Long-term Traffic Simulation with Interleaved Autoregressive Motion and Scenario Generation

Xiuyu Yang* Shuhan Tan* Philipp Krähenbühl
UT Austin

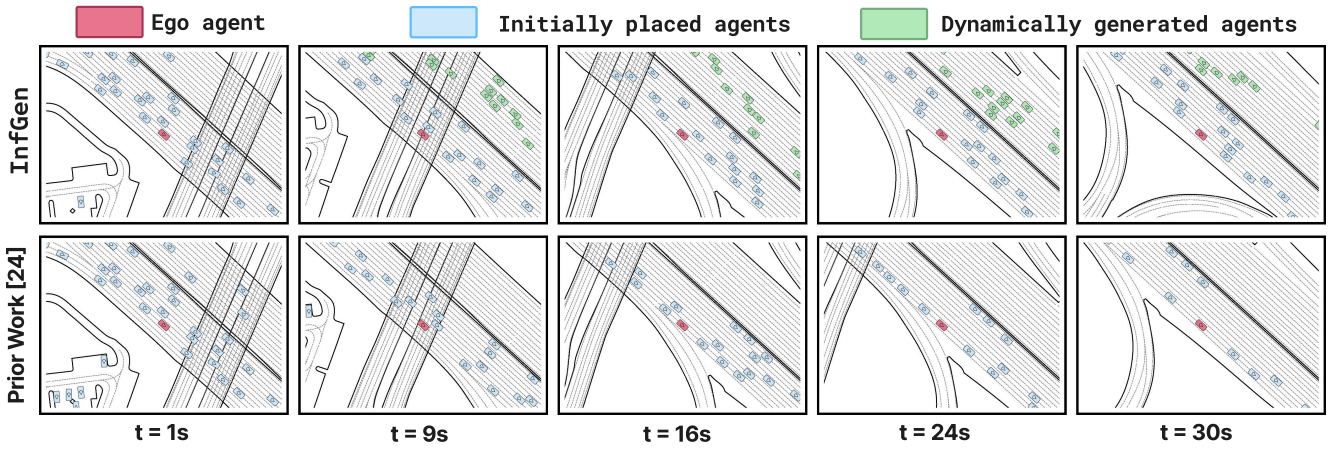

Figure 1. Long-term traffic simulation with InfGen and prior SOTA [28]. InfGen keeps scene layout realistic while [28] becomes empty.

Abstract

An ideal traffic simulator replicates the realistic long-term point-to-point trip that a self-driving system experiences during deployment. Prior models and benchmarks focus on closed-loop motion simulation for initial agents in a scene. This is problematic for long-term simulation. Agents enter and exit the scene as the ego vehicle enters new regions. We propose InfGen*, a unified next-token prediction model that performs interleaved closed-loop motion simulation and scene generation.* InfGen *automatically switches between closed-loop motion simulation and scene generation mode. It enables stable long-term rollout simulation.* InfGen *performs at the state-of-the-art in short-term (9s) traffic simulation, and significantly outperforms all other methods in long-term (30s) simulation. The code and model of* InfGen *will be released at* https://orangesodahub.github.io/InfGen.

1. Introduction

Traffic simulation is a cornerstone of the extensive and safe development of self-driving systems. The ultimate goal of traffic simulation is to create realistic trip-level driving experiences that faithfully reflect real-world self-driving conditions [1, 6, 8, 12, 27]. A simulator should provide a realistic model of the environment, the ego-vehicle, and all other traffic agents throughout the trip. Existing simulators easily handle an expansive static environment [1, 8, 27] and intricate ego-vehicle dynamics [6, 12]. However, they often lack a stable long-term simulation of non-ego traffic agents.

In this paper, we introduce InfGen, a long-term traffic simulator: Given a short (1 second) driving-log, InfGen simulates realistic traffic flow around the ego agent for up to 30 seconds. This long-term setting leads to new challenges. The ego agent may move outside its initial simulation area, leading to logged agents moving out of sight and becoming irrelevant. Furthermore, when the ego agents drive into new map area not covered in the log, these street areas have no agents. Gradually, a scenario becomes sparser and eventually empty (Fig. 1 bottom row). This is clearly unrealistic. InfGen models this by combining closed-loop *motion simulation* with *scene generation* to remove exiting agents and spawn new agents according to the spatial scene layout.

InfGen (Fig. 2) is a unified autoregressive transformer with interleaved token prediction. It handles *temporal* motion simulation and *spatial* scene generation in a unified model. We design a set of tokenizers to convert task-specific behaviors of motion simulation and scenario generation into discrete tokens. We then add mode-control tokens to mark

*Equal contribution. Work done when Xiuyu as an intern at UT Austin.

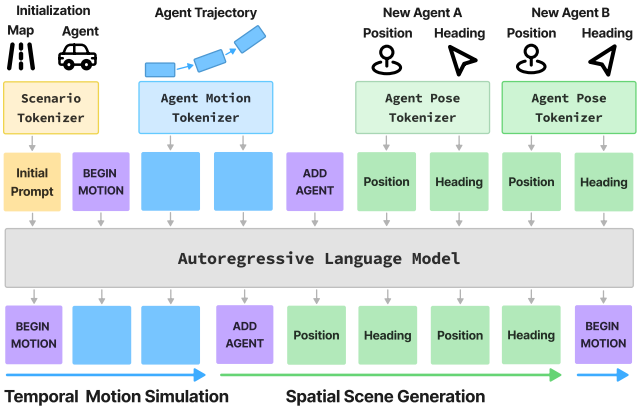

Figure 2. Overview of InfGen interleaved next-token-prediction process. Colors mark different token modalities.

the task switch between the two tasks, indicating what the current task is and when to switch. This design allows us to convert each real log into a single ordered sequence of tokens containing interleaved data of both tasks. We directly train InfGen with the next token prediction objective end-to-end on real data. Noticeably, thanks to the next token prediction formulation, we can train InfGen on short-term driving logs and produce stable long-term rollouts, up to $6\times$ longer than the training horizon (Fig. 1 top). We show a detailed pipeline of InfGen in Fig. 3.

We show in Section 5 that InfGen significantly outperforms prior SOTA models [28, 38] in 30-second long-term traffic simulation for in terms of both motion and scenario realism, showing its strong capacity for stable long-horizon rollout. We contrast different models' rollout with visualizations (Fig. 5). Furthermore, InfGen achieves strong performance even on the standard short-term Sim Agent setting [13], showing its strong adaptivity to different rollout horizons. In addition, we provide a comprehensive set of analysis on the long-term traffic simulation task to shed light on research towards trip-level driving simulators.

2. Related Work

Closed-loop Motion Simulation. In traffic simulators, motion simulation aims to model realistic multi-agent interactions that mimic real-world data. Early works like Wayformer [14] focus on open-loop imitation learning from real logs [19, 33, 35, 37, 40]. Other works like ProSim [24] and CAT-K [38] instead focus on modeling closed-loop interaction between agents [2, 21, 36]. CtRL-Sim [17] learns reactive agents by applying offline reinforcement learning to diverse traffic scenarios. More recent works like SMART [28] discover the effectiveness of modeling this task as an autoregressive next-token-prediction problem [10, 15, 20, 39, 43]. ProSim [24] enables multimodal prompts to control behavior semantics of any agent. Most recently, GIGAFLOW [6] shows strong agent performance emerges from large-scale self-play in simulation. For this direction, nuPlan [1] and WOSAC [13] provide data and benchmark for fair comparisons. All these works focus on simulating motions of agents existed in the history, leading to unrealistic scene layouts under long-term rollout. InfGen solves this issue with interleaved scene generation, maintaining realistic scene layout across the rollout horizon.

Traffic Scenario Generation. This line of work focuses on generation realistic and interesting traffic scenarios. Early works like SceneGen [22] and TrafficGen [9] generates agent initial poses on an empty map. Another popular direction is to generate near-collision scenarios with adversarial optimization [4, 16, 26, 30, 32]. More recent works like LCTGen [23] enables better customizations of the generated scenarios in forms of text [23, 41], scenario queries [7] or cost functions [42]. SLEDGE [5] combine generative models with rule-based traffic simulation to synthesize dynamic driving scenarios. These works either focus on static scene layout initialization, or only generate short-term open-loop scenarios. In contract, InfGen conducts dynamic scenario layout generation during closed-loop rollout, enabling stable long-term simulation. Some concurrent works like ScenarioDreamer [18] introduces a vectorized latent diffusion approach to generate realistic and diverse driving simulation environments.

Interleaved Next-Token Prediction. Recent advances in vision-language models have sparked works that unify generation and understanding tasks with interleaved mixed-modal token sequences. For example, Chameleon [3] proposes to train LLMs on interleaved text and image tokens in any arbitrary sequences. This line of works show strong performance on traditional multimodal tasks, but also long-form mixed modal generation that interleaves between image and text generation [3, 11, 25, 29, 34]. InfGen follows the same philosophy but focus on a different pair of modalities: temporal agent motion and spatial agent layout.

3. Problem Formulation

The goal of traditional traffic simulation [13] is to predict future agent trajectories given historical observations (with time span T_H) and a static map. Specifically, at timestep t_0, we are given the static map $\mathcal{M}$ and the history states of all the agents $\mathcal{A}_{0:t_0} = \{a^1_{0:t_0}, a^2_{0:t_0}, \dots, a^N_{0:t_0}\}$, where each agent a^i has history states up to timestep t_0. The standard task is to predict future agent states over a fixed horizon T, formulated as estimating the conditional distribution: $p(\mathcal{A}_{t_0+1:T} \mid \mathcal{M}, \mathcal{A}_{0:t_0})$. Prior work [24, 28] factorize the simulation on the time axis and turn it to an autoregres-

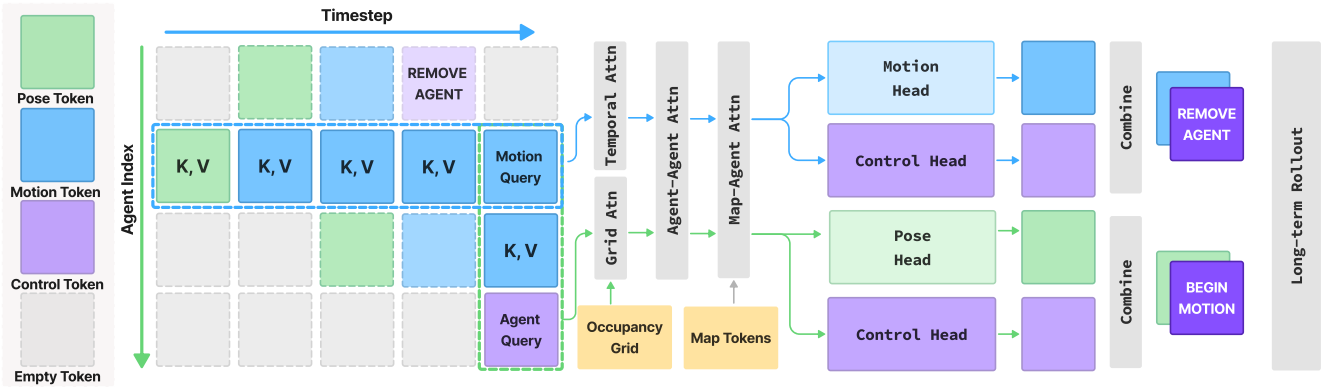

Figure 3. Pipeline of `InfGen` interleaved motion simulation (blue flow) and scene generation (green flow). For either task, we first pass its query feature through blocks of attention layers and feed it to a task-specific head and a control head. We then sample from both heads to obtain a motion token or pose token, as well as a control token, which determines determines which task to execute next.

