *Robotics Research*. Ed. by Bruno Siciliano et al. Vol. 70. Series Title: Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19. DOI: 10.1007/978-3-642-19457-3_1. URL: http://link.springer.com/10.1007/978-3-642-19457-3_1 (visited on 11/14/2024).

[65] Kien Nguyen et al. *Gaussian Process for Trajectories*. en. arXiv:2110.03712 [cs, stat]. Oct. 2021. URL: http://arxiv.org/abs/2110.03712 (visited on 02/28/2024).

[66] Alexandre Alahi et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 961–971. DOI: 10.1109/CVPR.2016.110. URL: http://ieeexplore.ieee.org/document/7780479/ (visited on 11/15/2024).

[67] Renhao Huang et al. *Multimodal Trajectory Prediction: A Survey*. en. arXiv:2302.10463 [cs]. Feb. 2023. URL: http://arxiv.org/abs/2302.10463 (visited on 02/28/2024).

[68] Bruno Brito et al. "Social-VRNN: One-Shot Multi-modal Trajectory Prediction for Interacting Pedestrians". en. In: ().

[69] Shaoshuai Shi et al. *Motion Transformer with Global Intention Localization and Local Movement Refinement*. en. arXiv:2209.13508 [cs]. Mar. 2023. URL: http://arxiv.org/abs/2209.13508 (visited on 06/18/2024).

[70] Jiquan Ngiam et al. *Scene Transformer: A unified architecture for predicting multiple agent trajectories*. en. arXiv:2106.08417 [cs]. Mar. 2022. URL: http://arxiv.org/abs/2106.08417 (visited on 12/30/2023).

[71] Nigamaa Nayakanti et al. *Wayformer: Motion Forecasting via Simple & Efficient Attention Networks*. arXiv:2207.05844. July 2022. DOI: 10.48550/arXiv.2207.05844. URL: http://arxiv.org/abs/2207.05844 (visited on 11/14/2024).

Part II

Scientific Article

Predictability Awareness For Efficient and Robust Multi-Agent Coordination

Roman Chiva Gil
Delft University Of Technology
Delft, Netherlands
R.ChivaGil@student.tudelft.nl

Daniel Jarne Ornia*†
University of Oxford
Oxford, United Kingdom
daniel.jarneornia@cs.ox.ac.uk

Khaled A. Mustafa*
Delft University Of Technology
Delft, Netherlands
k.a.mustafa@tudelft.nl

Javier Alonso Mora
Delft University Of Technology
Delft, Netherlands
j.alonsomora@tudelft.nl

ABSTRACT

To safely and efficiently solve motion planning problems in multi-agent settings, most approaches attempt to solve a joint optimization that explicitly accounts for the responses triggered in other agents. This often results in solutions with an exponential computational complexity, making these methods intractable for complex scenarios with many agents. While sequential predict-and-plan approaches are more scalable, they tend to perform poorly in highly interactive environments. This paper proposes a method to improve the interactive capabilities of sequential predict-and-plan methods in multi-agent navigation problems by introducing predictability as an optimization objective. We interpret predictability through the use of general prediction models, by allowing agents to predict themselves and estimate how they align with these external predictions. We formally introduce this behavior through the free-energy of the system, which reduces (under appropriate bounds) to the Kullback-Leibler divergence between plan and prediction, and use this as a penalty for unpredictable trajectories. The proposed interpretation of predictability allows agents to more robustly leverage prediction models, and fosters a 'soft social convention' that accelerates agreement on coordination strategies without the need of explicit high level control or communication. We show how this predictability-aware planning leads to lower-cost trajectories and reduces planning effort in a set of multi-robot problems, including autonomous driving experiments with human driver data, where we show that the benefits of considering predictability apply even when only the ego-agent uses this strategy. The code and experiment videos can be found in the following page: https://romanchiva.github.io/PAProjectPage/

KEYWORDS

Multi-Agent Systems, Motion Planning, Autonomous Navigation, Coordination, Prediction Models

1 INTRODUCTION

Many modern robotics applications involve autonomous agents navigating multi-agent environments where they will be required to interact with humans and other robots without full knowledge or extensive communication capabilities [39]. This involves planning trajectories in a complex system governed by a mix of rational and non-rational, stochastic and possibly game theoretic behaviors. To achieve safe and efficient interactions, agents need to reason about each other and coordinate. However, this poses critical challenges due to the high uncertainty associated with estimating other agent's objectives [16] and a computational complexity that renders problems intractable for more than a handful of agents.

Receding Horizon Trajectory Optimization allows for flexible and anticipative planning while ensuring compliance with *e.g.* safety constraints in multi-agent navigation problems. However, planning a trajectory that explicitly accounts for interactions among agents generally requires solving a joint optimization problem. A variety of joint planning methods can be found in literature, *e.g.* [10, 22], of which game theoretic approaches best capturing agent interaction complexities [39]. By modelling other agents as rational actors, game theoretic approaches cast the joint optimization as a constrained dynamic game and seek to find equilibrium solutions. Although this often results in stable and coordinated interactions [11, 16], game theoretic approaches suffer from the curse of dimensionality, as the planning complexity grows exponentially with the number of agents [34]. Additionally, modelling other agents as rational is a strong assumption which will not hold in practice, especially when interacting with human agents [4, 12].

Alternatively, predict-and-plan approaches scale well with number of agents, however they tend to perform poorly in interactive environments. By separating prediction and planning, the problem simplifies to a single-agent collision avoidance problem with dynamic obstacles [5, 13]. The accuracy of the prediction model limits how well agents can coordinate. A system of interacting agents is highly complex, making it difficult to predict the diversity of possible futures, especially when considering interactions. This can lead to ambiguous predictions, making agents unable to anticipate their environment, and thus have to re-plan more often or engage in riskier behaviors [39].

Ideally, every agent in the environment would be able to accurately anticipate surrounding agents' future trajectories allowing for efficient and safe interaction. Sequential planning agents use prediction models to avoid collisions with others, however, this fails to acknowledge that surrounding agents also hold predictions about the ego-agent, and plan their trajectory based on these predictions. Unless the optimal avoidance strategy falls within the

* ***Indicates equal contribution***, *This research is supported by funding from the Dutch Research Council NWO-NWA, within the "Acting under Uncertainty" (ACT) project (Grant No.NWA.1292.19.298),* † *Work done partially while at Delft University of Technology. Author acknowledges partial support from UKRI grant EP/W002949/1. .*

range of predicted behaviors, other agents will react to the unexpected avoidance strategy by modifying their own trajectory. To mitigate this issue, we propose the following: in the same way a prediction model is used to predict other agents, the ego-agent can use it to approximate how other agents expect it to behave. This information can be used in planning to introduce a penalty for trajectories other agents will find surprising, bringing the optimal trajectory closer to the expectation surrounding agents hold. Accounting for predictability in this way mirrors the principle of free-energy minimization in active inference [33] (and control systems [36]), where an agent not only seeks to maximize reward but also aims to minimize the discrepancy between some prediction model and observations. In multi-agent interactions [24], agents hold probabilistic beliefs about the behavior of others, and the accuracy of these beliefs is directly influenced by the agent's own actions. By minimizing free energy, the agent balances actions that reduce uncertainty and confirm its internal model of the world with those that maximize reward. This approach ensures that the agent's behavior is not only goal-directed but also aligned with maintaining coherent and accurate beliefs about the surrounding agents.

1.1 Contribution

We explore how sequential planning agents can improve their coordination capabilities by accounting for the predictability of their planned trajectories. When a group agents accounts for predictability, they are able to foster a 'soft social convention' dictated by the prediction model which results in a decrease of uncertainty about the environment for all agents in the group. This helps agents resolve coordination problems without having to explicitly model interactions. Formally, the contribution of this paper is threefold:

(1) We exploit ideas on free-energy to formulate a cost function that uses feedback from a prediction model to include predictability as an objective and analyze how this cost function can be integrated with a planner.
(2) We provide results showing how our predictability awareness mechanism leads to 'soft social conventions' forming-based interaction strategies encoded in prediction models for multi-robot navigation problems. This allows agents to achieve smoother coordination by improving the effectiveness of prediction models in interactive environments.
(3) Accounting for predictability causes agents to adopt *social norms* and pro-social behaviors encoded in learned prediction models, allowing to more closely mimic experts' behaviors without needing cost function learning. We provide evidence for these behaviors in an experiment where an agent interacts with human drivers in scenarios from the Waymo Open Motion Dataset.

2 RELATED WORK

Integration of Prediction Models and Planners. Trajectory prediction has significantly advanced in recent years, particularly with the development of transformer-based generative models capable of producing interaction-aware joint trajectory predictions, *e.g.* [8, 9, 21]. While these models show impressive performance in open-loop evaluations, integrating them with planners in highly interactive settings remains challenging [20]. Effective interactive planning often necessitates joint prediction and planning. Additionally, the planner often requires some form of learned cost function [23]. Otherwise, if the behavior of the expert significantly differs from the expert in the training data, this will throw the model out of distribution yielding low quality predictions.

Many studies have focused on developing ego-conditioned prediction models [30]; however, their integration with planners faces obstacles primarily due to computational complexity. For instance, in [22] Tree Policy Planning (TPP) has been employed to generate an initial set of partial trajectories, which condition the prediction model and create a scenario tree. This tree is evaluated using a cost function combining designed and learned features to identify and expand promising scenarios, efficiently allocating computational resources. A novel approach by [10] leverages unconditioned prediction models to provide initial estimates of other agents' trajectories, capitalizing on the models' ability to predict general intentions accurately while acknowledging their limitations in capturing short-term interaction details. This approach optimizes the ego and agent trajectories together, minimizing disturbances from the initial agent paths and utilizing homotopy classes to ensure diversity and avoid local minima. Instead of conditional prediction models, some methods develop fully differentiable stacks [23, 26] enabling gradient backpropagation through the planner, which allows for combined prediction model fine-tuning and cost function learning aligned with expert behavior in the training data. While avoiding the joint optimization, our approach links prediction and planning without the need for retraining or fine-tuning by including a term in the cost function that helps guide the agent's behavior to not compromise its predictions. This allows for maintaining flexibility in selecting prediction models and planner combinations while being compute-efficient.

Predictability and Legibility of Motion. In the field of Human-Robot Interaction, legibility and predictability of motion have been studied to improve coordination by designing agent behaviors that clearly communicate intention and avoid surprising observers [15]. Often both objectives overlap [3]. Traditional formulations of this problem are not well suited for receding horizon applications as they optimize over complete trajectories and rely on utility-based analytical models of observer expectations [14]. Additionally, the observer is modeled as inactive, thus having no influence on the planning agent. This assumption breaks down in multi-agent navigation where the observer and the agent share the workspace and influence each other. Several works have explored the adaptation of these concepts to an interactive multi-agent context. [2] define dynamic goal regions around neighboring agents and optimize for reduced uncertainty about the collision avoidance strategy. [19] show how increasing action penalties at later horizon steps causes agents to more rapidly demonstrate their avoidance strategy. This accelerates intent inference giving agents better anticipation. [6] defines hand-crafted legibility costs for planning in highway driving. These methods are often designed to target a specific type of observer model. In contrast, our approach minimizes a predictability surrogate that allows modeling the observers with an arbitrary prediction model choice.

3 TRAJECTORY PLANNING

The general optimization problem for a single-agent in stochastic motion planning can be formulated as follows:

$$\min_{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}} \quad \sum_{k=0}^{K-1} J_k(\boldsymbol{x}_k,\boldsymbol{u}_k) + J_K(\boldsymbol{x}_K) \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \tag{1b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k), \quad k = 0,...,K-1 \tag{1c}$$

$$\mathbb{P}\left[C(\boldsymbol{x}_k,\boldsymbol{\delta}_k^o),\forall o\right] \geq 1-\epsilon_k,\forall k, \tag{1d}$$

where $\boldsymbol{u} = \{\boldsymbol{u}_0,...,\boldsymbol{u}_K\} \in \mathbb{U}$ are the system inputs subject to input constraints, $\boldsymbol{x}_k \in \mathbb{X}$ denotes the states of the robot, $f(\cdot)$ corresponds to the nonlinear system's dynamics, $J_k(\boldsymbol{x}_k,\boldsymbol{u}_k) \geq 0$ is the cost function specifying performance metrics, and K is the length of the planning horizon. In this formulation, $C(\cdot)$ is the collision avoidance constraint, and $\boldsymbol{\delta}_k^o$ is the uncertain position of obstacle o at stage k obtained through a prediction model $\mathcal{P}(\mathbf{X})$ that takes into account the concatenated states of all agents in the scene. The chance constraint in Eq. (1d), guarantees that the probability that the robot collides with the dynamic obstacle is below a specified threshold ϵ_k.

