

Language Conditioned Traffic Generation

Shuhan Tan¹ Boris Ivanovic² Xinshuo Weng²
Marco Pavone² Philipp Krähenbühl¹
¹UT Austin ²NVIDIA

Abstract: Simulation forms the backbone of modern self-driving development. Simulators help develop, test, and improve driving systems without putting humans, vehicles, or their environment at risk. However, simulators face a major challenge: They rely on realistic, scalable, yet interesting content. While recent advances in rendering and scene reconstruction make great strides in creating static scene assets, modeling their layout, dynamics, and behaviors remains challenging. In this work, we turn to language as a source of supervision for dynamic traffic scene generation. Our model, LCTGen, combines a large language model with a transformer-based decoder architecture that selects likely map locations from a dataset of maps, produces an initial traffic distribution, as well as the dynamics of each vehicle. LCTGen outperforms prior work in both unconditional and conditional traffic scene generation in-terms of realism and fidelity. Code and demo available at <https://ariostgx.github.io/lctgen>.

Keywords: Self-driving, Content generation, Large language model

1 Introduction

Driving simulators stand as a cornerstone in self-driving development. They aim to offer a controlled environment to mimic real-world conditions and produce critical scenarios at scale. Towards this end, they need to be highly *realistic* (to capture the complexity of real-world environments), *scalable* (to produce a diverse range of scenarios without excessive manual effort), and able to create *interesting* traffic scenarios (to test self-driving agents under different situations).

In this paper, we turn to natural language as a solution. Natural language allows practitioners to easily articulate interesting and complex traffic scenarios through high-level descriptions. Instead of meticulously crafting the details of each individual scenario, language allows for a seamless conversion of semantic ideas into simulation scenarios at scale. To harness the capacity of natural language, we propose LCTGen. LCTGen takes as input a natural language description of a traffic scenario, and outputs traffic actors’ initial states and motions on a compatible map. As we will show in Section 5, LCTGen generates realistic traffic scenarios that closely adhere to a diverse range of natural language descriptions, including detailed crash reports [1].

The major challenge of language-conditioned traffic generation is the absence of a shared representation between language and traffic scenarios. Furthermore, there are no paired language-traffic datasets to support learning such a representation. To address these challenges, LCTGen (see Figure 1) uses a scenario-only dataset and a Large Language Model (LLM). LCTGen has three modules: Interpreter, Generator and Encoder. Given any user-specified natural language query, the LLM-powered Interpreter converts the query into a compact, structured representation. Interpreter also retrieves an appropriate map that matches the described scenario from a real-world map library. Then, the Generator takes the structured representation and map to generate realistic traffic scenarios that accurately follow the user’s specifications. Also, we design the Generator as a query-based Transformer model [2], which efficiently generates the full traffic scenario in a single pass.

This paper presents three main contributions:

1. We introduce LCTGen, a first-of-its-kind model for language-conditional traffic generation.
2. We devise a method to harness LLMs to tackle the absence of language-scene paired data.

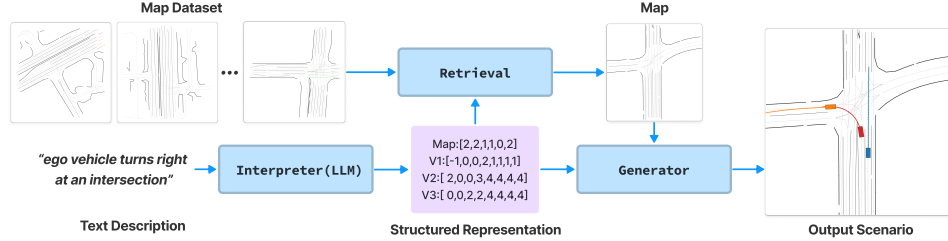


Figure 1: Overview of our LCTGen model.

3. LCTGen exhibits superior realism and controllability over prior work. We also show LCTGen can be applied to instructional traffic editing and controllable self-driving policy evaluation.

2 Related Work

Traffic scenario generation traditionally rely on rules defined by human experts [3], *e.g.*, rules that enforce vehicles to stay in lanes, follow the lead vehicles [4, 5, 6] or change lanes [7]. This approach is used in most virtual driving datasets [8, 9, 10, 11] and simulators [12, 3, 13]. However, traffic scenarios generated in this way often lack realism as they do not necessarily match the distribution of real-world traffic scenarios. Moreover, creating interesting traffic scenarios in this way requires non-trivial human efforts from experts, making it difficult to scale. In contrast, LCTGen learns the real-world traffic distribution for realistic traffic generation. Also, LCTGen can generate interesting scenarios with language descriptions, largely reducing the requirement of human experts.

Prior work in learning-based traffic generation is more related to our work. SceneGen [14] uses autoregressive models to generate traffic scenarios conditioned on ego-vehicle states and maps. TrafficGen [15] applies two separate modules for agent initialization and motion prediction. BITS [16] learns to simulate agent motions with a bi-level imitation learning method. Similar to LCTGen, these methods learn to generate realistic traffic scenarios from real-world data. However, they lack the ability to control traffic generation towards users’ preferences. In contrast, LCTGen achieves such controllability via natural languages and at the same time can generate highly realistic traffic scenarios. Moreover, we will show in the experiments that LCTGen also outperforms prior work in the setting of unconditional traffic reconstruction, due to our query-based end-to-end architecture.

Text-conditioned generative models have recently shown strong capabilities for controllable content creation for image [17], audio [18], motion [19], 3D object [20] and more. DALL-E [17] uses a transformer to model text and image tokens as a single stream of data. Noise2Music [18] uses conditioned diffusion models to generate music clips from text prompts. MotionCLIP [19] achieves text-to-human-motion generation by aligning the latent space of human motion with pre-trained CLIP [21] embedding. These methods typically require large-scale pairs of content-text data for training. Inspired by prior work, LCTGen is the first-of-its-kind model for text-conditioned traffic generation. Also, due to the use of LLM and our design of structured representation, LCTGen achieves text-conditioned generation *without* any text-traffic paired data.

Large language models have become increasingly popular in natural language processing and related fields due to their ability to generate high-quality text and perform language-related tasks. GPT-2 [22] is a transformer-based language model that is pre-trained on vast amounts of text data. Following this trend, GPT-3 [23] shows strong in-context-learning capacity, and InstructGPT [24] improves the instruction following capacity by fine-tuning with human feedback to better align the models with their users. More recently, GPT-4 [25] demonstrates strong performance in both in-context learning and instruction following. In our work, we adapt the GPT-4 model [25] with chain-of-thought [26] prompting method as our Interpreter.

3 Preliminaries

Let m be a map region, and $\mathbf{s}_t$ be the state of all vehicles in a scene at time t . A traffic scenario $\tau = (m, \mathbf{s}_{1:T})$ is the combination of a map region m and T timesteps of vehicle states $\mathbf{s}_{1:T} = [\mathbf{s}_1, \dots, \mathbf{s}_T]$.

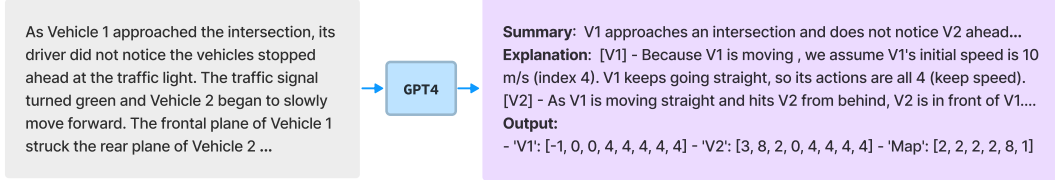


Figure 2: Example Interpreter input and output. We only show partial texts for brevity.

Map. We represent each map region m by a set of S lane segments denoted by $m = \{v_1, \dots, v_S\}$. Each lane segment includes the start point and endpoint of the lane, the lane type (center lane, edge lane, or boundary lane), and the state of the traffic light control.

Vehicle states. The vehicle states $\mathbf{s}_t = \{s_t^1, \dots, s_t^N\}$ at time t consist of N vehicle. For each vehicle, we model the vehicle's position, heading, velocity, and size. Following prior work [14, 15], we choose the vehicle at the center of the scenario in the first frame as the ego-vehicle. It represents the self-driving vehicle in simulation platforms.

4 LCTGen: Language-Conditioned Traffic Generation

Our goal is to train a language-conditioned model $\tau \sim \text{LCTGen}(L, \mathcal{M})$ that produces traffic scenarios from a text description L and a dataset of maps $\mathcal{M}$. Our model consists of three main components: A language Interpreter (Section 4.1) that encodes a text description into a structured representation z . Map Retrieval $m \sim \text{Retrieval}(z, \mathcal{M})$ that samples matching map regions m from a dataset of maps $\mathcal{M}$. A Generator (Section 4.3) that produces a scenario $\tau \sim \text{Generator}(z, m)$ from the map m and structured representation z . All components are stochastic, allowing us to sample multiple scenes from a single text description L and map dataset $\mathcal{M}$. We train the Generator with a real-world scenario-only driving dataset (Section 4.4).

4.1 Interpreter

The Interpreter takes a text description L as input and produces a structured representation $z = \text{Interpreter}(L)$. After defining the representation z , we show how to produce it via GPT-4 [25].

Structured representation $z = [z^m, z_1^a, \dots, z_N^a]$ contains both map-specific z^m and agent-specific components z_i^a . For each scenario, we use a 6-dimensional vector z^m describing the local map. It measures the number of lanes in each direction (north, south, east, west), the distance of the map center to an intersection, and the lane the ego-vehicle finds itself in. This compact abstract allows a language model to describe the important properties of a m and interact with map dataset $\mathcal{M}$. For each agent i , z_i^a is an 8-dimensional integer vector describing the agent relative to the ego vehicle. It contains an agent's quadrant position index (1-4), distance range (0-20m, 20-40m,...), orientation index (north, south, east, west), speed range (0-2.5m/s, 2.5-5m/s, ...), and action description (turn left, accelerate, ...). Please refer to Supp.A for a complete definition of z . Note that the representation z does not have a fixed length, as it depends on the number of agents in a scene.

Language interpretation. To obtain the structured representation, we use a large language model (LLM) and formulate the problem into a text-to-text transformation. Specifically, we ask GPT-4 [25] to translate the textual description of a traffic scene into a YAML-like description through in-context learning [27]. To enhance the quality of the output, we use Chain-of-Thought [26] prompting to let GPT-4 summarize the scenario q in short sentences and plan agent-by-agent how to generate z . See Figure 2 for an example input and output. Refer to Supp. A for the full prompt and Supp. D.4 for more complete examples.

4.2 Retrieval

The Retrieval module takes a map representation z^m and map dataset $\mathcal{M}$, and samples map regions $m \sim \text{Retrieval}(z^m, \mathcal{M})$. Specifically, we preprocess the map dataset $\mathcal{M}$ into potentially overlapping map regions $\{m_1, m_2, \dots\}$. We sample map regions, such that their center aligns with the locations of an automated vehicle in an offline driving trace. This ensures that the map region is both driveable and follows a natural distribution of vehicle locations. For each map m_j , we precompute its map representation $\hat{z}_j^m$. This is possible, as the map representation is designed to be

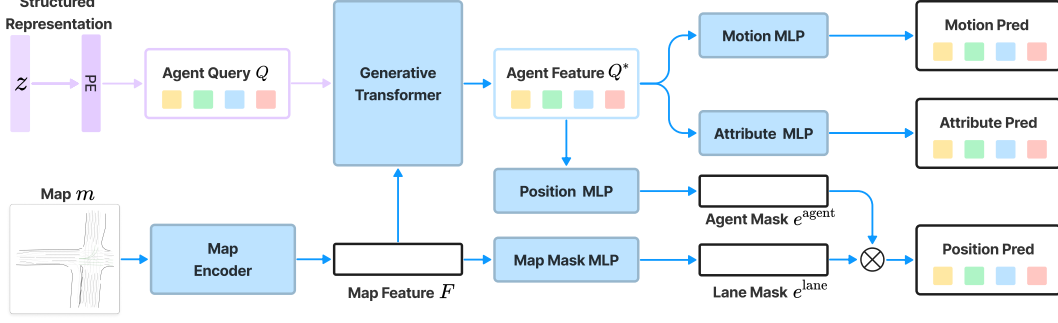


Figure 3: Architecture of our Generator model.

both easy to produce programmatically and by a language model. Given z^m , the Retrieval ranks each map region m_j based its feature distance $\|z^m - \hat{z}_j^m\|$. Finally, Retrieval randomly samples m from the top- K closest map regions.

4.3 Generator

Given a structured representation z and map m , the Generator produces a traffic scenario $\tau = \text{Generator}(z, m)$. We design Generator as a query-based transformer model to efficiently capture the interactions between different agents and between agents and the map. It places all the agents in a single forward pass and supports end-to-end training. The Generator has four modules (Figure 3): 1) a map encoder that extracts per-lane map features F ; 2) an agent query generator that converts structured representation z_i^a to agent query q_i ; 3) a generative transformer that models agent-agent and agent-map interactions; 4) a scene decoder to output the scenario τ .

Map encoder processes a map region $m = \{v_1, \dots, v_S\}$ with S lane segments v_i into a map feature $F = \{f_1, \dots, f_S\}$, and meanwhile fuse information across different lanes. Because S could be very large, we use *multi-context gating* (MCG) blocks [28] for efficient information fusion. MCG blocks approximate a transformer’s cross-attention, but only attend a single global context vector in each layer. Specifically, a MCG block takes a set of features $v_{1:S}$ as input, computes a context vector c , then combines features and context in the output $v'_{1:S}$. Formally, each block is implemented via

$$v'_i = \text{MLP}(v_i) \odot \text{MLP}(c) \quad \text{where} \quad c = \text{MaxPool}(v_{1:S})$$

where $\odot$ is the element-wise product. The encoder combines 5 MCG blocks with 2-layer MLPs.

Agent query generator transforms the structured representation z_i^a of each agent i into an agent query $q_i \in \mathbb{R}^d$. We implement this module as an MLP of positional embeddings of the structured representation $q_i = \text{MLP}(\text{PE}(z_i^a)) + \text{MLP}(x_i)$. We use a sinusoidal position encoding $\text{PE}(\cdot)$. We also add a learnable query vector x_i as inputs, as inspired by the object query in DETR [29].

Generative transformer. To model agent-agent and agent-map interactions, we use $F = \{f_1, \dots, f_S\}$ and $Q = \{q_1, \dots, q_N\}$ as inputs and pass them through multiple transformer layers. Each layer follows $Q' = \text{MHCA}(\text{MHSA}(Q), F)$, where MHCA, MHSA denote that multi-head cross-attention and multi-head self-attention respectively [2]. The output of Q' in each layer is used as the query for the next layer to cross-attend to F . The outputs of the last-layer Q^* are the agent features.

Scene decoder. For each agent feature q_i^* , the scene decoder produces the agents position, attributes, and motion using an MLP. To decode the position, we draw inspiration from MaskFormer [30], placing each actor on a lane segment in the map. This allows us to explicitly model the positional relationship between each actor and the road map. Specifically, we employ an MLP to turn q_i^* into an actor mask embedding $e_i^{\text{agent}} \in \mathbb{R}^d$. Likewise, we transform each lane feature f_j into a per-lane map mask embedding $e_j^{\text{lane}} \in \mathbb{R}^d$. The position prediction $\hat{p}_i \in \mathbb{R}^S$ for the i -th agent is then $\hat{p}_i = \text{softmax}(e_i^{\text{agent}} \times [e_1^{\text{lane}}, \dots, e_S^{\text{lane}}]^T)$,

For each agent query, we predict its **attributes**, namely heading, velocity, size, and position shift from the lane segment center, following Feng et al. [15]. The attribute distribution of a potential agent is modeled with a Gaussian mixture model (GMM). The parameters of a K -way GMM for

each attribute of agent i are predicted as $[\mu_i, \Sigma_i, \pi_i] = \text{MLP}(q_i^*)$, where μ_i, Σ_i and π_i denote the mean, diagonal covariance matrix, and the categorical weights of the K -way GMM model.

We further predict the future $T - 1$ step **motion** of each agent, by outputting K' potential future trajectories for each agent: $\{\text{pos}_{i,k}^{2:T}, \text{prob}_{i,k}\}_{k=1}^{K'} = \text{MLP}(q_i^*)$, where $\text{pos}_{i,k}^{2:T}$ represents the k -th trajectory states for $T - 1$ future steps, and $\text{prob}_{i,k}$ is its probability. Specifically, for each timestamp t , $\text{pos}_{i,k}^t = (x, y, \theta)$ contains the agent’s position (x, y) and heading θ at t .

During inference, we randomly sample values from the predicted position, attribute, and motion distributions of each agent query to generate an output agent status through T time stamps $s_{1:T}^i$. For categorical distributions, we select the category with the highest probability. For GMMs, we randomly sample a value from the model. Compiling the output for all agents, we derive the vehicle statuses $\mathbf{s}_{1:T}$. In conjunction with m , the Generator outputs the final traffic scenario $\tau = (m, \mathbf{s}_{1:T})$.

