2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

HMSim: A Hierarchical Multi-Agent Simulator For Autonomous Vehicles

Haolan Liu[1] Jishen Zhao[2] Liangjun Zhang[3]

***Abstract*— This paper addresses the challenge of developing a realistic urban-driving simulator to accurately model agent behaviors, a crucial component for self-driving car development. Most previous simulators focus on the plausibility of sensor data synthesis, whereas the plausibility of driving behaviors is poorly explored. To tackle this problem, we propose a hierarchical architecture, which comprises (i) a high-level intention simulation summarizing driving scenarios and (ii) a low-level policy trained by reinforcement algorithms to refine plans. Unlike existing simulators, our approach captures diverse behaviors, even sub-optimal ones, vital for robust policy training and evaluation. We also highlight the importance of interactive simulations over static scenarios for realistic policy development. Extensive experiments demonstrate that our approach significantly improves long-term behavior prediction and closed-loop simulation, enhancing the realism and diversity of urban-driving simulations. The videos of this work are available in our project page: https://sites.google.com/ucsd.edu/h-sim/home.**

I. INTRODUCTION

High-quality simulators with complex and diverse real-world traffic scenarios are proved valuable to self-driving car development, by facilitating scalable and efficient training and testing in the virtual world. Most prior works attempt to close the Sim2Real gap by improving the plausibility of the synthesized sensor data, including high-resolution synthesized images and point cloud [9], [18], [35], [1]. However, the plausibility of driving behaviors is poorly explored. For example, popular simulators including CARLA and MetaDrive [18], [9] deploy heuristic-based algorithms to model the behavior of traffic agents.

Our goal in this paper is to tackle a critical yet under-explored problem – How to build a realistic urban-driving simulator to model the agent behaviors? Such a behavioral-level simulator is useful in various ways. First, it can perform policy search and evaluation [17]. This is especially valuable for safety-critical domains, as experiments in real world can be dangerous and expensive. Second, by only simulating agent behaviors, it can work as a complement for high-fidelity perception simulators.

A fundamental obstacle in the development of behavioral simulators is the long-tailed nature of real-world data distributions. Existing datasets mostly comprise common and non-critical driving scenarios, while instances that are critical for safety assessments—such as near-collision events—are insufficient. This imbalance impacts the functionality of learned simulators in two aspects:

(1) *behavior diversity*: the simulators proficiently acquire major driving behaviors, such as lane-following and lane-changing, which are frequently represented in datasets. However, the rarity of complex maneuvers like U-turns and nudging in the data makes the accurate simulation challenging, as these scenarios are difficult to sample effectively.

(2) the scarcity of near-collision scenarios in the training data compromises the simulator's ability to reproduce these critical situations accurately. This limitation not only degrades the overall quality of the simulation but also introduces significant risks when testing and evaluating safety-critical driving policies.

Another challenge lies in enabling closed-loop simulations, where agents dynamically react to each other's actions. Previous simulators like TrafficSim have used generative models, specifically conditional variational autoencoders, to model multimodal future behaviors [30], [37], [34]. However, these methods are susceptible to error accumulation over long-term rollouts. While recent studies have employed techniques like closed-loop training, auxiliary losses, and trajectory perturbation to mitigate this [30], [37], [34], [3], our experiments (Section IV-D) reveal that these approaches compromise behavioral diversity, thereby reducing simulator effectiveness.

To address both challenges, we incorporate reinforcement learning (RL) algorithms with data-driven multi-agent simulations. We employ a hierarchical architecture that divides behavior simulation into two key tasks: (1) a high-level traffic controller for intention simulation, which forecasts agents' long-term goals, and (2) a low-level policy model that refines these plans using reinforcement learning (RL). Figure 1 provides an overview. Our hierarchical design allows the high-level policy (a probabilistic model) to focus on enhancing plausible and diverse long-term driving behaviors, we also propose a novel regularization term to promote diversity and reduce unrealistic behaviors. Our low-level policy is optimized for improving reactive closed-loop behaviors. It uses RL with common-sense rewards to guide agent behavior in near-collision scenarios, irrespective of data imbalances in real-world datasets. Our experiments show that our framework can generate diverse and also realistic scenarios in a closed-loop manner.

II. RELATED WORK

In this section, we discuss related works on building simulators for autonomous driving vehicles.