sive prediction task:

$$p(\mathcal{A}_{t_0+1:T} \mid \mathcal{M}, \mathcal{A}_{0:t_0}) = \prod_{t=t_0}^{T-1} p(\mathcal{A}_{t+1} \mid \mathcal{M}, \mathcal{A}_{0:t}). \quad (1)$$

However, this formulation assumes a fixed set of agents throughout the prediction horizon, which does not hold true for long-term simulations with the horizon $T'(\gg T)$. In realistic scenarios (*e.g.*, [13]), agents dynamically enter and leave the observable region around the ego vehicle. To address this issue, we model two interleaved processes at each timestep: 1) motion simulation: predicts future motions for existing agents; 2) scene generation: dynamically inserting new agents and removing agents exiting the scenario. At each timestep t, we first performs motion simulation: $\mathcal{A}_{t+1} \sim p_{\text{motion}}(\mathcal{A}_{t+1} \mid \mathcal{M}, \mathcal{A}'_{0:t})$, followed by scene generation: $\mathcal{A}'_{t+1} \sim p_{\text{scene}}(\mathcal{A}'_{t+1} \mid \mathcal{M}, \mathcal{A}_{t+1})$. Here, $\mathcal{A}_{t+1}$ contains the predicted motions of existing agents, while $\mathcal{A}'_{t+1}$ represents the updated set of agents after adding new agents and removing exiting agents. Then, we formulate the *long-term traffic simulation* task as:

$$p(\mathcal{A}'_{t+1:T'} \mid \mathcal{M}, \mathcal{A}_{0:t_0}) = \prod_{t=t_0}^{T'-1} p_{\text{scene}}(\mathcal{A}'_{t+1} \mid \mathcal{M}, \mathcal{A}_{t+1}) \times p_{\text{motion}}(\mathcal{A}_{t+1} \mid \mathcal{M}, \mathcal{A}'_{0:t}). \quad (2)$$

4. InfGen

`InfGen` is a unified autoregressive model for long-term traffic simulation. Regarding the inputs, it first tokenizes all scene context information ($\mathcal{M}$ and $\mathcal{A}_{0:t_0}$) into sequences of discrete tokens, see Sec. 4.1. It then uses an autoregressive model for interleaved next-token prediction, see Sec. 4.2. And Secs. 4.3 and 4.4 provide the details on the model architecture and training process of `InfGen`.

4.1. Tokenization

We aim to convert all agent motions, layouts and the map in a real log into a sequence of discrete tokens. We tokenize each modality differently.

Map Tokenizer. We adapt the map tokenizer from [28]: we uniformly segment all the road elements into a set of fixed-length road vectors. Each vector contains the corresponding features, including start/end points, directions and road type. We collect all road vectors to the map token set $\mathcal{V}_{\text{map}}$.

Agent Motion Tokenizer. We follow the motion tokenizer from prior works [28, 38].[1] Specifically, we segment the continuous trajectories of all agents in the dataset with a fixed time span of 0.5 seconds. Then we use k-disks algorithm [28] to cluster these trajectories into the set of motion vocabulary $\mathcal{V}_{\text{motion}}$. Finally, at every 0.5-second interval, we convert the continuous trajectory into a discrete token with the index of its nearest neighbor in the motion vocabulary.

Agent Pose Tokenizer. When a new agent is inserted into the scenario, its initial pose (position and heading) is given. We tokenize the pose of each inserted agent as a pair of discretized *position* and *heading* tokens. For position, we construct a grid with a radius of R, centered on the ego agent's location with x-axis aligned with ego agent's heading, resulting in position token set $\mathcal{V}_{\text{pos}}$. To get the position token, we obtain the index of the grid closest to the agent based on L2 distance. For heading, we divide the 360° range at the interval of $\Delta\theta$, resulting in heading token set $\mathcal{V}_{\text{head}}$. To get the heading token, we similarly obtain the index of the heading interval closest to the agent heading. For simplicity, we refer the pair of position and heading tokens as a *pose token*.

[1] Please refer to [28] for the details of motion tokenizer and k-disks approach.

Mode Control Tokenizer. We model long-term traffic simulation as interleaved motion simulation and scenario generation. We design the control tokens $\mathcal{V}_{\text{control}}$, consists of 4 special tokens, to mark mode transition between tasks: 1) <BEGIN MOTION>: the next token is an agent motion token to simulate current existing agents; 2) <ADD AGENT>: the next token is an agent pose token to insert a new agent; 3) <KEEP AGENT>: the current agent is kept in the scenario; 4) <REMOVE AGENT>: the current agent will be removed in the next timestep. In the next section we will show how to use $\mathcal{V}_{\text{control}}$ to control the interleaved simulation process.

Tokenizing a real log into a discrete token sequence allows us to convert the complex mixture-task simulation process into a simple interleaved next token prediction task.

4.2. Interleaved Next Token Prediction

Dynamic Agent Matrix. Traffic simulation can be represented by an agent matrix shown in Figure 3. The horizontal axis represent *temporal* lifecycle of each agent: being inserted, active moving and finally exit the scenario. The length of the temporal axis equals to the rollout horizon. On the other hand, the vertical axis represent *spatial* agent layout at each timestep, where the width represent the number of active agents at each step. When a new agent is inserted, a new row is created and append to the matrix. Conversely, when a current agent gets removed, its row gets deleted from the matrix. For long-term scenarios, because agents are frequently being inserted and removed from the scenario, the number of rows of the matrix are also constantly changing. Hence, we term it the *dynamic* agent matrix.

As shown in Figure 3, we represent the long-term traffic simulation task as extending the dynamic agent matrix on different axis. Motion simulation (the upper blue flow) extends the temporal axis by adding new columns with predicted motion tokens. In contrast, scenario generation (the lower green flow) extends the spatial axis by adding new rows with pose tokens of new agents, and removing current rows of exiting agents. The control tokens determine how to interleave these two processes.

Temporal Motion Simulation. We show this process in the blue flow of Figure 3. For the i_{th} active agent at timestep t, we use its current motion token m_i^t as the query $q_{m_i^t}$ to obtain its motion feature f_m[2]. Specifically, we input q_m to a Temporal Attention layer to attend to the key and value of all its own past motion tokens within t_{w} timesteps (within the same row as the query token):

$$q'_{m^t} = \texttt{MHSA}^{\text{t}}(q_{m^t}, \{k_{m^{t-\tau}}\}_{\tau=1}^{t_{\text{w}}}, \{v_{m^{t-\tau}}\}_{\tau=1}^{t_{\text{w}}}). \quad (3)$$

The output is then sent to an Agent-Agent Attention layer to attend to all the other N^t active agents within a valid range $r^{\text{a}\leftrightarrow\text{a}}$ at the same timestep t:

$$q''_{m_i} = \texttt{MHCA}^{\text{a}\leftrightarrow\text{a}}(q'_{m_i}, \{k_{m_j}\}_{j=1}^{N^{\text{t}}}, \{v_{m_j}\}_{j=1}^{N^{\text{t}}}). \quad (4)$$

Finally, the query goes through a Map-Agent Attention layer to attend to the N_r precomputed map tokens within a valid range $r^{\text{m}\leftrightarrow\text{a}}$:

$$f_{m_i} = \texttt{MHCA}^{\text{m}\leftrightarrow\text{a}}(q''_{m_i}, \{k_{m_j}\}_{j=1}^{N_{\text{r}}}, \{v_{m_j}\}_{j=1}^{N_{\text{r}}}). \quad (5)$$

The motion head and control head separately take f_m and output the probabilities over the motion and control tokens, from which we sample a motion token and a control token for each active agent. In this subtask, we enforce the control token to be sampled from <KEEP AGENT> and <REMOVE AGENT>. If control token is <KEEP AGENT>, we add the sampled motion tokens to the next column of each agent. Otherwise, if the control token is <REMOVE AGENT>, we add this control token to the next column and discard the motion token. The above process is conducted for all the current active agents *in parallel* in training and inference. After this, we switch to the scene generation step.

Spatial Scene Generation. We show this process in the green flow of Figure 3. After each motion simulation step, we use a learnable agent query a_0 to obtain the scene generation feature f_{a_0}. Same as the motion query, the agent query is also sent through three attention layers to collect the context information. The latter two layers are the same as the motion query, while the Temporal Attention layer is replaced by a Grid Attention layer. This layer allows the agent query to attend to the occupancy grid tokens g of total size $N_{\text{g}} = N_{\text{p}} = |\mathcal{V}_{\text{pos}}|$ derived from the position tokens:

$$q'_{a_0} = \texttt{MHCA}^{\text{g}}(q_{a_0}, \Gamma(\{k_{g_j}\}_{j=1}^{N_{\text{g}}}), \Gamma(\{v_{g_j}\}_{j=1}^{N_{\text{g}}})), \quad (6)$$

where $\Gamma(\cdot)$ presents the transformation preceding the attention calculation for efficiency. We then pass q'_{a_0} through the other two layers (Eq. 4 and Eq. 5) to produce f_{a_0}. Differently, q'_{a_0} will have various visible range for the active agents $r^{q\leftarrow a}$ at current timestep and for the map tokens $r^{q\leftarrow m}$, please refer to Appendix B for more details.

Then the pose head and control head take f_{a_0} and output distributions for pose and control tokens respectively, from which we sample a pose token and a control token for each generation step (Please refer to Appendix B for more details). We enforce the control token to be sampled from <ADD AGENT> and <BEGIN MOTION> for this subtask. Controlled by <ADD AGENT>, we append a new row to the agent matrix and assign the sampled pose token to the current timestep. Then, conditioned on the all active agents, including the newly inserted one, we repeat the above step to autoregressively insert another new agents. This process is terminated when the sampled control token is <BEGIN

[2]We omit the t and i in this section when possible for simplicity.

MOTION>. In this case, we end the scene generation process and move to motion simulation of the next timestep. Finally, we remove any row that has a <REMOVE AGENT> token at the current timestep from the agent matrix.

