Title

Towards Scalable Simulation for Human-Centered Autonomy

ISBN

9798189279850

Author

Chang, Wei-Jer

Publication Date

2026-05-01

Peer reviewed|Thesis/dissertation

Towards Scalable Simulation for Human-Centered Autonomy

by

Wei-Jer Chang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering Science - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Pieter Abbeel
Assistant Professor Negar Mehr

Spring 2026

Towards Scalable Simulation for Human-Centered Autonomy

Abstract

Towards Scalable Simulation for Human-Centered Autonomy

by

Wei-Jer Chang

Doctor of Philosophy in Engineering Science - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Robots are increasingly deployed in environments that involve close interaction with humans. To operate safely and intelligently in these settings, they must handle diverse and reactive human behavior. However, collecting such data at scale is costly and limited. Moreover, real-world data provides only passive observations and does not capture how humans would respond under different robot actions. While simulation offers a promising alternative, existing approaches struggle to capture realistic human behavior, support counterfactual reasoning, and scale to complex interactive settings.

We begin by quantifying human social preferences from real-world interaction data using a data-driven measure of courtesy that captures the influence of an agent's behavior on others. This enables socially controllable agent behaviors ranging from courteous to aggressive interactions to be learned directly from data. Importantly, we observe that courteous behavior manifests differently across interaction contexts. Second, we address the limited coverage of long-tail and safety-critical interactions in real-world datasets through controllable generative simulation. We propose a guided and partial diffusion framework for controllable generation of safety-critical interactions with diverse aggressiveness levels and collision types. Furthermore, we integrate natural language conditioning into joint multi-agent diffusion models, enabling high-level specification of interactive behaviors for counterfactual simulation. Finally, we address the scalability of closed-loop simulation, where large generative behavior models are often computationally expensive at inference time. We introduce a self-play reinforcement learning framework that uses pretrained tokenized reference models to guide policy learning toward human-like behavior. This enables scalable and realistic multi-agent simulation while achieving up to an order-of-magnitude faster inference than large generative models.

Together, this dissertation lays a practical foundation for realistic and controllable simulation of human behavior, enabling autonomous systems to reason about the counterfactual consequences of different actions in human-centered environments.

To my family, and to the one who never gave up on this journey

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my advisor, Professor Masayoshi Tomizuka, for giving me the opportunity to conduct research under his guidance. Throughout our meetings, I learned a lot about how to conduct practical research grounded in real-world problems. In addition, it has been very inspiring to witness his dedication to mentoring future generations of researchers. I also hope that I can inspire and guide future researchers with the same dedication.

I am also grateful to my qualifying committee members, Professor Koushil Sreenath, Professor Mark Mueller, Professor Somayeh Sojoudi, and Professor Jonathan Shewchuk, as well as my dissertation committee members, Professor Pieter Abbeel and Professor Negar Mehr. Thank you for your guidance and support throughout this journey.

In addition, I would like to thank Wei Zhan for guiding me throughout my PhD research. Your knowledge of both academia and industry has helped me select research directions that are both practical and impactful. Next, I would like to thank the MSC Lab members and senior lab members for their guidance. In particular, I would like to thank Yeping, Chen Tang, and Chenran for meeting with me regularly during my first two years. I would also like to thank Hengbo Ma, Linfeng Sun, Chenfeng Xu, Jessica Leu, Zheng Wu, Rian Tian, Jiachen Li, and ChanHao Wang for providing feedback on my research. Having mentors and peers to discuss ideas with and look up to has been extremely important throughout my journey. I am also grateful to all the MSC Lab members for creating such a supportive environment. Being part of a group with such talented and encouraging people has been truly rewarding. I would also like to thank my collaborators, Professor Yiting Chen and Yu-Hsiang Chen, for their valuable discussions and support. I am also grateful to Professor Changliu Liu for her guidance and insights during our weekly meetings.

Next, I would like to thank Manmohan and Francesco for their support during my internship at NEC Labs. I spent two wonderful summers there, which were productive and enjoyable, and they broadened my perspective on how industry values research and approaches different problems. I would also like to thank the researchers Yihan, Akshay, and Alex at Applied Intuition for our wonderful collaboration during my internship. It has been a pleasure working with all of you.

I would also like to thank my closest friends at Berkeley: Ting-Hao Wang, Chih-Che Lin, Jen-Wei Wang, and Shao-Yi Yu. We came here together as PhD students, and one of the best parts of this journey has been having friends who I could freely talk to about the experiences and challenges as a PhD student, while supporting one another throughout these five years.

I would like to thank myself for making it this far, for staying dedicated and maintaining confidence and passion throughout this journey, even during difficult times. This journey has taught me that life is not only about research itself, but also about learning how to stay healthy, take care of myself, my family, and my friends.

Finally, I would like to thank my family. I thank my father for inspiring me to pursue a career in research and for always giving me advice on how to navigate different challenges. I thank my mother for always taking care of our family and encouraging me throughout this journey. I never felt pressure from them and only the confidence that my family would

support me through the ups and downs of my PhD. Without your love and support, this journey would not have been possible. Also, thank you Chih-Ling Chang for always being by my side and supporting me throughout this journey, while also being the mastermind behind many of my research presentations and slides.

Chapter 1

Introduction

1.1 Motivation

Robots are increasingly deployed in environments that involve close interaction with humans. To operate safely and intelligently in these settings, they must handle diverse and often unpredictable human behavior. This challenge arises across domains, including autonomous driving, warehouse human–robot collaboration, and social navigation in public spaces. To operate effectively, robots require experience interacting with people. However, collecting such interactive data in the real world is slow, expensive, and often constrained by safety considerations. This creates a fundamental challenge for developing and validating autonomous systems.

1.2 Simulation: System-Level View

Simulation provides a promising tool for scaling interactive experience, evaluation, and stress-testing of autonomous systems under diverse interactive scenarios. Real-world testing remains necessary for validating autonomous systems in human environments. However, it is expensive, time-consuming, and often constrained by safety considerations. It also provides limited coverage of long-tail interactions, many of which are rare yet important for safety evaluation. Moreover, while real-world datasets provide realistic observations, they are limited for evaluation: they capture what happened, but not how humans would respond under different robot actions. As a result, even small changes to a robot's algorithm may require additional real-world testing.

Simulation offers a safer and more scalable complement to real-world testing. By enabling controlled and counterfactual interaction, it supports capabilities that are difficult to achieve in the real world, such as systematic evaluation and large-scale training. If simulation can faithfully capture real-world scenarios and human responses, it can significantly accelerate the development cycle of autonomous systems.

Recent advances in robotics have demonstrated the transformative impact of scalable simulation platforms such as MuJoCo [1] and Isaac Lab [2]. These simulators have accelerated robot learning and evaluation by enabling scalable, safe, and reproducible interaction in simulated physical environments. Similarly, for autonomous systems operating around humans, realistic human behavior simulation may play an important role in enabling scalable development, evaluation, and training.

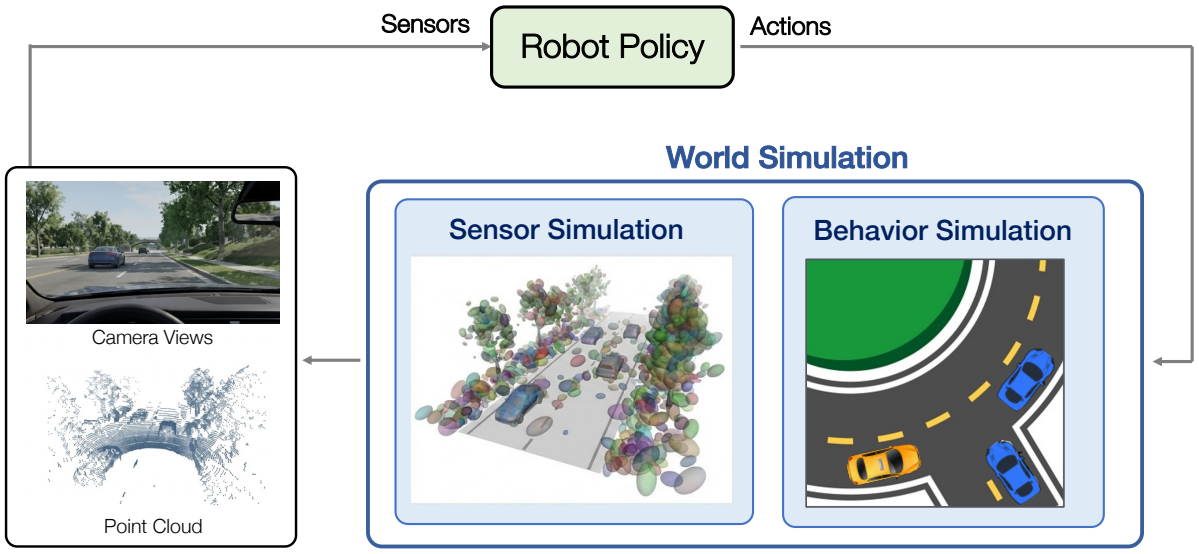

Figure 1.1: **System-level simulation for autonomous systems.** Behavior simulation models agent interactions and decision-making, while sensor simulation generates perceptual observations such as camera images and LiDAR point clouds. Together, they form a complete simulation framework from perception to interaction, enabling evaluation of autonomous systems from sensor inputs to actions.

As illustrated in fig. 1.1, simulation at the system level consists of both behavior simulation and sensor simulation. Behavior simulation models how humans and other agents behave and interact over time, including their decisions, motions, and reactions to surrounding agents. In autonomous driving, this problem is often referred to as traffic simulation, where the goal is to model realistic multi-agent interactions among vehicles and other road users. This dissertation focuses primarily on behavior simulation, where a key challenge is capturing realistic human decision-making and interaction dynamics. In contrast, sensor simulation models what the autonomous system perceives from the environment by generating observations such as camera images or LiDAR point clouds. Together, behavior simulation and sensor simulation provide complementary components for constructing a complete simulation from perception to interaction.

1.3 Simulation in Practice: Usage and Industry Perspective

Behavior simulation plays a central role in developing and evaluating autonomous systems that interact with humans. For modular systems, behavior simulation is used to model interactions between the ego agent and surrounding agents, enabling evaluation of prediction and planning components. When paired with sensor simulation, as shown in fig. 1.1, it enables evaluation of the full system, from sensor inputs to actions, for both modular and end-to-end architectures.

In addition, if we can learn a sufficiently expressive model of human behavior, it can also be used directly for planning in a way that is compatible with humans. However, it is important to distinguish that planning and simulation serve different goals. Planning focuses on selecting safe and optimal actions, and may go beyond typical human behavior for safety or efficiency. In contrast, simulation aims to reflect real-world behavior, requiring diversity and broad coverage of possible interactions.

Current industry bottlenecks include safety certification and edge-case discovery, which often require large amounts of real-world testing with diminishing returns. Simulation is essential to systematically explore scenarios that would be infeasible to cover physically. Safety standards such as SOTIF (Safety of the Intended Functionality, ISO 21448 [3]) emphasize the need for robust testing of unknown unsafe scenarios, highlighting the importance of practical simulation frameworks for scalable validation and certification.

In practice, autonomous systems are developed and evaluated using a combination of real-world testing and simulation-based approaches, each with different trade-offs in realism, scalability, and interaction.

Real-world testing. Real-world testing provides the most direct form of evaluation, since data is collected from actual system deployments and human interactions. One common practice is shadow mode evaluation, where the tested algorithm runs in parallel with a human operator or production control system without actually executing its actions. This allows developers to compare the algorithm's predicted actions against real-world outcomes while maintaining safety. However, real-world testing is expensive, time-consuming, and difficult to scale, and provides limited coverage of rare or safety-critical scenarios.

Log replay. Log replay evaluates systems using recorded real-world data, enabling reproducible benchmarking. Its main advantage is that it directly reflects real-world scenarios. However, it is inherently passive and lacks interaction, preventing it from capturing how humans would respond under different robot actions and limiting its use for evaluating closed-loop behavior.

Heuristic simulation. Heuristic simulation uses domain knowledge to specify agent behavior through predefined rules and manually designed scenarios, such as setting speeds, following behavior, or cut-in events. This enables controlled and targeted testing and is widely used in practice. However, it requires significant domain expertise and manual effort to design and maintain realistic scenarios.

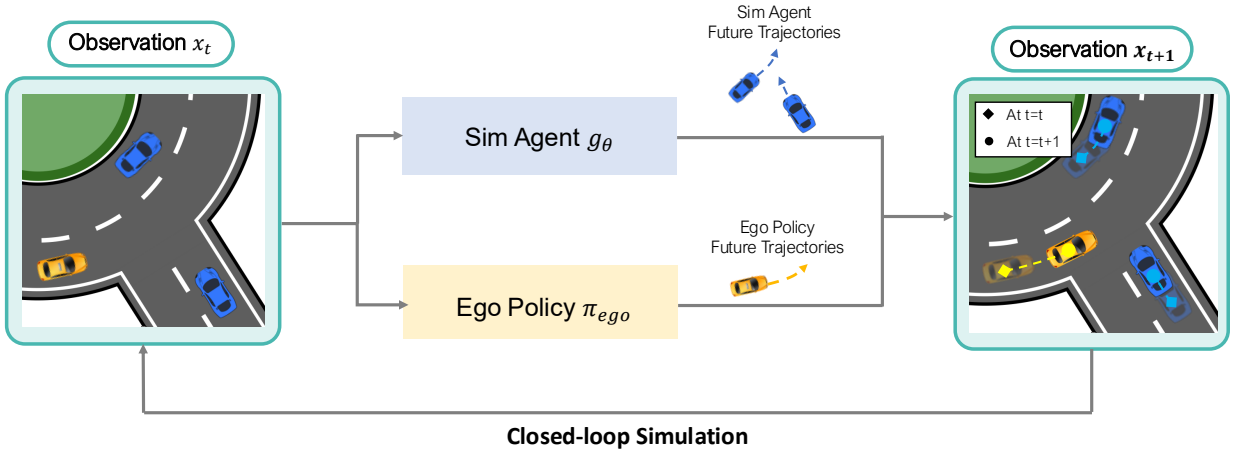

Figure 1.2: **Closed-loop interaction between the ego policy and simulation agents.** The ego policy and simulation agents independently generate future trajectories conditioned on the current observation. Interaction is modeled through recursive closed-loop environment updates, where generated trajectories update the scene and produce new observations for subsequent rollout steps.

Data-driven simulation. Data-driven approaches learn behavior directly from real-world data, aiming to transform the passive real-world datasets into reactive, closed-loop simulators that can model how agents respond to different actions. This reduces the need for manual scenario design and improves coverage of diverse environments and interactions. These approaches have become increasingly popular in industry [4, 5, 6] due to their potential to capture more complex patterns. They still face limitations in controllability and stable closed-loop interaction.

1.4 Closed-Loop Simulation Formulation

In this dissertation, behavior simulation is studied primarily in a closed-loop setting, where simulation agents and the tested autonomous system continuously interact with each other over time. Throughout this dissertation, we use the term simulation agents to refer to the surrounding agents modeled within the simulator. As illustrated in fig. 1.2, the simulator provides an observation x_t at each simulation step, which may include scene context such as map information, neighboring agents, and historical states. Based on the current observation, the ego policy π_{ego} generates a future trajectory τ_t^{ego} for the ego vehicle, while the simulation agent model g_θ generates future trajectories τ_t^{sim} for the surrounding simulation agents:

$$\tau_t^{\text{ego}} \sim \pi_{\text{ego}}(x_t), \qquad \tau_t^{\text{sim}} \sim g_\theta(x_t).$$

The simulator then updates the environment as

$$x_{t+1} = \mathcal{T}\left(x_t, \tau_t^{\text{ego}}, \tau_t^{\text{sim}}\right),$$

where $\mathcal{T}$ denotes the environment update process.

Interaction between the ego policy and simulation agents is modeled through recursive environment updates, where all agents condition their future behavior on the evolving scene state. Importantly, this formulation treats π_{ego} as an externally tested policy: it does not assume a specific internal structure for the ego policy, only that it produces future trajectories conditioned on the current observation.

In contrast to open-loop trajectory prediction, closed-loop simulation repeatedly updates the scene state based on generated trajectories, allowing agents to influence one another through interaction over time.

1.5 Dissertation Overview and Contributions

For behavior simulation, there are three key desiderata: realism, diversity, and controllability. Specifically, realism describes how well the simulation agents resemble actual human behaviors. This is a central challenge for data-driven models, as errors can accumulate over long horizons, leading to unrealistic interactions. Diversity refers to the ability to capture a wide range of scenarios, particularly rare and safety-critical events in the real world, which are essential for rigorous evaluation of autonomous systems. Controllability refers to the ability to manipulate simulated agents to generate targeted scenarios for testing and analysis. These desiderata motivate the main contributions of this dissertation, where each chapter focuses on different challenges in behavior simulation.

In Chapter 3, we explore how to quantify human social preferences from real-world data. This enables distinguishing diverse social behaviors, such as aggressiveness and courtesy, and provides signals for training controllable simulation agents that can modulate social behavior in a meaningful way. In Chapter 4, we tackle the challenge of capturing long-tail and safety-critical scenarios in data-driven simulation. We introduce a guided diffusion-based framework that combines data and domain knowledge to enable controllable generation of adversarial agent behaviors, improving coverage and realism in evaluation. Building on this, in Chapter 5 we further explore controllability by introducing a language-conditioned simulation framework. This enables counterfactual "what-if" scenario generation, allowing flexible and intuitive specification of target scenarios for evaluation. In Chapter 6, we focus on scalable and efficient closed-loop interaction by introducing a self-play reinforcement learning framework anchored by pretrained imitation-based behavior models. This enables faster and more reactive simulation agents, and improves scalability in closed-loop multi-agent settings. Finally, Chapter 7 summarizes the key insights from this dissertation and discusses their implications for future research in behavior modeling and simulation.

While this dissertation focuses primarily on the domain of autonomous driving, due to the richness and availability of large-scale human interaction data, the underlying principles and

approaches are broadly applicable to domains involving closed-loop interaction. Together, these chapters address key challenges in behavior simulation, including social behavior, controllability, and efficiency. The main contributions of this dissertation are:

- **Data-driven modeling of human social behavior** to enable interpretable and controllable simulation along meaningful social dimensions.
- **Generative frameworks for counterfactual simulation**, including diffusion-based and language-conditioned models for producing diverse and safety-critical scenarios.
- **Scalable closed-loop multi-agent simulation via self-play reinforcement learning**, guided by pretrained human reference models to enable efficient and realistic interaction.

Chapter 2

Related Work

2.1 Heuristic-Based Simulation

Traffic simulation can broadly be categorized into two main groups: heuristic-based and data-driven methods. In heuristic-based simulation, agents are controlled by human-specified rules [7, 8]. A representative example is the Intelligent Driver Model (IDM) [9], which models vehicle motion by following a leading vehicle while maintaining a safe distance. Widely used simulators such as SUMO [10] and CARLA [11] also rely on rule-based agents.

However, hand-coded agents often fail to capture the diversity of real-world driving behavior, resulting in a mismatch with the underlying traffic distribution. In addition, manually designed rules require extensive tuning to cover diverse behaviors, and this challenge is further compounded by regional variations in driving norms and interaction patterns, which are difficult to encode through fixed rules and demand substantial domain expertise.

2.2 Data-Driven Simulation

To close the gap, data-driven approaches imitate real-world behavior by learning from real-world driving datasets [12, 13]. Earlier works like TrafficSim [4] perform scene-level traffic simulation using a trained variational autoencoder. SimNet [14] demonstrates that a simple ResNet [15] architecture can capture more reactive failures compared to log replay. Symphony [16] combines parallel beam search and goal conditioning to improve the realism and diversity of simulated driving behavior. These early works mostly reuse the architecture and training methods of motion prediction models, which may induce compounding errors during closed-loop rollout [17].

2.3 Generative Models for Simulation

Broadly, recent data-driven advances can be categorized as autoregressive models and diffusion models. Autoregressive models formulate simulation as a next-token prediction task, where each agent's trajectory is discretized to a fixed vocabulary [18, 19, 20]. One key design for autoregressive models is to use a centralized architecture that models the joint distribution of all agents. SMART [19] introduced a centralized autoregressive architecture that many later works subsequently built upon, and demonstrated favorable scaling behavior with respect to model size.

In parallel, diffusion models have become increasingly popular for traffic simulation [21, 22, 23]. The idea is to model trajectories as an iterative denoising process [24]. One of the key advantages of diffusion models is their flexibility during sampling: they support gradient-based guidance signals for generation, as well as hard constraints, classifier-free guidance, and inpainting. Several works also formulate the diffusion process for agent initialization [25, 26]. Compared to autoregressive models, diffusion models are more flexible during sampling but may face challenges in maintaining realism during closed-loop rollout.

2.4 Controllable Simulation

As mentioned in Chapter 1, controllability is an important aspect of counterfactual simulation and targeted-case testing, as it allows simulation to stress-test the ego policy on rare or edge cases that are difficult to collect in the real world. Earlier works that focus on adversarial scenario generation are [27, 28, 29]. One representative work, STRIVE [29], uses adversarial optimization within the latent space to generate safety-critical scenarios for the planner. In addition, diffusion models have become widely adopted due to their flexibility at inference time [21, 30].

2.5 Self-Play Reinforcement Learning

Recently, self-play reinforcement learning has emerged as a promising paradigm for traffic simulation. Simulators such as GPUDrive [31] and PufferDrive [32] enable massively parallel training, while several works have demonstrated that self-play can produce human-compatible driving agents [33, 34, 35]. Cusumano-Towner et al. [36] show that robust autonomy can emerge from self-play at scale. Despite their scalability and strong interactive capabilities, self-play reinforcement learning methods are still less commonly used as simulation agents due to challenges in maintaining realistic human behavior during closed-loop interaction.

Part I

Learning Human Social Preferences from Data

Chapter 3

Editing Driver Character: Socially-Controllable Behavior Generation for Interactive Traffic Simulation

This chapter is based on the paper "Editing Driver Character: Socially-Controllable Behavior Generation for Interactive Traffic Simulation" [37], written in collaboration with Chen Tang, Chenran Li, Yeping Hu, Masayoshi Tomizuka, and Wei Zhan.

3.1 Introduction

Typically, robots are optimized for safety and efficiency. However, in real-world environments, they must interact with humans who exhibit diverse social preferences. For example, at an intersection, an autonomous vehicle may need to yield to aggressive drivers, while behaving more assertively around courteous drivers to maintain efficiency and avoid deadlock. Accounting for such social preferences enables robots to adapt their behavior to different interaction contexts, rather than optimizing solely for selfish objectives.

To ensure safe and efficient behavior in such interactive scenarios, it is important to evaluate autonomous systems against simulated agents exhibiting diverse social characteristics. However, these social preferences are inherently difficult to define and quantify, and are not explicitly annotated in real-world datasets.

Data-driven approaches, particularly deep learning methods, have been widely adopted to synthesize *realistic* reactive agents from large-scale driving data [4, 14, 38]. However, such approaches primarily reproduce the training data distribution and lack mechanisms to explicitly control high-level behavioral properties such as social preferences [21]. On the other hand, model-based approaches typically rely on hand-crafted reward functions or predefined interaction models, which require strong assumptions about agent behavior and may not fully

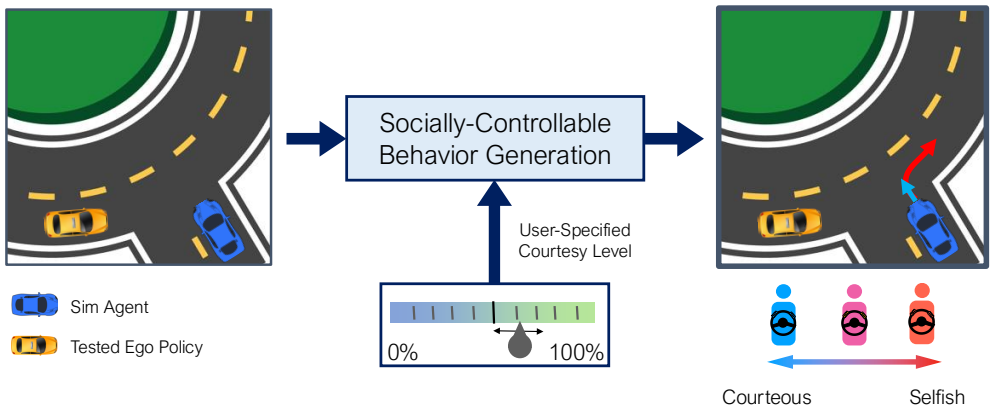

Figure 3.1: **Illustration of the proposed SCBG model**. The SCBG model aims to control the driving behavior of a simulated agent based on the input courtesy level. By generating driving behaviors with controlled courtesy levels, the SCBG model could potentially enable more efficient evaluation and training of autonomous driving algorithms in interactive traffic scenarios.

capture the diversity of real-world interactions. These limitations highlight the need for a framework that enables both realistic and controllable modeling of social behavior.

In this chapter, we take a data-driven approach to infer and quantify social behavior directly from real-world data. Different from prior works that focus on controlling safety criticality [28, 29, 39], we aim to explicitly model and control the *social preferences* of reactive agents. By specifying such preferences, simulation can capture a broader range of interaction behaviors, enabling more comprehensive evaluation of autonomous systems under diverse human driving styles. Furthermore, we may leverage the controllable simulation to design training curricula to accelerate policy training or synthesize driving policies with robust performance for AV as it interacts with human drivers with diverse social characteristics.

The main challenge lies in *how to quantify and label the social preference of human driving behavior*. Human driving behavior involves sophisticated reasoning procedures which may be characterized from various different aspects. Inspired by prior works [40, 41], we choose to control the level of *courtesy* of the reactive agents. Courtesy can be formalized as the change in the expected *reward* of the other agents due to the simulated agent's actions. Intuitively, selfish agents may negatively influence the utilities of other agents, whereas courteous agents take actions that maximize the utilities of other agents. Courtesy has been shown as an important factor describing social interaction in driving [41]. Furthermore, we propose a novel data-driven auto-labeling framework that leverages a marginal behavior predictor and a conditional behavior predictor [42, 43] to estimate the courtesy values of driving behaviors in real-world data. With the auto-labeling framework, we can avoid costly and biased manual efforts in labeling the social characteristics of driving data.

Given real-world driving data with courtesy value labels, we train a *socially-controllable*

behavior generation (SCBG) model synthesizing a simulated agent's future trajectory given an input courtesy value. During training, we leverage the auto-labeling framework to augment the training data with synthesized trajectories. We also utilize the differentiable nature of the auto-labeling operation to design a *courtesy loss*, which directly matches the courtesy values of the generated trajectories with the input courtesy values. Furthermore, we introduce a range predictor to estimate the range of feasible courtesy values in a given scenario, which ensures that the input courtesy value of the SCBG model is feasible during inference time. We examine the proposed method on the Waymo Open Motion Dataset (WOMD) [13] and show that we are able to control the SCBG model to generate realistic driving behaviors with desired courtesy levels. We find that the SCBG model is able to identify the different motion patterns of courteous behaviors according to the scenarios. This work is a crucial step toward developing a scalable traffic simulator with socially-controllable and realistic reactive agent models.

Note that this chapter studies socially controllable behavior generation in an open-loop setting from fixed scene observations. The focus of this chapter is on quantifying and modeling human social behavior, while later chapters study how such behaviors can be incorporated into closed-loop simulation settings.

3.2 Related Work

3.2.1 Behavior Generation for Traffic Simulation

The methods for traffic simulations can be broadly categorized into two main groups: heuristic-based and learning-based approaches. In heuristic-based simulations, the reactive agents are controlled with human-specified rules, such as the Intelligent Driver Model (IDM) [9, 11]. However, these methods lack the modeling capacity required to simulate realistic and human-like behavior. On the other hand, learning-based simulations leverage deep learning-based methods to mimic driving behavior from large-scale driving data [4, 14, 16]. For instance, TrafficSim [4] trained a variational autoencoder-based model to jointly simulate the agents interacting with each other. Symphony [16] combined parallel beam search and goal conditioning to improve the realism and diversity of the simulated driving behavior.

3.2.2 Controllable Behavior Generation

Recently, several methods have been developed to train controllable and realistic driving behavior generation models from large-scale driving datasets [21, 28, 29, 39]. Some of them aim to generate safety-critical scenarios for efficient evaluation. For instance, [29] and [39] generate challenging near-collision scenarios using adversarial optimization. [28] proposes a generative model that can produce interactions with varying safety levels controlled by a style coefficient in the latent space. Unlike them, [21] proposes a controllable behavior generation framework that allows users to specify the desired properties of the trajectories.

Our work differs from existing works in that we focus on controlling the *social* characteristics of the generated driving behaviors.

Our work is also related to the literature on modeling social interaction in human driving behavior. Several works have modeled human-robot interaction from a game-theoretic perspective where human's reactions to robot's actions are captured by seeking the equilibria of the underlying game [44, 45, 46, 47, 48]. Such a game-theoretic model allows for capturing various forms of human preferences. For example, [41] formalizes the courtesy level of human drivers as the effect of a driver's behavior on the other drivers' utilities. Based on this, [49] further characterizes human drivers' social preferences. They then formulate pairwise interaction as a Stackelberg game where each driver's utility is a weighted sum of three reward terms for these perspectives. These approaches can be seen as model-based methods for controllable behavior generation. Instead, we aim to explore a data-driven framework that can generate realistic and socially-controllable driving behavior in multi-agent scenarios.

3.3 Socially-Controllable Behavior Generation

In this section, we introduce the Socially-Controllable Behavior Generation (SCBG) framework. In §3.3.1, we introduce the problem setting considered in this work. In §3.3.2, we define a quantitative data-driven measure of courtesy, which is the core element of our SCBG framework. In §3.3.3, we introduce the data-driven framework to auto-label the courtesy values of vehicle trajectories from real-world data. In §3.3.4, we explain how we train an SCBG model from real-world data. In §3.3.5, we introduce a courtesy range predictor which is used to control the SCBG model during inference.

3.3.1 Problem Formulation

We consider a simulated interactive traffic scenario consisting of two vehicles denoted by Vehicle A and Vehicle B, where Vehicle A is controlled by the tested autonomous driving software and Vehicle B is controlled by the behavior generation model we develop. We aim to design a behavior generation model that allows the users to control *how courteous the behavior of Vehicle B is to Vehicle A*. Concretely, we denote the past observation by $\boldsymbol{x}$, which collects the historical trajectories of the two vehicles and other scene information (e.g., map, static or non-interacting objects). Given $\boldsymbol{x}$, the behavior generation model outputs a future trajectory for Vehicle B to follow, denoted by $\boldsymbol{y}^B$. In addition, the model takes a coefficient ψ as input, which controls the level of courtesy of the generated trajectory. Formally, the model is defined as:

$$\boldsymbol{y}^B = g_{\boldsymbol{\theta}}(\boldsymbol{x}, \psi), \tag{3.1}$$

where $\boldsymbol{\theta}$ denotes the model parameters. We aim to train the model from real-world driving data to ensure the realism of the generated trajectories. However, the courtesy level ψ is a latent variable that is not recorded in the dataset. Thus, we need to define a quantitative measure of courtesy that allows convenient auto-labeling without manual effort.

3.3.2 Quantifying Courtesy

Inspired by [40, 41], we formalize courtesy as the change in the expected *reward* of Vehicle A due to the actions of Vehicle B. Formally, given a prospective future trajectory of Vehicle B, $\boldsymbol{y}^B$, we define its level of courtesy as:

$$\psi(\boldsymbol{y}^B) = \mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}, \boldsymbol{y}^B\right] - \mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}\right], \tag{3.2}$$

where

$$\mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}, \boldsymbol{y}^B\right] = \int_{\boldsymbol{y}^A} r(\boldsymbol{y}^A) p(\boldsymbol{y}^A|\boldsymbol{x}, \boldsymbol{y}^B) d\boldsymbol{y}^A, \tag{3.3}$$

$$\mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}\right] = \int_{\boldsymbol{y}^A} r(\boldsymbol{y}^A) p(\boldsymbol{y}^A|\boldsymbol{x}) d\boldsymbol{y}^A. \tag{3.4}$$

The variable $\boldsymbol{y}^A$ denotes the future trajectory of Vehicle A. The distribution $p(\boldsymbol{y}^A|\boldsymbol{x}, \boldsymbol{y}^B)$ is the *conditional probability* of Vehicle A's future trajectory, given the observation and Vehicle B's future trajectory. The distribution $p(\boldsymbol{y}^A|\boldsymbol{x})$ is the *marginal probability* of Vehicle A's future trajectory. The reward function $r(\cdot)$ gives the cumulative reward of a given trajectory. In practice, we may specify different reward functions according to the scenarios and the user's needs.