[1]Haolan Liu is with the University of California San Diego, hal022@ucsd.edu. Work done as an intern at Baidu Research
[2]Jishen Zhao is with the University of California San Diego, jzhao@ucsd.edu
[3]Liangjun Zhang is with Baidu Research, Sunnyvale, CA 94089, USA liangjunzhang@baidu.com

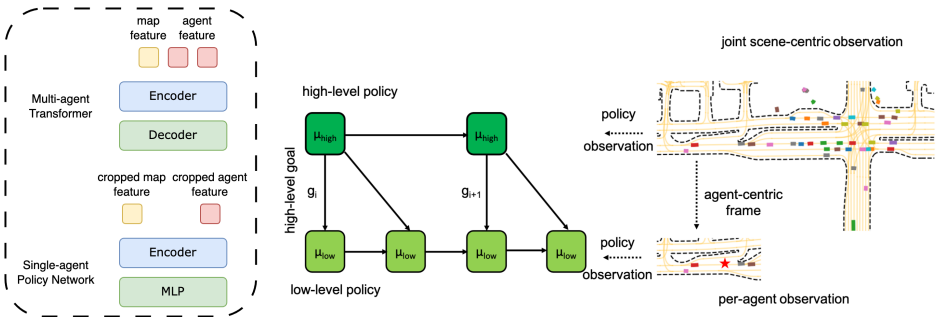

Fig. 1: Our framework has two hierarchical controllers, the high-level controller runs every K timesteps and produces goals for the low-level controller, which controls each agent per timestep. The high-level controller takes the scenario representation as input, including the map contexts and the agent history information. The high-level controller jointly infers the goal for each agents, and conditioned on that goal, the low-level controller generates actions to actuate the goal. We train the two-level networks jointly to achieve diversity and strong closed-loop performance in the same time.

a) Traffic Simulators: TrafficSim is a multi-agent behavior simulator that leverages conditional variational autoencoder to learn a joint actor policy [30], it leverages closed-loop training and auxiliary collision loss to learn consistent and plausible driving behavior.

However, to capture the diverse driving behaviors in the real world, a set of post-hoc methods [27], [36] search the latent space of a pre-trained generative model. STRIVE performs adversarial optimization in the latent space to generate safety-critical scenarios [27]. DLow is a post-hoc approach that samples diverse trajectories from a pre-trained generative model [36]. Specifically, DLow trains a differentiable latent mapping function that searches for a set with diverse trajectories.

Apart from the variational autoencoder, other generative models such as GAN are also used. Generative adversarial imitation learning and rare-event simulation are also used to generate adversarial scenarios in the highway [24]. GAN-based models are known to suffer from mode collapse, where they only generate a few reasonable modes instead of providing diverse samples [38]. In addition, GAN-based model is notoriously unstable to train and unable to provide exact or approximate likelihood, which is necessary for policy evaluation [17]. MIXSIM [31] is a hierarchical framework for mixed-reality traffic simulation that separates high-level goals, like taking an off-ramp, from low-level maneuvers, such as avoiding collisions. It uses road routes as goal representations and trains a route-conditional policy for human-like driving. This allows route-based high-level control while ensuring realistic, closed-loop interactions.

Our work stands apart from previous research by integrating reinforcement learning with a transformer-based multi-agent motion forecasting network. By decomposing the simulation task into joint goal prediction and individual agent control, we accurately model multi-agent interactions to simulate a diverse range of traffic scenarios as shown in evaluation and our website, while ensuring strong closed-loop performance—crucial for urban driving simulators.

b) Imitation Learning.: Learning the dynamics of traffic agents can also be achieved by learning an individual policy using imitation learning. SimNet proposes an end-to-end learning-based simulator that can generate reactive behaviors, compared with the old log replay approach [4]. However, most of the current imitation learning approach assumes the expert demonstration comes from a single expert [19], [20]. Such an assumption fails to present the irregularities and corner cases in the simulator, which are of importance in training and testing.

