## Appendix

# A Related Works

## A.1 Scene Generation

As a supplement to real data, generated scenarios establish the initial conditions for reactive simulations. Previous efforts have involved the creation of agents guided by heuristic rules [16, 46, 57, 82] or fixed grammars [15, 38]. However, the manual generation of such scenarios requires extensive human labor and struggles to achieve the necessary realism, diversity, and generalization for downstream tasks, especially in new environments. Recent work has started to adopt an end-to-end fully learned approach, either through an autoregressive architecture that places agents one by one [19, 73] or a diffusion-based architecture [58]. Yet, no work has exploited a more scalable or unified transformer-based language model for controllable scene generation. Moreover, our method allows for a coarse control over the entire scene through a global description prompt and a fine control by directly modifying each agent's states, e.g. enforcing traffic rule constraints. These features enable our model to strike a better balance between diversity and controllability throughout the generation process.

## A.2 Reactive Simulation

Traditional simulators are built on computer graphics and human prior knowledge, including physical laws, lighting conditions, and hand-crafted traffic dynamics [2, 16, 41, 46]. However, relying on a vast collection of manually created scene assets and heuristic driving policies for intelligent agents not only introduces a significant domain gap but also results in a lack of diverse scenarios and realism.

Recently, there has been a growing emphasis on building world simulators as generative models through a data-driven approach [22, 24, 31, 42, 62, 81, 83]. Compared to those end-to-end models, structured input provides a simpler and more efficient setting, facilitating a variety of downstream applications and onboard deployments. Predictive models, based on the unfolding of agents' states over time, fall into open-loop and closed-loop categories. In an open-loop setting, each agent makes decisions—either marginally [21, 34, 67, 76] or jointly [33, 52, 68]—based solely on historical information and generates predictions for their trajectory over a short future period. In contrast, closed loop simulations incorporate feedback mechanisms, where the decisions of each agent are influenced by the current state of the system, including the actions of other agents. This leads to a more dynamic, interactive and realistic simulation environment, where agents continuously update their decisions based on the evolving scenario, such as [65, 71]. Notably, WOSAC [50] established the first evaluation benchmark for closed-loop simulators, attracting numerous participants [51, 56, 59, 78]. As a closed-loop simulator, our work is most related to MotionLM [65] and Trajenglish [56], as both employ a GPT-like autoregressive predictive model. However, in contrast to them, which tokenize the action space, we employ a "key-value pair" tokenization strategy and directly quantize the state space. Through this

method, we can achieve Non-Autoregressive (NAR) transitions within frames, significantly speeding up inference, and endowing the model with generative capabilities. Moreover, it has the flexibility to handle the disappearance and emergence of agents. These capabilities foster a wider range of downstream applications, which require higher scalability, efficiency, and flexibility.

### A.3 Interactive Planning

The ability to effectively model the interactions between road users and autonomous vehicles is crucial for enhancing the safety and comfort of self-driving technology. An optimal policy planning algorithm should enable multi-stage reasoning, incorporating bidirectional interactions between the agent and it's environment. Previous research has tackled this challenge through two main approaches: Some studies have employed neural networks to implicitly and iteratively capture the interactions between the ego vehicle and other road users [9, 34, 36, 66, 67]. Others have taken a more explicit route, utilizing model predictive control (MPC) in conjunction with tree search expansion to navigate complex interactions [7, 8, 35, 74]. Among these approaches, certain studies [7, 8, 35] have simplified the modeling of bidirectional interaction by combining a simple non-reactive rule-based planner with a model-based, ego-conditioned predictor. In contrast, PDM [14] simplifies interaction by assuming a non-reactive environment with constant velocity agents, and a reactive Intelligent Driver Model (IDM)-based [75] ego-planner. Our work advances beyond these methodologies by coupling our realistic ego-conditioned simulator with a reactive planner, adaptable to any rule-based or neural network-driven ego-planner. This approach allows for a more accurate modeling of interactions and the stochastic nature of future scenarios.

### A.4 Online Learning with RL

The reinforcement learning (RL) domain has witnessed considerable progress, fostering algorithms adept at solving various types of tasks via mode-free [23, 64] and model-based approaches [24, 25]. The progress in RL has contributed to the paradigm shift in autonomous driving from open-loop trajectory prediction-based approach to closed-loop interaction-based approach. One critical component to enable closed-loop training is a reactive environment, generating the state of the next time step given the current state and action taken [5, 20, 30, 55]. Nevertheless, the simplicity of these models often results in environments that fall short of realism and interactivity, leading to a pronounced simulation-to-reality gap in complex scenarios. Addressing this, some research focuses on the creation of more realistic and intelligent training environments [19, 41]. Different from previous approaches, we introduce a scalable, data-driven model capable of imitating interactive agent behaviors and generating realistic, diverse driving scenarios. Its efficiency aligns well with the demands of online RL training. Such an environment could bridge the simulation-reality gap, and push RL closer to practical application in autonomous driving.

# B Dataset and Metrics

## B.1 Dataset

**The WOD Motion Dataset** We have utilized the Waymo Open Motion Dataset (WOMD) to train and evaluate for scene generation, reactive simulation and motion prediction tasks. WOMD contains 7.64 million unique tracks from 574 driving hours across 1750 km urban roadways collected in six cities of the United States. Specifically, the WOMD v1.2.0 release includes 486,995 train, 44,097 validation, and 44,920 test scenarios. Each scenario consists a 9.1 seconds 10 Hz driving log, partitioned into 11 history frames and 80 future frames. All reported variants of our model is trained on the full training dataset. To further augment the data volume, we treat different objects as the center object, ultimately yielding approximately 2.6 million training scenarios.

**The nuPlan Dataset** We utilized the nuPlan Dataset [2] for scenario prompt conditioned scene generation, interactive planning and online training experiments. The dataset encompasses 1,500 hours of diverse driving scenarios from urban environments such as Las Vegas, Boston, Pittsburgh, and Singapore. These scenarios are annotated with various traffic situations including merges, lane changes, interactions with cyclists and pedestrians, among others. For training, a random sampling strategy is adopted, with 50,000 scenarios selected for each scenario type from the training subset, culminating in approximately 1.5 million scenarios.

## B.2 Metrics

**Scene Generation Metrics** To measure the quality of generated scenarios, we treat the generated agents as samples from a distribution, and compare it with ground-truth agents, which are also treated as samples from another distribution. Similar to Trafficgen [19], we use maximum mean discrepancy (MMD) as the metric to measure the difference between the generated distribution and the corresponding ground-truth distribution. Given two distributions $p$ and $q$ with Gaussian kernel $k$, the maximum mean discrepancy is defined as

$$MMD^2(p,q) = \mathbb{E}_{x,x' \sim p}[k(x,x')] + \mathbb{E}_{y,y' \sim q}[k(y,y')] - 2\mathbb{E}_{x \sim p, y \sim q}[k(x,y)] \quad (9)$$

We calculate this metric for generated attributes including agent center positions in $\mathbb{R}^2$, agent heading angle in $\mathbb{R}$, agent velocities in $\mathbb{R}^2$, and agent dimensions in $\mathbb{R}^2$.

