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

Supplementary Material

Our supplementary contains following contents:

(A) **Demo Video.** We provide more illustrative videos to demonstrate the motivation and demos in Sec. A.

(B) **Model Details.** In addition to the key components introduced, we describe other modules of `InfGen` in Sec. B.

(C) **Token Details.** We have a comprehensive demonstration of tokens design in `InfGen` in Sec. C.

(D) **Training Details.** We explain how we train the interleaved autoregressive model, `InfGen`, in Sec. D.

(E) **Additional Results.** We have more experiments of long-term simulations and ablations in Sec. E.

(F) **Limitations and Future Direction.** We analyze the failure cases in `InfGen`. And further discuss the limitations and the potential improvements in Sec F.

(G) **License.** We list licenses of all assets used in `InfGen`.

A. Demo Video

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 [31] 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. 4.

Note that we also inherit the road network from [31] (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 4. 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, $\texttt{PE}$ 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 4) 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 4). 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 [14] ($n = 91$ 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:

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

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

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

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

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 $\hat{x}^{\text{v}} = \texttt{False}$, 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 $\hat{x}^{\text{v}} = \texttt{False}$, 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 $\hat{x}^{\text{m}}$, heading tokens $\hat{x}^{\text{h}}$, and $\hat{\mathbf{x}}_{\text{invalid}}$, $\hat{\mathbf{x}}_{\text{valid}}$ reflect their corresponding $\hat{x}^{\text{v}}$ 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 [31], 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 $\{\hat{x}^{\text{m}}_k\}_{k=0}^{N-1} \subseteq \mathcal{V}_{\text{motion}}$, validities $\{\hat{x}^{\text{v}}_k\}_{k=0}^{N-1} \subseteq \{0, 1\}$ ($0 =$ `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 $Y^{\text{m}} \in \mathbb{R}^N$ predicted by motion head, we derive the motion mask $M^{\text{m}} \in \mathbb{B}^N$ from the states[3] (Note that masks M are temporally aligned with prediction sequences, not groud-truth sequences). Assuming the step of the BOS and EOS are s_{BOS}, s_{EOS}, then we have:

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

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

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

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

where $\texttt{CE}$ 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 $\{\hat{x}^{\text{c}}_k\}_{k=0}^{N-1} \subseteq$ {`<NULL>`, `<KEEP AGENT>`, `<REMOVE AGENT>`} similar to motion ones. We have described how to derive the control 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: $w($`<KEEP AGENT>`$) = 0.1$ and $w($`<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 $\{\hat{x}^{\text{cs}}_k\}_{k=0}^{L} \subseteq$ {`<NULL>`, `<ADD AGENT>`, `<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: $w($`<ADD AGENT>`$) = 0.1$ and $w($`<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 $\{\hat{\mathbf{x}}_{t-\tau}\}_{\tau=1}^{t_{\text{w}}}$ 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: $M^{\text{ct}}_{\text{hybrid}}[1\!:\!L',\ 1\!:\!A] = \mathbf{0}^{L'\times A}$.
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).

Table 5. 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.

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

E.1. Long-term Traffic Simulation

More Qualitative Results. 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 Curves. 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, 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.

E.2. 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

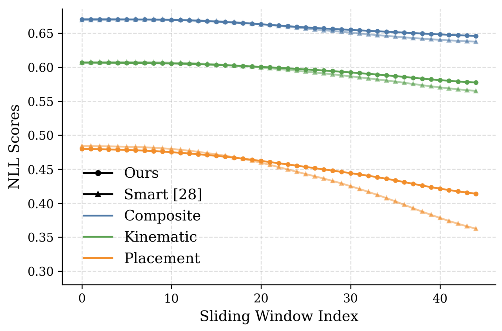

Figure 6. Metrics (adapted WOSAC) curve of `InfGen` against SMART [31] over the 30s long-term simulation rollouts.

cost of local evaluation, following [41], 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 5, 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. 5 also reflects that heading tokens can slightly improve the interactive and placement-based performance.

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 $t = 6\,\mathrm{s}$ and $t = 12\,\mathrm{s}$, 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 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 $0.5\,\mathrm{s}$; 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 $0.5\,\mathrm{s}$ token, and make `InfGen` learn it in temporal simulation. Given that each $0.5\,\mathrm{s}$ 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.