The term $\mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}\right]$ indicates the expected reward of Vehicle A under all the possible interactions between the two vehicles, whereas $\mathbb{E}\left[r(\mathbf{Y}^A)|\boldsymbol{x}, \boldsymbol{y}^B\right]$ indicates the expected reward of Vehicle A if Vehicle B executes a particular trajectory $\boldsymbol{y}^B$. The value ψ then indicates the effect of a given future trajectory of Vehicle B on the expected reward of Vehicle A. A positive ψ indicates a positive effect on the reward and thus implies a courteous and cooperative agent. Conversely, a negative ψ indicates a negative effect on the reward and thus implies a selfish agent.

This formulation provides flexibility to accommodate the behavior generation model to the particular aspects of social interaction we want to consider, by adopting different reward functions for courtesy quantification. For instance, to capture the behavioral impact of one vehicle on another, we can define the reward as the change in the trajectory distribution of the other vehicle, which can be quantified by KL-divergence [43]. In this work, we utilize a basic reward function, specifically average speed, to illustrate the proposed framework. In future work, we will conduct comprehensive evaluation to identify the most suitable rewards for different real-world scenarios.

3.3.3 Auto-Labeling Courtesy

The courtesy value defined above depends on the trajectory distribution of the interacting vehicle. However, we do not have access to the ground-truth trajectory distribution of the vehicles appearing in real-world data. Instead, we propose to estimate and auto-label the courtesy values leveraging trajectory predictors trained from real-world data. The auto-labeling framework is illustrated in Fig. 3.2. Given a sample from the dataset, i.e., $\boldsymbol{x}, \boldsymbol{y}^B \sim \mathcal{D}$,

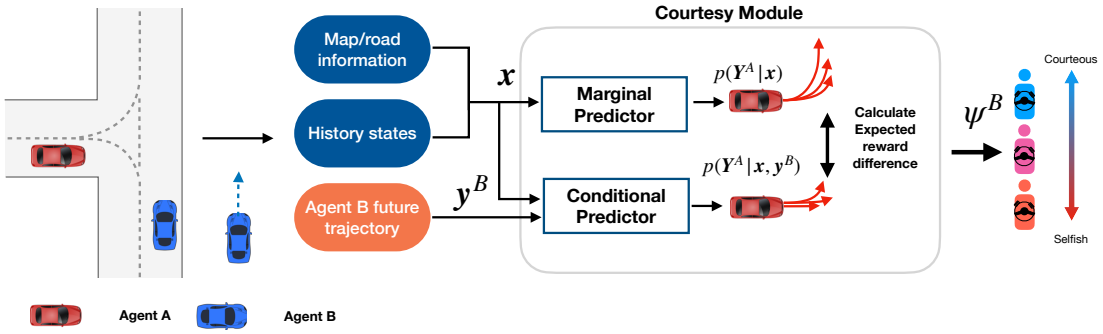

Figure 3.2: Our proposed auto-labeling framework. The proposed auto-labeling framework estimates and labels the courtesy values of agent B in a sample of interactive pairs by leveraging predictors trained on real-world data. Marginal and conditional trajectory distributions of agent A are estimated by querying the respective predictors, and the difference in the expected reward values under the two distributions is used to quantify the courtesy level. This approach provides an estimate of the courtesy value without requiring humans to manually label the courtesy values.

we query a marginal predictor for an estimated marginal distribution $p(\boldsymbol{y}^A|\boldsymbol{x})$, and query a conditional predictor for an estimated conditional distribution $p(\boldsymbol{y}^A|\boldsymbol{x}, \boldsymbol{y}^B)$. With the estimated distributions, we can then estimate the courtesy value of $\boldsymbol{y}^B$ using Eqn. (3.2). We denote the overall courtesy computation operation by $J_{\boldsymbol{\phi}}(\cdot, \cdot)$, where $\boldsymbol{\phi}$ denotes the parameters of the predictors:

$$\psi = J_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}^B). \tag{3.5}$$

Note that the auto-labeling procedure does not impose any restrictions on the prediction models used. In our experiment, we adopted the state-of-the-art Multipath++ [50] as the backbone prediction model, where the predicted trajectory distribution is represented as a Gaussian Mixture Model (GMM). The original Multipath++ model is designed for marginal prediction. We accommodate it to support both marginal and conditional predictions following the practice in [43]. Specifically, we add an additional future encoder to encode $\boldsymbol{y}^B$. Under the conditional prediction mode, the encoded embedding is fused with the embedding of the other input features. Under the marginal prediction mode, we turn off the future encoder so that the model does not take $\boldsymbol{y}^B$ as input. We follow the training scheme in [43] to train the model for both inference modes simultaneously.

3.3.4 Socially-Controllable Behavior Generation Model

We now present the SCBG model architecture and its training pipeline, which are summarized in Fig. 3.3. We implement the SCBG model based on a Multipath++ backbone, which consists of a scene encoder and a multi-context gating (MCG) trajectory decoder. The original MCG decoder in Multipath++ utilizes a set of learned anchor embeddings for multimodal

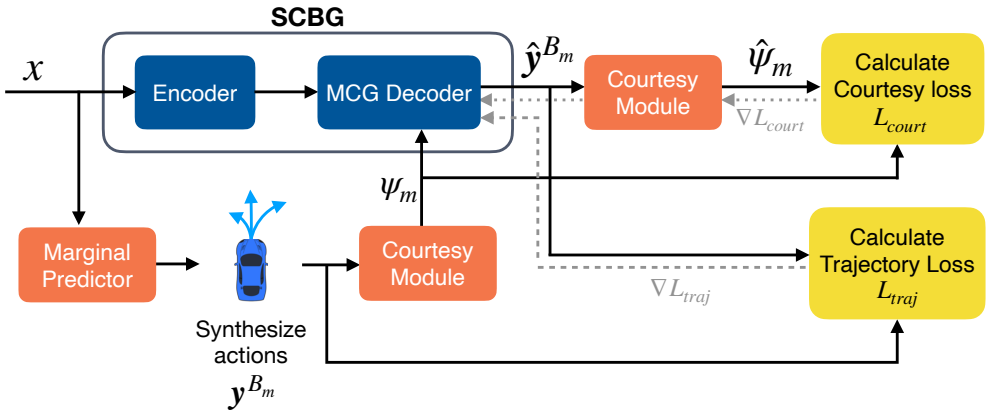

Figure 3.3: The SCBG model architecture and training pipeline. The SCBG model takes as input the scene observation $\boldsymbol{x}$ and conditions on the courtesy level to produce a trajectory $\hat{\boldsymbol{y}}^{\boldsymbol{B_m}}$. The model is trained by matching the generated trajectories and their courtesy values with the labeled ones. To enhance the model performance, we leverage a marginal predictor to augment the training data with predicted trajectories.

trajectory prediction. Since our SCBG model is designed for closed-loop simulation tasks, it is crucially important to ensure the long-term stability of the closed-loop behavior [4, 5, 17]. Modeling the multimodality at the trajectory level could be troublesome for closed-loop simulation because we need to ensure the consistency in modality across nearby timesteps, which requires non-trivial adaptation of the MCG decoder. For simplification, we remove the anchor embeddings and let the model output a single-modal trajectory given an input courtesy value in this work. In the future, we plan to introduce a high-level goal inference module, such as the one in [5], to model the multimodality when extending the current framework for socially-controllable closed-loop simulation. In particular, we want to highlight two key components of the training process.

Data Augmentation Real-world data can only provide one sample of the future trajectory per scenario, resulting in one sampled behavior that corresponds to a single courtesy value. However, to achieve controllable behavior generation, the model needs to be reliable under a diverse set of input courtesy values. To address this challenge, we augment the training dataset with synthesized trajectories sampled from the marginal predictor. While in auto-labeling, we query the marginal predictor for $p(\boldsymbol{y}^A|\boldsymbol{x})$, in this context, we query the marginal predictor for $p(\boldsymbol{y}^B|\boldsymbol{x})$ and sample M trajectories from the predicted distribution. By sampling from the marginal distribution, we augment the dataset with plausible trajectories of diverse courtesy levels. Our experiments demonstrate that data augmentation plays a critical role in ensuring the generated trajectories indeed correspond to the input courtesy value.

Loss Function During training, we use a loss function that consists of two parts. The first part computes the error between the generated and labeled trajectories. Specifically, given a sample from the dataset, $\boldsymbol{x}, \boldsymbol{y}^{B,0} \sim \mathcal{D}$, we denote the M synthesized trajectories as $\boldsymbol{y}^{B,m}, m = 1, \ldots, M$. And we denote the labeled courtesy values for $\boldsymbol{y}^{B,m}$ as ψ_m. The trajectory loss, L_{traj}, is then defined as:

$$L_{traj} = l\left(\boldsymbol{y}^{B,0}, \hat{\boldsymbol{y}}^{B,0}\right) + \alpha \sum_{m=1}^{M} l\left(\boldsymbol{y}^{B,m}, \hat{\boldsymbol{y}}^{B,m}\right), \tag{3.6}$$

where $\hat{\boldsymbol{y}}^{B,m}$ is the trajectory generated by the SCBG model given the courtesy value ψ_m, and l is a differentiable loss function, such as mean squared error. The coefficient α is a hyperparameter used to balance the losses between the ground truth and synthesized trajectories. And we use Huber loss as the loss function $l(\cdot, \cdot)$.

The second part of the loss function compares the input courtesy values against the courtesy values of the generated trajectories, which ensures the generated trajectories indeed match the input courtesy values. The courtesy loss L_{court} is defined as:

$$L_{court} = \sum_{m=0}^{M} \|\psi_m - J_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}^{B,m})\|^2. \tag{3.7}$$

Since the courtesy computation module $J_{\boldsymbol{\phi}}(\cdot, \cdot)$ is differentiable, we can directly leverage it to compute the courtesy loss and backpropagate through it for gradient computation. The overall loss function is then defined as:

$$L = L_{traj} + \beta L_{court}, \tag{3.8}$$

where the coefficient β is a hyperparameter balancing the trajectory loss and the courtesy loss.

3.3.5 Courtesy Range Predictor

Since the reward distribution varies across scenarios, so does the range of feasible courtesy values. As a result, feeding arbitrary courtesy values to the SCBG model may lead to out-of-distribution input which results in unreliable generated behavior. To this end, we train a courtesy range predictor to predict the interval of feasible courtesy values for a given scenario. Since we do not have access to the ground-truth distributions, we use quantile regression [51] to estimate the statistics of the courtesy value from data. As shown in Fig. 3.4, the estimated range is used to normalize the courtesy values across scenarios so that the user only needs to specify the level of courtesy as a value from the unit interval. The range predictor shares the encoder of the SCBG model. During training, we freeze the parameters of the encoder and train a new decoder to predict the 0.1 and 0.9 quantiles of ψ. We follow

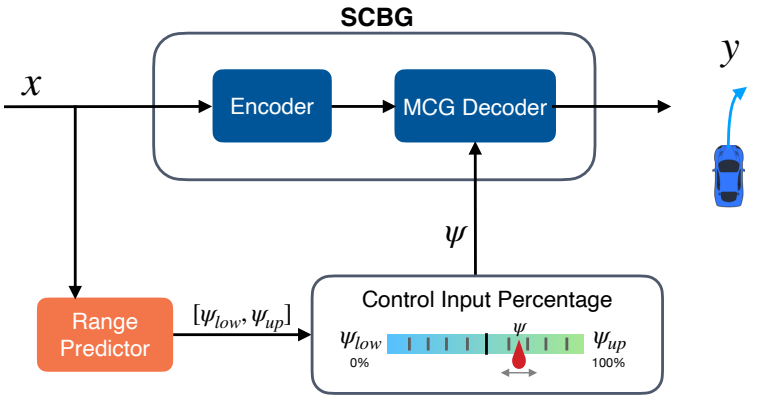

Figure 3.4: The paradigm of behavior generation at inference time. The range predictor predicts the feasible range of courtesy values for a given scenario. This range is then used to scale the user input percentage to the appropriate input value, which is fed into the SCBG model.

the practice in [52] and use the pinball loss function to train the range predictor:

$$L_\tau = \begin{cases} (\tau - 1) \cdot (\psi - \hat{\psi}_\tau), & \text{if } \psi < \hat{\psi}_\tau \\ \tau \cdot (\psi - \hat{\psi}_\tau) & \text{otherwise,} \end{cases} \tag{3.9}$$

where τ denotes the target quantile (i.e., 0.1 and 0.9) and $\hat{\psi}_\tau$ denotes the predicted courtesy value at the τ-quantile.

3.4 Experiments

In this section, we conduct experiments to validate our proposed SCBG framework on real-world driving data.

3.4.1 Dataset

We use the Waymo Open Motion Dataset (WOMD) [13]. The dataset provides a subset with labels identifying a pair of interacting agents in the scenarios. We refer to this subset as the interactive subset. In our work, we focus on vehicle-to-vehicle interaction and vehicle behavior generation while we still include the historical trajectories of the other types of agents (i.e., pedestrians and cyclists) in the model input.

3.4.2 Training Marginal and Conditional Predictors

We build our prediction model on the open-source implementation of Multipath++ [53]. We follow the practice in [43] and train a prediction model supporting both marginal and

conditional predictions during inference. We report the prediction errors (measurement time of 8s) on the interactive subset of the validation set in Table 3.1. The evaluation metrics are defined as in [13] and computed with respect to six predicted trajectory samples, except for the ADE metric. The ADE metric measures the error between the ground-truth trajectory and the predicted sample with the highest probability. Since the SCBG model generates a single trajectory, we list the ADE values here as references to validate the realism of the trajectories generated by the SCBG model, which will be discussed later in §3.4.4. Notably, the prediction errors are comparable with the results reported in [53], which achieved a top-10 performance in the Waymo Motion Prediction Challenge 2022.

3.4.3 Auto-labeling Courtesy Value

Using the trained predictor, we auto-labeled the courtesy values of the data, following the method discussed in §3.3.2. Since the courtesy values are non-trivial only in interactive scenarios, we auto-labeled the courtesy values and trained the SCBG model on the interactive subset. In our experiment, we use average speed as the reward function when quantifying the level of courtesy in Eqn. (3.2). The average speed indicates the agent's progress along its route, which is the primary driving target for on-road driving. Fig. 3.5 shows the histogram of the extracted courtesy values on the interactive subset. It is worth noting that the histogram concentrates around zero. One reason is that many scenarios contain weak or non-interactive behaviors, where the trajectory of one agent does not substantially influence the expected behavior of the other agent. In addition, trajectories that follow nominal interaction patterns in the dataset may also result in courtesy values close to zero, since the observed behavior approximately matches what other agents would already expect under the marginal interaction distribution.

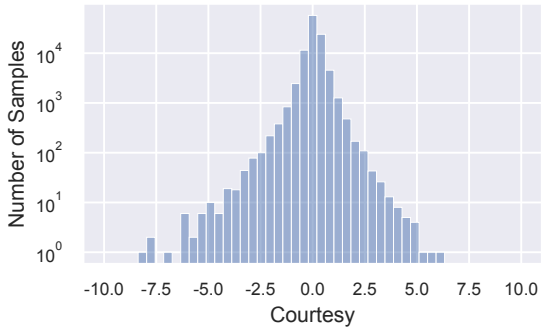

Figure 3.5: Histogram of courtesy values on the interactive subset of WOMD.

3.4.4 Socially-Controllable Behavior Generation

Implementation Details Since a large portion of the data has relatively low absolute courtesy values (Fig. 3.5), we split the data into two subsets with a threshold absolute courtesy value of 2. During training, we sample 50% of the batch data from each subset to avoid overwhelming the training data with trivial samples. To train the SCBG model, we load the encoder parameters from the Multipath++ model and only train the decoder for SCBG.

Evaluation Metrics We want to validate that the proposed SCBG model can achieve *socially-controllable* trajectory generation and that the generated trajectories are *realistic*. To evaluate the model's controllability, we compare the courtesy values of the generated trajectories with the input courtesy values. Since we do not have access to the ground-truth trajectory distributions, we leverage the auto-labeling method to define a data-driven evaluation metric for controllability, denoted by CourtesyMSE, which is the mean squared error (MSE) between the input and auto-labeled courtesy values of the generated trajectory:

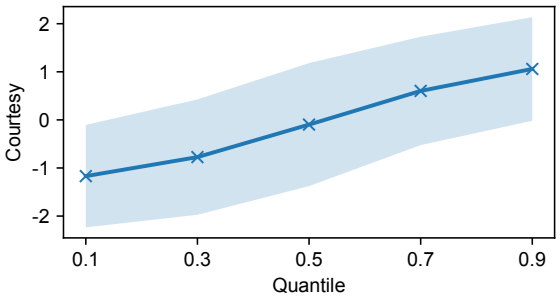

Figure 3.6: The relationship between input quantile and the courtesy value of the generated trajectory in the high-courtesy subset. The courtesy value increases as the input quantile increases, demonstrating the controllability of SCBG.

$$\text{CourtesyMSE} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|\Psi_n|}\sum_{\psi_{n,i}\in\Psi_n}(\psi_{n,i}-\hat{\psi}_{n,i})^2, \tag{3.10}$$

where $\hat{\psi}_{n,i} = J_{\boldsymbol{\phi}}\left(\boldsymbol{x}_n, g_\theta(\boldsymbol{x}_n, \psi_{n,i})\right)$ and N is the number of samples in the dataset. Ψ_n is a set of selected courtesy values of interest. We consider three strategies to define Ψ_n for different perspectives of evaluation:

- *Data*: We define the set Ψ_n with the courtesy values of the ground-truth and augmented trajectories:
$$\Psi_{n,data} = \left\{J_\phi\left(\boldsymbol{x}_n, \boldsymbol{y}_n^{B,m}\right)\right\}_{m=0}^{M}.$$
The resulting metric quantifies controllability over input courtesy values following the data distribution, eliminating the influence of infeasible input courtesy values.

- *Range*: We use the range predictor to predict the 0.1 and 0.9 quantiles of the courtesy values and interpolate between the predicted quantiles with a fixed interval:
$$\Psi_{n,range} = \left\{\hat{\psi}_{0.1}(\boldsymbol{x}_n), \hat{\psi}_{0.1+\delta\tau}(\boldsymbol{x}_n), \cdots, \hat{\psi}_{0.9}(\boldsymbol{x}_n)\right\}.$$
The resulting metric reflects the controllability of the SCBG model in a practical setting where the range of feasible courtesy values is unknown.

- *Arbitrary*: The input courtesy values are selected by interpolating between the minimum and maximum courtesy values of the entire dataset, denoted by $\psi_{\min}$ and $\psi_{\max}$, in a sample-agnostic way:
$$\Psi_{arbitrary} = \{\psi_{\min}, \psi_{\min} + \delta\psi, \cdots, \psi_{\max}\}.$$
The resulting metric serves as a baseline showing the model performance without the range predictor.

To evaluate realism, we follow the common practice [14, 21] to compare the generated trajectory against the ground truth from data. Specifically, we compare the trajectory generated with the courtesy value of the ground-truth trajectory against the ground-truth trajectory. We use ADE to quantify the trajectory error and define the metric as follows:

$$\text{TrajADE} = \frac{1}{TN}\sum_{n=1}^{N}\sum_{t=1}^{T}||\boldsymbol{y}_{n,t}^{B,0} - \hat{\boldsymbol{y}}_{t}^{B,0}||_2. \tag{3.11}$$

We can then evaluate the realism of the generated trajectories by comparing TrajADE with ADE reported in Table 3.1.

Mode	ADE (m)	minADE (m)	minFDE (m)	MR (%)	mAP
Marginal	3.18	1.16	2.52	20.6	0.262
Conditional	3.13	1.11	2.35	19.5	0.274

Table 3.1: Prediction Model Performance

model	CourtesyMSE			TrajADE
	data	*range*	*arbitrary*	
①	0.062 ± 0.321	0.154 ± 0.470	10.35 ± 9.40	3.27 ± 2.60
①+②	0.045 ± 0.252	0.137 ± 0.467	9.97 ± 9.26	3.22 ± 2.91
①+②+③	0.034 ± 0.172	0.120 ± 0.376	9.77 ± 9.09	3.02 ± 2.72

①: Baseline Model ②: Data Augmentation ③: Courtesy Loss

Table 3.2: Socially-Controllable Behavior Generation Evaluation Results

3.4.5 Quantitative Results

Table 3.2 summarizes the results of three model variants: 1) the baseline variant without data augmentation or courtesy loss; 2) the baseline model with data augmentation; and 3) the complete SCBG model with both data augmentation and courtesy loss. The results validate the effectiveness of the proposed data augmentation scheme and the courtesy loss. Both modules significantly reduce courtesy and trajectory errors, implying improved controllability and realism. Interestingly, the courtesy loss helps reduce TrajADE even though it does not directly penalize trajectory errors, which shows that matching the courtesy values provides informative supervision signals that guide the model to better capture the driving behaviors during training.

Table 3.2 also highlights the important role of the range predictor by comparing the courtesy errors under different strategies for constructing Ψ_n. Without the range predictor, the

courtesy error significantly increases (i.e., $\Psi_n = \Psi_{arbitrary}$) because of the out-of-distribution input courtesy values. Meanwhile, the courtesy error with the range predictor (i.e., $\Psi_n = \Psi_{n,range}$) is comparable to the courtesy error when the courtesy values from the data are used (i.e., $\Psi_n = \Psi_{n,data}$).

Overall, the proposed SCBG model enables socially controllable and realistic driving behavior generation, which is validated by the low CourtesyMSE value of 0.120 and a TrajADE value smaller than the ADEs of the trajectory prediction models reported in Table 3.1. We further illustrate the model's controllability in the practical setting where users give quantile commands as in Fig. 3.4. As shown in Fig. 3.6, the courtesy value of the generated trajectory effectively increases with the quantile input. The correlation coefficient between these two values is 0.60 on the entire interactive validation subset and 0.79 on the high-courtesy subset (i.e., data with absolute courtesy values larger than 2).

3.4.6 Qualitative Analysis

In this section, we visualize some representative examples showing that SCBG is able to generate different courteous behaviors according to the scenarios. In Fig. 3.7, we plot the trajectories generated for the controlled agent B with different input courtesy quantiles. We also plot Agent A's ground-truth trajectories observed in the dataset to visualize the nominal behavior of Agent A. For example, the controlled agent attempts to merge into the lane where Agent A is driving in Cases 1-3. The controlled agent accelerates to prevent blocking Agent A when the courtesy level increases. Interestingly, when the input quantiles are 0.7 and 0.9 in Case 1, the controlled agent further switches its lane to the right, creating more space for Agent A. In contrast, the controlled agent slows down and yields when the courtesy level increases in Cases 4-5. Despite the diverse behavior patterns observed in the visualization, all types of courteous behavior generated by the SCBG model result in a higher reward for the other agent (Fig. 3.7(b)).

3.5 Discussion

We demonstrate that the proposed SCBG framework can control the courtesy level of the generated driving behavior, which is a crucial first step toward achieving socially controllable traffic simulation. The proposed SCBG framework offers a key advantage in that it can be easily scaled to handle multi-agent interactions by calculating the joint influence of the controlled agent on all the other agents. One limitation of the current framework is that the auto-labeling method is affected by the causality issue of conditional behavior prediction [43, 54]. Specifically, a conditional prediction model cannot differentiate between the correlation and causation of two agents' trajectories. One potential solution is incorporating prior knowledge of causal relations when designing the closed-loop simulation [42], which can be directly modeled by recent advances in the autoregressive model family with causal masking [18, 19]. In addition, the current framework does not account for situations where

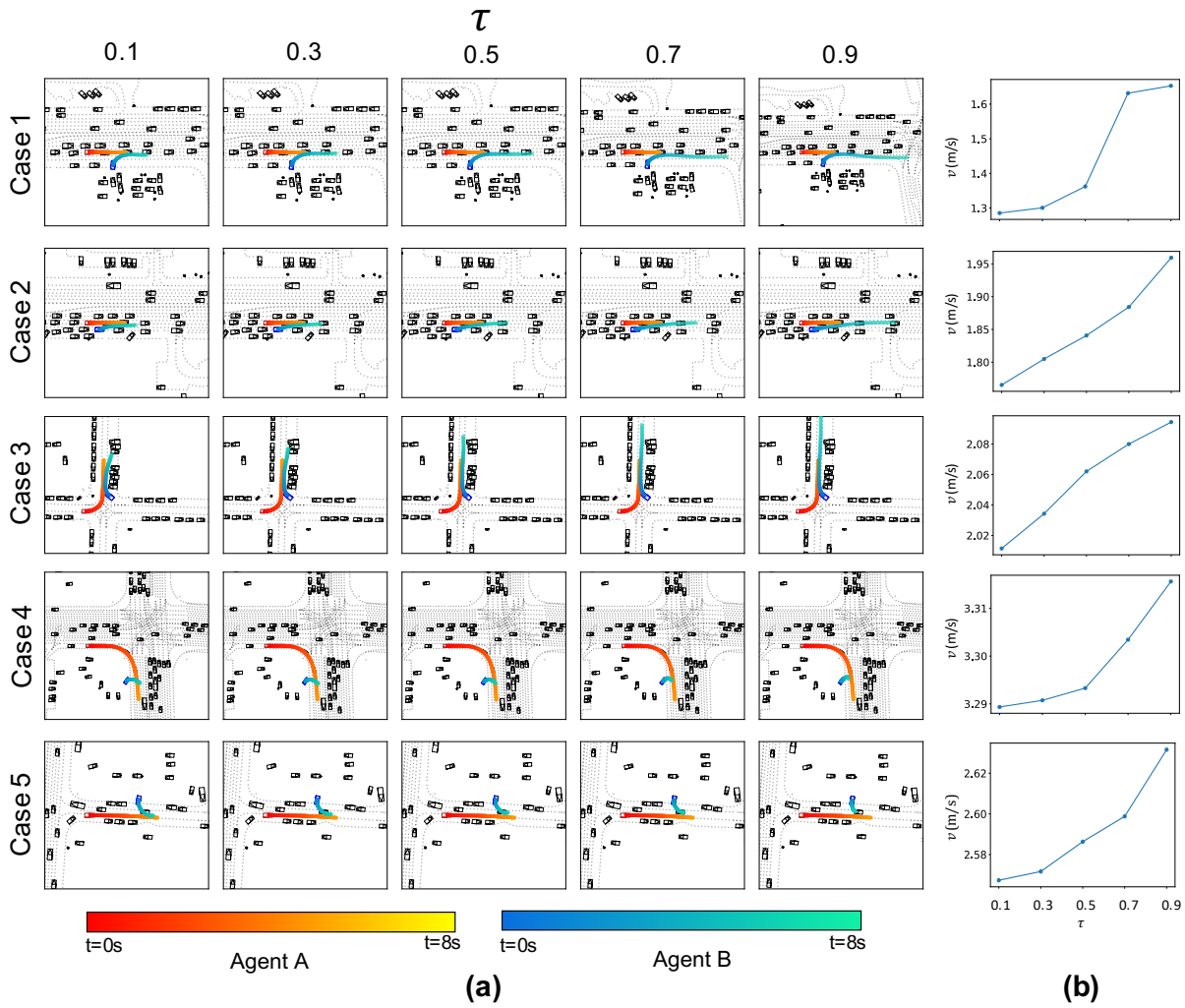

Figure 3.7: (a) Visualization of the generated trajectories by control courtesy value. The blue line indicates the trajectory of the controlled agent B, while the red line represents the ground truth trajectory of agent A. This visualization shows that as the input courtesy value increases, agent B increases agent A's reward by either changing lanes (Case 1), increasing its speed (Cases 2-3), or yielding (Cases 4-5). (b) Relationship between the input courtesy quantiles and the predicted average speed of agent A, given agent B's trajectory.

different interactive behaviors can lead to the same courtesy value, as it relies on generating a deterministic trajectory based on a courtesy value. One extension is to incorporate multi-modal behavior generation, for example, through diffusion or autoregressive models, to better capture the diversity of possible interactions.

3.6 Conclusion

In this chapter, we introduce socially-controllable behavior generation (SCBG), a model for generating realistic driving behavior based on desired courtesy levels. We propose a novel data-driven quantification of courtesy for socially interactive traffic scenarios, which enables auto-labeling of the latent courtesy values. We present a novel training algorithm to train the SCBG model from large-scale real-world driving data. In particular, we introduce a novel courtesy loss to improve the controllability and realism of the trained model. Our experimental results show that the SCBG model can learn to generate realistic driving behaviors with desired courtesy levels from real-world data. By utilizing this framework, we believe that robots will be able to interact with humans with diverse social characteristics in a safe and smooth manner.

Part II

Controllable Generative Modeling for Counterfactual Simulation

Chapter 4

Safe-Sim: Safety-Critical Closed-Loop Traffic Simulation with Controllable Adversaries

This chapter is based on the paper "SafeSim: Safety-Critical Closed-Loop Traffic Simulation with Diffusion-Controllable Adversaries" [55], written in collaboration with Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker.

4.1 Introduction

Building on the previous chapter, which focused on modeling and controlling social behavior in simulation, we now consider another critical dimension of controllability in data-driven simulation: the generation of safety-critical, long-tail scenarios.

A key safety feature of autonomous vehicles (AVs) is their ability to navigate near-collision events in real-world scenarios. However, these events rarely occur on roads and testing AVs in such high-risk situations on public roads is unsafe. Therefore, simulation is indispensable in the development and assessment of AVs, providing a safe and reliable means to study their safety and dependability. A critical aspect of simulation is modeling the behavior of other road users, since AVs must learn to interact with them safely.

A common method of safety-critical testing of AVs involves manually designing scenarios that could potentially lead to failures, such as collisions. While this approach allows for targeted testing, it is inherently limited in scalability and lacks the comprehensiveness required for thorough evaluation [39, 56, 57]. Some recent works focus on automatically generating challenging scenarios that cause planners to fail, but their emphasis has been mostly on static scenario generation rather than dynamic, *closed-loop* simulations. This results in a critical gap: the behavior of other agents often does not adapt or respond to the planner's actions, which is essential for a comprehensive safety evaluation. Furthermore, the results from these simulations often lack *controllability*, typically producing only a single adversarial outcome

Method	**Safety-Critical**	**Controllable**	**Controllable Adversary**	**Evaluate Planner**	**Closed-Loop**	**Real-World**
CTG [21]	×	✓	×	×	✓	✓
CTG++ [22]	×	✓	×	×	✓	✓
STRIVE [29]	✓	×	×	✓	✓	✓
DiffScene [27]	✓	✓	×	✓	×	×
SAFE-SIM (Ours)	✓	✓	✓	✓	✓	✓

Table 4.1: **Comparison of methods.** Our contribution is the development of a framework for (a) safety-critical (b) closed-loop (c) controllable adversarial simulations. These aspects are not concurrently present in previous frameworks. We formulate a novel partial diffusion with novel guidance functions for stable long-term simulation and are the first to enable an ego planner to be tested against controllable adversaries with varied behavior patterns.

per scenario without the flexibility to explore a range of conditions and responses.

In this work, we introduce SAFE-SIM[1], a *closed-loop* simulation framework for generating safety-critical scenarios, with a particular emphasis on *controllability* and *realism* for the behavior of agents, which allows simulations over a long-horizon as needed to evaluate AV planning algorithms (Fig. 4.1). Different from prior works [21, 22, 27, 29] that primarily adhere to rule-constraint satisfaction, our approach enhances *controllability* by modulating adversarial vehicle behaviors within identical scenarios, thereby facilitating a broader exploration of potential outcomes. See table 4.1 for a comprehensive comparison of these approaches.