The SimNets formulated driving task as supervised learning, the driver is going to output an optimal driving plan by minimizing the distance with expert demonstration. However, such methods can produce infeasible behaviors due to mode averaging [20]. Moreover, they do not learn sub-optimal behaviors such as aggressive driving, which is necessary for simulators since they are presented in the real world, such sub-optimal behaviors are valuable for safety-critical scenario generation and testing [27].

c) Reinforcement Learning.: Reinforcement learning approaches have made notable progress in decision-making applications in recent years, including self-driving vehicles [5], [10]. SMARTS is a behavioral simulator for self-driving cars, using multi-agent reinforcement learning (MARL) [39]. However, MARL can be unstable and difficult to train, and there are no direct ways to leverage large-scale driving datasets.

III. HMSim Framework

A. Overview

We design a hierarchical simulation framework, which consists of a high-level traffic controller and a low-level policy controller, to model the problem of interactive multi-agent behavioral simulation. As illustrated in Figure 1, the high-level controller predicts a distribution of the intention $p(I)$ of the simulated agents, where I can be a reference trajectory, a goal, or a route. Given an intention I, the low-level controller adopts a policy $\pi(a|o, I)$ that produces a distribution of action a, conditioned on the current observation o and the high-level intention I. The high-level controller updates its prediction every K step and the low-level controller outputs actions between the high-level decisions.

During the training process, we pre-train the high-level controller using state-of-the-art motion forecasting works, including generative models such as CVAE or goal-conditioned methods. The only requirement is their ability to evaluate the probability or score of a given candidate intention I_i. Subsequently, we employ RL to train the low-level policy controller. Compared with previous approaches that relied on imitation learning or rule-based methods, our RL-based design enhances the agent's policy by incorporating common sense knowledge, particularly in out-of-distribution states.

B. Scenario Representation

The observation includes the agent history tensor H, in the shape of (A, T, N). A indicates the agent number, T indicates the length of past trajectory, and N indicates the feature dimension, e.g. (x, y, θ), indicating position and heading. Another important observation is the map context M, which is usually polyline features of lane centerlines or polygons [12]. Some datasets also include the masked image of drivable regions and crosswalk in the scenario [30], [27]. In our framework, we both support (1) procedurally generated maps and (2) HD maps from self-driving datasets such as nuScenes and Waymo dataset [11], [7]. We leverage the HD Map to build a lane graph $G(V, E)$, the vertice set V indicates lane segments and the edge set E indicates their connectivity.

To avoid reconstructing the observation for each agent per timestep, our framework uses a scene-centric observation, instead of an agent-centric observation [23], where the coordinate frames are centered on the predicted agents. The scene-centric frame also allows us to reuse the representation between time steps.

C. High-level Policy

We employ a single network to simultaneously predict the future states of all N simulated agents. This approach is not only computationally efficient and easy to parallelize but also captures agent interactions effectively. Specifically, We parameterize their policy as $p_\phi(i|o)$: given observation o_i in the past frames, we output future intention $I_i(i = 0, 1, .., N-1)$ of N controlled agents. The intention I_i is a four-dimensional vector (x, y, hx, hy), indicating the coordinates and heading of the agent. We train the joint policy $p_\phi(i|o)$ by maximizing its likelihood on large-scale driving datasets, which is similar to the prior work in motion forecasting [13]. For training labels, we use endpoints of the predicted trajectories as their individual intention I,

The joint policy first encodes predicted agents and map contexts as polylines, similar to VectorNet [12]. To avoid coordinate transformation in each timestep, we fix the coordinate frame in the middle of the scenario. To keep the permutation invariance, we use PointNet to extract per-polyline features.

We use the transformer [33] as the network architecture, as they achieves superior performance in self-driving motion forecasting [40], [28]. Transformers apply the attention layer that parameterizes the input as query Q, key K, and value V. The output $y = softmax(\frac{(QK^T)V}{\sqrt{dim}})$ aggregates information from the entire sequence. We use the attention layer to encode the relationship between entities in the scenario, such as agent-to-map interaction and agent-to-agent interaction. Since in the driving scenarios, the agents and map contexts can be overwhelming in numbers, the attention layer needs huge memory and computation budget to handle such long sequences. We use a factorized attention layer that alternates attention across time dimension and spatial dimension [15]. such an approach keeps the expressiveness of the original architecture while reducing the memory and computation requirements.

