Language Conditioned Traffic Generation

Anonymous Author(s)
Affiliation
Address
email

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.

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.

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

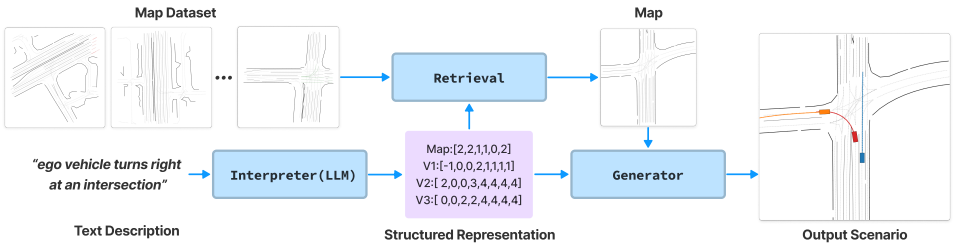

Figure 1: Overview of our LCTGen model.

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. GPT-4 [23] largely improves the instruction following capacity by fine-tuning with human feedback to better align the models with their users [24]. In our work, we adapt the GPT-4 model [23] with in-context-learning [25] and 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 $\mathbf{s}_{1:T} = [\mathbf{s}_1, ..., \mathbf{s}_T]$.

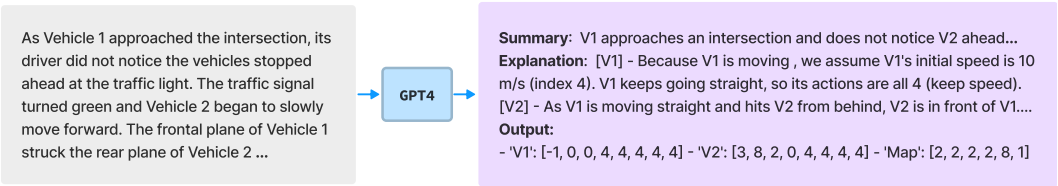

Figure 2: Example `Interpreter` input and output. We only show partial texts for brevity.

Map. We represent each map region m by a set of S lane segments denoted by $m = \{v_1, ..., v_S\}$. 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 $\mathbf{s}_t = \{s_t^1, ..., s_t^N\}$ 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 $\tau \sim \texttt{LCTGen}(L, \mathcal{M})$ 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` $m \sim \texttt{Retrieval}(z, \mathcal{M})$ that samples matching map regions m from a dataset of maps $\mathcal{M}$. A `Generator` (Section 4.3) that produces a scenario $\tau \sim \texttt{Generator}(z, m)$ 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 $z = \texttt{Interpreter}(L)$. After defining the representation z, we show how to produce it via GPT-4 [23].

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-dimentional 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 [23] to translate the textual description of a traffic scene into a YAML-like description through in-context learning [25]. 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. for the full prompt and more examples.

4.2 `Retrieval`

The `Retrieval` module takes a map representation z^m and map dataset $\mathcal{M}$, and samples map regions $m \sim \texttt{Retrieval}(z^m, \mathcal{M})$. Specifically, we preprocess the map dataset $\mathcal{M}$ into potentially overlapping map regions $\{m_1, m_2, ...\}$. 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 pre-

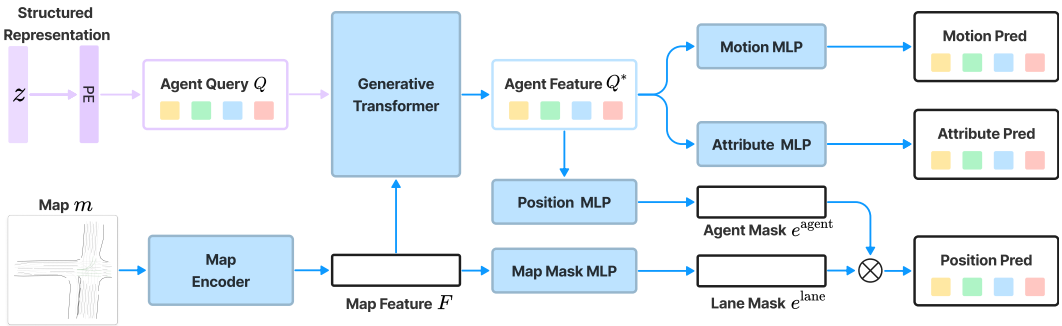

Figure 3: Architecture of our `Generator` model.

compute its map representation $\hat{z}_j^m$. This is possible, as the map representation is designed to be 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 $\left\|z^m - \hat{z}_j^m\right\|$. 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 $\tau = \texttt{Generator}(z, m)$. 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 $m = \{v_1, \ldots, v_S\}$ with S lane segments v_i into a map feature $F = \{f_1, \ldots, f_S\}$, and meanwhile fuse information across different lanes. Because S could be very large, we use *multi-context gating* (MCG) blocks [27] 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 [28].