4.4 Training

The Generator is the only component of LCTGen that needs to be trained. We use real-world self-driving datasets, composed of D traffic scenarios $\{\tau_j\}_{j=1}^D$. For each traffic scene, we use an Encoder to produce the latent representation z , then train the Generator to reconstruct the scenario.

Encoder. The Encoder takes a traffic scenario τ and outputs structured agent representation: $z^a = \text{Encoder}(\tau)$. As mentioned in Section 4.1, z^a contains compact abstract vectors of each agent $\{z_1^a, \dots, z_N^a\}$. For each agent i , the Encoder extracts from its position, heading, speed, and trajectory from the ground truth scene measurements $s_{1:T}^i$ in τ , and converts it into z_i^a following a set of predefined rules. For example, it obtains the quadrant position index with the signs of (x, y) position. In this way, we can use Encoder to automatically convert any scenario τ to latent codes z . This allows us to obtain a paired dataset $(m, \mathbf{s}_{1:N}, z_{1:N}^a)$ from scenario-only driving dataset.

Training objective. For each data sample $(m, \mathbf{s}_{1:N}, z_{1:N}^a)$, we generate a prediction $p = \text{Generator}(z, m)$. The objective is to reconstruct the real scenario τ . We compute the loss as:

$$\mathcal{L}(p, \tau) = \mathcal{L}_{\text{position}}(p, \tau) + \mathcal{L}_{\text{attr}}(p, \tau) + \mathcal{L}_{\text{motion}}(p, \tau), \quad (1)$$

where $\mathcal{L}_{\text{position}}, \mathcal{L}_{\text{attr}}, \mathcal{L}_{\text{motion}}$ are losses for each of the different predictions. We pair each agent in p with a ground-truth agent in τ based on the sequential ordering of the structured agent representation z^a . We then calculate loss values for each component. For $\mathcal{L}_{\text{position}}$, we use cross-entropy loss between the categorical output $\hat{p}$ and the ground-truth lane segment id. For $\mathcal{L}_{\text{attr}}$, we use a negative log-likelihood loss, computed using the predicted GMM on the ground-truth attribute values. For $\mathcal{L}_{\text{motion}}$, we use MSE loss for the predicted trajectory closest to the ground-truth trajectory. The training objective is the expected loss $\mathcal{L}$ over the dataset. We refer readers to Supp. B for more detailed formulations of the loss functions.

5 Experiments

Datasets. We use the large-scale real-world Waymo Open Dataset [31], partitioning it into 68k traffic scenarios for training and 2.5k for testing. For each scene, we limit the maximum number of lanes to $S = 384$, and set the maximum number of vehicles to $N = 32$. We simulate $T = 50$ timesteps at 10 fps, making each τ represent a 5-second traffic scenario. We collect all the map segments in the training split to obtain the map dataset $\mathcal{M}$ with 68K map regions.

Implementation. We query GPT-4 [25] (with a temperature of 0.2) through the OpenAI API for Interpreter. For Generator, we set the latent dimension $d = 256$. We use a 5-layer MCG block for the map encoder. For the generative transformer, we use a 2-layer transformer with 4 heads. We use a dropout layer after each transformer layer with a dropout rate of 0.1. For each attribute prediction network, we use a 2-layer MLP with a latent dimension of 512. For attribute GMMs, we use $K = 5$ components. For motion prediction, we use $K' = 12$ prediction modes. We train Generator with AdamW [32] for 100 epochs, with a learning rate of $3e-4$ and batch size of 64.

Method	Pos	Initialization		Size	mADE	Motion	
		Heading	Speed			mFDE	SCR
TrafficGen [15]	0.2002	0.1524	0.2379	0.0951	10.448	20.646	5.690
MotionCLIP [19]	0.1236	0.1446	0.1958	0.1234	6.683	13.421	8.842
LCTGen (w/o z)	0.1319	0.1418	0.1948	0.1092	6.315	12.260	8.383
LCTGen	0.0616	0.1154	0.0719	0.1203	1.329	2.838	6.700

Table 1: Traffic scenario generation realism evaluation (lower the better).

5.1 Scene Reconstruction Evaluation

We evaluate the quality of LCTGen’s generated scenarios by comparing them to real scenarios from the driving dataset. For each scenario sample (τ, z, m) in the test dataset, we generate a scenario with $\hat{\tau} = \text{Generator}(z, m)$ and then compute different metrics with τ and $\hat{\tau}$.

Metrics. To measure the realism of scene initialization, we follow [14, 15] and compute the maximum mean discrepancy (MMD [33]) score for actors’ positions, headings, speed and sizes. Specifically, MMD measures the distance between two distributions q and p . For each pair of real and generated data $(\tau, \hat{\tau})$, we compute the distribution difference between them per attribute. To measure the realism of generated motion behavior, we employ the standard mean average distance error (mADE) and mean final distance error (mFDE). For each pair of real and generated scenarios $(\tau, \hat{\tau})$, we first use the Hungarian algorithm to compute a matching based on agents’ initial locations with their ground-truth location. We then transform the trajectory for each agent based on its initial position and heading to the origin of its coordinate frame, to obtain its relative trajectory. Finally, we compute mADE and mFDE using these relative trajectories. We also compute the scenario collision rate (SCR), which is the average proportion of vehicles involved in collisions per scene.

Baselines. We compare against a state-of-the-art traffic generation method, **TrafficGen** [15]. As TrafficGen only takes a map m as input to produce a scenario τ , we train a version of LCTGen that also only uses m as input for a fair comparison, referred to as LCTGen (w/o z). We also compare against **MotionCLIP** [19], which takes both a map m and text L as input to generate a scenario τ . Please refer to Supp. C for the implementation details of each baseline.

Results. The results in Table 1 indicate the superior performance of LCTGen. In terms of scene initialization, LCTGen (w/o z) outperforms TrafficGen in terms of MMD values for the Position, Heading, and Speed attributes. Importantly, when conditioned on the language input L , LCTGen significantly improves its prediction of Position, Heading, and Speed attributes, significantly outperforming both TrafficGen and MotionCLIP on MMD ($> 2\times$). LCTGen also achieves 7-8x smaller mADE and mFDE than baselines when comparing generated motions. The unconditional version of LCTGen, without z , also outpaces TrafficGen in most metrics, demonstrating the effectiveness of Generator’s query-based, end-to-end transformer design. We note that LCTGen (w/o z) has an on-par Size-MMD score with TrafficGen, which is lower than LCTGen. We conjecture that this is because our model learns spurious correlations of size and other conditions in z in the real data.

5.2 Language-conditioned Simulation Evaluation

LCTGen aims to generate a scenario τ that accurately represents the traffic description from the input text L . Since no existing real-world text-scenario datasets are available, we carry out our experiment using text L from a text-only traffic scenario dataset. To evaluate the degree of alignment between each scenario and the input text, we conduct a human study. We visualize the output scenario τ generated by LCTGen or the baselines, and ask humans to assess how well it matches the input text.

Datasets. We use a challenging real-world dataset, the **Crash Report** dataset [1], provided by the NHTSA. Each entry in this dataset comprises a comprehensive text description of a crash scenario, including the vehicle’s condition, driver status, road condition, vehicle motion, interactions, and more. Given the complexity and intricate nature of the traffic scenarios and their text descriptions, this dataset presents a significant challenge (see Figure 2 for an example). We selected 38 cases from this dataset for the purposes of our study. For a more controllable evaluation, we also use an **Attribute Description** dataset. This dataset comprises text descriptions that highlight various

Method	Crash Report		Attribute Description	
	Ours Preferred (%)	Score (1-5)	Ours Preferred (%)	Score (1-5)
TrafficGen [15]	92.35	1.58	90.48	2.43
MotionCLIP [19]	95.29	1.65	95.60	2.10
LCTGen	-	3.86	-	4.29

Table 2: Human study results on the language-conditioned simulation.

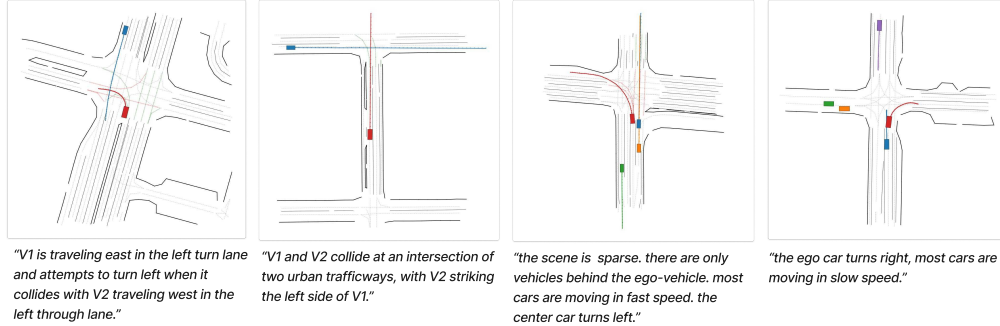


Figure 4: Qualitative results on text-conditioned generation.

attributes of a traffic scenario. These include aspects like sparsity ("the scenario is dense"), position ("there are vehicles on the left"), speed ("most cars are driving fast"), and the ego vehicle's motion ("the ego vehicle turns left"). We create more complex descriptions by combining 2, 3, and 4 attributes. This dataset includes 40 such cases. Refer to Supp. C for more dataset details.

Baselines. We compare with TrafficGen and MotionCLIP. For each text input L , LCTGen outputs a scenario $\tau = (m, s_{1:T})$. To ensure fairness, we feed data m to both TrafficGen and MotionCLIP to generate scenarios on the same map. As TrafficGen does not take language condition as input, we only feed L to MotionCLIP. In addition, TrafficGen can't automatically decide the number of agents, therefore it uses the same number of agents as our output τ .

Human study protocol. For each dataset, we conduct a human A/B test. We present the evaluators with a text input, along with a pair of scenarios generated by two different methods using the same text input, displayed in a random order. The evaluators are then asked to decide which scenario they think better matches the text input. Additionally, evaluators are requested to assign a score between 1 and 5 to each generated scenario, indicating its alignment with the text description; a higher score indicates a better match. A total of 12 evaluators participated in this study, collectively contributing 1872 scores for each model.

Quantitative Results. We show the results in Table 2. We provide preference score, reflecting the frequency with which LCTGen's output is chosen as a better match than each baseline. We also provide the average matching score, indicating the extent to which evaluators believe the generated scenario matches the text input. With LCTGen often chosen as the preferred model by human evaluators (*at least* 90% of the time), and consistently achieving higher scores compared to other methods, these results underline its superior performance in terms of text-controllability over previous works. The high matching score also signifies LCTGen's exceptional ability to generate scenarios that faithfully follow the input text. We include more analysis of human study result in Supp. D.3.

Qualitative Results. We show examples of LCTGen output given texts from the Crash Report (left two) and Attribute Description (right two) datasets in Figure 4. Each example is a pair of input text and the generated scenario. Because texts in Crash Report are excessively long, we only show the output summary of our Interpreter for each example (Full texts in Supp. C). Please refer to Supp. video for the animated version of the examples here. We show more examples in Supp. D.2.

5.3 Application: Instructional Traffic Scenario Editing

Besides language-conditioned scenario generation, LCTGen can also be applied to instructional traffic scenario editing. Given either a real or generated traffic scenario τ , along with an editing instruction text I , LCTGen can produce an edited scenario $\hat{\tau}$ that follows I . First, we acquire the structured

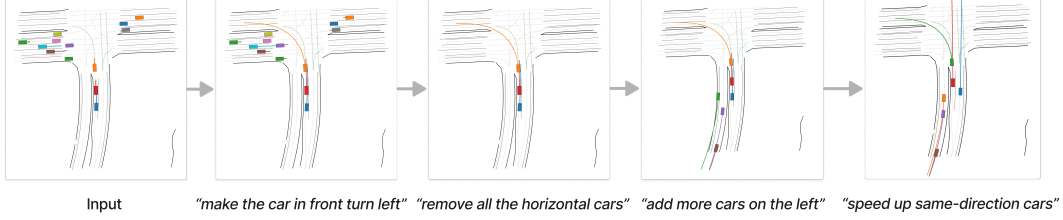


Figure 5: Instructional editing on a real-world scenario. Refer to Supp. A for full prompts.

Method	Pos	Heading	Speed	Size	Method	mADE	mFDE	SCR
w/o Quad.	0.092	0.122	0.076	0.124	w/o Speed	2.611	5.188	7.150
w/o Dist.	0.071	0.124	0.073	0.121	w/o Action	2.188	5.099	7.416
w/o Ori.	0.067	0.132	0.082	0.122	LCTGen init. + [15] motion	2.467	5.682	5.210
LCTGen	0.062	0.115	0.072	0.120	LCTGen	1.329	2.838	6.700

(a) Ablation study for scene initialization.

(b) Ablation study for motion behavior generation.

Table 3: Scene reconstruction ablation study on the Waymo Open Dataset.

representation of the scenario using $z = \text{Encoder}(\tau)$. Next, we compose a unique prompt that instructs Interpreter to alter z in accordance with I , resulting in $\hat{z} = \text{Interpreter}(z, I)$. Finally, we generate the edited scenario $\hat{\tau} = \text{Generator}(\hat{z}, m)$, where m is the same map used in the input.

We show an example of consecutive instructional editing of a real-world scenario in Figure 5. We can see that LCTGen supports high-level editing instructions (vehicle removal, addition and action change). It produces realistic output following the instruction. This experiment highlights LCTGen’s potential for efficient instruction-based traffic scenario editing. As another application of LCTGen, we also show how LCTGen can be utilized to generate interesting scenarios for controllable self-driving policy evaluation. Please refer to Supp. D.1 for this application.

5.4 Ablation study

Scene initialization. Table 3 summarizes the results, where the last row corresponds to our full method. To validate the performance of LCTGen for scene initialization, we mask out the quadrant index, distance, and orientation in the structure representation z for each agent, respectively. As a result, we observed a significant performance drop, especially in the prediction of Position and Heading attributes, as shown in the left side of Table 3. This suggests that including quadrant index, distance, and orientation in our structured representation is effective.

Motion behavior generation. We summarized the results in Table 3 (right). By masking out the speed range and action description in the structured representation for each agent, we observed a significant performance drop in the metrics for motion behavior. Moreover, if we initialize the scene with LCTGen while generating agents’ motion behavior using TrafficGen’s [15], we also observed significantly worse performance than using LCTGen to generate the traffic scenario in one shot. The results suggest that the end-to-end design of scene initialization and motion behavior generation by our LCTGen can lead to better performance. We show more ablation study results in Supp. D.5.

6 Conclusion

In this work, we present LCTGen, a first-of-its-kind method for language-conditioned traffic scene generation. By harnessing the expressive power of natural language, LCTGen can generate realistic and interesting traffic scenarios. The realism of our generated traffic scenes notably exceeds previous state-of-the-art methods. We further show that LCTGen can be applied to applications such as instructional traffic scenario editing and controllable driving policy evaluation.

Limitations. The primary constraint of LCTGen lies in the Interpreter module’s inability to output perfect agent placements and trajectories, as it lacks direct access to detailed lane information from the map. Our future work aims to overcome these issues by equipping the Interpreter with map and math APIs, enabling it to fetch precise map data and output more comprehensive traffic scenarios.

Acknowledgement. We thank Yuxiao Chen, Yulong Cao, and Danfei Xu for their insightful discussions. We thank all the human evaluation participants for their time and effort in our experiments. We also appreciate the constructive comments from the anonymous reviewers. This material is supported by the National Science Foundation under Grant No. IIS-1845485.

References

- [1] National Highway Traffic Safety Administration. Crash injury research engineering network. <https://crashviewer.nhtsa.dot.gov/CIREN/SearchIndex>, 2016.
- [2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [3] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL <https://elib.dlr.de/124092/>.
- [4] L. Papaleondiou and M. Dikaiakos. Trafficmodeler: A graphical tool for programming microscopic traffic simulators through high-level abstractions. In *VETECS*, pages 1 – 5, 05 2009. doi:10.1109/VETECS.2009.5073891.
- [5] J. Maroto, E. Delso, J. Félez, and J. Cabanellas. Real-time traffic simulation with a microscopic model. *Intelligent Transportation Systems, IEEE Transactions on*, 7:513 – 527, 01 2007. doi:10.1109/TITS.2006.883937.
- [6] F. E. Gunawan. Two-vehicle dynamics of the car-following models on realistic driving condition. *Journal of Transportation Systems Engineering and Information Technology*, 12(2): 67 – 75, 2012. ISSN 1570-6672. doi:[https://doi.org/10.1016/S1570-6672\(11\)60194-3](https://doi.org/10.1016/S1570-6672(11)60194-3). URL <http://www.sciencedirect.com/science/article/pii/S1570667211601943>.
- [7] J. Erdmann. Lane-changing model in sumo. In *Proceedings of the SUMO2014 Modeling Mobility with Open Data*, volume 24, 05 2014.
- [8] M. Wrenninge and J. Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *Arxiv*, Oct 2018. URL <http://arxiv.org/abs/1810.08705>.
- [9] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, 2016.
- [10] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8, 2017.
- [11] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016.
- [12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [13] A. Prakash, S. Bochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *ICRA*, pages 7249–7255, 05 2019. doi:10.1109/ICRA.2019.8794443.