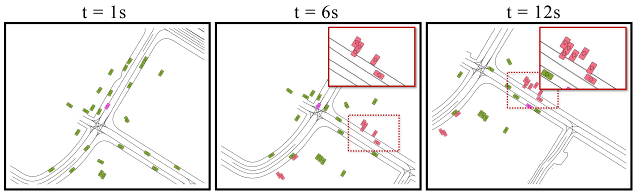

Figure 7. Failure case #1: newly-entered agents appear unreasonably on the boundary of the road.

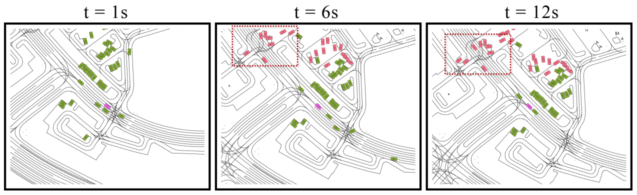

Figure 8. Failure case #2: In the region with complex map structure (highlighted by red box), newly-entered agents fail to correctly infer their initial velocity (or motion) and remain stationary, which is unrealistic.

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 [25, 41] 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 substan-

tial improvement, enabling longer and more flexible simulations. Some concurrent works, such as GPD-1 [34], have some progress in this point, making it a promising direction for near future.

G. License

The Waymo Open Motion Dataset (WOMD) [14] we used in our work is licensed under Waymo Dataset License Agreement for Non-Commercial Use[4].

The implementations of official WOSAC metrics[5] based on which we develop our extended metrics are licensed under the Apache License, Version 2.0.

