High-Fidelity Traffic Simulation with Camera Embeddings

Ryan Rong, Jerry Gu, Yina Jian
Stanford University

Abstract

Access to realistic traffic agent scenarios is crucial for the development of autonomous vehicles, improving the safety and efficiency of testing autonomous driving systems. However, creating simulators that generate realistic traffic scenarios matching the complexity and dynamism of the real world remains a significant challenge. Transformer-based next-token prediction models, such as Trajeglish and SMART, have demonstrated success in learning complex agent behaviors from historical trajectories and map data. Yet, these approaches operate within an information bottleneck, inherently lacking the rich visual context of the real environments and details like weather conditions, temporary obstacles, and fine-grained cues like hand gestures of cyclists. To address this limitation, we propose to enhance the capabilities of a pretrained SMART-7M model by integrating camera embeddings from the Waymo Open Motion Dataset 1.2.1. We employ a cross-attention mechanism to effectively fuse the rich visual information with the existing agent and map representations within the transformer decoder, demonstrating superior performance. This multimodal integration is designed to enable the model to generate more contextually aware, realistic, and responsive agent trajectories. We evaluate our model's performance against the baseline, against other state-of-the-art models, and conduct both quantitative and qualitative analyses.

1. Introduction

Simulation has become essential for advancing autonomous vehicle (ADV) development, as it allows testing at scale and with greater reliability. Simply replaying recorded sensor data and tweaking the software is a naïve approach, because it cannot account for how other road users would react to the ADV's altered behavior. Instead, we need "sim agents" whose actions are responsive and realistic, so that they can interact dynamically with the vehicle under test.

There are various strategies for autonomous driving such as cooperative driving, agent feature modeling to characterize individual behaviors, and reinforcement learning under both online/offline or on-policy/off-policy regimes [4]. Cooperative driving combines a human's steering or braking input with a model-generated action, blending them as $\alpha\,\mathbf{a}_{\text{human}} + \beta\,\mathbf{a}_{\text{model}}$. Agent feature modeling, a subfield of agent-based modeling, summarizes each actor's preferences or tendencies via a parameterized policy $\pi_\theta(o) = \mathcal{N}(\mu(o;\theta), \Sigma(o;\theta))$, learned from many simulated or real interactions; the resulting distribution can then inform other agents' decisions. Reinforcement learning formulates a mapping from observed states to control signals (steering, throttle, brake) that respect vehicle dynamics and environmental constraints.

Yet all of these methods ultimately rely on simulating human drivers accurately, which brings us to the Waymo Sim Agents Challenge. The challenge instead treats simulation as a distribution-matching problem: given one second of past trajectories and static map context, we seek a stochastic simulator whose sampled futures match the true, but unknown, distribution of all human driving behaviors. The challenge further factorizes into two autoregressive models: World and ADV. These models are conditionally independent given the state of all the objects in the scene. This conditional independence assumption ensures that the world model could, in principle, be used with new "releases" of the ADV model.

With the success of autoregressive LLMs [1], the field switched to modeling sim-agents as a next-token prediction task, with continuous agent trajectories refactored as discrete motion tokens. State-of-the-art next-token simulators such as SMART [10] and CAT-K [11] discretize agent trajectories and map features into token streams and autoregressively predict future tokens with transformer models. These approaches achieve impressive accuracy on rollouts but omit raw visual inputs, preventing them from capturing subtle, vision-dependent maneuvers such as emergency pull-overs, curbside alignments, and nuanced collision avoidance.

Inspired by single-agent motion predictors such as MoST [5], which show that camera embeddings improve per-agent forecast accuracy, we propose **Camera SMART**. We extract compact, 32-dimensional embeddings from each of Waymo's eight camera views via a pretrained tokenizer

and inject them into SMART's decoder through cross-attention layers. By allowing trajectory tokens to attend to visual features alongside map tokens, our fusion model reduces the realism gap: on the standard Realism Meta-Metric (RMM), we observe a **+1.6% improvement** over the original SMART baseline while still using only 10% of the training data for fine-tuning. Qualitatively, our camera-aware sim agents reproduce off-road pull-overs and curb-side corrections that map-only models frequently mishandle. In sum, this work demonstrates that integrating compact vision into multi-agent next-token simulators can substantially elevate the fidelity of simulated driving behaviors and accelerate the safe rollout of true self-driving systems.