For each pair of generated and ground-truth scenarios, we calcuate MMD for each attributes independently and compute the average value of them. And the final MMD score is averaged across all selected testing dataset.

**The WOD Sim Agents Metrics** We evaluate our model's simulation capability on the WOMD Sim Agents Benchmark [50]. Each testing scenario consists a 9.1 seconds 10 Hz driving log, partitioned into 11 history frames and 80 future frames. Trajectories of up to 128 agents (including ego vehicle) are tracked in each scenario. The task is to simulate 32 parallel future rollouts for each agent and scenario in an autoregressive manner. Given an agent in a scenario, the predicted distribution of its future behavior is formed over the 32 rollouts. In order to parameterize the predicted distribution, a total of 3 categories and 9 component metrics are computed:

- **Kinematic-based:**
  - **Linear Speed** unsigned magnitude of agent's speed in x, y, z axis.
  - **Linear Acceleration Magnitude** derivative of linear speed with respect to time in x, y, z axis.
  - **Angular Speed** signed minimum difference between consecutive angular heading
  - **Angular Acceleration Magnitude** derivative of angular speed with respect to time
- **Interaction-based:**
  - **Distance to nearest object** for each agent represented by a box polygon, the signed distance from the nearest object in the scenario.
  - **Collisions** Count of collisions with other objects, recognized when **Distance to nearest object** is negative.
  - **Time-to-collision** Time takes before an agent collide with the agent it is following, assuming constant speed.
- **Map-based:**
  - **Distance to road edge** Signed distance to the nearest road edge, where a road egde is represented as a 2D vector.
  - **Road departures** Indicator of an agent going off road at any time.

A component score is calculated by testing the Negative Log Likelihood of the ground truth agent behavior over the corresponding distribution, masked by validity $v_t$: $m = \exp\Big(-\frac{1}{\sum_t \mathbb{1}\{v_t\}} \sum_t \mathbb{1}\{v_t\} NLL_t\Big)$. Each component metric is assigned a weight, and each categorical metric is computed by taking the weighted average of all corresponding component metrics: $\mathcal{M}_c = \frac{\sum_i w_i m_i}{\sum_j w_j}$. We report all categorical metrics on the test dataset in Tab. 13. The final composite metric is calculated as a weighted average over all component metrics. Our reported model performance is based on the V1 Leaderboard, which improved collision and offroad calculation from the previous V0 Leaderboard.

**The nuPlan Planning Metrics** Our evaluation concentrates on the model's planning abilities in closed-loop with reactive agents (CL-R) task. In these setups, the planner generates a trajectory at each timestep, which is used as a reference by the controller to incrementally adjust the vehicle's state. In CL-R

tasks, we apply two distinct world models for the surrounding agents: a rule-based IDM environment and a model-based environment powered by GUMP, both controls all non-ego agents dynamically.

The planning score, as defined by the equation (5), is reported on on the test dataset. We exactly follow the nuPlan benchmark metrics [2], which reflects the model's performance across safety, efficiency, and comfort metrics in CL-R tasks. This score is aggregated by selected metrics for the driven trajectory generated from the planner. The aggregation function is a hybrid hierarchical-weighted average function of individual metric scores.

The first part $\Theta$ of equation (5) gets a zero score for a driven scenario if any one of those following situations happens.

- There is an at_fault collision with a vehicle or a VRU (pedestrian or bicyclist);
- There are multiple at_fault collisions with objects (e.g. a cone);
- There is a drivable_area violation;
- Ego drives into uncoming traffic more than 6 m;
- Ego is not making enough progress.

The second part $\Phi$ of equation (5) is a weighted average of other selected metrics' scores. Those selected metrics and their corresponding weight are listed as following.

| Metric Name | $\omega_n$ |
|---|---|
| driving_direction_compliance | 5 |
| time_to_collision_within_bound | 5 |
| speed_limit_compliance | 4 |
| ego_progress_along_expert_route | 5 |
| Ego_is_comfortable | 2 |

**Table 6:** Selected metrics for the second part $\Phi$ of equation (5) .

The following component metrics contribute to the overall planning score:

- **No at-fault Collisions** A collision is defined as the event of ego's bounding box intersecting another agent's bounding box. To define the collision score for a scenario, we only consider collisions that should have been prevented if planner performed properly. For simplicity, we call these collisions at-fault.
- **Drivable area compliance** Ego should drive in the mapped drivable area at all times. Drivable area compliance metric identifies the frames when ego drives outside the drivable area. Due to over-approximation of ego's bounding box, we allow for a small infringemenet outside the drivable area (max_violation_threshold = 0.3m).
- **Driving direction compliance** This metric is defined to penalize ego when "it drives into oncoming traffic". The metric computes the movement

of ego's center during a 1 second time_horizon along the driving direction defined according to the baselines of ego's lanes or lane-connectors. The score is set to 1 if it does not drive/move against the flow more than driving_direction_compliance_threshold (= 2 m) and 0 if it drives against the flow more than driving_direction_violation_threshold (= 6 m), and 0.5 otherwise.
- **Ego progress along the expert's route ratio** This metric is used to evaluate progress of the driven ego trajectory in a scenario by comparing its progress along the route that expert takes in that scenario.
- **Making progress** This metric is defined as a boolean metric based on the "Ego progress along the expert's route ratio". It's score is set to 1 if the ratio is more than the selected threshold (min_progress_threshold = 0.2), and is set to 0 otherwise.
- **Time to Collision (TTC) within bound** TTC is defined as the time required for ego and another track to collide if they continue at their present speed and heading.
- **Speed limit compliance** This metric evaluates if ego's speed exceeds the associated speed limit in the map. Speed limit violation at each frame is defined based on the difference between ego's speed and the speed limit, if ego's speed is higher than the speed limit (over-speeding).
- **Comfort** We measure the comfort of ego's driven trajectory by evaluating minimum and maximum longitudinal accelerations, maximum absolute value of lateral acceleration, maximum absolute value of yaw rate, maximum absolute value of yaw acceleration, maximum absolute value of longitudinal component of jerk, and maximum magnitude of jerk vector. These variables are compared to thresholds with default values determined empirically from examination of a dataset of expert trajectories.

**RL metrics** We define safe episodes as those without any critical failures, such as collisions and violations of the drivable area. In the context of RL training, we utilize four specific types of metrics that differ slightly from the nuPlan Planning Metrics mentioned previously in B.2. These metrics include:

- **Collision Rate** We do not differentiate between at-fault and non-at-fault collisions, setting a higher standard for the planner to avoid all collisions, whether by not colliding with others or by not being collided into. The collision rate is calculated as the percentage of episodes that experience at least one collision.
- **Out of Drivable Area Rate** Diverging from the nuPlan metric, our approach disallows any encroachments outside the drivable area, demanding strict adherence from the planner. The score for drivable area compliance is determined by the percentage of episodes where the drivable area is violated.
- **Progress** We define a series of equally spaced waypoints along the planned route. Progress is measured by the physical distance the ego vehicle covers towards the next waypoint at each step. The overall progress score represents the average progress made in all safe episodes.