[4]https://waymo.com/open/terms/

[5]https://github.com/waymo-research/waymo-open-dataset/

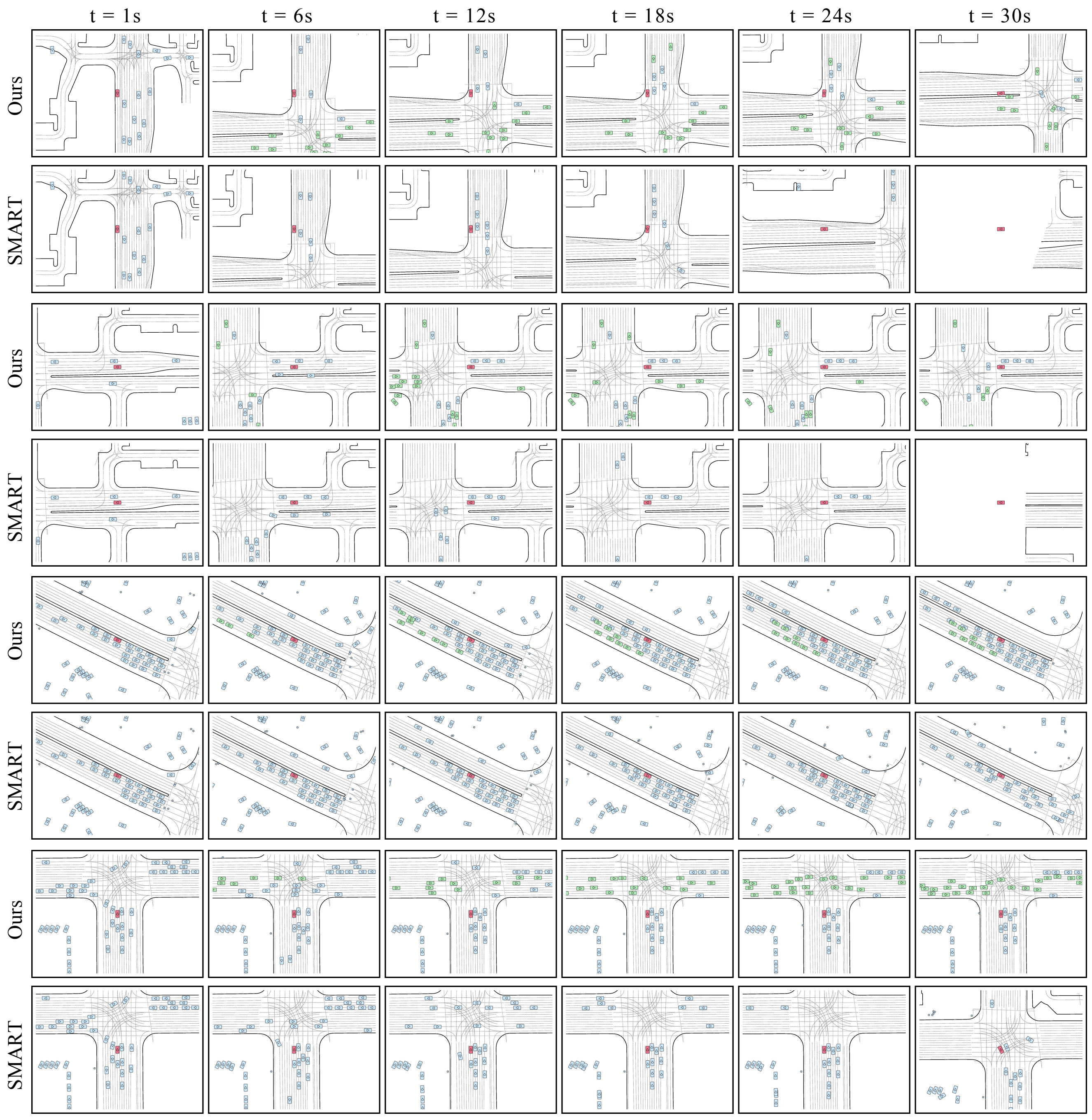

Figure 9. More qualitative comparison results #1.

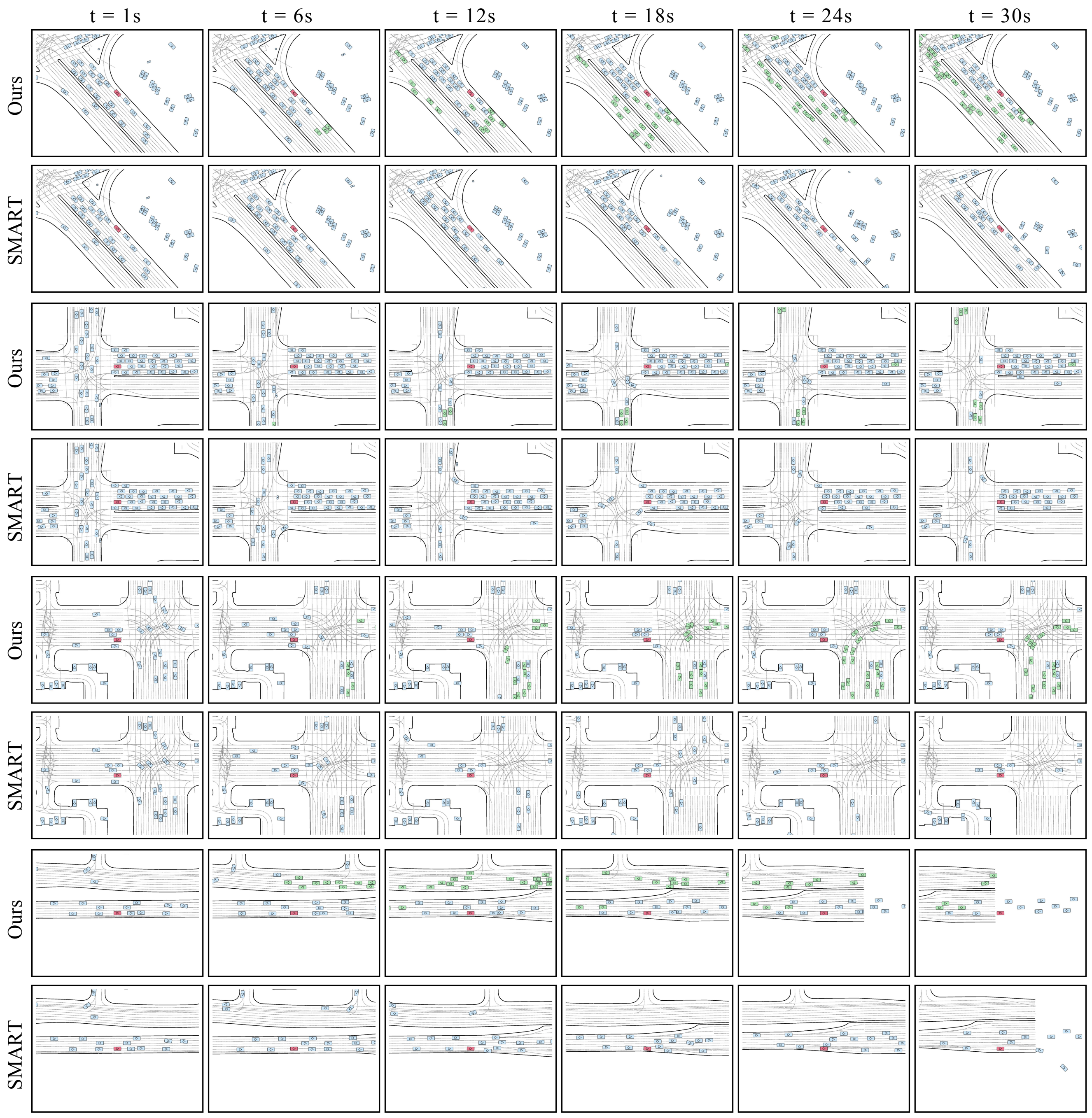

Figure 10. More qualitative comparison results #2.