Our approach builds upon recent developments in controllable diffusion models [21, 30, 58]. Specifically, we adopt a test-time guidance to direct the denoising phase of the diffusion process, using the gradients from differentiable objectives to enhance scenario generation, enabling generation of adversarial scenarios in which an adversarial agent collides with the ego agent behaving according to a specific planning policy. Additionally, we develop a novel approach, which we refer to as Partial Diffusion that introduces trajectory proposals into the diffusion process to provide a high degree of controllability over the type of collision scenario. Overall, our balanced integration of adversarial objectives with regularization during the guidance phase combined with Partial Diffusion allows for refined control over the conditions of the generated scenarios, ensuring both their realism and relevance to safety-critical testing.

In our study, we use the nuScenes [12] and nuPlan [59] datasets to evaluate the efficacy of our method in generating safety-critical closed-loop simulations. Our results demonstrate a marked improvement in the controllability and realism of scenarios compared to previous adversarial scenario generation methods. Furthermore, we showcase the advantage of our proposed framework in varying the safety-criticality and collision types of scenarios. These attributes make our approach particularly well-suited for the closed-loop simulation of AVs, providing a more reliable and comprehensive framework for safety evaluation.

[1] `https://safe-sim.github.io/`

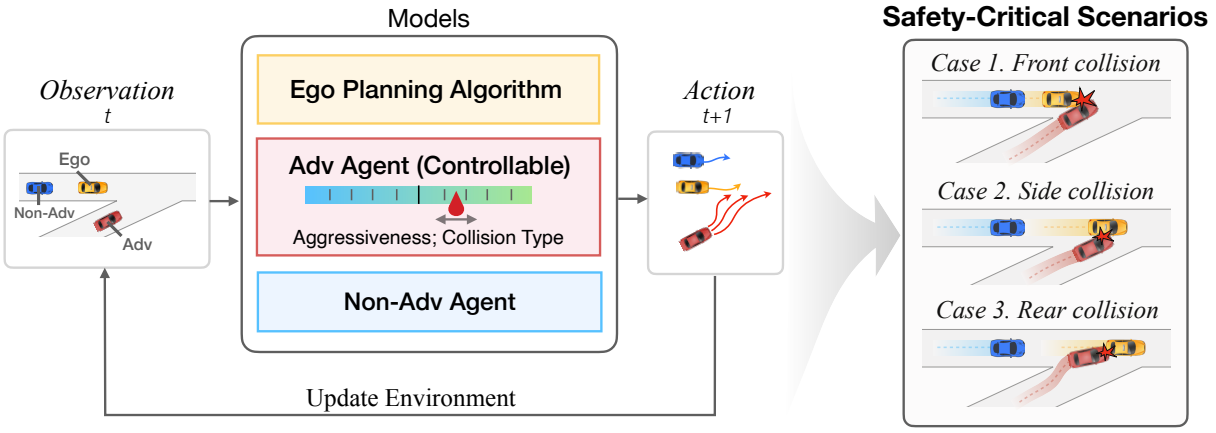

Figure 4.1: **Overview of Safe-Sim Framework for Controllable Safety-Critical Closed-Loop Simulation.** This framework evaluates a planner within scenarios featuring multiple controllable reactive agents. These agents have two distinct roles: adversarial agents, which actively challenge the planner by exhibiting controllable adversarial behaviors such as specific collision types and levels of aggressiveness, and non-adversarial agents, which follow normal driving behavior to maintain the realism of the entire scene. Such a setup facilitates the generation of various realistic, interactive, and safety-critical scenarios, providing a thorough evaluation of the planner's capabilities.

4.2 Related Work

4.2.1 Diffusion Models For Traffic Simulation

Recent advances in diffusion models have enabled powerful generative modeling with improved controllability through techniques such as classifier and classifier-free guidance [60, 61]. These controllable diffusion models have been employed for planning and traffic simulation [21, 22, 58] via guidance. We adopt trajectory diffusion models to develop a novel approach for generating safety-critical realistic traffic simulations in which an adversarial agent collides with an ego planner agent. Other works [25, 62] use diffusion with guidance or conditioning to achieve controllable scene initialization. In contrast, we emphasize closed-loop, controllable adversarial behavior simulation based on real-world data initialization rather than scene initialization.

4.2.2 Safety-Critical Traffic Simulation

Safety-critical traffic generation plays a crucial role in training and evaluating AV systems, enhancing their capability to navigate diverse real-world scenarios and enhancing robustness. Gradient-based methods that leverage back-propagation to create safety-critical scenarios have been proposed to evaluate AV prediction and planning models [63, 64]. Hanselmann et al. use

kinematic gradients to modify vehicle trajectories, with the goal of improving the robustness of imitation learning planners [63]. Cao et al. have developed a model with differentiable dynamics, enabling the generation of realistic adversarial trajectories for trajectory prediction models through backpropagation techniques [64]. Black-box optimization approaches include perturbing actions based on kinematic bicycle models [39] and using Bayesian Optimization to create adversarial self-driving scenarios that escalate collision risks with simulated entities [65]. Zhang et al. target trajectory prediction models via white- and black-box attacks that adversarially perturb real driving trajectories [66]. For an extensive review of this topic, we direct readers to Ding et al. [67].

The field has recently seen advancements in data-driven methods for safety-critical scenario generation [27, 28, 29]. For instance, Xu et al. introduce a diffusion-based approach in CARLA, applying various adversarial optimization objectives to guide the diffusion process for safety-critical scenario generation [27]. Rempe et al. proposed STRIVE to utilize gradient-based adversarial optimization on the latent space, constrained by a graph-based CVAE traffic motion model, to generate realistic safety-critical scenarios for rule-based planners [29]. However, a common limitation in these approaches is the absence of closed-loop interaction, essential for accurately simulating interactive real-world driving.

4.3 Problem Formulation

We consider a simulated interactive traffic scenario consisting of N agents; one is the ego vehicle controlled by the *planner* π, and the remaining $N-1$ are reactive agents modeled by a function g. Our objective is to create a safety-critical *closed-loop* collision simulation, where reactive agents demonstrate *realistic*, *controllable* behavior. Of the $N-1$ reactive agents, one or a subset is considered the adversarial agents (denoted as agent a), meant to collide with the ego vehicle.

The adversarial agent, formulated within the reactive agent model g, is governed by an adversarial term designed (shown in fig. 4.2 and detailed in section 4.5) to be both controllable and adversarial to the planner π. This setup allows the adversarial agent to pose direct challenges to π, testing its resilience in complex scenarios. Concurrently, the other non-adversarial agents, also controlled by g with varying parameters, emulate authentic, reactive behaviors, thus enriching the simulation scenario with realistic and diverse traffic conditions. The dual role of the adversarial agent and non-adversarial agents ensures that while it challenges π, the overall simulation environment plausibly represents real-world driving conditions.

At any given timestep t, the states of the N vehicles are represented as $\mathbf{s}_t = [\mathbf{s}_t^1, \ldots, \mathbf{s}_t^N]$, where $\mathbf{s}_t^i = (x_t^i, y_t^i, v_t^i, \theta_t^i)$ indicates the 2D position, speed, and yaw of vehicle i. The corresponding actions for each vehicle are $\mathbf{a}_t = [\mathbf{a}_t^1, \ldots, \mathbf{a}_t^N]$, with $\mathbf{a}_t^i = (\dot{v}_t^i, \dot{\theta}_t^i)$ representing the acceleration and yaw rate. To predict the state at the next timestep $t+1$, a transition function f is used, which computes $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ based on current state and action. We adopt unicycle dynamics as the transition function.

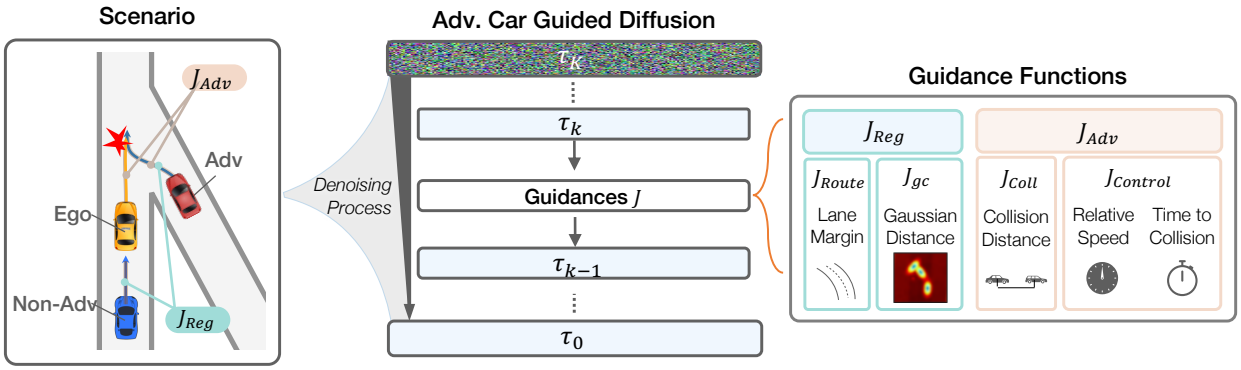

Figure 4.2: **Guided Diffusion Process for the Adversarial Agent.** This process optimizes the adversarial agent's trajectory using the adversarial cost function J_{adv} to the ego vehicle. In particular, we introduce J_{control} to vary the adversarial behavior. Simultaneously, it applies regularization through J_{reg} for maintaining realism.

Each agent's decision context is $\mathbf{c}_t^i$, which includes the agent-centric map I^i and the T_{hist} historical states of neighboring vehicles from time $t - T_{\text{hist}}$ to t, defined as $\mathbf{s}_{t-T_{\text{hist}}:t} = \{\mathbf{s}_{t-T_{\text{hist}}}, \ldots, \mathbf{s}_t\}$. In closed-loop traffic simulation, each agent continuously generates and updates its trajectory based on the current decision context $\mathbf{c}_t^i$. After generating a trajectory, the simulation executes the first few steps of the planned actions before updating $\mathbf{c}_t^i$ and re-planning. See section 4.6.2 for more implementation details.

Planner π The planner π determines the ego vehicle's future trajectory over a time horizon t to $t+T$. The planned state sequence is denoted by $s_{t:t+T}^1 = \pi(\mathbf{c}_t^1)$, where $\pi(\mathbf{c}_t^1)$ processes the historical states and map data within $\mathbf{c}_t^1$ to plan future states based on the current scene context.

Reactive Agents g The reactive agent model g, parameterized by θ, is designed to simulate the behavior of the $N-1$ non-ego vehicles, represented by the set $\{s_{t:t+T}^i\}_{i=2}^N$. Each vehicle's state sequence, $s_{t:t+T}^i$, is generated by $g_\theta(\mathbf{c}_t^i, \psi_i)$, which incorporates the decision context $\mathbf{c}_t^i$ and a set of control parameters ψ_i unique to each agent. These parameters ψ_i enable the fine-tuning of individual behaviors within the simulation. In our approach, we train the model g on real-world driving data to ensure the trajectories it produces are not only controllable, supporting the generation of various safety-critical scenarios, but also realistic.

4.4 Diffusion Models for Traffic Simulation

For closed-loop safety-critical traffic simulation, the reactive agents, especially the adversarial agent, should be 1) controllable, and 2) realistic. With recent advances in controllable diffusion

models [21, 30, 58], we adopt trajectory diffusion models to generate realistic simulations.

We define the model's operational trajectory as τ, which comprises both action and state sequences: $\tau := [\tau_a, \tau_s]$. Specifically, $\tau_a := [a_0, \ldots, a_{T-1}]$ represents the sequence of actions, while $\tau_s := [s_1, \ldots, s_T]$ denotes the corresponding sequence of states. Following the approach described in [21], our model predicts the action sequence τ_a, and the state sequence τ_s can be derived starting from the initial state s_0 and dynamic model f.

A diffusion model generates a trajectory by reversing a process that incrementally adds noise. Starting with an actual trajectory τ_0 sampled from the data distribution $q(\tau_0)$, a sequence of increasingly noisy trajectories $(\tau_1, \tau_2, \ldots, \tau_K)$ is produced via a forward noising process. Each trajectory τ_k at step k is generated by adding Gaussian noise parameterized by a predefined variance schedule β_k [24]:

$$q(\tau_{1:K}|\tau_0) := \prod_{k=1}^{K} q(\tau_k|\tau_{k-1}), \tag{4.1}$$

$$q(\tau_k|\tau_{k-1}) := \mathcal{N}(\tau_k; \sqrt{1-\beta_k}\tau_{k-1}, \beta_k \mathbf{I}). \tag{4.2}$$

The noising process gradually obscures the data, where the final noisy version $q(\tau_K)$ approaches $\mathcal{N}(\tau_K; \mathbf{0}, \mathbf{I})$. The trajectory generation process is then achieved by learning the reverse of this noising process. Given a noisy trajectory τ_K, the model learns to denoise it back to τ_0 through a sequence of reverse steps. Each reverse step is modeled as:

$$p_\theta(\tau_{k-1}|\tau_k, \mathbf{c}) := \mathcal{N}(\tau_{k-1}; \mu_\theta(\tau_k, k, \mathbf{c}), \Sigma_k), \tag{4.3}$$

where θ are learned functions that predict the mean μ of the reverse step, and Σ_k is a fixed schedule. By iteratively applying the reverse process, the model learns a trajectory distribution, effectively generating a plausible future trajectory from a noisy start.

During the trajectory prediction phase, the model estimates the final clean trajectory denoted by $\hat{\tau}_0$. This estimated trajectory is used to compute the mean μ as described in [68]. For more details, see appendix A.4.

4.5 Diffusion Models for Safety-Critical Traffic Simulation

The diffusion model, once trained on realistic trajectory data, inherently reflects the behavioral patterns present in its training distribution. However, to effectively simulate and analyze safety-critical scenarios, there is a crucial need for a mechanism that allows for the controlled manipulation of agent behaviors [21, 30]. This is particularly important for generating adversarial behaviors and ensuring scene consistency in simulations.

4.5.1 Guiding Reactive Agents

Our approach specifically introduces guidance to the sampled trajectories at each denoising step, aligning them with predefined objectives $J(\tau)$. The concept of guidance involves using the gradient of J to subtly perturb the predicted mean of the model at each denoising step. This process enables the generation of trajectories that not only reflect realistic behavior but also cater to specific simulation needs, such as adversarial testing and maintaining scene consistency over extended periods. We adopt the reconstruction guidance (clean guidance) introduced from [58, 69]:

$$\tilde{\tau}_0 = \hat{\tau}_0 - \alpha \Sigma_k \nabla_{\tau_k} J(\hat{\tau}_0) \tag{4.4}$$

This strategy improves guidance robustness, yielding smoother, more stable trajectories without the usual numerical issues from noisy data.

In practice, diversifying the behavior of adversarial agents within the *same* scenarios is crucial for a thorough assessment of AVs. Despite the significance of this challenge, it remains largely unexplored in previous works [21, 29].

The loss function for the adversarial agents, $J(\tau)$, consists of a collision term J_{coll}, which encourages collisions between the adversarial agent and the ego agent, two control terms J_{v} and J_{ttc}, which control the relative speed and time-to-collision between the ego and adversarial agent respectively, a regularization term J_{Gauss}, which discourages collisions between the reactive agents, and a route guidance term J_{route}, which discourages the reactive agents from going outside the road:

$$J(\tau) = \rho \underbrace{(J_{\text{coll}} + J_{\text{v}} + J_{\text{ttc}})}_{J_{\text{adv}}(\tau)} + \underbrace{J_{\text{route}} + J_{\text{Gauss}}}_{J_{\text{reg}}(\tau)}, \tag{4.5}$$

where ρ denotes a scalar weight that determines whether a reactive agent behaves adversarially towards the ego agent, i.e., whether it attempts to collide with the ego agent.

Collision with Planner We define J_{coll} to encourage the collision between the adversarial agent and the ego agent, given by:

$$J_{\text{coll}} = -\sum_{t=1}^{T} d(t), \tag{4.6}$$

where $d(t)$ represents the distance between the ego and the adversarial agent at each time step of the planning horizon T. The adversarial agent is either pre-selected based on lane proximity or dynamically selected based on the distance to the ego agent. Details are in section 4.5.3.

Safety Criticality of Collisions We control the relative speed J_v between the ego and adversary at each time step (v_t^1 and v_t^a), and the time-to-collision (TTC) cost J_{ttc} [70] to

control the safety criticality of potential collisions, with the latter given by:

$$J_{\text{ttc}} = \sum_{t=1}^{T} -\exp\left(-\frac{\tilde{t}^2_{\text{col}(t)}}{2\lambda_t} - \frac{\tilde{d}^2_{\text{col}(t)}}{2\lambda_d}\right), \tag{4.7}$$

where $\tilde{t}_{\text{col}(t)}$ is the time to collision at time t, $\tilde{d}_{\text{col}(t)}$ is the distance to collision and λ_t and λ_d are bandwidth parameters for time and distance. This formula uses a constant velocity assumption. Intuitively, the time-to-collision cost favors scenarios with high relative speeds and challenging collision angles for the ego vehicle to avoid. For details of J_v and J_{ttc}, see appendix A.3.2.

Route Guidance Given an agent's trajectory τ and the corresponding route r—the predefined path on a lane graph from its starting point to its destination—we compute the normal distance of each point τ_t on the trajectory to the route at each timestep. We then penalize deviations from the route that exceed a predefined margin d. This process is captured by the following route guidance cost function:

$$J_{\text{route}}(\tau, r) = \sum_{t=1}^{T} \max(0, |d_n(\tau_t, r) - d_m|), \tag{4.8}$$

where $d_n(\tau_n, r)$ denotes the normal distance from the point τ_t on the trajectory to the nearest point on the route r at timestep t, and d_m represents the acceptable deviation margin from the route. In contrast to the off-road loss in prior studies [21], our proposed route guidance system more effectively indicates each agent's intended path, improving adherence to traffic rules as demonstrated in section 4.6. The flexibility of route guidance supports diverse agent interactions, such as modifying routes to encourage lane changes among reactive agents [71].

Gaussian Collision Guidance Given the trajectories of agents, we calculate the Gaussian distance for each pair of agents (i, j) at each timestep t from 1 to T. The Gaussian distance between the agents takes into account both the tangential (d_t) and normal (d_n) components of the projected distances. The aggregated Gaussian distance is:

$$J_{\text{Gauss}} = \sum_{t=1}^{T} \sum_{i,j}^{N} \exp\left(-\frac{1}{2\sigma^2}\left(\lambda \cdot d_t^{ij}(t)^2 + d_n^{ij}(t)^2\right)\right) \tag{4.9}$$

where $d_t^{ij}(t)$ and $d_n^{ij}(t)$ represent the tangential and normal distances from agent j's trajectory point at time t to agent i's heading axis, respectively, and σ is the standard deviation for these distances. In this formulation, λ is a scaling factor applied to the tangential distance $d_t^{ij}(t)$. This approach contrasts with the disk approximation method, which primarily penalizes the Euclidean distance between agents. By accounting for both tangential and normal components, the Gaussian collision distance method significantly reduces collision rate, which we discuss in section 4.6.

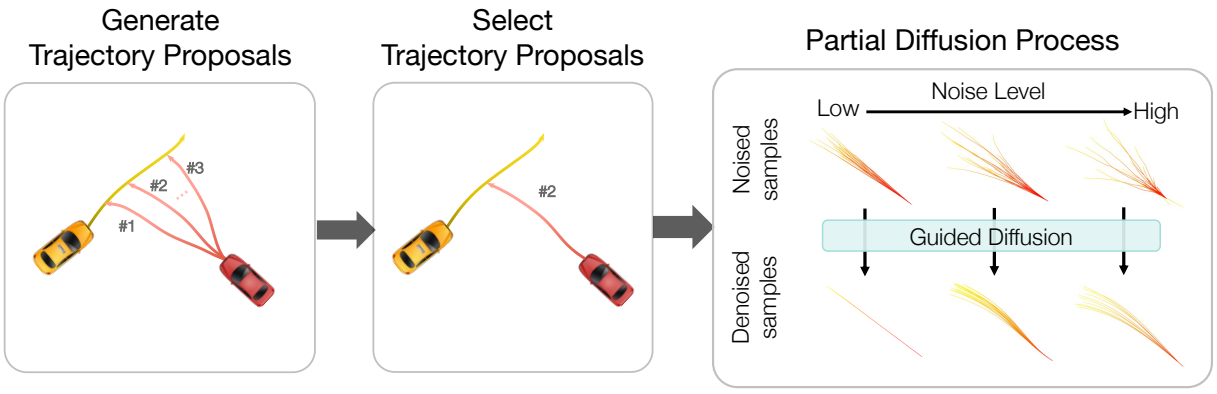

Figure 4.3: **Framework for Partial Diffusion.** We generate proposals based on domain knowledge (e.g., collision types). Users can adjust noise levels to balance between user control and the model's data distribution.

4.5.2 Partial Diffusion: Controlling Collision Types

We introduce a novel approach through a partial diffusion process, utilizing trajectory proposals to initiate the diffusion process. This methodology enables the variation in collision types by the adversarial agent within the diffusion, tailoring the adversarial outcomes to specific evaluation needs. The results are discussed in section 4.6.6.

Figure 4.3 illustrates our framework, which is divided into three main steps to generate trajectory proposals for various collision scenarios. First, we create initial trajectory proposals (τ_0) aimed at capturing different types of collisions. The next critical step involves setting the partial diffusion ratio γ, which defines the specific point in the process, $k_p = \gamma \cdot K$, at which we start modifying the trajectory. Starting from step k_p, we adjust the trajectory by adding a precise level of Gaussian noise $\epsilon \sim N(0, I)$: $\hat{\tau}_{k_p} = \sqrt{\bar{\alpha}_{k_p}}\tau_0 + \sqrt{1 - \bar{\alpha}_{k_p}}\epsilon$. The final stages include removing noise and using guided diffusion for the rest of the k_p steps to refine the trajectory into a realistic path that suits our collision scenario goals.

To generate the trajectory proposals, we develop a rule-based approach in which we first identify the centerlines of the ego and adversarial agent and then search for potential intersections of their respective centerlines. If such an intersection exists, we generate the proposals by selecting an acceleration value and lateral offset from the centerline that is likely to cause the desired collision type based on the projected plan of the ego agent. Note that the trajectory proposals are updated in a closed-loop manner to account for the interaction between the ego and the adversarial agent.

This method allows for precise control over the diffusion trajectory, enabling adversarial agents to create customized collision scenarios. Users can adjust γ to fine-tune the balance between explicit control and the model's trained data distribution.

4.5.3 Selecting Adversarial Agents

To effectively select adversarial agents for safety-critical simulation, we developed two strategies: dynamically selecting adversarial agents or selecting interacting agents. Inspired by [29], we propose dynamically adjusting the weighting coefficient ρ^i of J_{adv} during the guided diffusion process, encouraging a collision by minimizing the positional distance between controlled agents and the tested ego car:

$$\rho_{i,t} = \frac{\exp(-d^{i,1}(t))}{\sum_j \exp(-d^{j,1}(t))} \tag{4.10}$$

where $d^{i,1}(t)$ represents the euclidean distance between agent i and the ego vehicle at time t. Intuitively, the $\rho^{i,t}$ coefficients, defined by the softmax operation, identify a candidate agent to collide with the ego vehicle. The agent with the highest $\rho^{i,t}$ value is considered the most likely "adversary" based on proximity, and this formulation prioritizes causing a collision with this adversary. This approach weights the adversarial loss J_{adv} to highlight key interactions, preventing the unrealistic behavior of all agents acting adversarially towards the ego vehicle.

An alternate strategy selects interacting agents as adversaries based on their lane positions relative to the ego. Agents within a certain lane proximity to the ego are randomly chosen. In this scenario, the selected ith agent is treated as $\rho^{i,t} = 1$, with all others set to zero, for the duration of the simulation.

4.6 Experiments

We validate the efficacy of our proposed framework via experiments with real-world driving data. Our results demonstrate that the framework can generate *realistic* and *controllable* adversarial behavior to challenge the planner.

4.6.1 Dataset

We conduct our experiments on two large-scale real-world driving datasets: nuScenes [12], which consists of 5.5 hours of driving data from two cities, and nuPlan [59], which consists of 1500 hours of driving data from four cities. We train the model on scenes from the nuScenes train split and evaluate it on the scenes from the nuScenes validation splits and nuPlan mini validation splits. We focus on vehicle-to-vehicle interactions.

4.6.2 Implementation Details

Baselines. We compare our approach against STRIVE [29] using their open-source implementation and our re-implementation of DiffScene [27]. STRIVE is recognized for its proficiency in generating adversarial safety-critical scenarios using a learned traffic model and adversarial optimization in the latent space.

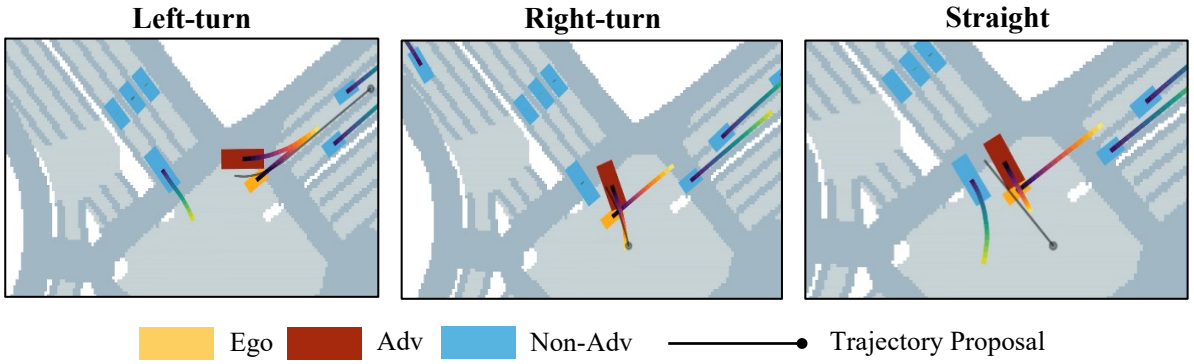

Figure 4.4: **Partial Diffusion Results of Rule-Based Planner on NuScenes Dataset.** The safety-critical scenarios show the framework's ability to create realistic and challenging situations, varying collision types based on different trajectory proposals. The gradient lines reflect the planned trajectories in the next 3.2 seconds.

Planner. Our experiments utilize a range of different planners: 1) A rule-based planner, as implemented in STRIVE [29], which operates on the lane graph and employs the constant velocity model to predict future trajectories of non-ego vehicles, generating multiple trajectory candidates and selecting the one least likely to result in a collision; 2) a hybrid planner BITS [5]; 3) PDM-Closed [72], the winner of the 2023 nuPlan Planning Challenge; 4) a deterministic learning-based Behavior Cloning (BC) planner, which utilizes a ResNet Encoder, followed by an MLP decoder to generate future trajectories; and 5) an Intelligent Driver Model (IDM) planner [9].

Diffusion Model. We follow the architecture described in [21]. We represent the context using an agent-centric map and past trajectories on a rasterized map. The traffic scene is encoded using a ResNet structure [15], while the input trajectory is processed through a series of 1D temporal convolution blocks in an UNet-like architecture, as detailed in [30]. The model uses $K = 100$ diffusion steps. During the inference phase, we generate a sample of potential future trajectories for each reactive agent in a given scene. From these, we select the trajectory that yields the lowest guidance cost. This process is referred to as *filtering*.

Closed-loop Simulation. The simulation framework for our experiments is built upon an open-source traffic behavior framework [5]. Within this framework, both the planner and reactive agents update their plans at a frequency of 2Hz.

4.6.3 Evaluation Metrics

Our goal is to validate that the proposed method can generate safety-critical scenarios that are both *realistic* and *controllable*. For *realism* assessment, in accordance with [21], we compare statistical data between simulated trajectories and actual ground trajectories. This involves calculating the Wasserstein distance between their driving profiles' normalized histograms,

focusing on the mean of mean values for three properties: longitudinal acceleration, lateral acceleration, and jerk. To evaluate *controllability*, we measure metrics related to parameters we can control, specifically **relative speed** and **time-to-collision cost** between the ego and adversarial agents.

Our method aims to evaluate the extent of control over adversarial behaviors within a single scenario. To do this, we focus on measuring ***collision diversity*** —a metric that quantifies the range of differences in collision angles, relative speeds, and collision points. By calculating the variance of these parameters within the same scenario, we can determine how diverse and controllable the adversarial behaviors are, ensuring the scenario's realism and controllability.

For a more nuanced understanding of *realism*, we analyze the **collision rate** – the average fractions of agents colliding, and the **offroad** rate – the percentage of reactive agents going off-road, both of which are considered failure rates [5]. All metrics are averaged across scenarios. For detailed definitions and metrics, see appendix A.3.

4.6.4 Evaluation of Safety-Critical Traffic Simulation with Baseline Methods

We compared our method with STRIVE [29] and DiffScene [27], utilizing the Lane-graph-based planner from STRIVE in table 4.2. The evaluation focused on collision rates between the ego and the adversarial agent ("Collision"), adversarial agent off-road rates ("Adv Offroad"), other agents' off-road rates ("Other Offroad"), collision relative speed between the ego and adversary ("Collision Rel Speed"), realism of all non-ego agents ("Realism"), and simulation time.

The results, presented in table 4.2, demonstrate that our method excels in all metrics compared to STRIVE, especially in collisions and realism. Compared to DiffScene, SAFE-SIM also exhibits a higher collision rate with better realism. Note that we trained models on nuScenes and tested them in nuPlan without additional fine-tuning. As illustrated in Figure 4.4, the qualitative examples from the NuScenes dataset demonstrate how our framework can challenge the rule-based planner with various driving situations. See appendix A.1 for more qualitative examples.

4.6.5 Safety-Critical Simulation with Different Planners

In table 4.3, we demonstrate our framework's ability to generate collisions across various planner types: rule-based, learning-based, and hybrid planners. Notably, the IDM planner exhibits the highest Ego-Adv collision rate. This heightened rate can be attributed to the IDM's focus on vehicles near the same lane, potentially overlooking other vehicles in the scene. Consequently, in scenarios with the IDM planner, our framework can induce collisions at relatively lower speeds.

Dataset	Method	Collision (%) ↑	Other Offroad (%)↓	Adv Offroad (%) ↓	Collision Rel Speed (m/s) ↓	Realism ↓	Time (s) ↓
nuScenes	SAFE-SIM	**43.2**	**1.8**	11.4	**-0.12**	**0.38**	$\mathbf{104.5 \pm 17.7}$
	STRIVE	36.4	2.2	11.4	5.52	0.85	427.2 ± 169.8
	DiffScene	18.2	11.4	**9.0**	16.4	0.52	105.4 ± 22.5
nuPlan	SAFE-SIM	**80**	**9.4**	11.7	6.75	**0.27**	$\mathbf{173.4 \pm 73.3}$
	DiffScene	56.7	14.0	**5.0**	**-2.81**	0.42	176.7 ± 77.5

Table 4.2: **Safety-critical Traffic Simulation.** We compare our approach against STRIVE [29] and DiffScene [27] for safety-critical traffic simulation with a rule-based planner. SAFE-SIM outperforms STRIVE on all metrics and demonstrates higher collision rates and better realism than DiffScene.

Planner	Ego-Adv Coll (%) ↑	Ego-Other Coll (%) ↑	Adv Offroad (%) ↓	Coll Speed (m/s) ↓	Ego Accel (m/s^2) ↓	Realism ↓
BC	38.8	37.3	9.0	2.47	**0.54**	0.79
IDM	**49.3**	**58.2**	3.0	**-0.40**	0.94	0.78
Lane-Graph	34.3	37.3	**1.5**	2.68	1.62	**0.57**
BITS	16.4	19.4	6.0	3.07	0.95	0.79
PDM-Closed	26.9	50.7	**1.5**	2.73	1.30	0.86

Table 4.3: **Safety-Critical Simulation for Different Planners.** SAFE-SIM can generate diverse safety-critical scenarios tailored to different planners, including rule-based, learning-based, and hybrid planners.

4.6.6 Evaluation: Controlling Safety-Criticality

A key feature of SAFE-SIM is its ability to generate controllable adversarial behaviors, offering variations not possible with previous methods.