In a game theoretic setting where all agents are controlled by a centralized planner, the problem reduces to solving a joint optimization program over all agents and all possible trajectories, such that from the set of joint trajectories that satisfy the constraints, the agents execute the optimal ones. This naturally carries high computational complexity, access to some centralized controller, and full information assumptions. Consider instead the case where N (interactive) agents solve the optimization problem (1) independently and use model $\mathcal{P}(\mathbf{X})$ to predict each other (and thus estimate the probabilities of constraint satisfaction). Agents can then query the prediction model to observe the predictions others have about them. Our method reduces to the following intuition: Agents can use this information to shift their behaviors towards the distribution coming out of the prediction model. This 'closes the loop' on prediction errors, intuitively improving the planning problem in two ways. First, inducing implicit decentralized coordination: an ideal situation is one where all agents act following the model-predicted distribution, and this distribution perfectly optimizes the cost of each agent. Second, it 'robustifies' the prediction model *a posteriori*: once the model has been trained on offline data, agents actively shift their plans towards the predicted distributions, collectively reducing prediction errors and widening the space of suitable prediction models for a given problem.

4 PROPOSED METHOD: FREE ENERGY AS A PREDICTABILITY SURROGATE

4.1 Derivation of a predictability aware cost function

Our objective is to design a framework that allows agents to trade off predictability with progress toward the goal. If we define an agent's optimal trajectory distribution as Q^*, in the best-case scenario, an agent's optimal trajectory distribution aligns with the predictions held by other agents. This alignment allows the agent to minimize its own cost while avoiding any disruption or interference with the trajectories of surrounding agents. In this case, no trade-off needs to be performed, however, deviations from this ideal scenario are to be expected. To formalize this as a planning objective, agents should seek to minimize the cost of trajectories sampled from their corresponding prediction in $\mathcal{P}(\mathbf{X})$. Drawing inspiration from the path integral control derivation in [42], we begin by defining the free energy of a trajectory distribution:

$$\mathcal{F}(S,\mathcal{P},\lambda) = -\lambda\log(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\exp(-\frac{1}{\lambda}S(\boldsymbol{x}))]),$$

where S is a state cost function that *represents some (trajectory planning) objective*, $\mathcal{P}$ denotes a prediction distribution, x is a trajectory sampled from $\mathcal{P}$, and λ represents the inverse temperature controlling the strictness of the efficiency criterion. This control theoretic free energy can be interpreted as a measure of how efficient a prediction distribution is at minimizing cost S. The free energy is minimized by pushing $\mathcal{P}$ as close as possible to Q^*.

The free energy as defined so far is a function of prediction distribution $\mathcal{P}$, however, agents won't plan trajectories by sampling from $\mathcal{P}$. Instead, we define Q as a trajectory distribution an agent has control over. Let the states $\boldsymbol{x} = \{\boldsymbol{x}_0,...,\boldsymbol{x}_K\}$, which the ego-agent occupies along its planned trajectory $\tau_{0,K}$, be represented as narrow Gaussians $q(\boldsymbol{x}_k)$ with mean $\boldsymbol{x}_k$ covariance Σ_k:

$$\begin{aligned}\tau_{0:K} &= \{q(\boldsymbol{x}_k)\}_{k=0}^K,\\ q(\boldsymbol{x}_k) &= \mathcal{N}(\boldsymbol{x}_k,\Sigma_k).\end{aligned} \tag{2}$$

By applying an expectation switch, these distributions can be incorporated into the free energy definition, making it a function of the agent's plan,

$$\mathcal{F}(S,\mathcal{P},\lambda) = -\lambda\log(\mathbb{E}_{\boldsymbol{x}\sim Q}[\exp(-\frac{1}{\lambda}S(\boldsymbol{x}))\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}]), \tag{3}$$

where p is the density function of the prediction. By concavity of the logarithm and Jensen's inequality,

$$\mathcal{F}(S,\mathcal{P},\lambda) \leq -\lambda\mathbb{E}_{\boldsymbol{x}\sim Q}[\log(\exp(-\frac{1}{\lambda}S(\boldsymbol{x}))) + \log(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})})].$$

Finally, using the definition of Kullback-Leibler Divergence and simplifying,

$$\mathcal{F}(S,\mathcal{P},\lambda) \leq \mathbb{E}_{\boldsymbol{x}\sim Q}[S(\boldsymbol{x})] + \lambda\mathbb{KL}(q(\boldsymbol{x})||p(\boldsymbol{x})), \tag{4}$$

where $\mathbb{KL}$ denotes the KL-Divergence. The right-hand side provides an upper bound on the free energy, and one can minimise this instead of the free energy. It resembles a standard control objective, and the terms allow for good conceptual understanding of the effect they have: A **Performance Cost** and **Predictability Cost** respectively, which penalizes agents for acting unpredictably. Using this newly found expression as a stage cost, we can craft the following cost function as a stage cost for a planning problem:

$$J(\tau_{0:K}) = \sum_{k=0}^{K} J_k(\boldsymbol{x}_k,\boldsymbol{u}_k) + \lambda\mathbb{KL}(q(\boldsymbol{x}_k)||p(\boldsymbol{x}_k)),$$

where we implicitly assume J_k to be composed by some state cost S and some control action cost. Minimizing this cost function allows agents to trade off predictability and progress toward the

goal by means of the free energy, and λ can be selected to control how much weight is assigned to predictability during planning.

Remark 1. *We can emphasize now the intuition behind using the free energy as a way of incorporating predictability into optimal control. Eq.* (4) *is minimised precisely when $Q = Q^* = \mathcal{P}$. That is, the trajectory distribution executed is exactly the optimal cost trajectory distribution, and this matches the predicted distribution. Under this condition, the agent is behaving without surprising external observers and simultaneously obtaining optimal cost in its objective.*

Further insights on the interpretation of free energy in this context and the derivation of the free energy term are provided in Appendix A.

4.2 Integration with a Planner and Practicalities

The KL-Divergence expression only has closed form solutions for a restricted set of distributions, thus to accommodate arbitrary distributions, the KL divergence term will often need to be evaluated through sampling with Q the candidate trajectory distribution and P the prediction distribution from $\mathbb{KL}(\mathcal{P}\|Q) = \mathbb{E}_{\boldsymbol{x}\sim P}\left[\log \frac{\mathcal{P}(\boldsymbol{x})}{Q(\boldsymbol{x})}\right]$. Since sampling is required to evaluate the cost function, this could render the use of gradient based MPC unfeasible for real time planning, additionally prediction distributions $Q(\boldsymbol{x})$ may not always be differentiable. We find it is more practical to rely on sampling based MPC approaches, as they don't require a differentiable cost function and computations can be easily parallelized to handle large numbers of samples even when it is computationally expensive to evaluate the cost function. In our experiments, Section 5, we rely on an Model Predictive Path Integral (MPPI) control method [42].

Another consideration is that predictions about an agent's future are updated as new observations are received. For this reason, it is most effective to focus on early horizon time-steps when evaluating a plan's predictability. Thus we propose to discount the predictability cost along the horizon with factor γ to account for uncertainty about future predictions:

$$J(\tau_{0:K}) = \sum_{k=0}^{K} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \gamma^t \lambda \mathbb{KL}(q(\boldsymbol{x}_k)||p(\boldsymbol{x}_k)). \tag{5}$$

5 EXPERIMENTS

We present here the experiments carried out to validate our method. The first experiment investigates how accounting for predictability affects an individual agent's behavior, comparing the results with other observer-aware planning approaches. The second experiment examines the impact of predictability within a group of agents, focusing on swapping tasks in an open environment to give insight without external environmental influences. In the third experiment, we explore a practical driving scenario, demonstrating how predictability-aware agents can better coordinate and utilize prediction models. We also observe that agents indirectly exhibit expert-like behaviors, such as following social norms, without explicitly encoding them in the planner. Finally, the fourth experiment explores this direction further by testing interactions with recorded human driver data using a state-of-the-art prediction model, showing that predictability-aware agents achieve safer trajectories as a result of more closely mimicking human behavior.

5.1 Planner

For all experiments in this section, we use a sampling-based planner, namely Model Predictive Path Integral (MPPI) control, based on the methodology presented in [42]. MPPI places no restrictions on dynamics model or cost function and converges well toward optima with a moderate amount of samples [41]. Given a nominal control sequence as an initial guess, MPPI applies Gaussian noise at each step to generate a set of M control sequence samples. It then uses a state transition function $f(\cdot)$ to simulate their corresponding M state trajectories. Each of the resulting state trajectories is evaluated based on the cost defined in (23), resulting in a total sample cost J_m. Once J_m, $\forall m \in [1, ..., M]$ is computed, importance sampling weights, w_m, can be calculated as:

$$w_m = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(J_m - J_{\min})\right), \quad \sum_{m=1}^{M} w_m = 1,$$

where $J_{\min}$ is the minimum sampled cost, η is a normalization factor and λ is a controlling parameter that controls the width of the weight distribution. These weights prioritize lower-cost trajectories. The optimal control sequence U^* is then calculated as the weighted sum of all sampled control sequences:

$$U^* = \sum_{m=1}^{M} w_m U_m,$$

where we use $U_m = \{\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_K\}$ to denote the m-th sampled control sequence. It is common to use a time-shifted version of U^* to warm-start the sampling strategy at the next time-step. Further details on the planner and cost functions used in the experiments can be found in Appendix C.

5.2 Metrics

As a proxy to measure coordination, we propose the use of planning effort. Planning effort is a metric taken from [7] to quantify how much trajectories deviate from an initial estimate. The authors point out this serves as a proxy for how well the agent is able to anticipate the evolution of its surroundings. We adapt planning effort for receding horizon tasks with the following formulation:

$$\begin{aligned} PE(\xi_{0:T}) &= \frac{1}{T-1} \sum_{t=0}^{T-1} MSE(\tau_t, \tau_{t+1}), \text{ with} \\ MSE(\tau_t, \tau_{t+1}) &= \sum_{k=0}^{K} \|\boldsymbol{x}_k^t - \boldsymbol{x}_k^{t+1}\|, \end{aligned} \tag{6}$$

where T and K denote the total time duration of the simulation and planning horizon length respectively. $\xi_{0:T}$ denotes the set of all the plans along a trajectory $\xi_{0:T} = \{\tau_0, \tau_1, \ldots, \tau_{T-1}\}$: with τ_t the plan at time-step t. $\boldsymbol{x}_k^t$ represents the state at horizon step k for the plan τ_t. In this context, planning effort measures, on average, the magnitude of an agent's plan update per time-step. Generally, for a given task, a more accurate prediction model corresponds to a lower planning effort.

5.3 Single Agent Experiments

Experiment Objective. In this experiment, we present a single agent interacting with a hand-crafted multi-modal prediction

model, serving as a model of an observer's expectation. This is a benchmark task used by previous works on legibility and predictability [14], [29] to provide clear insight into the relationship between predictability and the agent's intrinsic motivation.

Setup. Consider an environment with two possible goals: $\mathcal{G} = \{A : [20, 10], B : [20, -10]\}$. The robot starts at position $\boldsymbol{x}_0 = [0, 0]$ and is tasked with reaching goal B. The predictions model an uncertain observer that holds mistaken initial beliefs $\mathcal{B}$ about the agents goals: $b_0^A = 0.7$ and $b_0^B = 0.3$. Based on these beliefs a Gaussian Mixture $p_t(\boldsymbol{x})$ is used as a prediction, with each mode assuming a Constant Velocity (CV) trajectory towards its respective goal. For timestep t at each horizon step k:

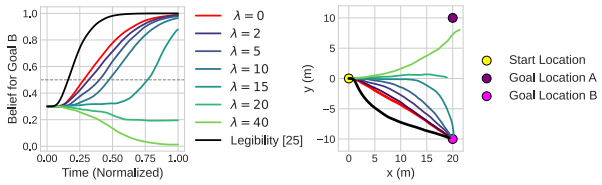
$$p_{t,k}(\boldsymbol{x}) = \sum_{g \in \mathcal{G}} b_t^g p_{t,k}(\boldsymbol{x}), \tag{7}$$

where $p_{t,k}(\boldsymbol{x}) = \mathcal{N}(\mu_t^g, \Sigma)$ with $\mu_{t,k}^g$ is the CV prediction for goal $g \in \mathcal{G}$ at horizon step k, Σ is a fixed covariance and $\boldsymbol{x}$ is a state. We model the observer's changing beliefs $\mathcal{B}$ via Bayesian inference. With every new observation, beliefs are updated using the mode predictions $p_{t,k}(\boldsymbol{x})$ as likelihood functions:

$$b_t^g = \frac{p_{t-1,0}(\boldsymbol{x}_t) b_{t-1}^g}{\sum_{g \in \mathcal{G}} p_{t-1,0}(\boldsymbol{x}_t) b_{t-1}^g}. \tag{8}$$

Keeping a fixed discount $\gamma = 0.6$ in Eq. (23), we vary the magnitude of λ to generate results shown in Figure 1.