2. Related Work

2.1. Tokenizing for Traffic Simulation

In traffic simulation, tokenization is a technique that aims to convert continuous traffic data, such as positions, velocities, and actions into a more manageable set of discrete actions [12][3]. In Janner et al.'s research, they applied uniform tokenization to traffic data, enabling them to model traffic trajectories with a transformer-based architecture that took in input sequences of tokens [3].

Subsequent work on tokenization for traffic simulation introduced more specialized tokenization methods, including the k-disks algorithm, a modified version of the k-means algorithm that minimizes the overlap of tokens to ensure diverse coverage [7].

2.2. Camera embeddings

Most traffic simulation models use only motion data and exclude other modalities such as camera data from the surrounding environment, limiting real-world performance. Specifically, camera data has been underutilized in traffic simulation due to difficulties dealing with its high-dimensional nature [5]. However, there exist techniques using transformer-based models such as CLIP, DINO, and VQ-GAN to create compact camera embeddings that efficiently integrate visual context[8][6][2]. In 2024, Mu et al. demonstrated that incorporating camera embeddings from various encoders on the Waymo Open Dataset led to a significant performance boost for motion prediction with *single* agents [5]. We focus on incorporating camera embeddings to achieve robust rollouts for *multi-agent* interactions.

2.3. Next-Token Prediction for Traffic Simulation

Trajeglish [7] was the first attempt by researchers to model the complex problem of traffic scenario generation using a next-token based approach. They first tokenized the continuous set of possibilities into a diverse set of 384 discrete potential driving actions using a K-disks algorithm. Then, they applied a encoder-decoder style architecture to predict the action of each agent at a given time step based on all action tokens taken in previous timestep and by earlier agents in the same time step.

Building on earlier next-token-based traffic simulation models, the Scalable Multi-Agent Real-Time Motion Generation via Next-token Prediction (SMART) [10] model was developed to improve generalization ability and efficiency. The researchers behind SMART selected a decoder-only model architecture, which increased inference efficiency by avoiding the costly act of re-encoding previous tokens. Additionally, fine-tuning techniques have been demonstrated to further enhance the realism of models such as SMART. Currently, CAT-K [11] is the state-of-the-art approach. This fine-tuning technique implements a top-k sampling for policy rollout and enforces a distance metric with respect to the ground truth (GT) to simulate realistic scenarios. Notably, these methods do not incorporate camera embeddings.

3. Dataset

The Waymo Open Motion Dataset is provided in proto format including map information, agent trajectories, and occupancy, as well as LiDAR and camera embeddings. Camera information is collected by Waymo's front, front-left, front-right, side-left, side-right, rear-left, rear-right, and rear sensors. Camera embeddings are then generated by Waymo's pretrained VQ-GAN network [5].

Figure 1. Camera Data sample

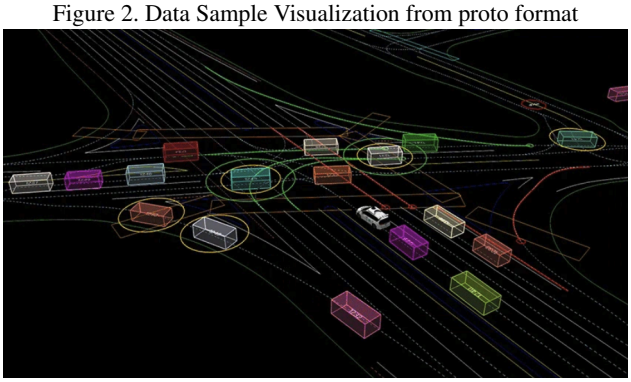
The dataset is split into train(70%), test(15%) and validation(15%) sets. It is composed of $103,354$ segments each containing 20 seconds of object tracks at $10Hz$ and map data for the area covered by the segment. Each segment is further divided into 9-second windows, where 1 second represents the history and 8 seconds represent the future data, as shown below:

Information	Scenario proto	tf.Example
Segment length	9 {1:h, 8:f}	9 {1:h, 8:f}
Maps	Vector maps	Sampled as points
Representation	Single proto	Set of tensors

Among the motion data, all coordinates in the dataset are in a global frame with X as East, Y as North and Z as up, and the unit for displacement is meters.

The Waymo Open Dataset provides scenario and map data in serialized proto format. This format organizes structured information about the driving environment into hier-

Figure 2. Data Sample Visualization from proto format



archical `message` blocks, where each block captures a different aspect of the scene or its dynamics.