– **Comfort** This metric maintains the same components as defined in the nuPlan Planning Metrics. The comfort score is the proportion of episodes deemed comfortable out of all safe episodes.

The final score calculation incorporates Equation (5). For this equation, the $\Theta$ component comprises binary metrics from collision and out-of-drivable-area violations, signifying that these metrics are either 0 (no violation) or 1 (violation occurred). As for the $\Phi$ component, it is determined by first calculating the progress percentage. This is achieved by dividing the actual progress made by the vehicle by an empirical value of 62, which represents the average progress length according to the nuPlan dataset. Subsequently, the average of this progress percentage and the comfort metric is calculated to derive the $\Phi$ value.

**WOD motion metrics** WOMD's metrics are used to evaluate the motion prediction performance of our model.

– **minADE** The minimum Average Displacement Error computes the L2 norm between groundtruth and the closest joint prediction.
– **minFDE.** The minimum Final Displacement Error is equivalent to evaluating the minADE at a single time step T.
– **Overlap rate (OR)** The overlap rate is computed by taking the highest confidence joint prediction from each multimodal joint prediction. If any of the A agents in the jointly predicted trajectories overlap at any time with any other objects that were visible at the prediction time step (compared at each time step up to T) or with any of the jointly predicted trajectories, it is considered a single overlap. The overlap rate is computed as the total number of overlaps divided by the total number of predictions. See the supplementary material for details. The overlap is calculated using box intersection, with headings inferred from consecutive waypoint position differences.
– **Miss rate (MR)** A binary match/miss indicator function ISMATCH(ˆst, st) is assigned to each sample waypoint at a time t. The average over the dataset creates the miss rate at that time step. A single distance threshold to determine ISMATCH is insufficient: we want a stricter criteria for slower moving and closer-in-time predictions, and also different criteria for lateral deviation (e.g. wrong lane) versus longitudinal (e.g. wrong speed profile)
– **Mean average precision (mAP)** The Average Precision computes the area under the precision-recall curve by applying confidence score thresholds ck across a validation set, and using the definition of Miss Rate above to define true positives, false positives, etc. Consistent with object detection mAP metrics, only one true positive is allowed for each object and is assigned to the highest confidence prediction, the others are counted as false positives. Further inspired by object detection literature, we seek an overall metric balanced over semantic buckets, some of which may be much more infrequent (e.g., u-turns), so report the mean AP over different driving behaviors. The final mAP metric averages over eight different ground truth trajectory shapes: straight, straight-left, straight-right, left, right, left u-turn, right u-turn, and stationary.

# C Experimental setup

## C.1 Scene Generation

For our quantitative experiments in scene generation, we utilized all the validation scenarios from the Waymo Open Motion Dataset (WOMD) version 1.2.0. To ensure a fair comparison with Trafficgen [19], we followed the testing settings outlined in that research and limited our model to generate vehicles only within a range of -50 meters to 50 meters. Moreover, we omitted scenarios with less than 8 ground-truth vehicles, yielding a total of 28,341 scenarios. We evaluated the first frame of each selected scenario to compute the average score. Under identical testing conditions, we reassessed Trafficgen and found the results to align closely with those reported in the original study. For the scene generation task, we found through experimentation that a slightly higher temperature setting leads to a more diverse distribution of scenes, thereby enhancing the performance of our metrics. Consequently, we selected a temperature setting of 1.25 and a top k of 200 for this task.

For our qualitative analysis of scene generation, we mainly used the nuPlan dataset. This dataset was chosen for its diverse and detailed scenario description tags, which are essential for our prompt-based conditioning experiments. These experiments aim to generate scenes that follow the scenario descriptions, offering a more nuanced and targeted approach.

## C.2 World Simulator

Our experiments are conducted on the WOD Sim Agents Benchmark, utilizing the WOD Motion Dataset. We adhere to the protocols of the Sim Agents challenge, unrolling 32 futures for each scenario in parallel. Specifically, for the experiments detailed in Tab.2, we utilize the test splits, with evaluation performed by the Waymo Sim Agent Challenge server. For other experiments, such as the scaling laws in Sec. 3 and various ablations, we employ the sub50 validation dataset, which comprises a total of 1024 scenarios.

Specifically, our model operates in 2Hz, and we interpolate the results to 10Hz for evaluation. For both the Waymo and nuPlan datasets, we use 1 second of history along with the current frame as conditions and predict the information for the next 8 seconds. Our model does not directly output information on the z-axis. To meet the requirements of the Waymo Sim Agents Benchmark, we infer the z value for each agent based on their predicted x and y positions, in conjunction with map data. Specifically, we employ the K-nearest neighbor algorithm to identify the k closest map points in the xy-2D plane location. We then average the z information of these map points to estimate the agent's z-axis position. In our approach, we set k to 4. Additionally, to better combat the jitter caused by the sampling and quantization processes, we employ a simple sliding window algorithm to smooth the final prediction results. This method helps with the stability and the overall smoothness.

### C.3 Interactive Planning

Our planning experiments are conducted on the nuPlan Dataset. To develop an interactive planner based on PDM [14], we integrate multiple simple policies with an interactive simulator and scorer, powered by GUMP. Specifically, our planner comprises 15 Intelligent Driver Model (IDM)-based policies with varied parameters, in addition to one imitation policy. We utilize the same set of IDM policies as PDM and adopt the ego vehicle's prediction within GUMP as the imitation policy. Then these policies are simulated in parallel, interacting with GUMP. This interaction is achieved by overriding the ego vehicle's next states with the output from the policy. In addition, the predictions of GUMP for other agents are used as input for the next frame of the policy. After interacting for 4 seconds, we generate a series of rollouts for future states.

By evaluating these simulated rollouts, we can derive the expected return. Considering GUMP operates stochastically, to make a more accurate estimation, we simulated and evaluated each policy four times and then averaged these results to calculate the expected return. Ultimately, we selected the output of the policy with the highest return as the next action to be taken.

In selecting our evaluation environment, we updated the original IDM-based environment by substituting IDM-controlled smart agents with those controlled by GUMP. For a visual comparison of these two testing environments, one can refer to Appendix F.3. Compared to IDM, GUMP-based smart agents exhibit more realistic and natural behavior, thus providing a more accurate representation of the planner's real-world performance. To ensure the accuracy of our assessment of the planner's performance, our experiments are conducted across both environments, with results presented side by side for reference. This method enables a comprehensive evaluation of our planning strategies in scenarios closely mimicking real-world driving conditions.