Results Discussion. We use this example to study how λ should be tuned to control the trade-off. If predictability dominates (e.g., $\lambda = 20$ or $\lambda = 40$), this results in observations that further reinforce the observer's mistaken belief. It becomes more costly for the robot to pursue its intrinsic motivation with each time-step, thus it fails to complete the task. Conversely, if λ is too low, the robot may still behave unpredictably[1]. For reference, the resulting behavior of an



(a) Belief updates over trajectories. **(b) Trajectories for different λ.**

Figure 1: Figure 1a shows that increasing λ effectively decrease the belief update rate for the observer. In Figure 1b, the nominal trajectory is rendered in red. Given the observer holds mistaken initial beliefs about the robot's goal, we observe that increasing the predictability score λ results in trajectories that are more compliant with the observer's expectation.

agent optimizing for *legibility* as per the method of [29] is shown as the black line in Figures 1b and 1a. From the perspective of coordination, [29] can be understood as an anticipatory mechanism: By conveying intention in advance, other agents anticipate better in their planning. Our approach similarly mitigates sudden environmental changes, however instead of aiming to directly influence the other agents' beliefs, we rely on a prediction model to avoid the surprising observations throughout the interaction. While this can occasionally result in slightly more costly trajectories for the agent, we achieve similar results without requiring explicit modeling of the other agent, making it more computationally efficient and robust to situations where the agent may not be able to successfully convey its intention. As demonstrated in Figure 1, when the observer's beliefs are misaligned, the agent adopts a pro-social behavior, gently guiding the observer toward the correct belief.

5.4 Robot-Robot Interactions

5.4.1 *Swapping Tasks.*

Experiment Objective. These experiments explore the benefits of accounting for predictability in robot-robot interactions through swapping-tasks, a common benchmark for robot coordination [3], [44]. By performing tests in an open environment these tests avoid interference of external environmental influences.

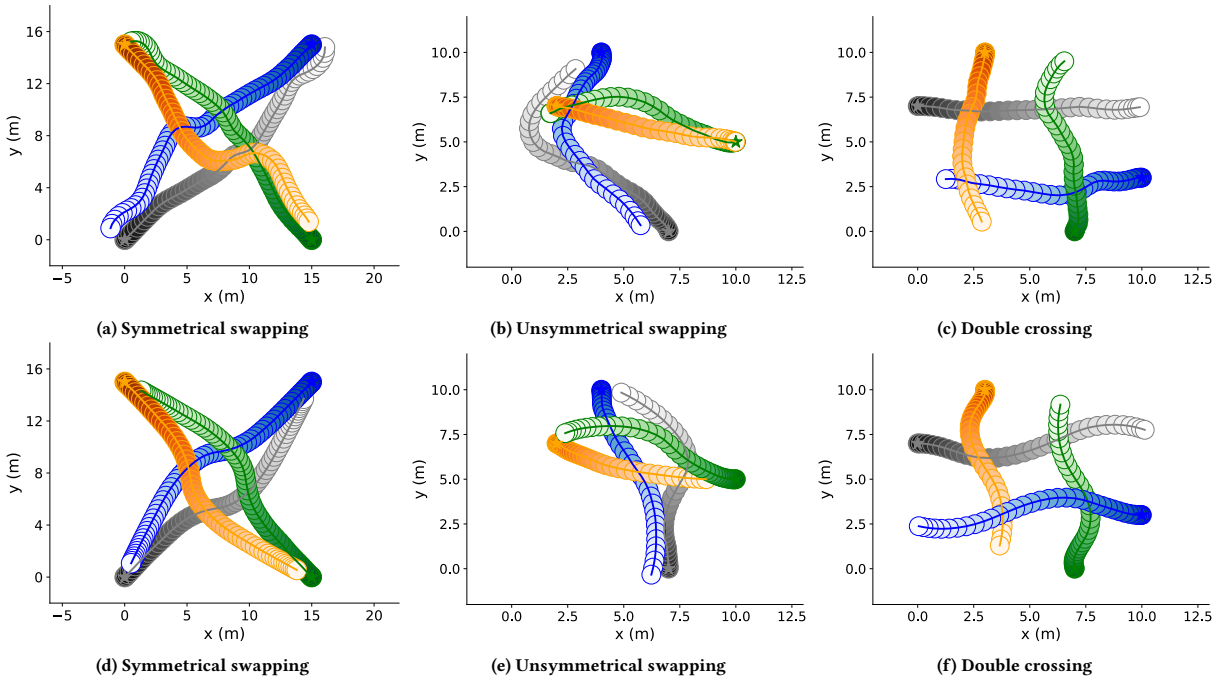
Setup. In the experiments, agents are initially positioned on the vertices of a square and tasked with swapping positions with the agent on the opposite vertex (Figure 2a). The optimal solution requires all agents to coordinate by selecting the same collision avoidance strategy, either passing left or right. Additionally, two more scenarios were tested: an asymmetrical swapping task and a double-crossing task, to explore different geometries and interactions. The experiments use a game-theoretic prediction model based on the ALGAMES framework [11], which solves constrained dynamic games to find an optimal joint strategy over a 20-step horizon. Further explanation on Game Theoretic Planning and details on the ALGAMES solver can be found in Appendix B. The model generates prediction distributions for each horizon step as a Gaussian with user-specified covariance Σ. By testing three values of the predictability parameter λ {0.0, 2.5, 5.0}, we investigate how accounting for predictability impacts agent coordination. Each task was run 50 times, and the results for all three tasks are reported in Table 1 [2]. An illustration comparing the trajectories for all 3 scenarios can be found in Figure 2.

Table 1: Table summarizing results for the 3 swapping tasks: Symmetrical, Unsymmetrical, and Double-Crossing

Exp.	Metric	$\lambda = 0.0$	$\lambda = 2.5$	$\lambda = 5.0$
Sym	PE (m²)	2.116 ±1.000	0.516 ±0.161	**0.501 ± 0.145**
	Acc (m/s²)	0.209 ±0.101	**0.038 ± 0.008**	0.043 ±0.009
	Ang (rad/s)	0.283 ±0.041	0.225 ±0.026	**0.219 ± 0.026**
Unsym	PE (m²)	0.877 ±0.489	0.291 ±0.178	**0.187 ± 0.162**
	Acc (m/s²)	0.196 ±0.114	0.138 ±0.090	**0.112 ± 0.080**
	Ang (rad/s)	0.363 ±0.112	0.221 ±0.095	**0.177 ± 0.071**
D-Cross	PE (m²)	0.969 ±0.416	0.388 ±0.124	**0.311 ± 0.116**
	Acc (m/s²)	0.249 ±0.124	**0.123 ± 0.080**	0.125 ±0.070
	Ang (rad/s)	0.434 ±0.128	0.283 ±0.096	**0.252 ± 0.078**

[1] In practice, the influence seems to be very dependent on the structure of the main objective cost function, so we recommend tuning λ empirically based on the specific planner and prediction model used.

[2] For $\lambda = 0$ safety constraint violations are low at 1-2 for all the tasks. For higher λ it was 0 for all tasks. As this is not a very informative result it was not included in the tables



(a) Symmetrical swapping (b) Unsymmetrical swapping (c) Double crossing

(d) Symmetrical swapping (e) Unsymmetrical swapping (f) Double crossing

Figure 2: The first row shows the results with $\lambda = 0$ whereas the second row shows the results for $\lambda = 5.0$. When agents account for predictability, aside from faster convergence to a coordination strategy, this also results in smoother trajectories as a consequence of better anticipation of the environment.

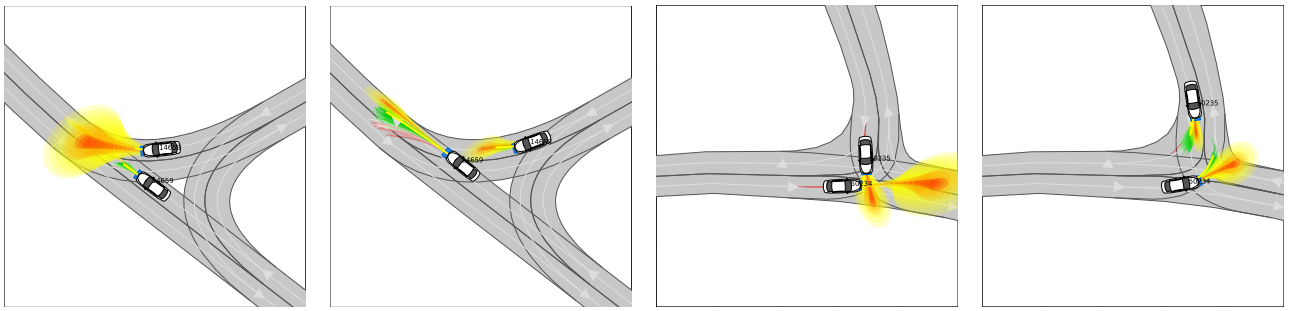
Results Discussion. As seen in Table 1, increasing the predictability parameter λ consistently led to improved performance across all metrics: planning effort (PE), acceleration (Acc), and angular velocity (Ang). Notably, even selecting a small λ causes a pronounced decrease in planning effort, with further increases in λ yielding diminishing returns. This phenomenon can be attributed to the coordination challenge agents face in this environment, which primarily involves equilibrium selection. In situations where agents must choose between two equally viable strategies, such as passing left or passing right, our method addresses this challenge by relying on a prediction model to establish a 'soft social convention'. This introduces a subtle bias towards one of the strategies, improving implicit coordination. This mechanism is particularly relevant, as prediction models often excel at capturing an agent's overarching intent and high-level strategy. However, equilibrium selection scenarios are inherently stochastic and unpredictable, making them challenging to model accurately [39]. Thus, our method enhances robustness in such situations by guiding agents towards a coordinated strategy selected by the prediction model. In general, agents need a precise and accurate prediction model for efficient coordination. However, due to the inherent uncertainty of interactions, this is often very hard to achieve. By accounting for predictability, a group of agents is able to establish a 'soft social convention' to mitigate some of this uncertainty. From the perspective of an agent, this results in more accurate predictions, allowing for smoother and more efficient coordination. This mechanism is especially effective for interactions where the main coordination challenge lies in equilibrium selection.

5.4.2 *Robot-Robot Traffic Scenario.*

Experiment Objective. In this experiment, we focus on robot-robot coordination in driving scenarios, where the environment has a stronger influence on agent's behavior. This time, we use a data-driven prediction model to explore how predictability impacts coordination in more complex environments. To test the robustness of our method when combined with a different planner we additionally performed an experiment using a Frenet frame planner. This experiment used the same environment and cost function. The results and discussion for this additional experiment can be found in Appendix D.

Setup. We use CommonRoad [1] as a simulator, which includes the Wale-Net [17] prediction model, a learning-based model that outputs predictions as Gaussians, accounting for uncertainty, road geometry, and the interaction with surrounding agents. Consistent with previous experiments, we employ an MPPI based planner. To account for safety in planning, we implement the constraints introduced by [18], building upon and extending the code from this prior work. We perform tests in two scenarios: A T-Junction and a Lane-Merge. For both scenarios, we perform tests with $\lambda = \{0.0, 2.5, 5.0\}$ for 30 iterations applying small changes in the initial positions and velocities. An illustration of the lane merge environment is presented in Figure 3. The results for T-Junction and lane-merge are presented in Table 2.

Results Discussion. When agents fail to coordinate in road scenarios, they often experience deadlocks or, in the worst case,



(a) Lane merge $\lambda = 0$ (b) Lane Merge $\lambda = 5$ (c) T-Junction $\lambda = 0$ (d) T-Junction $\lambda = 5$

Figure 3: a) Illustration of a deadlock With $\lambda = 0$, where a sequence of faulty predictions reinforces both agent's hesitation. b) For $\lambda = 5$, the agents can leverage the prediction model to coordinate which agent gives way and which passes first.

Table 2: Results for T-Junction and Lane Merge Scenarios (Dlk indicates Deadlocks)

Exp	Metric	$\lambda = 0.0$	$\lambda = 2.5$	$\lambda = 5.0$
T-J	Dlk (%)	30.0	**0.0**	**0.0**
	Dist (m)	30.172	**51.248**	47.000
	PE (m^2)	**1.366** ±1.126	2.318 ±0.313	2.507 ±0.759
	Acc (m/s^2)	-0.142 ±0.238	**0.293** ±0.045	0.287 ±0.233
	Ang (rad/s)	0.0037 ±0.0028	**0.0005** ±0.0002	0.0028 ±0.0026
LM	Dlk (%)	73.3	**0.0**	**0.0**
	Dist (m)	46.878	**75.800**	69.909
	PE (m^2)	**2.079** ±0.785	3.513 ±0.564	3.315 ±0.760
	Acc (m/s^2)	**0.111** ±0.092	0.337 ±0.054	0.317 ±0.087
	Ang (rad/s)	0.0032 ±0.0032	0.0009 ±0.0006	**0.0001** ±0.0001

collisions. In Figure 8a, an example of a deadlock is illustrated. Deadlocks are common in limited space environments such as intersections or narrow passages. Initially, the model may predict one agent will yield while the other advances. However, as deviations occur and both agents hesitate, their predictions begin to reinforce each other's hesitation, creating the deadlock. The model may then be unable to introduce asymmetry to prioritize one of the agents in ambiguous situations, preventing the agents from breaking away from the deadlock. Results show that agents incorporating predictability into their models achieve better coordination, as indicated by less pronounced slowdowns resulting in higher travelled distance and the *disappearance of deadlocks* as seen in Table 2. When examining other metrics, the benefits of incorporating predictability are not as pronounced, especially for higher λ. This occurs because the prediction model is not explicitly conditioned to align with the road geometry (Figures 8c,8d). Since the planner is required to track a reference path, deviations between the predictions and the reference path can push the agent to deviate from the path, requiring small adjustments more frequently for higher higher λ. Although this problem has marginal impact on the overall performance of the agent, the reduction in planning effort may be mitigated.