A single scenario file contains three primary message types: `ObjectState`, `Track`, and `Scenario`. The `ObjectState` message records the static properties of an object at a given timestep, including its center coordinates, dimensions, heading, velocity, and validity. The `Track` message bundles together a sequence of `ObjectState` entries for a single object over time and includes a unique ID and object type (e.g., vehicle, pedestrian). Finally, the `Scenario` message aggregates the dynamic state of all tracked objects, traffic signal states, static map features, autonomous vehicle index, camera tokens, and other metadata for a complete scene rollout.

The map files are similarly organized under the top-level Map `message`. This contains a list of static `MapFeature` messages (e.g., `LaneCenter`, `RoadEdge`, `RoadLine`, `StopSign`) as well as dynamic state updates captured by the `DynamicState` message. Static elements define persistent road geometry, such as lane boundaries and traffic signs, while dynamic states record time-varying features like traffic signal changes at each timestep.

All spatial elements (e.g., bounding boxes, polyline edges) are defined relative to a local origin and are represented using `MapPoint` coordinates in meters. Temporal alignment across messages is enforced via a shared timestamp array (`timestamps_seconds`), which ensures consistency when retrieving object states, dynamic map states, and sensor data (e.g., LiDAR or camera tokens) at a given timestep.

This structured proto design enables scalable simulation and model training by encoding agent trajectories, interactions, and rich environmental context in a machine-readable, temporally coherent format.

4. Methods

4.1. Linear Extrapolation

As a simple baseline, we linearly extrapolate each agent's 3D position using the last two observations. Denote an agent's position at time $t-1$ and t by $\mathbf{p}_{t-1}, \mathbf{p}_t \in \mathbb{R}^3$, observed at fixed interval Δt. We compute the constant velocity

$$\mathbf{v} \;=\; \frac{\mathbf{p}_t - \mathbf{p}_{t-1}}{\Delta t}\,. \tag{1}$$

Then, the k-step–ahead position is

$$\mathbf{p}_{t+k} \;=\; \mathbf{p}_t \;+\; k\,\Delta t\,\mathbf{v}\,, \quad k = 1, 2, \ldots\,. \tag{2}$$

To generate multiple rollouts, we add zero-mean Gaussian noise $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \sigma^2 I)$ to the velocity at each step:

$$\tilde{\mathbf{p}}_{t+k} \;=\; \mathbf{p}_t \;+\; k\,\Delta t\,\big(\mathbf{v} + \boldsymbol{\epsilon}_k\big)\,. \tag{3}$$

This baseline does not model inter-agent interactions or map constraints.

4.2. SMART Baseline

SMART [10] (Scalable Multi-agent Real-time Motion Generation via Next-token Prediction) is a purely autoregressive, decoder-only model that tokenizes both agent trajectories and map elements into discrete token sequences. These sequences are processed by a hierarchical transformer to predict each agent's next "motion" token.

Agent tokenization. For every agent type (vehicle, pedestrian, cyclist), historical trajectories are split into fixed-length segments of five time steps (0.5 s at 10 Hz). We run K-disk clustering on these segments to define a vocabulary of $V_{\text{agent}} = 2048$ discrete motion tokens. Each token represents the relative $\Delta x, \Delta y, \Delta z$ and heading change over that 0.5 s window.

Map tokenization. Map elements (lanes, crosswalks, stop signs, etc.) are represented by their geometric polylines (sequences of 2D points) and semantic labels. Each element is decomposed into small line-segment fragments and clustered into V_{map} discrete tokens based on orientation and type. The result is a sequence of map tokens per scene.

Model architecture. The SMART decoder has two main modules:

- **Map encoder:** A graph neural network constructs a radius-based connectivity graph over map-element tokens, then applies attention—capturing pairwise spatial relationships. The output is a set of map-feature embeddings.
- **Agent decoder:** At each decoding step, three attention layers are applied in sequence:
  1. Temporal self-attention over each agent's past motion tokens.

2. Map-to-agent cross-attention, which lets the agent attend to relevant map embeddings.
3. Agent-to-agent cross-attention, which models social interactions among agents.

Finally, a feed-forward network predicts the next motion token via softmax over the token vocabulary.

The model is trained end-to-end with a next-token prediction loss (cross-entropy), where the target is the actual next motion token in the ground truth trajectory.