D. Diversifying High-level Intention

Previous research has shown that learned motion models often lack diversity and produce unrealistic samples during inference [21], [20]. These shortcomings become more pronounced when applied to multiple interacting agents in a closed-loop setting, leading to an accumulation of errors over time. Therefore, we add a regularization term for our high-level multi-agent policy training, inspired by related works in motion forecasting [41], [21].

a) Regularization: We define *positive trajectories* as feasible future driving paths, and *negative trajectories* as risky plans that should be rejected—specifically, trajectories leading to collisions. To determine the collision with other dynamic objects, we use a trajectory prediction model [28] to forecast their most likely future movements. Assuming we have two p_{pos} and p_{neg} distribution that can sample diverse positive trajectories and negative trajectories, we sample them at training time and encourages diverse intention (positive trajectories) and penalizes unlikely trajectories (negative trajectories). Formally, we list the regularization training objectives L_{reg} as following

$$L_{reg} = \gamma_1 \mathbb{E}_{X\sim\mathbb{D}, Y_{i,pos}\sim p_{pos}, Y_{i,neg}\sim p_{neg}} (\gamma_2 \log(Y_{i,neg}|X) - \log(Y_{i,pos}|X)) \tag{1}$$

X indicates scenario maps and agent histories, Y indicates trajectories we need to predict. $\mathbb{D}$ refers to the dataset. γ_1 and γ_2 is used to balance different training loss.

b) Trajectory Sampler: We build a discrete lane graph inspired as in [8], by discretizing the lane segment to lane nodes. There are two kinds of edges, the connectivity edge and the proximity edge. The connectivity edge indicates lane topology and the proximity edge covers possible lane-changing behaviors. Figure 2 shows a specific example. For more specific details, readers can refer to [8].

To build p_{pos} and p_{neg}, we randomly sample N potential goals from a precomputed goal set S_{goal} to minimize training overhead. S_{goal} can be precomputed by motion model, to reduce overhead at training time. Next, we categorize these N goals (or their corresponding trajectories) as either positive (feasible) or negative (infeasible). Specifically, we can use a simple rule-based checker (collision between the controlled agent and other agents).

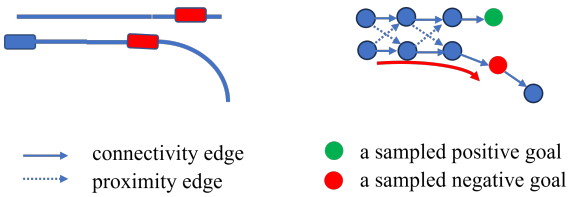

Fig. 2: This figure shows how we sample positive and negative trajectories for regularization. The left figure depicts a traffic scenario, featuring lane centerlines, other vehicles in red, and the ego vehicle in blue. The right figure presents a discretized lane graph, connecting lane nodes with directed edges. We sample target nodes to create a valid route (the red line indicates an example negative goal since it collide with the preceding car and the green indicates a positive goal). Through collision checks, we generate both positive and negative trajectories.

Subsequently, we sample positive and negative trajectories from the respective sets S_{pos} and S_{neg}. If either set is empty, the term is skipped for that training iteration.

E. Low-level Policy

Since our high-level policy only runs every K timesteps, we can support better simulation performance. Another benefit is that we can separately train a low-level policy at arbitrary time intervals, while the high-level policy is limited to the dataset sampling rate.

a) Policy Network: The low-level policy concentrates solely on maps and objects within a radius of R. This restricted receptive field is feasible because the high-level policy has already generated a viable intention. To compute a fixed scene embedding, the low-level policy employs a PointNet layer [25]. It then uses a Multi-Layer Perceptron (MLP) network to output a control vector. We first pretrain our policy network conditioned on the predicted goals of the high-level network on the large-scale datasets, then we fine-tune it through reinforcement learning. We also batch their inputs to a single inference to improve the GPU utilization.

b) Reward Function: Designing a reward function $R(s, a)$ for driving task is very challenging [18], since we have to consider factors including avoiding collisions, following rules, making progress, and driving smoothly. But for our low-level controller, we only care about reactive behaviors to avoid collision, and also reaching the goal provided by the high-level controller. To this end, we design a collision reward:

$$R_{collision} = -\mu_{collision}\mathbb{1}(d_{min} <= 0) \tag{2}$$

The $d_{collision}$ is the distance between the controlled agent and the nearest objects. We perform collision checks in each time step. The goal reward is the distance between the goal observation and the current observation

$$R_{goal} = \mu_{goal} d(g_d, g_o) \tag{3}$$

Reinforcement learning is known to introduce unnatural motion [22], we also introduce a motion reward to regularize the trained policy, introducing a penalty for drastic changes in actions, leading to jerky motion.