Similar to the Swapping Task tests, we see that agents are able to use the prediction model to coordinate by reducing uncertainty on equilibrium selection, namely, which agent gives way. However, a noteworthy observation is that, beyond reducing uncertainty, agents enhance their performance by adopting pro-social behaviors embedded in the model's latent space. These behaviors include adherence to social norms and subtle cues learned from training data, mirroring the behavior of experts used to train the model. This behavior resembles imitation learning, where agents learn cooperative strategies directly from expert demonstrations embedded in the prediction model. As seen in Figure 8b, although both outcomes are equally plausible from the raw planning problem, agents consistently converge on the solution where the merging agent yields, which aligns with typical human driving patterns.

5.5 Experiments with human-driver data

Experiment Objective. The goal of this experiment is to evaluate whether predictability can bridge the gap between algorithmic planning and the natural driving patterns observed in humans, facilitating smoother and more adaptive interactions in complex driving environments. We test this by incorporating predictability with simple MPPI-based reference-tracking planner using a SOTA data-driven prediction model.

Setup. We utilize a state-of-the-art (SOTA) prediction model introduced by [23], a multi-modal, transformer-based architecture trained on the Waymo Open Motion Dataset. The model generates scene-centric predictions with three modes, representing the most likely joint trajectories of up to 11 agents, including the ego agent. An MPPI planner is used for reference tracking, incorporating collision avoidance as outlined in [23]. For this experiment, we replay recorded scenes from the Waymo dataset's test set, meaning agents in the environment follow pre-recorded, non-interactive trajectories. The goal is for the ego agent to replicate expert behavior observed during training. We perform tests for $\lambda = 0, 75, 120$[3], over 30 iterations in selected scenarios that require human-like interactions, similar to the approach of [23]. A screenshot of the crossing scenario is shown in Figure 4, and results are reported in Table 3.

Results Discussion. From Table 3, it is evident that increasing the weight of the predictability objective results in fewer collisions and smoother control inputs. Interestingly, however, this does not necessarily correlate with improved progress along the reference

[3]The large λ values here respond to the particular magnitude of the planning cost function and the prediction model used. We found that values of a higher order of magnitude were needed to obtain predictable behavior shifts.

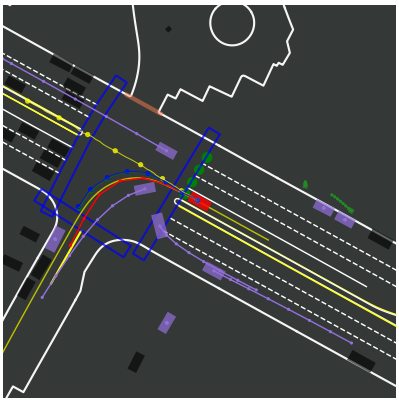


Figure 4: Illustration of the navigation problem in Crossing1. The reference global path is rendered as a smooth yellow line. The AV's plan is rendered in red. Predictions for other agents are rendered in purple, showing only the most likely mode for clarity. The ego-prediction is multi-modal with 3 modes represented by the yellow, green, and blue trajectories.

Table 3: Results comparing the performance of an MPPI-based planner on Waymo Open Motion Dataset scenarios for different λ values. For 30 iterations we present the number of collisions and the mean value of other performance metrics

Scenario	λ	Col (%)	Dist (m)	Acc (m/s²)	Lat_Acc (m/s²)	L2 (m)
Crossing1	0	43.3	74.540 ±1.928	1.085 ±0.121	1.615 ±0.578	4.101 ±0.532
	75	0	68.353 ±0.982	0.681 ±0.025	0.412 ±0.044	3.358 ±0.221
	120	0	55.796 ±1.321	0.472 ±0.035	0.149 ±0.025	2.554 ±0.053
Crossing2	0	86.6	75.843 ±4.803	1.418 ±0.099	2.121 ±0.295	12.707 ±1.112
	75	0	55.353 ±1.403	0.957 ±0.091	0.314 ±0.036	4.603 ±1.246
	120	23.3	37.518 ±12.954	1.534 ±0.736	0.227 ±0.084	2.947 ±3.206
Intersection	0	26.6	69.747 ±4.794	1.450 ±0.153	1.817 ±0.782	24.808 ±3.632
	75	0	72.700 ±0.115	0.709 ±0.029	0.493 ±0.075	24.618 ±1.036
	120	0	71.885 ±0.227	0.613 ±0.043	0.304 ±0.026	19.900 ±0.733
Emergency	0	53.3	61.377 ±20.244	1.258 ±0.175	0.882 ±0.141	8.631 ±5.877
	75	0	68.883 ±0.525	0.763 ±0.010	0.365 ±0.031	1.961 ±0.069
	120	0	60.058 ±0.797	0.581 ±0.019	0.184 ±0.016	1.515 ±0.091

path. This can be attributed to the planner inducing less distributional shift in the prediction model. The model, trained on scenes where all agents exhibit expert behaviors, struggles when the planner deviates significantly from these patterns, as it encounters situations outside its training distribution. In such cases, the model attempts to extrapolate and produces sub-optimal predictions, such as incorrectly anticipating that an agent may yield or maneuver differently than it actually does based on the recorded data. This misalignment leads to overconfident behavior in some instances, which, while promoting progress along the reference path, increases the risk of collisions. Evidence supporting this hypothesis is found in the Human L2 loss metric, which measures the L2 loss between the agent's trajectory and the corresponding human trajectory that the planner aims to replicate. For $\lambda = 0$, the higher L2 loss indicates significant deviation from human behavior, suggesting that the agent diverges more from the expert's trajectory. In contrast, when predictability is considered, the L2 loss decreases, indicating that the agent's behavior aligns more closely with the human data. This results in reduced distributional shift and, consequently, more accurate predictions and smoother trajectories.

6 DISCUSSION

Discussion. The method assumes that agents can approximate each other's expectations, often implying a shared prediction model. Although this might seem impractical, certain decentralized settings could accommodate shared models. For instance, in a warehouse environment where multiple Autonomous Ground Vehicles (AGVs) transport valuable goods, a shared prediction model could be feasibly developed and implemented [44]. When integrated with our methodology, such a model could establish 'operational norms', enabling agents to coordinate efficiently and robustly without the need for centralized control, thus reducing computational and infrastructure demands. A comparable scenario is anticipated in future markets where autonomous vehicles (AVs) from different manufacturers must interact. Recent studies pointed at the importance of establishing a unified driving convention [43], as the absence of such a standard could lead to exploitative strategies from different AV companies pursuing competitive advantage, and thereby compromise safety. Different companies can cooperate to develop a shared prediction model to serve as an industry standard. Given a model all AVs in traffic share, our method would enable AVs to anticipate each other's actions and more effectively settle on coordination, thereby providing enhanced road safety without explicit coordination or reliance on infrastructure for centralized coordination.

Future Work. Balancing predictability and performance cost, determined by λ, is complex and context-dependent. Dynamically adjusting λ as agents interact could improve performance, increasingly prioritizing predictability in safety-critical moments. Developing adaptive heuristics for this adjustment, as suggested by previous work [2, 14], would be a valuable research direction. The free energy framework in Section 4 could be extended to dynamically adjust prediction distributions over time in reaction to the samples, enhancing anticipation and balancing. However, continuous updates bring computational challenges, especially in multi-agent systems, making scalability difficult. Simplifications or assumptions may help overcome these challenges, merging predictability and legibility to boost performance. A more detailed discussion of this future work direction as well as preliminary experimental results can be found in Appendix E.

Conclusion. We present a novel approach to enhance multi-agent interaction capabilities for sequential predict-and-plan frameworks by introducing predictability as a key optimization objective. Accounting for predictability in this manner can be understood as an implicit cooperation mechanism whereby agents use a prediction model to actively reduce uncertainty about the environment for other agents. This not only improves the robustness of coordination strategies but also reduces planning effort without requiring

explicit communication or high-level control, and does so independently of the number of interacting agents. Through experiments, including robot-robot interactions and human-interaction scenarios, our method improved agent coordination, reduced collisions, and led to smoother, more efficient trajectories (particularly in complex coordination environments). We also demonstrated that the benefits extend to interactions with human drivers by allowing the agent to more reliably use its prediction model.

REFERENCES

[1] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. 2017. CommonRoad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*. 719–726. https://doi.org/10.1109/IVS.2017.7995802

[2] Jean-Luc Bastarache, Christopher Nielsen, and Stephen L. Smith. 2023. On Legible and Predictable Robot Navigation in Multi-Agent Environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 5508–5514. https://doi.org/10.1109/ICRA48891.2023.10160572

[3] Maulik Bhatt, Yixuan Jia, and Negar Mehr. 2023. Efficient Constrained Multi-Agent Trajectory Optimization using Dynamic Potential Games. https://doi.org/10.48550/arXiv.2206.08963 arXiv:2206.08963 [cs].

[4] Peter Bossaerts and Carsten Murawski. 2017. Computational Complexity and Human Decision-Making. *Trends in Cognitive Sciences* 21, 12 (Dec. 2017), 917–929. https://doi.org/10.1016/j.tics.2017.09.005 Publisher: Elsevier.

[5] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. 2020. Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments. http://arxiv.org/abs/2010.10190 arXiv:2010.10190 [cs].

[6] Tim Brüdigam, Kenan Ahmic, Marion Leibold, and Dirk Wollherr. 2018. Legible Model Predictive Control for Autonomous Driving on Highways. *IFAC-PapersOnLine* 51, 20 (2018), 215–221. https://doi.org/10.1016/j.ifacol.2018.11.016

[7] Daniel Carton, Wiktor Olszowy, and Dirk Wollherr. 2016. Measuring the Effectiveness of Readability for Mobile Robot Locomotion. *International Journal of Social Robotics* 8, 5 (Nov. 2016), 721–741. https://doi.org/10.1007/s12369-016-0358-7

[8] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. 2020. Implicit latent variable model for scene-consistent motion forecasting. *European Conference on Computer Vision (ECCV)* (Sept. 2020). https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123680613.pdf

[9] Yuxiao Chen, Boris Ivanovic, and Marc Pavone. 2022. Scept: Scene-consistent, policy-based trajectory predictions for planning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Sept. 2022). https://doi.org/10.1109/CVPR52688.2022.01659

[10] Yuxiao Chen, Sushant Veer, Peter Karkus, and Marco Pavone. 2023. InteractiveMotionPlanning for AutonomousVehicle with Joint Optimization. http://arxiv.org/abs/2310.18301v2

[11] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. 2020. ALGAMES: A Fast Solver for Constrained Dynamic Games. In *Robotics: Science and Systems XVI*. https://doi.org/10.15607/RSS.2020.XVI.091 arXiv:1910.09713 [cs].

[12] Andrew M. Colman. 2003. Cooperation, psychological game theory, and limitations of rationality in social interaction. *The Behavioral and Brain Sciences* 26, 2 (April 2003), 139–153; discussion 153–198. https://doi.org/10.1017/s0140525x03000050

[13] Oscar de Groot, Laura Ferranti, Dariu Gavrila, and Javier Alonso-Mora. 2023. Scenario-Based Motion Planning with Bounded Probability of Collision. http://arxiv.org/abs/2307.01070 arXiv:2307.01070 [cs].

[14] Anca Dragan and Siddhartha Srinivasa. 2014. Integrating human observer inferences into robot motion planning. *Autonomous Robots* 37, 4 (Dec. 2014), 351–368. https://doi.org/10.1007/s10514-014-9408-x

[15] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. 2013. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, Tokyo, Japan, 301–308. https://doi.org/10.1109/HRI.2013.6483603

[16] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. 2018. Hierarchical Game-Theoretic Planning for Autonomous Vehicles. http://arxiv.org/abs/1810.05766 arXiv:1810.05766 [cs, math].

[17] Maximilian Geisslinger, Phillip Karle, Johannes Betz, and Markus Lienkamp. 2021. Watch-and-Learn-Net: Self-supervised Online Learning for Probabilistic Vehicle Trajectory Prediction. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 869–875. https://doi.org/10.1109/SMC52423.2021.9659079 ISSN: 2577-1655.

[18] Maximilian Geisslinger, Franziska Poszler, and Markus Lienkamp. 2023. An ethical trajectory planning algorithm for autonomous vehicles. *Nature Machine Intelligence* 5 (Feb. 2023), 1–8. https://doi.org/10.1038/s42256-022-00607-z

[19] Jasper Geldenbott and Karen Leung. 2024. Legible and Proactive Robot Planning for Prosocial Human-Robot Interactions. http://arxiv.org/abs/2404.03734 arXiv:2404.03734 [cs, eess].