4.3. Cross Attention with Camera Embeddings

To incorporate visual context, we introduce camera-based cross-attention layers into pretrained SMART decoder blocks. In particular, at the second decoder layer (after temporal self-attention), we insert an extra multi-head attention module whose keys and values come from camera features. In terms of code, our contributions include scripts to download and preprocess camera embeddings, Camera-Aware SMART architecture and configs, modified DataLoader pipeline, and modified training, validation, and testing pipelines.

4.3.1 Camera Tokenization and Embedding

Let $\mathcal{C} = \{c_1, \ldots, c_C\}$ denote the set of C onboard cameras (e.g., front, front-left, front-right, side-left, side-right, rear, rear-left, rear-right). For each camera $c \in \mathcal{C}$, denote its raw image at time t by

$$I_t^{(c)} \in \mathbb{R}^{H \times W \times 3},$$

with $H \times W$ pixels. We convert each $I_t^{(c)}$ into a fixed-length embedding through four steps:

1. **Discrete tokenization.** Apply Waymo's pretrained tokenizer

$$\tau : \mathbb{R}^{H \times W \times 3} \longrightarrow \{1, \ldots, V\}^N, \quad N = 256,$$

to yield integer tokens

$$\big(t_{t,1}^{(c)}, \ldots, t_{t,N}^{(c)}\big) = \tau\big(I_t^{(c)}\big), \quad t_{t,i}^{(c)} \in \{1, \ldots, V\}.$$

2. **Codebook embedding.** Let $\mathbf{E} \in \mathbb{R}^{V \times d}$ be a learnable codebook with embedding dimension $d = 32$. Each token index is looked up as

$$\mathbf{e}_{t,i}^{(c)} = \mathbf{E}_{t_{t,i}^{(c)}} \in \mathbb{R}^d, \quad i = 1, \ldots, N.$$

Hence $\{\mathbf{e}_{t,i}^{(c)}\}_{i=1}^N$ are the d-dimensional embeddings for camera c at time t.

3. **Mean pooling.** We compute the average of these N codebook vectors:

$$\mathbf{h}_t^{(c)} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_{t,i}^{(c)} \in \mathbb{R}^d.$$

This yields a single per-camera, 32-dimensional vector.

4. **Learned projection.** Finally, map $\mathbf{h}_t^{(c)}$ into the transformer hidden dimension $D = 128$ via

$$\mathbf{z}_t^{(c)} = \mathbf{W}_p \, \mathbf{h}_t^{(c)} + \mathbf{b}_p \in \mathbb{R}^D,$$

where $\mathbf{W}_p \in \mathbb{R}^{D \times d}$ and $\mathbf{b}_p \in \mathbb{R}^D$ are learned. The set $\{\mathbf{z}_t^{(c)}\}_{c=1}^C$ forms the $C \times D$ camera feature matrix at time t.

4.3.2 Cross-Attention Layer

In decoder layer 2 of SMART, let $H_t^{(\ell)} \in \mathbb{R}^{L \times D}$ be the sequence representation after self-attention at time t (with token sequence length L). We then apply a camera-based multi-head cross-attention:

$$H_t^{(\ell+1)} = \mathrm{LayerNorm}\Big(H_t^{(\ell)} + \mathrm{MHCA}\Big(Q = H_t^{(\ell)}, \; K = F_t^{(\mathrm{cam})}, \; V = F_t^{(\mathrm{cam})}\Big)\Big). \tag{4}$$

where $F_t^{(\mathrm{cam})} = [\,\mathbf{z}_t^{(1)}; \mathbf{z}_t^{(2)}; \ldots; \mathbf{z}_t^{(C)}\,] \in \mathbb{R}^{C \times D}$ is the stacked camera-embedding matrix at time t. The output $H_t^{(\ell+1)}$ continues through the feed-forward network as in SMART.

At training time, we load the pretrained SMART weights and freeze all original parameters except:

- The new projection weights $\{\mathbf{W}_p, \mathbf{b}_p\}$ for all cameras,
- The added multi-head cross-attention weights in decoder layer 2.

We fine-tune on 10% of the Waymo scenarios, using the standard next-token cross-entropy loss.

5. Evaluation

The simulated agent behaviors are evaluated by a Realism Meta Metric (RMM). This RMM is derived by aggregating several component metrics, with each component metric quantifying realism by calculating the Negative Log Likelihood (NLL) of observed real-world driving behaviors given the distribution induced by the simulation model. These NLL-based component metrics are computed over specific measurements in three categories: kinematic, interative, and map-based metrics.