Controlling Time-to-Collision. We control the orientation and relative speed together using the time-to-collision (TTC) cost, as described in section 4.5. We manipulate the scenario's safety-criticality by adjusting the relative weight of the TTC cost. To assess the impact of these adjustments, we measure the average TTC cost shortly before a collision occurs (0.5 seconds). Our observations, detailed in table 4.4, show that increasing the TTC weight raises the TTC cost. Notably, while the relative collision speed remains fairly consistent, the collision angle shifts, indicating a greater difficulty in avoiding ego-adversary collisions. Additionally, our method can control other aspects, such as relative speed, as detailed in appendix A.6.

TTC Cost Weight	**TTC Cost**	**Coll Speed** (m/s)	**Coll Angle** (deg)	**Coll Rate** (%) ↑	**Realism** ↓
0.0	0.18	2.45	-7.43	48.2	0.76
1.0	0.21	2.30	0.43	53.6	0.79
2.0	0.26	3.78	-17.0	60.7	0.81

Table 4.4: **Controlling Time to Collision (TTC).** The table shows the impact of different TTC Cost weights on collision scenarios. Increasing the TTC Cost weight results in an increase in collision rate, suggesting a heightened challenge for the ego vehicle in avoiding collisions.

$\mathbf{J}_{\text{adv}}$	$\mathbf{J}_{\text{reg}}$	**Partial Diffusion**	**Collision** (%) ↑	**Adv Offroad** (%) ↓	**Realism** ↓	**Collision Angle** (rad) ↑	**Collision Rel Speed** (m/s) ↑	**Collision Point** (m) ↑
✓	✓	×	23.9	**13.8**	0.58	2.22	2.99	1.62
✓	✓	✓	29.0	14.6	**0.57**	3.10	1.96	**5.44**
✓	×	×	**53.5**	23.1	0.58	**3.34**	**4.81**	2.47

Table 4.5: **Ablation study of Controllability.** We perform ablation study on each element of our proposed method.

4.6.7 Ablation Study on Controllability

We performed an ablation study to assess the influence of different guidance strategies in diffusion models on the quality of simulations. This study, detailed in Table 4.5, aimed to quantify collision diversity. We compare against our baseline approach, consisting of regularized and adversarial objectives ($J_{reg} + J_{adv}$), by predetermining the selection of adversarial agents and conducting experiments with multiple seeds (three) in our framework to evaluate collision diversity. In partial diffusion, we manipulated the trajectory proposal selection mechanism across centerlines with normal offsets of -2.0, 0.0, and 2.0.

Effectiveness of Partial Diffusion. Table 4.5 reveals that partial diffusion significantly enhances both the collision point and angle diversity in comparison to the baseline approach. This underscores the capability of partial diffusion to generate a wider array of collision scenarios through various trajectory proposals, illustrating its potential to explore diverse collision dynamics effectively.

Impact of the Regularization Term (J_{reg}). Incorporating J_{reg} results in a notable decrease in the adversarial-collision rate, highlighting the importance of the regularization term in enhancing simulation realism. Without J_{reg}, the collision rate between the adversarial agent and ego vehicle increases, but the adversarial vehicle goes offroad more often, leading to scenarios that differ significantly from realistic behavior.

4.6.8 Limitation and Failure Cases

We identified areas for improvement in SAFE-SIM (see fig. A.4). In certain cases, the adversarial agent unrealistically collides with non-adversarial agents before reaching the ego agent. Additionally, some scenarios result in collisions where the ego planner is not at fault. While understanding how the ego planner can avoid such cases is important, creating more scenarios where the ego is at fault would be beneficial.

4.7 Conclusion

In this chapter, we present a closed-loop simulation framework utilizing guided diffusion models for creating safety-critical scenarios to assess autonomous vehicle (AV) algorithms. This formulation aligns with the principles of Safety Of The Intended Functionality (SOTIF) [3], focusing on how autonomous vehicles respond to dynamic scenarios like aggressive driving, underscoring our dedication to safety across diverse and unforeseeable conditions. Our framework introduces innovative guidance objectives, specifically tailored for controllable, stable, long-term safety-critical simulations. A key aspect of our method lies in its ability to vary the types of adversarial behavior within collision scenarios. By integrating adversarial objectives and partial diffusion into the diffusion model's architecture, we achieve detailed control over the spectrum of adversarial actions. This versatility enables our framework to produce a broader range of realistic and manageable scenarios, setting a new standard in adversarial scenario generation beyond the limitations of existing approaches.

Chapter 5

LangTraj: Diffusion Model and Dataset for Language-Conditioned Trajectory Simulation

This chapter is based on the paper "LangTraj: Diffusion Model and Dataset for Language-Conditioned Trajectory Simulation" [73], written in collaboration with Wei Zhan, Masayoshi Tomizuka, Manmohan Chandraker, and Francesco Pittaluga.

5.1 Introduction

Building on the diffusion-based framework introduced in chapter 4, which enables the generation of safety-critical scenarios through guided objectives, this chapter further extends controllable simulation by introducing natural language as a flexible interface for counterfactual scenario generation.

Traditionally, structured testing of AVs has relied on manually designed scenarios to simulate failure cases or counterfactual situations, such as collisions or typical interactive behaviors. While effective for targeted testing, this approach is inherently limited in scalability. Recent advances in diffusion-based generative models demonstrate strong capabilities in simulating complex distributions with flexible controllability [21, 55], though this often depends on domain-specific heuristic guidance functions constructed based on human knowledge *after training*. Our key insight is to **leverage language conditioning during training** to directly learn semantics from the data distribution, enabling users—including non-experts—to generate counterfactual scenarios with ease. By conditioning on natural language, we can significantly expand the range of possible scenarios, enabling more diverse driving behaviors and improving the model's capability as the data scale. Note that direct conditioning on language is orthogonal to guidance functions, allowing us to combine the strengths of both human knowledge and data-driven insights for more diverse simulations.

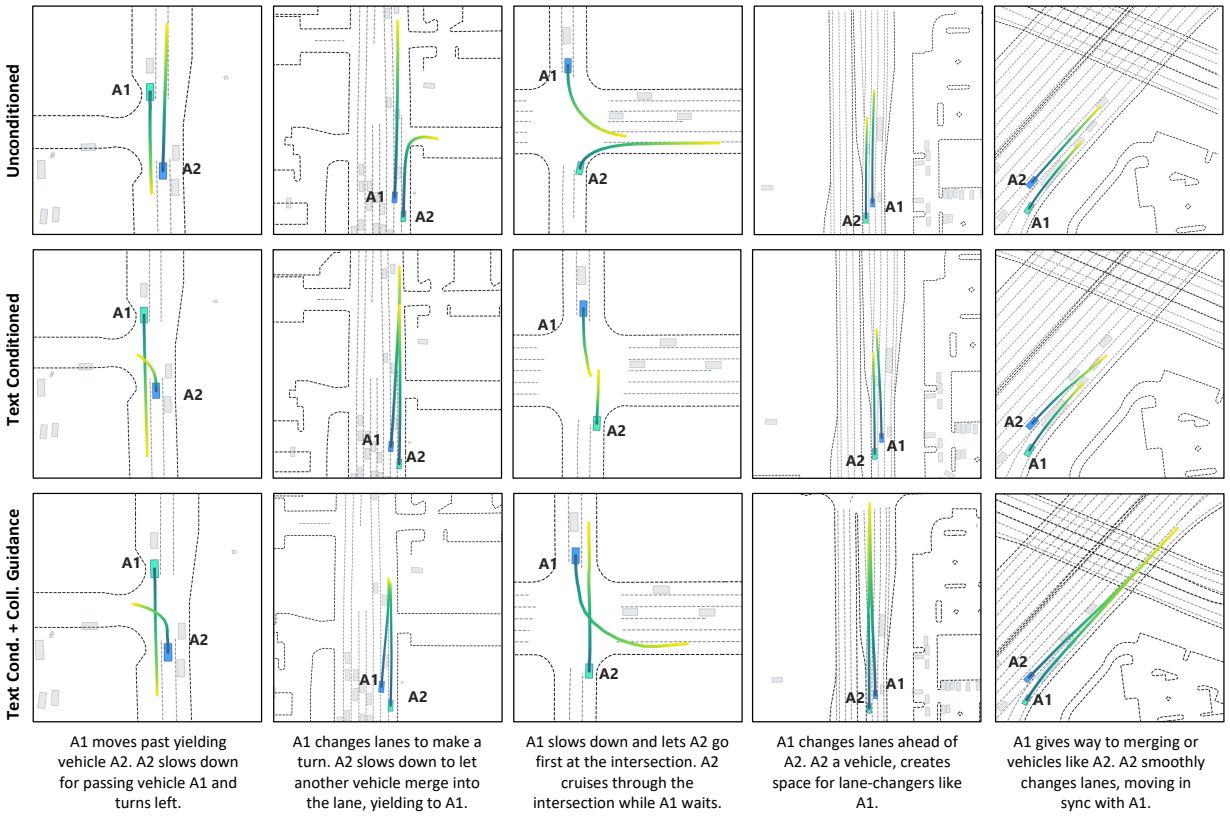

Figure 5.1: **Unconditioned (top), Text-Conditioned (mid), and Safety-Critical + Text-Conditioned (bot) Simulations by LangTraj.** Adversarial collision guidance is applied in conjunction with text conditioning to generate the safety-critical scenarios shown in the bottom row, where the dark blue car serves as the adversarial agent. Language annotations are from the test set of InterDrive.

We present LangTraj[1], a language-controlled diffusion model that generates realistic and controllable trajectories for AV simulation by conditioning on natural language prompts. This approach enables LangTraj to respond to diverse instructions such as "yield to the right" or "merge left," capturing complex interactive behaviors for counterfactual testing across varied driving scenarios.

Achieving this functionality requires overcoming two key challenges: developing a model that can effectively condition on diverse user inputs during closed-loop simulation and acquiring high-quality data with natural language labels. To address these challenges, we first introduce a scene-diffusion model, which jointly models multi-agent behaviors while conditioning on language inputs, enabling flexible and scalable AV testing. We propose a novel *closed-loop training strategy* for diffusion models to improve closed-loop realism and

[1] `https://langtraj.github.io/`

reduce error accumulation during iterative rollouts. Contrary to prior diffusion model works [21, 26, 55] that adopt inference techniques such as constraint and guidance methods to steer predictions, which can slow inference and require careful balancing of multiple objectives, we introduce a closed-loop training strategy that explicitly enhances model stability and realism during closed-loop simulation. Additionally, guidance methods can still be applied post-training to refine behavior further if needed.

To enable language-conditioned simulation, we introduce InterDrive, a comprehensive dataset with 150k human-labeled annotations, specifically focusing on *interactive* agent-agent behaviors (e.g., merging, yielding, and passing) with rich language annotations. In addition, we include heuristic-based labels for single-agent behaviors such as directional intent and lane changes. This dataset ensures high-quality annotations for interaction modeling, providing the necessary diversity and richness for training models that capture both interactive and individualistic driving behaviors.

Our contributions are as follows:

- **LangTraj: Language-Conditioned Diffusion for Interactive Simulation.** We introduce LangTraj, the first diffusion-based trajectory generation model that directly conditions on natural language inputs for interactive simulation.
- **Novel Closed-Loop Training Strategy for Diffusion Models.** We propose a novel closed-loop training strategy explicitly tailored for diffusion models to enhance stability and realism during closed-loop simulation.
- **InterDrive Dataset.** We present InterDrive, a new dataset of 150k human-labeled interactive traffic scenarios, supplemented with heuristically labeled single-agent behaviors, enabling scalable training of language-conditioned simulation models.

5.2 Related Work

5.2.1 Traffic Simulation and Controllable Diffusion

Controllable diffusion models have advanced traffic simulation [21, 22, 23, 26, 55]. Diffusion-ES [74] combines evolutionary search with diffusion models. CTG [21] introduces test-time guidance, while SAFE-SIM [55] models adversarial behaviors. CTG++ [22] enhances controllability with GPT-4 [75]-generated guidance, and SceneDiffuser [26] improves inference efficiency. However, these approaches rely on test-time guidance or heuristic control signals, and do not learn semantic behavior control directly from data via language conditioning during training.

5.2.2 Language in Autonomous Driving

Large Language Models (LLMs) have enabled language integration into autonomous driving. Early datasets [76, 77] focus on spatial annotations, while DriveVLM's Graph VQA explores

vision-language reasoning. WOMD-Reasoning [78] introduces 409K QAs on traffic-rule interactions, and ProSim-Instruct-520k [79] pairs 10M Llama3-70B-generated prompts with 520K scenarios from the Waymo Open Dataset. In contrast, InterDrive is directly constructed from the interactive subset of the Waymo Open Dataset, ensuring a targeted selection of interactive scenarios. Additionally, our annotations are collected from human experts rather than LLMs, focusing specifically on high-quality interactive behavior labeling.

Language-conditioned simulation methods vary. LCTGen [80] generates trajectories in an *open-loop* manner based on discrete attributes, while ProSim [79] uses an autoregressive framework for goal-driven simulation. We adopt a diffusion-based approach for greater flexibility beyond the training distribution, enabling more adaptive and interactive simulations, as discussed in section 5.6.5.

5.3 InterDrive Dataset

Natural language conditioning for traffic simulation relies heavily on the scale and quality of the underlying data. To address this, we introduce InterDrive, a central contribution of this work, designed to capture nuanced agent-agent interactions in real-world driving contexts. Our dataset is constructed through an extensive human-labeling effort on the Waymo Motion [13] and NuPlan [59] datasets, with additional single-agent behavioral labels generated via heuristic annotations.

Unlike previous datasets such as ProSimInstruct [79], which provide general trajectory labels, we focus specifically on *interactive agents*, ensuring that agent-agent interactions—such as merging, yielding, and passing—are explicitly labeled. This emphasis enables more realistic and flexible training of language-conditioned diffusion models for diverse traffic scenarios.

5.3.1 Dataset Composition

The InterDrive dataset includes annotations from both the Waymo [13] and NuPlan [59] datasets, covering a diverse range of environments, agent types, and interaction scenarios. For Waymo, 125k scenes were annotated with human-generated interaction labels and 405k scenes annotated with heuristic labels. For nuPlan, 12k scenes were annotated with interaction labels and 150k scenes annotated with heuristic labels, each lasting 15 seconds.

5.3.2 Human-Labeling Process

To ensure high-quality and efficient interaction annotations, our labeling protocol was carefully designed to leverage human expertise in defining complex driving behaviors. Labelers were provided with top-down video representations of each scene, enabling them to accurately observe and annotate agent interactions. Using these visualizations, we developed a scalable and efficient pipeline for annotating complex interactions through a structured multi-choice question framework. The process consists of the following steps:

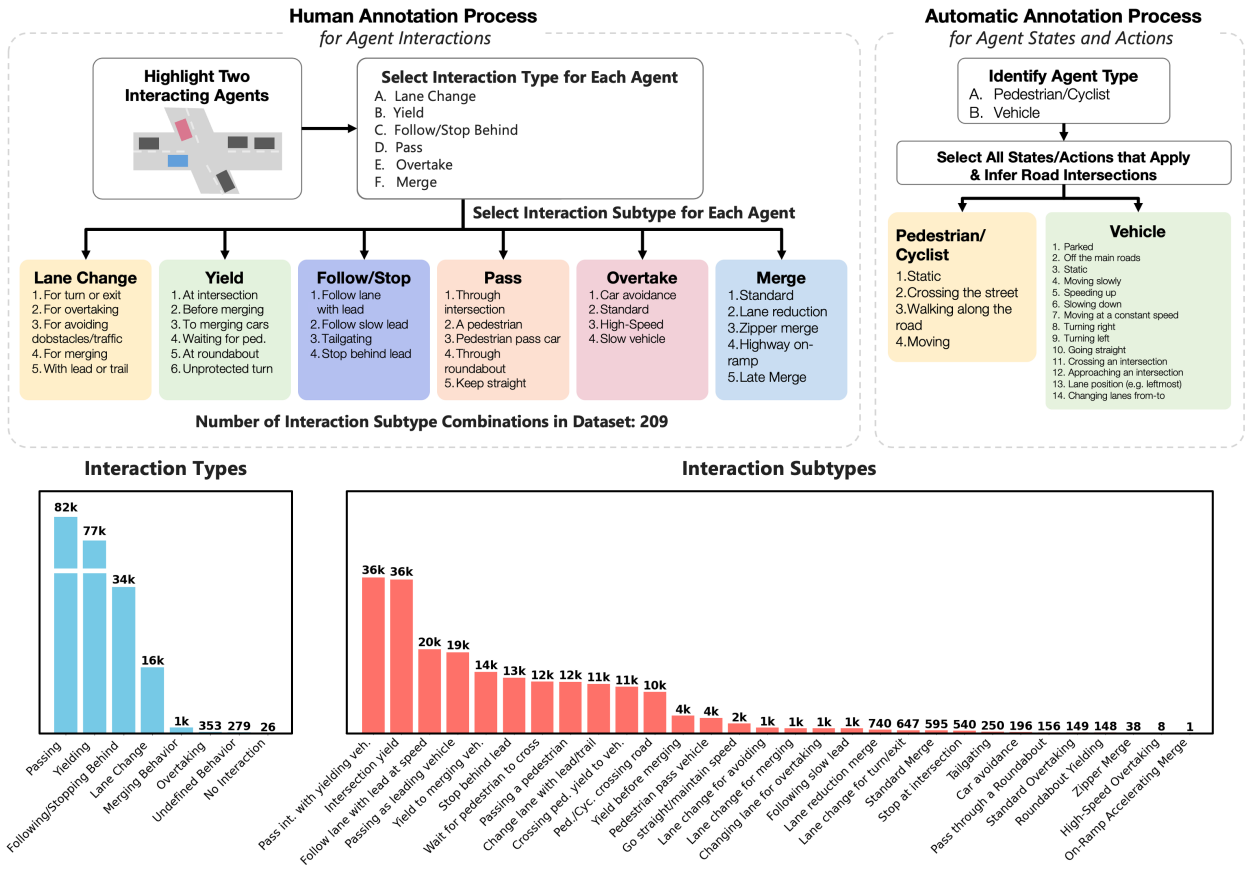

Figure 5.2: **Overview of InterDrive dataset.** InterDrive captures nuanced agent-agent interactions in real-world driving contexts. It includes human-labeled traffic interaction annotations from Waymo Motion and NuPlan datasets, along with single-agent behavioral labels generated through heuristic annotations, offering a comprehensive view of agent actions and interactions in diverse traffic scenarios. The top part of the figure shows the annotation processes for the human and heuristic annotations. The bottom part of the figure shows the counts of each interaction and interaction subtypes in InterDrive.

1. *Identify Interacting Agents:* For Waymo, we used the interacting agent IDs provided by the dataset. For nuPlan, we asked labelers to manually identify one agent within each scene that exhibits an interaction with the ego vehicle.

2. *Classify Interaction Type:* Labelers select the primary behavior of each interaction agent, choosing from six interaction types: (1) Lane Changing, (2) Following/Stopping Behind, (3) Yielding, (4) Passing, (5) Overtaking, and (6) Merging.

3. *Classify Interaction Subtypes:* For each chosen interaction type, labelers provide a more granular interaction subtype, capturing specifics like "Changing lane for overtaking," and "Intersection yielding". A full list of interaction subtypes is provided in appendix B.5 and in fig. 5.2.

As shown in fig. 5.2, most real-world driving behaviors can be effectively captured by our carefully designed interaction types, providing a comprehensive framework for annotating complex agent interactions.

5.3.3 Heuristic Annotation Process

In addition to interaction labels, InterDrive contains labels for single-agent states/actions, which were generated automatically using a set of carefully calibrated heuristics. For the pedestrians and cyclists labels, we selected all that applied from the following list: static, crossing the street, walking along the road, and moving. For the vehicle labels, we selected all that apply from the following list: parked, off the main roads, static, moving slowly, speeding up, slowing down, moving at a constant speed, turning right, turning left, going straight, crossing an intersection, approaching an intersection, lane position (e.g. rightmost), changing lanes from-to (e.g. middle-to-rightmost). This enabled comprehensive behavioral modeling in scenarios with both explicit and implicit agent interactions.

Our human-annotated dataset provides rich annotations for interactive simulation and behavior studies, enabling in-depth analysis of agent interactions and unlocking future possibilities for language-to-simulation research.

5.4 Problem Formulation

In traffic simulation, we model behaviors of N agents through a centralized function g_θ, enabling realistic and controllable agent actions via language-conditioned commands $\mathbf{e}_{\text{lang}}$. The joint agent state at each timestep t is $\mathbf{s}_t = [\mathbf{s}_t^1, \ldots, \mathbf{s}_t^N]$ with actions $\mathbf{a}_t = [\mathbf{a}_t^1, \ldots, \mathbf{a}_t^N]$, transitioning via f based on unicycle dynamics: $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$.

Each agent shares context $\mathbf{c}_t$, including HD map I and historical states $\mathbf{S}_{t-T_{\text{hist}}:t}$. The function g_θ generates future trajectories $\{\mathbf{s}_{t:t+T}^i\}_{i=1}^N$ based on $(\mathbf{c}_t, \mathbf{e}_{\text{lang}})$, trained on real-world data to ensure realism and flexibility to user-defined scenarios.

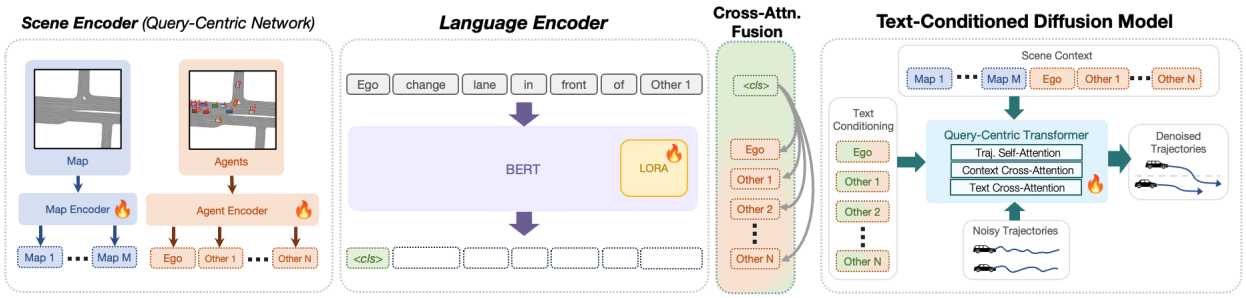

Figure 5.3: **Overview of LangTraj.** We introduce LangTraj, a novel language-controlled diffusion-based model for trajectory simulation that incorporates HD maps, agent histories, and text descriptions, enabling behaviorally nuanced trajectory generation.

We use diffusion models to produce text-conditioned trajectories, reversing a forward noising process from real trajectories $\tau_0 \sim q(\tau_0)$ into noisy sequences $(\tau_1, \ldots, \tau_K)$ via Gaussian noise:

$$q(\tau_{1:K}|\tau_0) := \prod_{k=1}^{K} q(\tau_k|\tau_{k-1}),$$

$$q(\tau_k|\tau_{k-1}) := \mathcal{N}(\tau_k; \sqrt{1-\beta_k}\tau_{k-1}, \beta_k \mathbf{I}).$$

The model learns to denoise τ_K back to τ_0, integrating text encoding $\mathbf{e}_{\text{text}}$ to influence mean predictions:

$$p_\theta(\tau_{k-1}|\tau_k, \mathbf{c}, \mathbf{e}_{\text{lang}}) := \mathcal{N}(\tau_{k-1}; \mu_\theta(\tau_k, k, \mathbf{c}, \mathbf{e}_{\text{lang}}), \Sigma_k).$$

This enables generation of scene- and text-aligned future trajectories.

5.5 LangTraj

We introduce LangTraj, a scene-diffusion model designed to capture the joint distribution of interactive behaviors among all agents within multi-agent environments. To achieve this, LangTraj comprises two key components: 1) an encoder that effectively represents the scene context, including map features and historical agent behaviors, and 2) a denoiser capable of jointly predicting future agent behaviors while providing flexibility and user-driven control via natural language inputs. LangTraj supports trajectory generation conditioned on textual prompts and diffusion guidance, ensuring both flexibility and precise control. Additionally, we propose a novel algorithm for closed-loop training of diffusion models, further enhancing the model's performance and applicability in closed-loop simulation settings.

5.5.1 Architecture

Scene Encoder Inspired by [81, 82], we adopt a query-centric approach combined with Graph Neural Networks (GNNs) to model spatiotemporal relationships in the scene. A key component of this approach is the attention mechanism, which encodes relative spatial and temporal information, capturing interactions and dependencies between scene elements. The encoder operates in a scene-independent local coordinate system. Each scene element extracts features within its own local reference frame, independent of global coordinates or the ego vehicle's position. This formulation allows for symmetric encoding across agents without being affected by variations in absolute positioning. Through this process, the encoder produces an embedding $\mathbf{z}^i_{\text{enc}} = E_{enc}(I, \mathbf{S}_{t-T_{\text{hist}}:t})$ for each agent i.

Language Encoder The Language Encoder module extracts agent-specific embeddings from natural language inputs and integrates them with the scene's spatiotemporal context (see fig. 5.3). This module serves two key functions: 1) providing explicit agent-specific conditioning and 2) encoding spatiotemporal data to capture agent interactions.

To achieve agent-specific conditioning, we first process the input sentence through a language model, where agent roles are explicitly *rephrased* for direct conditioning. Specifically, for each agent, its role in the sentence is labeled as the "target agent," while other agents are designated as "other agent1," "other agent2," and so forth. This rephrasing ensures clear distinctions in agent roles within the sentence. After processing through the language model, we obtain a sentence embedding $\mathbf{e}_{\text{lang}}$ that captures the overall context.

We define the language encoder function E_L to integrate the language and scene context. Specifically, E_L combines the language-conditioned sentence embedding $\mathbf{e}_{\text{lang}}$ with each agent's context embedding $\mathbf{z}^i_{\text{enc}}$ from the Scene Encoder:

$$\mathbf{z}^i_{\text{lang}} = E_L(\mathbf{e}_{\text{lang}}, \mathbf{z}^i_{\text{enc}}),$$

where $\mathbf{z}^i_{\text{lang}}$ represents the final language-conditioned embedding for each agent i. This embedding incorporates both the spatial relationships within the scene and the agent-specific conditioning derived from the language input.

Our design is flexible regarding the choice of language model; following the practice of [83], we use DistilBERT [84], a smaller language encoder, to extract the <cls> token embedding as a summary of the sentence, reducing computation cost. Empirically, we find this sufficient for language conditioning. In practice, we train the language encoder end-to-end with the diffusion model using LoRA (Low-Rank Adaptation), enabling parameter-efficient fine-tuning.

Denoiser The denoiser consists of multiple stacked transformer blocks, each incorporating different types of attention to model complex agent behaviors and interactions. Specifically, the denoiser employs: 1) Query-centric attention across agents' future trajectories to capture inter-agent relationships, allowing each agent to attend to other agents' future actions; 2) Agent-Context Cross-Attention, where each agent attends to its context embedding

Algorithm 1 Closed-Loop Training of Diffusion Models

Require: Pretrained diffusion model θ_{init}, dataset $\mathcal{D} = \{s_{0:T}^{(i)}\}_{i=1}^{N}$, distance metric $D(\cdot,\cdot)$, planning horizon T_{replan}, number of samples M, noise scale γ, denoising steps K
Ensure: Optimized model θ
Initialize $\theta \leftarrow \theta_{\text{init}}$
for each trajectory $s_{0:T} \sim \mathcal{D}$ **do**
  for $t = 0$ **to** $T - T_{\text{replan}}$ **step** T_{replan} **do**
    Extract target sequence: $s_{\text{target}} = s_{t:t+T_{\text{replan}}}$
    Sample noise level: $\tau \sim \mathcal{U}(1, K\gamma)$
    Add noise: $s_\tau = \sqrt{\alpha_\tau} s_{\text{target}} + \sqrt{1-\alpha_\tau}\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$
    Generate M denoised candidates $\hat{s}^{(1:M)}$
    Select $\hat{s}^{(m^*)}$ via $\arg\min_m D(\hat{s}^{(m)}, s_{\text{target}})$
    Execute $\hat{s}^{(m^*)}$, update state
  end for
  Compute loss $\mathcal{L} = \|\hat{s}_{0:T} - s_{0:T}\|^2$, update θ
end for
return θ

$\mathbf{z}_{\text{enc}}^i$ generated by the Scene Encoder, ensuring that the agent's behavior aligns with the spatiotemporal context of the environment; and 3) Text-Cross Attention, where if a language description is provided, each agent also attends to the language-conditioned embedding $\mathbf{z}_{\text{lang}}^i$, incorporating conditioning from user-defined language inputs.

5.5.2 Closed-loop Training

Typically, trajectory diffusion models are trained in an open-loop fashion [22, 23, 26, 55]. However, this mismatch between training and inference can lead to distribution shifts, where the model encounters compounding errors due to deviations from the expected trajectory distribution. A common approach to mitigate this issue during inference is to apply guidance methods or hard constraints to steer model predictions toward physically feasible behaviors. However, these techniques slow down inference speed and require careful balancing of multiple objectives, limiting adaptability in real-time simulations.

In contrast, closed-loop training is a widely used technique for mitigating compounding errors in closed-loop rollouts, as seen in autoregressive models [4, 79], where past predictions are recursively used as input to better reflect real-world deployment scenarios. However, applying closed-loop training to diffusion models presents a unique challenge: the iterative denoising process involves a double for-loop, making it computationally expensive to incorporate self-generated states into training.

To address this, we propose a closed-loop training tailored for diffusion models, inspired by [85, 86], which better aligns the training and inference distributions without requiring multiple denoising steps during training. Instead of training with purely ground-truth trajectories, we

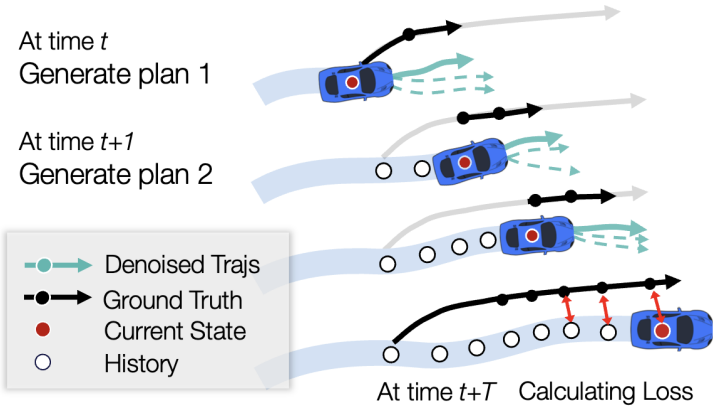

Figure 5.4: **Illustration of Closed-loop Training of diffusion models.** The figure demonstrates the procedure of training diffusion models in a closed-loop setting. First, the model generates multiple denoised trajectory candidates. The closest candidate to the ground truth is selected and then executed, enabling the model to experience its own distribution during training.

integrate model-generated samples into training to allow the model to experience its own predictions.

As shown in fig. 5.4, our method modifies the standard diffusion training process by incorporating model-generated samples into training. At each training step, we apply forward diffusion to perturb the ground-truth trajectory segment, followed by a one-step reverse diffusion to generate multiple candidate trajectories. We then select the best candidate using a predefined distance function and execute the selected trajectory before repeating the process. The final loss is computed between the executed and ground-truth states in the global coordinate space, allowing the model to learn how to recover from accumulated errors. For a detailed breakdown of this procedure, refer to Algorithm 1. In practice, we employ a teacher-forcing strategy, where a subset of agents follows ground-truth states during training. This stabilizes learning and improves adherence to map-related metrics, as shown in table 5.6.

5.5.3 Controllable Behavior via Diffusion Guidance

To enhance control over trajectory generation, we can incorporate two types of guidance: *classifier-based guidance*, which optimizes differentiable objectives, and *classifier-free guidance*, which blends conditional and unconditional predictions.