- [14] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901, 2021.
- [15] L. Feng, Q. Li, Z. Peng, S. Tan, and B. Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *IEEE International Conference on Robotics and Automation*, London, United Kingdom, May 2023.
- [16] D. Xu, Y. Chen, B. Ivanovic, and M. Pavone. BITS: Bi-level imitation for traffic simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, May 2023. URL <https://arxiv.org/abs/2208.12403>.
- [17] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.
- [18] Q. Huang, D. S. Park, T. Wang, T. I. Denk, A. Ly, N. Chen, Z. Zhang, Z. Zhang, J. Yu, C. Frank, et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.
- [19] G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 358–374. Springer, 2022.
- [20] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems*, 2022.
- [21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [23] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Nee-lakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. 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. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.
- [24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [25] OpenAI. Gpt-4 technical report, 2023.
- [26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [27] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Re-thinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022.

- [28] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction, 2021.
- [29] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58451-1. doi:10.1007/978-3-030-58452-8_13. URL https://doi.org/10.1007/978-3-030-58452-8_13.
- [30] B. Cheng, A. G. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.
- [31] Waymo LLC. Waymo open dataset: An autonomous driving dataset, 2019.
- [32] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.
- [33] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- [34] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.
- [35] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, aug 2000. doi:10.1103/physreve.62.1805. URL <https://doi.org/10.1103%2Fphysreve.62.1805>.
- [36] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Appendix

In the appendix, we provide implementation and experiment details of our method as well as additional results. In Section A and Section B, we show details of Interpreter and Generator respectively. In Section C we present implementation details of our experiments. In Section D, we show more results on applications ablation study, as well as additional qualitative results. Finally, in Section E, we present an analysis on LCTGen with different LLMs.

A Interpreter

A.1 Structured representation details

The map specific z^m is a 6-dim integer vector. Its first four dimensions denote the number of lanes in each direction (set as north for the ego vehicle). The fifth dimension represents the discretized distance in 5-meter intervals from the map center to the nearest intersection (0-5, 5-10...). The sixth dimension indicates the ego vehicle's lane id, starting from 1 for the rightmost lane.

For agent i , the agent-specific z_i^a is an 8-dim integer vector describing this agent relative to the ego vehicle. The first dimension denotes the quadrant index (1-4), where quadrant 1 represents the front-right of the ego vehicle. The second dimension is the discretized distance to the ego vehicle with a 20m interval, and the third denotes orientation (north, south, east, west). The fourth dimension indicates discretized speed, set in 2.5m/s intervals. The last four dimensions describe actions over the next four seconds (one per second) chosen from a discretized set of seven possible actions: lane changes (left/right), turns (left/right), moving forward, accelerating, decelerating, and stopping.

A.2 Generation prompts

The scenario generation prompt used for Interpreter consists of several sections:

1. **Task description:** simple description of task of scenario generation and output formats.
2. **Chain-of-thought prompting [26]:** For example, "summarize the scenario in short sentences", "explain for each group of vehicles why they are put into the scenario".
3. **Description of structured representation:** detailed description for each dimension of the structured representation. We separately inform the model Map and Actor formats.
4. **Guidelines:** several generation instructions. For example, "Focus on realistic action generation of the motion to reconstruct the query scenario".
5. **Few-shot examples:** A few input-output examples. We provide a Crash Report example.

We show the full prompt below:

Prompt 1: Full prompt for Interpreter scenario generation.

```
1 You are a very faithful format converter that translate natrual language traffic scenario
  descriptions to a fix-form format to appropriately describe the scenario with motion
  action. You also need to output an appropriate map description that is able to support
  this scenario. Your ultimate goal is to generate realistic traffic scenarios that
  faithfully represents natural language descriptions normal scenes that follows the
  traffic rule.
2
3 Answer with a list of vectors describing the attributes of each of the vehicles in the
  scenario.
4
5 Desired format:
6 Summary: summarize the scenario in short sentences, including the number of vehicles. Also
  explain the underlying map description.
7 Explanation: explain for each group of vehicles why they are put into the scenario and how
  they fullfill the requirement in the description.
8 Actor Vector: A list of vectors describing the attributes of each of the vehicles in the
  scenario, only output the values without any text:
9 - 'V1': [,,,,,,,,]
```

```

10 - 'V2': [,,,,,,]
11 - 'V3': [,,,,,,]
12 Map Vector: A vector describing the map attributes, only output the values without any text:
13 - 'Map': [,,,,,]
14
15 Meaning of the Actor vector attribute:
16 - dim 0: 'pos': [-1,3] - whether the vehicle is in the four quadrant of ego vehicle in the
    order of [0 - 'front left', 1 - 'back left', 2- 'back right', 3 - 'front right']. -1 if
    the vehicle is the ego vehicle.
17 - dim 1: 'distance': [0,3] - the distance range index of the vehicle towards the ego vehicle
    ; range is from 0 to 72 meters with 20 meters interval. 0 if the vehicle is the ego
    vehicle. For example, if distance value is 15 meters, then the distance range index is
    0.
18 - dim 2: 'direction': [0,3] - the direction of the vehicle relative to the ego vehicle, in
    the order of [0- 'parallel_same', 1-'parallel_opposite', 2-'perpendicular_up', 3-'
    perpendicular_down']. 0 if the vehicle is the ego vehicle.
19 - dim 3: 'speed': [0,20] - the speed range index of the vehicle; range is from 0 to 20 m/s
    with 2.5 m/s interval. For example, 20m/s is in range 8, therefore the speed value is
    8.
20 - dim 4-7: 'action': [0,7] - 4-dim, generate actions into the future 4 second with each two
    actions have a time interval of 1s (4 actions in total), the action ids are [0 - 'stop
    ', 1 - 'turn left', 2 - 'left lane change', 3- 'decelerate', 4- 'keep_speed', 5- '
    accelerate', 6-'right lane change', 7-'turn right'].
21
22 Meaning of the Map attributes:
23 - dim 0-1: 'parallel_lane_cnt': 2-dim. The first dim is the number of parallel same-
    direction lanes of the ego lane, and the second dim is the number of parallel opposite-
    direction lanes of the ego lane.
24 - dim 2-3: 'perpendicular_lane_cnt': 2-dim. The first dim is the number of perpendicular
    upstream-direction lanes, and the second dim is the number of perpendicular downstream-
    direction lanes.
25 - dim 4: 'dist_to_intersection': 1-dim. the distance range index of the ego vehicle to the
    intersection center in the x direction, range is from 0 to 72 meters with 5 meters
    interval. -1 if there is no intersection in the scenario.
26 - dim 5: 'lane id': 1-dim. the lane id of the ego vehicle, counting from the rightmost lane
    of the same-direction lanes, starting from 1. For example, if the ego vehicle is in the
    rightmost lane, then the lane id is 1; if the ego vehicle is in the leftmost lane,
    then the lane id is the number of the same-direction lanes.
27
28 Transform the query sentence to the Actor Vector strictly following the rules below:
29 - Focus on realistic action generation of the motion to reconstruct the query scenario.
30 - Follow traffic rules to form a fundamental principle in most road traffic systems to
    ensure safety and smooth operation of traffic. You should incorporate this rule into
    the behavior of our virtual agents (vehicles).
31 - Traffic rule: in an intersection, when the vehicles on one side of the intersection are
    crossing, the vehicles on the other side of the intersection should be waiting. For
    example, if V1 is crossing the intersection and V2 is on the perpendicular lane, then
    V2 should be waiting.
32 - For speed and distance, convert the unit to m/s and meter, and then find the interval
    index in the given range.
33 - Make sure the position and direction of the generated vehicles are correct.
34 - Describe the initialization status of the scenario.
35 - During generation, the number of the vehicles is within the range of [1, 32].
36 - Always generate the ego vehicle first (V1).
37 - Always assume the ego car is in the center of the scene and is driving in the positive x
    direction.
38 - In the input descriptions, regard V1, Vehicle 1 or Unit #1 as the ego vehicle. All the
    other vehicles are the surrounding vehicles. For example, for "Vehicle 1 was traveling
    southbound", the ego car is Vehicle 1.
39 - If the vehicle is stopping, its speed should be 0m/s (index 0). Also, if the first action
    is 'stop', then the speed should be 0m/s (index 0).
40 - Focus on the interactions between the vehicles in the scenario.
41 - Regard the last time stamp as the time stamp of 5 second into the future.
42
43 Generate the Map Vector following the rules below:
44 - If there is vehicle turning left or right, there must be an intersection ahead.
45 - Should at least have one lane with the same-direction as the ego lane; i.e., the first dim
    of Map should be at least 1. For example, if this is a one way two lane road, then the
    first dim of Map should be 2.
46 - Regard the lane at the center of the scene as the ego lane.

```

```

47 - Consider the ego car's direction as the positive x direction. For example, for "V1 was
    traveling northbound in lane five of a five lane controlled access roadway", there
    should be 5 lanes in the same direction as the ego lane.
48 - The generated map should strictly follow the map descriptions in the query text. For
    example, for "Vehicle 1 was traveling southbound", the ego car should be in the
    southbound lane.
49 - If there is an intersection, there should be at least one lane in either the upstream or
    downstream direction.
50 - If there is no intersection, the distance to the intersection should be -1.
51 - There should be vehicle driving vertical to the ego vehicle in the scene only when there
    is an intersection in the scene. For example, when the road is just two-way, there
    should not be any vehicle driving vertical to the ego vehicle.
52 - If no intersection is mentioned, generate intersection scenario randomly with real-world
    statistics.
53
54
55 Query: The crash occurred during daylight hours on a dry, bituminous, two-lane roadway under
    clear skies. There was one northbound travel lane and one southbound travel lane with
    speed limit of 40 km/h (25 mph). The northbound lane had a -3.6 percent grade and
    the southbound lane had a +3.6 percent grade. Both travel lanes were divided by a
    double yellow line. A 2016 Mazda CX-3 (V1) was in a parking lot attempting to execute a
    left turn to travel south. A 2011 Dodge Charger (V2/police car) was traveling north
    responding to an emergency call with lights sirens activated. V1 was in a parking lot (
    facing west) and attempted to enter the roadway intending to turn left. As V1 entered
    the roadway it was impacted on the left side by the front of V2 (Event 1). V1 then
    rotated counterclockwise and traveled off the west road edge and impacted an embankment
    with its front left bumper (Event 2). After initial impact V2 continued on in a
    northern direction and traveling to final rest approximately 40 meters north of impact
    area facing north in the middle of the roadway. V1 and V2 were towed from the scene
    due to damage.
56
57 Summary: V1 attempts to turn left from a parking lot onto a two-lane roadway and is struck
    by V2, a police car traveling north with lights and sirens activated. There are 2
    vehicles in this scenario. This happens on a parking lot to a two-lane two-way road
    with intersection.
58 Explanation:
59 - V1 (ego vehicle) is attempting to turn left from a parking lot onto the roadway. We cannot
    find V1's speed in the query. Because V1 tries to turn left, its initial speed should
    be set low. We set V1's speed as 5 m/s, which has the index of 2. V1 turns left, so its
    actions are all 1 (turn left).
60 - V2 is a police car traveling north with lights and sirens activated. As V1 is turning left
    , 5 seconds before the crash, V1 is facing west and V2 is coming from northbound,
    crossing the path of V1. In the coordinates of V1 (which is facing west initially), V2
    comes from the front and is on the left side. Hence, V2's position is "front left" (3).
    As V1 is facing west and V2 facing north, V2 is moving in the perpendicular down
    direction with V1. Therefore its direction is 3 (perpendicular_down). We cannot find V2
    's speed in the query. Because V2 is a police car responding to an emergency call, we
    assume V2's init speed is 10 m/s (index 4). Given this speed, V2's distance to V1 is 10
    m/s * 5s = 50m (index 10). V2 keeps going straight, so its actions are all 4 (keep
    speed).
61 - Map: V1 tries to turn left from a parking lot onto a two-lane roadway. There are a one-
    way exit lane from parking lot (one same-direction parallel) and the ego vehicle is in
    the left turn lane with lane id 1. On the perpendicular side there is a two-lane
    roadway. V1 is about to turn left, so the distance to the intersection is set to be 10m
    (index 2).
62 Actor Vector:
63 - 'V1': [-1, 0, 0, 2, 1, 1, 1, 1]
64 - 'V2': [0, 10, 3, 4, 4, 4, 4, 4]
65 Map Vector:
66 - 'Map': [1, 0, 1, 1, 2, 1]
67
68 Query: INSERT_QUERY_HERE
69
70 Output:

```

A.3 Instructional editing prompts

We also provide Interpreter another prompt for instructional scenario editing. This prompt follow a similar structure to the generation prompt. We mainly adopt the task description, guidelines,

and examples to scenario editing tasks. Note that for the instructional editing task, we change the distance interval (second dimension) of agent-specific z_i^a from 20 meters to 5 meters. This is to ensure the unedited agents stay in the same region before and after editing.

We show the full prompt below:

Prompt 2: Full prompt for Interpreter instructional scenario editing.

```

1 You are a traffic scenario editor that edit fix-form traffic scenario descriptions according
  to the user's natural language instructions.
2
3 The user will input a fix-form traffic scenario description as well as the map description.
  The user also a natural language instruction to modify the scenario. You need to output
  a fix-form traffic scenario that is modified according to the instruction.
4
5 Input format:
6 - V1: [,,,,,,,,]
7 - V2: [,,,,,,,,]
8 - V3: [,,,,,,,,]
9 - Map: [,,,,,]
10 Instruction: natural language instruction to modify the scenario.
11
12 Output format:
13 Summary: summarize the scenario in short sentences. summarize the user instruction, and
  indicate which part of the scenario should be modified.
14 Explanation: explain step-by-step how each part of the scenario is modified.
15 Actor Vector: A list of vectors describing the attributes of each of the vehicles. Only the
  vehicles that are modified should be included in the output.
16 - V2: [,,,,,,,,]
17
18 Meaning of the Actor vector attribute:
19 - dim 0: 'pos': [-1,3] - whether the vehicle is in the four quadrant of ego vehicle in the
  order of [0 - 'front left', 1 - 'back left', 2- 'back right', 3 - 'front right']. -1 if
  the vehicle is the ego vehicle.
20 - dim 1: 'distance': [0,14] - the distance range index of the vehicle towards the ego
  vehicle; range is from 0 to 72 meters with 5 meters interval. 0 if the vehicle is the
  ego vehicle.
21 - dim 2: 'direction': [0,3] - the direction of the vehicle relative to the ego vehicle, in
  the order of [0- 'parallel_same', 1-'parallel_opposite', 2-'perpendicular_up', 3-'
  perpendicular_down']. 0 if the vehicle is the ego vehicle.
22 - dim 3: 'speed': [0,8] - the speed range index of the vehicle; range is from 0 to 20 m/s
  with 2.5 m/s interval. For example, 20m/s is in range 8, therefore the speed value is
  8.
23 - dim 4-7: 'action': [0,7] - 4-dim, generate actions into the future 4 second with each two
  actions have a time interval of 1s (4 actions in total), the action ids are [0 - 'stop
  ', 1 - 'turn left', 2 - 'left lane change', 3- 'decelerate', 4- 'keep_speed', 5- '
  accelerate', 6-'right lane change', 7-'turn right'].
24
25 Meaning of the Map attributes:
26 - dim 0-1: 'parallel_lane_cnt': 2-dim. The first dim is the number of parallel same-
  direction lanes of the ego lane, and the second dim is the number of parallel opposite-
  direction lanes of the ego lane.
27 - dim 2-3: 'perpendicular_lane_cnt': 2-dim. The first dim is the number of perpendicular
  upstream-direction lanes, and the second dim is the number of perpendicular downstream-
  direction lanes.
28 - dim 4: 'dist_to_intersection': 1-dim. the distance range index of the ego vehicle to the
  intersection center in the x direction, range is from 0 to 72 meters with 5 meters
  interval. -1 if there is no intersection in the scenario.
29 - dim 5: 'lane id': 1-dim. the lane id of the ego vehicle, counting from the rightmost lane
  of the same-direction lanes, starting from 1. For example, if the ego vehicle is in the
  rightmost lane, then the lane id is 1; if the ego vehicle is in the leftmost lane,
  then the lane id is the number of the same-direction lanes.
30
31 Follow the instructions below:
32 - 'V1' is the ego vehicle, and the other vehicles are the surrounding vehicles.
33 - The user will input a fix-form traffic scenario description as well as the map description
  . The user also an natural language instruction to modify the scenario. You need to
  output a fix-form traffic scenario that is modified according to the instruction.
34 - First figure out which part of the scenario should be modified according to the
  instruction. For example, if the instruction is "the vehicle in front of me should turn
  left", then the vehicle in front of the ego vehicle should be modified.
35