$$R_{motion} = -\mu_{motion} d(a_i, a_{i-1}) \tag{4}$$

The total reward sums up the above reward.

c) Vehicle Dynamics: To guarantee physical plausibility, we update each vehicle's state with a kinematic bicycle dynamics model [26]. Each controlled agent will produce a 2-dimensional action (a_{accel}, a_{steer}), and the bicycle model will compute its state vector in each time step.

F. Training Low-level Policies

Reinforcement learning (RL) policies, π, map observation o to action a. RL optimizes the policy function π^* to maximize the expected cumulative reward, $J(\pi_\theta)$. Deep RL methods use a deep neural network (DNNs) and optimize θ. To train our policy, we use the soft actor-critic framework (SAC) [14] with hindsight experience replay (HER) [2] to overcome the sparse reward problem.

We build our environment based on scenarios in large-scale driving datasets such as Waymo dataset [11]. Other agents in the scenario just follow the trajectory of log replay. However, in the real world, other traffic agents will act differently to different ego vehicle behavior, such a discrepancy can lead to suboptimal policy. As such, when training our low-level policies, we identify the vehicle that has to yield to the controlled agents, which is the vehicle behind the controlled agents, and also in the same lane or neighboring lane. We use a rule-based planner (Intelligent Driver Model) [32] to generate replanned trajectories for the impacted agents. One possible improvement is to identify the reactor&influencer relationship between the controlled agents and all vehicles in the scenarios as in [29].

IV. Experiments

We compare our system with several baselines for the learning-based autonomy simulators, including (1) Rule-based methods are most widely used in previous simulators [9], [18], since they are highly interpretable and easily perform reasonable behaviors. However, rule-based agents require heavy manual tuning to exhibit human-like behaviors, thus not automatic enough like the data-driven approach. We use the Intelligent Driver Model (**IDM**) [32] as the rule-based baseline here. (2) Imitation learning agents, we indicate them as **BC** (behavior cloning) [4] and **IL** [3], the difference is whether we apply closed-loop training on the policy network [30] (3) hierarchical policy, we measured the MixSim [31] (**Hierarchical IL**) whose low-level policy is imitation learning agents.

A. Datasets

We train and evaluate our multi-agent simulator using the Waymo datasets [11] have over 100,000 scenarios. Each scenario lasts 20 seconds and is captured at a frequency of 10 Hz. This extensive dataset was curated by seeking out noteworthy interactions among vehicles, pedestrians, and cyclists across six different cities within the United States. We use 8000 scenarios as the validation set.

	Prediction Accuracy		Diversity	Scene Plausibility			Motion Plausibility	
	minADE↓ (m)	minFDE↓ (m)	MASD↑ (m)	Collision Rate↓ (%)	Off Road↓ (%)	Traffic Light Violation↓ (%)	Speed↓ (JSD)	Acceleration↓ (JSD)
Log Replay	-	-	-	0.0	0.0	0.0	-	-
IDM[32]	-	-	**3.24**	7.7	0.0	0.0	0.17	0.21
BC[4]	-	-	1.29	13.6	9.4	14.9	0.15	0.18
IL[3]	-	-	0.87	11.3	9.2	14.5	0.13	0.17
TrafficSim[30]	1.86	2.98	2.13	10.3	8.1	9.2	0.11	**0.10**
Hierarchical IL[31]	1.56	2.86	2.45	4.2	2.0	3.4	0.10	0.13
Our work	**1.33**	**2.65**	2.51	**3.7**	1.1	**0.0**	**0.09**	0.13

TABLE I: The comparison on prediction accuracy, diversity, scene plausibility, and motion plausibility between our work and previous works, evaluated on Waymo Open Dataset. Our work presents best results in all metrics in prediction accuracy, scene plausibility and the speed metric in motion plausibility. We achieve the second-best result in diversity (MASD metric) and the acceleration metric in motion plausibility.