For classifier-based guidance, we modify the predicted mean at each denoising step using the gradient of an objective function $J(\tau)$, steering trajectories toward desired behaviors while maintaining realism. We adopt reconstruction guidance (clean guidance) from [58, 69]

to improve stability:

$$\tilde{\tau}_0 = \hat{\tau}_0 - \alpha \sum_k \nabla_{\tau_k} J(\hat{\tau}_0). \tag{5.1}$$

The details of the adopted guidance functions can be found in appendix B.4. For classifier-free guidance, we interpolate between conditional and unconditional predictions from g:

$$\hat{\tau}_0 = (1 + w) \cdot g(\mathbf{e}_{\text{lang}}, \mathbf{z}_{\text{enc}}) - w \cdot g(\emptyset, \mathbf{z}_{\text{enc}}) \tag{5.2}$$

where $g(\mathbf{e}_{\text{lang}}, \mathbf{z}_{\text{enc}})$ includes language input $\mathbf{e}_{\text{lang}}$, while $g(\emptyset, \mathbf{z}_{\text{enc}})$ omits it. The guidance weight w controls the influence of language conditioning, allowing more flexible behavior synthesis without relying on predefined objectives.

By integrating these two diffusion-based approaches, we enable more flexible and adaptable simulations, which are crucial for testing autonomous vehicles in diverse and challenging scenarios.

5.6 Experimental Results

5.6.1 Preliminaries

We validate our framework through experiments on real-world driving data from the Waymo Open Motion Dataset (WOMD) [13]. We train LangTraj on the training splits of InterDrive and ProSim-Instruct-520k, described in appendix B.1, and evaluate its ability to generate behaviors that are both *realistic* and *controllable*.

Metrics. For realism (denoted "Meta" in the tables), we follow the Waymo Open Sim Agent Challenge (WOSAC), which evaluates the distributional realism of generated trajectories [6]. The challenge assesses how well joint simulation rollouts recover held-out ground truth behavior across multiple aspects, including kinematics, agent interactions, and map adherence. Further details can be found in appendix B.1.5. To evaluate **controllability**, we condition the model on ground-truth future behavior descriptions and measure the improvement in minADE compared to the unconditional model. This comparison quantifies the model's ability to align generated behaviors with user-specified input when provided with behavior descriptions.

5.6.2 Evaluation of InterDrive

We train separate instantiations of LangTraj on the training sets of InterDrive (InterDrive) and ProSim-Instruct-520k, performing closed-loop evaluations with and without text conditioning. As shown in table 5.1, language conditioning on InterDrive yields a 13.6% reduction in minADE while maintaining a meta-realism score of 0.72, indicating that our language annotations provide useful supervisory signals. A similar improvement is observed on ProSim-Instruct-520k (12.8% minADE reduction). Notably, InterDrive focuses on

Datset	Text	Meta ↑	Kinematic ↑	Interactive ↑	Map ↑	mADE ↓
InterDrive	×	0.72	0.41	0.80	0.80	2.65
	✓	0.72	0.42	0.80	0.79	2.29
ProSim-Instruct	×	0.72	0.42	0.79	0.79	2.89
	✓	0.72	0.43	0.80	0.78	2.52

Table 5.1: **Evaluation of InterDrive:** We train LangTraj separately on the InterDrive and ProSim-Instruct-520k training sets and evaluate its performance on their respective test sets.

Text Conditioning	Meta ↑	Kinematic ↑	Interactive ↑	Map ↑	mADE ↓
None	0.72	0.41	0.80	0.80	2.65
Direct Condition (Ours)	0.72	0.42	0.80	0.79	2.29
LLM-Based Guidance (CTG++ Style)	0.70	0.42	0.75	0.80	2.70

Table 5.2: **Text Conditioning Evaluation.** We compare our proposed direct text conditioning method to the LLM-based guidance method proposed by CTG++ [22].

interactive agent-agent behaviors, whereas ProSim-Instruct primarily consists of single-agent maneuvers.

5.6.3 LLM-based Guidance vs. Direct Conditioning

In table 5.2, we compare our proposed direct language conditioning method with the LLM-based guidance approach from CTG++ [22], which uses GPT to generate differentiable code-based guidance functions from language prompts (see appendix B.1.3 for details). On InterDrive scenarios, we find that LLM-based guidance leads to worse alignment with the text prompt—even underperforming the no-text baseline. In contrast, direct conditioning produces behaviors more aligned with the ground truth (13.6% reduction) while preserving simulation realism. Direct conditioning also outperforms LLM-based guidance in counterfactual settings, as discussed in section 5.6.5. While LLM-based guidance is effective for enforcing constraints like collision avoidance and on-road driving [22], it still struggles with interaction-level behavior descriptions, underscoring the benefit of direct language grounding.

5.6.4 Evaluation of LangTraj

To enable a fair comparison with ProSim, we train LangTraj on the ProSim-Instruct-520k training set and evaluate its performance against the publicly released ProSim model, which is trained on the same dataset. As shown in table 5.3, both methods achieve comparable performance in simulation realism and gain in terms of minADE. However, LangTraj employs diffusion-based modeling, which unlocks inference-time guidance and controlled

Method	Text	Meta ↑	Kinematic ↑	Interactive ↑	Map ↑	mADE ↓
LangTraj	×	0.72	0.42	0.79	0.79	2.89
	✓	0.72	0.43	0.80	0.78	2.52
ProSim	×	0.69	0.42	0.73	0.80	2.73
	✓	0.69	0.42	0.73	0.81	2.35

Table 5.3: **Evaluation of LangTraj.** Closed-loop evaluation on the ProSim-Instruct-520k test set, with and without text conditioning. We compare the performance of our method (LangTraj) against the publicly released ProSim model, both trained on the ProSim-Instruct-520k training set, ensuring a fair evaluation.

Method	Meta ↑	Kinematic ↑	Interactive ↑	Map ↑
UniMM [85]	0.769	0.491	0.811	0.874
SMART-tiny-CLSFT [86]	0.762	0.458	0.811	0.872
VBD [87]	0.720	0.417	0.814	0.776
LangTraj	0.719	0.426	0.795	0.789
ProSim [79]	0.718	0.401	0.778	0.822
SceneDiffuser [26]	0.703	0.430	0.776	0.768
SceneDMF [88]	0.628	0.371	0.683	0.703

Blue: Diffusion-Based Methods

Table 5.4: **Results on WOSAC Test Set.** LangTraj performs competitively among the best diffusion-based approaches, such as VBD [87] and SceneDiffuser [26].

sampling, offering greater flexibility beyond the training distribution that autoregressive-based methods like ProSim do not have. We discuss these advantages in section 5.6.5.

Beyond instruction-following tasks, we evaluate LangTraj on the Waymo Sim Agents Challenge Benchmark (unconditioned simulation), with results presented in table 5.4. Compared to state-of-the-art diffusion models, LangTraj performs competitively among the best diffusion-based approaches, such as VBD [87] and SceneDiffuser [26].

5.6.5 Text-Conditioned Safety-Critical Simulation

We demonstrate LangTraj's ability to extend beyond the training distribution by generating safety-critical scenarios conditioned on text inputs using guided sampling, as illustrated in table 5.5 and fig. 5.1. Given the focus on collision-prone situations, standard interactive metrics are less informative; thus, we instead evaluate simulation quality using kinematics and map adherence metrics. Our findings indicate that LangTraj successfully generates realistic traffic scenarios as well as targeted, rare, safety-critical events, achieving collision rates as high as 40%. Compared to LLM-based guidance, LangTraj achieves a 10% higher collision

		COLLISION			
Text Conditioning	**Guidance**	**Rate ↑**	**Kinematic ↑**	**Map ↑**	**mADE ↓**
None	×	0.04	0.42	0.81	3.04
None	✓	0.41	0.39	0.70	4.93
Direct Condition (Ours)	✓	**0.43**	**0.41**	**0.74**	**4.43**
LLM-based Guidance (CTG++ Style)	✓	0.33	0.37	0.72	4.67

Table 5.5: **Text-Conditioned Safety-Critical Simulation with LangTraj.** We demonstrate LangTraj's ability to extend beyond the training distribution by generating safety-critical scenarios conditioned on text inputs using guided sampling.

Setting	**Meta ↑**	**Kinematic ↑**	**Interactive ↑**	**Map ↑**
Open-loop (K=100)	0.68	0.42	0.77	0.73
Open-loop (K=5)	0.68	0.41	0.78	0.72
Closed-loop	0.69	0.38	0.78	0.75
Closed-loop w/ Teacher	0.70	0.39	0.78	0.79

Table 5.6: **Ablation Study on Closed-Loop Training.** Study conducted on validation set of WOSAC challenge.

rate and outperforms CTG++ across all metrics in safety-critical situations. Additionally, integrating direct text conditioning enhances map adherence even under collision-focused guidance, highlighting a promising approach that combines explicit guidance with textual inputs to improve scenario diversity and simulation control. Qualitative examples are available on the project website.[1]

5.6.6 Closed-loop Training

We analyze the effects of denoising steps, closed-loop training, and teacher forcing in table 5.6. First, we find that reducing denoising steps from $K = 100$ to $K = 5$ maintains realism metrics, suggesting that trajectory simulation can be potentially effectively learned with fewer denoising steps, reducing computational cost. Next, introducing closed-loop training with teacher forcing improves both map adherence ($0.72 \rightarrow 0.79$) and meta realism ($0.68 \rightarrow 0.70$). Note that we also observe vehicles tend to slow down and drive off of the road when trained in a closed-loop manner without teacher forcing, despite achieving similar realism scores. This suggests that having a portion of agents follow the ground-truth trajectories helps stabilize training, preventing the model from drifting into unrealistic behaviors.

5.7 Conclusion

LangTraj advances AV simulation by leveraging language-conditioned diffusion models to generate diverse, behaviorally rich scenarios. Unlike prior works, it supports direct language conditioning for intuitive behavior specification and guidance-based control for counterfactual and targeted scenario generation. This flexibility makes LangTraj well-suited for scalable, controllable AV simulation, enabling safety-critical testing and interactive scenario design. This significantly expands the space of scenarios that can be generated and lowers the barrier to controllable simulation.

A key component of our approach is InterDrive, which focuses on interactive agents, providing rich human-labeled and heuristic annotations that enhance realism and diversity. By prioritizing agent-agent interactions, InterDrive strengthens training for language-conditioned models. Empirical results on the Waymo Motion Dataset show that LangTraj generates realistic, language-aligned trajectories while preserving simulation realism. In summary, LangTraj demonstrates the potential of language-conditioned diffusion models for AV simulation by combining natural language with guidance-based control.

This chapter reveals a key insight: controllability can be learned through language, where data, model design, and training must be jointly considered to achieve flexible and scalable simulation. Language controllability opens the possibility of automatically discovering interesting and diverse scenarios while maintaining interpretability through natural language descriptions. Furthermore, such language-conditioned simulation provides a bridge between high-level reasoning and embodied control, connecting simulation with broader developments in embodied AI and robotics.

Part III

Scalable Closed-Loop Multi-Agent Simulation

Chapter 6

SPACeR: Self-Play Anchoring with Centralized Reference Models

This chapter is based on the paper "SPACeR: Self-Play Anchoring with Centralized Reference Models" [89], written in collaboration with Akshay Rangesh, Kevin Joseph, Matthew Strong, Masayoshi Tomizuka, Yihan Hu, and Wei Zhan.

6.1 Introduction

Previous chapters have established advances in controllability and realism, yet scalability remains a critical challenge, particularly for efficient large-scale closed-loop simulation across many agents and long horizons. Modern generative models, while expressive, are often computationally expensive and difficult to deploy in such settings. In this chapter, we focus on self-play reinforcement learning (RL) as a framework for improving efficiency and scalability, while preserving realistic and interactive behaviors.

As discussed in chapter 1, simulation policies aim to capture realism, controllability, and diversity. In closed-loop settings, realism is particularly critical, as it requires agents to *react* appropriately to other behaviors. Beyond realism, *scalability* becomes an important practical consideration, requiring efficient deployment in large-scale simulation across many agents and long horizons. Broadly, there are two paradigms for constructing such policies: imitation learning and self-play reinforcement learning (RL).

Imitation learning starts from expert demonstrations, with recent advances including tokenized models [18, 19, 86] and diffusion models [22, 26, 90] to generate realistic multi-agent behaviors. The key advantage of imitation learning is that it learns directly from the human data distribution. However, such approaches struggle in reactive settings [91] and often require computationally heavy models such as Transformer-based architectures that limit scalability, as we demonstrate in section 6.4.3.

On the other hand, self-play RL has become popular [31, 36] for building driving policies. In self-play, agents learn policies by repeatedly playing against one another in closed-loop

interaction. However, successful training often requires extensive heuristics and careful reward shaping [92], and the resulting policies can still diverge from human norms, leading to unrealistic and non-human like behaviors.

To address both challenges, we introduce SPACER[1], which leverages a pretrained tokenized model [19] to provide a reference policy distribution and a likelihood signal over the outputs of self-play policies (fig. 6.1). This serves as an on-policy reward provider, shaping learning toward human-like behavior while preserving the scalability of self-play RL. Specifically, we align the self-play policy's action space with that of the tokenized model, enabling tractable likelihood estimation and KL-based distributional alignment. This design eliminates the need for ground-truth logged trajectories and instead provides probabilistic estimates that directly guide learning.

We validate our approach on the Waymo Sim Agents Challenge (WOSAC) [93], where SPACER significantly improves realism and human-likeness compared to prior self-play RL methods. In addition to benchmark performance, we perform closed-loop policy evaluation across diverse planners, using the reference tokenized model as a baseline. Our experiments show that Human-Like Self-Play policies are more reactive and avoid the false-positive collisions often seen in imitation-based approaches, yielding more realistic and reliable estimates for planner evaluation.

Beyond performance, the resulting policies are remarkably lightweight: decentralized MLPs with only ∼65k parameters that run over 10× faster and are up to 50× smaller than most tokenized models. This efficiency enables scalable, real-time multi-agent simulation at unprecedented scale, while maintaining the realism necessary for AV testing and deployment.

6.2 Related Work

6.2.1 Self-Play RL for autonomous driving

Self-play reinforcement learning has been studied extensively in multi-agent reinforcement learning (MARL) as a way for agents to learn by interacting with copies of themselves [94, 95, 96] and has recently been applied to autonomous driving [31, 35, 36]. One representative work, GIGAFlow [36] demonstrates that large-scale self-play can produce robust autonomous driving policies, showcasing the scalability and effectiveness of self-play for autonomy. Nevertheless, one of the main challenges for self-play methods in real-world deployment is the divergence from human behaviors. Policies that maximize rewards may behave unpredictably, and be unable to coordinate with humans safely. While Human-Regularized PPO [33] introduces a small amount of demonstration data as a regularizer to bias policies toward human-likeness, our approach leverages a large-scale tokenized model to provide a human-like reward signal during multi-agent learning, making it the first to demonstrate competitive performance with state-of-the-art imitation learning policies while preserving the scalability of self-play.

[1] `https://spacer-ai.github.io/`

6.2.2 Imitation-Learning Based Traffic simulation

Recently, data-driven methods like imitation learning have gained popularity due to their capacity to learn from an expert data-distribution with minimal human effort. Early works have explored generative imitation-learning approaches for modeling human traffic behaviors [97, 98, 99]. Recent works can mainly be categorized as diffusion models and tokenized models. Diffusion Models such as [22, 26, 55, 73] offer flexibility and controllability to potentially generate long-tail driving scenarios. On the other hand, tokenized models such as SMART [19] or CAT-K [86] have become more popular due to their high realism and capacity to simulate realistic multi-agent behavior. However, both approaches are computationally expensive at inference time: diffusion models require multiple denoising steps, and tokenized models generate actions sequentially. This limits scalability in large-scale closed-loop simulation. In this work, we instead leverage pretrained imitation-learning models to guide self-play RL toward human-like behaviors. There are several works that focus on RL fine-tuning of pretrained imitation-learning models [100, 101, 102]: for example, using Group Relative Proximal Optimization to improve realism and controllability [101], leveraging human feedback for post-training alignment [102], or exploiting implicit preferences from pre-training demonstrations to avoid costly human annotations [103]. In contrast to this pretrain-then-finetune paradigm, we take an RL-first approach, where self-play serves as the foundation and imitation-learning models are incorporated as a reward provider.

6.3 Human-like self-play

We aim to learn a human-like driving policy $\pi_\theta(a|o)$ whose behaviors match the real human driving distribution in closed-loop. The desired policy should reach goals, avoid collisions, and stay on-road, while producing predictable and realistic behaviors.

6.3.1 Problem Formulation

We formulate this as a partially observable multi-sequential decision-making problem. At each timestep t, the global simulator state $s_t \in \mathcal{S}$ encodes the static road graph and the dynamic states of all agents. Each agent i receives only a partial observation $o_t^i = \mathcal{O}(s_t, i)$ within its local field-of-view. The agent then chooses an action $a_t^i \in \mathcal{A}$ (e.g., acceleration, steering, or maneuver) according to its policy $\pi_\theta(a_t^i|o_t^i)$, and the environment transitions according to $s_{t+1} \sim T(s_t, a_t^1, \ldots, a_t^N)$.

The key challenge in training a human-like policy π_θ is that task-based rewards (e.g., reaching goals, avoiding collisions, or staying on the road) encourage efficiency and safety but do not guarantee realism. Such rewards often lead to behaviors that maximize task success while diverging from human driving norms, for example, accelerating unnaturally to reach goals or performing overly sharp maneuvers to avoid collisions. To address this, we introduce a pretrained reference policy π_{ref} that captures the human driving distribution and provides

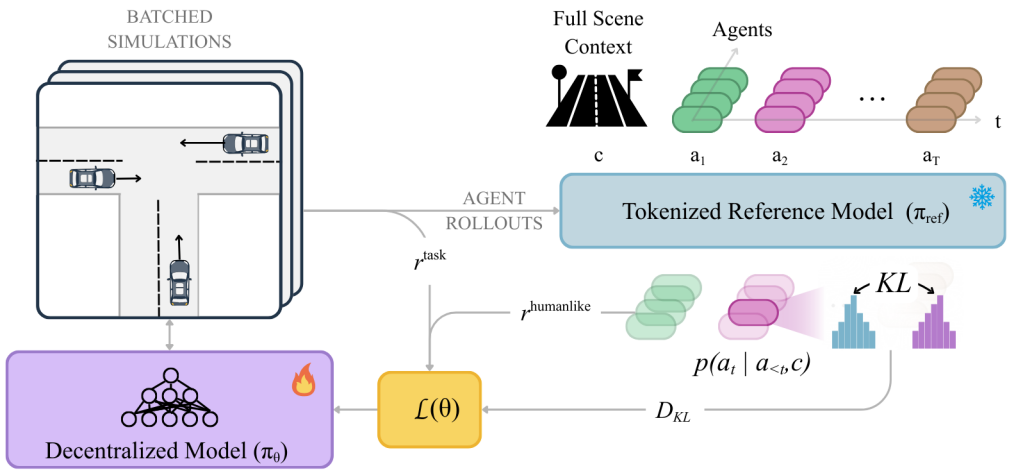

Figure 6.1: **Overview of SPACeR Framework.** Self-play reinforcement learning is anchored to a pretrained tokenized reference model π_{ref}, which provides a human-likeness distributional signal. The self-play policy π_θ is decentralized and conditioned on local observations, whereas π_{ref} is centralized and conditioned on the full scene context.

a realism signal during self-play, anchoring learning to behaviors observed in real data.

$$r_t \;=\; r_t^{\text{task}} \;+\; \alpha\, r^{\text{humanlike}}(s_t, a_t), \tag{6.1}$$

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PPO}}(\theta; A[r]) - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid o_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t)\big), \tag{6.2}$$

Eq. 6.2 combines three components:

- **Task performance:** $\mathcal{L}_{\text{PPO}}(\theta)$ is the standard PPO objective [104], encouraging agents to reach goals, avoid collisions, and stay on the road.
- **Human-likeness reward:** Inspired by prior work [105], we define a log-likelihood based reward that provides dense, per-timestep feedback by scoring each executed action under the reference distribution:

  $$r_{\text{humanlike}}(s_t, a_t) = \log \pi_{\text{ref}}(a_t \mid s_t). \tag{6.3}$$

  This formulation corresponds to maximizing the likelihood of human-like actions, thereby encouraging realistic driving behaviors beyond task success.
- **Distributional alignment:** The KL divergence $D_{\text{KL}}(\pi_\theta(\cdot \mid o_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t))$ acts as a dense signal at every timestep, directly encouraging the policy to align its action distribution with the human driving distribution captured by the reference model. For notational simplicity, we omit the agent index i; the KL divergence is computed independently for each agent.

6.3.2 Pretrained Reference Tokenized Model

To incorporate human-likeness into self-play, we introduce a pretrained reference policy π_{ref}, trained on real-world human driving trajectories, to serve as a proxy for the human driving distribution. Unlike task-based rewards, which only encourage efficiency and safety, π_{ref} provides a distributional signal that anchors policies toward realistic behavior. In principle, π_{ref} can be any data-driven generative motion model, such as a diffusion model or a tokenized model. In this work, we focus on tokenized models (e.g., SMART [19], CAT-K [86]), since they provide tractable likelihoods estimates suitable for distributional alignment.

Centralized Reference Tokenized Model. Tokenized models such as SMART [19] operate in an autoregressive fashion: given the past joint actions $a_{<t}$ and the scene context c (e.g., road graph and initial states), the model predicts a distribution over the next joint action $a_t = (a_t^1, \ldots, a_t^N)$, where t denotes the discrete timestep and N is the number of agents in the scene. Under a conditional independence assumption across agents, this factorizes as

$$p(a_t \mid a_{<t}, c) = \prod_{i=1}^{N} p(a_t^i \mid a_{<t}, c). \tag{6.4}$$

In our framework, we treat $p(\cdot)$ as the pretrained reference policy π_{ref}, which provides a distinct distribution for each agent's action a_t^i at every timestep, conditioned on the shared scene context and action history.

Importantly, the ability of π_{ref} to assign a distinct distribution to each agent's action at every timestep directly addresses a central challenge in multi-agent reinforcement learning: the credit assignment problem. It is often unclear which agent, and at which timestep, is responsible for a positive or negative outcome. This fine-grained feedback, grounded in the distributional signal of π_{ref}, enables shaping learning signals on a per-agent, per-timestep basis rather than relying solely on sparse, trajectory-level rewards.

(1) Tractable training without autoregressive sampling. Unlike autoregressive generation, which requires sequential token sampling, our approach only requires a single forward pass of the reference model per rollout batch. This provides the full per-agent, per-timestep action distribution, making the training pipeline efficient and scalable.

(2) Aligned action space. By ensuring that π_θ and π_{ref} operate over the same discrete action space, we avoid online tokenization during training. This alignment not only reduces computational overhead but also enables the KL divergence to be computed directly in closed form:

$$D_{\text{KL}}\big(\pi_\theta(\cdot \mid o_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t)\big) = \sum_{a \in \mathcal{A}} \pi_\theta(a \mid o_t) \, \log \frac{\pi_\theta(a \mid o_t)}{\pi_{\text{ref}}(a \mid s_t)}. \tag{6.5}$$

(3) Privileged information and comparison to log-replay data. Note that in Eq. 6.2, the reference tokenized model is centralized, observing the full state of all agents rather than local views. This setup resembles privileged information in a teacher–student framework in Robotics and Computer Vision [106, 107]. Unlike log replay, the reference

model generalizes beyond recorded trajectories and provides guidance in unseen self-play states.

Reference Models vs. WOSAC Metrics. WOSAC metrics [19] estimate per-feature likelihoods (kinematic, interactive, and map-based) based on ground-truth future trajectories. In contrast, tokenized reference models directly define a distribution over entire action sequences, offering two advantages: (i) evaluation does not require logged future trajectories, and (ii) the model provides a principled sequence-level distribution rather than feature-wise likelihoods tied to manual design metrics.

6.3.3 Practical Considerations of building human-like self-play

Here, we outline common problems in standard RL settings and the corresponding considerations for building human-like self-play.

Goal-reaching reward and post-goal behavior In previous works [31, 34], agents are typically rewarded only upon reaching their goal, and are removed from the scenario once the goal is reached. This formulation has two main drawbacks: (i) agents are incentivized to accelerate unnaturally in order to reach the goal as quickly as possible, and (ii) the number of active agents decreases after goals are reached, making the scenario progressively easier. Qualitatively, with this original formulation, if the agents are not removed, they would mostly stop once they reach the goal, making the scenarios unrealistic. To address this issue, we adopt *goal-dropout*: agents are trained with and without goal conditioning while receiving a terminal goal reward. This reduces reliance on explicit goal inputs. We further show that, when anchored to a reference model, the explicit goal reward can be removed without performance loss.

Tokenized Action Space and policy frequency: Contrary to previous works that utilize unicycle dynamics, we align the action space with the reference model by adopting a tokenized trajectory action space with K-disk from the data [18, 19]. This ensures compatibility for computing likelihoods and KL divergence without online tokenization. Therefore, the simulation frequency and policy frequency may differ; for example, the policy can operate at 5 Hz while the simulator runs at 10 Hz.

6.4 Experimental Results

We validate whether self-play policies trained with a reference model produce behaviors that are both *human-like* and *reactive*. Experiments are conducted on large-scale traffic scenarios from the Waymo Open Motion Dataset (WOMD) [13], measuring how closely the learned policies match the human driving distribution and how effectively they adapt to interactions in closed-loop simulation.

6.4.1 Implementation Details

We conduct all experiments in GPUDrive [31], a GPU-accelerated, data-driven simulator built on WOMD [13]. Each WOMD scenario spans 9 seconds; following the setup of [93], we initialize at 1 second and simulate the remaining 8 seconds. We train on 10k scenarios. In the main vehicle experiments (table 6.1), we control only vehicles, while pedestrians and cyclists follow their logged trajectories. For the VRU experiments (table 6.2), we train SPACeR by controlling all agent types, but report metrics only on pedestrian/cyclist targets.

Observation Space. We model the RL problem as a Partially Observed Stochastic Game [108], where agents act simultaneously under partial observability. Each controlled vehicle receives a local observation o_t^i in an ego-centric coordinate frame, including nearby vehicles, lane geometry, goal points (optionally), and relevant road features within a 50 m radius. Agents do not receive temporal history, and all features are normalized to the range $[-1, 1]$.

Action Space. We adopt a tokenized trajectory action space following [18, 19]. Using the K-disk algorithm, we cluster short-horizon trajectories in Cartesian space into $K = 200$ discrete tokens. Each token represents a 0.1-second step with a horizon length of 2, corresponding to a 5 Hz action frequency. This setup balances two considerations: providing sufficiently fine-grained distributional signals from the SMART reference model while keeping memory usage manageable. Since actions are defined directly in Cartesian space, no explicit dynamics model is assumed; the simulator advances the scenario according to the selected token.

Reward Formulation. For each agent, the task reward is

$$\begin{aligned} r^{\text{task}}(s_t, a_t) \;=\; & w_{\text{goal}}\, \mathbb{I}[\text{Goal achieved}] - w_{\text{collided}}\, \mathbb{I}[\text{Collided}] \\ & - w_{\text{offroad}}\, \mathbb{I}[\text{Offroad}] + w_{\text{humanlike}}\, r^{\text{humanlike}}(s_t, a_t), \end{aligned}$$

where $\mathbb{I}[\cdot] \in \{0, 1\}$. By default, we set $w_{\text{collided}} = w_{\text{offroad}} = 0.75$. The weights w_{goal} and $w_{\text{humanlike}}$ are varied in the following ablation experiments to study the trade-off between task completion, safety, and human-likeness. Note that other than the reward, we also directly compare the KL divergence between trained policy and referenced policy, where we use CAT-K [86] as the reference model.

Model Architecture and Training Details: We use a late-fusion feedforward model [31, 34], where ego, partner, and road-graph features are each embedded with a two-layer MLP and then concatenated. The fused representation is fed into an actor head and a critic head. During training, we control up to 64 agents with a shared decentralized policy π_θ, where each agent samples actions based on its local observation. We optimize with Proximal Policy Optimization (PPO) [104], with hyperparameters provided in appendix C.3. Training is performed on a single NVIDIA A100 GPU for 1 billion environment steps. During training, all VRUs (pedestrians and cyclists) follow the logged trajectories except for the VRU experiments in Table 6.2, where we train SPACeR for all agent types, and calculate metrics on VRUs.

Method	Composite ↑	Kinematic ↑	Interactive ↑	Map ↑	minADE ↓	Collision ↓	Off-road ↓	Throughput ↑
PPO	0.710 ± 0.01	0.327 ± 0.01	0.751 ± 0.01	0.875 ± 0.00	12.725 ± 2.53	0.038 ± 0.005	0.053 ± 0.00	211.8 ± 5.6
HF-PPO	0.716 ± 0.00	0.341 ± 0.00	0.756 ± 0.01	0.880 ± 0.00	12.254 ± 1.02	0.044 ± 0.006	0.053 ± 0.00	211.8 ± 5.6
SPACeR	0.741 ± 0.00	0.411 ± 0.00	0.779 ± 0.01	0.880 ± 0.00	4.101 ± 0.09	0.036 ± 0.010	0.056 ± 0.00	211.8 ± 5.6
SMART	0.720	0.450	0.725	0.870	1.840	0.17	0.13	22.5 ± 0.0
CAT-K	0.766	0.490	0.792	0.890	1.470	0.06	0.09	22.5 ± 0.0

Table 6.1: **Results on the WOSAC Validation Set.** Our proposed method outperforms other self-play approaches across all realism metrics, while achieving ∼10× higher throughput than imitation-learning (shaded) methods, with competitive performance and lower collision/off-road rates. Throughput is measured in scenarios/sec at 5 Hz on a single A100 GPU.

6.4.2 Human-Like Self-Play Evaluation on WOSAC

To measure humanlike behaviors, we adopt the evaluation protocol from the Waymo Open Sim Agent Challenge (WOSAC) [93]. WOSAC evaluates the distributional realism of simulated agents by comparing their rollouts against held-out human driving data, details provided in appendix C.4. Metrics assess kinematics (e.g., speed, acceleration), inter-agent interactions, and adherence to map constraints, with results aggregated into a composite realism score.

PPO (Self-Play). Decentralized PPO trained only with the task reward r_{task} [31]; no realism signals are used.

HR-PPO [33]. PPO regularized toward a behavioral cloning (BC) reference using KL divergence only (no likelihood term). For fairness, the BC model shares the same backbone as π_θ, is decentralized, and conditions only on local observations; it is trained on the WOMD training split.

Imitation learning baselines. We evaluate two tokenized closed-loop models—SMART [19] and CAT-K [86]. Both use the same backbone and action vocabulary ($K = 200$) and run at 5 Hz. SMART is first trained by behavior cloning on WOMD and then fine-tuned following the CAT-K closed-loop fine-tuning protocol; Implementation details and hyperparameters are provided in appendix C.3.

Discussion of Main Results. In table 6.1, anchoring self-play with a pretrained reference model substantially improves realism across all Sim Agent metrics. By contrast, HR-PPO reduces *minADE* but yields only a modest realism gain, whereas our method achieves clear improvements in kinematics, underscoring the value of alignment with a strong reference model. Relative to imitation-learning methods, our approach attains lower collision and off-road rates and even surpasses SMART in composite realism.

For efficiency and scale, our decentralized MLP has ∼65k parameters versus ∼3.2M for CAT-K (∼50× smaller). Briefly, we observe ∼10× higher closed-loop throughput (single A100, 5 Hz); see the rightmost column of table 6.1 and section 6.4.3 for details. CAT-K remains the strongest IL baseline, but our method preserves human-likeness at substantially higher speed and greater reactivity, which we further demonstrate in the closed-loop policy

Method	Composite ↑	Kinematic ↑	Interactive ↑	Map ↑	minADE ↓
PPO	0.648	0.242	0.683	0.835	7.712
HR-PPO	0.668	0.285	0.700	0.847	7.014
SPACeR (Ours)	**0.729**	**0.413**	**0.762**	**0.866**	**2.066**

Table 6.2: **VRU realism metrics on WOSAC (pedestrians and cyclists).** SPACeR outperforms PPO and HR-PPO by a large margin, achieving substantial gains across all realism metrics and minADE.

evaluation (section 6.4.3).

Reference Model Quality. We further study how reference model quality affects SPACeR performance by training SMART models of three sizes: 0.3M, 1M, and 3M parameters. Each model is also fine-tuned with CAT-K, producing a total of six reference models (fig. 6.2). SPACeR policy still reaches 0.732 realism even with the smallest underperforming reference model (0.636). We observe a similarly stable performance when varying the token vocabulary size with 3M params (100, 200, 400; different markers in fig. 6.2): larger vocabularies increase reference model's realism, but the resulting SPACeR policies all remain clustered around 0.73. This indicates that the reference model serves mainly as a soft prior that guides humanlike behavior, rather than a target for direct imitation.