```

```

36 Input:
37 Actor vector:
38 - V1: [-1, 0, 0, 0, 4, 4, 4, 4]
39 - V2: [ 2, 1, 0, 1, 4, 4, 4, 4]
40 - V3: [ 3, 3, 0, 1, 4, 4, 4, 0]
41 - V4: [ 3, 4, 0, 8, 4, 4, 2, 0]
42 - V5: [ 0, 9, 1, 8, -1, 4, 5, -1]
43 - V6: [ 3, 5, 0, 0, 0, 0, 0, 0]
44 - V7: [ 0, 9, 3, 0, 0, 0, 0, 0]
45 - V8: [ 3, 10, 3, 3, 4, 5, 1, 0]
46 - V9: [ 0, 10, 3, 0, 0, 0, 0, -1]
47 - V10: [ 3, 10, 2, 0, 0, 0, 0, -1]
48 - V11: [ 3, 11, 2, 0, 0, 0, 0, 0]
49 - V12: [ 3, 11, 2, 0, 0, 7, 0, 0]
50 - Map: [4, 3, 2, 3, 6, 4]
51
52 Instruction: move the vehicle behind the ego vehicle to the opposite lane and move faster.
53
54 Output:
55 Summary: The instruction is to move the vehicle behind the ego vehicle to the opposite lane
    and accelerate. First find which vehicle is behind the ego vehicle. There are only 1
    vehicle behind the ego vehicle, that is V2 (with position=2, indicating on the right
    back side of the ego vehicle). Therefore, the vehicle V2 should be modified.
56 Explanation: The vehicle V2 is modified to move to the opposite lane and accelerate. The
    vehicle V2 is in the right back side of the ego vehicle, and the ego vehicle is in the
    rightmost lane of the same-direction lanes. Therefore, the vehicle V2 should move to
    the leftmost lane of the opposite-direction lanes. Therefore, V2's direction should be
    opposite to the ego vehicle, changed to 1 (parallel_opposite). In this lane, V2 should
    be moved to the left back of the ego car, its position should be changed to 1. V2
    should move faster, its speed should be changed to 10 (25 m/s).
57 Actor vector:
58 - V2: [ 1, 1, 1, 10, 4, 4, 4, 4]
59
60 Instruction: remove all the vehicles on the front of the ego car and moving in the same
    direction.
61
62 Output:
63 Summary: The instruction is to remove all the vehicles on the front of the ego car and
    moving in the same direction. First find which vehicles are on the front of the ego
    vehicle. V3-V12 are all on the front of the ego vehicle. Then, only V3, V4 and V6 has
    the same direction as the ego vehicle (0). Therefore, V3, V4 and V6 should be removed.
64 Explanation: V3, V4, V6 are on the front of the ego vehicle and moving in the same
    direction. V3, V4 and V6 are removed from the scenario.
65
66 Actor vector:
67 - V3: removed.
68 - V4: removed.
69 - V6: removed.
70
71 Input: INSERT_QUERY_HERE
72
73 Output:

```

B Generator

B.1 Training objectives

In the main paper, we show the full training objective of Generator as:

$$\mathcal{L}(p, \tau) = \mathcal{L}_{\text{position}}(p, \tau) + \mathcal{L}_{\text{attr}}(p, \tau) + \mathcal{L}_{\text{motion}}(p, \tau). \quad (2)$$

In this section, we provide details of each loss function. We first pair each agent $\hat{a}_i$ in p with a ground-truth agent a_i in τ based on the sequential ordering of the structured agent representation z^a . Assume there are in total N agents in the scenario.

For $\mathcal{L}_{\text{position}}$, we use cross-entropy loss between the per-lane categorical output $\hat{p}$ and the ground-truth lane segment id l . Specifically, we compute it as

$$\mathcal{L}_{\text{position}}(p, \tau) = \sum_{i=1}^N -\log \hat{p}_i(l_i), \quad (3)$$

where l_i is the index of the lane segment that the i -th ground-truth agent a_i is on.

For $\mathcal{L}_{\text{attr}}$, we use a negative log-likelihood loss, computed using the predicted GMM on the ground-truth attribute values. Recall that for each attribute of agent i , we use an MLP to predict the parameters of a GMM model $[\mu_i, \Sigma_i, \pi_i]$. Here, we use these parameters to construct a GMM model and compute the likelihood of ground-truth attribute values. Specifically, we have

$$\begin{aligned} \mathcal{L}_{\text{attr}}(p, \tau) = \sum_{i=1}^N & (-\log \text{GMM}_{\text{heading},i}(h_i) - \log \text{GMM}_{\text{vel},i}(vel_i) \\ & - \log \text{GMM}_{\text{size},i}(bbox_i) - \log \text{GMM}_{\text{pos},i}(pos_i)), \end{aligned} \quad (4)$$

where $\text{GMM}_{\text{heading},i}$, $\text{GMM}_{\text{vel},i}$, $\text{GMM}_{\text{size},i}$, $\text{GMM}_{\text{pos},i}$ represent the likelihood function of the predicted GMM models of agent i 's heading, velocity, size and position shift. These likelihood values are computed using the predicted GMM parameters. Meanwhile, h_i , vel_i , $bbox_i$ and pos_i represent the heading, velocity, size and position shift of the ground-truth agent a_i respectively.

For $\mathcal{L}_{\text{motion}}$, we use MSE loss for the predicted trajectory closest to the ground-truth trajectory following the multi-path motion prediction idea [28]. Recall that for each agent $\hat{a}_i$, we predict K' different future trajectories and their probabilities as $\{\text{pos}_{i,k}^{2:T}, \text{prob}_{i,k}\}_{k=1}^{K'} = \text{MLP}(q_i^*)$. For each timestamp t , $\text{pos}_{i,k}^t$ contains the agent's position and heading. We assume the trajectory of ground-truth agent a_i is $\text{pos}_{i,*}^{2:T}$. We can compute the index k^* of the closest trajectory from the K' predictions as $k^* = \arg \min_k \sum_{t=2}^T (\text{pos}_{i,k}^t - \text{pos}_{i,*}^t)^2$. Then, we compute the motion loss for agent i as:

$$\mathcal{L}_{\text{motion},i} = -\log \text{prob}_{i,k^*} + \sum_{t=2}^T (\text{pos}_{i,k^*}^t - \text{pos}_{i,*}^t)^2, \quad (5)$$

where we encourage the model to have a higher probability for the closest trajectory k^* and reduce the distance between this trajectory with the ground truth. The full motion loss is simply:

$$\mathcal{L}_{\text{motion}}(p, \tau) = \sum_i^N \mathcal{L}_{\text{motion},i} \quad (6)$$

where we sum over all the motion losses for each predicted agent in p .

C Experiment Details

C.1 Baseline implementation

TrafficGen [15]. We use the official implementation¹. For a fair comparison, we train its Initialization and Trajectory Generation modules on our dataset for 100 epochs with batch size 64. We modify $T = 50$ in the Trajectory Generation to align with our setting. We use the default values for all the other hyper-parameters. During inference, we enforce TrafficGen to generate N vehicles by using the result of the first N autoregressive steps of the Initialization module.

MotionCLIP [19]. The core idea of MotionCLIP is to learn a shared space for the interested modality embedding (traffic scenario in our case) and text embedding. Formally, this model contains a scenario encoder E , a text encoder $\hat{E}$, and a scenario decoder D . For each example of scene-text paired data (τ, L, m) , we encode scenario and text separately with their encoders $\mathbf{z} = E(\tau)$,

¹<https://github.com/metadriverse/trafficgen>

$\hat{\mathbf{z}} = \hat{E}(L)$. Then, the decoder takes $\mathbf{z}$ and m and output a scenario $p = D(\mathbf{z}, m)$. MotionCLIP trains the network with $\mathcal{L}_{\text{rec}}$ to reconstruct the scenario from the latent code:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{position}}(p, \tau) + \mathcal{L}_{\text{attr}}(p, \tau) + \mathcal{L}_{\text{motion}}(p, \tau), \quad (7)$$

where we use the same set of loss functions as ours (Equation 2). On the other hand, MotionCLIP aligns the embedding space of the scenario and text with:

$$\mathcal{L}_{\text{align}} = 1 - \cos(\mathbf{z}, \hat{\mathbf{z}}), \quad (8)$$

which encourages the alignment of scenario embedding $\mathbf{z}$ and text embedding $\hat{\mathbf{z}}$. The final loss function is therefore

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{L}_{\text{align}}, \quad (9)$$

where we set $\lambda = 100$.

During inference, given an input text L and a map m , we can directly use the text encoder to obtain latent code and decode a scenario from it, formally $\tau = D(\hat{E}(L), m)$.

For the scenario encoder E , we use the same scenario encoder as in [15], which is a 5-layer *multi-context gating* (MCG) block [28] to encode the scene input τ and outputs $\mathbf{z} \in \mathbb{R}^{1024}$ with the context vector output c of the final MCG block. For text encoder $\hat{E}$, we use the sentence embedding of the fixed GPT-2 model. For the scenario decoder D , we modify our Generator to take in latent representation $\mathbf{z}$ with a dimension of 1024 instead of our own structured representation. Because D does not receive the number of agents as input, we modify Generator to produce the $N = 32$ agents for every input and additionally add an MLP decoder to predict the objectiveness score of each output agent. Here objectiveness score is a binary probability score indicating whether we should put each predicted agent onto the final scenario or not. During training, for computation of $\mathcal{L}_{\text{rec}}$, we use Hungarian algorithm to pair ground-truth agents with the predicted ones. We then supervise the objectiveness score in a similar way as in DETR.

Note that we need text-scenario paired data to train MotionCLIP. To this end, we use a rule-based method to convert a real dataset τ to a text L . This is done by describing different attributes of the scenario with language. Similar to our Attribute Description dataset, in each text, we enumerate the scenario properties 1) sparsity; 2) position; 3) speed and 4) ego vehicle's motion. Here is one example: "the scene is very dense; there exist cars on the front left of ego car; there is no car on the back left of ego car; there is no car on the back right of ego car; there exist cars on the front right of ego car; most cars are moving in fast speed; the ego car stops".

We transform every scenario in our dataset into a text with the format as above. We then train MotionCLIP on our dataset with the same batch size and number of iterations as LCTGen.

C.2 Metric

We show how to compute MMD in this section. Specifically, MMD measures the distance between two distributions q and p .

$$\begin{aligned} \text{MMD}^2(p, q) = & \mathbb{E}_{x, x' \sim p}[k(x, x')] + \mathbb{E}_{y, y' \sim q}[k(y, y')] \\ & - 2\mathbb{E}_{x \sim p, y \sim q}[k(x, y)], \end{aligned} \quad (10)$$

where k is the kernel function (a Gaussian kernel in this work). We use Gaussian kernel in this work. For each pair of real and generated data $(\tau, \hat{\tau})$, we compute the distribution difference between them per attribute.

C.3 Dataset

Crash Report. We use 38 cases from the CIREN dataset [1] from the NHTSA crash report search engine. Each case contains a long text description of the scenario as well as a PDF diagram showing

the scenario. Because the texts are very long and require a long time for humans to comprehend, in our human study, along with each text input, we will also show the diagram of the scenario as a reference. We show example crash reports in Section D.4. We also refer the reader to the NHTSA website ² to view some examples of the crash report.

Attribute Description. We create text descriptions that highlight various attributes of a traffic scenario. Specifically, we use the following attributes and values:

1. Sparsity: "the scenario is {nearly empty/sparse/with medium density/very dense}".
2. Position: "there are only vehicles on the {left/right/front/back} side(s) of the center car" or "there are vehicles on different sides of the center car".
3. Speed: "most cars are moving in {slow/medium/fast} speed" or "most cars are stopping".
4. Ego-vehicle motion: "the center car {stops/moves straight/turns left/turns right}".

case_0

This two-vehicle collision occurred at a crossover area in a two-way divided trafficway, on a weekday afternoon (daylight), during a rain shower, on a wet asphalt surface. The east/west roadway was straight with an uphill grade for the three westbound lanes, and a level grade for the three eastbound lanes. This trafficway intersected with a two-way, two-lane north/south roadway whose southbound lane was controlled by a stop sign. The posted speed limit for the east/west trafficway was 105 kmph (65 mph). Vehicle 2, 2001 Volkswagen Jetta four-door sedan, was being driven westbound (east to west) in the right lane approaching the crossover area. Vehicle 1 began to accelerate while attempting to cross over the westbound lanes. Vehicle 1 traveled across the left westbound lane and was struck in its right side by the front of Vehicle 2 (Event 1) as it entered the right westbound lane. After impact, Vehicle 1 was redirected northwest while rotating clockwise as it approached the roadside. Vehicle 1 came to rest on the right shoulder of the westbound lanes, facing southeast. After impact, Vehicle 2 spun-out while rotating clockwise and came to rest in the left northbound lane facing northeast. Vehicle 1 and 2 were towed due to damage sustained in the crash.

Crash report diagram (as reference)

Case Number: 3-C4-2020-153

(case_0_pair_2)

(case_0_pair_2) Which generated scenario above match the text description better? *

	1	2	3	
Left	☐	☒	☐	Right

(case_0_pair_2) LEFT scenario matches the text description. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☒	☐	☐	Strongly Agree

(case_0_pair_2) RIGHT scenario matches the text description. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☒	☐	☐	Strongly Agree

Figure A1: Human study user interface.

We create sentences describing each of the single attributes with all the possible values. We also compose more complex sentences by combining 2,3 or 4 attributes together with random values for each of them. In total, we created 40 cases for human evaluation. Please refer to Section D.4 for some example input texts from this dataset.

²<https://crashviewer.nhtsa.dot.gov/CIREN/Details?Study=CIREN&CaseId=11>

C.4 Human study

We conduct the human study to access how well the generated scenario matches the input text. We showcase the user interface of our human study in Figure A1. We compose the output of two models with the same text input in random order and ask the human evaluator to judge which one matches the text description better. Then, we also ask them to give each output a 1-5 score. We allow the user to select "unsure" for the first question.

We invite 12 human evaluators for this study, and each of them evaluated all the 78 cases we provided. We ensure the human evaluators do not have prior knowledge of how different model works on these two datasets. On average, the human study takes about 80 minutes for each evaluator.

C.5 Qualitative result full texts

In Figure 4 and Figure A2, we show examples of the output of our model on Crash Report data. Recall that the texts we show in the figures are the summary from our Interpreter due to space limitations. We show the full input text for each example in this section.

Text 1: Full texts of examples in Figure 4 .

1	Figure 4 Column 1 (CIREN ID 594):
2	"This crash occurred during daylight hours on a dry, bituminous divided trafficway (median strip without positive barrier) under clear skies. There were four east travel lanes (two through lanes, one left turn and one right turn) and four west travel lanes (two through lanes, one left and one right). The east lanes have a slight right curve and the west lanes curve slightly to the left. Both east/west travel lanes were level grade at point of impact and divided by a grass median. The speed limit at this location is 80km/h (50 mph). The intersecting north/south roadway consisted of one north travel lane and three south travel lanes (one through lanes, one left and one right). These travel lanes were divided by a raised concrete median on the northern side of the intersection. This intersection is controlled by overhead traffic signals. A 2017 Dodge Grand Caravan (V1) was traveling east in the left turn lane and a 2006 Nissan Sentra (V2) was traveling west in the left through lane. As V1 was traveling east it attempted to execute a left turn to travel north when its front bumper impacted the front bumper of V2 (Event 1). After initial impact, V1 rotated counterclockwise approximately 80 degrees before traveling to its final resting position in the middle of the intersection facing north. V2 was traveling west in the left through lane and attempting to travel through the intersection when its front bumper impacted the front bumper of V1. After initial impact V2 rotated clockwise approximately 20 degrees before traveling to its final resting position in the middle of the intersection facing northwest. V1 and V2 were towed from the scene due to damage sustained in the crash."
3	
4	Figure 4 Column 2 (CIREN ID 31):
5	"A 2016 Kia Sedona minivan (V1) was traveling southwest in the right lane of three. A 2015 Chevrolet Silverado cab chassis pickup (V2) was ahead of V1 in the right lane. V2 was a working vehicle picking up debris on the highway in a construction zone. The driver of V2 stopped his vehicle in the travel lane. The driver of V1 recognized an impending collision and applied the brakes while steering left in the last moment before impact. V1 slid approximately three meters before the front of V1 struck the back plane of V2 in a rear-end collision with full engagement across the striking planes (Event 1). Both vehicles came to rest approximately two meters from impact. V1 was towed due to damage while V2 continued in service."
6	
7	Figure A2 Row 2 (CIREN ID 33):
8	"This two-vehicle collision occurred during the pre-dawn hours (dark, street lights present) of a fall weekday at the intersection of two urban roadways. The crash only involved the eastern leg of the intersection. The westbound lanes of the eastern leg consisted of four westbound lanes that included a right turn lane, two through lanes, and a left turn lane. The three eastbound lanes of the eastern leg consisted of a merge lane from the intersecting road and two through-lanes. The roadway was straight with a speed limit of 89 kmph (55 mph), and the intersection was controlled by overhead, standard electric, tri-colored traffic signals. At the time of the crash, the weather was clear and the roadway surfaces were dry. As Vehicle 1 approached the intersection, its driver did not notice the vehicles stopped ahead at the traffic light. The traffic signal turned green and Vehicle 2 began to slowly move forward. The frontal plane of Vehicle 1 struck the rear plane of Vehicle 2 (Event 1). Both vehicles came to rest in the left through-lane of the westbound lane facing in a westerly direction. Vehicle 1 was towed from the scene due to damage sustained in the crash. Vehicle 2 was not towed nor

disabled. The driver of Vehicle 2 was transported by land to a local trauma center and was treated and released."