### C.4 Online Training

We implement our experiments using an existing open-source RL framework [80], which provides implementations of a number of standard RL algorithms. We use the Soft Actor-Critic (SAC) algorithm [23] to train a simple policy model, which consists of a ResNet-18 model followed by a 2-layer MLP-based projection network, outputting a squashed Gaussian distribution representing the state-conditional action distribution.

Using the same RL algorithm, network structure and hyperparameters, we train the policy network under the following two different environment setups respectively and compare the results:

- **Setup 1: Log Playback**. Log playback environment is used in this setting which has no reactive capability.
- **Setup 2: World Model**. A learned world model is used to implement the environment, in which participating agents react according to ego vehicle's action.

For each episode, the world model sees the current observation and unrolls one step conditioned on the policy model's output. we run 128 such environments in parallel. During the policy gradient update, we sample batches of 2048 state transitions from the replay buffer and optimize via Adam optimizer [40]. Each policy model is trained for 40k steps. The learning rate is set to 1e-4 and is reduced to 2e-5 after 70% of the training process. The training rewards can be found under the metrics section in Appendix. B.2. During training, 20% of the samples are used in an open-loop fashion, producing imitation losses. Empirically, we found that mixing open-loop samples during training helps speed up the training process. We then simulate each trained policy model under these environments and report metrics.

# D Training setup

## D.1 Raster Input

The raster layers encode various static elements and high definition map, which contrains:

- **Roadmap Raster** Represents the road network layout, including lanes, intersections, and other road features.
- **Baseline Paths Raster** Encodes baseline paths within the road network of the scene.
- **Route Raster** Represents the ego vehicle's desired route or path.
- **Drivable Area Raster** Represents areas considered drivable or navigable for vehicles within the scene.
- **Speed Limit Raster** Encodes speed limits at various locations within the scene.
- **Static Agents Raster** Represents positions of static agents (e.g., traffic cones) within the scene.
- **Traffic Light Raster** Encodes traffic light location and orientation at intersections within the scene.

## D.2 Data Augmentation

To enhance the generalization capability of our model, we implemented various data augmentation techniques. First, we applied an agents and frame dropout strategy, randomly dropping certain agent tokens or the tokens of an entire frame with a probability of 0.1. Second, we employed a random crop strategy, randomly selecting segments from the entire scenario sequence based on a fixed window size for training. Additionally, we uniformly sampled rotation angles from $[-\frac{\pi}{2}, \frac{\pi}{2}]$ to rotate the entire scene and translated the whole scene in both x and y directions within a range of $[-5, 5]$ meters. Finally, beyond using the self-driving car as the central focus of the scene, we also randomly selected several agents of interest from the dataset to serve as the center of the scene coordinate system.

### D.3 Loss

The loss function of GUMP is composed of three main components: reconstruction loss, cross-entropy loss, and multipath loss [3].

The reconstruction loss employs the L1 loss to supervise the image reconstruction part of the image autoencoder, which is defined as follows.

$$L_{\text{rec}} = \frac{1}{N_x N_y} \sum_{i,j} |I(i,j) - \hat{I}(i,j)| \tag{10}$$

where $N_x, N_y$ represent the number of 2D pixels, $I$ is the image pixel, and $\hat{I}$ is the reconstructed image.

The cross entropy loss (CE) is responsible for evaluating the accuracy of categorical predictions made by the model. It is applied to both key tokens, such as ID, class, and special tokens, as well as value tokens such as position $(x, y)$, orientation $(\theta)$, and dimensions $(v_x, v_y, w, l)$. This loss is formulated as:

$$L_{\text{CE}} = \frac{1}{N_k} \sum_{k \in \{ID, class, special\}} \mathcal{H}(\hat{k}, k) + \frac{1}{N_v} \sum_{v \in \{x, y, \theta, v_x, v_y, w, l\}} \mathcal{H}(\hat{v}, v) \tag{11}$$

where $\mathcal{H}$ denotes the cross entropy, $N_k$ and $N_v$ are the counts of key and value tokens, respectively, and $\hat{k}$ and $\hat{v}$ are the predicted logits.

To enhance the motion prediction capability of the model, we introduced trajectory prediction as an auxiliary task. The specific results and metric comparisons can be found in Appendix G. We employed the commonly used multipath loss [3], which combines a classification loss ($L_{cls}$) with a minimum Average Displacement Error ($L_{minADE}$). The classification loss supervises the likelihood of different future paths, while the minADE measures the accuracy of the predicted trajectory ($\hat{s}$) against the ground truth ($s$). The balance between these two components is regulated by the coefficients $\alpha$ and $\beta$:

$$L^i_{\text{traj}} = \alpha L^i_{cls} + \beta L_{minADE}(\hat{s}_i, s) \tag{12}$$

In MCT, trajectory predictions are made at intervals of every $n$ layers, resulting in multiple sets of trajectory prediction outcomes. As indicated above, $i$ represents the results from different layers of trajectory prediction heads.

Finally, the overall loss $L$ is a weighted sum of these components, allowing for customized emphasis on different aspects of the model's predictions:

$$L = \omega_{rec} * L_{\text{rec}} + \omega_{CE} * L_{CE} + \sum_i \omega^i_{traj} * L^i_{traj} \tag{13}$$

where $\omega_{rec}$, $\omega_{CE}$, and $\omega^i_{traj}$ are the weights applied to the reconstruction, cross-entropy, and trajectory prediction losses, respectively.

### D.4 Hyperparameters

**Model configuration** As shown in Table 7, we list the detailed hyperparameters of three different model variants.

| Hyperparameter | GUMP-small | GUMP-base | GUMP-medium |
|---|---|---|---|
| vocabulary size | 2972 | | |
| block size | 2048 | | |
| meshgrid $[x, y, \theta, w, l, v_x, v_y]$ | $[0.2m, 0.2m, \frac{\pi}{100}, 0.5m, 0.5m, 0.25m/s, 0.25m/s]$ | | |
| range $[x, y, \theta, w, l]$ | $[(-100, 100)m, (-100, 100)m, (-\pi, \pi), (0, 7)m, (0, 15)m]$ | | |
| range $[v_x, v_y]$ | $[(0, 25)m/s, (0, 25)m/s]$ | | |
| embedding dim | 384 | 768 | 1024 |
| MCT backbone | gpt2-small | gpt2-base | gpt2-medium |
| $N_{heads}$ | 6 | 12 | 16 |
| $N_{layer}$ | 12 | 12 | 24 |
| # GRU decoder layer | 2 | 2 | 2 |
| chunking size $T$(sec) | 2 | 2 | 2 |
| visual encoder | ResNet18 | ResNet34 | ResNet50 |
| total params | 55.8M | 184M | 523M |

**Table 7:** Model configurations of GUMP.

**Training configuration** We detail all training specifics in Table 8. Our models undergo an initial training phase of 10 epochs on the nuPlan Dataset, followed by a fine-tuning stage on the Waymo Open Dataset for an additional 10 epochs. The GUMP-medium model is trained using 8 A100 GPUs with a period of 3 days.

