Towards Faithful and High-Fidelity Synthesis of Driving Scene Videos

Jiani Huang[1], Siyang Zhang[2], Ziyang Li[1], Ser-Nam Lim[2], and Mayur Naik[1]

[1] University of Pennsylvania,
[2] University of Central Florida

Abstract. Driving scene generation is a fundamental problem in autonomous driving. We propose TP2V, a graphic procedural approach for generating driving scene videos from text, with an emphasis on faithful and high-fidelity synthesis. TP2V combines the complementary capabilities of language models and game engines. Language models enable to bridge the gap between textual descriptions of driving scenes and programmatic inputs expected by game engines. Likewise, language models in conjunction with object detection and tracking enable semantic correctness checking of the generated video at the pixel level, thereby enhancing robustness against errors stemming from the intricacies of game engines. The resulting pipeline ensures correct physical dynamics, supports generating video of arbitrary length, and provides the generated 3D model for fine-grained manipulation. To effectively evaluate methods for driving scene video synthesis, we design a comprehensive benchmark comprising a dataset that encapsulates a wide array of traffic scenarios, together with metrics that aim to assess faithfulness and fidelity. Our evaluation shows that TP2V surpasses existing text-to-video generative models while also motivating the need for future advances.

Keywords: Text-to-Video Generation · Autonomous Driving · Benchmark · Metrics

1 Introduction

The development and training of self-driving models relies heavily on large-scale video data [1, 30, 51]. Obtaining and labeling such data for scenarios encompassing diverse traffic environments, weather conditions, driving behaviors, and other factors is a formidable and ongoing endeavor [4, 7, 15, 41].

In this paper, we take a fresh look at this fundamental challenge by addressing the problem of synthesizing driving scene videos from textual descriptions. Synthetically generated videos offer a unique advantage in covering scenarios that are challenging, if not impossible, to capture in the real world [10, 12, 24, 36, 38]. Furthermore, textual descriptions are effective at specifying a wide range of scenarios in an intuitive yet fine-grained manner [8, 20, 28, 39, 43, 47, 48].

Two indispensable properties in this setting are *faithfulness* and *fidelity*. Faithfulness characterizes the adherence of the videos to physical laws, common sense, and accuracy with respect to the provided textual descriptions. On

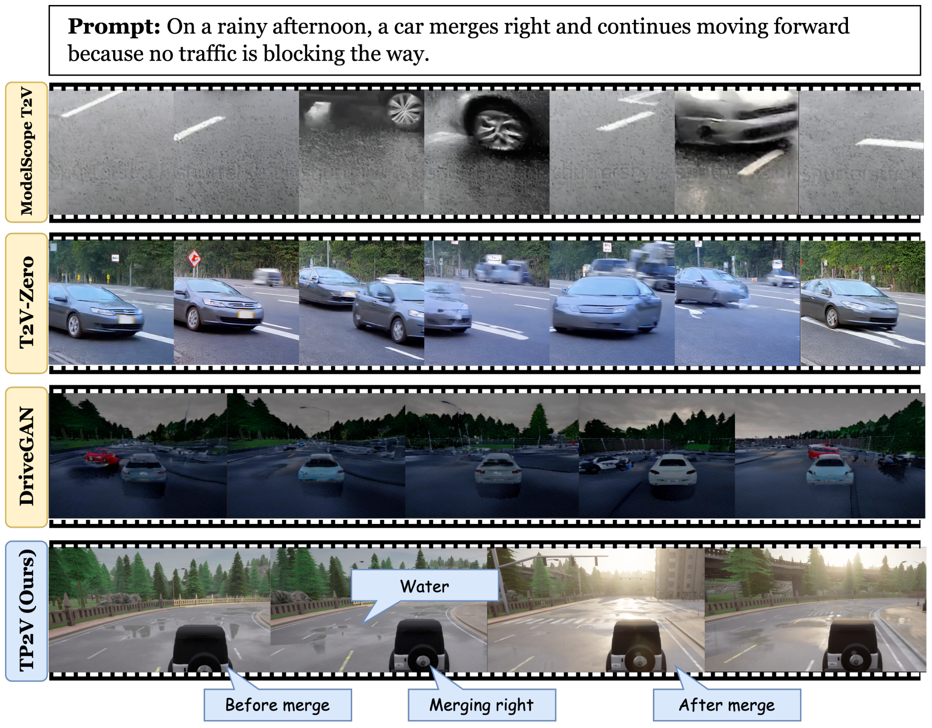

Prompt: On a rainy afternoon, a car merges right and continues moving forward because no traffic is blocking the way.
ModelScope T2V
T2V-Zero
DriveGAN
TP2V (Ours)
Water
Before merge
Merging right
After merge

Fig. 1: Comparison of videos generated by existing methods and ours. Diffusion- and GAN-based models can generate semantically incorrect or temporally inconsistent videos. TP2V overcomes these challenges using a graphic procedural pipeline depicted in Fig. 2, wherein (1) a language model generates a *scene program* from the text prompt, (2) a scene sampler generates a detailed scene configuration, (3) a game engine renders the video, and (4) a faithfulness checker assesses the final video quality.

the other hand, fidelity refers to the photo-realism, spatial, and temporal consistency of the videos. Training a robust downstream model necessitates a dataset satisfying both these properties.

Recent advances in general-purpose text-to-video models herald a promising direction in multimedia content generation [2, 16, 40]. However, when these models are applied to generate driving scene videos, they reveal significant shortcomings in terms of fidelity and faithfulness, as shown in Fig. 1. Furthermore, existing benchmarks fall short in evaluating these models adequately, particularly in the context of fidelity and faithfulness, signaling a gap in the assessment tools available for driving scene video synthesis.

In this work, we present a graphic procedural approach, Text-to-Procedural-to-Video (TP2V), for synthesizing driving scene videos from text. TP2V combines the strengths of language models for textual interpretation and game engines for advanced rendering and simulation. Our approach utilizes a scene programming language [12] as the intermediate representation to bridge the gap between textual descriptions and game engine scene configurations. We further extend our methodology to employ a foundation model backed faithfulness checking component, which ensures the quality of the synthesized video by checking common as well as unforeseen pitfalls that arise when utilizing game engines.

In a further contribution, we establish a comprehensive benchmark, Text-to-Drive-Video-Benchmark (T2DV-Bench), designed specifically to evaluate methods for driving scene video synthesis. We curate a dataset comprising 1.2K textual prompts that encapsulate a wide array of driving scenarios, and we propose a set of compound metrics aimed at assessing both faithfulness and fidelity. Among these metrics, a notable innovation is the *text-video semantic alignment score* (TVSAS), which leverages foundation models to provide an objective measure of alignment between the generated videos and their textual descriptions.

Our approach draws inspiration from recent works such as Infinigen [33], which similarly utilizes a graphics procedural language to generate photo-realistic images, as well as other works on 3D model generation [6,27,31,50]. Our paper takes a first step in this direction, albeit for videos in driving scenarios as well as facilitating text to video generation with graphics procedural generation.

Our evaluation demonstrates that TP2V outperforms existing diffusion- or GAN-based video generation models across multiple evaluation metrics. Most interestingly, compared to videos synthesized by end-to-end neural video generation models, our rendered video outperforms on fidelity by a wide margin as measured by Fréchet Video Distance (FVD) [42] against the real-life datasets BDD [20] and KITTI [15].

We summarize the contributions of this paper as follows:

1. we introduce TP2V, a graphic procedural approach towards generating faithful and high-fidelity driving scene videos from text;
2. we establish a comprehensive benchmark, T2DV-Bench, including a dataset and a compound metrics for evaluating faithfulness and fidelity;
3. for evaluating faithfulness, we propose a novel metrics called *text-video semantic alignment score* (TVSAS); and
4. we conduct extensive quantitative and qualitative evaluations of driving scene video generation approaches, showing superior performance of TP2V compared to existing baselines.

In summary, this paper represents a significant stride towards faithful and high-fidelity synthesis of driving scene videos from textual inputs, aiming to establish a common ground for evaluation and reveal insights for subsequent research.