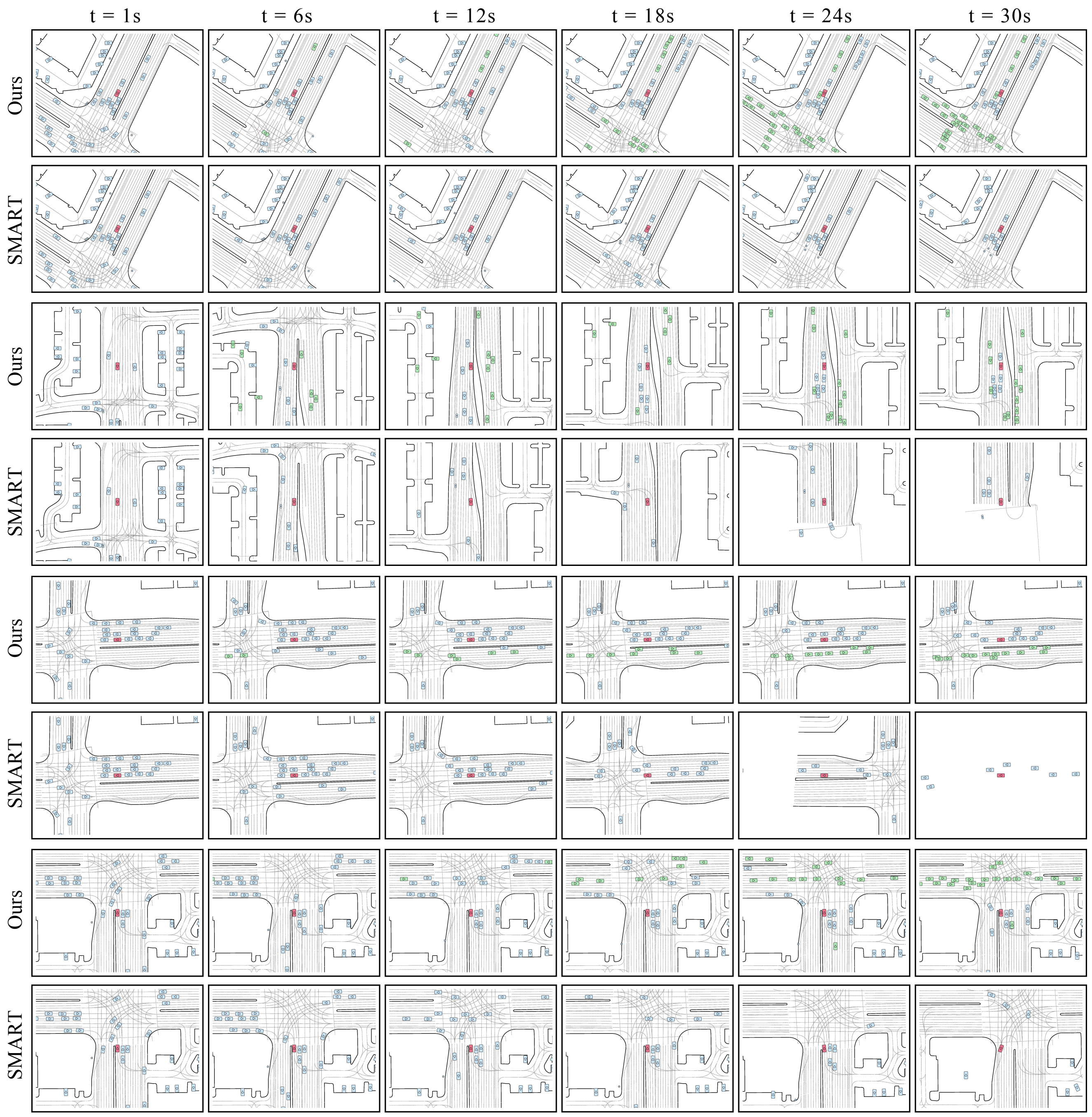

Figure 11. More qualitative comparison results #3.