5.1. Kinematic Metrics

To quantify physical realism we follow the official Waymo Open Dataset Sim Agents evaluation protocol. At every discrete time-step t (with fixed interval Δt) we record the agents' linear speed $v(t)$, linear acceleration $a(t)$, angular speed $\omega(t)$, and angular acceleration $\alpha(t)$ and place each scalar sequence into a fixed-width histogram:

Table 1. Kinematic Metrics and their binning.

Quantity	Range	Bins	Weight
v	$[0,\ 25]$	10	0.05
a	$[-12,\ 12]$	11	0.5
ω	$[-0.628,\ 0.628]$	11	0.05
α	$[-3.14,\ 3.14]$	11	0.05

All timesteps are treated as i.i.d. samples. Laplace (additive) smoothing with $\varepsilon = 0.1$ prevents zero-bin issues. For each kinematic variable we compute the symmetrised Kullback–Leibler divergence between the agent's empirical histogram $\hat{p}$ and the reference distribution $p^\star$ provided by Waymo. The four divergences are then combined via a weighted sum (weights listed above) to yield the overall kinematic score $\mathcal{M}_{\text{kin}}$.

5.2. Interactive Metrics

Interactive safety is captured by 1) distance-to-nearest-object $d_{\min} = \min_{B \neq A} \|\mathbf{p}_A(t) - \mathbf{p}_B(t)\|_2$, 2) time-to-collision TTC: analytical time until Minkowski-sum overlap assuming constant velocities; clamped at 5 s, and 3) a binary collision flag $\mathbb{I}\big(\text{bbox}_A(t) \cap \text{bbox}_B(t) \neq \varnothing\big)$. For the two continuous signals we form fixed–width histograms (again with $\varepsilon = 0.1$ smoothing; i.i.d. timesteps); the Bernoulli channel is scored with binary cross-entropy. Meta-metric weights follow the public Sim-Agents specifications:

Table 2. Interactive metrics and their binning.

Quantity	Range	Bins	Weight
$d_{\min}$	$[-5,\ 40]$	10	0.10
TTC	$[0,\ 5]$	10	0.10
Collision flag	–	Bernoulli	0.25

5.3. Map-based Metrics

Road-rule compliance is evaluated with distance-to-road-edge $d_{\text{edge}}(t) = \min_{\mathbf{q} \in P_{\text{edge}}} \|\mathbf{p}_A(t) - \mathbf{q}\|_2$, a binary off-road indicator $\mathbb{I}(\mathbf{p}_A(t) \notin \text{drivable})$, and traffic-light violation. Histogram and Bernoulli channels are handled exactly as in Sec. 5.2; weights are set by the benchmark.

5.4. Minimum Average Displacement Error

Aside from the RMM, minADE is a tie-breaker metrics that measures prediction accuracy. For M agents in group G, K multi-modal predictions ($S^G_{i,j,t}$), and ground truth ($S^{G*}_{j,t}$), over T timesteps:

Table 3. Map-based metrics and their binning.

Quantity	Range	Bins	Weight
d_{edge}	$[-20,\ 40]$	10	0.05
Off-road flag	Bernoulli	–	0.25
TL violation flag	Bernoulli	–	0.05

$$\text{minADE}(G) = \frac{1}{M} \sum_{j=1}^{M} \left(\min_{1 \leq i \leq K} \left(\frac{1}{T} \sum_{t=1}^{T} \| S^G_{i,j,t} - S^{G*}_{j,t} \|_2 \right) \right)$$

6. Experiments

6.1. Hyperparameters

For our camera-aware SMART model, we carefully selected hyperparameters to balance training efficiency and model performance. We initialized the model with a pretrained SMART checkpoint and employed a finetuning strategy with a learning rate of 5×10^{-5} and a learning rate multiplier of 0.1 for pretrained weights to prevent catastrophic forgetting. The model was trained with mixed precision (16-bit) to optimize memory usage while maintaining numerical stability. We used a small batch size of 4 for both training and validation to accommodate the memory overhead from camera embeddings and cross-attention layers. The training process spanned 50 epochs with validation checks every 5 epochs, using gradient clipping at 1.0 to prevent exploding gradients. For the camera-aware components, the given camera embedding dimension is 32 and we strategically placed cross-attention layers at the second decoder block to allow the model to effectively integrate camera information while preserving the core SMART architecture. We employed a learning rate warmup of 500 steps followed by a cosine decay schedule over 10,000 total steps, with a minimum learning rate ratio of 0.1.