2 Related Work

Image Generation and Text-to-Image Generation. Advancements in computer vision have led to significant progress in image generation. There are GAN-based models such as DM-GAN [53], and diffusion-based models such as DALL-E [34] and Stable Diffusion [35]. While most models are used to generate general images, however, a few targets specific domains. For instance, models have been developed to generate driving and traffic scenarios [13,18,22].

Text-to-Video Generation. Motivated by great successes in text-to-image generation, people extend the capability into video generation. This extension requires addressing additional challenges such as temporal coherence, dynamic scene understanding, and motion prediction. Example video generation models include Phenaki [44], which introduces a bidirectional masked transformer with a causal attention mechanism, allowing the generation of arbitrary-long videos from text prompt sequences. CogVideo [17] extends the ability of the text-to-image model CogView 2 [9] by finetuning it with a multi-frame-rate hierarchical training strategy to better align text and video clips. Tune-A-Video [49] proposes a strategy to use a reference video for one-shot finetuning by extending and finetuning a text-to-image Stable Diffusion model. More recently, Text2Video-Zero [23] leverages pretrained text-to-image diffusion models to do zero-shot learning and ModelScope Text-to-Video [45] uses UNets with 2D convolution blocks to capture spatial features and 1D convolution blocks to capture temporal features.

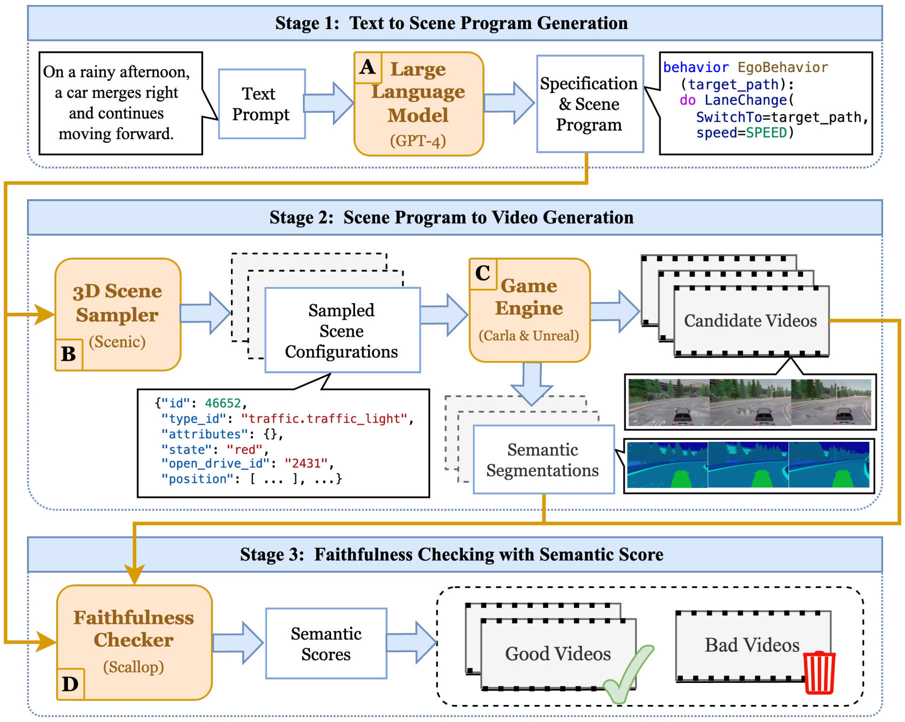
Synthesizing Dataset for Autonomous Driving. In the context of autonomous driving and scene understanding, datasets and benchmarks play a critical role in evaluating the performance of computer vision systems [1, 30, 51]. There are synthetic environments and corresponding toolchain for such generation. CARLA [10] and LGSVL [36] are open-sourced simulators for autonomous driving research, provides rich environments for generating diverse driving scenarios. Notably, AirSim [38] is also a simulator but used for drone controlling. While most simulators require a fine-grained specification of the scenarios, Scenic [12] offers a common ground to programmatically specify the scenario in a high-level manner, simplifying workflows of using the simulators. Our proposed approach, TP2V, takes this further by integrating large language models and faithfulness checkers, allowing to conveniently specify driving scenes using natural language.

3 Methodology

In this section, we describe TP2V, a graphics procedural approach that can generate driving scene videos from text. The first design decision we make is the use of a game engine for rendering and simulating the driving scenario. In this way, as opposed to end-to-end neural text-to-video models, our approach is inherently temporally consistent and adherent to physical laws. However, using game engines to create faithful and high-fidelity videos presents several challenges.

Challenge A: game engines require detailed scene configurations, including precise information of entity positions, states, and dynamics. Bridging the gap between textual descriptions and scene configurations remains an unresolved issue, especially as descriptions become more high-level and abstract.

Challenge B: textual descriptions often specify only a few key objects and properties, leaving the rest of the scene undefined. Without a model capable of completing these scenes, it is difficult to generate diverse scenarios, and the absence of detail can diminish the realism of the video.


Stage 1: Text to Scene Program Generation
Text Prompt
A Large Language Model (GPT-4)
Specification & Scene Program
Stage 2: Scene Program to Video Generation
3D Scene Sampler (Scenic) B
Sampled Scene Configurations
C Game Engine (Carla & Unreal)
Candidate Videos
Semantic Segmentations
Stage 3: Faithfulness Checking with Semantic Score
Faithfulness Checker (Scallop) D
Semantic Scores
Good Videos
Bad Videos

On a rainy afternoon, a car merges right and continues moving forward.

```
behavior EgoBehavior
  (target_path):
  do LaneChange(
    SwitchTo=target_path,
    speed=SPEED)
```

```
{"id": 46652,
 "type_id": "traffic.traffic_light",
 "attributes": {},
 "state": "red",
 "open_drive_id": "2431",
 "position": [ ... ], ...}
```

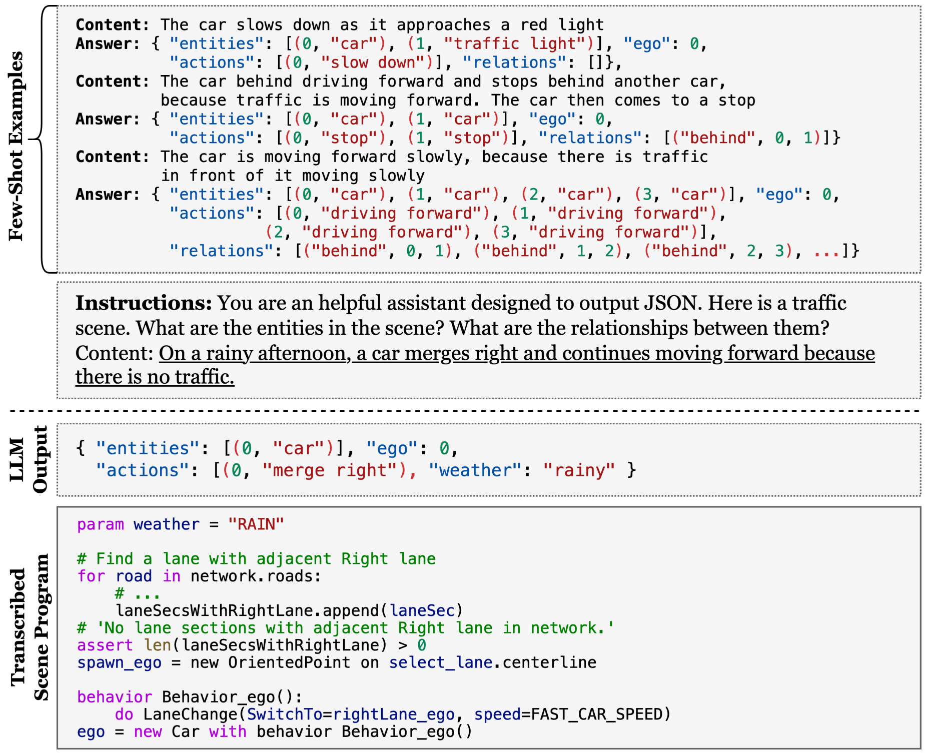
Fig. 2: An overview of our text-to-video pipeline.

Challenge C: the assets used in simulated environments come with their own set of limitations, such as inaccurate collision boxes, which can lead to unpredictable errors during simulation.