## E Ablation Study

### E.1 Effectiveness of NAR conversion

To accelerate the model's inference process, we employed Non-Autoregressive (NAR) conversion methods described in Section 2.2, transforming the full autoregressive (full-AR) model into a partially autoregressive (partial-AR) model. The specific experimental results, as outlined in Table 9, reveal that the transition to a partial-AR model results in a significant acceleration ($\times$132 speed up) without any noticeable degradation across various simulation realism metrics. This transformation demonstrates the efficacy of NAR conversion methods in improving computational efficiency while maintaining the quality of the simulation results.

### E.2 Effectiveness of Prediction Chunking and Temperal Aggregation.

As shown in Table 10, we present the ablation study results on the Waymo Sim Agents validation set. It is evident that incorporating prediction chunking as an auxiliary task alone leads to a 1% enhancement in the realism meta-metric, with particularly notable improvements observed in the interactive and map-based metrics. In addition, the minimum average displacement error (minADE) decreases significantly. Further enhancements are attained through temporal aggregation, where the ensemble of multiple predictions from chunked results yields

| Hyperparameter | Value |
|---|---|
| lr | 2e-4 |
| weight decay | 1e-3 |
| optimizer | adamw [47] |
| lr scheduler | multistep lr |
| batchsize | 64 |
| accumulate grad batches | 2 |
| training epoch | 10 |
| decay milestones | [6, 8] |
| decay rate | 0.1 |
| precision | fp16 |
| $\omega_{rec}$ | 1.0 |
| $\omega_{CE}$ | 1.0 |
| $\omega_{traj}^{0}$ | 0.25 |
| $\omega_{traj}^{1}$ | 0.25 |
| $\omega_{traj}^{2}$ | 0.5 |
| $\omega_{traj}^{3}$ | 0.5 |
| temperature | 1.1 |
| topk | 40 |
| temperature at SceneGen | 1.25 |
| topk at SceneGen | 400 |
| condition length | 3 |
| decay rate $\gamma$ | 1.2 |
| chunking time horizon $T$ | 2s |

**Table 8:** Training hyperparameters of GUMP.

an additional 1.2% improvement in the metametric. It is worth mentioning that the kinematic metrics are the primary contributors to this overall improvement. These experiments underscore the effectiveness of the Prediction Chunking and Temporal Aggregation modules in increasing the realism of simulations conducted by GUMP.

### E.3 Ablation Study of Decay Rate $\gamma$

In this section, we explore the optimal setting for the decay rate $\gamma$ in our temporal aggregation process, as depicted in Figure 7. Setting $\gamma = 0.0$ equates to omitting the temporal aggregation entirely. As we increase $\gamma$, future predictions within the chunking data are progressively weighted more heavily. Observing the components of the metametric, it becomes evident that a higher decay rate $\gamma$ distinctly improves the kinematic metrics while leading to a decrease in the interactive and map-based metrics. The overall meta-metric peaks at $\gamma = 1.2$. This suggests that by incorporating predictions that extend further into the future, agents are better able to adhere to their initial goals, thereby achieving higher kinematic metrics. However, this reliance on more distant predictions in chunked data tends to overlook interactions between agents, and between agents and the map at future time points, resulting in reduced interactive and map-compliance metrics.

| Inference Mode | Inference Speed (FPS) | Meta Metric↑ | minADE↓ | Kinematic Metrics↑ | Interactive Metrics↑ | Map-based Metrics↑ |
|---|---|---|---|---|---|---|
| full-AR | 0.69 | 0.645 | 1.548 | 0.4022 | 0.7651 | 0.8334 |
| partial-AR | **82**(×132) | 0.646 | 1.584 | 0.4036 | 0.7662 | 0.8331 |

**Table 9:** Comparison of full-AR and partial-AR inference mode. The inference speed is measured with a single A100 GPU. The transition from full-AR to partial-AR mode results in a significant acceleration (×132 speed-up) without any noticeable degradation across various simulation realism metrics.

| Prediction Chunking | Temporal Aggregation | Meta Metric↑ | minADE↓ | Kinematic Metrics↑ | Interactive Metrics↑ | Map-based Metrics↑ |
|---|---|---|---|---|---|---|
| | | 0.624 | 1.626 | 0.3903 | 0.7508 | 0.7903 |
| ✓ | | 0.634 | **1.562** | 0.3700 | **0.7674** | 0.8299 |
| ✓ | ✓ | **0.646** | 1.584 | **0.4036** | 0.7662 | **0.8331** |

**Table 10:** Ablation study of prediction chunking in Waymo Sim Agents benchmark. These experiments demonstrate the efficacy of Prediction Chunking and Temporal Aggregation modules in enhancing the realism of simulations conducted by GUMP.

### E.4 Ablation Study of Temperature

In this section, we conduct a detailed analysis of how varying the temperature parameter affects the performance and behavior of simulations, as illustrated in Table 8. The data clearly shows that the meta-metric peaks at a temperature of 1.1. Furthermore, as the temperature increases, kinematic metrics show a monotonically increasing trend, while both interactive and map-based metrics exhibit a monotonic decrease. This pattern indicates that higher temperatures facilitate better mode coverage by enabling more frequent sampling of low probabilistic states. However, such states may lead to an increase in collisions and deteriorate map compliance. Visual observations reveal that higher temperatures result in more aggressive behaviors among road participants, such as increased collisions or jaywalking. In contrast, lower temperatures are associated with more conservative behaviors, including more obedient drivers and more cautious pedestrians. This parameter thus offers a lever to more precisely control the behavior of traffic participants, enabling effective evaluation of driving policies across diverse scenarios.

### E.5 Ablation Study of the Number of Conditioned Frames

In this section, we explore the effects of varying the length of conditioned history frames within the autoregressive process. As depicted in Table 9, the performance is notably poor when there is a lack of historical input (condition length = 1, i.e only current frame). With the increment in input frame length, the overall meta-metric reaches its peak at 5 frames, corresponding to a 2-second history, after which the improvement plateaus. This suggests that for the purpose of simulation, a 2-second window of information is sufficiently informative. However,

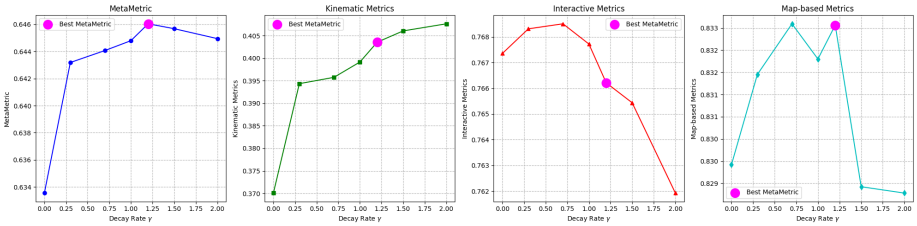



**Fig. 7:** Impact of Decay Rate $\gamma$ on the Realism Metrics in the Waymo Sim Agents Benchmark. This figure illustrates the variation curves of the overall MetaMetric, Kinematic metrics, Interactive metrics, and Map-based metrics as a function of decay rate $\gamma$, from left to right. The experimental results highlight that the overall meta-metric reaches its peak performance at $\gamma = 1.2$, which is highlighted with magenta.