Table 4. Preliminary comparison of simulation methods on the Sim-Agents. Higher is better for RMM and its sub-scores (↑), lower is better for minADE (↓). Δ(%) is relative to SMART.

Method	RMM ↑	ΔRMM (%)	Kinematic ↑	ΔKin (%)	Interactive ↑	ΔInt (%)	Map ↑	ΔMap (%)	minADE ↓	ΔminADE (%)
Camera SMART (ours)	**0.7769**	+1.57%	0.4799	−1.30%	**0.8237**	+2.46%	**0.8862**	+1.39%	2.2600	+64.78%
SMART [10]	0.7649	—	**0.4862**	—	0.8039	—	0.8741	—	**1.3716**	—
Linear extrapolation	0.3985	−47.90%	0.2253	−53.66%	0.4327	−46.18%	0.3848	−55.99%	7.5148	+447.81%

6.2. Results

Although we pretrained the full SMART-7M model on 100% of the scenario data, the camera embeddings is large (around 2.5 TB), so we only finetuned the cross attention model on 10% of the data and tested it on 10% of the data. Table 4 shows the performance of our model against the baseline models.

6.3. Quantitative Analysis

Our camera-aware SMART variant improves the Realism Meta-Metric (RMM) to $\mathbf{0.7769}$, a $\mathbf{1.6\%}$ improvement over the original SMART baseline (Table 4). This lift aligns with the Sim-Agents rubric, which weights interactive and map-based metrics more than pure kinematics (refer to tables 1 2 3) The *interactive* and *map* sub-scores increase by $+2.46\%$ and $+1.39\%$, respectively, supporting the hypothesis that camera texture helps the agent anticipate merges, yields, and lane adherence.

The slight drop in the kinematic NLL (-1.3% relative) could be due to image-induced over-responsive braking. Adding a physics-guided residual head is therefore a promising next step.

Accuracy vs. diversity. minADE deteriorates from $1.37 \rightarrow 2.26$. Because minADE rewards the single best hypothesis among $K{=}32$ samples, any sampler that *widens* mode coverage will see this metric rise—a trade-off noted in the official challenge write-up [9].

Position in the SoTA landscape. Table 5 contrasts our system with recent publications:

Table 5. Comparison with recent Sim-Agents models.

Method	RMM ↑	minADE ↓
CAT-K [11]	0.7846	1.3065
Camera SMART (ours)	0.7769	2.26
SMART [10]	0.7649	1.3716

Despite being fine-tuned on only 10% of the data and retaining the compact 7 M-parameter SMART backbone, our model achieves comparable performance with state-of-the-art, while outperforming our baseline model SMART on the RMM metrics. CAT-K employs additional finetuning based on distance from the ground truth, but camera embeddings provide addition context. So we expect adding camera embeddings to CAT-K would boost its performance too. The result underscores the leverage gained from visual context even with limited finetuning.

6.4. Qualitative Analysis

We visualize two representative scenarios for comparison. In the visualizations for the SMART and Camera SMART models, the ground truth trajectories are translucently overlayed.

In Scenario 02 (Figure 3), which appears to be an intersection on a major street, there are two notable interactions:

Top-right pair of cars: In the ground truth, the rightmost vehicle pulls off the road next to another car that is already parked off-road—perhaps indicating an emergency-light situation or an inspection. Our model rollout replicates this behavior correctly. In contrast, the baseline model stops the leftmost car directly behind its neighbor, resulting in a slight overlap and collision. We hypothesize that our camera-aware agent uses visual cues—such as the curb's geometry or flashing lights—to recognize that it should pull over, rather than stopping immediately behind. This extra context helps it execute the off-road maneuver more precisely.

Tailgating behavior in the top lane: The ground truth shows one car closely following another (tailgating). In our rollout, the following car actually collides with the lead vehicle, and the baseline behaves similarly, although less pronounced. This collision highlights a shortcoming in our kinematic modeling, causing it to fail to brake in time during this edge-case scenario.