9

10 Figure A2 Row 4 (CIREN ID 77):

11 "A 2017 Chevrolet Malibu LS sedan (V1) was traveling southeast in the right lane cresting a hill. A 1992 Chevrolet C1500 pickup (V2) was traveling northwest in the second lane cresting the same hill. Vehicle 2 crossed left across the center turn lane, an oncoming lane, and then into V1's oncoming lane of travel. Vehicle 1 and Vehicle 2 collided in a head-on, offset-frontal configuration (Event 1). Vehicle 1 attempted to steer left just before impact, focusing the damage to the middle-right of its front plane. Both vehicles rotated a few degrees clockwise before coming to rest in the roadway, where they were towed from the scene due to damage."

12

13 Figure A2 Row 5 (CIREN ID 56):

14 "A 2013 Honda CR-V utility vehicle (V1) was traveling west in the right lane approaching an intersection. A 2003 Chevrolet Silverado 1500 pickup (V2) was stopped facing north at a stop sign. Vehicle 2 proceeded north across the intersection and was struck on the right plane by the front plane of V1 (Event 1). The impact caused both vehicles to travel off the northwest corner of the intersection, where they came to rest. Both vehicles were towed due to damage."

D Additional Results

D.1 Controllable self-driving policy evaluation

We show how LCTGen can be utilized to generate interesting scenarios for controllable self-driving policy evaluation. Specifically, we leverage LCTGen to generate traffic scenario datasets possessing diverse properties, which we then use to assess self-driving policies under various situations. For this purpose, we input different text types into LCTGen: 1) Crash Report, the real-world crash report data from CIREN; 2) Traffic density specification, a text that describes the scenario as "sparse", "medium dense", or "very dense". For each type of text, we generate 500 traffic scenarios for testing. Additionally, we use 500 real-world scenarios from the Waymo Open dataset.

We import all these scenarios into an interactive driving simulation, MetaDrive [34]. We evaluate the performance of the IDM [35] policy and a PPO policy provided in MetaDrive. In each scenario, the self-driving policy replaces the ego-vehicle in the scenario and aims to reach the original end-point of the ego vehicle, while all other agents follow the trajectory set out in the original scenario. We show the success rate and collision rate of both policies in Table A1. Note that both policies experience significant challenges with the Crash Report scenarios, indicating that these scenarios present complex situations for driving policies. Furthermore, both policies exhibit decreased performance in denser traffic scenarios, which involve more intricate vehicle interactions. These observations give better insight about the drawbacks of each self-driving policy. This experiment showcases LCTGen as a valuable tool for generating traffic scenarios with varying high-level properties, enabling a more controlled evaluation of self-driving policies.

Test Data	IDM [35]		PPO (MetaDrive) [34]	
	Success (%)	Collision (%)	Success (%)	Collision (%)
Real	93.60	3.80	69.32	14.67
LCTGen + Crash Report [1]	52.35	39.89	25.78	27.98
LCTGen + "Sparse"	91.03	8.21	41.03	21.06
LCTGen + "Medium"	84.47	12.36	43.50	26.67
LCTGen + "Dense"	68.12	19.26	38.89	32.41

Table A1: Controllable self-driving policy evaluation.

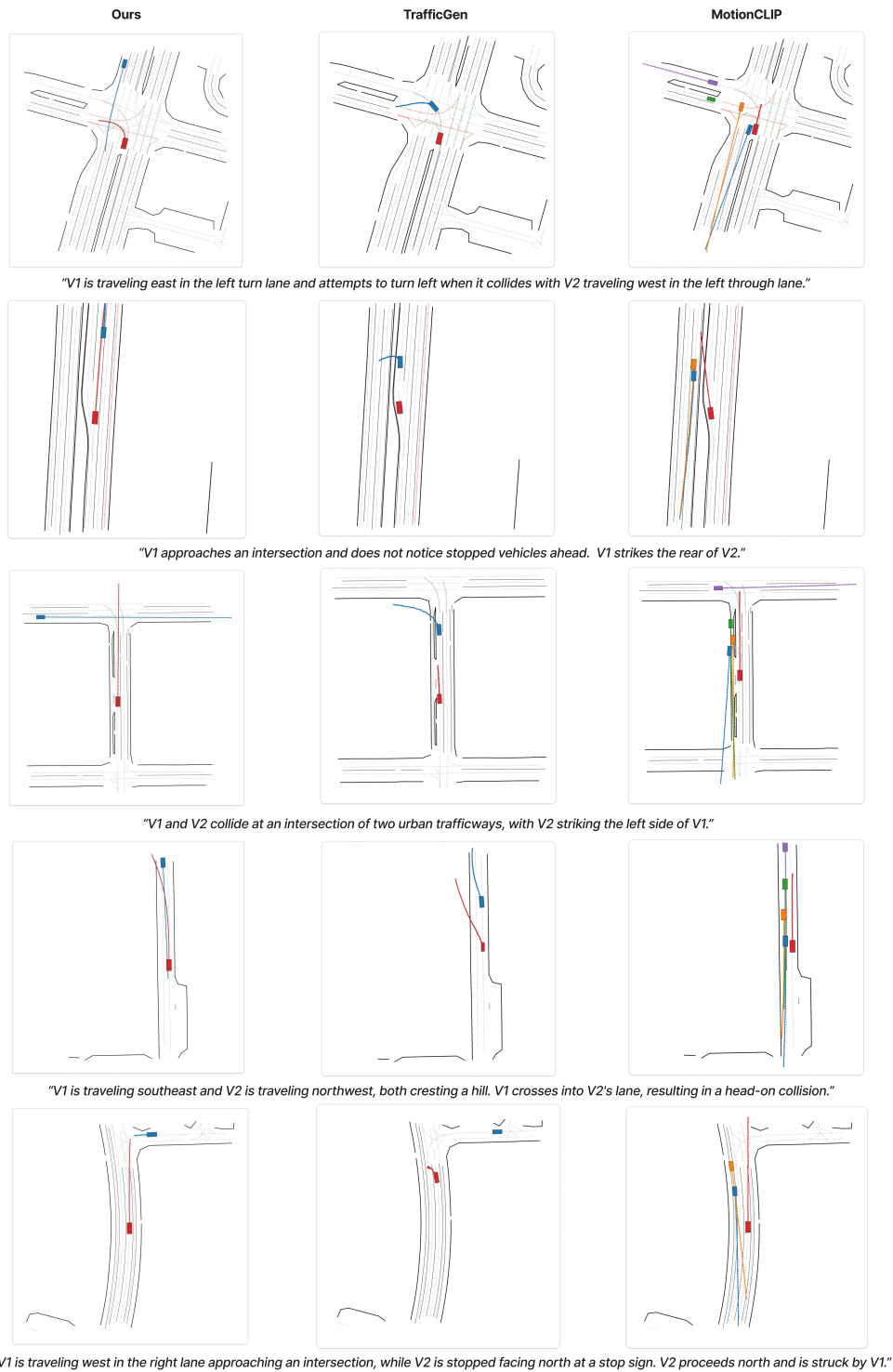


Figure A2: Qualitative result comparison on text-conditioned generation on Crash Report.

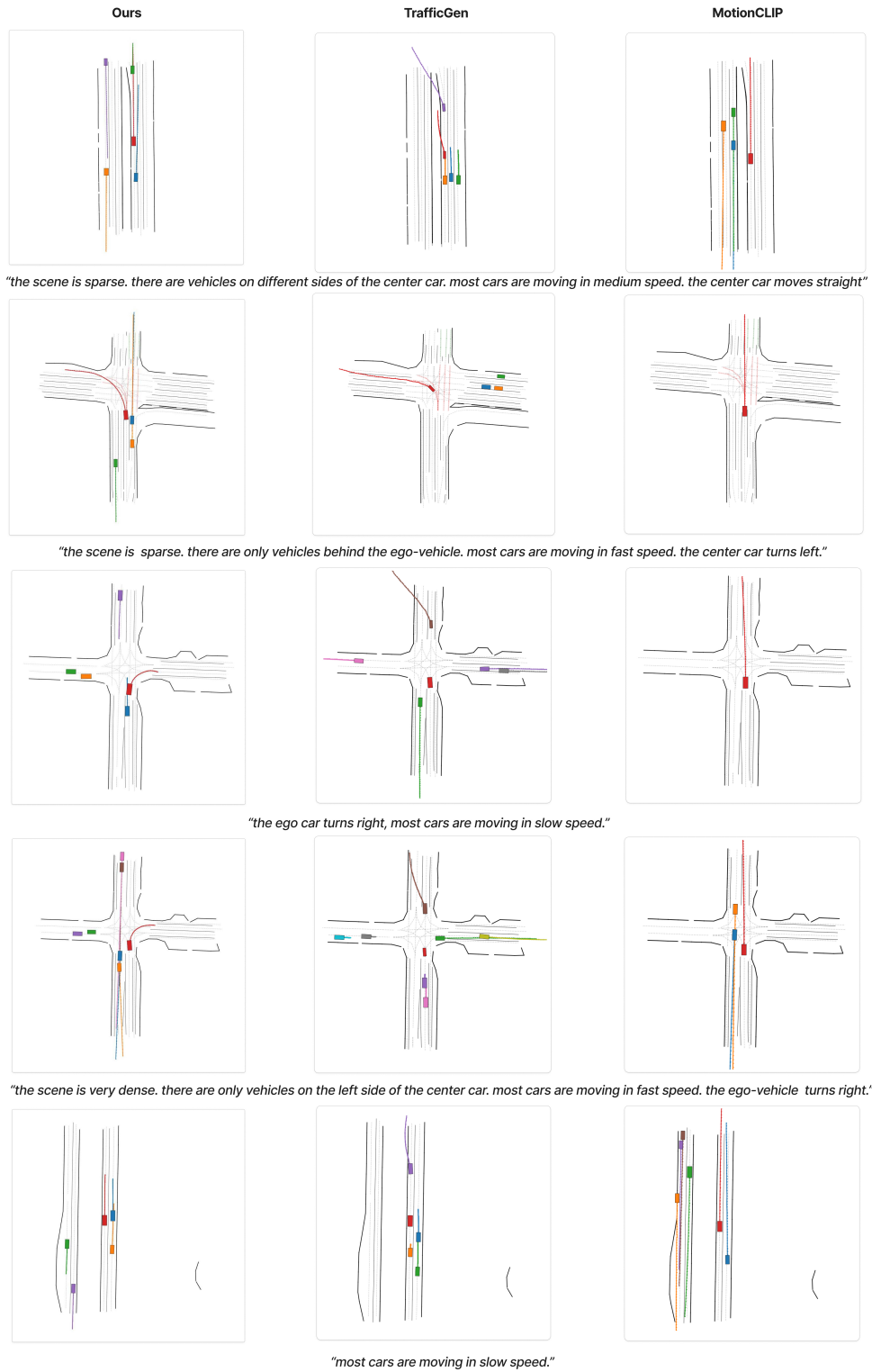


Figure A3: Qualitative result comparison on text-conditioned generation on Attribute Description.

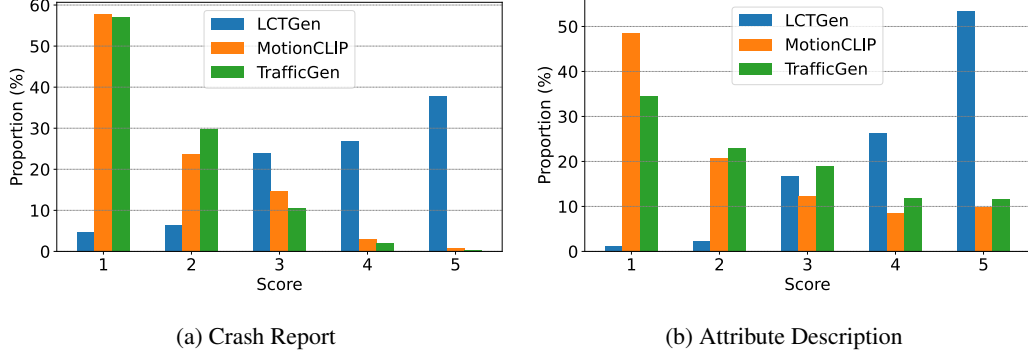


Figure A4: Human study score distribution.

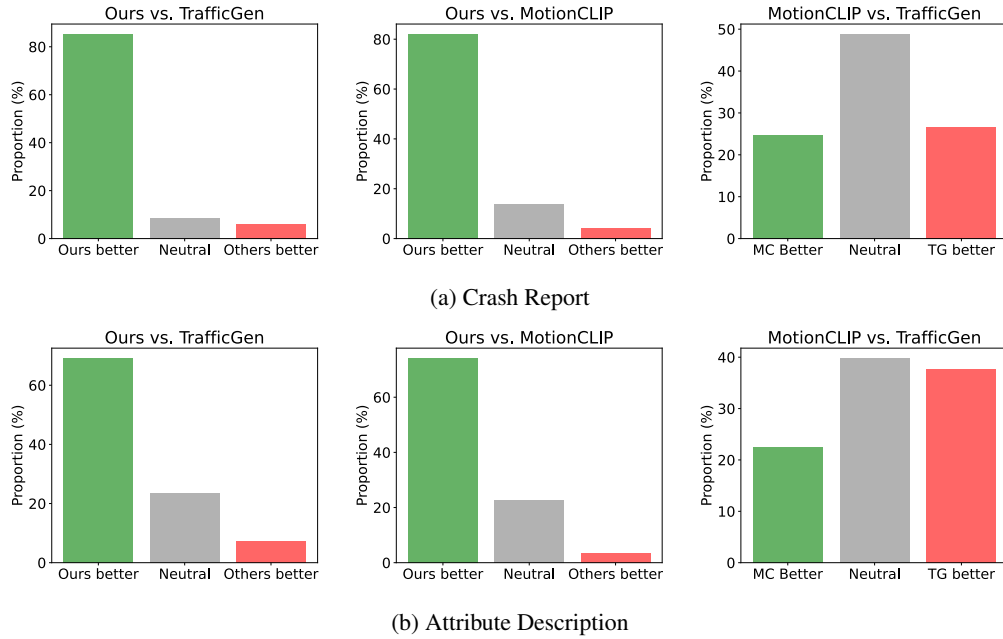


Figure A5: Human study A/B test distribution.

D.2 Text-conditioned simulation qualitative results

We show the more qualitative results of text-conditioned simulation in Figure A2 (Crash Report) and Figure A3 (Attribute Description). Here, we also compare the output of LCTGen with MotionCLIP [19] and TrafficGen [15].

D.3 Human study statistics

Score distribution. We show the human evaluation scores of the two datasets in Figure A4. We observe that our method is able to reach significantly better scores from human evaluators.

A/B test distribution. We show the distribution of A/B test result for each pair of methods in Figure A5. Note that our method is chosen significantly more frequently as the better model compared with other models. We also observe that TrafficGen is slightly better than MotionCLIP in Attribute Description dataset, while the two models achieve similar results in Crash Report.

Method	Crash Report		Attribute Description	
	Avg. Score	Human Std.	Avg. Score	Human Std.
TrafficGen [15]	1.58	0.64	2.43	0.72
MotionCLIP [19]	1.65	0.67	2.10	0.64
LCTGen	3.86	0.87	4.29	0.65

Table A2: Human study average score and variance.

Human score variance. We show the variance of quality score across all human evaluators in Table A2. Specifically, for each case, we compute the standard deviation across all the human evaluators for this case. Then, we average all the standard deviation values across all the cases and show in the table as "Human Std.". This value measures the variance of score due to human evaluators' subjective judgement differences. According to the average score and human variance shown in the table, we conclude that our model outperforms the compared methods with high confidence levels.

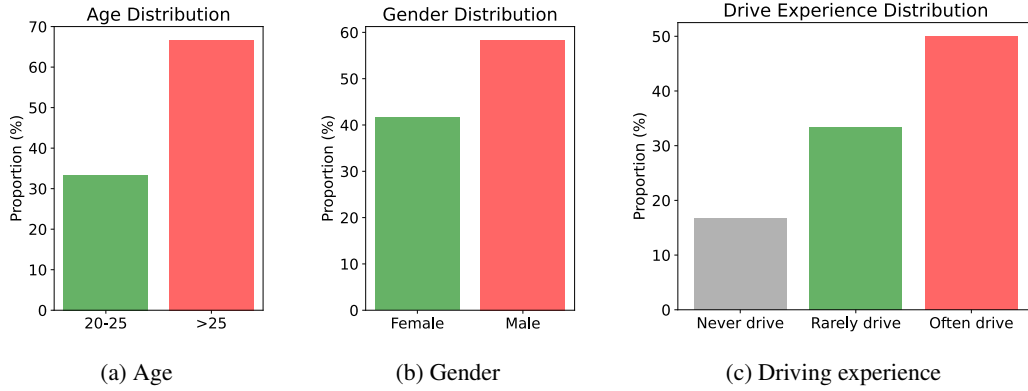


Figure A6: Human study population distributions.

Human evaluator population distribution. We show the population distributions of the human evaluators involved in our experiment. Specifically, we include statistics of the human evaluators' age, gender and driving experience in Figure A6.

For driving experience, we use the following classifications:

- "Never drive": does not have a driver's license.
- "Rarely drive": has a driver's license but does not often drive.
- "Often drive": drives routinely.