	Closed Loop Training	Aux Loss	Perturb.	Reg. (Sec.III-D)	MASD	Success Rate
M0					2.62	0.72
M1	✓				3.58	0.94
M2		✓			3.23	0.77
M3			✓		3.36	0.91
M4				✓	2.33	0.74

TABLE II: The comparison of prior training techniques closed-loop training, auxiliary loss (Aux Loss), and our regularization term (Reg.) as discussed in III-D. The table shows the diversity and simulation safety result, evaluated on Waymo Open Dataset. M0 is a vanilla baseline that uses none of those training techniques. M1-M4 uses the individual training technique with the mark.

B. Metrics

Prediction Accuracy: The **minADE** metric calculates the Euclidean distance between the predicted trajectory and the ground truth trajectory across all time steps [30]. For probabilistic models, we sample K trajectories and select the trajectory with the minimum ADE. A lower minADE indicates a more accurate prediction. **minFDE** focuses solely on the error at the final time step.

Diversity: Following [30], we employ a map-aware average self distance (MASD) metric to quantify the diversity of the scenarios generated. Specifically, this metric calculates the average distance between the two most dissimilar sampled scenarios that adhere to traffic regulations. It is important to acknowledge, however, that this metric may be susceptible to manipulation by models that produce a wide range of scenarios, including those that are diverse yet lack realism in terms of traffic behavior.

Scene Plausibility: We also compute some common sense metrics such as **collision rate**, **off-road rate**, **traffic light violation**. To compute the motion plausibility, we compute the **Distribution JSD** (Jensen-Shannon Divergence), based on the distribution histogram of agents' speed and acceleration, as in [31], [16]. Those metrics can measure the plausibility of our generated scenario in various aspects.

C. Analysis & Visualization

Table I presents our results on the Waymo Datasets [11]. Our approach surpasses all baselines in almost every metric (except diversity term compared with **IDM** agents), demonstrating its capability to produce realistic scenarios. Since **IDM** agents can only take predefined actions (e.g., accelerate, lane changing), its MASD is usually higher than learning-based methods [30]. We note that our approach has substantial advantages over **hierachical-IL**, which proves the effectiveness of training low-level policies using reinforcement learning.

We also visualize the generated scenario in Figure 3, which depicts three scenarios spanning 8 seconds. We only re-simulate the green vehicle and the blue vehicle uses log-replay trajectories. The first scenario features controlled agents navigating an intersection, with options to turn either left or right. The second scenario presents an agent entering traffic from a driveway and safely interacting with other vehicles. The third scenario involves a multi-lane setting where agents are trained to change lanes as needed. From the figure, we can see that our framework is able to learn social interactions, follow traffic rules, and avoid collisions. We also draw the velocity and acceleration of each controlled agent with regard to time, which is shown in Figure 4.

D. Ablation Study

For better closed-loop performance, previous work proposes some training techniques such as closed-loop training that unrolls the predicted trajectory autoregressively (decodes trajectories one step at a time and feeds the generated waypoints as the input of the next step inference) for a few timesteps and computes the loss through time [30], [27]. Such techniques mitigate the error accumulation in closed-loop rollouts, at the expense of longer training time.

To understand those design choices and their implication on sampling behavior quality and closed-loop performance, we study closed-loop training (**BPTT**) [27], [30], trajectory perturbation (**perturb.**) [3], auxiliary loss (**aux loss**) [27], [30], and also our proposed regularization term (**reg.**) in Section III-D.

We compare the performance with/without those design choices. Our evaluation uses a manually-curated set of 50 diverse scenarios, featuring maneuvers like U-turns, lane