[20] Steffen Hagedorn, Marcel Hallgarten, Martin Stoll, and Alexandru Condurache. 2023. Rethinking Integration of Prediction and Planning in Deep Learning-Based Automated Driving Systems: A Review. http://arxiv.org/abs/2308.05731 arXiv:2308.05731 [cs].

[21] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. 2022. A Survey on Trajectory-Prediction Methods for Autonomous Driving. *IEEE Transactions on Intelligent Vehicles* 7, 3 (Sept. 2022), 652–674. https://doi.org/10.1109/TIV.2022.3167103

[22] Zhiyu Huang, Peter Karkus, Boris Ivanovic, Yuxiao Chen, Marco Pavone, and Chen Lv. 2024. DTPP: DifferentiableJointConditionalPrediction and Cost Evaluation for Tree Policy Planning in Autonomous Driving. http://arxiv.org/abs/2310.05885v2

[23] Zhiyu Huang, Haochen Liu, Jingda Wu, and Chen Lv. 2023. Differentiable Integrated Motion Prediction and Planning With Learnable Cost Function for Autonomous Driving. *IEEE Transactions on Neural Networks and Learning Systems* (2023), 1–15. https://doi.org/10.1109/TNNLS.2023.3283542

[24] David Hyland, Tomáš Gavenčiak, Lancelot Da Costa, Conor Heins, Vojtech Kovarik, Julian Gutierrez, Michael J. Wooldridge, and Jan Kulveit. 2024. Free-Energy Equilibria: Toward a Theory of Interactions Between Boundedly-Rational Agents. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*. https://openreview.net/forum?id=4Ft7DcrjdO

[25] Xiaosong Jia, Liting Sun, Masayoshi Tomizuka, and Wei Zhan. 2020. IDE-Net: Interactive Driving Event and Pattern Extraction from Human Data. https://doi.org/10.48550/arXiv.2011.02403 arXiv:2011.02403.

[26] Peter Karkus, Boris Ivanovic, Shie Mannor, and Marco Pavone. 2022. DiffStack: A Differentiable and Modular Control Stack for Autonomous Vehicles. http://arxiv.org/abs/2212.06437 arXiv:2212.06437 [cs].

[27] Gustav Markkula and Mehmet Dogar. 2024. Models of human behavior for human-robot interaction and automated driving: How accurate do the models of human behavior need to be? *IEEE Robotics & Automation Magazine* 31, 3 (Sept. 2024), 115–120. https://doi.org/10.1109/MRA.2022.3182892 arXiv:2202.06123 [cs].

[28] Pietro Mazzaglia, Tim Verbelen, Ozan Çatal, and Bart Dhoedt. 2022. The Free Energy Principle for Perception and Action: A Deep Learning Perspective. *Entropy* 24, 2 (Feb. 2022), 301. https://doi.org/10.3390/e24020301 arXiv:2207.06415 [cs].

[29] D. Livingston McPherson and S. Shankar Sastry. 2021. An Efficient Understandability Objective for Dynamic Optimal Control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Prague, Czech Republic, 986–992. https://doi.org/10.1109/IROS51168.2021.9636007

[30] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David Weiss, Ben Sapp, Zhifeng Chen, and Jonathon Shlens. 2022. Scene Transformer: A unified architecture for predicting multiple agent trajectories. http://arxiv.org/abs/2106.08417 arXiv:2106.08417 [cs].

[31] Thomas Parr, Giovanni Pezzulo, and Karl J. Friston. 2022. *Active inference: the free energy principle in mind, brain, and behavior*. The MIT Press, Cambridge, Massachusetts London, England.

[32] C. Pezzato. 2023. *Exploring Active Inference and Model Predictive Path Integral Control*. Ph.D. Dissertation. Delft University of Technology. https://doi.org/10.4233/UUID:4FA3A292-477C-4FF0-B01A-E7D90B66EC2A

[33] Noor Sajid, Philip J Ball, Thomas Parr, and Karl J Friston. 2021. Active inference: demystified and compared. *Neural computation* 33, 3 (2021), 674–712.

[34] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. 2018. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* 1, 1 (May 2018), 187–210. https://doi.org/10.1146/annurev-control-060117-105157

[35] Yuchen Sun, Dongchun Ren, Shiqi Lian, Mingyu Fan, and Xiangyi Teng. [n.d.]. *An Efficient Generation Method based on Dynamic Curvature of the Reference Curve for Robust Trajectory Planning*. https://doi.org/10.48550/arXiv.2012.14617

[36] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. 2010. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research* 11 (2010), 3137–3181.

[37] Evangelos A. Theodorou. 2015. Nonlinear Stochastic Control and Information Theoretic Dualities: Connections, Interdependencies and Thermodynamic Interpretations. *Entropy* 17, 5 (May 2015), 3352–3375. https://doi.org/10.3390/e17053352 Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[38] Evangelos A. Theodorou and Emanuel Todorov. 2012. Relative entropy and free energy dualities: Connections to Path Integral and KL control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, Maui, HI, USA, 1466–1473. https://doi.org/10.1109/CDC.2012.6426381

[39] Wenshuo Wang, Letian Wang, Chengyuan Zhang, Changliu Liu, and Lijun Sun. 2022. Social Interactions for Autonomous Driving: A Review and Perspectives. *Foundations and Trends® in Robotics* 10, 3-4 (2022), 198–376. https://doi.org/10.1561/2300000078 arXiv:2208.07541 [cs].

[40] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. [n.d.]. Optimal trajectory generation for dynamic street scenarios in a Frenét Frame. In *2010 IEEE International Conference on Robotics and Automation* (Anchorage, AK, 2010-05). IEEE, 987–993. https://doi.org/10.1109/ROBOT.2010.5509799

[41] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. 2017. Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics* 40, 2 (Feb. 2017), 344–357. https://doi.org/10.2514/1.G001921

[42] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. 2017. Information Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. http://arxiv.org/abs/1707.02342 arXiv:1707.02342 [cs].

[43] Xiaojuan Yu, Vincent A.C. Van Den Berg, Erik T. Verhoef, and Zhi-Chun Li. 2022. Will all autonomous cars cooperate? Brands' strategic interactions under dynamic congestion. *Transportation Research Part E: Logistics and Transportation Review* 166 (Oct. 2022), 102825. https://doi.org/10.1016/j.tre.2022.102825

[44] Hai Zhu, Francisco Martinez Claramunt, Bruno Brito, and Javier Alonso-Mora. 2021. Learning Interaction-Aware Trajectory Predictions for Decentralized Multi-Robot Motion Planning in Dynamic Environments. http://arxiv.org/abs/2102.05382 arXiv:2102.05382 [cs].

APPENDIX A: BACKGROUND ON FREE ENERGY AND LINK TO ACTIVE INFERENCE

Background on Free-Energy

In thermodynamics and statistical physics, the concept of *free energy* embodies a balance between a system's internal energy and entropy, with temperature acting as a regulator of this balance. The evolution of a system can then be modeled as balancing energy and entropy so as to minimize free energy. Depending on the system's properties different formulations of the free energy are used. For systems at a constant termperature and volume the *Helmholtz free energy* F is used as a potential function to quantify free energy:

$$F = U - TS, \tag{9}$$

where U is the internal energy, T is the temperature, and S is the entropy of the system. This equation formulates how systems naturally evolve toward states that minimize free energy to reach equilibrium. The internal energy U tends to be minimized, while the entropy S (the number of accessible system states), tends to be maximized. Temperature T modulates the relative importance of energy and entropy. For example, at higher temperatures, maximizing the entropy term TS becomes a more prominent behavior. An alternative formulation of the Helmholtz free energy can be written in terms of a *partition function* Z:

$$F = -kT \ln Z \tag{10}$$

where k is the Boltzmann constant. The partition function Z is defined as:

$$Z = \int e^{-E(x)/kT}\, dx \tag{11}$$

where $E(x)$ represents the energy of state x. The partition function integrates over all possible states, weighing each with the Boltzmann factor $e^{-E/kT}$, which favors lower-energy states but allows for higher-energy states at higher temperatures, acting similar to a soft-max (random fluctuations to higher energy states become more likely at higher temperatures).

The ideas encapsulated by free energy minimization are popular in optimizing trade-offs beyond physical systems. This framework allows for principled decision-making in applications where competing factors must be balanced optimally and has been applied across various domains. In stochastic control for instance, it is common to design controllers that balance performance and robustness by minimizing a cost function analogous to free energy [37]. In machine learning, especially for variational inference, models are often trained to minimize a free energy-like objective that balances model accuracy with model complexity [28]. In neuroscience, the free energy principle provides a framework for understanding a wide range of cognitive and neural processes, highlighting how the brain optimizes behavior by balancing the trade-off between exploring new information and exploiting existing knowledge [31].

Derivation of Free Energy as a Stage Cost for Stochastic Optimal Control

In stochastic optimal control, the objective is to find a control policy that minimizes the expected cost $S(x)$ in the presence of stochastic dynamics. Analogies can be drawn with the free energy minimization framework introduced previously. Namely:

- **Energy vs. Cost**: The energy $E(x)$ function is analogous to the cost $S(x)$ in the context of control.
- **Temperature vs. Noise Level**: The temperature T can be compared to the noise level λ in control, representing the uncertainty in the state transitions.
- **Partition Function vs. Path Integral**: The partition function Z captures a sum over all possible states of the system. For control, this quantity would be analogous to a path integral, integrating over all possible states or control inputs.

Starting from the partition function in statistical mechanics, it is possible to derive an analogous expression for control systems inspired by [37][38][36][32], where we derive the free energy expression as a stage cost for a planning problem:

$$Z = \int e^{-E(x)/kT}, dx. \tag{12}$$

Assuming $P(x)$ is a probability distribution over states x (or trajectories), it can represent the stochastic dynamics of the system or a sampling distribution over possible control inputs. In turn, the partition function can be re-formulated as an expectation over $P(x)$:

$$Z = \int e^{-E(x)/kT}, dx = \int \frac{e^{-E(x)/kT}}{P(x)} P(x), dx = \mathbb{E}_{x\sim P}\left[\frac{e^{-E(x)/kT}}{P(x)}\right]. \tag{13}$$

In the expectation formulation of the partition-function, replacing the energy $E(x)$ with the cost function of the control problem $S(x)$ and kT with λ, representing the noise level or control temperature, we arrive at a control theoretic formulation of the partition function, namely a path integral:

$$Z = \mathbb{E}_{x\sim P}\left[e^{-S(x)/\lambda}\right]. \tag{14}$$

This expression computes the expected value of the exponentiated negative scaled cost over trajectories x drawn from P. Analogous to the Helmholtz free energy in physics: $F = -kT \ln Z$, it is possible to define the free energy for the control theoretic context as:

$$\mathcal{F}(S, P, \lambda) = -\lambda \ln\left(\mathbb{E}_{x\sim P}\left[e^{-S(x)/\lambda}\right]\right) \tag{15}$$

We can identify this expression as the free energy introduced in the methodology. By minimizing this control-theoretic free energy $\mathcal{F}(S, P, \lambda)$, we effectively balance performance (through the cost $S(x)$) and robustness (through the noise level λ and distribution P). A lower λ places more emphasis on cost minimization, favoring trajectories with lower cost but potentially less robustness to uncertainty. A higher λ increases the weight of the entropy-like term, promoting exploration and robustness by considering a broader range of trajectories.

As a practical example, consider an environment where an agent is tasked with navigating from a start to a goal location with an obstacle in the path. Due to stochastic transition dynamics, the agent estimates a distribution of possible trajectories as a result of its nominal trajectory. An illustration of this scenario is provided in Figure 5. Colliding with the obstacle results in a high cost for the agent. Trajectories corresponding to different values of λ are shown

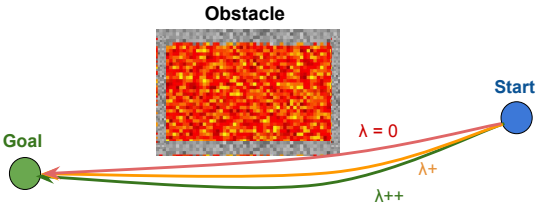


Figure 5: An agent minimizing a free energy cost function is tasked with navigating from a start to a goal location while avoiding an obstacle. The transition dynamics for the environment are stochastic, and thus an agent's plan results in a distribution around the nominal trajectory. Increasing λ results in increasingly risk-averse behaviors, pushing the plan further from the obstacle to minimize the probabilities of collision.

in green, orange, and red, representing $\lambda = 0$ and increasing values of λ, respectively. By minimizing the free energy in this manner, the agent evaluates a nominal path based on the exponentially weighted cost of sampled trajectories, with lower cost trajectories more heavily weighted.

The parameter λ regulates the strictness of this trade-off: for $\lambda = 0$, the agent assigns all weight to the lowest cost trajectory, prioritizing cost minimization over robustness. As λ increases, the agent assigns weight to a a broader range of trajectories, enhancing robustness by preferring distributions that better avoid the probability of high-cost collisions. This demonstrates how adjusting λ allows the agent to balance between cost minimization and robust performance under stochastic dynamics.