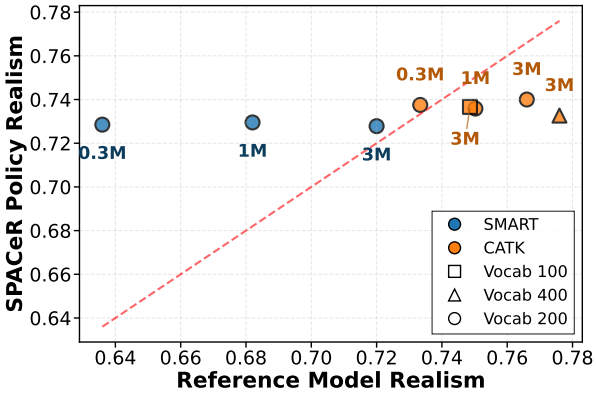

Figure 6.2: **Reference Model Quality Effect on SPACeR Realism.** SPACeR maintains high composite realism (0.73) even when anchored to a reference model of only 0.3 M parameters with realism 0.64.

Because SPACeR agents learn through closed-loop interaction, they can refine their behavior beyond the limitations of the reference model. In addition, using a reference model that has itself undergone closed-loop fine-tuning (CAT-K) provides stronger behavioral signals, which further improves the resulting SPACeR agents performance.

Evaluation on Pedestrian and Cyclist Behavior. We evaluate SPACeR on the WOSAC validation set restricted to pedestrians and cyclists to isolate VRU behavior (table 6.2). Although absolute composite realism scores are 0.1 lower than for vehicles, SPACeR still improves performance by a large margin over PPO and HR-PPO across all metrics, including Composite, Kinematic, Interactive, and Map realism, as well as minADE. This demonstrates that the anchoring mechanism remains effective even under the higher stochasticity and variability of VRU motion. Additional design choices and component ablations for VRU simulation are provided in appendix C.1.

Effect of Anchoring Parameters on SPACeR Performance. We investigate the effect of the anchoring parameters, namely the likelihood weight α and the KL alignment weight β, by activating each term independently. Top-1 refers to evaluating agents by always taking the single most probable action, whereas top-5 evaluates realism under stochastic sampling from the top five most probable actions. Since SimAgent measures distributional realism over 32 joint rollouts, maintaining diversity through top-k sampling generally yields higher composite scores [93].

The likelihood and KL alignment terms play different roles in shaping policy behavior. Optimizing the likelihood term alone increases log likelihood but reduces diversity, as reflected by a drop in entropy in the training curves (fig. C.1). This produces small improvements in top-1 composite realism but harms top-5 performance. In contrast, the KL alignment term raises both likelihood and KL consistency while preserving entropy, which improves both top-1 and top-5 realism. The training curves in fig. C.1 illustrate how log likelihood, KL divergence, and entropy evolve under each parameter setting.

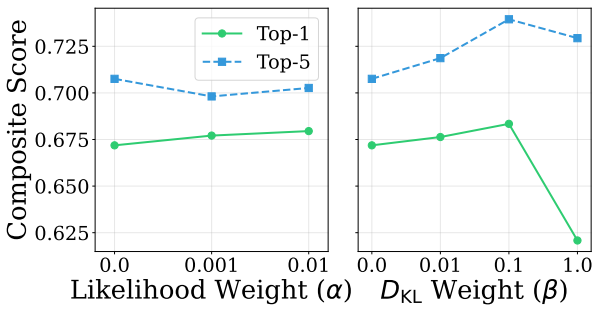

Figure 6.3: **Anchoring parameter ablation.** Likelihood-only improves top-1, while KL alignment improves both top-1 and top-5 realism.

In the ablation study (table C.2), KL alignment contributes most to realism. While log-likelihood rewards are popular [105], their impact on realism is modest, likely because the real-world driving distribution is highly multi-modal, so maximizing likelihood alone does not yield a stable or sufficiently discriminative signal. A second insight is that once the policy is anchored to the reference distribution, the explicit goal-reaching reward can be removed, further improving realism.

6.4.3 SPACeR Agents for Fast & Accurate Closed-Loop Planner Evaluation

A key application of humanlike agents is the closed-loop evaluation of ego-vehicle motion planners. In this setup, the planner under test controls the ego vehicle, while surrounding agents are driven by learned policies. This enables realistic modeling of inter-agent interactions, reactive behaviors, and humanlike decision-making without risking safety.

We evaluate our SPACeR agents along two dimensions: *closed-loop evaluation realism* and *throughput*. Unlike distributional realism in section 6.4.2, which measures similarity to human data, realism here refers to how sim agents change planner rankings under closed-loop interaction.

To assess realism, we compare against two baselines: (i) open-loop log-replay evaluation,

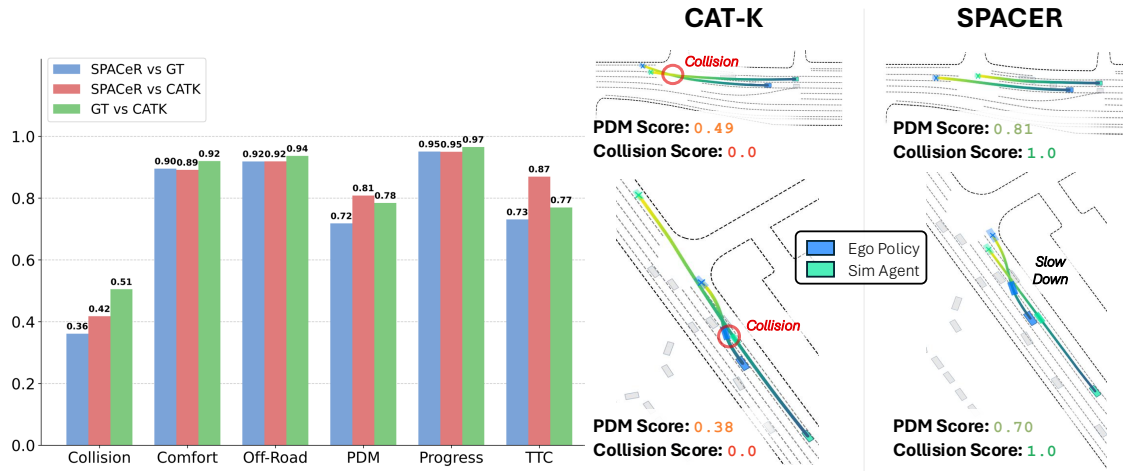

Figure 6.4: **Correlation Coefficients of PDM Scores Across Policy Evaluation Strategies.** Using our approach leads to consistently lower correlations with ground-truth log replays across all metrics, suggesting more realistic penalization of unsafe policies—especially for collisions.

and (ii) closed-loop simulation using CAT-K [86], a state-of-the-art traffic simulator on WOSAC. We evaluate a diverse set of planners: 22 self-play–trained policies, 10 sampling-based Frenet planners, and 10 IDM-based planners [9] (more details can be found in appendix C.6). For each planner, we compute PDM scores [109] across multiple scenes under three traffic-simulation strategies: ground-truth logs (GT), CAT-K rollouts, and SPACER agent policies. We then measure the correlation of PDM scores between strategies. As shown in fig. 6.4, our approach yields consistently lower correlations with GT replays when compared to CAT-K. We interpret this as evidence that our simulation is more reactive, and suppresses unrealistic planner behavior more effectively—particularly in collision scenarios. Qualitative examples (see fig. 6.4 and the project website[1]) further highlight that SPACER agents respond naturally in diverse scenarios, avoiding collisions when reasonable and minimizing unrealistic off-road behaviors.

For throughput, we benchmark against SMART [19], a leading multi-agent motion generator on the WOSAC leaderboard [93]. To ensure fairness, we run both methods at 5 Hz for full 8-second episodes on a single NVIDIA A100 GPU. SMART achieves 22.5 ± 0.01 scenarios/sec, while our method reaches 211.8 ± 5.64 scenarios/sec—a $\sim 10\times$ speedup. Moreover, GPUDrive can be optimized for another order-of-magnitude efficiency gain [36, 110]. All experiments were run on a dual Intel Xeon Platinum 8358 (64 cores / 128 threads, 2.6 GHz) server with a single A100 GPU, and results are averaged over 5 seeds.

In summary, SPACER agents deliver both *more realistic closed-loop evaluation* and *significantly higher throughput*, enabling reliable and scalable benchmarking of ego-motion planning policies.

6.5 Discussion

Limitation of WOSAC Metrics. During experiments, we found that WOSAC metrics can produce misleading scores. In fig. 6.5, the logged agent turns into a parking lot, while SPACeR continues straight without crossing the curb. Although this behavior has an off-road rate of 0.0, WOSAC penalizes it with a low map score because the metric rewards reproducing the logged trajectory rather than recognizing alternative valid behaviors. Likewise, when logs contain sensor noise leading to collisions or off-road trajectories, WOSAC assigns higher likelihood to agents that repeat these errors [111]. We provide more qualitative results in appendix C.5. This reveals a key limitation: WOSAC evaluates similarity to logged distributions, not necessarily safety or human-likeness, and can misalign with the goals of self-play RL.

Simulating VRUs (Pedestrians and Cyclists). SPACeR improves pedestrian and cyclist realism compared to prior baselines by a large margin, showing that self-play anchoring transfers beyond vehicle agents section 6.4.2. Nevertheless, VRU self-play remains underexplored: current WOSAC metrics (e.g., collision, off-road) are tailored to vehicles and do not capture sidewalk adherence, crosswalk usage, or other pedestrian specific behaviors. Reward design must likewise reflect these VRU-specific considerations. In this work we primarily use likelihood and collision signals for VRU motion. Developing VRU-aware metrics, reward shaping, and scene-level infrastructure is an important direction for enabling more realistic pedestrian and cyclist simulation.

Training Efficiency Bottlenecks. In our current setting, each run requires roughly 24–48 hours, partly due to GPUDrive's lack of multi-GPU support. Future extensions could exploit multi-GPU training or alternative backends such as PufferLib [110], which has demonstrated order-of-magnitude speedups. Memory usage also limits scalability, especially for architectures like SMART that encode pairwise interactions explicitly. Recent advances in memory-efficient design (e.g., [112]) may help reduce overhead and increase training throughput.

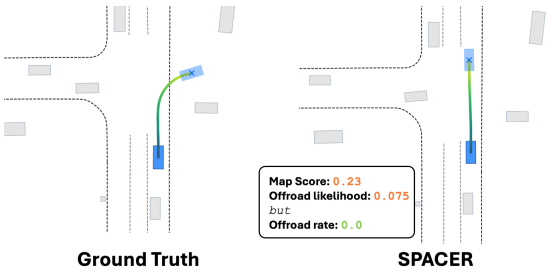

Figure 6.5: **Example scenarios where WOSAC metrics produce unrealistic estimates.**

6.6 Conclusion

In this chapter, we explore scalability as a key dimension of simulation, focusing on enabling efficient closed-loop interaction while preserving realistic, human-like behavior. While generative models have become a dominant approach for modeling such behaviors, their computational cost limits their deployment at scale.

We introduced SPACeR, which anchors self-play RL to a pretrained tokenized model via

KL alignment, providing lightweight realism signals without relying on logged trajectories. On the Waymo Sim Agent Challenge, it achieves higher realism than prior self-play while producing policies that are $\sim$50$\times$ smaller and $\sim$10$\times$ faster than state-of-the-art imitation models. Our policies remain reactive and humanlike, avoiding the unrealistic collisions often observed in imitation methods during closed-loop planner evaluation. Together, these results position SPACER as a step toward scalable, real-time closed-loop evaluation, and ultimately training, of planners under realistic large-scale traffic scenarios.

Looking forward, this work opens several new directions for self-play-based simulation. One promising direction for self-play reinforcement learning is to train a diverse population of policies. A key question is whether controllability, extensively studied in generative models and in previous chapters, can be extended to self-play, enabling structured and counterfactual scenario generation with significantly higher efficiency. In addition, self-play provides a natural framework for closed-loop training of ego policies, where all agents interact and co-evolve, allowing experience to be accumulated jointly rather than from isolated single-agent data. This multi-agent learning paradigm offers a promising path toward more scalable and realistic training pipelines, where both simulation agents and ego policies can improve together over time.

Chapter 7

Conclusion and Future Work

Robots are becoming increasingly capable and are being deployed alongside humans. To operate safely and intelligently in such environments requires experience with human interaction. However, collecting such data at scale is costly, risky, and inherently limited. Even with arbitrarily large datasets, real-world observations only capture what happened, not how behavior would change under different robot actions. This dissertation addresses these challenges through data-driven behavior simulation, enabling counterfactual reasoning in closed-loop interactive environments.

A central theme of this dissertation is that useful behavior simulation requires jointly modeling social behavior, enabling controllable counterfactual generation, and scaling closed-loop multi-agent simulation. We begin by modeling socially compatible behavior through a data-driven approach that quantifies social preferences, enabling simulation of diverse behaviors from courteous to aggressive. We then address counterfactual scenario generation for targeted evaluation, introducing a controllable diffusion framework for long-tail and safety-critical scenarios, enabling "what-if" generation and stress testing under adversarial behaviors. Building on this, we extend controllability by bridging language and trajectories, developing a language-conditioned model for flexible and interpretable scenario specification. A key limitation of current generative models is their computational cost, which limits their use in fast closed-loop simulation. To address this, we propose a self-play reinforcement learning framework anchored by a pretrained reference model, enabling lightweight decentralized multi-agent simulation with high realism and reactivity at scale. These contributions open several promising directions for future research.

Closed-loop training. With improved controllability of counterfactual behaviors and the ability to run large-scale closed-loop simulations, a natural next step is to move beyond evaluation toward training robot policies. This opens up opportunities for reinforcement learning with closed-loop simulation, raising the challenge of designing curricula and coverage-driven strategies that prioritize informative and safety-critical scenarios to enable robust and adaptable policies. As autonomous systems move toward end-to-end and vision–language–action paradigms, behavior simulation can be combined with sensor simulation (e.g., Gaussian splatting) for fully closed-loop training. However, such approaches remain computationally

expensive at scale. A key question is what level of representation is sufficient: while pixel-level simulation offers high fidelity, more abstract representations may be more efficient for learning [113].

Simulation Abstractions and World Models. Another important direction is understanding the appropriate level of abstraction for simulation. The dissertation primarily studies explicit behavior simulation, where interaction dynamics are modeled through structured representations such as trajectories and agent states, and can optionally be combined with sensor simulation methods such as Gaussian splatting for closed-loop end-to-end world simulation. Such representations intentionally abstract away low-level sensor details to enable more controllable, and computationally efficient modeling.

Recently, there has been increasing interest in world models that learn predictive dynamics directly from high-dimensional observations such as images or videos [114, 115]. Compared to explicit simulation pipelines, these approaches operate at a more fine-grained and potentially less lossy representation level, enabling richer perceptual grounding. However, current world models still face major challenges, including computational cost, speed, and maintaining a consistent long-horizon simulation. Rather than viewing these paradigms as competing approaches, they may be complementary. Explicit simulation can provide fast and controllable interaction rollouts for efficient policy training, while world models may serve as high-fidelity perceptual environments that ground policies in pixel-space observations during fine-tuning or final adaptation stages.

Simulation Fidelity and Metrics. To make simulation informative for real-world autonomy, a key challenge is to measure and improve simulation fidelity. In particular, this requires quantifying distributional realism—whether simulation matches real-world behavior at scale, both in statistical properties and in the coverage of interaction patterns. This raises the question of metric design. Beyond raw trajectories, metrics should capture interaction-relevant structure, such as relative motion, spacing, right-of-way intent, and safety-critical proximity, while incorporating temporal dynamics and behavioral intent.

Safety Evidence. A central challenge is how to turn simulation into safety evidence that transfers to real-world deployment. This includes: (i) scenario-structured safety reasoning, connecting simulation-based evaluation with operational design domains and SOTIF-style hazard analysis [3]; (ii) human-referenced safety baselines, comparing policy behavior to human behavior in safety-critical regimes; and (iii) statistical grounding of simulation, using limited real-world data to calibrate metrics and, more importantly, quantify the *correlation* between simulation and real-world outcomes, enabling reliable estimation of tail risk beyond average performance [116].

Heterogeneous Agents and Mobility Platforms. Humans interact in the environment through different mobility platforms, such as walking, cycling, or driving, each with distinct dynamics and behavioral patterns [117]. These differences lead to heterogeneous interactions that significantly affect system behavior and safety risks. Modeling such variation is essential for realistic simulation and for ensuring generalization across deployment settings.

Extension to Embodied Systems. An important direction is extending behavior simulation to different forms of embodiment, where choosing representation becomes critical.

This work adopts a trajectory-based representation, modeling motion and interaction patterns where intent is implicitly encoded in behavior. This is well-suited for high-level interaction tasks such as navigation and driving, where interactions are primarily governed by relative motion. In other settings, such as human–robot manipulation, interaction also includes physical contact and geometry. In these cases, behavior must be coupled with lower-level representations that capture contact, force, and 3D structure. While high-level intent remains relevant, it must be grounded in physical interaction to produce realistic outcomes. Despite these differences, the core paradigm of data-driven, closed-loop simulation of interactive behavior remains applicable across embodiments. A key open question is identifying the appropriate level of abstraction for each setting, balancing fidelity and efficiency.

Bibliography

[1] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.

[2] Mayank Mittal et al. "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning". In: *arXiv preprint arXiv:2511.04831* (2025).

[3] International Organization for Standardization. *ISO 21448: Road Vehicles — Safety of the Intended Functionality (SOTIF)*. ISO 21448:2022. 2022.

[4] Simon Suo et al. "TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10400–10409.

[5] Danfei Xu et al. "BITS: Bi-level Imitation for Traffic Simulation". In: *IEEE International Conference on Robotics and Automation*. 2023, pp. 2929–2936.

[6] Nico Montali et al. "The Waymo Open Sim Agents Challenge". In: *Advances in Neural Information Processing Systems* 36 (2024).

[7] Shital Shah et al. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: *Field and Service Robotics: Results of the 11th International Conference*. 2018, pp. 621–635.

[8] Panpan Cai et al. "SUMMIT: A Simulator for Urban Driving in Massive Mixed Traffic". In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 4023–4029.

[9] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical Review E* 62.2 (2000), p. 1805.

[10] Pablo Alvarez Lopez et al. "Microscopic Traffic Simulation Using SUMO". In: *IEEE Intelligent Transportation Systems Conference*. 2018, pp. 2575–2582.

[11] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Conference on Robot Learning*. 2017, pp. 1–16.

[12] Holger Caesar et al. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11621–11631.

[13] Scott Ettinger et al. "Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset". In: *IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9710–9719.

[14] Luca Bergamini et al. "SimNet: Learning Reactive Self-Driving Simulations from Real-World Observations". In: *IEEE International Conference on Robotics and Automation*. 2021, pp. 5119–5125.

[15] Kaiming He et al. "Deep residual learning for image recognition". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[16] Maximilian Igl et al. "Symphony: Learning Realistic and Diverse Agents for Autonomous Driving Simulation". In: *IEEE International Conference on Robotics and Automation*. 2022, pp. 2445–2451.

[17] Wei-Jer Chang et al. "Analyzing and Enhancing Closed-loop Stability in Reactive Simulation". In: *IEEE Intelligent Transportation Systems Conference*. 2022, pp. 3665–3672.

[18] Jonah Philion, Xue Bin Peng, and Sanja Fidler. "Trajeglish: Traffic Modeling as Next-Token Prediction". In: *International Conference on Learning Representations*. 2024.

[19] Wei Wu et al. "SMART: Scalable Multi-Agent Real-Time Simulation via Next-Token Prediction". In: *Advances in Neural Information Processing Systems*. 2024.

[20] Ari Seff et al. "MotionLM: Multi-Agent Motion Forecasting as Language Modeling". In: *IEEE/CVF International Conference on Computer Vision*. 2023, pp. 8579–8590.

[21] Ziyuan Zhong et al. "Guided conditional diffusion for controllable traffic simulation". In: *IEEE International Conference on Robotics and Automation*. 2023, pp. 3560–3566.

[22] Ziyuan Zhong et al. "Language-Guided Traffic Simulation via Scene-Level Diffusion". In: *Conference on Robot Learning*. 2023.

[23] Chiyu Jiang et al. "MotionDiffuser: Controllable Multi-Agent Motion Prediction Using Diffusion". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9644–9653.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[25] Ethan Pronovost et al. "Scenario diffusion: Controllable driving scenario generation with diffusion". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 68873–68894.

[26] Chiyu Max Jiang et al. "SceneDiffuser: Efficient and Controllable Driving Simulation Initialization and Rollout". In: *Advances in Neural Information Processing Systems*. 2024.

[27] Chejian Xu et al. "DiffScene: Diffusion-Based Safety-Critical Scenario Generation for Autonomous Vehicles". In: *The Second Workshop on New Frontiers in Adversarial Machine Learning*. 2023.

[28] Zhao-Heng Yin et al. "Diverse critical interaction generation for planning and planner evaluation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2021, pp. 7036–7043.

[29] Davis Rempe et al. "Generating useful accident-prone driving scenarios via a learned traffic prior". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17305–17315.

[30] Michael Janner et al. "Planning with diffusion for flexible behavior synthesis". In: *International Conference on Machine Learning*. 2022.

[31] Saman Kazemkhani et al. "GPUDrive: Data-Driven, Multi-Agent Driving Simulation at 1 Million FPS". In: *arXiv preprint arXiv:2408.01584* (2024).

[32] Daphne Cornelisse et al. *PufferDrive: A Fast and Friendly Driving Simulator for Training and Evaluating RL Agents*. Version 2.0.0. 2025. URL: `https://github.com/Emerge-Lab/PufferDrive`.

[33] Daphne Cornelisse and Eugene Vinitsky. "Human-compatible driving partners through data-regularized self-play reinforcement learning". In: *arXiv preprint arXiv:2403.19648* (2024).

[34] Daphne Cornelisse et al. "Building reliable sim driving agents by scaling self-play". In: *arXiv preprint arXiv:2502.14706* (2025).

[35] Chris Zhang et al. "Learning to drive via asymmetric self-play". In: *European Conference on Computer Vision*. 2024, pp. 149–168.

[36] Marco Cusumano-Towner et al. "Robust autonomy emerges from self-play". In: *arXiv preprint arXiv:2502.03349* (2025).

[37] Wei-Jer Chang et al. "Editing Driver Character: Socially-Controllable Behavior Generation for Interactive Traffic Simulation". In: *IEEE Robotics and Automation Letters* 8.9 (2023), pp. 5432–5439.

[38] Quanyi Li et al. "MetaDrive: Composing Diverse Driving Scenarios for Generalizable Reinforcement Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2023), pp. 3461–3475.

[39] Jingkang Wang et al. "AdvSim: Generating Safety-Critical Scenarios for Self-Driving Vehicles". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9909–9918.

[40] Wilko Schwarting et al. "Social behavior for autonomous vehicles". In: *Proceedings of the National Academy of Sciences* 116.50 (2019), pp. 24972–24978.

[41] Liting Sun et al. "Courteous autonomous cars". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2018, pp. 663–670.

[42] Qiao Sun et al. "M2I: From Factored Marginal Trajectory Prediction to Interactive Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6543–6552.

[43] Ekaterina Tolstaya et al. "Identifying driver interactions via conditional behavior prediction". In: *IEEE International Conference on Robotics and Automation*. 2021, pp. 3473–3479.

[44] Dorsa Sadigh et al. "Planning for autonomous cars that leverage effects on human actions." In: *Robotics: Science and Systems*. 2016, pp. 1–9.

[45] David Fridovich-Keil et al. "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games". In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 1475–1481.

[46] Jaime F Fisac et al. "Hierarchical game-theoretic planning for autonomous vehicles". In: *IEEE International Conference on Robotics and Automation*. 2019, pp. 9590–9596.

[47] Zach Williams, Jushan Chen, and Negar Mehr. "Distributed Potential iLQR: Scalable Game-Theoretic Trajectory Planning for Multi-Agent Interactions". In: *arXiv preprint arXiv:2303.04842* (2023).

[48] Maulik Bhatt, Yixuan Jia, and Negar Mehr. "Strategic decision-making in multi-agent domains: A weighted constrained potential dynamic game approach". In: *IEEE Transactions on Robotics* (2025).

[49] Letian Wang et al. "Socially-compatible behavior design of autonomous vehicles with verification on real human data". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3421–3428.

[50] Balakrishnan Varadarajan et al. "MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction". In: *IEEE International Conference on Robotics and Automation*. 2022, pp. 7814–7821.

[51] Roger Koenker and Gilbert Bassett Jr. "Regression quantiles". In: *Econometrica: Journal of the Econometric Society* (1978), pp. 33–50.

[52] James W Taylor. "A quantile regression neural network approach to estimating the conditional density of multiperiod returns". In: *Journal of Forecasting* 19.4 (2000), pp. 299–311.

[53] Stepan Konev. "MPA: MultiPath++ Based Architecture for Motion Prediction". In: *arXiv preprint arXiv:2206.10041* (2022).

[54] Chen Tang, Wei Zhan, and Masayoshi Tomizuka. "Interventional behavior prediction: Avoiding overly confident anticipation in interactive prediction". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2022, pp. 11409–11415.

[55] Wei-Jer Chang et al. "SAFE-SIM: Safety-Critical Closed-Loop Traffic Simulation with Diffusion-Controllable Adversaries". In: *European Conference on Computer Vision*. 2024, pp. 242–258.

[56] Wenhao Ding et al. "Learning to collide: An adaptive safety-critical scenarios generating method". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2020, pp. 2243–2250.

[57] Wenhao Ding et al. "Multimodal Safety-Critical Scenarios Generation for Decision-Making Algorithms Evaluation". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1551–1558.

[58] Davis Rempe et al. "Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13756–13766.

[59] Holger Caesar et al. "nuPlan: A Closed-Loop ML-Based Planning Benchmark for Autonomous Vehicles". In: *arXiv preprint arXiv:2106.11810* (2021).

[60] Prafulla Dhariwal and Alexander Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[61] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021.

[62] Jack Lu et al. "SceneControl: Diffusion for Controllable Traffic Scene Generation". In: *IEEE International Conference on Robotics and Automation*. 2024.

[63] Niklas Hanselmann et al. "KING: Generating Safety-Critical Driving Scenarios for Robust Imitation via Kinematics Gradients". In: *European Conference on Computer Vision*. 2022, pp. 335–352.

[64] Yulong Cao et al. "AdvDO: Realistic Adversarial Attacks for Trajectory Prediction". In: *European Conference on Computer Vision*. 2022, pp. 36–52.

[65] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. "Generating adversarial driving scenarios in high-fidelity simulators". In: *IEEE International Conference on Robotics and Automation*. 2019, pp. 8271–8277.

[66] Qingzhao Zhang et al. "On adversarial robustness of trajectory prediction for autonomous vehicles". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15159–15168.

[67] Wenhao Ding et al. "A survey on safety-critical driving scenario generation—A methodological perspective". In: *IEEE Transactions on Intelligent Transportation Systems* (2023).

[68] Alexander Quinn Nichol and Prafulla Dhariwal. "Improved denoising diffusion probabilistic models". In: *International Conference on Machine Learning*. 2021, pp. 8162–8171.

[69] Jonathan Ho et al. "Video diffusion models". In: *arXiv preprint arXiv:2204.03458* (2022).

[70] Haruki Nishimura et al. "RAP: Risk-Aware Prediction for Robust Planning". In: *Conference on Robot Learning*. 2023, pp. 381–392.

[71] Simon Suo et al. "MixSim: A Hierarchical Framework for Mixed Reality Traffic Simulation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 9622–9631.

[72] Daniel Dauner et al. "Parting with misconceptions about learning-based vehicle motion planning". In: *Conference on Robot Learning*. 2023.

[73] Wei-Jer Chang et al. "LangTraj: Diffusion Model and Dataset for Language-Conditioned Trajectory Simulation". In: *IEEE/CVF International Conference on Computer Vision*. 2025, pp. 26622–26631.

[74] Brian Yang et al. "Diffusion-ES: Gradient-Free Planning with Diffusion for Autonomous and Instruction-Guided Driving". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 15342–15353.

[75] OpenAI. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[76] Fisher Yu et al. "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2636–2645.

[77] Tianwen Qian et al. "NuScenes-QA: A Multi-Modal Visual Question Answering Benchmark for Autonomous Driving Scenario". In: *AAAI Conference on Artificial Intelligence*. Vol. 38. 5. 2024, pp. 4542–4550.

[78] Yiheng Li et al. "WOMD-Reasoning: A Large-Scale Language Dataset for Interaction and Driving Intentions Reasoning". In: *International Conference on Machine Learning*. 2025.

[79] Shuhan Tan et al. "Promptable Closed-loop Traffic Simulation". In: *Conference on Robot Learning*. 2025, pp. 5087–5105.

[80] Shuhan Tan et al. "Language Conditioned Traffic Generation". In: *Conference on Robot Learning*. 2023, pp. 2714–2752.

[81] Zikang Zhou et al. "Query-Centric Trajectory Prediction". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 17863–17873.

[82] Shaoshuai Shi et al. "MTR++: Multi-Agent Motion Prediction with Symmetric Scene Modeling and Guided Intention Querying". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[83] Lucy Xiaoyang Shi et al. "Yell at your robot: Improving on-the-fly from language corrections". In: *arXiv preprint arXiv:2403.12910* (2024).

[84] V Sanh. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[85] Longzhong Lin et al. "Revisit Mixture Models for Multi-Agent Simulation: Experimental Study within a Unified Framework". In: *arXiv preprint arXiv:2501.17015* (2025).

[86] Zhejun Zhang et al. "Closed-loop supervised fine-tuning of tokenized traffic models". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2025, pp. 5422–5432.

[87] Zhiyu Huang et al. "Versatile Scene-Consistent Traffic Scenario Generation as Optimization with Diffusion". In: *arXiv preprint arXiv:2404.02524* (2024).

[88] Zhiming Guo et al. "SceneDM: Scene-Level Multi-Agent Trajectory Generation with Consistent Diffusion Models". In: *arXiv preprint arXiv:2311.15736* (2023).

[89] Wei-Jer Chang et al. "SPACeR: Self-Play Anchoring with Centralized Reference Models". In: *International Conference on Learning Representations*. 2026. URL: https://openreview.net/forum?id=Q5H3NCy18S.

[90] Dayi Dong et al. "MIMIC-D: Multi-Modal Imitation for Multi-Agent Coordination with Decentralized Diffusion Policies". In: *arXiv preprint arXiv:2509.14159* (2025).

[91] Antoine Bergerault, Volkan Cevher, and Negar Mehr. "Matching Multiple Experts: On the Exploitability of Multi-Agent Imitation Learning". In: *arXiv preprint arXiv:2602.21020* (2026).

[92] Kartik Nagpal et al. "Leveraging large language models for effective and explainable multi-agent credit assignment". In: *arXiv preprint arXiv:2502.16863* (2025).

[93] Nico Montali et al. "The Waymo Open Sim Agents Challenge". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 59151–59171.

[94] Micah Carroll et al. "On the utility of learning about humans for human-ai coordination". In: *Advances in Neural Information Processing Systems* 32 (2019).

[95] Chao Yu et al. "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24611–24624.

[96] Yuxin Chen et al. "Quantifying Interaction Level Between Agents Helps Cost-efficient Generalization in Multi-agent Reinforcement Learning". In: *Reinforcement Learning Journal* 4 (2024), pp. 1950–1964.

[97] Alex Kuefler et al. "Imitating Driver Behavior with Generative Adversarial Networks". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 204–211.

[98] Raunak Bhattacharyya et al. "Modeling Human Driving Behavior through Generative Adversarial Imitation Learning". In: *arXiv preprint arXiv:2006.06412* (2020).

[99] Negar Mehr et al. "Maximum-entropy multi-agent dynamic games: Forward and inverse solutions". In: *IEEE Transactions on Robotics* 39.3 (2023), pp. 1801–1815.

[100] Zhenghao Peng et al. "Improving Agent Behaviors with RL Fine-Tuning for Autonomous Driving". In: *European Conference on Computer Vision*. 2024, pp. 165–181.