To overcome the aforementioned challenges, we propose a multi-stage approach, TP2V (Fig. 2). Our pipeline involves using 1) a large language model (LLM) for text understanding, 2) a scene programming language as an intermediate representation before driving scene configuration, 3) a game engine for simulation and rendering, and 4) a faithfulness checker for final quality of video. We now elaborate on each stage to explain how each challenge is solved, resulting in high-fidelity and faithful videos.

3.1 Stage 1: Text to Scene Program Generation

We address **Challenge A** by using LLMs in conjunction with a *scene programming language* to generate scene configurations for game engines. We employ Scenic [12] as our language for specifying complex driving scenarios. Programs in Scenic can declare entities (e.g. vehicles), their properties (e.g. color, type, orientation), and dynamic behaviors (e.g. speed, turns, lane switches). The runtime of Scenic can then execute the program and generate actual scene configurations as input to game engines. An important benefit of Scenic is that it allows scene


Few-Shot Examples
LLM Output
Transcribed Scene Program

```
Content: The car slows down as it approaches a red light
Answer: { "entities": [(0, "car"), (1, "traffic light")], "ego": 0,
          "actions": [(0, "slow down")], "relations": []},
Content: The car behind driving forward and stops behind another car,
         because traffic is moving forward. The car then comes to a stop
Answer: { "entities": [(0, "car"), (1, "car")], "ego": 0,
          "actions": [(0, "stop"), (1, "stop")], "relations": [("behind", 0, 1)]}
Content: The car is moving forward slowly, because there is traffic
         in front of it moving slowly
Answer: { "entities": [(0, "car"), (1, "car"), (2, "car"), (3, "car")], "ego": 0,
          "actions": [(0, "driving forward"), (1, "driving forward"),
                    (2, "driving forward"), (3, "driving forward")],
          "relations": [("behind", 0, 1), ("behind", 1, 2), ("behind", 2, 3), ...]}
```

Instructions: You are an helpful assistant designed to output JSON. Here is a traffic scene. What are the entities in the scene? What are the relationships between them?
Content: On a rainy afternoon, a car merges right and continues moving forward because there is no traffic.

```
{ "entities": [(0, "car")], "ego": 0,
  "actions": [(0, "merge right"), "weather": "rainy" }
```

```
param weather = "RAIN"

# Find a lane with adjacent Right lane
for road in network.roads:
    # ...
    laneSecsWithRightLane.append(laneSec)
# 'No lane sections with adjacent Right lane in network.'
assert len(laneSecsWithRightLane) > 0
spawn_ego = new OrientedPoint on select_lane.centerline

behavior Behavior_ego():
    do LaneChange(SwitchTo=rightLane_ego, speed=FAST_CAR_SPEED)
ego = new Car with behavior Behavior_ego()
```

Fig. 3: An illustration of our LLM prompting strategy to generate Scenic programs. We use manually crafted 3-shot examples so that the LLM can consistently generate structured JSON for subsequent transcribing to Scenic program. Here, the LLM output suggests that there is only one entity (numbered `0`) in the scene. The entity is going to take the action of "merge right", and the weather is rainy. In the transcribed program, the first line configures the environment to have a rainy weather. It then seeks a non-rightmost lane, so that a car can perform the desired action "merge right". Lastly, it spawns a car on the selected lane with the `LaneChange` behavior.

specifications to be incomplete, as the probabilistic sampler underlying Scenic can randomly sample unspecified properties. As such, we have reduced our scene configuration generation problem into a Scenic program synthesis problem.

In order to synthesize a Scenic program from text, we invoke LLM with few-shot examples and instruction prompts, as illustrated in Fig. 3. We note that the Scenic program is not the direct output expected from LLM, since although LLMs are shown to be good at programming in general-purpose languages [3], they do not perform as well at synthesizing in low-resource domain-specific languages (DSLs) like Scenic. Therefore, we introduce another layer of abstraction which we call "scene specifications" that can be specified in the JSON format. We have crafted a transcriber to take JSON "scene specifications" and produce the Scenic program. The details of the scene specification format and the transcriber are included in the Appendix. In Fig. 3, we illustrate a Scenic program specifying configurations of lanes, the ego car, and the change-lane behavior of the car. The program is indeed faithful to the prompt.

3.2 Stage 2: Scene Program to Video Generation

The Scenic engine is *probabilistic* [12], meaning that it is capable of sampling multiple scene configurations from executing a single program[3]. This addresses **Challenge B**, as diverse scenarios will be generated by Scenic. Now, each scene configuration is passed to the game engine for simulation and rendering. In TP2V, we employ CARLA [10] as the underlying simulation environment, which is based on Unreal Engine 4 (UE4) [11], well-known for its graphics capability to generate photo-realistic images. We note that with a game engine such as UE4, we can not only obtain the resulting video but also information such as depth buffer and semantic segmentation masks. These are important information that can be used in the subsequent stage for precise faithfulness checking.

3.3 Stage 3: Faithfulness Checking with Semantic Score

While Stages 1 and 2 combined have already closed the gap between text and driving scene videos, we observe many game engine failures that could result in bad videos. Particularly, the failure could come from penetrating objects, clipped camera viewport, and jittering physics engine. These failures are not only hard to fix but also hard to detect, since no explicit errors can be thrown. Therefore, we introduce Stage 3 for post-hoc filtering of the generated end-videos: we discard "bad videos" whose reported semantic scores are low[4], addressing **Challenge C**. Our semantic score is computed by a faithfulness checker written in Scallop, a probabilistic and relational programming language [25,26], and invokes the CLIP model [32] on checks related to entities' appearance in the output video. At a high level, the score comprises the result of checking the following information:

1. every mentioned entity in the scene specification appears in the video;
2. certain spatial relations mentioned in the scene specification hold; and
3. certain actions mentioned in the scene specification are performed.

For instance, when the sampled scene contains a clipped viewport that occludes the target vehicle, the semantic segmentation mask of the vehicle is applied to the rendered video and passed to CLIP to check if the object is indeed a vehicle. Since the vehicle is occluded, our aggregated semantic score will be low due to CLIP returning a low score of the object being a vehicle. Notably, Scallop is a probabilistic and relational language, making it a suitable solver for computing semantic score with relational specifications, scene segmentation, and probabilities obtained from CLIP. More details of the implementation of semantic score computation are provided in the Appendix.

[3] In practice, we set the number of scene configurations we generate from a single Scenic program to be 3.

[4] We declare failure if no good videos remain. While we did not observe this case in practice, an alternative would be jump to Stage 2 to re-sample scene configurations.

The car accelerates as it approaches the intersection and passes through the zebra line, while the traffic light is red.	The car suddenly swerves to the left, crossing into the opposing lane and begins driving on the wrong side of the road.
The car starts moving in reverse direction on the main road after missing the intended turn-off.	A car suddenly changes lanes without signaling, causing a rear-end collision with the car behind it.

Fig. 4: Four examples of Rare prompts of driving scenarios we provide as basis in our dataset. The described scenarios are rarely seen in real-life driving scene video datasets due to their irreplicability in real-life. Specific irregular situations are marked in blue.

4 Benchmark

In this section, we establish a new benchmark, T2DV-Bench, for evaluating text-to-video generation approaches. It is evaluation-only, meaning that the models evaluated on the benchmark are not limited to certain datasets for training. However, it is suggested that models use the BerkeleyDeepDrive (BDD) [52] and the KITTI [15] datasets for reference driving scene videos. We now discuss the included dataset and the metrics to evaluate performance.

4.1 Dataset

To encompass diverse driving scenarios, our dataset contains two categories of prompts: normal driving scenes (denoted Normal) and rare cases (denoted Rare). For Normal, we collect 200 prompts from captions manually crafted in the BDD-X [20] dataset, which serves as captions for the BDD videos. For Rare prompts, we manually crafted 10 prompts, of which 4 are shown as examples in Fig. 4. They cover rare driving scenarios that are dangerous to perform in real-life, and can be used to thoroughly test the capability of models. Using the given manually crafted prompts as basis, we further invoke GPT-4 in a few-shot manner to generate 1K prompts, comprising 700 Normal and 300 Rare prompts. Specifically for each call to GPT-4, we sample 3 example prompts from the corresponding category to generate 50 new prompts.