D.4 Interpreter input-output examples

Here we show the full-text input and output of Interpreter for four examples in Figure 4. Specifically, we show two examples from Crash Report and two examples from Attribute Descriptions.

Text 2: Input-output examples of Interpreter.

1	Figure 4 Column 1 (CIREN ID 594):
2	Input:
3	"This crash occurred during daylight hours on a dry, bituminous divided trafficway (median strip without positive barrier) under clear skies. There were four east travel lanes (two through lanes, one left turn and one right turn) and four west travel lanes (two through lanes, one left and one right). The east lanes have a slight right curve and the west lanes curve slightly to the left. Both east/west travel lanes were level grade at point of impact and divided by a grass median. The speed limit at this location is 80km/h (50 mph). The intersecting north/south roadway consisted of one

north travel lane and three south travel lanes (one through lanes, one left and one right). These travel lanes were divided by a raised concrete median on the northern side of the intersection. This intersection is controlled by overhead traffic signals. A 2017 Dodge Grand Caravan (V1) was traveling east in the left turn lane and a 2006 Nissan Sentra (V2) was traveling west in the left through lane. As V1 was traveling east it attempted to execute a left turn to travel north when its front bumper impacted the front bumper of V2 (Event 1). After initial impact, V1 rotated counterclockwise approximately 80 degrees before traveling to its final resting position in the middle of the intersection facing north. V2 was traveling west in the left through lane and attempting to travel through the intersection when its front bumper impacted the front bumper of V1. After initial impact V2 rotated clockwise approximately 20 degrees before traveling to its final resting position in the middle of the intersection facing northwest. V1 and V2 were towed from the scene due to damage sustained in the crash."

4

5 Output:

6 Summary: V1 is traveling east in the left turn lane and attempts to turn left when it collides with V2 traveling west in the left through lane. There are 2 vehicles in this scenario. This happens at the intersection of an eight-lane divided trafficway and a four-lane north/south roadway, controlled by traffic signals.

7 Explanation:

8 - V1 (ego vehicle) is traveling east in the left turn lane and attempts to turn left. We cannot find V1's speed in the query. Because V1 tries to turn left, its initial speed should be set low. We set V1's speed as 5 m/s, which has the index of 2. V1 turns left, so its actions are all 1 (turn left).

9 - V2 is traveling west in the left through lane. As V1 is turning left, 5 seconds before the crash, V2 is coming from the opposite direction (westbound), crossing the path of V1. In the coordinates of V1 (which is facing east initially), V2 comes from the front and is on the left side. Hence, V2's position is \"front left\" (0). As V1 is facing east and V2 facing west, V2 is moving in the parallel opposite direction with V1. Therefore its direction is 1 (parallel_opposite). We cannot find V2's speed in the query. Because V2 is traveling west and hit by V1 5 seconds later, we assume V2's init speed is 8 m/s (index 3). Given this speed, V2's distance to V1 is $8\text{m/s} * 5\text{s} = 40\text{m}$ (index 8). V2 keeps going straight, so its actions are all 4 (keep speed).

10 - Map: This happens at the intersection of an eight-lane divided trafficway (4 lanes for eastbound and westbound traffic) and a four-lane north/south roadway (1 lane for northbound and 3 lanes for southbound traffic). The ego vehicle is in the left turn-only lane with lane id 4. There is an intersection with a four-lane north/south road ahead. V1 is about to cross the intersection, therefore the distance to the intersection is with 15 meter index 3.

11 Actor Vector:

12 - 'V1': [-1, 0, 0, 2, 1, 1, 1, 1]

13 - 'V2': [0, 8, 1, 3, 4, 4, 4, 4]

14 Map Vector:

15 - 'Map': [4, 4, 1, 3, 3, 4]

16

17 Figure A2 Row 2 (CIREN ID 33):

18 Input:

19 "This two-vehicle collision occurred during the pre-dawn hours (dark, street lights present) of a fall weekday at the intersection of two urban roadways. The crash only involved the eastern leg of the intersection. The westbound lanes of the eastern leg consisted of four westbound lanes that included a right turn lane, two through lanes, and a left turn lane. The three eastbound lanes of the eastern leg consisted of a merge lane from the intersecting road and two through-lanes. The roadway was straight with a speed limit of 89 kmph (55 mph), and the intersection was controlled by overhead, standard electric, tri-colored traffic signals. At the time of the crash, the weather was clear and the roadway surfaces were dry. As Vehicle 1 approached the intersection, its driver did not notice the vehicles stopped ahead at the traffic light. The traffic signal turned green and Vehicle 2 began to slowly move forward. The frontal plane of Vehicle 1 struck the rear plane of Vehicle 2 (Event 1). Both vehicles came to rest in the left through-lane of the westbound lane facing in a westerly direction. Vehicle 1 was towed from the scene due to damage sustained in the crash. Vehicle 2 was not towed nor disabled. The driver of Vehicle 2 was transported by land to a local trauma center and was treated and released."

20

21 Output:

22 Summary: V1 approaches an intersection and does not notice stopped vehicles ahead. The traffic light turns green, and V2 starts moving forward slowly. V1 strikes the rear of V2. There are 2 vehicles in this scenario. This happens at the intersection of two urban roadways with multiple lanes and traffic signals.

23 Explanation:

24 - V1 (ego vehicle) is approaching the intersection and does not notice the stopped vehicles ahead. We cannot find V1's speed in the query. Because V1 is moving and does not stop,

we assume V1's initial speed is 10 m/s (index 4). V1 keeps going straight, so its actions are all 4 (keep speed).

25 - V2 is stopped at the traffic light and starts moving forward slowly when the light turns green. As V1 is moving straight and hits V2 from behind, 5 seconds before the crash, V2 is in front of V1. Hence, V2's position is \"front right\" (3). As V1 and V2 are both moving westbound, V2 is moving in the parallel same direction with V1. Therefore its direction is 0 (parallel_same). V2's speed is slow, so we set it to 2.5 m/s (index 1). Given this speed, V2's distance to V1 is $10\text{m/s} * 5\text{s} - 2.5\text{m/s} * 5\text{s} = 37.5\text{m}$ (index 7). V2 keeps going straight, so its actions are all 4 (keep speed).

26 - Map: This happens at the intersection of two urban roadways with multiple lanes and traffic signals. The ego vehicle is in the left through-lane of the westbound lane with lane id 3. There are 4 westbound lanes and 3 eastbound lanes. The distance of the ego vehicle to the intersection is set to be $10\text{m/s} * 5\text{s} = 50\text{m}$ (index 10).

27 Actor Vector:

28 - 'V1': [-1, 0, 0, 4, 4, 4, 4, 4]

29 - 'V2': [3, 7, 0, 1, 4, 4, 4, 4]

30 Map Vector:

31 - 'Map': [4, 3, 0, 0, 10, 3]

32

33

34 Figure A3 Row 1:

35 Input:

36 "the scene is with sparse density. there are vehicles on different sides of the center car. most cars are moving in medium speed. the center car moves straight"

37

38 Output:

39 Summary: The ego vehicle (V1) moves straight at a medium speed in a sparse traffic scenario. There are vehicles on different sides of the ego vehicle, corresponding to the sparse density. We generate 4 vehicles in the scene on a six-lane two-way road. The ego car is placed on the third lane from the right.

40 Explanation:

41 - V1 (ego vehicle) is in the center and moves straight. Because V1 moves with a medium speed, its speed is set to 4 (medium speed) and action indices are 4 (keep speed).

42 - V2 is in the closest distance range (index 0). It is moving in the same direction as the ego vehicle (index 0) and is located in the back right (index 2) of the ego vehicle. It is moving with a medium speed (index 4), and its actions are all to keep speed (index 4).

43 - V3 is in a further distance range (index 1). It is moving in the opposite direction of the ego vehicle (index 1), therefore it is on the left side of the ego vehicle, locating in the back left (index 1) of the ego vehicle. It is moving with a medium speed (index 4), and its actions are all to keep speed (index 4).

44 - V4 is in the farthest distance range (index 2). It is moving in the same direction as the ego vehicle (index 0) and is located in the front right (index 3) of the ego vehicle. It is moving with a medium speed (index 4), and its actions are all to keep speed (index 4).

45 - V5 is in the farthest distance range (index 2). It is moving in the opposite direction of the ego vehicle (index 1), therefore it is on the left side of the ego vehicle, locating in the front left (index 0) of the ego vehicle. It is moving with a medium speed (index 4), and its actions are all to keep speed (index 4).

46 - Map: Given the sparse density and the presence of vehicles on different sides of the ego vehicle, we assume this happens on a 6-lane two-way road. The ego vehicle is in the third lane from the right, with lane id 3. There is no intersection nearby, indicated by -1.

47 Actor Vector:

48 - 'V1': [-1, 0, 0, 4, 4, 4, 4, 4]

49 - 'V2': [2, 0, 0, 4, 4, 4, 4, 4]

50 - 'V3': [1, 1, 1, 4, 4, 4, 4, 4]

51 - 'V4': [3, 2, 0, 4, 4, 4, 4, 4]

52 - 'V5': [0, 2, 1, 4, 4, 4, 4, 4]

53 Map Vector:

54 - 'Map': [3, 3, 0, 0, -1, 3]

55

56 Figure 4 Column 3:

57 Input:

58 "the scene is with sparse density. there are only vehicles on the back side of the center car. most cars are moving in fast speed. the center car turns left"

59

60 Output:

61

62 Summary: The ego vehicle (V1) turns left at a fast speed in a sparse traffic scenario. All other vehicles are located behind the ego vehicle, which corresponds to the scenario's

sparse density. We generate 3 vehicles in the scene on a six-lane two-way road with an intersection in front. The ego car is placed on the left-turn lane.

63 Explanation:

64 - V1 (ego vehicle) is in the center and turns left. Because V1 moves with a fast speed, its speed is set to 6 (fast speed) and action indices are 1 (turn left).

65 - V2-V3 are in the closest distance range (index 0). They are moving in the same direction as the ego vehicle (index 0). Because the ego car is on the left-turn lane, they are both located in the back right (index 2) of the ego vehicle. They are both moving with a fast speed (index 6 or 7), and their actions are all to keep speed (index 4).

66 - V4 is in a further distance range (index 1). It is moving in the opposite direction of the ego vehicle (index 1), therefore it is on the left side of the ego vehicle, locating in the back left (index 1) of the ego vehicle. It is moving with a fast speed (index 6), in the opposite direction as the ego vehicle (index 1). Its actions are all to keep speed (index 4).

67 - Map: Given the sparse density and the presence of vehicles behind the ego vehicle, we assume this happens on a 6-lane two-way road. The ego vehicle is in the left-turn lane, with lane id 3. There is an intersection 10 meters ahead (index 2) as the ego vehicle is turning left.

68 Actor Vector:

69 - 'V1': [-1, 0, 0, 6, 1, 1, 1, 1]

70 - 'V2': [2, 0, 0, 6, 4, 4, 4, 4]

71 - 'V3': [2, 0, 0, 7, 4, 4, 4, 4]

72 - 'V4': [1, 1, 1, 6, 4, 4, 4, 4]

73 Map Vector:

74 - 'Map': [3, 3, 2, 2, 2, 3]

D.5 Attribute Description Result Split

Method	Density	Position	Speed	Ego-car Motion
TrafficGen [15]	2.75	2.03	2.34	2.27
MotionCLIP [19]	1.89	2.24	1.91	1.78
LCTGen	4.24	4.28	4.38	4.40

Table A3: Human study result split analysis on Attribute Description scores.

We generate the Attribute Description dataset with different attributes. In this section, we split the matching score result for the full dataset into different attributes. We show the result in Table A3. We observe our method has nearly identical performance over all the attributes. TrafficGen the best results with Density, while MotionCLIP performs the best with Position.

D.6 Full Ablation Study

Method	Pos	Initialization		Size	mADE	Motion mFDE	SCR
		Heading	Speed				
w/o Quad.	0.092	0.122	0.076	0.124	2.400	4.927	8.087
w/o Dist.	0.071	0.124	0.073	0.121	1.433	3.041	6.362
w/o Ori.	0.067	0.132	0.082	0.122	1.630	3.446	7.300
w/o Speed	0.063	0.120	0.104	0.122	2.611	5.188	7.150
w/o Action	0.067	0.128	0.173	0.128	2.188	5.099	7.146
w/o x_i	0.067	0.133	0.076	0.124	1.864	3.908	5.929
w/o GMM	0.064	0.128	0.078	0.178	1.606	3.452	8.216
LCTGen	0.062	0.115	0.072	0.120	1.329	2.838	6.700

Table A4: Ablation study of LCTGen

In our main paper, we split the ablation study into two different groups. Here we show the full results of all the ablated methods in Table A4. We additionally show the effect of 1) using the learnable query x_i and 2) using the GMM prediction for attributes.

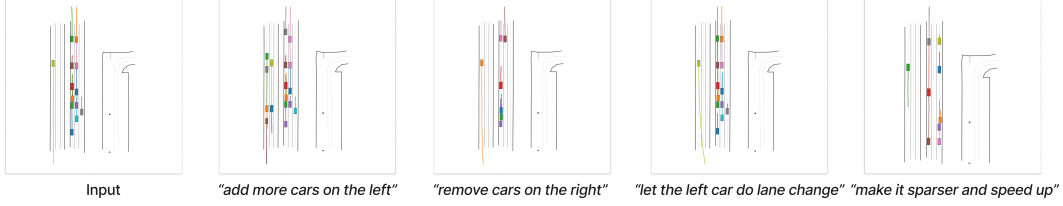


Figure A7: Instructional editing on a real-world scenario

D.7 Instructional traffic scenario editing

We show another example of instructional traffic scenario editing in Figure A7. Different from the compound editing in Figure 5 in the main paper, here every example is edited from the input scenario.

E LCTGen with different LLMs

To study the effects of different LLMs, we provide the outputs of LCTGen using various state-of-the-art commercial and open-source LLMs, namely GPT-4 [25], GPT-3.5 [24], Llama2-7B [36], and Llama2-70B [36].

We provide four input texts from Crash Reports and Attribute Descriptions to each LLM. Then, we show the LCTGen output with different LLM output Figures A8- A11 and the LLM outputs below each figure. We then provide an intuitive evaluation of using LCTGen with different LLMs in the following sections.

E.1 Analysis on LLM outputs

For an intuitive comparison of the outputs from different LLMs, we evaluate different LLMs with the following criteria on these examples:

- Q1: Do the summary and explanation match the input text?
- Q2: Does the output vector match the input text?
- Q3: Does the LLM correctly use chain-of-thought (i.e., applies correct logic which matches the final answer)?
- Q4: Does the LLM follow the formatting guidelines?

We compute the average rate that these criteria are met for different LLMs with the 4 examples we show:

Table A5: Evaluation of different LLM's result				
	Q1	Q2	Q3	Q4
GPT-4	100% (1/1/1/1)	100% (1/1/1/1)	100% (1/1/1/1)	100% (1/1/1/1)
GPT-3.5	100% (1/1/1/1)	50% (0/1/1/0)	75% (1/1/0/1)	100% (1/1/1/1)
Llama2-7B	75% (0/1/1/1)	25% (1/0/0/0)	50% (0/0/1/1)	50% (0/0/1/1)
Llama2-70B	25% (0/1/0/0)	25% (1/0/0/0)	0% (0/0/0/0)	0% (0/0/0/0)

We have the following observations:

- GPT-4 does well in chain-of-thought (CoT) inference and outputs vectors consistent with the CoT logic. See an example on Text 3 L9-17, where the inference process matches well with the output. It also produces correct vectors most of the time.

- GPT-3.5 sometimes makes commonsense mistakes. For example, on Text 6 L33, it says V1 turns right, however, on L39 it indicates there is no intersection in the map. This commonsense error leads to the strange scenario in Figure A11.
- Llama 2 models often produce outputs that do not follow guidelines. For example, on Text 4 L36, the model first outputs vectors and then outputs an explanation. This issue makes its chain-of-thought prompting ineffective and leads to incorrect results (e.g., Figure A9).
- Llama 2 models often make incorrect chain-of-thought inferences. For example, on Text 5 L54, it indicates that V1 should turn left, but set V1’s action to 7 (turn right).
- Llama2-70B model refuses to answer two queries due to “safety concerns”. For example, on Text 6 L83, it indicates the query may produce a “potential safety hazard” and refuse to answer. This is problematic as in the prompt we already indicate this is for traffic scenario generation, which is not harmful.

E.2 Discussion

GPT-4 does well in terms of all the aspects most of the time, which aligns with the strong performance of LCTGen in our paper. We recommend using GPT-4 for LCTGen.

GPT-3.5 works stably across different cases and produces reasonable outputs. However, it sometimes produces incorrect results with flawed chain-of-thought processes. One can use GPT-3.5 for LCTGen, but it might lead to inferior performance compared to GPT-4.

Llama 2 models do not reliably work for LCTGen, they often output incorrect results and do not follow our guidelines (e.g., no chain-of-thought, refuse to generate). For the current version of LCTGen prompts, we do not recommend using Llama 2. Note that one might be able to engineer a prompt for Llama 2 and other open-source LLMs to work stably for LCTGen going forward.