Incorporation of a Prior Distribution

In equation 15, $P(X)$ is treated as the state distribution the agent has control over. However, in certain scenarios, there may be prior information about this state distribution, such as a reference path, the agent is expected to follow. Using expectation switch, we can rewrite the expectation over $P(x)$ as an expectation over a distribution $Q(X)$, which the agent has control over:

$$F(S, P, \lambda) = -\lambda \log \left(\mathbb{E}_{x \sim Q} \left[\exp\left(-\frac{1}{\lambda} S(x) \right) \frac{p(x)}{q(x)} \right] \right), \qquad (16)$$

where $p(x)$ and $q(x)$ are the probability densities under P and Q, respectively. This formulation allows for minimizing the free energy while balancing two competing objectives: minimizing the expected cost and staying close to the prior. For example, in [42], they use the uncontrolled path distribution as a prior, which represents the trajectory of the agent when no control input is applied (i.e., minimal control effort is expected). By minimizing this free energy term, the agent optimizes its trajectory to align with the prior P, effectively placing itself in a position where it would be acting optimally by following the prior, in this case by applying minimal control inputs. In this manner introducing the prior is the equivalent of introducing a control cost.

The desired effect of this formulation can be viewed more clearly if we apply some transformations to the function. Applying Jensen's inequality to the concave logarithm function and using the definition of the Kullback-Leibler divergence yields the following upper bound for the free energy term:

$$-\lambda F(S, P, \lambda) \leq \mathbb{E}_{x \sim Q} \left[S(x) \right] + \lambda D_{\mathrm{KL}}(q(x) \,\|\, p(x)) \qquad (17)$$

This upper bound can then be used as an intuitive cost function in planning. The term $\mathbb{E}_{x \sim Q} \left[S(x) \right]$ represents the expected cost under the controlled distribution, reflecting the agent's intrinsic motivation. The KL divergence term $\lambda D_{\mathrm{KL}}(q(x) \,\|\, p(x))$ penalizes deviations from the prior distribution. This formulation allows for a comprehensive understanding of how this formulation allows for a principled trade-off between performance and adherence to prior beliefs about the system's behavior, regulated by λ.

For our case, we propose to use the outputs from a prediction model, queried on the ego agent's state history, as a prior. Similar to how the prior used in [42] , serves as a control cost, we propose that using the output from a prediction model as a prior can serve as a predictability cost. A justification for this choice is presented in the following subsection.

Prediction Models as Priors

Let us assume an environment with several interacting agents using a shared prediction model $P(x)$ to predict each other's trajectories in order to coordinate during interactions. Under our assumption, we treat other agents as sequential planning agents. By querying the model, an agent can gain access to the expectations others have about it. In this way, the prediction model serves to capture the natural or expected unfolding of an interaction, representing what everyone is anticipating. It functions as a 'social convention' or a manual that an agent can query to understand how it is supposed to behave or how it has led others to believe it will behave.

Deviating from this shared prediction incurs a *social cost*, as it results in surprising trajectories that could potentially force other agents to re-plan their own trajectories. Therefore, an agent should aim to align its actions as closely as possible with its own best interests while not surprising others. By avoiding unexpected behaviors, it is more likely that others will also act in ways that are predictable and align with the agent's expectations, and thereby facilitating coordination for both parties. Thus, a socially aware agent should plan to influence the predictions of the model about itself to match its optimal trajectory as closely as possible. In this manner, it can pursue its intrinsic motivation without incurring a social cost. Mathematically, we can formalize this objective by interpreting the output from the prediction model as the prior (as discussed in the previous section), which introduces a *predictability cost*. Minimizing this cost compels an agent to balance cost minimization with remaining predictable by adjusting the prior to align with its optimal trajectory. By employing the free energy cost formulation to achieve this, the lowest free energy state for the system corresponds to all agents following their desired optimal trajectories, as captured by the shared prediction model.

This approach offers two primary benefits: improved coordination among agents due to the 'social convention' induced by the shared prediction model accessible to all agents, and enhanced accuracy of the prediction model itself, as agents align their behaviors with predictable trajectories.

Link to Active Inference

Link to Active Inference. Active Inference, originating from research on cognition and neuroscience, is a theoretical framework proposing that agents perceive and act in their environment by continuously minimizing the *variational free energy* of their internal world model. This framework aligns closely with our approach to multi-agent coordination, where the objective is to balance the minimization of expected costs with adherence to prior distributions. Prior beliefs are specified by the internal world model, or by the shared prediction model in our approach.

In Active Inference, agents update their internal model based on sensory input to minimize the discrepancies between predictions and observations. The key idea is that this goal can be achieved via two complimentary mechanisms: *perception*, where the agent refines its beliefs to better account for incoming data, and *action*, where the agent interacts with the environment to fulfill its predictions. Mathematically, this objective is formalized via minimizing variational free energy, which serves as an upper bound on the surprise of observations (the negative log probability of observations). This objective effectively balances perception and action by naturally trading them off. Consequently, Active Inference addresses a fundamental trade-off similar to the exploration-exploitation trade-off commonly encountered in reinforcement learning.

To further reinforce these points we can take a look at the mathematical formulation of variational free energy provided by [31]:

$$
\begin{aligned}
F[Q, y] &= \underbrace{-\mathbb{E}_{Q(x)}[\ln P(y, x)]}_{\text{Energy}} \underbrace{- H[Q(x)]}_{\text{Entropy}} \\
&= \underbrace{D_{KL}[Q(x) \| P(x)]}_{\text{Complexity}} \underbrace{- \mathbb{E}_{Q(x)}[\ln P(y \mid x)]}_{\text{Accuracy}} \\
&= \underbrace{D_{KL}[Q(x) \| P(x \mid y)]}_{\text{Divergence}} \underbrace{- \ln P(y)}_{\text{Evidence}}
\end{aligned}
$$

$Q(x)$ represents the approximate posterior distribution over hidden states x, which the agent uses to estimate the hidden causes of the observed data. It is a variational approximation of the true posterior $P(x \mid y)$ because the true posterior is often intractable to work with (The environment or model is often highly complex). $P(y, x)$, often referred to as the generative model, is the joint probability distribution that combines the prior beliefs ($P(x)$) about the hidden states and the likelihood ($P(y \mid x)$) of observations y given those states. Finally, y denotes the observations. It can be seen that the variational free energy objective can be formulated in 3 possible ways, each offering a unique perspective:

- **Energy and Entropy** Minimizing free energy requires consistency between the approximate posterior $Q(x)$ and the generative model $P(y, x)$ (energy term), while maintaining high posterior entropy. High entropy ensures maximal uncertainty in the absence of data, aligning with the maximum entropy principle by adopting the least committed belief.
- **Complexity and Accuracy:** Free energy minimization balances reducing complexity (KL divergence between $Q(x)$ and $P(x)$) and maximizing accuracy (how well $Q(x)$ explains $P(y \mid x)$). This trade-off captures Occam's razor principle, favoring explanations that are both simple and effective.
- **Divergence and Evidence:** This formulation is the most relevant to our discussion. It shows that free energy can be reduced either by improving the model $Q(X)$ or by changing future observations y to align better with the agent's expectations. Crucially, this means that free energy can be minimized through planning, where the agent actively selects behaviors that lead to observations consistent with its generative model. This is important because it highlights how in this case minimizing free energy integrates planning and learning into a single task. Instead of passively updating beliefs about the world, the agent can actively shape its environment to confirm its beliefs.

The third formulation, expressed in terms of Divergence and Evidence, provides a crucial realization: rather than solely focusing on developing increasingly accurate predictive models, the actions of the agent itself play a large role in determining how successful the model can be in practice. In the context of sequential planning in multi-agent environments, this becomes particularly relevant. When an agent is equipped with a pre-trained prediction model, further minimization of the Divergence term is no longer feasible. However, this opens up the possibility of improving performance in the environment by leveraging a sub-optimal model more effectively. Through principled action selection, the agent can influence the environment in a manner that enhances the alignment between predictions and actual outcomes, thereby achieving greater accuracy despite the limitations of the model. In particular, principled action selection would involve behaving predictably for others so they remain predictable for the ego as discussed in the section above.

Relevance for the use of Prediction Models. This argument links nicely with some remarks provided in the survey by [39] on social interactions for autonomous vehicles. The survey highlights that while trajectory forecasting has been an indispensable part of safety-critical interactive system design, the focus on ever-increasing accuracy is yielding diminishing returns. Early improvements in trajectory prediction (e.g., from 40% to 80%) required relatively low effort, but further gains (e.g., less than 3%) now demand extensive fine-tuning and optimization [25]. Furthermore, pursuing model accuracy alone does not necessarily improve overall interaction performance. Instead, successful interaction often hinges on understanding key aspects of cognition and behavior, such as goals, objectives, and social compatibility [27]. For example, humans, despite their less precise predictive abilities, achieve efficient interactions through goal-based reasoning and most importantly social alignment. Thus, rather than striving for maximal accuracy, it is often more effective to focus on understanding and incorporating socially compatible behaviors. Thus we motivate our approach based on this perspective, highlighting the importance of principled action selection in achieving successful interactions in multi-agent environments. We propose that predictability-aware action is a key aspect of achieving socially compatible behaviors.

APPENDIX B: GAME-THEORETIC PLANNING AND THE ALGAMES SOLVER

Background on Receding Horizon Game Theoretic Planning

Game theory provides a powerful framework for modeling and solving interactive multi-agent decision-making problems. In these problems, agents aim to optimize their own objectives while explicitly accounting for the interdependence of decisions among the agents. For interactive path planning applications, the problem is cast as a trajectory game, a subset of dynamic games where agents strategies are represented as trajectories over time, and are coupled by being subject to collision avoidance and kinematic feasibility constraints. One of the central solution concepts in game theory is the *Nash equilibrium*. At a Nash equilibrium, no player can improve their outcome by unilaterally changing their strategy, assuming other players maintain theirs.

ALGAMES

The ALGAMES (Augmented Lagrangian GAME-theoretic Solver) algorithm is a specialized approach for solving trajectory games developed by [11]. It relies on an augmented Lagrangian formulation of the optimization problem aiming to account for non-linear state and action constraints commonly encountered in trajectory games. These constraints introduce a coupling between the optimization problems of different players, and their inclusion changes the solution concept from a standard Nash equilibrium to a Generalized Nash Equilibrium (GNE), where the strategies of players must satisfy shared constraints in addition to individual optimality. Traditional game-theoretic optimizations often do not account for constraints, and their inclusion significantly complicates the optimization process. However, by using Lagrangian multipliers, ALGAMES reformulates the problem by transforming it into an unconstrained optimization again. This allows the algorithm to efficiently find Generalized Nash Equilibria while enforcing the shared constraints.

Each player ν aims to minimize a cost function $J^\nu(X, U^\nu)$ subject to shared dynamics and constraints:

$$\min_{X,U^\nu} \quad J^\nu(X, U^\nu), \tag{18}$$

$$\text{s.t.} \quad D(X, U) = 0, \tag{19}$$

$$C(X, U) \leq 0. \tag{20}$$

Where X represents the state trajectory of the system, U^ν denotes the control inputs of player ν, $D(X, U) = 0$ defines the system dynamics (Equality Constraints) and $C(X, U) \leq 0$ enforces collision avoidance constraints (inequality constraints). The solver then defines the augmented Lagrangian for player ν as:

$$L^\nu(X, U, \mu^\nu, \lambda) = J^\nu + \mu^{\nu\top} D(X, U) + \lambda^\top C(X, U) + \frac{1}{2} C(X, U)^\top I_\rho C(X, U), \tag{21}$$

Where μ^ν and λ are the Lagrange multipliers, and I_ρ is a penalty matrix. The optimality conditions for a Nash equilibrium are then defined as follows:

$$\nabla_{X,U^\nu} L^\nu(X, U, \mu^\nu, \lambda) = 0, \quad \forall \nu, \tag{22}$$

The problem is solved iteratively using a quasi-Newton method, ensuring that both constraints and equilibrium conditions are satisfied. Intuitively, it can be seen that when the gradient of the Lagrangian is zero for all agents simultaneously, this represents an equilibrium point where all agents reach a local minimum they are not incentivized to deviate from.

Although the Lagrangian approach improves computational efficiency, the method still faces challenges with scalability and real-time performance in dense multi-agent interactions. The time taken to find a solution heavily depends on how close the initial guess used to warm-start the optimization is to a feasible Nash equilibrium. Even with only 4 agents, in the experiments this method was used in, it failed to reliably perform in real time with an average solution time of $0.25 \pm 0.2s$.

APPENDIX C: EXPERIMENT DETAILS

For all experiments, the cost function introduced in the methodology section is used. For clarity, this equation is included again in this section:

$$J(\tau_{0:K}) = \sum_{k=0}^{K} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \gamma^t \lambda \mathbb{KL}(q(\boldsymbol{x}_k) || p(\boldsymbol{x}_k)). \tag{23}$$

Here, $J_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ concisely represents the intrinsic objectives of the agent, such as reaching the goal, along with the control costs, while the KL-divergence term $\mathbb{KL}(q(\boldsymbol{x}_k)||p(\boldsymbol{x}_k))$ accounts for the predictability cost.