In Scenario 04 (Figure 4), set in a residential parking area, the ground truth shows a car veering slightly off the road before correcting and exiting to the right. Our rollout successfully reproduces this subtle off-road deviation and return. The baseline, however, veers significantly farther off the road and fails to drive back onto the lane. This behavior explains our model's higher map-based score: by leveraging visual context, the camera-aware agent "sees" the drivable boundaries and corrects itself, whereas the baseline lacks sufficient information to stay on-road.

7. Conclusion

We presented a camera-aware extension to SMART, a scalable multi-agent traffic simulator built on next-token prediction. By fusing compact visual embeddings from

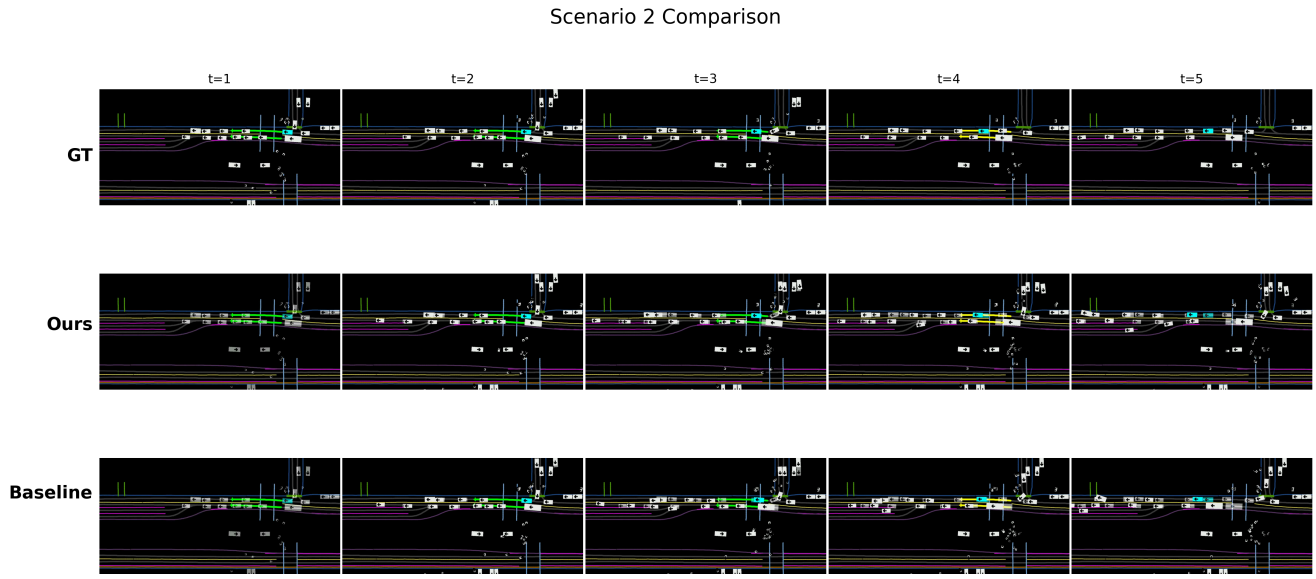


Figure 3. Visualization of sample rollouts of Scenario 02 across methods. GT is the ground truth. Each timestamp is evenly spaced.