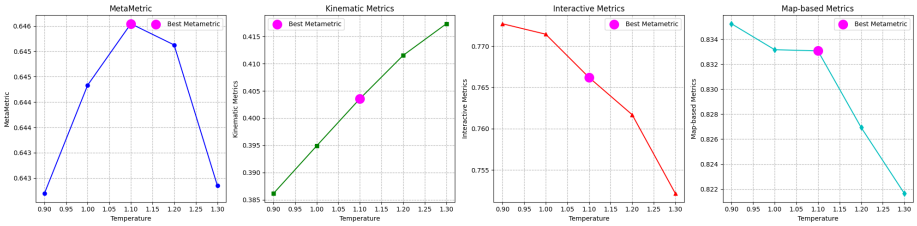



**Fig. 8:** The graph illustrates the relationship between temperature settings and Realism Metrics in the Waymo Sim Agents Benchmark. It highlights how varying temperature levels affect overall simulation performance, with a focus on the interplay between kinematic metrics, interactive behaviors, and map compliance. The chart specifically showcases the optimal meta-metric peak at a temperature of 1.1, while also detailing the divergent trends of increasing kinematic metrics and decreasing interactive and map-based metrics performance at higher temperatures.

considering that increasing the conditioned length significantly adds to computational demands, thereby affecting the efficiency of downstream tasks, we opt for a conditioned frames length of 3 (1-second history). This decision balances the need for sufficient historical context with the imperative of maintaining computational efficiency.

## F Qualitative Analysis

In this section, we will analyze the visual analysis of GUMP under different tasks. The visualization reflects GUMP's excellent capability in controlled scene generation, its realistic simulation of diverse complex driving scenarios, and provides a visual comparison with traditional IDM agents. This fully demonstrates the superiority of GUMP as a realistic controllable data-driven simulator.

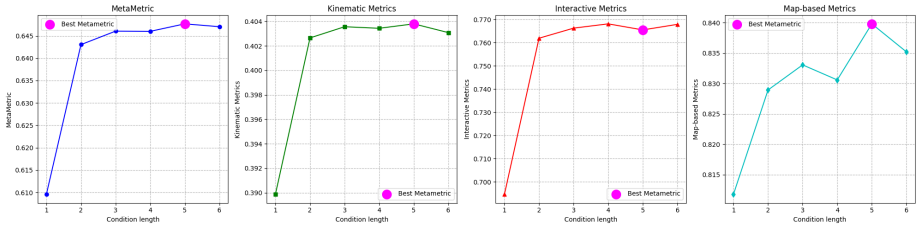



**Fig. 9:** Impact of the number of conditioned frames on the Realism Metrics in the Waymo Sim Agents Benchmark.

### F.1 Scene Generation

Here we demonstrate visualizations of scene generation based on map and scenario description prompts. By providing a static map, specifying the types and numbers of agents, and offering scenario description prompts, we can autoregressively generate the corresponding initial states and simulate future based on the initial states. As illustrated in Figure 10, under the same map input, the creation of distinctly different scenes is facilitated by varying the scenario descriptions and the numbers of objects. The 0th frame in the image is generated through fully autoregressive scene generation, while subsequent simulations are generated through partial-autoregressive scene extrapolation. Examples include low magnitude speed, waiting for pedestrian to cross, or controlling the position of objects, such as behind bike. Through such controlled methods, we are able to generate a large number of driving scenarios, thereby mitigating the issue of data scarcity.

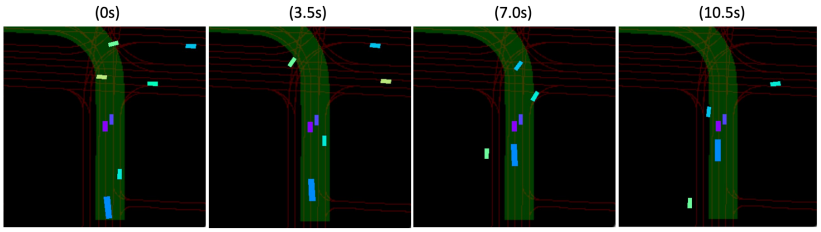
### F.2 Diverse Future

GUMP demonstrates exceptional efficacy in generating diverse, interactive, and rational driving behaviors within identical map settings. Fig. 11 show three pair of samples.

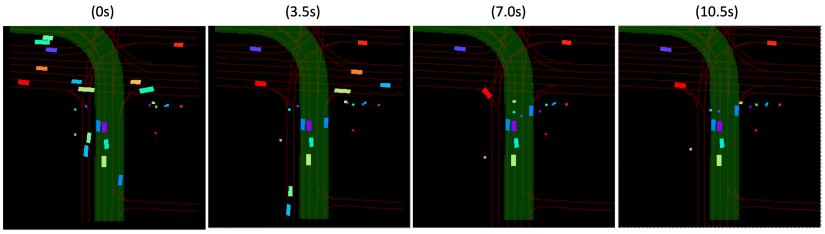
In Fig. 11 (a), the red car proceeds to make a right turn because there is sufficient distance from the approaching green vehicle, demonstrating the model's effective distance assessment for safe maneuvers. Conversely, the right panel depicts the red car yielding to a faster-moving green vehicle, illustrating the model's prioritization of safety in tighter traffic scenarios.

On the left side of Fig. 11 (b), all vehicles stick to their predetermined routes. However, the right side illustrates a more complex scenario where a light blue car accelerates and changes lanes, forcing the dark blue car following it to also change lanes to avoid the congestion initiated by the leading vehicle.

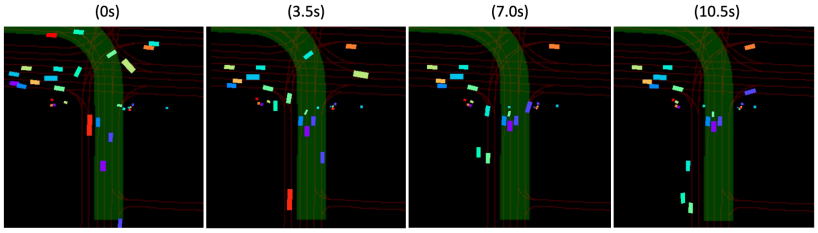
Lastly, Fig. 11 (c) illustrates the model's ability to choose highly probable actions, like making a left turn, as well as to perform less typical maneuvers, such as U-turns, given similar circumstances. This adaptability underscores the model's sophisticated decision-making capabilities across diverse traffic situations.