Due to the inherent ambiguity in natural language, we further manually label the ground truth specifications corresponding to these captions. Each programmatic specification consists of (a) the entities occurring in the generated video (b) the spatial relationships between the entities, and (c) the actions performed by each entity throughout the video. The specifications encompass 5 entities, 6 relations, 5 attributes, and 13 actions. Notably, rare events such as “driving in reverse direction” are represented using multiple inter-object relations, like “the ego car is moving *forward* on a lane which has its orientation *backward*”.

4.2 Evaluation Metrics

The primary goal of our benchmark is to evaluate the faithfulness and fidelity of the generated videos. For faithfulness, we aim to measure the adherence of the

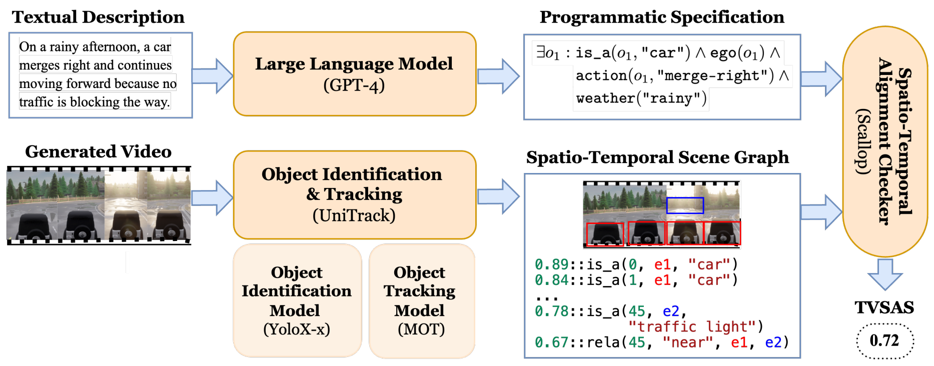

Textual Description
On a rainy afternoon, a car merges right and continues moving forward because no traffic is blocking the way.
Large Language Model (GPT-4)
Programmatic Specification
∃o1 : is_a(o1, "car") ∧ ego(o1) ∧ action(o1, "merge-right") ∧ weather("rainy")
Spatio-Temporal Alignment Checker (Scallop)
Generated Video
Object Identification & Tracking (UniTrack)
Object Identification Model (YoloX-x)
Object Tracking Model (MOT)
Spatio-Temporal Scene Graph
0.89::is_a(0, e1, "car")
0.84::is_a(1, e1, "car")
...
0.78::is_a(45, e2, "traffic light")
0.67::rela(45, "near", e1, e2)
TVSAS
0.72

Fig. 5: An illustration of the TVSAS calculation pipeline. A large language model takes in the textural description and converts it into a programmatic specification, a first-order logic expression describing the desired properties of the generated video. An object identification and tracking neural network then takes in the generated video, and yields a probabilistic spatio-temporal scene graph (STSG) which identifies the semantic details with confidence scores in the video. A spatio-temporal alignment checker verifies how likely does the specification hold given the STSG, and produces the alignment score reflecting the semantic faithfulness of the video to the text.

videos to physical laws, common sense, and accuracy with respect to the provided textual descriptions. We employ two quantitative metrics, namely CLIP-Sim, which utilizes the CLIP model [32], and Text-Video Semantic Alignment Score (TVSAS), a novel metric computed using foundation models and a neuro-symbolic programming language, Scallop [25,26]. For fidelity, on the other hand, we employ Fréchet Video Distance (FVD) [42] and CLIP-Aesthetic to evaluate its similarity with real-life video datasets. We now explain each metric in detail.

CLIP-Similarity (CLIP-Sim). CLIP-Sim is an embedding based similarity score that utilizes the multi-modal foundation model CLIP [32]. Given a video $\mathbf{v} = \{v_1, \ldots, v_n\}$ where v_i is the i-th frame, and a textual description $\mathbf{d}$, we use CLIP to obtain E_{v_i} and $E_{\mathbf{d}}$, the embeddings for frame v_i in the video and the text $\mathbf{d}$. The CLIP-Sim score is defined as

$$\text{CLIP-Sim}(\mathbf{v}, \mathbf{d}) = \frac{1}{n} \sum_{i=1}^{n} \max(100 * \cos(E_{v_i}, E_{\mathbf{d}}), 0) \tag{1}$$

Text-Video Semantic Alignment Score (TVSAS). Text-video semantic alignment score aims to evaluate the adherence of the generated video to the text[5]. It measures the fine-grained semantic alignment level between the given text and the generated video in object identity, attribute agreement, and temporal consistency. As opposed to CLIP-Sim which is based on black-box embeddings of texts

[5] While the computation of this metric shares commonalities with the faithfulness checker in TP2V (Sec. 3.3), this metric is independent from our approach and can end-to-end evaluate the alignment between any text-video pairs of driving scenes.

and videos, we compute TVSAS using a *neuro-symbolic* approach [19] that combines GPT-4 [3] and UniTrack [46] for faithful and interpretable analysis of text and videos. A visualization of TVSAS is portrayed in Fig. 5. The textual description is passed to GPT-4 to extract a relational programmatic specification. The video, on the other hand, is passed to the UniTrack framework [46] which consists of YoloX-x [14] for object identification and MOT [29] for object tracking. The result is a probabilistic spatio-temporal scene graph (STSG) [19] that contains object categories and their trajectories with confidence scores. We compare the scene graph against the programmatic specification through a Spatio-Temporal Alignment checker implemented in Scallop. This checker searches for a variable assignment that maps each variable defined in the programmatic specification to a concrete object instance that occurs in the STSG. The resulting TVSAS reflects the likelihood that the programmatic specification aligns with the STSG.

Fréchet Video Distance (FVD). The Fréchet Video Distance (FVD) [42] is an extension of the Fréchet Inception Distance (FID), a widely used metric for evaluating the quality of generated images in comparison to real images. FVD is designed to capture the temporal dynamics of videos, in addition to the visual quality and diversity of individual frames. In this work, we compute FVD between the generated set of videos and real videos from two datasets, BDD [52] and KITTI [15], yielding two scores, FVD-BDD and FVD-KITTI.

CLIP-Aesthetic Score. The CLIP-Aesthetic Score [37] is a metric designed to evaluate the aesthetic and stylistic qualities of a single image using the capabilities of CLIP. By evaluating the image against a set of predefined aesthetic criteria expressed in textual form, the CLIP-Aesthetic Score can provide insights into the aesthetic qualities of the image, indicating the fidelity. We augment the score to evaluate videos by computing the average of CLIP-Aesthetic score obtained on each frame of the video.

5 Evaluation

5.1 Experimental Setup

Implementation Details. Our TP2V pipeline consists of four components: (a) an LLM, (b) a 3D scene sampler, Scenic [12], (c) a game engine, CARLA [10], and (d) a faithfulness checker, Scallop [26]. For LLM, we use the `gpt-4-turbo-preview` model to transform input texts into the scene specifications. We utilize Scenic version 3.0.0 and an adapted version of CARLA 0.9.13 to synthesize world configurations, run simulations, and render videos.

Baselines. As the baselines, we choose two open-source general-purpose text-to-video models, Text2Video-Zero [23] and ModelScope Text-to-Video [45]. Both models accept textual conditions as input and produce videos as output, utilizing the Stable Diffusion model with a U-Net architecture as their backbone. Text2Video-Zero enhances the diffusion model by augmenting its encoding space

Model	**FVD-BDD** ↓	**FVD-KITTI** ↓	**Clip-Aesthetic** ↑
Text2Video-Zero	2340.24	2523.68	4.78
ModelScope Text-to-Video	3607.11	3806.16	3.62
TP2V (Ours)	**2047.37**	**2427.23**	**5.08**
(Videos in BDD)	704.62	–	5.13
(Videos in KITTI)	–	715.18	5.25

Table 1: Fidelity evaluation of TP2V against baselines. We use FVD (BDD), FVD (KITTI), and Clip-Aesthetic Score as fidelity metrics. The experiments show that TP2V outperforms end-to-end neural methods by a large margin. We also include, for reference, the empirical upper bound scores obtained by evaluating the metrics on the videos in BDD and KITTI. Symbols ↑ and ↓ denotes larger-the-better and lower-the-better, respectively. Note that there is still a huge gap between the best performing approach (2047/2427) and the empirical upper bound (704/715) on the two FVD scores, suggesting a huge space for improvement.

with motion dynamics. On the other hand, ModelScope Text-to-Video introduces spatio-temporal blocks into the framework, enabling the generation of videos by better capturing both spatial and temporal relationships. The diffusion model `damo-vilab/text-to-video-ms-1.7b` is the backbone for ModelScope Text-to-Video, and `runwayml/stable-diffusion-v1-5` is the one for Text2Video-Zero. At inference time, each text-to-video model generates two videos given each textual description.