4.3. Model Architecture

Token Embedding. Our model takes tokens of different modalities. To model them with a single token sequence, we take different MLP layers to embed different kinds of tokens into the same latent dimension D before entering the model. Please refer to Appendix B for more details.

Modeling Layer. Our transformer model is composed of L blocks of attention layers. As mentioned in Section 4.2, each block contains 4 attention layers: Temporal Attention, Agent-Agent Attention, Map-Agent Attention and Grid Attention. Here the first layer are implemented with multi-head self attention layer (MHSA), while the other layers are multi-head cross attention layer (MHCA). Furthermore, we apply the position-aware attention from prior works [24, 28] to explicitly model the relative positions between tokens. Please refer to the Appendix for the details.

Occupancy Grid Encoder. We obtain occupancy grid features f_{g} of the current scenario with the agent position tokens and map tokens via $\Gamma(\cdot)$. Specifically, given the vocabulary size N_{p} of position tokens $\mathcal{V}_{\text{pos}}$, we directly assign each token with occupation indication $\{0, 1\}$, leading to an agents occupancy grid $g^{1\times N_{\text{p}}} \in \{0, 1\}$. We utilize the occupancy maps of agents in decoding pose token process, to make the agent query efficiently infer the spatial distributions of agents. We use an MLP Layer to convert $g^{1\times N_{\text{p}}}$ to its features $f_{\text{g}}^{N_{\text{p}}\times D}$ before feed it into Grid Attention layers.

4.4. Training

Ground-truth Sequence. As described in Section 3, real-world traffic scenario log [13] naturally contains data of agent motion, insertion and removal behaviors. To train our model with the interleaved NTP problem explained in Section 4.2, we convert each ground-truth log into an ordered sequence of token labels. To this end, we enforce a specific ordering to chain tokens from different modalities. Specifically, at each timestep we arrange each type of tokens with a fixed order: 1) motion tokens from all the current agents; 2) control tokens <REMOVE AGENT> and <KEEP AGENT> for the current agents; 3) pose tokens and <ADD AGENT> for any inserted agents; 4) <BEGIN MOTION> that mark the transition to the next timestep. For the same type of tokens we order them following to the agent's distance to the ego agent from near to far. With this rule we obtain a sequence of ground-truth (GT) token labels from each real log to train our model. Please refer to Appendix C for more details about the GT tokens.

Table 1. Short-term traffic simulation in WOSAC [13] (↑).

Method	Composite	Kinematic	Interactive	Map
TrafficBots [37]	0.6976	0.3994	0.7103	0.8342
GUMP [10]	0.7404	0.4773	0.7872	0.8339
SMART-7M [28]	0.7521	0.4799	0.8048	0.8573
CatK [38]	0.7603	0.4611	0.8103	0.8732
InfGen	0.7514	0.4754	0.7936	0.8502

Learning Objective. As shown in Figure 2, given the GT input tokens as input, our model predicts the distributions of different kinds of tokens at the corresponding position. We then train our model with a set of standard NTP objective for each type of tokens. For example, the loss function for the motion token is:

$$\mathcal{L}_{\text{motion}} = -\sum_{t=1}^{T-1} \log p_\theta(m_i^{t+1} \mid c^{1:t}), \tag{7}$$

where T is the total number of timesteps, m_i^{t+1} is the GT motion token in the next timestep, $p_\theta(m_i^{t+1}|c^{1:t})$ is the model-predicted probability of the GT token. Here, $c^{1:t}$ is the ensemble of all history context the model attends to, including history motions of the same agent, positions of other agents in the current timestep, and the map. We formulate the loss function in the same way for agent pose tokens $\mathcal{L}_{\text{pose}}$ and mode control tokens $\mathcal{L}_{\text{control}}$. We also have supervision of shapes and types for those new agents. Our total training loss can be written as:

$$\mathcal{L} = \lambda_1\mathcal{L}_{\text{motion}} + \mathcal{L}_{\text{pose}} + \lambda_4\mathcal{L}_{\text{control}} + \mathcal{L}_{\text{attr}}, \tag{8}$$

where $\mathcal{L}_{\text{pose}} = \lambda_2\mathcal{L}_{\text{pos}} + \lambda_3\mathcal{L}_{\text{head}}$ and $\mathcal{L}_{\text{attr}} = \lambda_5\mathcal{L}_{\text{shape}} + \lambda_6\mathcal{L}_{\text{type}}$. We directly end-to-end train InfGen with $\mathcal{L}$ on the fully tokenized dataset. During training InfGen not only learns how to conduct the two tasks respectively, but also learns to automatically and seamlessly switch between them. Please refer to Appendix D for more training explanations.

5. Experiments

In this section, we validate our work from two aspects: *1) How does InfGen compare to the SOTA baselines on conventional short-term traffic simulation task in standard benchmarks? 2) How does InfGen perform on our mainly introduced long-term traffic simulation task?*

Dataset. We train and validate InfGen on Waymo Open Motion Dataset (WOMD) [8], with $\sim$ 480K scenarios for training and $\sim$ 44K scenarios for validation. Each scenario consists of a $T = 9.1$ s recorded rollout, with the first $T_{\text{H}} = 1.1$ s historical frames and the subsequent 8 s future frames.

Training. To train InfGen, we use a total batch size of 8 on 8 NVIDIA A5000 GPUs with the AdamW optimizer

Table 2. Long-term traffic simulation evaluation (↑) on WOMD validation split measured by the metrics introduced in Sec. 5.2.

Method	Composite	Kinematic	Interactive	Map-based	Placement-based				
					overall	N_+	N_-	D_+	D_-
SMART-7M [28]	0.6519	0.5839	0.7542	0.8102	0.4324	0.5713	0.4964	0.3371	0.3248
CatK [38]	0.6584	0.5850	0.7584	0.8186	0.4424	0.5842	0.5233	0.3371	0.3248
`InfGen`	0.6606	0.5966	0.7619	0.8087	0.4542	0.6273	0.5635	0.3169	0.3092

and cosine annealing learning rate scheduler. The initial learning rate is 0.0005. For the loss function in Eq. 8, we set $\lambda_1 = \lambda_3 = 1$, $\lambda_2 = \lambda_4 = 10$, $\lambda_5 = 0.2$, $\lambda_6 = 5$.

5.1. WOMD Sim Agent Challenge

We first evaluate `InfGen` on the standard *short-term* traffic simulation task in WOSAC [13]. For this setting, we enforce `InfGen` to skip all the scene generation steps, and predict the rollout for only $8s$. We then evaluate the rollouts under the official WOSAC metrics and compare the results with the top-performing methods in Table. 1. The results show `InfGen` performs very competitive even under the short-term setting without any task-specific tuning, achieving similar performance to SOTA [38] and outperforms strong models [10, 37].

5.2. Long-term Traffic Simulation Setup

Rollout Setup. To be compatible with the map size of WOMD, we set the total rollout duration of our long-term traffic simulation experiments as $T' = 31.1\,\mathrm{s}$ with FPS $= 10$. The first $1.1\,\mathrm{s}$ corresponds to the historical segment, from which we simulate 300 future steps.

WOSAC Metric Adaption. The standard WOSAC metric [13] evaluates short-term traffic simulation by comparing simulated rollouts directly with ground-truth (GT) logs over an 8-second horizon. Specifically, it computes 9 metrics that assess aspects such as kinematic realism, interaction realism, and map adherence, assuming a one-to-one correspondence between simulated and logged agents.

However, in our long-term simulation setting, the rollout duration extends significantly beyond the standard 8-second window, reaching up to 30 seconds. This creates two critical challenges: (1) there is no direct one-to-one correspondence between simulated agents and logged agents over the entire long-term rollout, as agents dynamically enter and exit the scene; (2) the evaluation window (8 seconds in WOSAC) is shorter than our simulation horizon (30 seconds). Therefore, we need to adapt the original WOSAC metric to suit our long-term simulation setting.

Specifically, we adapt the WOSAC evaluation as follows. Given a long-term simulated rollout $\sigma' = (\mathcal{M}, \mathcal{A}'_{0:T'})$, we extract short segments using a sliding window approach. Specifically, we slide a window of length $T_\mathrm{w} = T - T_\mathrm{H}$ (matching the standard $8\,\mathrm{s}$ evaluation window) at a fixed interval Δt throughout the entire simulated rollout. Each sliding window generates a short-term segment $\mathcal{A}'_{t:t+T_\mathrm{w}}$ that matches the length of standard evaluation segments. Finally, we get $\mathcal{A}'_{0:T'} = \{\mathcal{A}'_{\Delta s(i-1):\Delta s(i-1)+T_\mathrm{w}}\}_{i=1}^{P}$ with the number of segments equal to P, and $\Delta s(P-1) + T_\mathrm{w} = T'$.

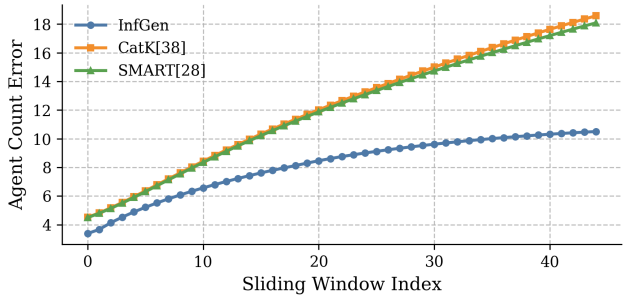

Figure 4. Agent Count Error (ACE) curves of `InfGen` against baselines over 30s long-term simulation rollouts.

Since the number of simulated agents N' in these segments may differ from the logged agents N, we cannot directly apply the original WOSAC evaluation. Instead, we compute agent-level Negative Log-Likelihood (NLL) scores for all simulated agents by evaluating their behaviors against a global distribution learned from the entire validation dataset ($\sim$ 48K scenarios). Concretely, we first estimate empirical distributions for agent motions, interactions, and placements from the entire validation dataset, and then measure how well our simulated agent behaviors conform to these reference distributions. This modified approach ensures a fair and consistent evaluation of long-term simulation realism with varying numbers of agents and rollout durations.

Placement-based Metrics. The standard WOSAC metrics evaluate simulation realism through multiple components: *kinematic-based*, *interaction-based*, and *map-based*. These metrics are then aggregated into an overall realism composite metric using predefined weights. However, these metrics assume a fixed set of agents throughout the evaluation horizon and thus fail to capture the realism of agent insertion and removal events that are essential in long-term traffic simulations. We then propose the *placement-based* component, which comprises 4 types of statistics: the num-

Table 3. Ablation study of `InfGen` on long-term traffic simulation (↑). ✓ indicates remaining unchanged as in Table. 2.