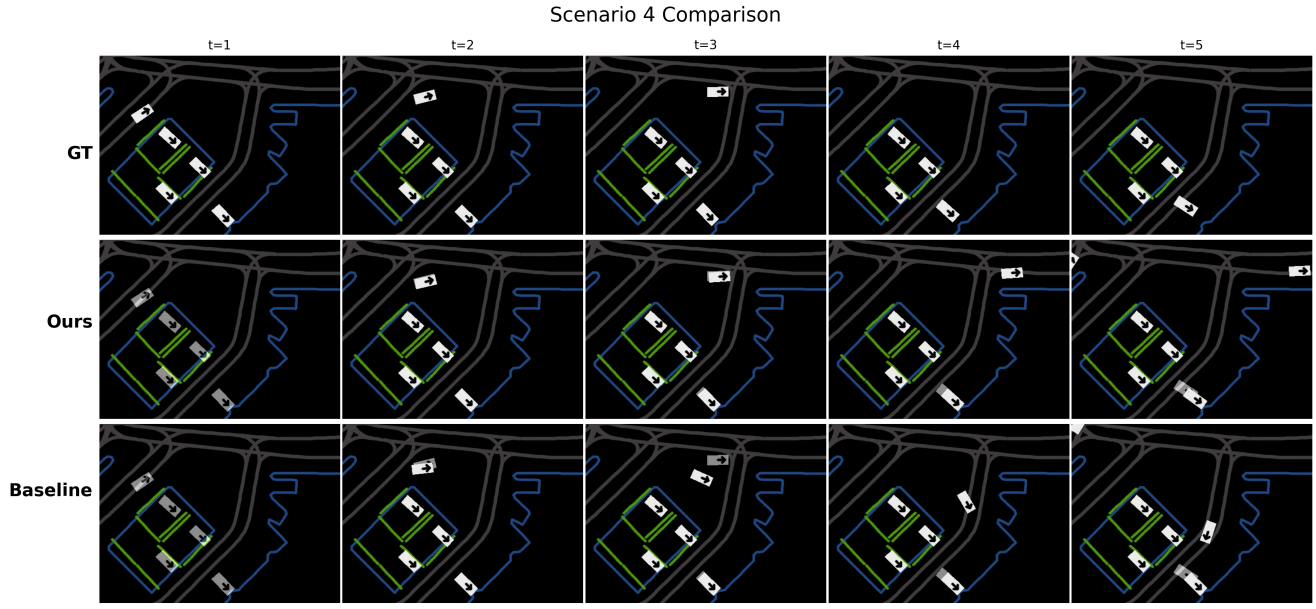


Figure 4. Visualization of sample rollouts of Scenario 04 across methods. GT is the ground truth. Each timestamp is evenly spaced.

Waymo's multi-camera system into the SMART decoder via cross-attention, our model improves realism metrics by up to 2.5% in key subcategories like interaction and map compliance. These gains suggest that even lightweight visual context—when properly integrated—can bridge critical gaps in modeling nuanced driving behaviors such as emergency pull-overs, curb alignments, and off-road recovery.

Our method achieves these improvements while fine-tuning only a small portion of the parameters and using just 10% of the training data. This demonstrates that vision-equipped simulation can scale efficiently and complements structure-aware baselines without needing wholesale retraining. Although minADE rises due to broader mode coverage, this is an expected trade-off when sampling more diverse futures in a multimodal setting.

Moving forward, our results indicate promising avenues such as applying vision-informed attention to CAT-K, incorporating LiDAR as additional modalities, and developing strategies to balance diversity and precision in trajectory sampling. Overall, our approach brings simulation one step closer to real-world fidelity—an essential milestone for

safely deploying autonomous systems at scale.

Contributions

Ryan Rong conceived the project idea and led the development of the overall research framework. He set up the codebase for the data pipeline, managed model training, designed the model architecture, produced figures, wrote the methods and results sections and edited the report.

Jerry Gu conducted a comprehensive literature review to support the theoretical framework of the project. He contributed to setting up the baseline models and also assisted in refining and documenting the experimental process.

Yina Jian summarized the formulas, frameworks, and context of state-of-the-art methods in automatic driving, analyzed existing methods, delved deeply into the details of the data used in the project, guided the theoretical part of the multi-agent driving problem, and wrote the introduction and dataset section of the report.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. 1

[2] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, 2021. 2

[3] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 34, pages 1273–1286, 2021. 2

[4] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, D. Anguelov, and S. Levine. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios, 2023. 1

[5] N. Mu, J. Ji, Z. Yang, N. Harada, H. Tang, K. Chen, and Y. Zhou. MOST: Multi-Modality Scene Tokenization for Motion Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14988–14999, 2024. 1, 2

[6] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Bojanowski, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 2

[7] J. Philion, X. B. Peng, and S. Fidler. Trajeglish: Traffic modeling as next-token prediction. *arXiv preprint arXiv:2312.04535*, 2023. 2

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 8748–8763. PmLR, 2021. 2

[9] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. *CoRR*, abs/1912.04838, 2019. 6

[10] W. Wu, X. Feng, Z. Gao, and Y. Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 114048–114071, 2024. 1, 2, 3, 6

[11] Z. Zhang, P. Karkus, M. Igl, W. Ding, Y. Chen, B. Ivanovic, and M. Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. *arXiv preprint arXiv:2412.05334*, 2024. 1, 2, 6

[12] J. Zhao, J. Zhuang, Q. Zhou, T. Ban, Z. Xu, H. Zhou, J. Wang, G. Wang, Z. Li, and B. Li. Kigras: Kinematic-driven generative model for realistic agent simulation, 2024. 2