Evaluation Metrics. We evaluate the generated video quality on the 4 metrics as described in Sec. 4.2. Specifically, `gpt-4-turbo-preview` [3], UniTrack [46], YoloX-x [14], and MOT [29] are used in the computation of TVSAS. On the other hand, *CLIP-Aesthetic Score* and *CLIP-Sim* are measured using CLIP ViT-L/14. All experiments are conducted on a machine with 2 24-core Intel Xeon CPUs, 8 Nvidia A100 GPUs, and 1.58TB RAM.

5.2 Quantitative Results

Fidelity. We quantitatively evaluate fidelity using the Fréchet Video Distance (FVD) and CLIP-Aesthetic metrics, as presented in Table 6. TP2V significantly outperforms the evaluated diffusion-based models by a considerable margin, suggesting that videos generated by TP2V is closer in distance to real-life driving scene videos in BDD and KITTI. Remarkably, despite our model relying solely on simulated graphics, it surpasses the diffusion-based video generation baselines—trained on real-life footage—on FVD metrics. However, compared to empirical upper bounds, the FVD between two randomly sampled batches of video from BDD (and similarly for KITTI), there is still a huge space for improvement. On the other hand, under CLIP-Aesthetic, our approach also outshines the baselines in this domain. The close proximity of our aesthetic scores to those of actual real-life videos, as noted in the empirical upper bounds, underscores the high fidelity of our generated videos.

Model	CLIP-Sim ↑	TVSAS ↑
Text2Video-Zero	23.74	0.03
ModelScope Text-to-Video	22.65	0.30
TP2V (Ours)	**23.90**	**0.52**

Table 2: Faithfulness evaluation of TP2V against baselines. We use CLIP-Sim and TVSAS as the two faithfulness metrics.

Faithfulness. As shown in Table 5, for CLIP-Sim, TP2V obtains marginal improvement over the baselines. TVSAS offers a more nuanced and interpretable assessment. It is worth noting that Text2Video-Zero's scores on TVSAS are notably low, indicative of significant semantic discrepancies. This is further corroborated by manual inspections revealing that temporally inconsistent videos generated by baseline models led to the UniTracker algorithm detecting flickering and distorted objects as multiple entities. Such inconsistencies negatively impact TVSAS scores for these baselines. Conversely, our approach benefits from the utilization of a game engine for video generation, enabling UniTracker to recognize consistent objects and relationships throughout the video sequences. This consistency substantially boosts our TVSAS scores, reinforcing the semantic faithfulness of our generated videos.

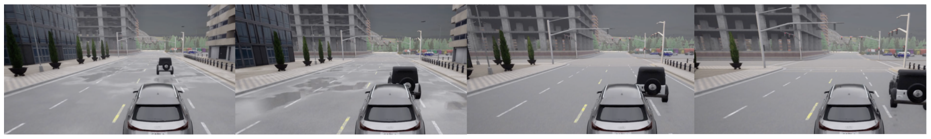
5.3 Qualitative Results

In our qualitative analysis, we manually compare TP2V against baselines, focusing on the visual integrity and fidelity illustrated in Fig. 8. The use of a game engine for rendering in our approach inherently ensures a high degree of visual temporal consistency, a crucial advantage not shared by the baseline models. These baselines are frequently marred by significant visual artifacts, such as unexpected image distortions and watermarks, and suffer from jittering alongside spatial and temporal distortions, all of which considerably degrade the video quality, as depicted in Fig. 8b and 8c.

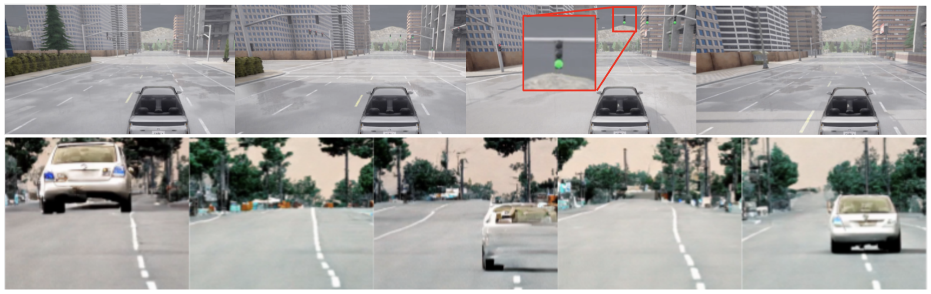
Moreover, our method demonstrates superior adherence to textual descriptions, significantly enhancing the faithfulness of the synthesized videos. For example, our approach successfully synthesized a video depicting a crash scenario as specified in the textual description (Fig. 8a)—a feat that poses ethical and practical challenges in real-life video collection. Furthermore, in Fig. 8b, our generated video includes details such as a green traffic light, emphasizing the precision of our method. In contrast, baseline methods like Text2Video-Zero struggle to include the green traffic light. This comparative analysis underscores our method's faithfulness to corresponding textual descriptions.

5.4 Ablation Studies

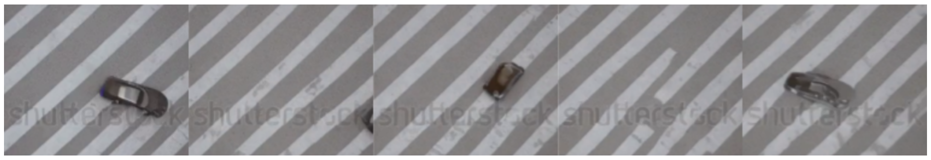
LLM and prompting strategies. In this study, we scrutinize the efficacy of LLMs and prompting strategies in generating programmatic specifications from text. As shown in Table 3, using GPT-4 with 3-shot significantly outperforms other



(a) The video generated by TP2V given the caption "the car continues to drive despite another vehicle ahead, resulting in a rear-end collision."



(b) A pair of video snippets generated by TP2V (↑) and Text2Video-Zero (↓), with the prompt "the car is driving forward because the light is green". TP2V successfully generate a scene with the green light while it is not faithfully represented in the video generated by Text2Video-Zero.



(c) A video snippet generated by ModelScope Text-to-Video, with the prompt "the car drives down a street. because the street is clear ahead". Neither the vehicle nor the background moves smoothly, and the video jitters. Also notice the unexpected watermark in the generated video.

Fig. 6: Qualitative analysis of videos generated by various models.

configurations. Note that the distribution of facts varies across categories, entities, actions, and relations. The calculation of average accuracy takes into account all facts collectively. The experimental result suggests that in-context learning is crucial to our approach.

Faithfulness Checking with Semantic Score. In TP2V, the semantic score computed in Stage 3 serves as a crucial filter to selectively exclude generated videos that meet the scene specification with low probability. Upon manually reviewing 30 generated videos before and after implementing video faithfulness check, we observed a 10% improvement in semantic consistency. The main factors leading to the exclusion of certain video segments were instances of object occlusion and situations where objects exceeding culling distance were not rendered.

5.5 Limitations and Future Work

While our approach demonstrates superior quantitative and qualitative results in the realm of video generation, it is not without its limitations:

1. **Austerity of Simulated Scenes**: the simulated driving scenes are still relatively barren, as the textual descriptions guiding these simulations typically

Models	**GPT-3** (0-shot)	**GPT-3** (3-shot)	**GPT-4** (0-shot)	**GPT-4** (3-shot)
Entity Acc.	60.00	85.09	58.28	**73.37**
Action Acc.	63.42	71.30	76.85	**86.57**
Relation Acc.	0.00	17.24	0.0	**67.85**
Average Acc.	56.61	73.86	60.36	**77.54**

Table 3: Quantitative results of ablation study on text-to-programmatic specification conversion. The evaluation metric (Acc.) is accuracy, measured as the ratio of correctly identified categories to total categories evaluated.

involve fewer than five entities, resulting in scenarios that are considerably less complex than real-life situations, such as traffic jams.