Cont. token	Pos. token	Head. token	Composite	Kinematic	Interactive	Map-based	Placement-based
	✓	✓	0.6328	0.5493	0.7043	0.7961	0.4513*
✓		✓	0.6564	0.5580	0.7768	0.8077	0.4378
✓	✓		0.6509	0.5866	0.7276	0.8107	0.4445
		✓	0.6297	0.5422	0.6962	0.7939	0.4499
✓	✓	✓	0.6674	0.5921	0.7688	0.8003	0.4503

* We take the heuristic approach to remove the agents.

Table 4. Long-term motion prediction evaluation (↑).

Method	Composite	Kinematic	Interactive	Map
SMART-7M [28]	0.7428	0.5413	0.7626	0.8349
CatK [38]	0.7316	0.5216	0.7347	0.8495
`InfGen`	0.7432	0.5495	0.7685	0.8213

ber of placement N_+, the number of removal N_-, the distance of placement D_+ and the distance of removal D_-, where the distances here are relative to the ego agent. We aim to use the placement-based metrics to assess the realism of `InfGen` in modeling the entry and exit of agents during the long-term rollout. Similarly, we calculate the NLLs of the placement-based statistics under the logged distributions in the same way as the other components.

Agent Count Error Metrics. In addition to WOSAC metrics extensions, we also introduce a new Agent Count Error (ACE) metric to evaluate scene realism during long-horizon rollouts. For each sliding window over the 30 s rollout, we compute the mean absolute difference in agent count between simulation and the ground-truth distribution of the validation set. We summarize the results with two scalar metrics (lower is better): *Mean ACE* (overall error) and *ACE Slope* (error growth rate via linear regression).

With the above setup we are now ready to evaluate different methods for long-term traffic simulation.

5.3. Long-term Traffic Simulation Evaluation

Baselines. Since existing works do not focus on long-term rollout, to fairly compare, our baselines are derived by improving upon the SOTA simulation methods **SMART** [28] and **CatK** [38]. SMART leverages vectorized map and agent trajectory data, predicting motion sequences through a decoder-only transformer architecture. CatK further introduces a closed-loop supervised fine-tuning technique and achieve better WOSAC simulation performance.

First, we extend their rollout duration to $T' = 31.1s$ and then calculate the kinematic, interactive and map-based metrics. For the placement-based metrics, we design a heuristic approach: we obtain the entered and exited agents by partitioning the distance between agents and the ego agent by the radius R, which corresponds to the *distance placement* and *distance removal* of placement-based measurements, introduced in Sec. 5.2. During the rollout process, we have the distances from each agent to the ego agent $\{\mathcal{D}^i_{0:T'}\}_{i=1}^N$. At step t, the agents that first run within the range R, whose $\mathcal{D}_t \leq R$ given $\mathcal{D}_{0:t-1} > R$, are considered as *entered* agents, while those agents with $\mathcal{D}_t > R$ and $\mathcal{D}_{0:t-1} \leq R$, are considered as *exited* agents. Throughout the experiment, we adjust R to achieve the highest placement-based score for baselines on validation set. In this case, we assign additional validity values to each agent at each timestep, thus only those valid agents will be included in baseline inferences and evaluation metrics. We follow the default settings in their papers.

Quantitative Results. Under the settings and proposed evaluation metrics in Sec. 5.2, Table. 2 shows the results of long-term traffic simulation under extended WOSAC metrics. As can be seen, `InfGen` achieves better realism performance than baselines. For placement-based metrics, our method significantly outperforms the baselines, demonstrating that our approach effectively models the spatial sequences of agents. The lower scores on map-based metrics may be attributed to the insertion of agents in regions outside driving lanes or near road boundaries, leading to less realistic motion trajectories.

Figure 4 shows the curve of comparison results under our introduced ACE metrics, where our method significantly outperforms prior methods: `InfGen` achieves Mean ACE of 8.1 while baselines (CatK [38],SMART [28]) have scores of 12.2 and 12.0, which reflects the improvements of our dynamically scenario generation; `InfGen` has ACE Slope: 0.15, however, baselines accumulate such errors with slopes of 0.32 and 0.31. This significantly confirms the much better long-term stability observed in Figure 5.

Qualitative Analysis. We further show `InfGen` through the visualizations of the long-term rollouts. As depicted in Figure. 5, the results highlight two core properties of `InfGen`: *long-term* and *closed-loop*. The first scenario depicts a bidirectional driving road, where `InfGen` successfully simulates oncoming traffic. In the third scenario, when the rollout reaches $t = 18s$, most of active agents

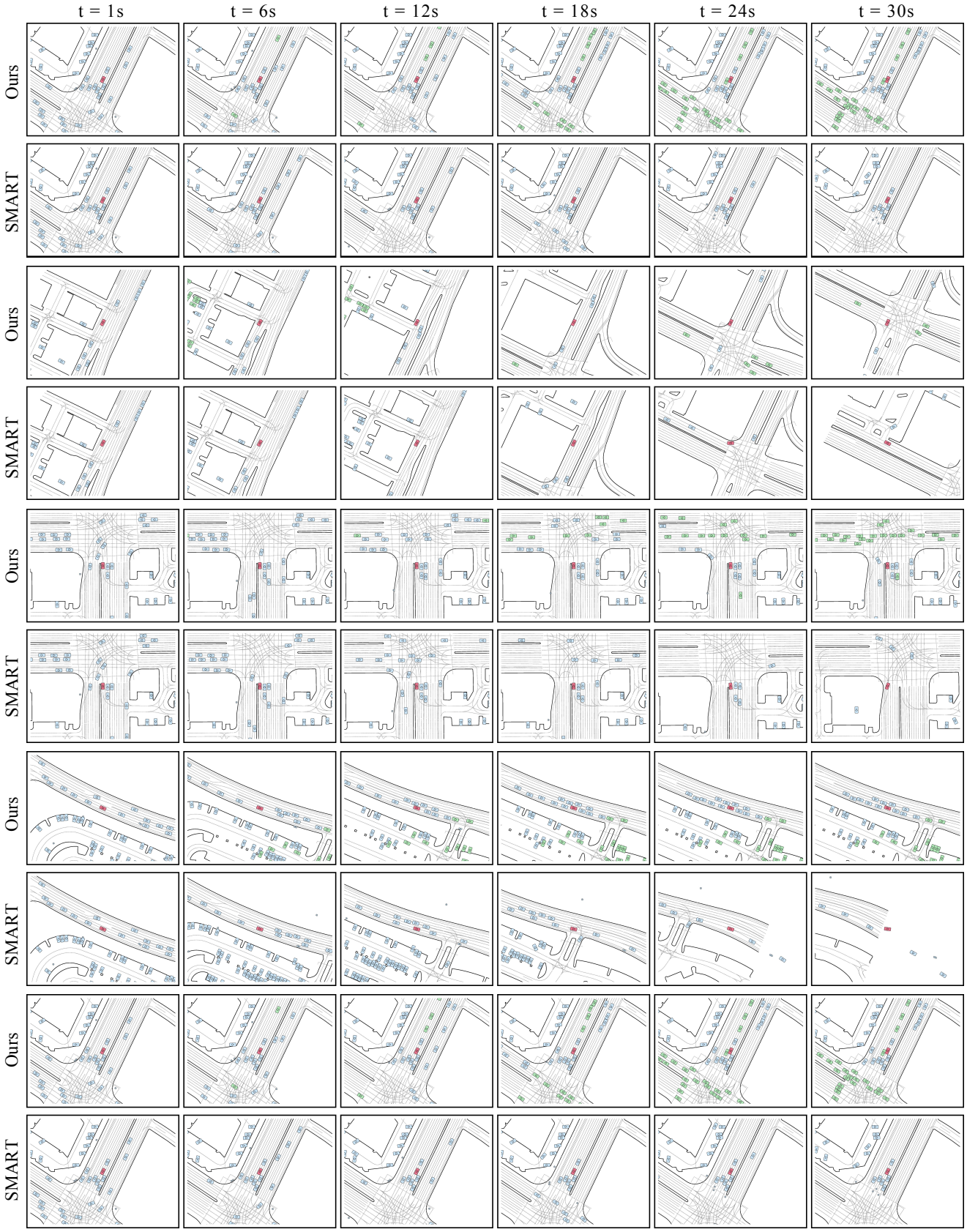

Figure 5. **Qualitative results of long-term closed-loop rollouts for 5 scenarios.** We compare rollouts of `InfGen` and SMART [28] here. are the initially placed agents, are the new agents inserted by `InfGen`, and are the ego agents.

run outside the scenario, resulting in an empty region, as reflected by prior works. Under the control of InfGen, new agents enter the scenario, allowing continued agent-agent and agent-map interactions with high realism. While in baselines, those regions around the ego agent remains empty. Moreover, from the forth scenario, we observe our model can place new agents not only on driving roads but also in parking lots or other open areas.

Motion-only Analysis. In Section 3 we formulate the long-term traffic simulation task as a product of motion simulation and scene generation (Equation 2). Here we investigate the performance of different models in long-term traffic simulation when we only consider motion simulation. Specifically, we disable agent insertion and removal for all the methods, and simply let them rollout for 30 seconds. We then evaluate the rollouts with the adapted WOSAC metrics. As shown in Table 4, all the compared methods have very similar performance. This indicate that simply extending rollout horizon will not reveal important aspects of long-term traffic simulation, like the empty scenarios we see for SMART rollouts in Figure 5.

5.4. Ablation Study

We conduct various ablation studies to validate our methods. We ablate the impact of designed control tokens, position tokens and heading tokens on our task. Due to the high cost of local evaluation, following [38], we use 5% (2204 out of ∼44K scenarios) of the validation split in this part.

Effect of Control Token. As discussed in Sec.4.1, we introduce the control tokens to determine the spatial scene generation sequence. The baselines, to some extent, can be regarded as versions without the <ADD AGENT> token, and naturally, the <REMOVE AGENT> token is also absent. To fully validate the control tokens, we additionally conduct long-term rollout tests while retaining the <ADD AGENT> token but removing the <REMOVE AGENT> token.

Note that we use the heuristic approach to remove agents with distances exceed the R, for two reasons: 1) Adding agents without removing any results in an unrealistic scenario that would not naturally exist; and 2) to ensure a fairer comparison. As shown in Table 3, removing <REMOVE AGENT> token in long-term rollout severely degrades the kinematic and interactive metrics. Unsurprisingly, as the continuously increasing number of agents over time significantly impacts bother their motion states and internal interactions.

Effect of Position Token. We take position tokens to efficiently capture the environment information of local regions, which are ultimately aggregated into the agent query in spatial scene generation. We have an ablation experiment by completely removing the position tokens along with its token embedding, and we instead directly predict the (x, y) locations of agents. The results reveal that the position tokens help the model better address the placement-related issues, as grids simplify the search space. Additionally, through position tokens embedding, the agent query can more efficiently perceive the spatial distribution of the environment.