Unstructured Tasks

For open navigation tasks in unstructured environments, including single agent experiments and swapping tasks, the term $J_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is defined as follows:

$$J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = w_{\text{goal}} \cdot \|x_k - x_{\text{goal}}\|^2 + w_{\text{control}} \cdot \|u_k\|^2 \tag{24}$$

The first term penalizes the distance between the state x_k of the system at timestep k and the desired goal state x_{goal}, pushing the system to move towards the goal. The second term $w_{\text{control}} \cdot \|u_k\|^2$ penalizes the magnitude of the control input u_t, encouraging smoothness in the trajectories and minimal control effort. The weights w_{goal} and w_{control} can be used to balance the trade-off between reaching the goal efficiently and minimizing control effort.

Structured Tasks

Robot-Robot Traffic Tasks. In this experiment, the agent is tasked with navigating a traffic environment while interacting with other dynamic agents. The experiments use the simulator developed by [18], however it was adapted to integrate an MPPI planner. The simulator out of the box incorporates a sampling-based Frenet planner as proposed by [40], which was employed in an additional experiment to evaluate the compatibility of the proposed methodology with alternative planning frameworks. Detailed descriptions of this planner and the cost function are provided in Appendix D. For this experiment, the cost function outlined in Appendix D was implemented as the $J_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$ term within the MPPI planner. Sampling for the MPPI planner is done in the Cartesian frame and then transformed into the Frenet frame for evaluation. This approach ensured

compatibility with the Frenet-based cost function and facilitated integration with the experimental simulator, as designing another cost function would have been time consuming and out of the scope of the study.

Human-Driver Data Experiment. In these experiments the agent is tasked with navigating in an environment populated by human agents replaying recorded trajectories from the Waymo Open Motion Dataset. The simulator used in this work was developed by [23] to accommodate their research needs. The environments in the waymo dataset are highly interactive and have many traffic rules. For their work [23] developed a structured cost function that accounts for travel efficiency, ride comfort, traffic rule adherence, and safety. For our work we take this cost function and accommodate it to work with an MPPI planner. The overall stage cost is formulated as:

$$J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_i \|\omega_i c_i(\boldsymbol{x}_k, \boldsymbol{u}_k)\| \tag{25}$$

where c_i are the different cost terms, ω_i are their respective weights. The components of the cost function are defined below.

To encourage efficient driving, the speed cost penalizes deviations from the speed limit:

$$c_{\text{speed}} = v_k - v_{\text{limit}},$$

where v_k is the vehicle's speed at timestep k, and v_{limit} is the speed limit. Ride comfort is ensured through penalties for abrupt vehicle maneuvers by penalizing large control inputs:

$$c_{\text{acc}} = a_k, \quad c_{\text{jerk}} = \dot{a}_k, \quad c_{\text{steer}} = \delta_k, \quad c_{\text{rate}} = \dot{\delta}_k, \tag{26}$$

where a_k and $\dot{a}_k$ are acceleration and jerk, and δ_k and $\dot{\delta}_k$ are the steering angle and its rate of change. The framework enforces adherence to traffic rules by encouraging lane following and alignment of the heading angle as well as stopping at red lights. Lane following and heading alignment are enforced with the following term:

$$c_{\text{pos}} = p_k - p_{\text{l},\perp}, \quad c_{\text{head}} = \theta_k - \theta_{\text{l},\perp}, \tag{27}$$

where p_k and θ_k are the cartesian position and heading of the vehicle, and $p_{\text{l},\perp}$ and $\theta_{\text{l},\perp}$ are the position and heading of the closest lane centerline point. Stopping at red lights is enforced with the following term:

$$c_{\text{traffic},t} = \begin{cases} s_t - s_{\text{stop}}, & \text{if } s_t \geq s_{\text{stop}}, \\ 0, & \text{otherwise}, \end{cases} \tag{28}$$

where s_k is the vehicle's running distance, and s_{stop} is the position of the red light. s distance is measured along the lane centerline relative to the positon of the vehicle at $k = 0$. Finally, safety is enforced by maintaining a minimum distance ϵ from other traffic participants. The safe distance cost is computed in Frenet frame:

$$d_{\text{safe}} = \min_i \|p_k - p_k^i\|_2, \tag{29}$$

where p_k and p_k^i are the positions of the ego vehicle and the i-th agent, respectively. A hinge function is used to compute the cost:

$$c_{\text{safety},t} = \begin{cases} \epsilon - d_{\text{safe}}, & \text{if } d_{\text{safe}} \leq \epsilon, \\ 0, & \text{otherwise}. \end{cases} \tag{30}$$

This term measures distance to other agents in the frenet frame along the vehicles reference trajectory and is used to focus attention

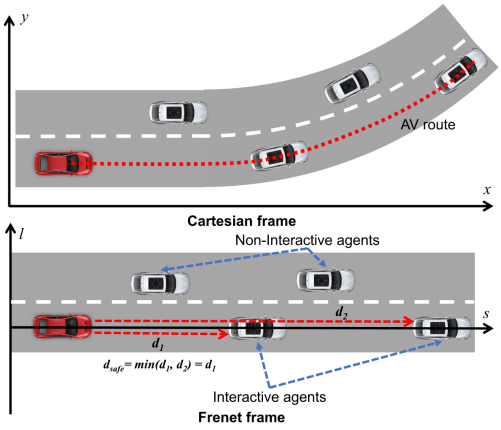


Figure 6: Illustration of the safety distance calculation. Other agents are first projected to the Frenet frame to find the interacting agents. and the calculation of distances is in the Cartesian frame

solely on those agents whose predicted positions are within the reference path's conflict area. The authors of [23] designed this term to improve efficiency by only accounting for interactive agents. To further clarify the functioning of this term an illustration of its functioning is provided in figure 6

APPENDIX D: ADDITIONAL EXPERIMENT WITH FRENET PLANNER

Experiment Objective: This experiment was conducted with the objective of demonstrating the compatibility of the method with different planners. The motivation for this experiment was to showcase that this method is agnostic to the planner being used and can improve performance in general.

Setup: The setup for these experiments is identical to the one presented in the previously introduced robot-robot traffic scenario experiment, with the only difference being that a different trajectory planner is employed. The trajectory planner used follows the methodology of [40] and is designed to generate trajectories in Frenét Frame. The Frenét Frame is a moving reference frame defined along a given curve, in this case the center-line of the road, and decouples lateral and longitudinal motions along the road. This reference frame consists of an axis defined by the tangential vector $\mathbf{t}_r$ in the direction along the curve and another axis defined by the normal vector $\mathbf{n}_r$ in the direction perpendicular to the curve. The vehicle's position $\mathbf{x}(s, d)$ in Cartesian coordinates is expressed in terms of arc length s and lateral offset d as per equation 31:

$$\mathbf{x}(s(t), d(t)) = \mathbf{r}(s(t)) + d(t)\mathbf{n}_r(s(t)) \tag{31}$$

With $\mathbf{r}(s(t))$ the position of the reference curve parameterized by the arc length s, $d(t)$ the lateral deviation from the reference curve and $\mathbf{n}_r(s(t))$ is the normal vector at position s along the curve. A set of candidate trajectories is generated for both lateral and longitudinal directions by uniformly sampling end conditions within feasible ranges (lateral position and longitudinal velocity in this case). Often an evenly spaced distribution of values is used giving rise to a grid like array of end conditions. Trajectories are

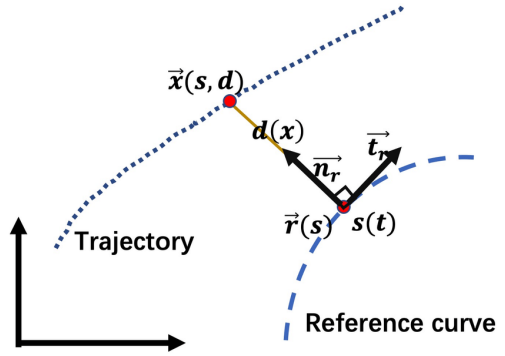


Figure 7: Illustration of transformation from Cartesian to Frenét frame [35]

then combined and evaluated based on a cost function and checked for collision avoidance and kinematic constraints. An example of the resulting sample trajectory distribution can be seen as the green trajectories in figure 8

For lateral motions, trajectories $d(t)$ are modeled using quintic polynomials:

$$d(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \tag{32}$$

where coefficients a_i are determined based on the sampled end conditions d_1, T (time duration), and the initial conditions $[d_0, \dot{d}_0, \ddot{d}_0]$.

For longitudinal motion, trajectories $s(t)$ are also modeled using quintic polynomials:

$$s(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5, \tag{33}$$

where the end conditions s_1, $\dot{s}_1$, T, and the initial state $[s_0, \dot{s}_0, \ddot{s}_0]$ are sampled.

Each trajectory is checked for kinematic feasibility and collision avoidance, and invalid trajectories are discarded. In [40], a cost function is defined directly in Frenet frame as seen in equation 34

$$C_{\text{tot}} = k_{\text{lat}} C_{\text{lat}} + k_{\text{lon}} C_{\text{lon}} \tag{34}$$

where C_{lat} penalizes lateral deviations and C_{lon} accounts for longitudinal metrics such as acceleration, travel time, and matching a target velocity. For the implementation used in our work, the trajectories are further converted back to Cartesian frame as per equation 31. In this manner, they can be compared to predictions from the Wale-Net prediction model which are outputted in Cartesian frame, which is necessary for computing the predictability cost. This cost is added to the previous cost computed in Frenét frame. Among all valid trajectories, the one with the lowest cost is selected as the optimal solution. Finally, the generated trajectories represent desired state transitions. A feedback controller is used to translate these trajectories into action sequences.

Results Discussion: The results for varying values of $\lambda = 0, 2.5, 5$ are presented in Table 4. Similar to the findings from the previous experiment, the inclusion of predictability as an optimization objective significantly reduces the likelihood of deadlock. This reduction in the deadlock probability directly influences the performance in the distance traveled metric, as fewer deadlocks correlate to smoother and more efficient trajectories. These results align with those obtained using the MPPI planner discussed earlier, reinforcing

Table 4: Results for T-Junction and Lane Merge Scenarios using Frenét Planner

Exp	Metric	$\lambda = 0.0$	$\lambda = 2.5$	$\lambda = 5.0$
T-J	Dlk (%)	63.3	10.0	**0.0**
	Dist (m)	22.942	59.369	**65.093**
	PE (m^2)	**1.125** ±0.931	12.462 ±7.093	13.899 ±5.448
	Acc (m/s^2)	-0.431 ±0.151	0.216 ±0.309	**0.460** ±0.344
	Ang (rad/s)	0.0043 ±0.0015	0.0034 ±0.0014	**0.0047** ±0.0032
LM	Dlk (%)	86.6	**0.0**	**0.0**
	Dist (m)	36.226	107.239	**103.353**
	MSE	**4.768** ±1.680	21.586 ±6.550	21.145 ±6.97
	Acc (m/s^2)	-0.194 ±0.098	1.674 ±0.388	**1.314** ±0.643
	Ang (rad/s)	0.0024 ±0.0011	0.0058 ±0.0027	**0.0134** ±0.0093

the consistency of the observed trends. The observed differences in performance metrics primarily arise from the use of distinct planners, however they follow the same trends and the benefits can be tracked to the predictability term.

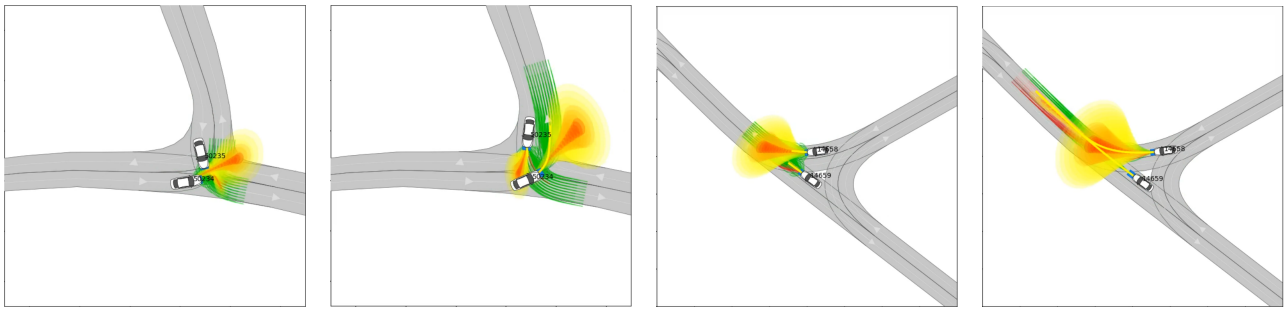
From these observations, it can be concluded that incorporating predictability as an optimization objective provides a scalable and generalizable approach for improving multi-agent coordination. Importantly, this approach is not limited to a specific planner but can be seamlessly integrated into a broad range of planners via the cost function. Furthermore, it can again be seen that the deadlock resolution strategies adopted by the agents resemble social norms observable in human behavior on which , as observed in the previous experiment. Figure 8 provides a visual representation of the two scenarios with agents employing a Frenet planner, further illustrating these findings.