2. **Dependency on Asset Library**: our approach relies on the CARLA simulation engine's asset library. For example, if a prompt describes a "pink sedan" that is not available in the asset library, our system is inherently unable to generate a video that fully aligns with the prompt's specifications.
3. **Gap in Photo-realistic Quality**: despite the superiority of TP2V over diffusion-based video generation baselines, there is still a significant gap between our FVD and the empirical performance upper bound (Table 6). This gap suggests that our approach has yet to achieve the level of detail and environmental authenticity of real-life videos. Contributing factors include the granularity of the provided assets, the lack of real-world road intricacies, and limitations in the rendering capabilities of UE4. Nevertheless, we believe that TP2V provides a good starting point for future research to improve its photo-realism along with temporal consistency and faithfulness.

In the future, we aim to tackle these three limitations by 1) improving the text-to-scene-configuration pipeline for natural simulation of complex driving scenarios, 2) incorporating procedural asset and scene generation tools such as InfiniGen [33], and 3) develop novel simulation-to-real (Sim2Real) pipelines that can augment rendered scenes with extra photo-realism.

6 Conclusion

We propose TP2V, a fully automatic pipeline that generates driving scene videos from natural text. It follows a graphic procedural approach that relies on game engines for faithful and high-fidelity synthesis, and augments them with language models to generate programmatic scene specifications from text descriptions and to enhance robustness against errors. The resulting pipeline ensures correct physical dynamics, supports generating video of arbitrary length, and provides the generated 3D model for fine-grained manipulation. Benchmarking demonstrates TP2V's ability in temporal consistency, semantic integrity, adherence to textual descriptions, and photo-realism.

References

1. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016) 1, 4
2. Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), https://openai.com/research/video-generation-models-as-world-simulators 2
3. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., et al.: Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712 (2023) 6, 10, 11
4. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR (2020) 1
5. Chen, H., Zhang, Y., Cun, X., Xia, M., Wang, X., Weng, C., Shan, Y.: Videocrafter2: Overcoming data limitations for high-quality video diffusion models (2024) 22
6. Chen, Y., Pan, Y., Li, Y., Yao, T., Mei, T.: Control3d: Towards controllable text-to-3d generation. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 1148–1156 (2023) 3
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) 1
8. Deruyttere, T., Vandenhende, S., Grujicic, D., Van Gool, L., Moens, M.F.: Talk2car: Taking control of your self-driving car. arXiv preprint arXiv:1909.10838 (2019) 1
9. Ding, M., Zheng, W., Hong, W., Tang, J.: Cogview2: Faster and better text-to-image generation via hierarchical transformers. Advances in Neural Information Processing Systems **35**, 16890–16902 (2022) 4
10. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on Robot Learning (CoRL). PMLR (2017) 1, 4, 7, 10
11. Epic Games: Unreal engine 4. https://www.unrealengine.com (2024) 7
12. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: a language for scenario specification and scene generation. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (2019) 1, 2, 4, 5, 7, 10
13. Gao, R., Chen, K., Xie, E., Hong, L., Li, Z., Yeung, D.Y., Xu, Q.: Magicdrive: Street view generation with diverse 3d geometry control (2024) 3
14. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021) 10, 11
15. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11) (2013) 1, 3, 8, 10
16. Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D.P., Poole, B., Norouzi, M., Fleet, D.J., et al.: Imagen video: High definition video generation with diffusion models. arXiv preprint arXiv:2210.02303 (2022) 2
17. Hong, W., Ding, M., Zheng, W., Liu, X., Tang, J.: Cogvideo: Large-scale pretraining for text-to-video generation via transformers. arXiv preprint arXiv:2205.15868 (2022) 4, 22

18. Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., Corrado, G.: Gaia-1: A generative world model for autonomous driving (2023) 3
19. Huang, J., Li, Z., Naik, M., Lim, S.N.: Laser: A neuro-symbolic framework for learning spatial-temporal scene graphs with weak supervision (2023) 10
20. Kim, J., Rohrbach, A., Darrell, T., Canny, J., Akata, Z.: Textual explanations for self-driving vehicles. In: ECCV (2018) 1, 3, 8
21. Kim, J., Rohrbach, A., Darrell, T., Canny, J.F., Akata, Z.: Textual explanations for self-driving vehicles. CoRR **abs/1807.11546** (2018), http://arxiv.org/abs/1807.11546 19
22. Kim, S.W., Philion, J., Torralba, A., Fidler, S.: Drivegan: Towards a controllable high-quality neural simulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5820–5829 (2021) 3
23. Levon, K., Andranik, M., Vahram, T., Roberto, H., Zhangyang, W., Shant, N., Humphrey, S.: Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In: ICCV (2023) 4, 10, 21
24. Li, X., Zhang, Y., Ye, X.: Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. arXiv preprint arXiv:2310.07771 (2023) 1
25. Li, Z., Huang, J., Liu, J., Zhu, F., Zhao, E., Dodds, W., Velingker, N., Alur, R., Naik, M.: Relational programming with foundation models. In: AAAI (2024) 7, 9
26. Li, Z., Huang, J., Naik, M.: Scallop: A language for neurosymbolic programming. Proceedings of the ACM on Programming Languages **7**(PLDI) (2023) 7, 9, 10, 19
27. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 300–309 (June 2023) 3
28. Malla, S., Choi, C., Dwivedi, I., Choi, J.H., Li, J.: Drama: Joint risk localization and captioning in driving. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (2023) 1
29. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016) 10, 11
30. Osiński, B., Jakubowski, A., Zięcina, P., Miłoś, P., Galias, C., Homoceanu, S., Michalewski, H.: Simulation-based reinforcement learning for real-world autonomous driving. In: ICRA (2020) 1, 4
31. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion (2022) 3
32. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021) 7, 9
33. Raistrick, A., Lipson, L., Ma, Z., Mei, L., Wang, M., Zuo, Y., Kayan, K., Wen, H., Han, B., Wang, Y., Newell, A., Law, H., Goyal, A., Yang, K., Deng, J.: Infinite photorealistic worlds using procedural generation (2023) 3, 14
34. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: ICML (2021) 3
35. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022) 3
36. Rong, G., Shin, B.H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al.: Lgsvl simulator: A high fidelity simulator for autonomous driving. In: IEEE 23rd International conference on intelligent transportation systems (ITSC) (2020) 1, 4

37. Schuhmann, C.: Aesthetic score predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor (2022) 10
38. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and Service Robotics: Results of the 11th International Conference (2018) 1, 4
39. Sima, C., Renz, K., Chitta, K., Chen, L., Zhang, H., Xie, C., Luo, P., Geiger, A., Li, H.: Drivelm: Driving with graph visual question answering. arXiv preprint arXiv:2312.14150 (2023) 1
40. Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al.: Make-a-video: Text-to-video generation without text-video data. arXiv preprint arXiv:2209.14792 (2022) 2
41. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020) 1
42. Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Fvd: A new metric for video generation (2019) 3, 9, 10
43. Vasudevan, A.B., Dai, D., Van Gool, L.: Object referring in videos with language and human gaze. In: CVPR (2018) 1
44. Villegas, R., Babaeizadeh, M., Kindermans, P.J., Moraldo, H., Zhang, H., Saffar, M.T., Castro, S., Kunze, J., Erhan, D.: Phenaki: Variable length video generation from open domain textual description. arXiv preprint arXiv:2210.02399 (2022) 4
45. Wang, J., Yuan, H., Chen, D., Zhang, Y., Wang, X., Zhang, S.: Modelscope text-to-video technical report. arXiv preprint arXiv:2308.06571 (2023) 4, 10, 21
46. Wang, Z., Zhao, H., Li, Y.L., Wang, S., Torr, P., Bertinetto, L.: Do different tracking tasks require different appearance models? Advances in Neural Information Processing Systems **34**, 726–738 (2021) 10, 11
47. Wu, D., Han, W., Wang, T., Dong, X., Zhang, X., Shen, J.: Referring multi-object tracking. In: CVPR (2023) 1
48. Wu, D., Han, W., Wang, T., Liu, Y., Zhang, X., Shen, J.: Language prompt for autonomous driving. arXiv preprint arXiv:2309.04379 (2023) 1
49. Wu, J.Z., Ge, Y., Wang, X., Lei, S.W., Gu, Y., Shi, Y., Hsu, W., Shan, Y., Qie, X., Shou, M.Z.: Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7623–7633 (2023) 4
50. Wu, J., Gao, X., Liu, X., Shen, Z., Zhao, C., Feng, H., Liu, J., Ding, E.: Hd-fusion: Detailed text-to-3d generation leveraging multiple noise estimation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 3202–3211 (January 2024) 3
51. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: CVPR (2017) 1, 4
52. Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 8, 10, 19
53. Zhu, M., Pan, P., Chen, W., Yang, Y.: Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 3