[101] Keyu Chen et al. "RIFT: Closed-Loop RL Fine-Tuning for Realistic and Controllable Traffic Simulation". In: *arXiv preprint arXiv:2505.03344* (2025).

[102] Yulong Cao et al. "Reinforcement learning with human feedback for realistic traffic simulation". In: *IEEE International Conference on Robotics and Automation*. 2024, pp. 14428–14434.

[103] Ran Tian and Kratarth Goel. "Direct Post-Training Preference Alignment for Multi-Agent Motion Generation Models Using Implicit Feedback from Pre-training Demonstrations". In: *arXiv preprint arXiv:2503.20105* (2025).

[104] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *International Conference on Machine Learning*. 2017.

[105] Alejandro Escontrela et al. "Video prediction models as rewards for reinforcement learning". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 68760–68783.

[106] Tairan He et al. "OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning". In: *arXiv preprint arXiv:2406.08858* (2024).

[107] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: *IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9630–9640.

[108] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. "Dynamic programming for partially observable stochastic games". In: *AAAI Conference on Artificial Intelligence*. 2004, pp. 709–715.

[109] Daniel Dauner et al. "NavSim: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 28706–28719.

[110] Joseph Suárez. "PufferLib: Making Reinforcement Learning Libraries and Environments Play Nice". In: *arXiv preprint arXiv:2406.12905* (2024).

[111] Mingyi Wang et al. "Do LLM Modules Generalize? A Study on Motion Generation for Autonomous Driving". In: *arXiv preprint arXiv:2509.02754* (2025).

[112] Jianbo Zhao et al. "DRoPE: Directional Rotary Position Embedding for Efficient Agent Interaction Modeling". In: *arXiv preprint arXiv:2503.15029* (2025).

[113] Lan Feng et al. "RAP: 3D Rasterization Augmented End-to-End Planning". In: *arXiv preprint arXiv:2510.04333* (2025).

[114] Danijar Hafner et al. "Mastering Atari with Discrete World Models". In: *International Conference on Learning Representations*. 2021.

[115] Anthony Hu et al. "GAIA-1: A Generative World Model for Autonomous Driving". In: *arXiv preprint arXiv:2309.17080* (2023).

[116] Rachel Luo et al. "Sim2Val: Leveraging Correlation Across Test Platforms for Variance-Reduced Metric Estimation". In: *arXiv preprint arXiv:2506.20553* (2025).

[117] Yu-Hsiang Chen et al. "HetroD: A High-Fidelity Drone Dataset and Benchmark for Autonomous Driving in Heterogeneous Traffic". In: *IEEE International Conference on Robotics and Automation*. 2026.

[118] Tianpei Gu et al. "Stochastic trajectory prediction via motion indeterminacy diffusion". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17113–17122.

[119] Boris Ivanovic et al. "trajdata: A Unified Interface to Multiple Human Trajectory Datasets". In: *arXiv preprint arXiv:2307.13924* (2023).

[120] Edward J Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2021.

[121] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*. 2015.

[122] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *International Conference on Learning Representations*. 2021.

[123] Adam Gleave et al. "imitation: Clean Imitation Learning Implementations". In: *arXiv preprint arXiv:2211.11972* (2022).

Appendix A

Supplementary Material for Safe-Sim

Supplementary videos are available on the project website.[1]

A.1 Qualitative Results

We present two sets of qualitative results. The first set, illustrated in Figure A.1, displays a variety of safety-critical simulations where altering the trajectory proposals modifies the collision types. The second set, depicted in Figure A.2, showcases simulations that demonstrate different collision scenarios achieved by adjusting the Time-To-Collision (TTC) to influence the safety-criticality of the situation. Unlike the STRIVE method, which tends to generate scenarios with limited variability, our approach utilizes multiple control mechanisms (such as varying trajectory proposals and safety-criticality levels) to create a broader spectrum of safety-critical conditions. This flexibility is particularly beneficial for testing and evaluating autonomous driving algorithms under various challenging conditions.

A.2 Details on Partial Diffusion

A.2.1 Methodology for Generating Trajectory Proposals

To generate trajectory proposals for the partial diffusion process, which aims to create potential collision scenarios, we present a straightforward method based on lane relationships. In addition to selecting different lane relationships to represent various types of collisions, we further refine our control over these scenarios by introducing two primary variations: 1) the relative distance to the conflict point and 2) the normal offsets of the lane, as illustrated in Figure A.3:

1. **Relative Distance to the Conflict Point:** This adjustment allows for the precise management of how vehicles navigate interactions, such as passing or yielding, by selecting specific accelerations that achieve the desired distance to the conflict point.

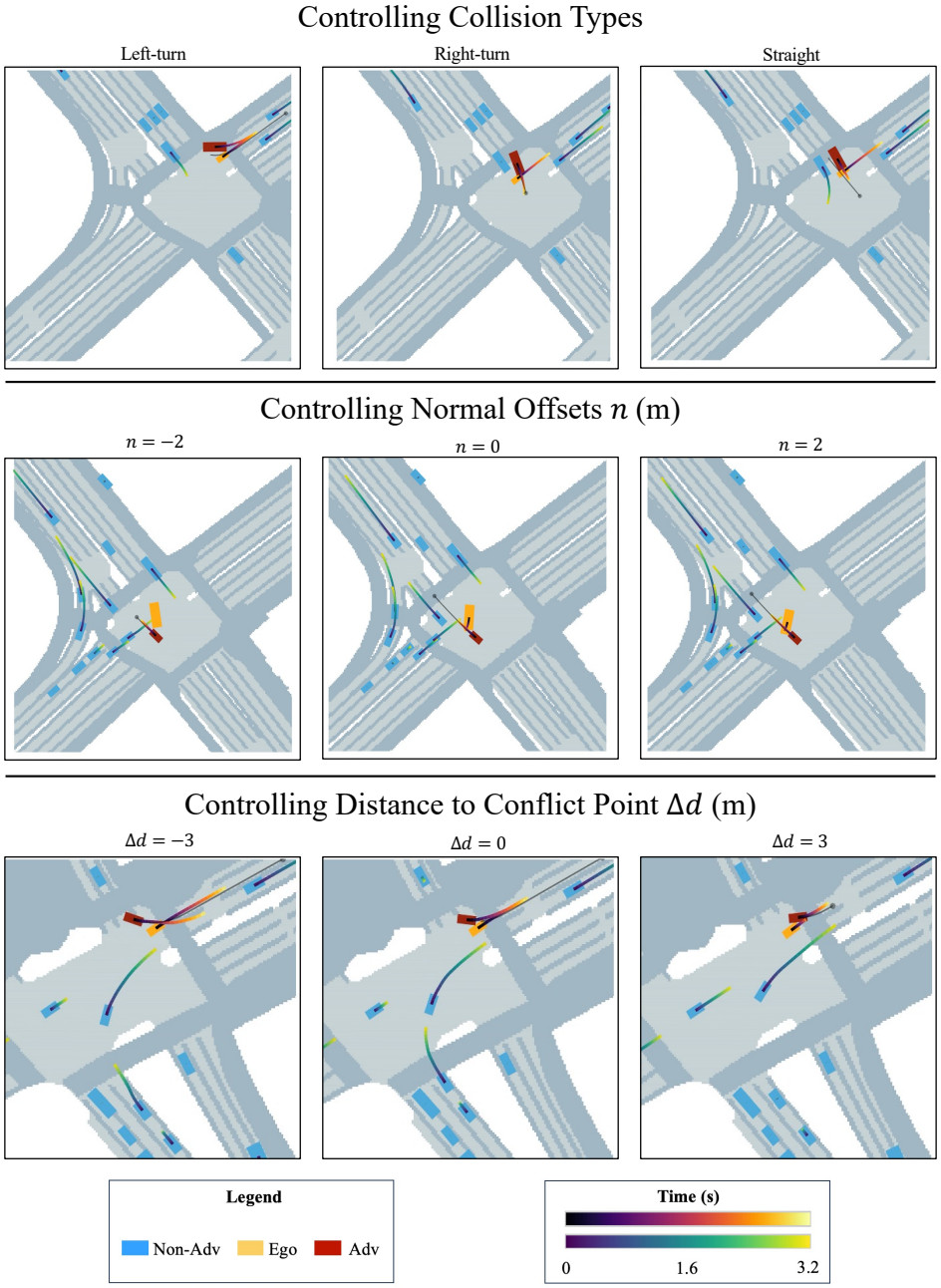

Figure A.1: **Illustration of Diverse Collision Scenarios via Partial Diffusion.** This figure showcases example simulations that highlight how varying trajectory proposals can influence the occurrence and type of collisions. The black line represents the trajectory proposals for the adversarial vehicle.

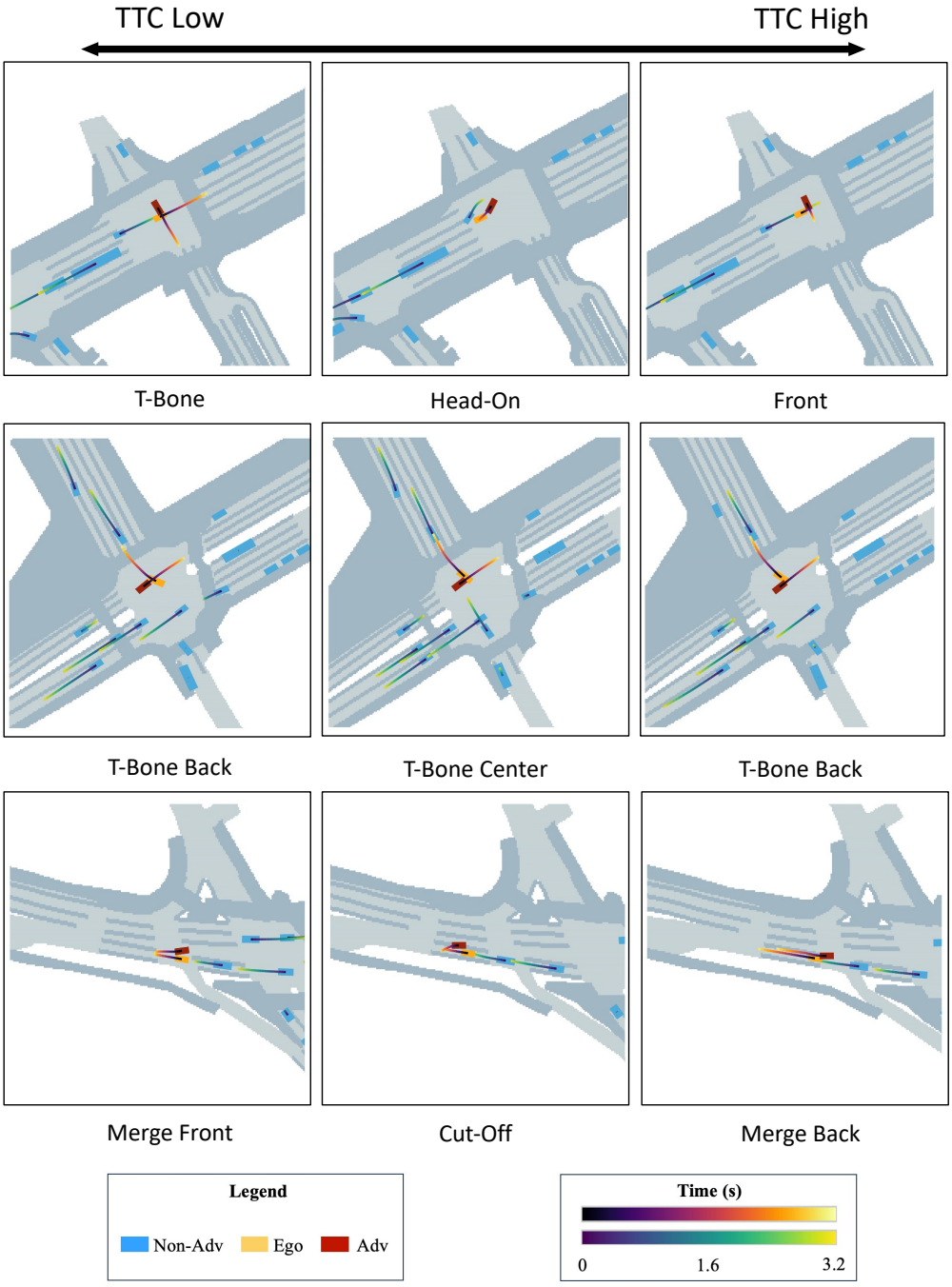

Figure A.2: **Impact of Time-To-Collision (TTC) Control on Collision Scenarios.** This figure demonstrates example simulations where adjusting the TTC parameter influences the dynamics and outcomes of collision scenarios, showcasing the method's versatility in testing autonomous driving algorithms under different conditions.

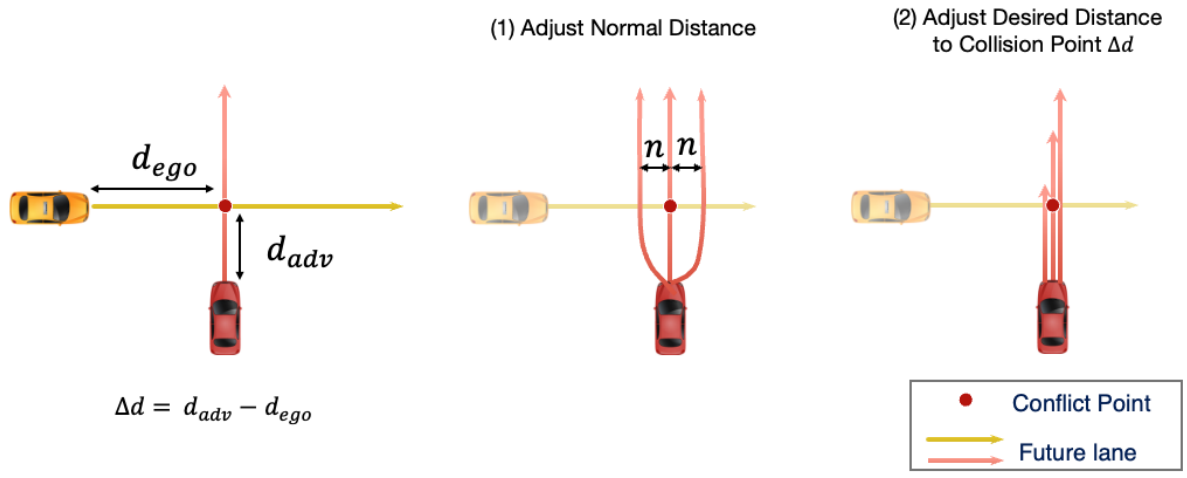

Figure A.3: **Methods for generating different trajectory proposals.**

2. **Lane's Normal Offsets:** Modifying these offsets enables the generation of trajectories that accurately reflect the spatial dynamics of vehicle positioning within lanes.

In addition, we can also generate proposals based on different lanes to have different relationships.

Note that it is essential to generate trajectory proposals within a closed-loop simulation, updated at every planning cycle. Since the diffusion model outputs action sequences, after generating the initial proposals, we employ inverse dynamics to calculate the corresponding turning rates.

A.2.2 Ablation study of Partial Diffusion

We measure the Mean Squared Error (MSE) to quantify the difference between initial trajectory proposals and the outcomes from the partial diffusion model, focusing on the first second of the trajectory in each planning iteration. Table A.1 reveals that the trajectory MSE varies with the diffusion ratio. This ratio is adjustable, enabling the calibration of the model to align with user needs and maintain a balance between the original proposals and the diffusion model's output. A partial diffusion ratio of $\gamma = 0.0$ corresponds to the highest collision rate, suggesting that our initial trajectory proposals effectively signal potential collisions. After the diffusion model's denoising step, the lateral acceleration diminishes significantly, leading to more realistic trajectory generations. This underscores the importance of our proposed partial diffusion process, highlighting its effectiveness in balancing the alignment between trajectory proposals and the model's output, which represents the underlying data distribution.

Partial Diffusion ratio γ	**Coll Rate** (%) ↑	**Adv Offroad** (%) ↓	**Traj MSE** (m^2) ↓	**Adv max lateral acc** (m/s^3) ↓
0.0	**26.8**	**7.5**	**3.09**	17.5
0.2	14.6	12.5	20.2	2.3
0.4	17.1	10.0	19.2	2.3
0.6	9.8	12.5	16.6	2.0
0.8	12.2	10.0	22.2	2.44
1.0	14.6	**7.5**	35.6	**1.75**
w/o Partial Diffusion	7.4	10.0	33.4	2.22

Table A.1: **Ablation study on the Partial Diffusion ratio.**

A.3 Metrics Definitions

This section outlines the definitions of the metrics used in our evaluations, averaged across all scenarios, except for the realism metric.

A.3.1 Traffic Simulation Metrics

Off-road. This metric measures the percentage of agents that go off-road in a given scenario. An agent is considered off-road if its centroid moves into a non-drivable area.

Collision. This metric represents the percentage of agents involved in collisions with other agents during the simulation.

Realism. Adopting the approach from [21], realism is quantified using the Wasserstein distance. This metric compares the normalized histograms of the driving profiles, focusing on the mean values of three key properties: longitudinal acceleration, lateral acceleration, and jerk. A lower value indicates a higher degree of realism.

A.3.2 Adversarial Behavior and Collision Metrics

Collision Relative Speed. Collision Relative Speed is defined as the ego planner's speed minus the adversarial vehicle's speed at the collision timestep.

To control the relative speed, we introduce the relative speed cost function:

$$J_v = \sum_{t=1}^{T} |v_t^1 - v_t^a - v_{\text{diff}}| \cdot \mathbf{1}\{d(t) < d_{\text{col}}\}, \tag{A.1}$$

where v_{diff} is the desired speed difference between the ego and the adversarial vehicles, influencing the relative speed at the point of collision. The function $\mathbf{1}\{d(t) < d_{\text{col}}\}$ is an indicator function that applies the cost only when the distance $d(t)$ between the ego and adversarial vehicle is less than a specified threshold d_{col}.

Time-to-Collision Cost. The Time to Collision (TTC) cost [70] assesses collision risk based on the relative speed and orientation between agents. For two agents located at positions (x_i, y_i) and (x_j, y_j) with respective velocities (v_{x_i}, v_{y_i}) and (v_{x_j}, v_{y_j}), we define their relative position and velocity. The relative position is given by $dx = x_i - x_j$ and $dy = y_i - y_j$, representing the positional differences along the x and y axes. Similarly, the relative velocity is calculated as $dv_x = v_{x_i} - v_{x_j}$ and $dv_y = v_{y_i} - v_{y_j}$, which are the differences in their velocities along the x and y axes. The TTC is computed under a constant velocity assumption, solving a quadratic equation to find the time of collision t_{col}, with a collision considered when relative distance is minimal.

The real part of the solution provides the time to the point of closest approach, $\tilde{t}_{\text{col}}$, calculated as:

$$\tilde{t}_{\text{col}} = \begin{cases} -\frac{dv_x dx + dv_y dy}{\tilde{dv}^2} & \text{if } \tilde{t}_{\text{col}} \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{A.2}$$

and the distance at that time, $\tilde{d}_{\text{col}}$, is given by:

$$\tilde{d}^2_{\text{col}} = \begin{cases} \frac{(dv_x dy - dv_y dx)^2}{\tilde{dv}^2} & \text{if } \tilde{t}_{\text{col}} \geq 0, \\ dx^2 + dy^2 & \text{otherwise.} \end{cases} \tag{A.3}$$

We define the TTC cost J_{ttc} as:

$$J_{\text{ttc}} = \sum_{t=1}^{T} -\exp\left(-\frac{\tilde{t}^2_{\text{col}(t)}}{2\lambda_t} - \frac{\tilde{d}^2_{\text{col}(t)}}{2\lambda_d} \right), \tag{A.4}$$

where λ_t and λ_d are the time and distance bandwidth parameters. This cost is evaluated over a time horizon T, with a higher cost for scenarios having low time to collision and proximity. For further details on the derivation of this cost function, we direct readers to [70].

In our evaluations, we focus on the average TTC cost of 0.5 seconds preceding a collision. This metric effectively captures the criticality of the safety scenarios, reflecting the potential risk of imminent collisions.

Time-to-Collision. Additionally, we compute the average Time-to-Collision (TTC) for each timestep within the crucial 0.5-second window before collisions occur in our scenarios. It's important to note that this TTC is not the actual time until a collision, but rather a theoretical estimate based on the constant velocity model assumption for each timestep.

A.4 Implementation Details

A.4.1 Diffusion Model Training and Parameterization

The training objective is to minimize the expected difference between the true initial trajectory and the one estimated by the model, formalized by the loss function [21, 68]:

$$\mathcal{L} = \mathbb{E}_{\epsilon,k,\tau_0,c}\left[\|\tau_0 - \hat{\tau}_0\|^2\right] \tag{A.5}$$

where τ_0 and $\mathbf{c}$ are sampled from the training dataset, $k \sim \mathcal{U}\{1, 2, \ldots, K\}$ is the timestep index sampled uniformly at random, and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is Gaussian noise used to perturb τ_0 to produce the noised trajectory τ_k.

In each denoising step, our model predicts the mean of the next denoised action trajectory eq. (4.3). Instead of predicting the noise ϵ that is used to corrupt the trajectory [118], we directly output the denoised clean trajectory $\hat{\tau}_0$ [21, 68]. The predicted mean based on $\hat{\tau}_0$ and τ_k:

$$\tau_{k-1} = \mu_\theta(\tau_k, \hat{\tau}_0) = \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\hat{\tau}_0 + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\tau_k \tag{A.6}$$

where β_k represents the variance from the noise schedule in the diffusion process, α_k is defined as $\alpha_k := 1 - \beta_k$, indicating the incremental noise reduction at each step, and $\bar{\alpha}_k$ is the cumulative product of α_j up to step k, mathematically expressed as $\bar{\alpha}_k = \prod_{j=0}^{k} \alpha_j$.

A.4.2 Diffusion Process Details

For the diffusion process, we utilize a cosine variance schedule as described in [30], with the number of diffusion steps set to $K = 100$. The variance scheduler parameters are configured with a lower bound β_1 of 0.0001 and an upper bound β_K of 0.05. The diffusion model takes in a 1-second history and is trained to predict the next 3.2 seconds with a step time $dt = 0.1$. Our model was trained on four NVIDIA RTX A6000 GPUs for 70000 iterations using the Adam optimizer, with a learning rate set to 1×10^{-5}. The diffusion model's implementation is based on methodologies from open-source repositories [30, 118], and the simulation framework is developed based on [5, 119].

A.4.3 Guidance Details

To simultaneously incorporate multiple guidance functions in our model, we assign weights to balance their contributions. In our experiments, particularly with non-adversarial agents, we implement a combination of route guidance (J_{route}) and Gaussian collision guidance (J_{gc}) across $M = 20$ examples. Notably, we apply a filtration process exclusively to J_{gc}, aiming to prevent imminent collisions For adversarial agents, we maintain the same weighting across all guidance functions, but uniquely control the weighting for J_{ttc} to achieve controllable behavior. In this setting, we select the sample that yields the highest adversarial cost (J_{adv}), ensuring effective and targeted adversarial scenarios.

Rel Speed Control (m/s)	**Ego-Adv Rel Speed** (m/s)	**Coll Rate** (%) ↑	**Realism** ↓
-2.0	0.90	0.29	0.83
0.0	1.26	0.38	0.89
2.0	1.94	0.44	0.88

Table A.2: **Controlling relative collision speed.** This table illustrates the ability of our framework to modulate the relative speed between ego and adversarial agents, influencing collision rates while maintaining realism.

Collision Guidance: The collision guidance is based on different agent interactions. Following the methodology of [21], we extend the denoising process of all agents within a scene into the batch dimension. During inference, to generate M samples, we proceed under the assumption that each sample corresponds to the same m-th example of the scene. For the ego vehicle, the future state predictions are derived from a diffusion model identical to the one used for other agents. The collision distance for the ego vehicle is then computed considering these predictions and their interactions with other agents within the scene.

A.5 Experimental Settings

We dynamically select adversarial agents as described in eq. (4.10), based on the criteria outlined in table 4.2. In contrast, Tables 4.4 and 4.5 use preselected and fixed adversarial agents. Additionally, Tables 4.4 and 4.5 focus on intersection scenarios where interactions are more involved.

A.6 Controllability: Controlling Relative Speed

In our safety-critical simulation framework, we examine the effects of manipulating the desired relative speed between the ego vehicle and the adversarial agent. As shown in table A.2, our proposed relative speed control results in a notable impact on both the actual ego-adversary relative speed and the collision rate. For instance, setting a lower desired relative speed target (-2.0 m/s) generally results in a decreased ego-adversary relative speed, and vice versa for a higher target (2.0 m/s). However, these adjustments do not directly translate to matching values in the simulations due to the nature of closed-loop interactions. The planner's reactive behavior to the adversarial agent's actions contributes to this discrepancy, as it may take evasive maneuvers or adjust its speed, potentially avoiding collisions altogether. Moreover, the realism metric across different relative speed settings remains relatively consistent, suggesting that the adjustments do not compromise the realism of the driving scenarios.

Method	Collision (%) ↑	Other Offroad (%) ↓	Other Collision (%) ↓	Adv Offroad (%) ↓	Collision Rel Speed (m/s) ↓	Realism ↓
Ours	**43.2**	**1.9**	1.90	**11.4**	**-0.12**	0.38
Our (-J_{route})	38.6	5.6	2.91	15.9	1.07	**0.29**
Ours (-J_{col})	25.0	4.9	**1.41**	**11.4**	0.94	0.33

Table A.3: **Ablation Study for J_{reg}.**

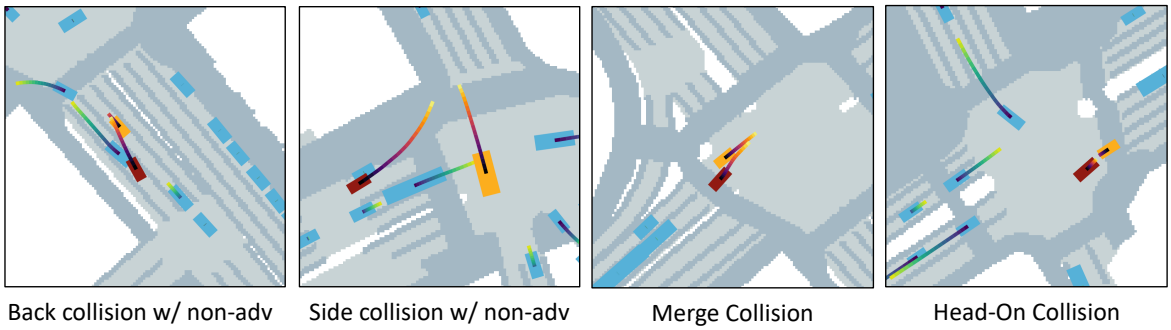

Figure A.4: **Qualitative Samples of Safe-Sim Limitation and Failure Cases.** In certain scenarios, the adversarial agent collides with non-adversarial agents before challenging the ego agent. Additionally, the adversarial agent may cause at-fault collisions.

A.7 Ablation Study for J_{reg}

We provide ablation study for the regularization term J_{reg}. Note, for table 4.5 in the main text, adversarial agents were selected before simulation based on their lane proximity to the ego. For table A.3, adversarial agents were selected dynamically during simulation via eq. (4.10).

A.8 Qualitative Analysis of Safe-Sim's Limitations

In fig. A.4, we present qualitative examples highlighting areas where Safe-Sim can be improved, including collisions with non-adversarial agents and at-fault collisions.

Appendix B

Supplementary Material for LangTraj

Supplementary videos are available on the project website.[1]

B.1 Experimental Details

B.1.1 InterDrive

The training split of InterDrive includes 100k human-annotated language-trajectory pairs and 405k pairs annotated heuristically. The test split of InterDrive contains 25k prompt-scenario pairs with both human and heuristic annotations. To address open-set language input, we augment InterDrive's categorical annotations using GPT-4 [75] to generate approximately 20 rephrasings for each annotated behavior. This augmentation expands the range of language variations the model encounters, improving its robustness to diverse user inputs.

During training, we also use *compositional* prompts that combine agent-agent interaction descriptions with heuristic action labels (e.g., *speed up*, *turn left*, *wait*) into unified instructions. This compositionality supports the flexibility of behavior expression, and we apply it consistently across both training and evaluation. Note that our instructions do not include explicit temporal conditioning, which we leave for future work.

B.1.2 ProSim-Instruct-520k

ProSim-Instruct-520k is a multimodal dataset designed for promptable traffic simulation, containing over 10 million text prompts paired with 520,000 driving scenarios. Each scenario includes goal points, route sketches, action tags describing agent behaviors, and text instructions generated by Llama3-70B. In contrast, InterDrive is directly constructed from the interactive subset of the Waymo Open Dataset, ensuring a targeted selection of interactive scenarios. Additionally, our annotations are collected from human experts rather than LLMs, focusing specifically on high-quality interactive behavior labeling.

B.1.3 LLM-Based Guidance Details

In this section, we provide implementation details for LLM-based guidance conditioning (CTG++ style [22]) method in table 5.2, which generates differentiable loss functions conditioned on text descriptions. We use the o3-mini model via OpenAI APIs to generate loss functions that guide vehicle behaviors, following the implementation. The method leverages the same backbone and weights for all experiments to ensure consistency.

Since InterDrive uses a fixed vocabulary, we generate a unique function for each interactive description and heuristic, and combine them by scaling the loss to a common range. LLM-based guidance may not work in the first iteration, as the generated functions often contain errors or inconsistencies. Common failure cases include issues with array shape mismatches, map-related functions, and the assumption of unseen functions. To address this, we manually correct the generated functions by providing more specific instructions to GPT. This process typically requires 3-5 cycles to refine the guidance functions and ensure there are no compilation errors.

B.1.4 Testing Subsets

We evaluate all experiments on a 2% subset of the data, consisting of approximately 1,100 scenarios. Specifically, InterDrive is tested on the validation interactive subset of the Waymo dataset, while ProSim-Instruct-520k is evaluated on the validation subset.

B.1.5 WOSAC Challenge Metrics

The Waymo Open Sim Agent Challenge (WOSAC) evaluates simulation quality by computing negative log-likelihood (NLL) scores over nine predefined statistical features, covering kinematics, agent interactions, and map adherence, where each feature is evaluated independently. The challenge requires simulating up to 128 agents per scene for 8 seconds, generating 32 joint agents future samples per scenario. The negative log-likelihood is then computed based on an approximate empirical distribution constructed from the simulated trajectories.

For a given scenario i and target agent a, the likelihood of the true trajectory under the empirical distribution of simulated samples is given by:

$$\mathrm{NLL}(i, a, t, j) = -\log p_{i,j,a}(F_j(x^*(i, a, t))) \tag{B.1}$$

where $p_{i,j,a}(.)$ is the empirical histogram distribution of statistic F_j obtained from the generated samples. A lower NLL indicates that the simulated distribution closely matches real-world behavior.

To obtain a per-scenario metric, the NLL values are aggregated over all valid timesteps:

$$m(a, i, j) = \exp\left(-\left[\frac{1}{N(i, a)}\sum_t v(i, a, t)\mathrm{NLL}(i, a, t, j)\right]\right) \tag{B.2}$$

where $N(i,a) = \sum_t v(i,a,t)$ represents the number of valid timesteps for target agent a. The final scenario-level metric is then computed by averaging over the target agents:

$$m(i,j) = \frac{1}{A_{\text{target}}} \sum_a m(a,i,j) \tag{B.3}$$

where A_{target} represents the number of target agents.

To adapt WOSAC for language-conditioned interactive driving, we focus on evaluating target agents that have explicit interactive descriptions in natural language on InterDrive, which is drawn from the validation interactive set, referred to as the objects of interest field in the Waymo Open Dataset format [13].

For ProsimInstruct evaluation, since the annotations are constructed from the validation set and may not contain objects of interest labels, we follow the original target agents from the validation set.