Effect of Heading Token. The initialization of newly-entered agents' poses, is essential to the their subsequent motions and further the interactions with others. Similarly, we validate replacing the head token prediction with the direct prediction of continuous angle values. The results at table. 3 also reflects that heading tokens can slightly improve the interactive and placement-based performance.

6. Conclusion

In this work we propose InfGen, a unified next-token prediction model for long-term traffic simulation. InfGen learns to automatically switch between temporal motion simulation and spatial scene generation. Our experiments show InfGen significantly outperforms prior methods in long-term simulation, while keeping strong performance on standard short-term simulation. We believe InfGen is a steady step towards realistic trip-level traffic simulation.

Limitations. The main limitation of this paper is that we did not evaluate InfGen under a real trip-level rollout duration (> 5 minutes). Our main constraint is the map areas available in WOSAC scenarios are too small: even in our 30s rollout the ego agent often drive to areas where no map is available. We plan to further explore with suitable data.

References

[1] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric M Wolff, Alex H Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021. 1, 2

[2] Sergio Casas, Cole Gulino, Shuai Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[3] Chameleon Team. Chameleon: Mixed-Modal Early-Fusion Foundation Models. *arXiv preprint arXiv:2405.09818*, 2024. 2

[4] Wei-Jer Chang, Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker. Safe-sim: Safety-critical closed-loop traffic simulation with diffusion-controllable adversaries. *arXiv preprint arXiv:2401.00391*, 2024. 2

[5] Kashyap Chitta, Daniel Dauner, and Andreas Geiger. Sledge: Synthesizing driving environments with generative models and rule-based traffic. In *European Conference on Computer Vision*, pages 57–74. Springer, 2024. 2

[6] Marco Cusumano-Towner, David Hafner, Alex Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor Killian, Stuart Bowers, Ozan Sener, Philipp Krähenbühl, and Vladlen Koltun. Robust autonomy emerges from self-play. *arXiv preprint arXiv:2502.03349*, 2025. 1, 2

[7] Wenhao Ding, Yulong Cao, Ding Zhao, Chaowei Xiao, and Marco Pavone. Realgen: Retrieval augmented generation for controllable traffic scenarios. *arXiv preprint arXiv:2312.13303*, 2023. 2

[8] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 1, 5

[9] Lan Feng, Quanyi Li, Zhenghao Peng, Shuhan Tan, and Bolei Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3567–3575, 2023. 2

[10] Yihan Hu, Siqi Chai, Zhening Yang, Jingyu Qian, Kun Li, Wenxin Shao, Haichao Zhang, Wei Xu, and Qiang Liu. Solving motion planning tasks with a scalable generative model. In *European Conference on Computer Vision*, pages 386–404. Springer, 2024. 2, 5, 6

[11] Siqi Kou, Jiachun Jin, Chang Liu, Ye Ma, Jian Jia, Quan Chen, Peng Jiang, and Zhijie Deng. Orthus: Autoregressive Interleaved Image-Text Generation with Modality-Specific Heads. *arXiv preprint arXiv:2412.00127*, 2024. 2

[12] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[13] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open sim agents challenge. *Advances in Neural Information Processing Systems*, 36:59151–59171, 2023. 2, 3, 5, 6

[14] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 2

[15] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. *arXiv preprint arXiv:2312.04535*, 2023. 2

[16] Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17284–17294, 2022. 2

[17] Luke Rowe, Roger Girgis, Anthony Gosselin, Bruno Carrez, Florian Golemo, Felix Heide, Liam Paull, and Christopher Pal. Ctrl-sim: Reactive and controllable driving agents with offline reinforcement learning. *arXiv preprint arXiv:2403.19918*, 2024. 2

[18] Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. *arXiv preprint arXiv:2503.22496*, 2025. 2

[19] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[20] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8579–8590, 2023. 2

[21] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021. 2

[22] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[23] Shuhan Tan, Boris Ivanovic, Xinshuo Weng, Marco Pavone, and Philipp Kraehenbuehl. Language conditioned traffic generation. In *7th Annual Conference on Robot Learning (CoRL)*, 2023. 2

[24] Shuhan Tan, Boris Ivanovic, Yuxiao Chen, Boyi Li, Xinshuo Weng, Yulong Cao, Philipp Krähenbühl, and Marco Pavone. Promptable closed-loop traffic simulation. *arXiv preprint arXiv:2409.05863*, 2024. 2, 5, 6

[25] Changyao Tian, Xizhou Zhu, Yuwen Xiong, Weiyun Wang, Zhe Chen, Wenhai Wang, Yuntao Chen, Lewei Lu, Tong Lu, Jie Zhou, Hongsheng Li, Yu Qiao, and Jifeng Dai. Mm-interleaved: Interleaved image-text generative modeling via multi-modal feature synchronizer. *arXiv preprint arXiv:2401.10208*, 2024. 2

[26] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021. 2

[27] Cathy Wu, Abdelrahman Kreidieh, Karthik Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: A modular learning framework for mixed autonomy traffic. In *IEEE Transactions on Robotics*, pages 1677–1689, 2021. 1

[28] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng KAN. Smart: Scalable multi-agent real-time motion generation via next-token prediction. In *Advances in Neural Information Processing Systems*, pages 114048–114071. Curran Associates, Inc., 2024. 1, 2, 3, 5, 6, 7, 8

[29] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024. 2

[30] Yuting Xie, Xianda Guo, Cong Wang, Kunhua Liu, and Long Chen. Advdiffuser: Generating adversarial safety-critical driving scenarios via guided diffusion. *arXiv preprint arXiv:2410.08453*, 2024. 2

[31] Zixun Xie, Sicheng Zuo, Wenzhao Zheng, Yunpeng Zhang, Dalong Du, Jie Zhou, Jiwen Lu, and Shanghang Zhang. Gpd-1: Generative pre-training for driving, 2024. 6

[32] Chejian Xu, Ding Zhao, Alberto Sangiovanni-Vincentelli, and Bo Li. Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. *AdvML-Frontiers 2023*, 2023. 2

[33] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, page 2929–2936. IEEE, 2023. 2

[34] Zhiyang Xu, Minqian Liu, Ying Shen, Joy Rimchala, Jiaxin Zhang, Qifan Wang, Yu Cheng, and Lifu Huang. Modality-specialized synergizers for interleaved vision-language generalists. In *International Conference on Learning Representations (ICLR)*, 2025. 2

[35] Xiuyu Yang, Zhuangyan Zhang, Haikuo Du, Sui Yang, Fengping Sun, Yanbo Liu, Ling Pei, Wenchao Xu, Weiqi Sun, and Zhengyu Li. Rmmdet: Road-side multitype and multigroup sensor detection system for autonomous driving. *arXiv preprint arXiv:2303.05203*, 2023. 2

[36] Chris Zhang, James Tu, Lunjun Zhang, Kelvin Wong, Simon Suo, and Raquel Urtasun. Learning realistic traffic agents in closed-loop. In *7th Annual Conference on Robot Learning*, 2023. 2

[37] Zhejun Zhang, Christos Sakaridis, and Luc Van Gool. Trafficbots v1. 5: Traffic simulation via conditional vaes and transformers with relative pose encoding. *arXiv preprint arXiv:2406.10898*, 2024. 2, 5, 6

[38] Zhejun Zhang, Peter Karkus, Maximilian Igl, Wenhao Ding, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 3, 5, 6, 7, 9

[39] Jianbo Zhao, Jiaheng Zhuang, Qibin Zhou, Taiyu Ban, Ziyao Xu, Hangning Zhou, Junhe Wang, Guoan Wang, Zhiheng Li, and Bin Li. Kigras: Kinematic-driven generative model for realistic agent simulation. *arXiv preprint arXiv:2407.12940*, 2024. 2

[40] Tianyang Zhao, Yuke Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[41] Ziyuan Zhong, Davis Rempe, Yuxiao Chen, Boris Ivanovic, Yulong Cao, Danfei Xu, Marco Pavone, and Baishakhi Ray. Language-guided traffic simulation via scene-level diffusion. In *Conference on Robot Learning*, pages 144–177. PMLR, 2023. 2

[42] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3560–3566. IEEE, 2023. 2

[43] Zikang Zhou, Haibo Hu, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *arXiv preprint arXiv:2405.17372*, 2024. 2

Long-term Traffic Simulation with Interleaved Autoregressive Motion and Scenario Generation

Supplementary Material

A. Supplementary Videos

We provide additional videos for better demonstration of `InfGen`. In this video, we showcase the existing problem of current baselines, as introduced in Sec.1, and the comparison between baselines and our method. Then we have more qualitative examples. Please refer to our project page for details.

B. Model Details

In this section, we provide more details of `InfGen` model. We give an overview of the main hyperparameters of the model architecture, and explain the setup of agent embedding and query which are directly operated by transformer decoder. Then we describe more about how to decode various outputs.

Hyperparameters. We extend the base model from SMART-7M [28] to train out `InfGen` model. We also have the detailed descriptions about the transformer decoders in Sec. 4.2, and we list all the main hyperparameters of the model architecture and our implementations in Table. 5.

Note that we also inherit the road network from [28] (See Sec.3.3 of their paper) in our NTP pre-training process (where the Map-Map Attention Layers are incorporated). Since we adopt the map tokenizer and transform maps into discrete tokens, this network will help the model to understand the topological connectivity and continuity of unordered map tokens. Finally, the total model size of `InfGen` is around 11M.

Table 5. Main hyperparameters of `InfGen` model.