APPENDIX E: DIRECT MINIMIZATION OF FREE ENERGY AS A COST FUNCTION

In the methodology, the free energy term is expressed as a function of the prediction distribution, $\mathcal{P}(X)$, which is outside the agent's direct control in our planning problem formulation. To address this, we apply an expectation switch to introduce a plan distribution, $Q(X)$, which is controllable by the agent. Using Jensen's inequality, we derive an upper bound on the free energy that the agent minimizes as its planning objective. Thus, the agent minimizes not the free energy itself but its upper bound.

During the planning phase, the prediction model remains static, with $\mathcal{P}(X)$ formally defined as $\mathcal{P}(\mathbf{X}) = \mathcal{P}_k(x \mid \mathbf{X}_{-H:0})$ for all $k \in \{1, 2, \ldots, K\}$, where H is the historical context length and K is the planning horizon. While this approach provides valuable planning insights, it is passive in the sense that it does not consider how the agent's actions influence the predictions of surrounding agents, thereby limiting its ability to adaptively interact with these expectations and instead only align with the predictions.

To address this limitation, we explore using the free energy term directly as an optimization objective. This requires granting the agent control over the prediction distribution by dynamically updating the prediction model during planning. The updated model, $\mathcal{P}_k(x \mid \mathbf{X}_{-H+k:k-1})$, conditions predictions on the evolving state history throughout the planning horizon. This dynamic adaptation allows the agent to anticipate the effects of its actions on future predictions, enabling behaviors such as legibility and reducing



(a) Lane merge $\lambda = 0$ (b) Lane Merge $\lambda = 5$ (c) T-Junction $\lambda = 0$ (d) T-Junction $\lambda = 5$

Figure 8: a) Illustration of a deadlock With $\lambda = 0$, where a sequence of faulty predictions reinforces both agent's hesitation. b) For $\lambda = 5$, the agents can leverage the prediction model to coordinate which agent gives way and which passes first.

uncertainty for surrounding agents by minimizing the complexity of the prediction distribution. Under this new approach the cost function used can remain unchanged:

$$J(\tau_{0:K}) = \sum_{k=0}^{K} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \gamma^t \lambda \mathbb{KL}\big(q(\boldsymbol{x}_k) \parallel p(\boldsymbol{x}_k)\big).$$

However, incorporating the dynamically updated prediction model requires continuous updates during the optimization. For sampling-based planners like Model Predictive Path Integral (MPPI), this approach requires reevaluating the prediction model for each sample at every time step. For a planning horizon $K = 20$ and 800 samples, this translates to 16,000 evaluations per cycle, making real-time application computationally intractable in practical use cases, where a more sophisticated prediction model could me more expensive to evaluate. Thus, adopting a more sampling-efficient planner is critical to address these computational challenges. However, in this section we use a simple prediction model an explore some of the behavioral properties that can be induced when minimizing the free energy term directly.

6.1 Unifying Predictability and Legibility

Given the iterative updates of the prediction model, the agent now has feedback on how its planned actions will affect the prediction distribution in future time steps. The agent can then minimize the KL term in its cost function by both aligning its plan with the prediction distribution and by selecting actions that push the distribution to better align with its objective or reduce its complexity. These are both desirable traits for social navigation and represent equivalents to predictability and legibility respectively for receding horizon planning. The main advantage of approaching the optimization in this manner, through minimizing free energy, is that it automatically handles the trade-off between both of these properties in a principled manner, thereby automatically addressing a trade-off akin to the common exploration-exploitation trade-off. Furthermore, framing social navigation objectives in this manner serves as a generalization, as it not only automatically balances the trade-off in a principled way but is also compatible with any form of prediction distribution.

6.2 Practicalities

Remark 2. *The KL divergence between two distributions is non-symmetric and depends on the respective shapes of the distributions. To illustrate how the shapes of the distributions affect the KL-divergence term, we can examine a simple illustrative example with two univariate Gaussian distributions: P representing the plan and Q the prediction distributions. The analytical expression can be found in equation 35:*

$$D_{KL}(P \parallel Q) = \ln\left(\frac{\sigma_q}{\sigma_p}\right) + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{2\sigma_q^2} - \frac{1}{2} \tag{35}$$

With μ representing the mean of a distribution and σ its standard deviation, the Kullback-Leibler (KL) divergence quantitatively measures how one probability distribution diverges from a second. Conceptually, the KL divergence measures how much information is lost when approximating P with Q. Thus, if samples drawn from P have a low probability under Q compared to their respective probability under P, this will result in a large KL divergence.

By examining the KL divergence equation, we observe distinct behaviors in response to differences in means and variances. The distance between the means is penalized quadratically, as evidenced by the term $(\mu_p - \mu_q)^2$. This quadratic penalization results in a rapid increase in divergence when there are large discrepancies between the means. On the other hand, the difference in variances is penalized logarithmically, as seen in the term $\ln\left(\frac{\sigma_q}{\sigma_p}\right)$. For Gaussians, this logarithmic penalization results in a slower increase in divergence.

When minimizing free energy through the minimization of the KL term, the influence of predictability is stronger than legibility. When planning, this occurs because differences in the means of distributions exert a significantly greater impact on the KL term than differences in their variances, as pointed out in remark 2. Consequently, the agent is driven to plan trajectories that closely align with the most probable outcomes of the prediction distribution, as the penalty for deviations grows quadratically. This tendency hinders the optimization of legibility in ambiguous settings, which may necessitate several unpredictable actions in the early time steps to achieve benefits in later stages. Under the current formulation, such benefits are unattainable, as the advantages of reducing distribution variance are outweighed by the substantial cost associated with early deviations. This issue can be mitigated in two ways.

First, extending the planning horizon would allow the legibility benefits to haver a greater weight on the overall cost, however, this approach may render the optimization process computationally intractable. Second, modifying the discount factor offers a viable solution. While a discount factor $\gamma < 1$ previously emphasized predictability in early time steps, increasing the discount factor to $\gamma > 1$ would prioritize legibility gains at later time steps. This adjustment outweighs the penalties for early unpredictability with the benefits of a better aligned and narrower prediction distribution at later horizon steps, thereby enabling the robot to better reason on how to optimize for long-term gains.

Although setting $\gamma > 1$ allows the agent to better reason about future gains, acting in benefit of future gains might not be desirable in all settings, as it might induce unpredictable behavior in situations where remaining predictable is critical. For this reason, we suggest it would instead be better to modulate the value of lambda with some hand-crafted heuristic in reaction to the incoming observations: $\gamma(O)$, where O is an observation about the environment. This heuristic should emphasize that while in crowded interacting environments it is more important to prioritize remaining predictable for others, whereas on the lead-up to an interaction or in less crowded environments it is more beneficial to prioritize legibility.

If an adaptive $\gamma(O)$ is used, it is also important to consider that changing the discount also has an effect on the overall weight of the KL-Divergence term in the cost function, with the magnitude being reduced for $\gamma < 1$ and increased for $\gamma > 1$. This can in turn have adverse effects when combined with λ. Thus, to simply achieve a redistribution of the weight as opposed to changing the overall weight, we propose to normalize the discount. A visualization of the normalized weight distributions for different γ can be seen in figure 9 The resulting cost function we propose can be seen in equation 36:

$$J(\tau_{0:K}) = \sum_{k=0}^{K} J_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \frac{\gamma(O)^k}{\sum_{t=0}^{K} \gamma(O)^t} \lambda \mathbb{KL}\big(q(\boldsymbol{x}_k) \parallel p(\boldsymbol{x}_k)\big). \quad (36)$$

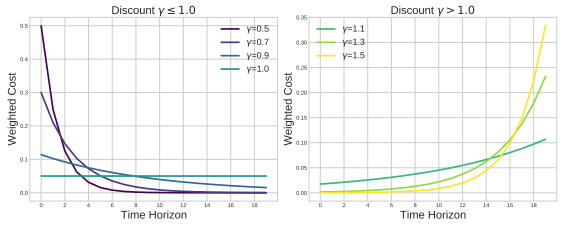


Figure 9: Different levels of discount over a 20 time-step horizon, normalized to keep a constant area under the curve. The normalization on the discount is introduced to keep similar magnitude for the predictability cost in the cost function, with regard to the cost function.

6.3 Iterative Prediction Updates Experiment

Experiment Objective: Pro-social behavior from an agent requires trading off legible and predictable behavior. If the predictions an agent holds about itself are misaligned with its goal, optimizing for legibility can serve as a quick way of rapidly re-aligning them with its objective by acting unpredictably in the near-term to achieve greater gains in the long term. However, the KL Divergence objective weighs predictability more heavily than legibility, thus from the perspective of the agent, the long term gains are often not enough to offset the short term cost of unpredictable actions. This severely limits the extent to which an agent can pursue legibility objectives. Motivated by the fact that displaying legible behaviors in an important aspect of pro-social behavior, in this experiment we show how varying γ can control wether the agent prioritizes legible or predictable behavior.

Setup: We recreate the single agent experiment introduced above with a multi-modal prediction and observer with mistaken initial beliefs. The difference is that the prediction model is now updated in reaction to the planned trajectory along the horizon: $\mathcal{P}_k(x \mid \mathbf{X}_{-H+k:k-1})$ for all $k \in \{1, 2, \ldots, K\}$. Since the model used is analytical, the operation can be vectorized for all MPPI samples, keeping the optimization tractable. Cost function 36 was used for planning, with the discounts being normalized. To illustrate the impact of setting $\gamma > 1$ two sets of experiments for $\gamma = 0.7, 1.3$ shown in blue and green respectively in figure 10. Each value of γ is tested for $\lambda = 20, 40, 60$.

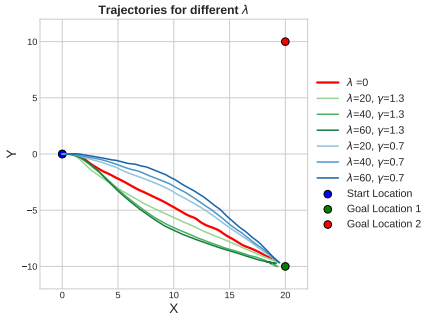


Figure 10: Results of planning with dynamic updates to the prediction model and using cost function 36. It can be observed that setting $\gamma < 1$ results in behaviors that prioritize legibility, whereas setting $\gamma > 1$ results in behaviors that prioritize legibility. Modulation of γ can then be used to adapt the agents behavior in response to the environment.

Results Discussion: Upon examining the observed values for λ, it is clear that they are significantly higher than those in the previous experiment. This increase is attributable to the dynamic updating of the prediction model in response to the agent's trajectory. As a result, the agent's planned trajectory remains closely aligned with the updated prediction distribution, consistently yielding lower Kullback–Leibler (KL) divergence between the prediction and the plan. Consequently, the influence of the KL term is reduced, allowing the agent to more readily pursue its objective, which necessitates a higher value of λ to maintain the trade-off.

Furthermore, the impact of the discount factor γ reveals that when $\gamma > 1$, the resulting trajectories align with the legibility benchmarks observed in previous single-agent experiments. A discount factor greater than one mitigates the higher penalties associated

with unpredictability, enabling the agent to optimize for long-term gains. Specifically, initial unpredictable actions encourage the prediction model to rapidly align with the agent's intentions, resulting in benefits in subsequent time-steps.

This experiment demonstrates that the trade-off between legibility and predictability can be effectively managed using a single parameter that can be automatically adapted based on environmental conditions. Designing a heuristic to modulate this parameter is straightforward, and enables the agent to exhibit more pro-social and adaptive behaviors.

6.4 Limitations

Although the results presented offer a promising approach to pro-social and adaptive planning, the prediction model and environment considered are relatively simplistic and do not fully represent more complex, multi-agent settings. In real-world scenarios, the agent operates within environments populated by other agents and typically relies on more sophisticated prediction models that are not easily vectorized for numerous samples. Adapting the proposed method to these conditions poses significant challenges:

- In multi-agent environments, the behaviors of other agents significantly influence the prediction outcomes. Consequently, when updating the planning trajectory, it is necessary to propagate the expected behaviors of all surrounding agents. This process is complex and can introduce substantial uncertainty.
- Utilizing more computationally intensive prediction models renders sampling-based planners like Model Predictive Path Integral (MPPI) infeasible for real-time applications due to the excessive number of model evaluations required. Addressing this limitation necessitates the adoption of more sample-efficient planners. One potential solution is employing tree-based planners, as demonstrated in existing literature where Tree Policy Planning has been successfully applied to similar problems involving numerous model evaluations [22]. However, adapting tree-based methods to our framework may be challenging, as the policy tree construction could inadvertently exclude optimal trajectories that enhance legibility. Alternatively, a hierarchical planning approach could be utilized, where a high-level planner initially generates a rough trajectory using the complex cost function, which is subsequently refined through a simpler gradient-based method [10].

Addressing these challenges presents valuable opportunities for future research.