To compute the final composite metric for ranking submissions, WOSAC takes a weighted average over all component metrics:

$$M_K = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} w_j m_K(i,j), \quad \sum_{j=1}^{M} w_j = 1 \tag{B.4}$$

where $M = 9$ represents the nine statistical features, and w_j are manually assigned weights for each metric. We detail the definitions of the component metrics below:

Kinematic Metrics:

- **Linear Speed:** Measures the magnitude of the first derivative of position, $\|v\| = \left\| \frac{x_{t+1} - x_t}{\Delta t} \right\|_2$, reflecting the agent's velocity in 3D space.
- **Linear Acceleration Magnitude:** Represents the magnitude of the second derivative of position, $\frac{\|v_{t+1} - v_t\|}{\Delta t}$, describing the agent's acceleration.
- **Angular Speed:** Calculates the rate of change of the agent's heading, $\omega = \frac{d(\theta_{t+1}, \theta_t)}{\Delta t}$, where $d(\cdot)$ is the minimal angular difference on the unit circle.
- **Angular Acceleration Magnitude:** Measures the rate of change of angular speed, $\frac{d(\omega_{t+1}, \omega_t)}{\Delta t}$.

Interaction Metrics:

- **Distance to Nearest Object:** The signed distance to the nearest object in the scene, calculated using the GJK distance algorithm.
- **Collisions:** Detected when the signed distance to the nearest object becomes negative, indicating that two objects have collided.

CFG Weight	Meta ↑	Kinematic ↑	Interactive ↑	Map ↑	mADE ↓
-1.0	0.72	0.40	0.79	0.81	3.38
0.0	0.72	0.41	0.79	0.81	2.90
0.5	0.71	0.41	0.79	0.78	2.90
1.0	0.70	0.41	0.78	0.77	2.97
2.0	0.68	0.40	0.76	0.73	3.21

Table B.1: **Analysis of Text Conditioning Strength.** We evaluate the impact of text conditioning in LangTraj by varying the classifier-free guidance (CFG) weight. CFG=-1.0 represents the unconditional setting.

- **Time-to-Collision (TTC):** Estimates the time before a collision occurs, assuming constant velocities.

Map Metric:

- **Distance to Road Edge:** The signed distance to the nearest road edge in the scene.
- **Road Departure:** Indicates whether an agent has left the road at any point in time, based on the signed distance to the road edge.

For more details on each metric, refer to the original WOSAC Challenge paper [6].

B.2 Implementation Details

B.2.1 Training Details

To maximize use of the human annotations in InterDrive, we apply a biased sampling strategy to balance the training data. Specifically, we upsample human-annotated samples to represent 50% of each training batch and include 30% of heuristic descriptions in a given scene per sample. This approach allows for the simultaneous training of language-conditioned and unconditional diffusion models, optimizing both modes within the framework.

The training process for LangTraj consists of two stages. In the first stage, the scene encoder and diffusion model are trained without text conditioning for 60,000 iterations using a batch size of 32. In the second stage, the scene encoder, language encoder, and diffusion model are trained with text conditioning for an additional 20,000 iterations using a batch size of 2048. The language encoder is implemented with a LoRA module [120], which updates only the linear projection layers of the query and key matrices in DistilBERT [84], configured with $R = 16$ and $\alpha = 0.4$. Both stages employ the Adam optimizer [121] with an initial learning rate of 1×10^{-3}. The diffusion model implementation follows methodologies from open-source repositories [21, 119].

To distill our pretrained scene-diffusion model from $K = 100$ to $K = 5$, we train the model using a new denoising schedule for 20,000 iterations with a batch size of 256. The original pretrained scene-diffusion model has a prediction horizon of $T = 16$ with a frequency of 0.5 Hz.

B.2.2 Closed-loop Training Details

For closed-loop training of diffusion models, we first pre-train our scene-diffusion model with $K = 100$ denoising steps. We then distill the denoiser to $K = 5$ steps before applying closed-loop training, as described in section 5.5.2.

We modify the model's prediction horizon from $T = 16$ to $T = 8$ under 2 Hz to accommodate multi-step unrolling with ground truth actions, using a replanning interval of $T_{\text{replan}} = 2$. Given 16 steps of ground truth future trajectories, we can perform four iterations of closed-loop training. During this process, the best sample among $M = 8$ candidates is selected for execution, and the adopted forward diffusion ratio $\gamma = 0.6$.

The teacher-forcing ratio is set to 50%. When applied, 70% of agents are randomly sampled to follow the ground truth states throughout the unrolling process. The model is trained with a learning rate of 1×10^{-5} using an effective batch size of 32 for around 50,000 iterations, which takes around 12 hours on 8×A6000 GPUs and 32 CPU cores.

Additionally, we incorporate an auxiliary non-collision loss from appendix B.4.3 with a relative weighting of 0.1. To enhance robustness, we randomly drop text conditioning and agent history with a probability of 50% during training.

B.2.3 Inference Frequency

Per-sample inference with 5 de-noising steps takes 4.66±0.06 seconds per 8-second simulation (1 Hz replan) using 1×A6000 GPU and 4 CPU cores. Speedups via map caching, parallelism, and distillation can be adopted for scalability.

B.2.4 Denoising Process

At each denoising step, the model predicts the mean of the next denoised action trajectory. Instead of predicting the noise ϵ used to corrupt the trajectory, the model directly outputs the clean denoised trajectory $\hat{\tau}_0$. The predicted mean for reconstructing τ_{k-1} from τ_k is defined as:

$$\tau_{k-1} = \mu_\theta(\tau_k, \hat{\tau}_0) = \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\hat{\tau}_0 + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\tau_k, \tag{B.5}$$

Where β_k represents the variance from the noise schedule in the diffusion process, $\alpha_k = 1 - \beta_k$ denotes the incremental noise reduction at each step, and $\bar{\alpha}_k = \prod_{j=0}^{k} \alpha_j$ is the cumulative product of α_j up to step k.

B.2.5 Diffusion Process and Inference Details

For the diffusion process, we utilize a cosine variance schedule, with the number of diffusion steps set to $K = 100$ for pretrained diffusion model and $K = 5$ for the closed-loop trained diffusion models. The cosine variance scheduler followed [68], with $s = 0.008$. The model operates on a 1.1-second trajectory history and is trained to predict the next 8.0 seconds with a time step $\Delta t = 0.5$. During inference, we use a DDIM sampler [122] with a stride step 1 during inference. During inference, we sample $M = 64$ joint future samples for all agent, and only select the one joint agent sample with lowest collision loss. The **per-sample inference** time is 4.66±0.06 seconds for an 8-second simulation, with a 1 Hz replan rate. This process utilizes a 1×A6000 GPU and 4 CPU cores. To improve scalability, speedups can be achieved through techniques such as map caching, parallelism, and distillation.

B.3 Discussion and Limitation

While the model generally follows instructions well, we observe that failure cases often involve conflicts between language input and recent history (e.g., past 1s of motion). In such cases, the model tends to prioritize history, leading to behavior misaligned with the instruction. This is particularly noticeable when the vehicle is static, making conditioning signals harder to take effect. Additionally, minADE may not fully capture instruction adherence; future work could explore human evaluations to better assess language-action alignment.

B.4 Guidance Details

B.4.1 Classifier-Free Guidance

Contrary to previous diffusion models that rely on classifier guidance, direct conditioning enables control through the learned data distribution rather than predefined objectives. Classifier-free guidance leverages this distribution for generation without requiring domain-specific priors.

As shown in table B.1, we observe that moderate classifier-free guidance (weights 0.0–1.0) maintains realism and controllability, while higher weights (1.0) degrade map adherence (0.81 → 0.73) and interactive realism (0.79 → 0.76), suggesting that excessive text conditioning misaligns trajectories with the map structure. Qualitatively, while stronger guidance improves instruction-following, it also tends to produce more off-road samples.

B.4.2 Collision Guidance

We define the collision cost as

$$J_{\text{coll}} = -\sum_{t=1}^{T} d(t),$$

where d(t) is the distance between the adversarial and target agents at time step t over the planning horizon T. Minimizing J_{coll} encourages collisions. In table 5.5, one of the interactive agent from the InterDrive is designated as adversarial and the other as the target. Note that, we only compute the gradient with respect to the adversarial agent.

B.4.3 Non-Collision Guidance

To detect collisions in a differentiable way, we approximate each agent i with D equally spaced disks of radius r_i [4]. For any pair of agents (i, j), let d be the minimal distance between their respective disk centers at time t. If d is less than the sum of their radii, the circles overlap. Formally, the pairwise collision loss for agents i and j at time τ is:

$$J_{\text{pair}}\big(\tau_i,\, \tau_j\big) = \begin{cases} 1 - \dfrac{d}{r_i + r_j}, & \text{if } d \le r_i + r_j, \\ 0, & \text{otherwise.} \end{cases} \tag{B.6}$$

We sum over all agent pairs and all timesteps $t = 0, \ldots, T$ to obtain the total collision loss:

$$J_{\text{no collision}} = \frac{1}{N^2} \sum_{i \neq j} \max\Big(1,\, \sum_{\tau=0}^{T} J_{\text{pair}}\big(\tau_i,\, \tau_j\big)\Big), \tag{B.7}$$

where N is the total number of agents, T is the planning horizon, and τ_i represents the state of agent i at time τ. If no disks overlap, J_{pair} is zero; fully overlapping disks produce a maximum penalty of 1.

B.5 InterDrive Interaction Type Definitions

This section defines the detailed behaviors considered in our study and the corresponding labeling process used to categorize them.

InterDrive uses a scalable human labeling process based on multiple-choice questions organized into large categories and subcategories to define and categorize behaviors efficiently. This structured approach reduces ambiguity, provides clear guidance for annotators, and ensures consistent, high-quality labels. By leveraging this method, InterDrive achieves 2.5 times the size of the WOMD-Reasoning dataset, with higher-quality annotations and slightly lower overall labeling costs.

In addition to categorizing interaction types, annotators also identify whether a pair of agents is interacting. This process accounts for the possibility of *asymmetric interactions*, where one agent interacts with another, but the reverse may not be true. For example, Agent 2 may adjust its behavior in response to Agent 1 (e.g., yielding or avoiding), while Agent 1 may proceed unaffected, exhibiting no interaction.

- **Lane Change:** Lane change interactions involve moving from one lane to another for various purposes:
  - **Changing lane for turn or exit:** Moving to another lane in preparation for turning or exiting.
  - **Changing lane for overtaking:** Moving to another lane to pass a slower vehicle.
  - **Lane-change for avoiding obstacles or slower traffic:** Changing lanes to bypass road obstacles or slower-moving vehicles.
  - **Lane-change for merging:** Changing lanes to merge into another stream of traffic.
  - **Changing lane with lead or trail:** Performing a lane change with another vehicle directly ahead (lead) or behind (trail), requiring extra caution.
- **Following/Stopping Behind:** These interactions involve adjusting speed and distance while following or stopping behind another vehicle:
  - **Following with a lead vehicle:** Adjusting speed to maintain a safe distance while following another vehicle.
  - **Following a slow-moving lead:** Driving slower than desired to follow a slower vehicle ahead.
  - **Tailgating:** Driving too closely behind another vehicle, often considered aggressive driving.
  - **Stopping behind a lead vehicle:** Coming to a stop behind another vehicle, typically at traffic lights or stop signs.
  - **Stopping behind an intersection:** Coming to a stop before entering an intersection.
- **Yielding:** Yielding involves giving the right of way to other road users in specific situations:
  - **Intersection yielding:** Yielding to oncoming traffic or other road users at intersections.
  - **Yielding before merging or lane-change:** Yielding to ongoing traffic when changing a lane or merging.
  - **Yielding to merging or lane-change cars:** Yielding to cars that change lanes or merge into the current lane.
  - **Waiting for a pedestrian to cross:** Yielding to a pedestrian at a crosswalk or intersection, allowing them to cross safely.
  - **Roundabout yielding:** Yielding to vehicles already in a roundabout before entering.

  - **Pedestrian yielding to vehicles:** Pedestrians pause and give way to oncoming vehicles before crossing the road.

- **Passing:** Passing involves moving past vehicles, pedestrians, or obstacles without yielding:
  - **Passing through an intersection with yielding vehicles:** Moving through an intersection without yielding, while other cars yield.
  - **Passing a pedestrian:** Moving past a pedestrian walking near the road or on a crosswalk, ensuring a safe distance.
  - **Pedestrian passing a vehicle:** A pedestrian moves around or crosses in front of a stationary vehicle.
  - **Passing through a roundabout:** Navigating through a roundabout without stopping, maintaining the right of way.
  - **Maintaining speed while driving:** Driving straight or turning while maintaining the original speed without yielding or merging.
  - **Passing as a leading vehicle:** Passing with other vehicles following.
  - **Pedestrian or cyclist crossing the road:** Pedestrians or cyclists crossing the road, usually with vehicles yielding.

- **Overtaking:** Overtaking involves actively moving ahead of another vehicle:
  - **Car avoidance:** Taking evasive action to avoid another vehicle, often involving swerving, braking, or accelerating.
  - **Standard overtaking:** Passing a slower vehicle by moving to an adjacent lane and returning to the original lane.
  - **High-speed overtaking:** Passing at higher speeds on highways, requiring careful attention to speed and distance.

- **Merging:** Merging involves entering a lane of traffic from a merging lane, on-ramp, or after a lane reduction:
  - **Standard merge:** Entering the flow of traffic from a merging lane or on-ramp.
  - **Lane reduction merge:** Merging into an adjacent lane when a lane ends due to road conditions.
  - **Zipper merge:** A coordinated merge where vehicles in two lanes alternate into a single lane.
  - **Highway on-ramp accelerating merge:** Entering a highway while accelerating to match traffic speed.
  - **Late merge:** Merging closer to the end of the merging lane in congested traffic.

- **Other:**
  - **Undefined behavior:** For scenarios where a specific behavior type does not exist in the predefined options. In this case, the human labeler will type in behavior descriptions.
  - **No interaction:** For cases where no interaction occurs.
  - **Unknown status:** For scenarios where the behavior cannot be determined due to insufficient or unclear data.

Appendix C

Supplementary Material for SPACeR

Supplementary videos are available on the project website.[1]

C.1 Details on SPACeR design choices for simulating VRU

To extend SPACeR to simulating VRUs (pedestrians and cyclists), we ablate several design choices, as shown in table C.1. We use separate action heads for each agent type and include a one-hot agent-type indicator in the ego observation so the policy can specialize its behavior. We also find that the reference KL loss consistently improves VRU realism, helping reduce unstable or abrupt pedestrian behaviors. Finally, both the goal-reaching loss and the multi-action head contribute to stable and human-like VRU motion, with noticeable drops in kinematic and interaction realism when either component is removed.

For evaluation, all agents in the scene (including vehicles) are simulated, but WOSAC metrics are computed exclusively for pedestrian and cyclist targets. For training, we follow the same approach as [19], where different agent types use their own token vocabularies.

C.2 Anchoring Parameters Ablation Study

In the ablation study table C.2, KL alignment contributes most to realism. While log-likelihood rewards are popular [105], their impact on realism is modest—likely because the real-world driving distribution is highly multi-modal, so maximizing likelihood alone does not yield a stable or sufficiently discriminative signal. A second insight is that, once the policy is anchored to the reference distribution, the explicit goal reward can be removed, which further improves realism.

In fig. C.1, we visualize the training dynamics of the anchoring terms. Likelihood-only optimization steadily increases the log-likelihood but also drives entropy downward, indicating reduced diversity as training progresses. In contrast, adding KL alignment not only increases

Table C.1: Ablation study on key components of the SPACeR adapted to VRU simulation settings. Each row removes one design choice. Metrics are reported over VRU target agents only.

Ablation	**Composite ↑**	**Kinematic ↑**	**Interactive ↑**	**Map-based ↑**	**minADE ↓**
Full Model	**0.729**	**0.413**	0.762	**0.866**	**2.066**
– goal-reaching weight	0.728	0.405	**0.769**	0.859	2.295
– multi-action head	0.685	0.323	0.742	0.818	3.416
– reference KL loss	0.607	0.222	0.626	0.804	12.844

Table C.2: **Ablations on WOSAC (validation).** r_{inf} = infraction penalties (off-road, collision); LLH = log-likelihood reward; KL alignment essential for realism; goals unnecessary.

Variant	**Composite.↑**	**minADE↓**
r_{task} only	0.70	14.43
Goal + LLH	0.69	21.05
Goal + KL	0.73	**4.08**
KL + r_{inf}	**0.74**	4.73
KL + r_{inf} + LLH	**0.74**	4.68

log-likelihood but also keeps the KL term stable and preserves higher entropy, preventing collapse and supporting more reliable top-k realism.

C.3 Training Hyperparameters and Implementation Details of Self-Play

The core PPO hyperparameters are summarized in table C.3. When training without a reference model, we employ 600 parallel worlds on a single A100 (80 GB, PCIe) GPU. For human-regularized PPO and reference-model training, we reduce the number of worlds to 300, which leads to an approximately 2× slowdown due to the lack of multi-GPU support in GPUDRIVE with madrona backend. With multi-GPU support, we would expect comparable throughput to the reference-free setting. Our policy/value network is configured with an input embedding dimension of 64, a hidden dimension of 128, and a dropout rate of 0.01. To reduce GPU memory consumption in the reference-model setting, we highlight two adjustments: (1) we cap the maximum number of unique scenarios per batch at 200 to increase training speed, and (2) we limit the maximum number of map elements per agent from 200 [31] to 120.

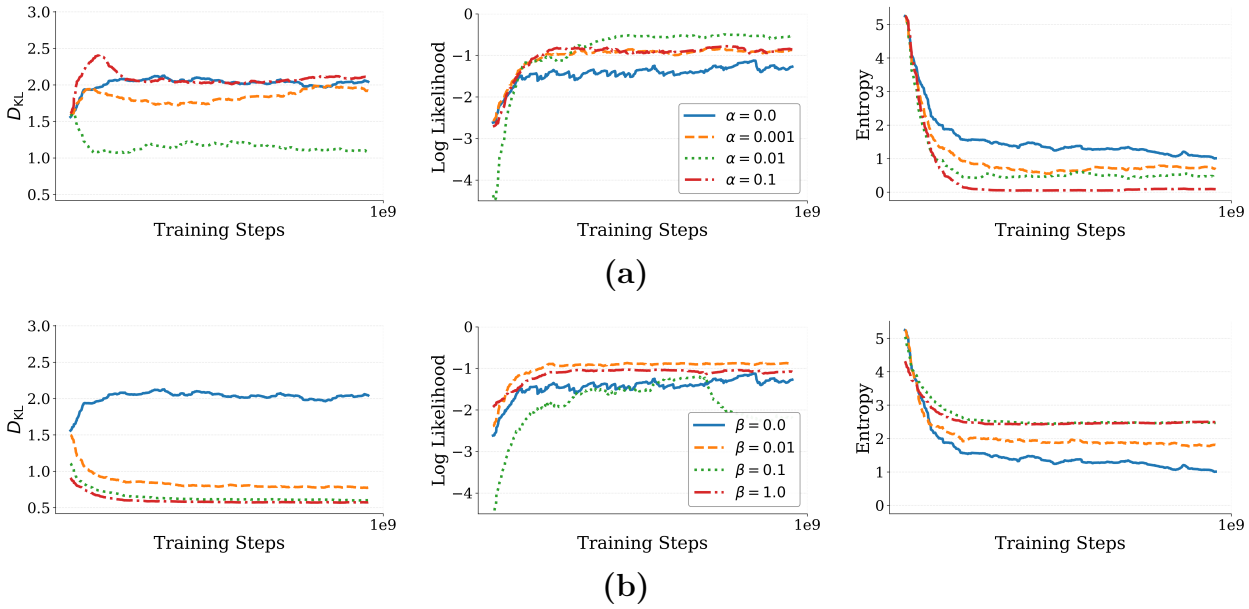

Figure C.1: **Anchoring parameters: effects of likelihood and KL-divergence weights.** (a) Likelihood weight effect without KL loss. (b) KL divergence weight sweep without likelihood loss. Likelihood-only optimization increases the probability of the most likely action but collapses entropy, while KL alignment maintains diversity and improves likelihood.

Human-Regularized PPO Baseline. We follow the setup of human-regularized PPO [33], but adapt it to our tokenized trajectory action space instead of low-level control actions. To train the behavior cloning (BC) reference policy, we use observation–action pairs from the full Waymo Open Motion Dataset (WOMD) rather than the 200-scenario subset originally used, ensuring a stronger expert baseline. Our BC model is parameterized with roughly $2\times$ the capacity of the self-play policy network to increase expressiveness, achieving a validation accuracy of 92%. We implement BC training using the `imitation` package [123] and train for 60 epochs on the full dataset.

During HR-PPO training, we regularize the learned policy against the BC reference policy using a KL-divergence penalty with weight $\beta = 0.01$. Larger values of β destabilize training, while using tokenized model can generally increase to $\beta = 1.0$ w/ similar performance. In addition, when applying human-regularized PPO in the original action space, we found it necessary to clamp the minimum reference log-probability to 10^{-20} to avoid unstable gradients; interestingly, this instability does not occur when using our tokenized trajectory reference model.

In the first submission, HR-PPO used the reverse KL divergence, the same as SPACeR. In the camera-ready version, we use the forward KL divergence and report the better-performing variant, without affecting the main conclusions.

Table C.3: PPO Training Hyperparameters.

Parameter	Value	Description
seed	42	Random seed.
total_timesteps	1,000,000,000	Total number of environment timesteps.
batch_size	131,072	Timesteps collected per rollout.
minibatch_size	8,192	Timesteps per optimization minibatch.
learning_rate	3e-4	Optimizer learning rate.
anneal_lr	false	Learning rate annealing.
gamma	0.99	Discount factor.
gae_lambda	0.95	GAE parameter λ.
update_epochs	4	Optimization epochs per rollout.
norm_adv	true	Normalize advantages.
clip_coef	0.2	PPO policy clip coefficient.
clip_vloss	false	Clip value loss.
vf_clip_coef	0.2	Value function clipping coefficient.
ent_coef	0.0001	Entropy coefficient.
vf_coef	0.3	Value loss coefficient.
max_grad_norm	0.5	Gradient clipping (max L2 norm).

C.3.1 Training Details of SMART and CatK.

We pretrain the base SMART model on 16×A100 (80 GB, PCIe) GPUs for 10 epochs and select the checkpoint with the best validation loss (5×10^{-4}). We then apply closed-loop supervised finetuning as described in [86] for 6 additional epochs with an effective batch size of 64 and a learning rate of 1×10^{-5}. The final model contains 3.2M parameters and is trained on the full training dataset of approximately 500k scenarios. To make the self-play policy more responsive while maintaining a balance between control frequency and memory usage, we use a sampling frequency of 5 Hz instead of the original 2 Hz.

C.4 Waymo Sim Agent Challenge Metrics

We follow the evaluation protocol of the Waymo Open Sim Agent Challenge (WOSAC) [93]. The full definition of WOSAC metrics, including the NLL formulation, per-agent and scenario-level aggregation, composite scoring, and individual sub-metric definitions, is provided in appendix B.1.5.

In our experiments, we restrict evaluation to vehicles only: agents corresponding to pedestrians or cyclists are excluded as targets and fixed to their ground-truth trajectories. All reported results are computed on a 2% validation subset of the dataset [86].

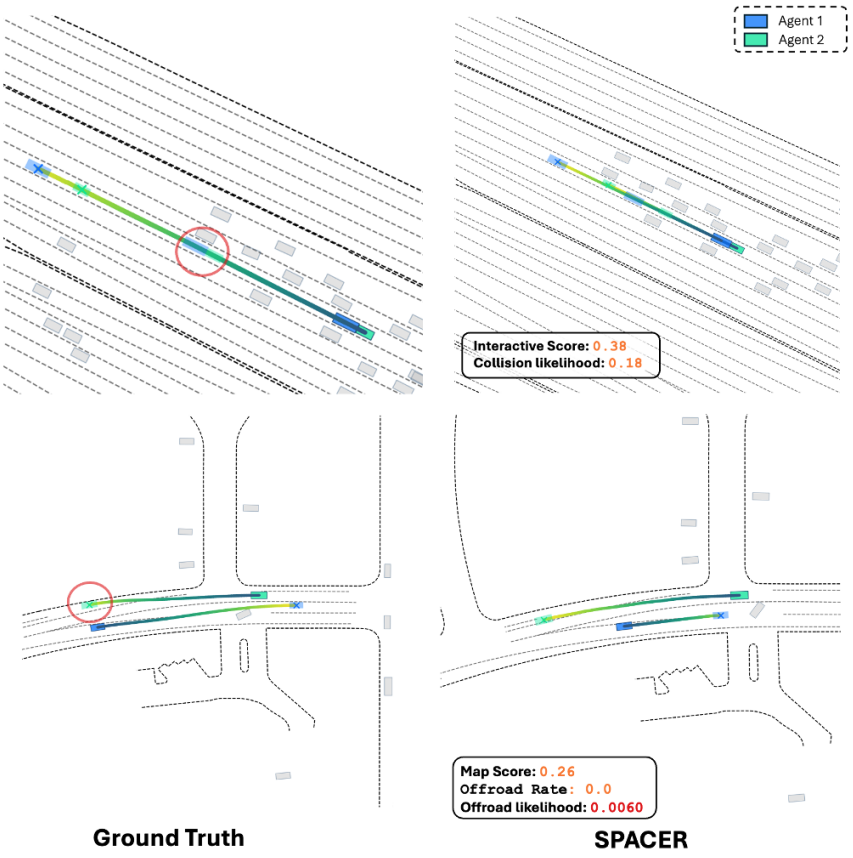

Figure C.2: **Additional qualitative examples of WOSAC limitations.** Top row: due to sensor noise, the ground-truth agents collided in the data; consequently, safe simulated behaviors are assigned low collision likelihood. Bottom row: the ground-truth agent collided with the road edge, so remaining on-road is given near-zero off-road likelihood and a low map score. These cases illustrate how WOSAC can penalize safe and realistic behaviors when they diverge from noisy or imperfect logged trajectories.

C.5 WOSAC Failure Qualitative Results

More qualitative results of WOSAC limitation provided in fig. C.2

C.6 Planner Variants Overview

Our evaluation framework covers two categories: **learning-based** and **rule-based** planners.

Learning-based. We trained 22 self-play reinforcement learning policies in GPUDrive [31], all sharing the same decentralized late-fusion MLP architecture (controlling up to 64 agents) but differing in reward weights (see Sec. A.5.1).

Rule-based. We include two families of classical planners: Frenet-based trajectory samplers and Intelligent Driver Model (IDM) planners, each with 10 parameterized variants ranging from conservative to aggressive driving styles.

C.6.1 Self-Play Policies

In addition to handcrafted planners, we trained 22 self-play policies using the GPUDrive framework [31]. All policies share the same decentralized late-fusion MLP architecture (controlling up to 64 agents), but differ in reward weighting.

We varied the weights of three components: goal-reaching ($w_{\text{goal}} \in \{1.0, 0.5\}$), collision penalties ($w_{\text{collided}} \in \{0, -0.375, -0.75, +0.1\}$), and off-road penalties ($w_{\text{offroad}} \in \{0, -0.375, -0.75, +0.1\}$). This grid covers both standard reward shaping (non-negative penalties) and unconventional settings with negative weights, where agents are encouraged to collide or go off-road.

The resulting 22 policies span a wide spectrum of behaviors, from highly conservative (strong safety penalties) to adversarial (negative penalties), allowing us to stress-test planner evaluation under diverse multi-agent conditions. All policies are trained for 1B environment steps on 10k resampled WOMD scenarios, with 600 parallel worlds and up to 64 controlled agents per rollout.

C.6.2 Frenet-based Planners

The Frenet planner uses quintic/quartic polynomials to generate trajectory candidates in the Frenet frame, evaluating them based on a weighted cost function that considers:

- Lateral deviation from centerline (w_{lateral})
- Velocity tracking (w_{velocity})
- Acceleration smoothness ($w_{\text{acceleration}}$)
- Progress along path (w_{progress})
- Jerk minimization (w_{jerk})
- Collision avoidance (collision penalty)

Table C.4 summarizes the 10 Frenet-based planner variants, showing their key characteristics including speed ranges, lateral weights, safety focus levels, and sampling densities. These variants range from conservative safety-focused configurations to aggressive high-speed optimized settings.

Table C.4: Summary of Frenet-based Planner Variants

Variant	**Description**	**Speed (m/s)**	**Lateral Weight**	**Safety Focus**	**Sampling[a] (d,v,t)**	**Key Features**
Baseline	Balanced	0-30	10.0	Medium	10,5,3	Standard configuration
Aggressive	High progress	0-35	5.0	Low	10,5,3	Progress weight = 2.0
Conservative	Safety-first	0-20	50.0	High	10,5,3	Collision penalty = 5000
Smooth Rider	Comfort	0-30	20.0	Medium	10,5,3	Jerk weight = 3.0
Lane Keeper	Centerline	0-30	100.0	Medium	15,5,3	Lateral span = 1.5m
Wide Search	Comprehensive	0-30	10.0	Medium	20,10,7	Large search space
Fast Planner	Quick	0-30	10.0	Medium	5,3,2	Reduced horizon
Long Horizon	Strategic	0-30	10.0	Medium	10,5,3	40 horizon steps
No Collision	Test baseline	0-30	10.0	None	10,5,3	Collision disabled
High Speed	Highway	5-40	10.0	Medium	10,5,3	Velocity span = 15

[a] Sampling notation: (d,v,t) represents (lateral samples, velocity samples, time samples)

C.6.3 IDM-based Planners

The IDM planner implements the Intelligent Driver Model for longitudinal control combined with a PID controller for lateral tracking. Key parameters include:

- Desired velocity (v_0)
- Minimum spacing (s_0)
- Safe time headway (T)
- Maximum acceleration (a)
- Comfortable deceleration (b)
- Aggressiveness factor (0.0-1.0)

Table C.5 presents the 10 IDM-based planner variants, each configured to represent different driving styles from cautious urban driving to aggressive highway scenarios. The aggressiveness factor plays a key role in determining the overall behavior of each variant.

Table C.6 provides a detailed comparison of the key configuration parameters for representative variants from both planner types, highlighting the differences in their weight distributions and fundamental parameters that lead to their distinct behaviors.

Table C.5: Summary of IDM-based Planner Variants

Variant	**Description**	**Desired Vel (m/s)**	**Min Gap s_0 (m)**	**Headway T (s)**	**Aggress.[b] Factor**	**Special Features**
IDM Baseline	Standard	30	2.0	1.5	0.5	Balanced behavior
IDM Conservative	Cautious	25	3.0	2.0	0.2	Safety factor = 1.5
IDM Aggressive	Dynamic	35	1.5	1.0	0.8	Safety factor = 0.9
IDM Comfort	Smooth	28	2.5	1.8	0.3	Max jerk = 2.0
IDM Highway	High-speed	40	3.0	1.2	0.6	Perception = 100m
IDM City	Urban	15	2.0	1.5	0.4	Perception = 30m
IDM Truck	Heavy	25	4.0	2.0	0.3	Length = 8.0m
IDM Emergency	Urgent	40	1.5	0.8	0.9	Max accel = 4.0
IDM Adaptive	Balanced	30	2.5	1.5	0.5	Reaction = 0.2s
IDM Defensive	Safety	25	4.0	2.5	0.1	TTC[c]= 3.0s

[b] Aggressiveness factor: Ranges from 0.0 (very conservative) to 1.0 (very aggressive)
[c] TTC: Time-to-collision threshold

Table C.6: Key Configuration Parameters Comparison

Parameter	**Baseline**	**Aggressive**	**Conservative**	**Smooth**	**Lane Keeper**
Frenet Planner Weights					
Lateral (w_l)	10.0	5.0	50.0	20.0	100.0
Velocity (w_v)	1.0	0.5	1.0	2.0	1.0
Acceleration (w_a)	1.0	1.0	3.0	5.0	1.0
Progress (w_p)	1.0	2.0	1.0	1.0	1.0
Jerk (w_j)	0.5	0.5	1.5	3.0	0.5
IDM Parameters					
Desired vel (v_0)	30.0	35.0	25.0	28.0	-
Min spacing (s_0)	2.0	1.5	3.0	2.5	-
Time headway (T)	1.5	1.0	2.0	1.8	-
Max accel (a)	2.0	3.0	1.5	1.5	-
Comfort decel (b)	3.0	4.0	2.0	2.0	-

Note: All planners use dt=0.1s time step and wheelbase=2.8m