Hyperparameter	Value
Transformer Decoder	
attention head dimension	16
number of attention heads	8
number of motion transformer blocks	6
number of scene transformer blocks	3
number of map transformer blocks	3
D, feature dimension of token embedding	128
number of frequency bands	64
t_{w}, Agent temporal Attention radius	12
$r^{\text{a}\leftrightarrow\text{a}}$, Agent-Agent Attention radius	60
$r^{\text{m}\leftrightarrow\text{a}}$, Map-Agent Attention radius	30
$r^{\text{m}\leftrightarrow\text{m}}$, Map-Map Attention radius	10
$r^{\text{q}\leftarrow\text{a}}$, Query-Agent Attention radius	10
$r^{\text{q}\leftarrow\text{m}}$, Query-Map Attention radius	75,10
Tokenizer	
R, radius of position grids	75
Δg, grid interval of $\mathcal{V}_{\text{pos}}$	3
$\Delta\theta$, angle interval of $\mathcal{V}_{\text{head}}$	3
$\lvert\mathcal{V}_{\text{motion}}\rvert$, vocabulary size of motion tokens	2048
$\lvert\mathcal{V}_{\text{map}}\rvert$, vocabulary size of map tokens	1024
$\lvert\mathcal{V}_{\text{pos}}\rvert$, vocabulary size of position tokens	1849
$\lvert\mathcal{V}_{\text{head}}\rvert$, vocabulary size of heading tokens	120
$\lvert\mathcal{V}_{\text{control}}\rvert$, vocabulary size of control tokens	4

B.1. Agent Feature Learning

Agent Embedding. In our stacked attention layers, we directly operate on different tokens according to each specific task. For practical purposes, we construct a comprehensive aggregated agent embedding $F_{\text{agent}} \in \mathbb{R}^{T\times D}$ (where T denotes the rollout horizon) to process them uniformly as token sequences in the transformer layers, as mentioned in Sec. 4.3. Specifically, we obtain the k_m, v_m in Eq. 3, 4, 5 from tokens of different modalities through direct fusion, as Eq. 9. We first concatenate all the features on channel dimension and then produce the agent embedding through 1-layer MLP which is directly operated by transformer layers:

$$\texttt{MLP}(\texttt{Concat}(F_{\text{motion}}, F_{\text{position}}, F_{\text{validity}}, F_{\text{attribute}})), \quad (9)$$

where *validity* denotes if the agent at current timestep is visible by the environment, which is labeled in real log data. And $F_{\text{attribute}}$ aggregates the shape and type values of each agent. Notably, *validity* $\in \mathbb{R}^{T\times 1}$ here reflects not only the state of agent but also implicitly embeds the control tokens. To get all these features from their low-dimension values, we use MLP Layers for those in continuous space (*e.g.*, agent shapes) and learnable embeddings for those in discrete space. Finally, we have the dynamic agent matrix tensor $F_{\mathcal{A}'} \in \mathbb{R}^{A\times T\times D}$ (as the matrix in Fig. 3) which is continuously updated during the rollout in an interleaved autoregressive manner.

Agent Query. As we explained in Sec. 4.2, we start from an agent query a_0 to autoregressively insert the agents in spatial scene generation. Ideally, we consider that the spatiotemporal features of the agent query are fully aligned with the ego agent, *i.e.*, it follows the ego agent's position and motion. Since we are primarily concerned with the environment around the ego agent in this task, and the position tokens $\mathcal{V}_{\text{pos}}$ are also centered at the ego agent. To build this query, we take exactly the same approach as Eq. 9. For its

motion token, we directly use another new special token instead of any existing token from $\mathcal{V}_{\text{motion}}$. For its position token, we fix it as the centroid of the $\mathcal{V}_{\text{pos}}$. For its validity, we set it as invalid. And we use new special agent type and shape values as its inherent attributes.

B.2. Modeling Layer

Position-aware Attention. We explicitly model the relative spatial-temporal positions between input tokens in attention calculation (as in Eq. 3, 4, 5). For each query-context token pair (*e.g.*, agent-agent or agent-map), we add the relative positional encoding from context token F_{c} to query token F_{q} ($F_{\text{c}}, F_{\text{q}} \in \mathbb{R}^D$ are from $F_{\mathcal{A}'}$).

Specifically, we incorporate 3 types of descriptors: relative distance Δp, the relative direction Δd, the relative heading $\Delta\theta$, where $p \in \mathbb{R}^2$ and $\theta \in (-\pi, \pi)$ denotes the positions and headings of input tokens. For temporal attention module (Eq. 3), we add additional time span Δt to formulate 4D descriptors:

$$\begin{aligned}&\Delta p_{\text{cq}} = ||p_{\text{c}} - p_{\text{q}}||_2,\ \Delta\theta_{\text{cq}} = \theta_{\text{c}} - \theta_{\text{q}},\ \Delta t_{\text{cq}} = t_{\text{c}} - t_{\text{q}},\\ &\Delta d_{\text{cq}} = \text{atan2}(\text{p}_{\text{c,y}} - p_{\text{q,y}},\ p_{\text{c,x}} - p_{\text{q,x}}) - \theta_{\text{q}},\end{aligned} \tag{10}$$

which are formulated to $r_{ij} \in \mathbb{R}^D$, the relative positional encodings of tokens F_i,F_j and then added to keys, values in geometric attention layers (Eq.3,4,5):

$$r_{ij} = \texttt{PE}([\Delta p_{ij}, \Delta d_{ij}, \Delta\theta_{ij}, \Delta t_{ij}]), \tag{11}$$

$$F^l_{\mathcal{A}'} = \texttt{Attn}(\phi_q(F^{l-1}_{\mathcal{A}'}), \phi_k(F^{l-1}_{\mathcal{A}'}), \phi_v(F^{l-1}_{\mathcal{A}'}), \mathbf{r}, \mathbf{I}). \tag{12}$$

In Equation 11, PE is the Fourier embedding layers. In Equation 12, $(i, j) \in \mathbf{I}$ and $\mathbf{I} \in \mathbb{N}^{K\times 2}$ indicates the directed or symmetric index set of total involved K context-query token pairs which are determined through distance thresholds $r^{\text{q}\leftarrow\text{c}}$ (visible range centered on query token specified in Table 5) with the upper limitation of number of total context tokens N_c. While $\mathbf{r} \in \mathbb{R}^{K\times D}$ denotes the stacked positional encodings of total K pairs. And ϕ represents the projection function to query, key and value, l reflects the index of attention layer.

Decoding Pose Token. As described in Sec. 4.2 and Fig. 3, we sample new pose token from the categorical distributions from the updated query feature f_{a_0} at each autoregressive step, simultaneously with the control token. We detail the decoding process of the pose tokens:

we formulate the pose head as the sequentially connected position head and heading head, in this case we decode pose token through two separate steps. Given an agent query a_0, we attach it to the ego agent to attend to environments with position-awareness, and produce the scene generation feature q'_{a_0}. We then first input it to position head to get the distribution over the position tokens V_{pos}, from which we get the position token of the new agent candidate through top-K sampling. Secondly, we get the updated agent query a'_0 by locating it at the accurate position which is translated from the position token, with its heading same as the one of the ego agent. And again let a'_0 to attend to its environment and output refined feature $q'_{a'_0}$, then send it to the heading head to get another probability distribution over the heading tokens V_{head}. The headings of the new agents are determined by sample the token with the highest probability. In step of decoding headings of new agents, only Agent-Agent Attention and Map-Agent Attention will be employed, since we consider the headings are highly related to the surrounding agents and maps, but not the occupancy grids.

Note that, in Map-Agent Attention Layers, we use different visible range when decoding position tokens and heading tokens. For position tokens, we use an $r = 75m$ while for heading tokens, $r = 10m$ (also specified in Table 5). We autoregressively repeat such position-heading decoding step which is controlled by control tokens.

Decoding Attributes. As mentioned in Eq. 8, for those newly-inserted agents, we also calculate the losses of their shapes and types, since these attributes also have inherent relationship with their initial pose. We directly predict the continuous shapes value (l, h, w) through 3-layer MLP head. To get the types, we predict the categorical distributions over 3 defined types in WOMD (vehicle, cyclist and pedestrian) and take top-1 sampling.

C. Token Details

In this section, we have more details regarding the design of our tokenization mechanisms, especially the control tokens, and the formulations of GT tokens used for training.

Temporal Tokenization. Regarding the tokens associated with temporal transitions, *e.g.*, motion tokens, control tokens, we first tokenize the time axis at time span of $\delta = 5$ step (0.5 s at 10 FPS), as Eq. 13. Accordingly, a real log from WOMD [13] ($n = 91$ steps for 9.1 s) results in $N = \lfloor n/\delta \rfloor = 18$ discrete tokens.

$$\text{Tokenize}(\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{n-1}\}) = \{\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_{N-1}\}, \tag{13}$$

where $\mathbf{x}$ denotes features with temporal transitions, *e.g.*, motions x^{m}, validities x^{v}. Importantly, the value of each $\hat{\mathbf{x}}$ is defined based on different rules. For motions, each $\hat{x}^{\text{m}}_k$ is aggregated motions vector $x^{\text{m}}_{\delta k:\,\delta(k+1)}$ over the k-th time segment, as mentioned in Sec. 4.1. Furthermore, regarding the validity of agents, we consider both the starting and ending steps within its time segment: $\hat{x}^{\text{v}}_k$ is considered True if and only if both $x^{\text{v}}_{\delta k}$ and $x^{\text{v}}_{\delta(k+1)}$ are True, *i.e.*, $\hat{x}^{\text{v}}_k = x^{\text{v}}_{\delta k} \wedge x^{\text{v}}_{\delta(k+1)}$, as a motion token is meaningful only when $\hat{\mathbf{x}}$ is valid.

Control Tokens. We aim to explain *how to derive the control tokens from real logs.* Starting from the token-wise

validity sequences (Eq. 13), we define the <ADD AGENT> token as:

$$\hat{x}^{\mathrm{c}}_{k_1} \text{ when } \hat{x}^{\mathrm{v}}_{k_1} = \texttt{True} \text{ and } \forall\, 0 \le i < k_1,\ \hat{x}^{\mathrm{v}}_{i} = \texttt{False}, \tag{14}$$

and the <REMOVE AGENT> token $\hat{x}^{\mathrm{c}}_{k_2}$ is symmetrically defined as the last valid token such that all subsequent ones are invalid, *i.e.*, $\forall\, k_2 < i \le N-1,\ \hat{x}^{\mathrm{v}}_{i} = \texttt{False}$. Thus, we set all tokens between <ADD AGENT> and <REMOVE AGENT>, $\hat{x}^{\mathrm{c}}_{k}\ (k_1 < k < k_2)$, as the <KEEP AGENT> token.

Notably, for two special cases: 1) we force the $\hat{x}^{\mathrm{c}}_{k_1}$ with $k_1 = 0$ of Eq. 14 to be <ADD AGENT>; 2) for any <REMOVE AGENT> token $\hat{x}^{\mathrm{c}}_{k_2}$ with $k_2 = N-1$, we force it to be instead <KEEP AGENT>.

As we explained in Sec. 4.2, the tokens <ADD AGENT> and <BEGIN MOTION> only present when we examine the dynamic agent matrix column-wise. Therefore, when we organize the tokens sequence according to the spatial layout (as opposed to Eq. 13), <BEGIN MOTION> is defined as the next token after all the <ADD AGENT> tokens:

$$\hat{x}^{\mathrm{c}}_{l} \text{ when } \forall\, i < l,\ \hat{x}^{\mathrm{c}}_{i} \text{ is } \texttt{<ADD AGENT>}. \tag{15}$$

Moreover, for these <ADD AGENT> tokens of the GT spatial sequence, they are ordered according to the distances from the ego agent, as explained in Sec. 4.4.

Empty Token. Since we incorporate the invalid steps/agents, *e.g.*, those with $\hat{x}^{\mathrm{v}} = \texttt{False}$, into `InfGen` model, we also involve the empty tokens shown in Figure 3, as part of dynamic agent matrix tensor $F_{\mathcal{A}'} \in \mathbb{R}^{A \times T \times D}$. To build these invalid agent embeddings, we follow the similar methods of the agent query, but set their positions and headings to zeros.