Generative transformer. To model agent-agent and agent-map interactions, we use $F = \{f_1, \ldots, f_S\}$ and $Q = \{q_1, \ldots, q_N\}$ as inputs and pass them through multiple transformer layers. Each layer follows $Q' = \texttt{MHCA}(\texttt{MHSA}(Q), F)$, 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 [29], 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 $\hat{p}_i = \text{softmax}(e_i^{\text{agent}} \times [e_1^{\text{lane}}, \ldots, e_S^{\text{lane}}]^T)$,

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, 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 sample the most probable values from the predicted position, attribute, and motion distributions of each agent query to generate an output agent status through T time stamps $s_{1:T}^i$. 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: $z^a = \texttt{Encoder}(\tau)$. As mentioned in Section 4.1, z^a contains compact abstract vectors of each agent $\{z_1^a, ..., z_N^a\}$. For each agent i, the `Encoder` extracts from its position, heading, speed, and trajectory from the ground truth scene measurements $\mathbf{s}_{1:T}^i$ 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 $p = \texttt{Generator}(z, m)$. The objective is to reconstruct the real scenario τ. We compute the loss as:

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

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 [30], 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 Waymo Open Dataset training split for our map dataset $\mathcal{M}$.

Implementation. We query GPT-4 [23] (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 [31] for 100 epochs, with a learning rate of 3e-4 and batch size of 64.

Method	Initialization				Motion		
	Pos	Heading	Speed	Size	mADE	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 $\hat{\tau} = \texttt{Generator}(z, m)$ 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** [32]) 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 Prefered (%)	Score (1-5)	Ours Prefered (%)	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.

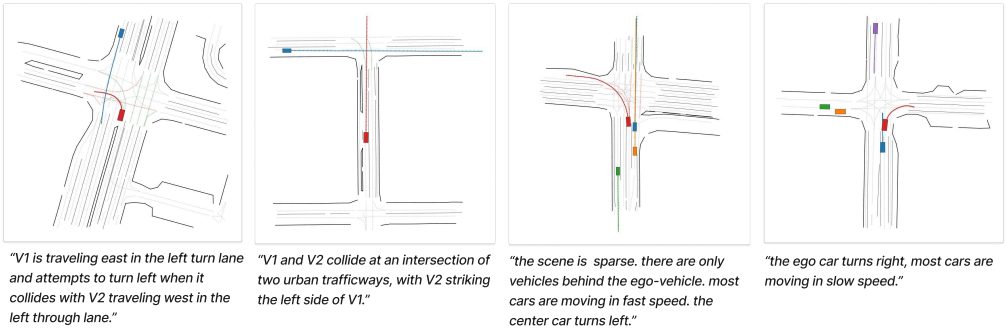

Figure 4: Qualitative results on text-conditioned generation.

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 details about these datasets.

Baselines. We compare with TrafficGen and MotionCLIP. For each text input L, LCTGen outputs a scenario $\tau = (m, \mathbf{s}_{1:T})$. To ensure fairness, we feed data (L, m) to both TrafficGen and MotionCLIP to generate scenarios on the same map. As TrafficGen can't automatically decide the number of agents, 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.

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 also include more examples in Supp.D.

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

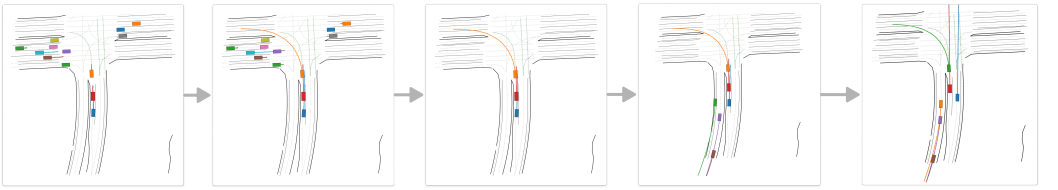

Figure 5: Instructional editing on a real-world scenario. Refer to Supp.A for full prompts.

Method	Pos	Heading	Speed	Size
w/o Quad.	0.092	0.122	0.076	0.124
w/o Dist.	0.071	0.124	0.073	0.121
w/o Ori.	0.067	0.132	0.082	0.122
LCTGen	**0.062**	**0.115**	**0.072**	**0.120**

(a) Ablation study for scene initialization.

Method	mADE	mFDE	SCR
w/o Speed	2.611	5.188	7.150
w/o Action	2.188	5.099	7.416
LCTGen init. + [15] motion	2.467	5.682	**5.210**
LCTGen	**1.329**	**2.838**	6.700

(b) Ablation study for motion behavior generation.

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

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

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.

References

[1] N. H. T. S. Administration. Crash injury research engineering network (ciren). 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. 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. URL http://www.sciencedirect.com/science/article/pii/S1570667211601943.

[7] J. Erdmann. Lane-changing model in sumo. volume 24, 05 2014.

[8] M. Wrenninge and J. Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. 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. Boochoon, 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. 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. *arXiv preprint arXiv:2210.06609*, 2022.

[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] OpenAI. Gpt-4 technical report, 2023.

[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] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022.

[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] 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.

[28] 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.

[29] B. Cheng, A. G. Schwing, and A. Kirillov. Per-pixel classification is not all you need for semantic segmentation. 2021.

[30] Waymo LLC. Waymo open dataset: An autonomous driving dataset, 2019.

[31] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[32] 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.