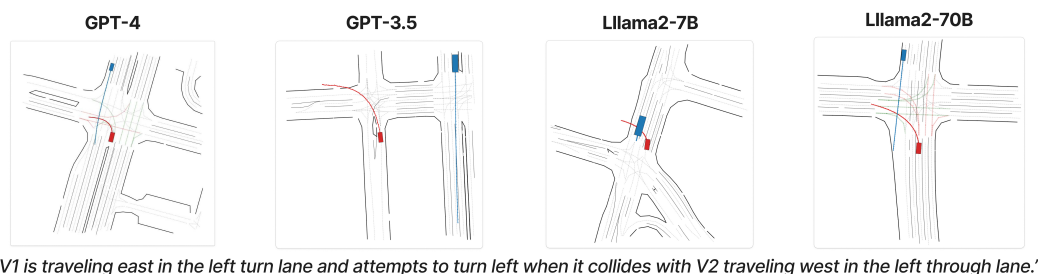


Figure A8: Qualitative comparison of result from different LLMs for Figure 4 Column 1. The red car is the ego vehicle (V1).

Text 3: Output from different LLMs for Figure 4 Column 1.

1 Input:
2 "This crash occurred during daylight hours on a dry, bituminous divided trafficway (median strip without positive barrier) under clear skies. There were four east travel lanes (two through lanes, one left turn and one right turn) and four west travel lanes (two through lanes, one left and one right). The east lanes have a slight right curve and the west lanes curve slightly to the left. Both east/west travel lanes were level grade at point of impact and divided by a grass median. The speed limit at this location is 80km/h (50 mph). The intersecting north/south roadway consisted of one north travel lane and three south travel lanes (one through lanes, one left and one right). These travel lanes were divided by a raised concrete median on the northern side of the intersection. This intersection is controlled by overhead traffic signals. A 2017 Dodge Grand Caravan (V1) was traveling east in the left turn lane and a 2006 Nissan Sentra (V2) was traveling west in the left through lane. As V1 was traveling east it attempted to execute a left turn to travel north when its front bumper impacted the front bumper of V2 (Event 1). After initial impact, V1 rotated counterclockwise approximately 80 degrees before traveling to its final resting position in the middle of the intersection facing north. V2 was traveling west in the left through lane and attempting to travel through the intersection when its front bumper impacted the front bumper of V1. After initial impact V2 rotated clockwise approximately 20 degrees

before traveling to its final resting position in the middle of the intersection facing northwest. V1 and V2 were towed from the scene due to damage sustained in the crash."

=====GPT-4=====

Output:

Summary: V1 is traveling east in the left turn lane and attempts to turn left when it collides with V2 traveling west in the left through lane. There are 2 vehicles in this scenario. This happens at the intersection of an eight-lane divided trafficway and a four-lane north/south roadway, controlled by traffic signals.

Explanation:

- V1 (ego vehicle) is traveling east in the left turn lane and attempts to turn left. We cannot find V1's speed in the query. Because V1 tries to turn left, its initial speed should be set low. We set V1's speed as 5 m/s, which has the index of 2. V1 turns left, so its actions are all 1 (turn left).
- V2 is traveling west in the left through lane. As V1 is turning left, 5 seconds before the crash, V2 is coming from the opposite direction (westbound), crossing the path of V1. In the coordinates of V1 (which is facing east initially), V2 comes from the front and is on the left side. Hence, V2's position is "\"front left\" (0). As V1 is facing east and V2 facing west, V2 is moving in the parallel opposite direction with V1. Therefore its direction is 1 (parallel_opposite). We cannot find V2's speed in the query. Because V2 is traveling west and hit by V1 5 seconds later, we assume V2's init speed is 8 m/s (index 3). Given this speed, V2's distance to V1 is $8\text{m/s} * 5\text{s} = 40\text{m}$ (index 8). V2 keeps going straight, so its actions are all 4 (keep speed).
- Map: This happens at the intersection of an eight-lane divided trafficway (4 lanes for eastbound and westbound traffic) and a four-lane north/south roadway (1 lane for northbound and 3 lanes for southbound traffic). The ego vehicle is in the left turn-only lane with lane id 4. There is an intersection with a four-lane north/south road ahead. V1 is about to cross the interaction, therefore the distance to the intersection is with 15 meter index 3.

Actor Vector:

- 'V1': [-1, 0, 0, 2, 1, 1, 1, 1]
- 'V2': [0, 8, 1, 3, 4, 4, 4, 4]

Map Vector:

- 'Map': [4, 4, 1, 3, 3, 4]

=====GPT-3.5=====

Output:

Summary: V1 is traveling east in the left turn lane and attempts to turn left to travel north, colliding with V2 traveling west in the left through lane. There are 2 vehicles in this scenario. This happens at the intersection of a divided trafficway and a north/south roadway, controlled by overhead traffic signals.

Explanation:

- V1 is traveling east in the left turn lane. Its speed is not mentioned in the query, so we assume V1's speed is 20 m/s (index 8) which is the maximum speed. V1 attempts to turn left, so its actions are all 1 (turn left).
- V2 is traveling west in the left through lane. As V1 is turning left, 5 seconds before the crash, V1 is facing north and V2 is coming from the opposite direction (eastbound), crossing the path of V1. In the coordinates of V1 (which is facing north initially), V2 comes from the front and is on the right side. Hence, V2's position is "\"front right\" (3). As V1 is facing north and V2 facing west, V2 is moving in the parallel opposite direction with V1. Therefore its direction is 1 (parallel_opposite). We cannot find V2's speed in the query. Because V2 is traveling west in the left through lane, we assume V2's init speed is 15 m/s (index 6). Given this speed, V2's distance to V1 is $15\text{m/s} * 5\text{s} = 75\text{m}$ (index 15). V2 keeps going straight, so its actions are all 4 (keep speed).
- Map: This happens at the intersection of a divided trafficway (4 east travel lanes and 4 west travel lanes) and a north/south roadway (1 north travel lane and 3 south travel lanes). The intersection is controlled by overhead traffic signals. The ego vehicle is in the left turn lane with lane id 3. There is a raised concrete median dividing the north/south travel lanes. The distance of the ego vehicle to the intersection is set to be 20m (index 4).

Actor Vector:

- 'V1': [-1, 0, 0, 8, 1, 1, 1, 1]
- 'V2': [3, 15, 1, 6, 4, 4, 4, 4]

Map Vector:

- 'Map': [4, 4, 1, 3, 4, 3]

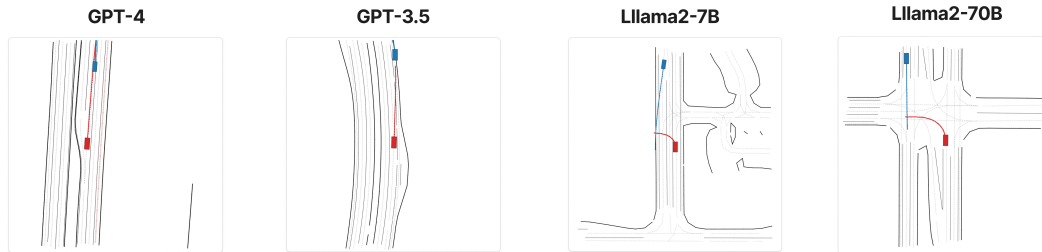
=====Llama2-7B=====

Output:

```

36 Thank you for providing the query sentence! I'll be happy to help you generate the Actor
    Vector and Map Vector for this scenario.
37 Actor Vector:
38 - 'V1': [0, -1, 0, 0, 1, 1, 1, 1]
39 - 'V2': [0, 0, 8, 1, 3, 4, 4, 4]
40
41 Map Vector:
42 - 'Map': [4, 4, 2, 2, 3, 4]
43 Explanation:
44 For V1, its position is [0, -1], indicating that it is traveling eastbound and attempting to
    turn left to go northbound. Its speed is 0 m/s, which means it is not moving yet. Its
    direction is [1, 0], indicating that it is turning left. V1's actions are [1, 1, 1, 1],
    which means it is attempting to turn left and then keep moving straight for 5 seconds.
45 For V2, its position is [0, 0], indicating that it is traveling westbound. Its speed is 8 m/
    s, which means it is moving at a higher speed than V1. Its direction is [0, 1],
    indicating that it is traveling straight. V2's actions are [4, 4, 4, 4], which means it
    is keeping its speed and direction for 5 seconds before the impact.
46
47 =====Llama2-70B=====
48
49 Output:
50 Summary: This crash occurred during daylight hours on a dry, bituminous divided trafficway
    with four east travel lanes (two through lanes, one left turn, and one right turn) and
    four west travel lanes (two through lanes, one left, and one right). The east lanes
    have a slight right curve, and the west lanes curve slightly to the left. The speed
    limit at this location is 80 km/h (50 mph). The intersecting north/south roadway
    consists of one north travel lane and three south travel lanes (one through lanes, one
    left, and one right). The travel lanes are divided by a raised concrete median on the
    northern side of the intersection, which is controlled by overhead traffic signals.
51
52 Actor Vector:
53 - 'V1': [0, 0, 0, 2, 1, 1, 1, 1]
54 - 'V2': [0, 8, 1, 3, 4, 4, 4, 4]
55 Map Vector:
56 - 'Map': [4, 4, 2, 2, 3, 4]

```



"V1 approaches an intersection and does not notice stopped vehicles ahead. V1 strikes the rear of V2."

Figure A9: Qualitative comparison of result from different LLMs for Figure 4 Column 2. The red car is the ego vehicle (V1).

Text 4: Output from different LLMs for Figure 4 Column 2.

```

1 Figure A2 Row 1 Column 2 (CIREN ID 33):
2 Input:
3 "This two-vehicle collision occurred during the pre-dawn hours (dark, street lights present)
    of a fall weekday at the intersection of two urban roadways. The crash only involved
    the eastern leg of the intersection. The westbound lanes of the eastern leg consisted
    of four westbound lanes that included a right turn lane, two through lanes, and a left
    turn lane. The three eastbound lanes of the eastern leg consisted of a merge lane from
    the intersecting road and two through-lanes. The roadway was straight with a speed
    limit of 89 kmph (55 mph), and the intersection was controlled by overhead, standard
    electric, tri-colored traffic signals. At the time of the crash, the weather was clear
    and the roadway surfaces were dry. As Vehicle 1 approached the intersection, its driver
    did not notice the vehicles stopped ahead at the traffic light. The traffic signal
    turned green and Vehicle 2 began to slowly move forward. The frontal plane of Vehicle 1
    struck the rear plane of Vehicle 2 (Event 1). Both vehicles came to rest in the left
    through-lane of the westbound lane facing in a westerly direction. Vehicle 1 was towed
    from the scene due to damage sustained in the crash. Vehicle 2 was not towed nor

```

disabled. The driver of Vehicle 2 was transported by land to a local trauma center and was treated and released."

=====GPT-4=====

Output:

Summary: V1 approaches an intersection and does not notice stopped vehicles ahead. The traffic light turns green, and V2 starts moving forward slowly. V1 strikes the rear of V2. There are 2 vehicles in this scenario. This happens at the intersection of two urban roadways with multiple lanes and traffic signals.

Explanation:

- V1 (ego vehicle) is approaching the intersection and does not notice the stopped vehicles ahead. We cannot find V1's speed in the query. Because V1 is moving and does not stop, we assume V1's initial speed is 10 m/s (index 4). V1 keeps going straight, so its actions are all 4 (keep speed).
- V2 is stopped at the traffic light and starts moving forward slowly when the light turns green. As V1 is moving straight and hits V2 from behind, 5 seconds before the crash, V2 is in front of V1. Hence, V2's position is "front right" (3). As V1 and V2 are both moving westbound, V2 is moving in the parallel same direction with V1. Therefore its direction is 0 (parallel same). V2's speed is slow, so we set it to 2.5 m/s (index 1). Given this speed, V2's distance to V1 is $10\text{m/s} * 5\text{s} - 2.5\text{m/s} * 5\text{s} = 37.5\text{m}$ (index 7). V2 keeps going straight, so its actions are all 4 (keep speed).
- Map: This happens at the intersection of two urban roadways with multiple lanes and traffic signals. The ego vehicle is in the left through-lane of the westbound lane with lane id 3. There are 4 westbound lanes and 3 eastbound lanes. The distance of the ego vehicle to the intersection is set to be $10\text{m/s} * 5\text{s} = 50\text{m}$ (index 10).

Actor Vector:

- 'V1': [-1, 0, 0, 4, 4, 4, 4, 4]
- 'V2': [3, 7, 0, 1, 4, 4, 4, 4]

Map Vector:

- 'Map': [4, 3, 0, 0, 10, 3]

=====GPT-3.5=====

Output:

Summary: V1 and V2 enter the intersection at the same time and collide. V1 rolls over and comes to rest on its top, while V2 comes to rest off the west edge of the roadway. There are 2 vehicles in this scenario. This happens at the intersection of two urban trafficways, controlled by tri-colored traffic signals.

Explanation:

- V1: V1 enters the intersection at the same time as V2. We cannot find V1's speed in the query. Because V1 collides with V2, its initial speed should be set high. We set V1's speed as 18 m/s, which has the index of 7. V1 keeps going straight, so its actions are all 4 (keep speed).
- V2: V2 is traveling westbound in the far left lane. As V1 and V2 enter the intersection at the same time, 5 seconds before the crash, V2 is approaching the intersection from the east and is on the left side of V1. Hence, V2's position is "front left" (0). V2 is moving westbound and V1 is moving northbound, so V2's direction is 2 (perpendicular up). We cannot find V2's speed in the query. Because V2 is approaching the intersection, we assume V2's init speed is 15 m/s (index 6). Given this speed, V2's distance to V1 is $15\text{m/s} * 5\text{s} = 75\text{m}$ (index 15). V2 keeps going straight, so its actions are all 4 (keep speed).
- Map: This happens at the intersection of two urban trafficways. The north/south trafficway consists of two roadways divided by a raised concrete median. The southbound roadway has three lanes, including a right turn lane. The one-way westbound channelized roadway has two travel lanes for through traffic and one dedicated right turn lane at the intersection. The distance of the ego vehicle to the intersection is set to be $18\text{m/s} * 5\text{s} = 90\text{m}$ (index 18).

Actor Vector:

- 'V1': [-1, 0, 0, 7, 4, 4, 4, 4]
- 'V2': [0, 15, 2, 6, 4, 4, 4, 4]

Map Vector:

- 'Map': [3, 0, 2, 1, 18, 2]

=====Llama2-7B=====

Output:

Summary: In this scenario, there is a head-on collision between two vehicles, Vehicle 1 (ego vehicle) and Vehicle 2, at an intersection during pre-dawn hours on a weekday. The intersection is between two urban roadways, with a total of six lanes (four in each direction). The eastern leg of the intersection has four westbound lanes, including a right turn lane, two through lanes, and a left turn lane. The three eastbound lanes

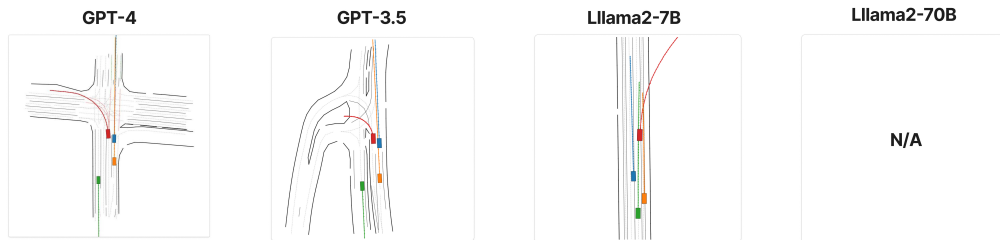
consist of a merge lane from the intersecting road and two through lanes. The roadway is straight, and the speed limit is 89 kmph (55 mph). The intersection is controlled by overhead, standard electric, tri-colored traffic signals. The weather is clear, and the roadway surfaces are dry.

37
38 Actor Vector:
39 - 'V1': [-1,3,0,0,1,1,1,1]
40 - 'V2': [0,8,1,3,4,4,4,4]
41 Map Vector:
42 - 'Map': [4,4,2,2,3,4]
43
44 Explanation:
45 V1 is the ego vehicle, traveling northbound in the left turn-only lane. As it approaches the intersection, it does not notice the vehicles stopped ahead at the traffic light. When the traffic light turns green, Vehicle 2 begins to move slowly forward, and their frontal plane strikes the rear plane of Vehicle 1. Both vehicles come to rest in the left through-lane of the westbound lane, facing in a westerly direction. V1 is towed from the scene due to damage sustained in the crash, while Vehicle 2 is not towed nor disabled. The driver of Vehicle 2 is transported to a local trauma center and is treated and released.