7 Appendix

Illegal Action	**Scenario**
Run red light	In an attempt to beat the red light, a vehicle accelerates but misjudges the timing and runs the red light.
Driving on the wrong side of the road	A driver, while trying to make a U-turn on a narrow road, miscalculates and ends up driving in the opposite lane towards oncoming cars.
Driving reverse on the road	A car mistakenly enters a freeway off-ramp and reverses back towards the highway.
Collision with another car	A vehicle attempts a left turn at an intersection without yielding to oncoming traffic, causing a head-on collision.
Failure to yield	At a T-junction, a vehicle doesn't yield to through traffic, attempting to merge without caution
Illegal Parking	A car is parked with its rear protruding into the road, causing a hazard for passing vehicles.
Tailgating	On an icy road, a vehicle tailgates another, disregarding the increased stopping distance required on slippery surfaces.
Ignoring Pedestrian Crossing	A driver, focused on finding a parking spot, overlooks pedestrians in the crossing zone and narrowly misses them.
Blocking the Crosswalk	A taxi stops to pick up passengers right on the crosswalk, blocking the path for those trying to cross the street.
Unsafe Lane Changes	During a sudden slow-down on the highway, a vehicle attempts to swerve around the traffic, changing lanes erratically without looking.
Using emergency lane for regular driving	To quickly reach an exit ramp, a driver illegally drives down the emergency lane, bypassing other cars.
Swerving and Erratic Braking	A driver swerves and brakes erratically in heavy rain to avoid what they mistakenly perceive as an obstacle in the road, causing panic among other road users.

Table 4: A list of illegal actions with one scenario corresponding to the action.

7.1 Benchmark Details

Dataset Collection Our dataset consist of two parts: one representing regular driving scenes and the other depicting illegal driving scenarios.

Regular Driving Scene Prompts Collection We leverage the Berkeley Deep Drive-X (BDD-X) dataset [21] for sourcing video prompts depicting regular driving scenes. The BDD-X collection is curated by selecting $6,970$ videos from the Berkeley Deep Drive (BDD) dataset [52], each manually annotated with descriptions of actions taken within the videos and their corresponding justifications. For our purposes, a regular driving scene prompt is formed by combining an action description with its justification. Through this approach, we have compiled a total of $26K$ prompts that represent daily driving scenarios.

Rare Case Prompt Generation We curated a dataset comprising $1,200$ text-based prompts, each detailing scenarios of illegal driving that are challenging, or even impossible, to gather in real-world settings. Each data point is structured as a tuple (a, s), where a represents an illegal action, and s describes a scenario illustrating that action, as shown in Table 4. The dataset encompasses 12 distinct illegal actions, with 100 prompts dedicated to each.

The dataset collection process is a collaborative work between human and large language model. Initially, we crafted four pairs of illegal actions and corresponding scenarios manually. Subsequently, we employed `GPT-4` to expand this foundation by generating an additional 100 varied illegal actions. Through a meticulous manual filtering process, we excluded actions that are unidentifiable through video analysis, such as "Driving without a License," and eliminated duplicates, which yields a refined list of 12 illegal actions. For each of these actions, we further engaged `GPT-4` to generate 100 specific scenarios, thus enriching our dataset accordingly.

Semantic Score Implementation We implement the semantic score aligning the programmatic specification and the generated video with Scallop [26]. A sample of the alignment code is presented in Figure 7, where we demonstrate the algorithm for aligning the extracted scene graphs from the generated video with the programmatic specification. It's important to note that this probabilistic program takes in distributions of facts and outputs distributions of feasible alignments. Consequently, the semantic score quantitatively represents the likelihood of the video's semantic alignment with the programmatic specification, encompassing the quality of recognized objects.

```
type VarID = String
type ObjID = u32

type VarAssign = Cons(VarID, ObjID, VarAssign) | Nil()

type va_to_str(bound VarAssign, String)
rel va_to_str(va, "") = case va is Nil()
rel va_to_str(va, $format("v({})␣|->␣o({}),␣{}", v, o, r))
    = case va is Cons(v, o, rva) and va_to_str(rva, r)

type all_var_assigns(assignment: VarAssign)
type get_obj_id(bound assignment: VarAssign, bound var_id:
     VarID, obj_id: ObjID)
rel get_obj_id(va, v, o) = case va is Cons(v, o, _)
rel get_obj_id(va, v, o) = case va is Cons(vp, _, rs) and
    vp != v and get_obj_id(rs, v, o)

type cate(obj: ObjID, cate_name: String)
type attr(obj: ObjID, attr_name: String, attr_value: String)
type rela(rela_name: String, o1: ObjID, o2: ObjID)

type va_sat_spec(assignment: VarAssign, spec: usize)
rel va_sat_spec(va, 0) = all_var_assigns(va)
rel va_sat_spec(va, n + 1) =
    spec_cate(n + 1, v, cate) and va_sat_spec(va, n) and
    get_obj_id(va, v, o) and cate(o, cate)
rel va_sat_spec(va, n + 1) =
    spec_attr(n + 1, v, attr, val) and va_sat_spec(va, n) and
    get_obj_id(va, v, o) and attr(o, attr, val)
rel va_sat_spec(va, n + 1) =
    spec_rela(n + 1, rela_name, v1, v2) and
    va_sat_spec(va, n) and get_obj_id(va, v1, o1)
    and get_obj_id(va, v2, o2) and rela(rela_name, o1, o2)

rel sat_va(va) = va_sat_spec(va, n) and spec_root(n)
rel exist_sat_va() = exists(va: sat_va(va))

type spec_root(eid: usize)
type spec_cate(eid: usize, vid: VarID, cate_type: String)
type spec_attr(eid: usize, vid: VarID, attr_type: String,
     attr_val: String)
type spec_rela(eid: usize, rela: String, sub: VarID,
     obj:VarID)

rel sat_va_str(vas) = sat_va(va) and va_to_str(va, vas)
```

Fig. 7: Semantic score implementation for frame-wise comparison between video and programmatic specification.

Model	CLIP-Sim ↑	TVSAS ↑
Text2Video-Zero	23.74	0.03
ModelScope Text-to-Video	22.65	0.30
CogVideo	22.30	0.20
VideoCrafter	23.55	0.11
TP2V (Ours)	**23.90**	**0.52**

Table 5: Faithfulness evaluation of TP2V against baselines. We use CLIP-Sim and TVSAS as the two faithfulness metrics.