Introducing these invalid values can bring unexpected noise within the modeling layers (in Sec. 4.2), specifically, interactions between tokens from different timesteps or different agents when any side of them has $\hat{x}^{\mathrm{v}} = \texttt{False}$, which arise from two sources: 1) the construction of dynamic agent matrix tensor $F_{\mathcal{A}'}$ (Eq. 9); 2) position-aware attention layers (Eq. 10, 11). We address them through applying the rules:

$$\hat{\mathbf{x}}_{\text{invalid}} - \hat{\mathbf{x}}_{\text{invalid}} \leftarrow -\mathbf{z}_{\text{invalid}}, \tag{16a}$$

$$\hat{\mathbf{x}}_{\text{valid}} - \hat{\mathbf{x}}_{\text{invalid}} \leftarrow \mathbf{z}_{\text{trans}}, \tag{16b}$$

$$\hat{\mathbf{x}}_{\text{invalid}} - \hat{\mathbf{x}}_{\text{valid}} \leftarrow -\mathbf{z}_{\text{trans}}, \tag{16c}$$

where $\hat{\mathbf{x}}$ denotes tokens of various features, *e.g.*, motion tokens $\hat{x}^{\mathrm{m}}$, heading tokens $\hat{x}^{\mathrm{h}}$, and $\hat{\mathbf{x}}_{\text{invalid}}$, $\hat{\mathbf{x}}_{\text{valid}}$ reflect their corresponding $\hat{x}^{\mathrm{v}}$ are `False`, `True`, respectively. We force these values to be our predefined constant values $\mathbf{z}$ to eliminate such noises due to the non-constant $\hat{\mathbf{x}}^{\text{valid}}$. Since the model actually only needs the qualitative characteristic of the transition between invalid and valid states, rather than the specific quantitative values. In our experiments, we set $\mathbf{z}_{\text{trans}} = 1$ and $\mathbf{z}_{\text{invalid}} = -2$. Without Equation 16, `InfGen` will suffer from the disruptive noises, preventing effective modeling of the control sequence.

D. Training Details

As we summarized in Sec. 4.4, we efficiently end-to-end train `InfGen` model on multimodal token sequences. Basically, we parallelly train temporal motion simulation and spatial scene generation as standard NTP task of each individual token modality, and perform interleaved autoregression in inference stage. We break down the details of the each aspect for training process in this section.

D.1. Temporal Simulation

Temporal Motions Training. We train on temporal motion tokens similar to prior works [28], and additionally deal with the transition of before and after an agent is inserted or removed. Given the temporal discrete tokens of one agent in Eq. 13, we have their GT motion tokens $\{\hat{x}^{\mathrm{m}}_k\}_{k=0}^{N-1} \subseteq \mathcal{V}_{\text{motion}}$, validities $\{\hat{x}^{\mathrm{v}}_k\}_{k=0}^{N-1} \subseteq \{0, 1\}$ ($0 =$ `False`, $1 =$ `True`), and, furthermore, the control tokens. From the view of the temporal axis, <ADD AGENT> denotes the start of the sequence (BOS) while <REMOVE AGENT> denotes the end of the sequence (EOS). Therefore, we refer to the states of the agent as its validities combined with BOS and EOS.

To supervise the motion tokens tensor $Y^{\mathrm{m}} \in \mathbb{R}^N$ predicted by motion head, we derive the motion mask $M^{\mathrm{m}} \in \mathbb{B}^N$ from the states[3]. Assuming the step of the BOS and EOS are s_{BOS}, s_{EOS}, then we have:

1) $M_{s_{\text{BOS}}} = 1$: the step of BOS;
2) $M_{s_{\text{BOS}}+1} = x^{\mathrm{v}}_{s_{\text{BOS}}+2}$ (with $x^{\mathrm{v}}_{s_{\text{BOS}}} = x^{\mathrm{v}}_{s_{\text{BOS}}+1} = 1$): the next step after BOS;
3) $M_s = x^{\mathrm{v}}_{s-1} \cdot x^{\mathrm{v}}_{s} \cdot x^{\mathrm{v}}_{s+1},\ \forall s,\ s_{\text{BOS}} + 1 < s < s_{\text{EOS}}$: the steps between the step after BOS and EOS (not included) only when the corresponding GT motions are valid.

Otherwise, the steps not satisfy the conditions above have $M_s = 0$, including the EOS. Let $X^{\mathrm{m}} := \{\hat{x}^{\mathrm{m}}_k\}_{k=0}^{N-1}$, then the total loss for N motion tokens is:

$$\mathcal{L}^{\mathrm{m}}_{1:N} = \frac{1}{|\mathcal{I}|} \sum_{k \in \mathcal{I}} \texttt{CE}\big(Y^{\mathrm{m}}_k,\ X^{\mathrm{m}}_k\big),\ \mathcal{I} = \{k \mid M^{\mathrm{m}}_k = 1\}, \tag{17}$$

where `CE` is the CrossEntropy loss, as also described in Eq. 7.

Temporal Controls Training. In the part of temporal simulation, we train on temporal control tokens $\{\hat{x}^{\mathrm{c}}_k\}_{k=0}^{N-1} \subseteq \{$<NULL>, <KEEP AGENT>, <REMOVE AGENT>$\}$ similar to motion ones. We have described how to derive the control

[3] We omit the superscript of M in this section when possible for simplicity.

tokens <KEEP AGENT> and <REMOVE AGENT> from the validities in Sec. C. And we have <NULL> as the placeholder token to indicate those steps without any control operations, which allows X^{ct} to fully represent the entire GT temporal token sequence.

To supervise the control tokens tensor $Y^{\text{ct}} \in \mathbb{R}^N$ predicted by control head, we derive the temporal control mask $M^{\text{ct}} \in \mathbb{B}^N$ from the states. Here we have:

1) $M_{s<s_{\text{BOS}}} = 0$: the steps before BOS (not included);
2) $M_{s_{\text{BOS}}} = 1$: the step of BOS;
3) $M_{s_{\text{BOS}}+1} = x^{\text{v}}_{s_{\text{BOS}}+2}$ (with $x^{\text{v}}_{s_{\text{BOS}}} = x^{\text{v}}_{s_{\text{BOS}}+1} = 1$): the next step after BOS;
4) $M_{s\geq s_{\text{EOS}}} = 0$: the steps after EOS (included);
5) $M_s = x^{\text{v}}_{s-1} \cdot x^{\text{v}}_s \cdot x^{\text{v}}_{s+1},\ \forall s,\ s_{\text{BOS}}+1 < s < s_{\text{EOS}}$: the steps between the step after BOS and EOS (not included) only when the corresponding GT motions are valid.

Then the total loss $\mathcal{L}^{\text{ct}}_{1:N}$ for the entire temporal control token sequence is calculated similar to Equation 17 which takes Y^{ct},X^{ct},M^{ct} as inputs. Note that the steps with $M_s = 1(s_{\text{BOS}} < s < s_{\text{EOS}} - 1)$ correspond to the <KEEP AGENT> tokens, while those with $M_s = 1(s = s_{\text{EOS}} - 1)$ correspond to the <REMOVE AGENT> tokens. To alleviate the imbalance of these two control tokens, we set the label weights: $w(\texttt{<KEEP AGENT>}) = 0.1$ and $w(\texttt{<REMOVE AGENT>}) = 0.9$ when calculating the CrossEntropy Loss.

D.2. Spatial Generation

Spatial Controls Training. For the spatial scene generation, we train on the control sequence $\{\hat{x}^{\text{cs}}_k\}^L_{k=0} \subseteq \{\texttt{<NULL>}, \texttt{<ADD AGENT>}, \texttt{<BEGIN MOTION>}\}$. And L denotes the total number of agents (including those existing and to be added) in a real log and we force $L = 32$ in training process for saving memory. We also use <NULL> as the placeholder for those agents without controls (*e.g.*, existing and not to be added ones), allowing X^{cs} to fully represent the entire GT spatial token sequence.

To formulate the spatial token sequence X^{cs}, we reorganize the all L agents along the spatial axis, as explained in Sec. C, the first sequence (BOS) when scene generation begins, while <BEGIN MOTION> denotes sequence (EOS). Hence, the tokens between BOS and EOS (not included) are all <ADD AGENT> tokens, and <NULL> only present after EOS which corresponds to those agents currently in motion simulation.

As a considerable number of the trailing tokens in X^{ct} are <NULL> that cannot be trained, we further truncate the trailing part of X^{ct}—only consider the first $L' = 10$ tokens in training for more efficiency. And the size of X^{cs} in inference is *scalable* since we train in an autoregressive way.

To supervise the control tokens tensor $Y^{\text{cs}} \in \mathbb{R}^{L'}$ predicted by control head, we also have the mask $M^{\text{ct}} \in \mathbb{B}^{L'}$ following:

1) $M_{s_{\text{BOS}}} = 1$: the step of BOS;
2) $M_{s\geq s_{\text{EOS}}} = 0$: the steps after EOS (included);
3) $M_s = 1,\ \forall s,\ s_{\text{BOS}} < s < s_{\text{EOS}}$: the steps between BOS and EOS (not included).

Then the total loss $\mathcal{L}^{\text{cs}}_{1:M'}$ for the spatial control tokens is obtained similar to Equation 17. We also have label weights: $w(\texttt{<ADD AGENT>}) = 0.1$ and $w(\texttt{<BEGIN MOTION>}) = 0.9$ to deal with the class imbalance.

Spatial Hybrid Attention. To model such spatial sequence X^{ct} in an autoregressive manner, we adapt the causal attention mechanism (in Agent-Agent Attention layers) similar to the Temporal Attention layers.

Specifically, when predicting token $\hat{\mathbf{x}}_t$ (*e.g.*, motion token $\hat{x}^{\text{m}}_t$, control token $\hat{x}^{\text{c}}_t$) in temporal simulation, it will only involve the history tokens $\{\hat{\mathbf{x}}_{t-\tau}\}^{t_{\text{w}}}_{\tau=1}$ within the time window as the context (as Eq. 3). Therefore, given the spatial token sequence $X^{\text{cs}} \in \mathbb{R}^{L'}$ and there exist A' agents already in motion simulations, we construct a hybrid mask $M^{\text{ct}}_{\text{hybrid}} \in \mathbb{B}^{L'\times(A'+L')}$ which consists of two parts:

1) For those A' agents that already exist, they will not be masked out: $M^{\text{ct}}_{\text{hybrid}}[1:L',\ 1:A] = \mathbf{1}^{L'\times A}$.
2) For those L' agents to be predicted (may not all correspond to <ADD AGENT>), we have $M^{\text{ct}}_{\text{hybrid}}[1:L',\ A+1:A+L']$ to be a standard causal mask to exclude the futures in attention layers.