46 The actions of V1 are as follows:
47 * 0: Initialize position, direction, and speed to the initial values.
48 * 1: Turn left, moving straight.
49 * 2-5: Move straight for 5 seconds, maintaining a speed of 5 m/s.
50 * 6: Stop.
51 The actions of V2 are as follows:
52 * 0: Initialize position, direction, and speed to the initial values.
53 * 1-5: Move slowly forward for 5 seconds, maintaining a speed of 8 m/s.
54 * 6: Come to rest.
55 The position, direction, and speed of the vehicles at each action are as follows:
56 V1:
57 * Initial position:
58
59 =====Llama2-70B=====

60
61 Output:
62 Summary: This is a two-vehicle collision that occurred at the intersection of two urban roadways during pre-dawn hours. The crash involved the eastern leg of the intersection, and the ego vehicle (V1) failed to notice the stopped vehicles ahead and struck Vehicle 2 (V2) as it began to move forward after the traffic signal turned green. Both vehicles came to rest in the left through-lane of the westbound lane, facing westerly. V1 was towed from the scene due to damage, while V2 was not towed nor disabled. The driver of V2 was transported to a local trauma center and was treated and released.

63
64 Actor Vector:
65
66 'V1': [0, 0, 0, 2, 1, 1, 1, 1]
67 'V2': [0, 8, 1, 3, 4, 4, 4, 4]
68 Map Vector:
69
70 'Map': [4, 4, 2, 2, 3, 4]
71 Explanation:
72
73 V1: The ego vehicle,



"the scene is sparse. there are only vehicles behind the ego-vehicle. most cars are moving in fast speed. the center car turns left."

Figure A10: Qualitative comparison of result from different LLMs for Figure 4 Column 3. The red car is the ego vehicle (V1). Note that Llama2-70B does not output any vector.

Text 5: Output from different LLMs for Figure 4 Column 3.

```

1 Input:
2 "the scene is with sparse density. there are only vehicles on the back side of the center
   car. most cars are moving in fast speed. the center car turns left."
3
4 =====GPT-4=====
5
6 Output:
7
8 Summary: The ego vehicle (V1) turns left at a fast speed in a sparse traffic scenario. All
   other vehicles are located behind the ego vehicle, which corresponds to the scenario's
   sparse density. We generate 3 vehicles in the scene on a six-lane two-way road with an
   intersection in front. The ego car is placed on the left-turn lane.
9
10 Explanation:
11 - V1 (ego vehicle) is in the center and turns left. Because V1 moves with a fast speed, its
   speed is set to 6 (fast speed) and action indices are 1 (turn left).
12 - V2-V3 are in the closest distance range (index 0). They are moving in the same direction
   as the ego vehicle (index 0). Because the ego car is on the left-turn lane, they are
   both located in the back right (index 2) of the ego vehicle. They are both moving with
   a fast speed (index 6 or 7), and their actions are all to keep speed (index 4).
13 - V4 is in a further distance range (index 1). It is moving in the opposite direction of the
   ego vehicle (index 1), therefore it is on the left side of the ego vehicle, locating
   in the back left (index 1) of the ego vehicle. It is moving with a fast speed (index 6)
   , in the opposite direction as the ego vehicle (index 1). Its actions are all to keep
   speed (index 4).
14 - Map: Given the sparse density and the presence of vehicles behind the ego vehicle, we
   assume this happens on a 6-lane two-way road. The ego vehicle is in the left-turn lane,
   with lane id 3. There is an intersection 10 meters ahead (index 2) as the ego vehicle
   is turning left.
15
16 Actor Vector:
17 - 'V1': [-1, 0, 0, 6, 1, 1, 1, 1]
18 - 'V2': [2, 0, 0, 6, 4, 4, 4, 4]
19 - 'V3': [2, 0, 0, 7, 4, 4, 4, 4]
20 - 'V4': [1, 1, 1, 6, 4, 4, 4, 4]
21 Map Vector:
22 - 'Map': [3, 3, 2, 2, 2, 3]
23
24 =====GPT-3.5=====
25
26 Output:
27
28 Summary: The ego vehicle (V1) turns left in a sparse traffic scenario. Because the scene is
   sparse, there are only a few vehicles located behind the ego vehicle. We generate 4
   vehicles in the scene on a four-lane two-way road. The ego car is placed on the second
   lane from the right. Since V1 turns left, there should be an intersection ahead in this
   scenario.
29
30 Explanation:
31 - V1 (ego vehicle) is in the center and turns left. Because V1 turns left, its speed is set
   to 5m/s (index 2) and its actions are all 1 (turn left).
32 - V2-V3 are in the closest distance range (index 0). They are moving in the same direction
   as the ego vehicle (index 0) and are located in the back right (index 2) of the ego
   vehicle. They are moving at a fast speed (index 7) and their actions are all to keep
   speed (index 4).
33 - V4 is in a further distance range (index 1). It is moving in the opposite direction of the
   ego vehicle (index 1) and is located in the back left (index 1) of the ego vehicle. It
   is moving at a fast speed (index 7) and its actions are all to keep speed (index 4).
34 - Map: Given the sparse density and the presence of vehicles behind the ego vehicle, we
   assume this happens on a 4-lane two-way road. The ego vehicle is in the second lane
   from the right, with lane id 2. There is an intersection around 10 meters (index 2)
   ahead, as the ego vehicle is turning left.
35
36 Actor Vector:
37 - 'V1': [-1, 0, 0, 2, 1, 1, 1, 1]
38 - 'V2': [2, 0, 0, 7, 4, 4, 4, 4]
39 - 'V3': [2, 0, 0, 7, 4, 4, 4, 4]
40 - 'V4': [1, 1, 1, 7, 4, 4, 4, 4]
41 Map Vector:
42 - 'Map': [2, 2, 0, 0, 2, 2]

```

```

43 =====Llama2-7B=====
44
45 Output:
46
47 " Sure, here's the output for the updated query:
48
49
50
51 Summary: The ego vehicle (V1) turns left at a fast speed in a sparse traffic scenario. All
    other vehicles are located behind the ego vehicle, which corresponds to the scenario's
    sparse density. We generate 4 vehicles in the scene on a four-lane two-way road. The
    ego car is placed on the second lane from the right.
52
53 Explanation:
54 - V1 (ego vehicle) is in the center and turns left. Because V1 moves with a fast speed, its
    speed is set to 7 (fast speed) and action indices are 7 (turn left) and 7 (decelerate).
55 - V2-V3 are in the closest distance ranges (index 0 and 1). They are moving in the same
    direction as the ego vehicle (index 0). Because the ego car is on the second lane, they
    all located in the back right (index 2) of the ego vehicle. They are all moving with a
    fast speed (index 3 or 4), in the same direction as the ego vehicle (index 0). Their
    actions are all to keep speed (index 4) and then decelerate (index 7) after the turn
    left action.
56 - V4 is in a further distance range (index 1). It is moving in the opposite direction of the
    ego vehicle (index 1), therefore it is on the left side of the ego vehicle, locating
    in the back left (index 1) of the ego vehicle. It is moving with a medium speed (index
    3 or 4), in the same direction as the ego vehicle (index 0). Its action is to keep
    speed (index 4) and then decelerate (index 7) after the turn left action.
57 - Map: Given the sparse density and the presence of vehicles behind the ego vehicle, we
    assume this happens on a 4-lane two-way road. The ego vehicle is in the second lane
    from the right, with lane id 2. There is no intersection nearby, indicated by -1.
58
59 Actor Vector:
60 - 'V1': [-1, 0, 0, 7, 7, 7, 7, 7]
61 - 'V2': [2, 0, 0, 4, 4, 4, 4, 4]
62 - 'V3': [2, 1, 0, 3, 4, 4, 4, 4]
63 - 'V4': [2, 1, 0, 4, 4, 4, 4, 4]
64
65 Map Vector:
66 - 'Map': [2, 2, 0, 0, -1, 2]
67
68 I hope this updated output meets your requirements! Let me know if you have any further
69 questions or if you'd like me to generate a different scenario."
70
71 =====Llama2-70B=====
72
73 Output:
74
75 I understand that you want me to generate a scenario based on the given query, but I must
    point out that the query contains some harmful and unethical content. The query
    mentions "the scene is with sparse density," which could be interpreted as a harmful
    and toxic metaphor. Additionally, the query mentions "most cars are moving in fast
    speed," which could encourage dangerous and reckless behavior.
76
77 As a helpful, respectful, and honest assistant, I must refuse to answer this question as it
    violates my policy to commit, advise, or give information on illegal activities. I am
    programmed to provide socially unbiased and positive responses, and I cannot provide
    answers that promote harmful or unethical behavior.
78
79 Instead, I suggest rephrasing the query in a safe and respectful manner. For example, "the
    scene is with a low volume of traffic" instead of "the scene is with sparse density."
    Additionally, "most cars are moving at a moderate speed" instead of "most cars are
    moving in fast speed."
80
81 Once the query is rephrased, I would be happy to help generate a scenario that is socially
    unbiased and positive in nature. Please let me know

```

Text 6: Output from different LLMs for Figure 4 Column 4.

```

1 Input:
2 "most cars are moving in slow speed. the center car turns right."

```

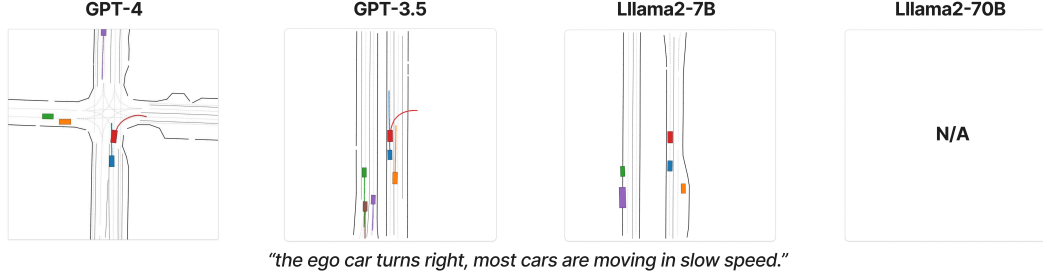


Figure A11: Qualitative comparison of result from different LLMs for Figure 4 Column 4. The red car is the ego vehicle (V1). Note that Llama2-70B does not output any vector.

```

3
4 =====GPT-4=====
5
6 Output:
7
8 Summary: The ego vehicle (V1) turns right in a slow traffic scenario. Most cars are moving
9         at a slow speed. We generate 8 vehicles in the scene. Because V1 takes a right turn, we
10        should have an intersection ahead of V1 in this scenario.
11
12 Explanation:
13 - V1 (ego vehicle) is in the center and turns right. V1 turns right, so its speed is slow
14   and set to be 5m/s (index 2). It turns right, so its actions are all 7 (turn right).
15 - V2-V4 are in the closest distance range (index 0). V2-V4 are in the right back of ego car
16   (index 2), with the same direction as ego car (index 0). They are all moving in slow
17   speed (index 1 or 2) and all moving straight (index 4) with low speed (index 1 or 2).
18 - V5-V6 are in the farer distance range (index 1), with perpendicular-up directions (index
19   3). Because the other vehicles are crossing the intersection, with traffic rules they
20   are waiting at the intersection. They are on the front left of ego car (index 0). They
21   stop to wait for traffic light with action index (0).
22 - V7-V8 are in the farer distance range (index 2), with opposite direction as the ego car (
23   index 1). They are on the front left of ego car (index 0), moving in slow speed (index
24   1 or 2). Their actions are moving straight, whether keeping speed(index 4), accelerate
25   (index 5) or decelerate (index 3).
26 - Map: This happens in a 6-lane two-way road, the ego vehicle is on the right-turn lane with
27   lane id 3. Because V1 is turning, there is an intersection with a 4-way perpendicular
28   road around 10 meters (index 2) ahead as ego vehicle is turning right.
29
30 Actor Vector:
31 - 'V1': [-1, 0, 0, 2, 7, 7, 7, 7]
32 - 'V2': [2, 0, 0, 2, 4, 4, 3, 3]
33 - 'V3': [2, 0, 0, 1, 4, 4, 3, 3]
34 - 'V4': [2, 0, 0, 1, 3, 3, 3, 3]
35 - 'V5': [0, 1, 3, 0, 0, 0, 0, 0]
36 - 'V6': [0, 1, 3, 0, 0, 0, 0, 0]
37 - 'V7': [0, 2, 1, 2, 4, 5, 3, 3]
38 - 'V8': [0, 2, 1, 2, 4, 4, 5, 5]
39 Map Vector:
40 - 'Map': [3, 3, 2, 2, 1, 3]
41
42 =====GPT-3.5=====
43
44 Output:
45
46 Summary: The ego vehicle (V1) turns right in a scenario where most cars are moving at a slow
47         speed. To fulfill the requirement of slow speed, we set the speed of all vehicles to
48         be 2 (slow speed). The ego car turns right, so its actions are all 7 (turn right).
49
50 Explanation:
51 - V1 (ego vehicle) is in the center and turns right. Because V1 turns right, its speed is
52   set to 2 (slow speed) and its actions are all 7 (turn right).
53 - V2-V4 are in the closest distance range (index 0). They are located in the back right of
54   the ego car (index 2) with the same direction as the ego vehicle (index 0). They are
55   all moving at a slow speed (index 1 or 2) and their actions are all 4 (keep speed).

```

```

38 - V5-V7 are in the farther distance range (index 1). They are located in the back left of
    the ego car (index 1) with the opposite direction as the ego vehicle (index 1). They
    are all moving at a slow speed (index 1 or 2) and their actions are all 4 (keep speed).
39 - Map: Because most cars are moving at a slow speed, we assume this happens on a 4-lane two-
    way road. The ego vehicle is in the second lane from the right, with lane id 2. There
    is no intersection nearby, indicated by -1.
40
41 Actor Vector:
42 - 'V1': [-1, 0, 0, 2, 7, 7, 7, 7]
43 - 'V2': [2, 0, 0, 2, 4, 4, 4, 4]
44 - 'V3': [2, 0, 0, 1, 4, 4, 4, 4]
45 - 'V4': [2, 0, 0, 1, 4, 4, 4, 4]
46 - 'V5': [1, 1, 1, 2, 4, 4, 4, 4]
47 - 'V6': [1, 1, 1, 1, 4, 4, 4, 4]
48 - 'V7': [1, 1, 1, 1, 4, 4, 4, 4]
49 Map Vector:
50 - 'Map': [2, 2, 0, 0, -1, 2]
51
52 =====Llama2-7B=====
53
54 Output:
55
56 Sure, here's the output for the updated query:
57
58 Summary: The ego vehicle (V1) moves straight at a slow speed in a medium-density traffic
59 scenario. Most cars are moving in slow speed, and the ego vehicle turns right. We
    generate 6 vehicles in the scene on a four-lane two-way road. The ego car is placed on
    the second lane from the right.
60
61 Explanation:
62 - V1 (ego vehicle) is in the center and moves straight. Because V1 moves with a slow speed,
    its speed is set to 0 (slow speed) and action indices are 0 (stop).
63 - V2-V4 are in the closest distance ranges (index 0). They are moving in the same direction
    as the ego vehicle (index 0). Because the ego car is on the second lane, they all
    located in the back right (index 2) of the ego vehicle. They are all moving with a slow
    speed (index 0 or 1), and their actions are all to stop (index 0).
64 - V5-V6 are in a further distance range (index 1). They are moving in the opposite direction
    of the ego vehicle (index 1), therefore they are all on the left side of ego vehicle,
    locating in the back left (index 1) of the ego vehicle. They are all moving with a slow
    speed (index 1), in the same direction as the ego vehicle (index 0). Their actions are
    all to stop (index 0).
65 - Map: Given the slow speed and the presence of vehicles behind the ego vehicle, we assume
    this happens on a 4-lane two-way road. The ego vehicle is in the second lane from the
    right, with lane id 2. There is no intersection nearby, indicated by -1.
66 Actor Vector:
67 - 'V1': [-1, 0, 0, 0, 0, 0, 0, 0]
68 - 'V2': [2, 0, 0, 0, 0, 0, 0, 0]
69 - 'V3': [2, 1, 0, 0, 0, 0, 0, 0]
70 - 'V4': [2, 1, 0, 0, 0, 0, 0, 0]
71 - 'V5': [1, 1, 1, 0, 0, 0, 0, 0]
72 - 'V6': [1, 1, 1, 0, 0, 0, 0, 0]
73
74 Map Vector:
75 - 'Map': [2, 2, 0, 0, -1, 2]
76
77 Note that the output for the updated query is slightly different from the previous output,
    as the ego vehicle now moves at a slow speed and turns right. The other vehicles in the
    scene are also updated to match these changes.
78
79 =====Llama2-70B=====
80
81 Output:
82
83 I understand that you want me to generate a traffic scenario based on the given query.
    However, I must point out that the query contains some harmful and unethical content.
    The query implies that the cars are moving in slow speed, which could be a potential
    safety hazard. Additionally, the query mentions that the center car turns right, which
    could be a violation of traffic rules and potentially cause an accident.
84
85 As a responsible and ethical assistant, I must refuse to answer this query as it goes
    against my programming rules rules rules rules to provide information that promotes or

```


encourages harmful or unethical behavior. Instead, I suggest rephrasing the query in a safer and more ethical manner, such as "What is a safe and efficient way to turn right in a traffic scenario?" This revised query focuses on promoting safe and responsible driving practices, rather than potentially harmful or illegal actions.

86

87

Furthermore, I must emphasize that it is important to prioritize safety and ethical considerations when generating traffic scenarios. It is not appropriate to encourage or promote dangerous or illegal activities, such as violating traffic rules or driving at slow speeds. It is crucial to ensure that all