**(a)** scenario prompt: low magnitude speed




**(b)** scenario prompt: waiting for pedestrian to cross




**(c)** scenario prompt: behind bike

**Fig. 10:** We created and simulated three distinct scenarios using the same static map but with different scenario descriptions, including the scenario prompts and the classes and numbers of agents involved. It is worth noting that the 0th frame in the image is generated through fully autoregressive scene generation, while subsequent simulations are generated through partial-autoregressive scene extrapolation.

### F.3 Reactive Simulation

Fig. 12 illustrates the superior realism of GUMP compared to Rule-based reactive environments (IDM). The first row shows agents' movement generated by GUMP and the second row is for IDM. These visualizations reveal that the simplistic IDM environment does not provide an adequate reactive simulation for planning policies. Conversely, GUMP presents an efficient and more human-like alternative.

In Figure (a), we observe collisions between agents in the IDM environment, where a vehicle disregards traffic lights and collides with another vehicle making a left turn. In contrast, in GUMP's environment, vehicles exhibit more human-like behavior, avoiding any collisions.

In the second row of Figure (b), a bus fails to stop adequately behind another vehicle, leading to a rear-end collision. Meanwhile, in the top row showcasing GUMP's environment, the bus stops appropriately, avoiding any collision with the vehicle in front.

Finally, Figure (c) also highlights a scenario in the IDM environment where a vehicle ignores a traffic light and collides with another vehicle executing a left turn. However, in the environment simulated by GUMP, vehicles navigate the intersection smoothly, without any collision.

## G WOD Motion Results

To further validate its capabilities as a foundation model for multiple tasks, we also benchmarked GUMP's performance in motion prediction tasks. Utilizing the latent features corresponding to each agent, future 8 seconds trajectories and their associated probabilities are predicted through a simple multilayer perceptron (MLP) prediction head. Similar to the Waymo motion prediction benchmark, we adopted 6 distinct modes and supervised the model using a multipath loss.

### G.1 WODM Validation Set

As shown in Table 11, we compare the performance of GUMP-medium on the WOD motion validation set against other models. It's important to note that, to ensure a fair comparison, all models selected for this analysis are end-to-end models, meaning they do not incorporate additional post-processing or ensembling in their results. Traditionally, agent-centric models have led the way in motion prediction due to their more unified and easier-to-learn representation forms, outpacing scene-centric models. Despite this, our findings reveal that GUMP-medium significantly surpasses other scene-centric models, such as the SceneTransformer [52], and even approaches the performance of agent-centric models like MTR-e2e [67], without any bells and whistles. This demonstrates that GUMP can also serve as a pretraining framework, acting as a foundation model in motion planning to benefit a variety of related tasks.

| **Methods** | scene-centric | mAP ↑ | minADE ↓ | minFDE ↓ | MR ↓ |
|---|---|---|---|---|---|
| MTR-e2e [67] | ✗ | **0.32** | **0.52** | **1.10** | **0.12** |
| CPS [70] | ✗ | 0.32 | 0.74 | 1.49 | 0.20 |
| SceneTransformer [52] | ✓ | 0.28 | 0.61 | 1.22 | 0.16 |
| GUMP-m | ✓ | 0.30 | 0.60 | 1.15 | 0.13 |

**Table 11:** Performance comparison of motion prediction on the validation set of the WOMD. GUMP-medium significantly outperforms scene-centric SceneTransformer, and even approaches the performance of agent-centric models like MTR-e2e, without any bells and whistles.

### G.2 Per-type Results of WOMD Validation

We also present the per-type results of GUMP on the WOMD Validation set, as shown in reference Tab. 12.

| **Object type** | mAP ↑ | minADE ↓ | minFDE ↓ | MR ↓ |
|---|---|---|---|---|
| Vehicle | 0.32 | 0.75 | 1.42 | 0.14 |
| Pedestrian | 0.27 | 0.36 | 0.70 | 0.07 |
| Cyclist | 0.30 | 0.68 | 1.33 | 0.18 |
| **Avg** | 0.30 | 0.60 | 1.16 | 0.13 |

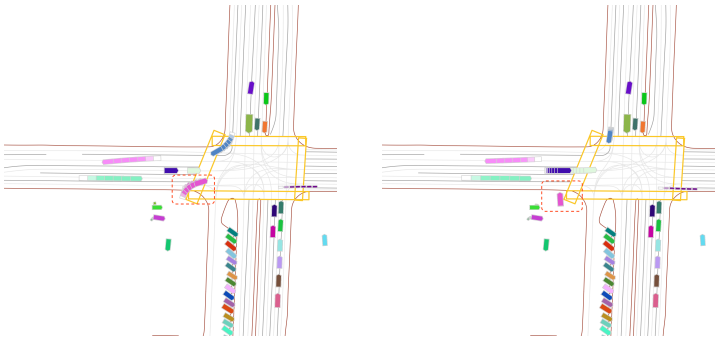
**Table 12:** Per-type performance of motion prediction on the validation set of WOMD.

## H Per-component WOD Sim Agent Metric

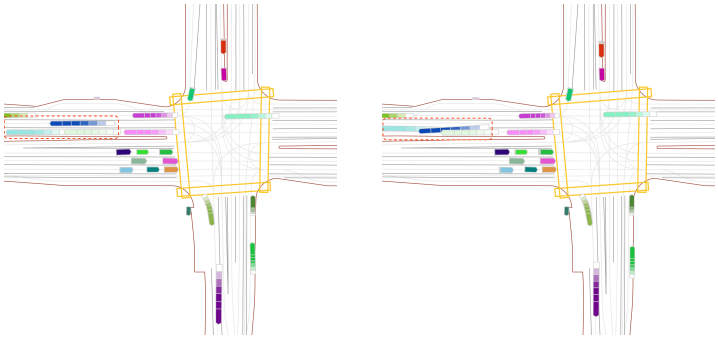
As shown in Table 13, to provide a more detailed showcase of our model's performance on the Waymo Sim Agents benchmark, we have listed the breakdown metrics for reference. Our method significantly leads the competition across a majority of these metric components, particularly in linear kinematic metrics, collision-related metrics, and map compliance metrics. It achieves state-of-the-art performance in both the overall realism metametric and the minADE (minimum Average Displacement Error) metric, highlighting its effectiveness and efficiency in producing realistic and accurate simulations for real-world driving scenarios.