We can wite it as:

$$M^{\text{cs}}_{\text{hybrid}}[i,j] = \begin{cases} 0, & \text{if } j \leq i \text{ or } j < A' \\ -\infty, & \text{otherwise} \end{cases}, \tag{18}$$

where $i \in [1, L']$, $j \in [1, A' + L']$. Note that the ultimate context which query features attend to is determined by jointly applying $M^{\text{cs}}_{\text{hybrid}}$ and other possible masks (*e.g.*, from the visible range).

In this way, we train on spatial token sequence with smaller context length $L' = 10$ while extend it to a scalable number (> 10) with an upper limit of 128.

E. Additional Results

Long-term Traffic Simulation. We show more qualitative comparison results of InfGen and the baselines [24,32] in Fig. 9, 10,and 11. In these scenario, we again demonstrate the strengths of our approach. As the ego agent travels for away from the initial locations, new agents appear in our examples while maintaining a great realism, seamlessly continuing the interaction process. This indicates that InfGen can effectively one of the major challenges of long-term traffic simulation task.

Metrics Curve for Long-term Simulation. We further investigate how simulation realism evolves over the duration of long-term rollouts by plotting metric scores for each sliding window index in Figure 6. Specifically, we show the evolution of three WOSAC metric components (Composite,

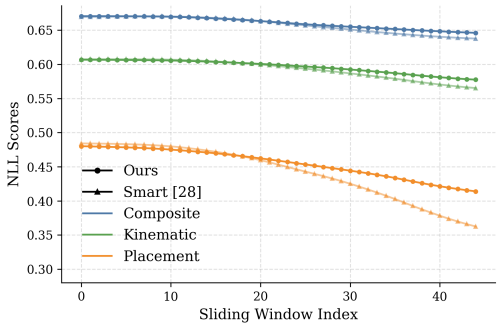

Figure 6. Metrics (adapted WOSAC) curve of InfGen against SMART [28] over the 30s long-term simulation rollouts.

Kinematic, and Placement-based) for both our method and SMART [24].

As expected, we observe a gradual decrease in realism scores across all methods, reflecting the increasing difficulty of maintaining realism over extended simulation periods. However, our method consistently outperforms SMART by exhibiting a notably slower decline in all metrics, particularly in the *placement-based* component. This significant improvement highlights the effectiveness of our proposed interleaved scene generation and motion simulation approach, enabling sustained realism by dynamically handling agent insertions and removals. These quantitative results strongly align with our qualitative observations, further emphasizing the importance of explicitly modeling agent placement and removal to achieve realistic long-term traffic simulations.

F. Limitations and Future Direction

Although InfGen has achieved promising results on long-term traffic simulation, our method is limited in some aspects, as briefly described in Sec. 6. In this section, we have detailed discussions on these terms.

Failure Cases. We have some failure cases existed:

(1) Unreasonable inserted agents. In some examples, our method may have unreasonable newly entered agents in traffic scenario. As shown in Fig. 7, at $t = 6\,\text{s}$ and $t = 12\,\text{s}$, the agents highlighted by red boxes occupy the road boundaries, which is unrealistic in real-world scenarios. It indicates that our method lacks sufficient control at such a fine-grained level. Notably, we did not impose any explicit constraints on this kind of cases, such as regularization losses.

(2) Incorrect initial motion inferring. During the spatial sequence prediction, InfGen first observes overall traffic scenario before placing new agents in potential locations. When these agents are located in a complex road situation, they may fail to accurately infer their initial velocity (or motion) for the subsequent rollout. For example, as shown in Fig. 8, some new agents incorrectly remain stationary on

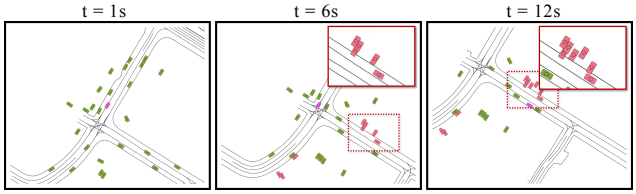

Figure 7. Failure case #1: newly-entered agents appear unreasonably on the boundary of the road.

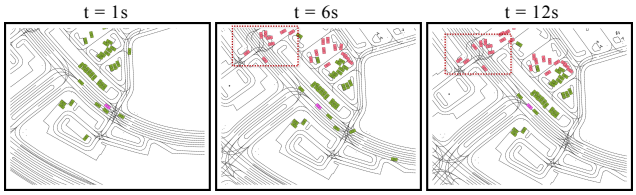

Figure 8. Failure case #2: In the region with complex map structure (highlighted by red box), newly-entered agents fail to correctly infer their initial velocity (or motion) and remain stationary, which is unrealistic.

the driving lanes, which also impacts the motions of other agents, ultimately reducing the realism.

(3) Agents flickering modeling. According to our observations, the phenomenon of agents "flickering" is prevalent in real-world data, particularly in regions farther from the ego agent. And it arises due to the instability of the ego agent's remote perception. In our work, we handle these flickering agents in two ways: first, we discard agents with a presence duration shorter than $0.5\,\text{s}$; second, we change the flickering frequency, which can be attributed to the resolution of discretization on time axis (as introduced in Sec. C). Specifically, the minimum temporal granularity that each token can represent is $0.5s$. As a result, InfGen inherently struggles to generate highly realistic agents exhibiting flickering behavior.

A potential solution is to introduce another format of tokens $\mathcal{V}_{\text{validity}}$ which reflects the frame-wise validity within even one $0.5\,\text{s}$ token, and make InfGen learn it in temporal simulation. Given that each $0.5\,\text{s}$ segment contains 5 valid timesteps, with each step has 2 possible states (invalid, and valid), we can derive the vocabulary size: $2^5 = 32$. But the task becomes more complex under such conditions. We leave this as a future direction.

Future Directions. In addition to addressing the limitations discussed, some other potential interesting improvements may be as below:

(1) Long context understanding and learning. While InfGen effectively addresses the challenge faced by Long-term Sim Agent—modeling the insertion and deletion of agents interleaved with their motions in continuously evolving traffic scenarios to maintain high realism over extended

rollout durations—we believe that another critical bottleneck for even longer horizons is the accumulation of errors over the rollout process. In our work, we adhere to a unified next-token prediction paradigm for end-to-end training, with reference to a historical information constrained by the temporal length. Some hard cases in a long-long-term rollout may fail: in a busy intersection where the lateral traffic passes, followed by the longitudinal traffic while other different lanes remain open, the execution intervals of different actions can be significantly long, and the rules can be greatly complex. Some works [24, 38] utilize closed-loop training or finetuning for improvements, which also cannot be a solution. Thus, it remains an other challenge.

(2) Driving Map Generation. The duration of long-term rollout controlled by `InfGen` is significantly constrained by the size of the map region—-without this limitation, it would be possible to extend the rollout even further. Therefore, integrating the map generation would be a substantial improvement, enabling longer and more flexible simulations. Some concurrent works, such as GPD-1 [31], have some progress in this point, making it a promising direction for near future.

Figure 9. More qualitative comparison results #1.

Figure 10. More qualitative comparison results #2.

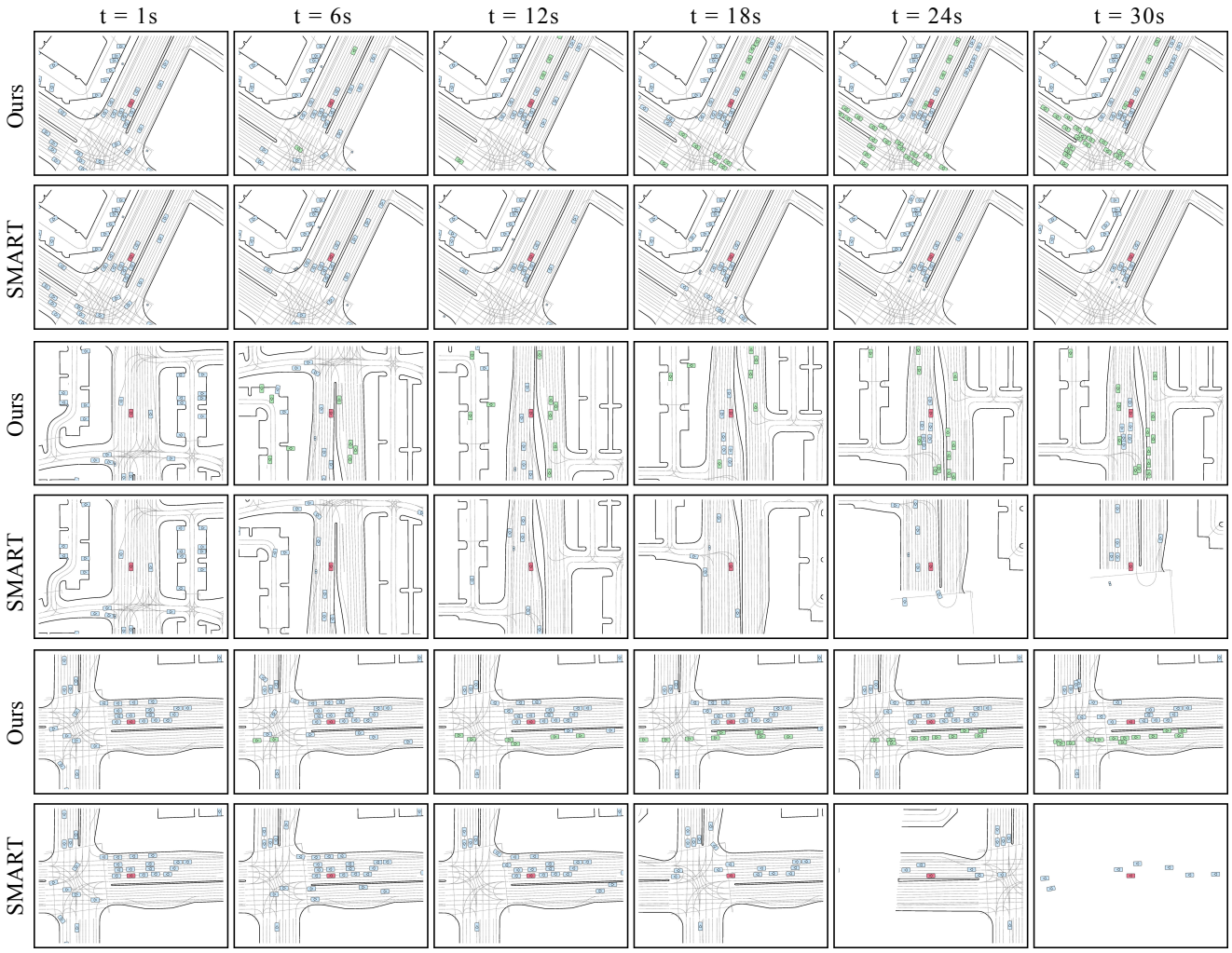

Figure 11. More qualitative comparison results #3.