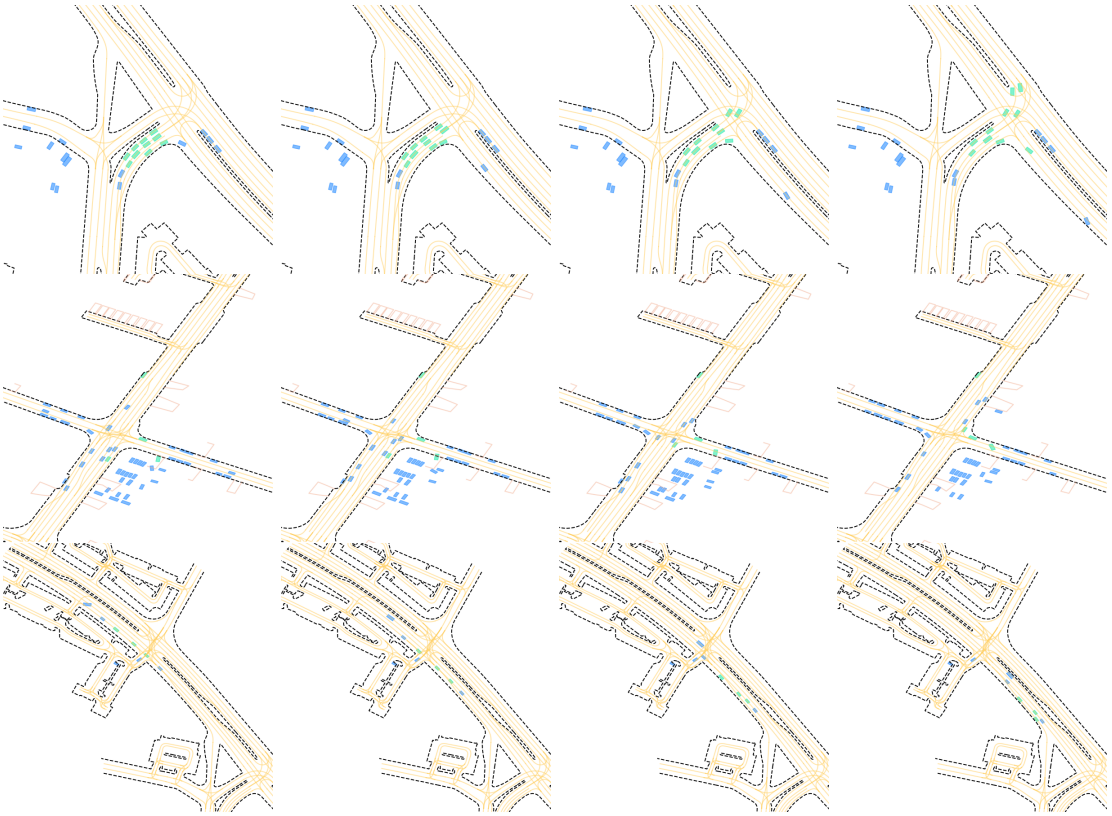

Fig. 3: We unroll our learned policy based on the history (1 second for Waymo Datasets) of existing scenarios. The green vehicles indicate the controlled agents, blue vehicles indicate other agents controlled by the log replay. We visualize the scenario in the 2nd, 4th, 6th, and 8th seconds, including an intersection scenario, yield negotiation scenario, and a multilane lane-changing scenario.

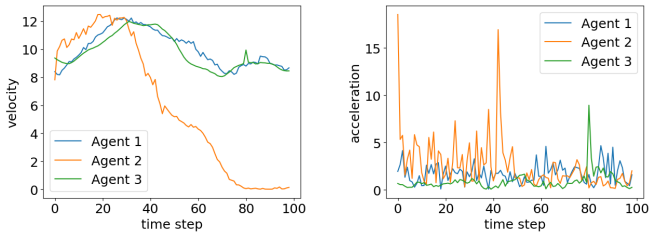

Fig. 4: The velocity and acceleration profile of controlled agents in our simulated scenario (the 3-rd scenario in Figure 3). Our framework is able to provide plausible smooth velocity profile.

changes, nudging, and cutting-in. We measure the **minADE** by sampling 20 trajectories, highlighting our model's behavioral diversity (We opt for 20 samples because these behaviors are less likely to be sampled due to their low probability). Additionally, we run each policy for 10 seconds in a closed-loop setting and calculate the **success rate**, considering a rollout successful if it avoids collisions and curb-hitting. The results are presented in Table II. We observe that while prior work's design choices improve closed-loop performance, they compromise the learning of less frequent behaviors. In contrast, our approach maintains behavioral diversity by delegating closed-loop performance to the low-level policy.

E. Application

We also use our framework to build a behavioral simulator that integrates OpenAI gym [6] and train ego vehicle policy with those learning-based agents, leveraging reinforcement learning. Trained in such environments, the agents learn reasonable driving behaviors and even good social interaction with other agents, demonstrating our methods' efficacy. The video is shown in this link.