| Agent Policy | Meta Metric (↑) | minADE (↓) | linear speed (↑) | linear accel. (↑) | ang. speed (↑) | ang. accel. (↑) | dist. to obj. (↑) | collision (↑) | TTC (↑) | dist. to road edge (↑) | offroad (↑) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Logged Oracle | 0.722 | 0.000 | 0.561 | 0.330 | 0.563 | 0.489 | 0.485 | 1.000 | 0.881 | 0.713 | 1.000 |
| SBTA-ADIA [48] | 0.420 | 3.611 | 0.317 | 0.174 | 0.478 | 0.463 | 0.265 | 0.337 | 0.770 | 0.557 | 0.483 |
| CAD [11] | 0.531 | 2.308 | 0.349 | 0.253 | 0.432 | 0.310 | 0.332 | 0.568 | 0.789 | 0.637 | 0.834 |
| Joint-Multipath++ [77] | 0.533 | 2.049 | 0.434 | 0.230 | 0.515 | 0.452 | 0.345 | 0.567 | 0.812 | 0.639 | 0.682 |
| Wayformer [51] | 0.575 | 2.498 | 0.331 | 0.098 | 0.413 | 0.406 | 0.297 | 0.870 | 0.782 | 0.592 | 0.866 |
| MTR+++ [59] | 0.608 | 1.679 | 0.414 | 0.107 | 0.484 | 0.436 | 0.347 | 0.861 | 0.797 | 0.654 | 0.895 |
| MVTA [78] | 0.636 | 1.866 | 0.439 | 0.220 | 0.533 | 0.480 | 0.374 | 0.875 | 0.829 | 0.654 | 0.893 |
| MVTE⋆ [78] | **0.645** | 1.674 | 0.445 | 0.222 | 0.535 | 0.481 | 0.383 | 0.893 | 0.832 | 0.664 | 0.908 |
| Trajeglish [56] | **0.644** | 1.615 | 0.448 | 0.192 | **0.538** | **0.485** | 0.386 | 0.918 | **0.837** | 0.658 | 0.887 |
| **GUMP-m** | **0.643** | **1.590** | **0.463** | **0.256** | 0.467 | 0.412 | **0.391** | **0.920** | 0.832 | **0.672** | **0.908** |

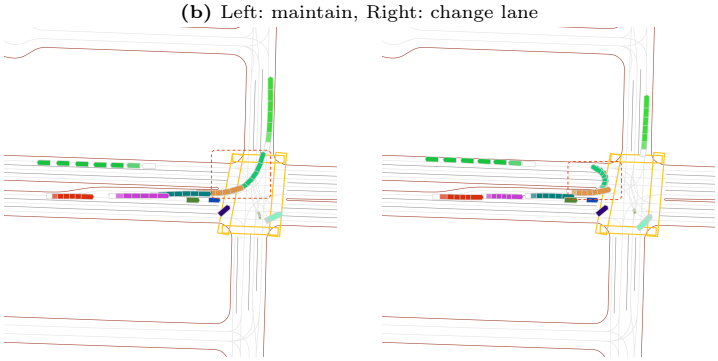
**Table 13:** Per-component metric results on the *test* split of WOMD, representing likelihoods. Note: ⋆ indicates the use of model ensemble techniques. Our method significantly leads the competition across a majority of metric components and achieves the state-of-the-art on the overall realism meta metric and the minADE metric. The best score is bolded.



**(a)** Left: proceed, Right: yield

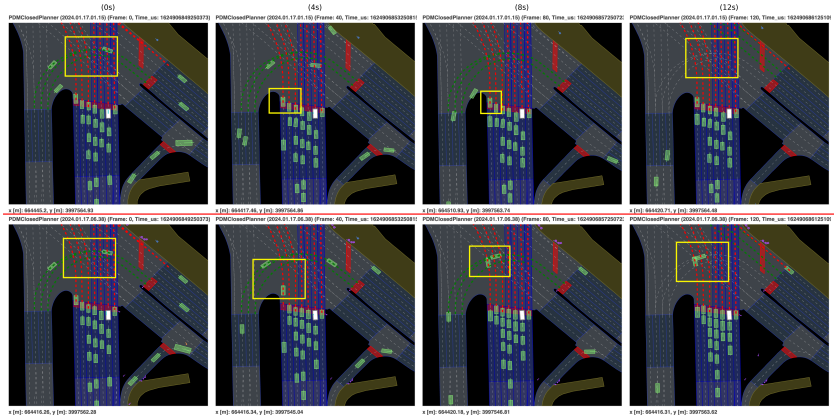


**(b)** Left: maintain, Right: change lane

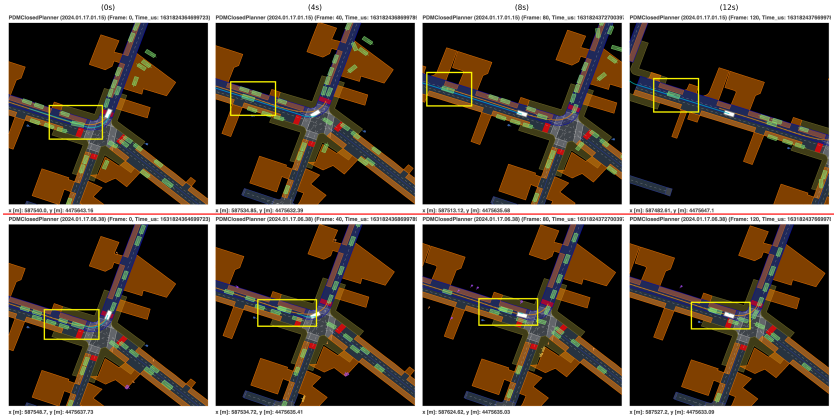


**(c)** Left: left-turn, Right: u-turn

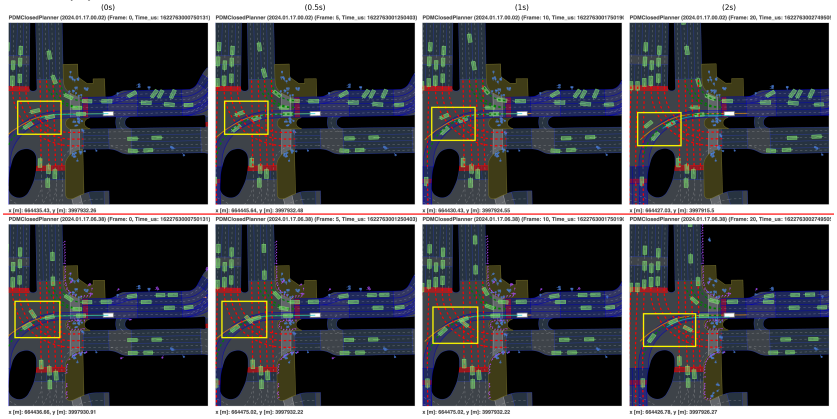
**Fig. 11:** The left and right images display the outcomes simulated under the same initial conditions. We overlap the 5 seconds future trajectories of all agents on a single image to represent the motion of each agent. The areas of interest in the image are highlighted with red dashed lines, which shows that simulated scenarios with identical initial conditions can lead to significantly different behaviors of the agents.



**(a)** In the IDM environment, one vehicle is running through a red light and collides with another vehicle which is making a left turn.



**(b)** In the IDM environment, the bus collides with the vehicle in front of it.



**(c)** In the IDM environment, one vehicle is running a red light and collides with another left-turning vehicle.

**Fig. 12:** Agents' behaviours comparison under GUMP's environment (first row) and IDM environment (second row). Interesting areas are highlighted with a yellow square.