V. Conclusion

In summary, this paper presents a novel hierarchical architecture for simulating realistic urban-driving scenarios. By separating high-level intentions from low-level maneuvers, our approach addresses the limitations of existing simulators and data-driven methods. Utilizing reinforcement learning, we enhance the diversity and realism of agent behaviors, particularly in safety-critical situations. Our work not only improves the fidelity of virtual training environments but also serves as a complementary tool for high-fidelity perception simulators, thereby advancing the state-of-the-art in autonomous driving simulation.

REFERENCES

[1] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426, 2021.

[2] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *ArXiv*, abs/1707.01495, 2017.

[3] M. Bansal, A. Krizhevsky, and A. S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *Robotics: Science and Systems*, 2019.

[4] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osiński, H. Grimmett, and P. Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5119–5125, 2021.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[7] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[8] N. Deo, E. Wolff, and O. Beijbom. Multimodal trajectory prediction conditioned on lane-graph traversals. In *5th Annual Conference on Robot Learning*, 2021.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[10] R. Emuna, A. Borowsky, and A. Biess. Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars. *CoRR*, abs/2006.04218, 2020.

[11] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset. *arXiv*, 2021.

[12] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] J. Gu, C. Sun, and H. Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15283–15292, 2021.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290, 2018.

[15] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans. Axial attention in multidimensional transformers, 2019.

[16] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, page 2445–2451. IEEE Press, 2022.

[17] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray. *Algorithms for decision making*. MIT press, 2022.

[18] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[19] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] H. Liu, J. Zhao, and L. Zhang. Interpretable and flexible target-conditioned neural planners for autonomous vehicles. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10076–10082, 2023.

[21] Y. Liu, Q. Yan, and A. Alahi. Social nce: Contrastive learning of socially-aware motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15118–15129, October 2021.

[22] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko. Regularizing action policies for smooth control with reinforcement learning. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1810–1816, 2020.

[23] J. Ngiam, B. Caine, V. Vasudevan, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, D. J. Weiss, B. Sapp, Z. Chen, and J. Shlens. Scene transformer: A unified multi-task model for behavior prediction and planning. *ArXiv*, abs/2106.08417, 2021.

[24] M. O' Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[26] R. Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[27] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022.

[29] Q. Sun, X. Huang, J. Gu, B. Williams, and H. Zhao. M2I: From factored marginal trajectory prediction to interactive prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[30] S. Suo, S. Regalado, S. Casas, and R. Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10395–10404, 2021.

[31] S. Suo, K. Wong, J. Xu, J. Tu, A. Cui, S. Casas, and R. Urtasun. Mixsim: A hierarchical framework for mixed reality traffic simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9622–9631, June 2023.

[32] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[34] Y. Wang, T. Zhao, and F. Yi. Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023, 2023.

[35] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W. Ma, A. J. Yang, and R. Urtasun. Unisim: A neural closed-loop sensor simulator. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 1389–1399. IEEE, 2023.

[36] Y. Yuan and K. Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[37] Y. Yuan, X. Weng, Y. Ou, and K. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[38] Z. Zhang, M. Li, and J. Yu. On the convergence and mode collapse of gan. In *SIGGRAPH Asia 2018 Technical Briefs*, SA '18, New York, NY, USA, 2018. Association for Computing Machinery.

[39] M. Zhou, J. Luo, J. Villela, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, A. C. Huang, Y. Wen, K. Hassanzadeh, D. Graves, D. Chen, Z. Zhu, N. M. Nguyen, M. Elsayed, K. Shao, S. Ahilan, B. Zhang, J. Wu, Z. Fu, K. Rezaee, P. Yadmellat, M. Rohani, N. P. Nieves, Y. Ni, S. Banijamali, A. I. Cowen-Rivers, Z. Tian, D. Palenicek, H. Bou-Ammar, H. Zhang, W. Liu, J. Hao, and J. Wang. SMARTS: scalable multi-agent reinforcement learning training school for autonomous driving. *CoRR*, abs/2010.09776, 2020.

[40] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[41] D. Zhu, M. Zahran, L. E. Li, and M. Elhoseiny. Motion forecasting with unlikelihood training in continuous space. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1003–1012. PMLR, 08–11 Nov 2022.