Model	FVD-BDD ↓	FVD-KITTI ↓	Clip-Aesthetic ↑
Text2Video-Zero	2340.24 ± 41.79	2523.68 ± 76.14	4.78
ModelScope Text-to-Video	3607.11 ± 71.89	3806.16 ± 119.58	3.62
CogVideo	2211.44 ± 98.51	2635.02 ± 111.29	3.73
VideoCrafter	4650.11 ± 136.69	5059.01 ± 80.39	3.91
AnimateDiff	3036.60 ± 121.35	3573.59 ± 150.31	4.88
TP2V (Ours)	**2047.37 ± 55.06**	**2427.23 ± 49.60**	**5.08**
(Videos in BDD)	704.62 ± 51.56	–	5.13
(Videos in KITTI)	–	715.18 ± 41.36	5.25

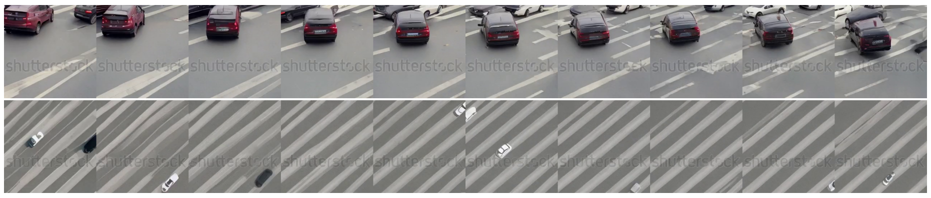
Table 6: Fidelity evaluation of TP2V against baselines. We use FVD (BDD), FVD (KITTI), and Clip-Aesthetic Score as fidelity metrics. For each FVD run, we randomly sample 100 videos from corresponding video sets. For each pair of video set, we run 10 times to record average FVD and standard deviation.

7.2 Experimental Details

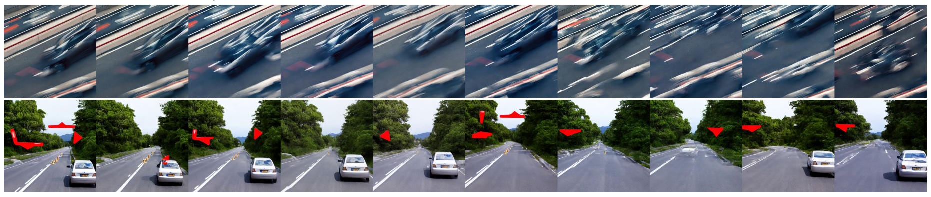
Baselines

Text2Video-Zero [23] is a zero-shot text-to-video generation derived by Stable Diffusion, a pretrained text-to-image generative model, making it suitable for the video domain. It enriches the latent codes of the generated frames with motion dynamics to keep the global scene and the background temporal consistent. It also has frame-level self-attention with a new cross-frame attention of each frame on the first frame, to preserve the context, appearance, and identity of the foreground object. In our experiment, we use the pretrained checkpoint of stable diffusion `runwayml/stable-diffusion-v1-5`.

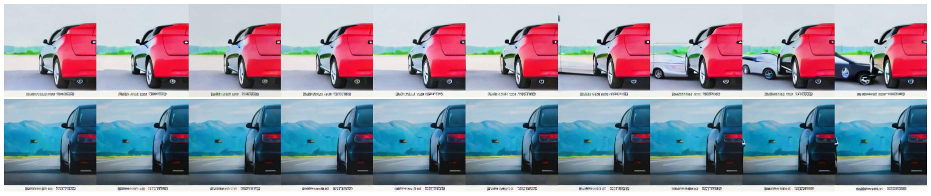
ModelScope Text-to-Video [45] is a text-to-video generative model that evolves from Stable Diffusion. It introduces spatio-temporal blocks to ensure smooth temporal motion consistency. The model could adapt to varying frame numbers during training and inference, rendering it suitable for both image-text and video-text datasets. ModelScope Text-to-Video contains 3 components: a VQGAN, a text encoder, and a UNet, The model demonstrates superior performance over state-of-the-art methods. In our experiment, we use the pretrained checkpoint: `damo-vilab/text-to-video-ms-1.7b`.



(a) The videos generated by ModelScope Text-to-Video given the captions "`The car moves forward then comes to a stop. because the light is red.`" and "`The car drives slowly down a highway. because there is heavy traffic.`", respectively.



(b) The videos generated by Text2Video-Zero given the captions "`The car is accelerating.Because traffic in front of it is moving normally.`" and "`The car turns right then moves along the road. because traffic is clear.`", respectively.



(c) The videos generated by CogVideo given the captions "`The car moves forward then comes to a stop. because no traffic is moving from behind.`" and "`The car is traveling quickly in the left lane. Because traffic in that lane is moving well.`", respectively.

Fig. 8: Examples of outputs by baseline models with unexpected artifacts.

CogVideo [17] is a large pretrained transformer for text-to-video generation in the general domain. It elegantly and efficiently finetunes a pretrained text-to-image generative model(CogView2) for text-to-image generation, avoiding the expensive full pretraining from scratch. Its multi-frame-rate hierarchical training can better align text-video pairs and significantly improve the generation accuracy, in particular for movements of complex semantics. In our experiments, we use the checkpoints: `CogVideo-Stage1, CogVideo-Stage2, CogView2-dsr`.

VideoCrafter [5] is a text-to-video generative model also extended from Stable Diffusion. Its training strategy takes advantage of both low-quality videos and synthesized high-quality videos. In our experiments, we use the checkpoint: `VideoCrafter/VideoCrafter2`.

Error Analysis

Object Flickering We noticed that in the videos genreated by ModelScope Text-to-Video and Text2Video-Zero, there exist some flickerings such as local distortions and sudden appearing or disappearing of objects. These artifacts strongly affect the quality of videos in terms of temporal consistency. Although these model are able to generated multiple frames in an identical scene, the transition among them is not smooth enough.

Frozen Frames On the other hand, videos generated by CogVideo have tendency to be stagnant, reducing the motion dynamic in videos. The main object in a video seems to have little visual motion even though we provide prompts containing obvious motion semantics in language.

7.3 Methodology Details

Programmatic Specification Interface Configuring a game engine to generate a specific scenario demands detailed settings, whereas a textual prompt typically offers only a broad overview. To bridge this gap between high-level descriptions and low-level programming requirements, we have developed an intuitive programmatic interface. This interface allows a large language model to efficiently translate high-level descriptions into detailed, executable programs, as illustrated in Table 7. The interface serves as a specification tool, offering programmatic abstractions that capture the intended semantics of the video to be generated. We demonstrate its functionality through three examples, as shown in Figure 3.

Transcriber Implementation Executing a specification program necessitates a transcriber that interprets the specification language and converts it into a scenic program. The resulting scenic program must not only ensure semantic accuracy but also exhibit efficiency and variability in its 3D scene configuration sampling procedure. To tackle these challenges, we have curated a library of templates and chosen constants tailored to various specification categories. For weather adjustments, our library includes several presets for parameters such as `cloudiness`, `precipitation`, `precipitation_deposits`, `fog_density`, `fog_distance`, `fog_falloff` combinations. In terms of time adjustments, we incorporate constants for `sun_altitude_angle` to enhance visibility in nighttime driving scenarios. For location specifications, we offer detailed templates for positions like `near`, `far`, `on` with numerical distributions. Likewise, for scenarios involving heavy traffic, different velocities and accelerations, and the orientation between objects, we provide comprehensive templates enriched with numerical distributions to facilitate precise and varied environmental setups.

Faithfulness Checker The faithfulness checker within our pipeline serves as an initial filter designed to eliminate low-quality videos that do not semantically align with the programmatic specification. Leveraging the semantic segmentation sensor integrated into the CARLA engine, we utilize ground truth object trajectories for our analyses. To ensure the 3D model quality, we opt for a more refined approach; instead of relying on ground truth object labels, we employ the CLIP similarity score for fine-grained object-level matching. The remainder of the pipeline adheres to the implementation strategy used for the semantic scoring process.

Specification Type	Specification Example
`weather(e: str)`	`rainy`, `sunny`, `snowy`, `foggy`, . . .
`time(e: str)`	`day`, `night`, `afternoon`, `8:00 pm`, . . .
`OID: int`	`0`, `1`, `2`, . . .
`name(i: OID, n: str)`	`(1, car)`, `(2, pedestrian)`, , `(3, bicycle)`, . . .
`loc(i: OID, n: str)`	`(4, intersection)`, `(5, road)`, `(6, lane)`, , `(7, curb)`, . . .

`attr(i: OID, attr: str,`
`arg: str)`
	`(1, color, black)`,
	`(1, brand, lincoln)`, . . .

`rela(rel: str,`
`i1: OID, i2: OID)`
	`(left, 1, 2)`,
	`(right, 2, 1)`,
	`(near, 1, 4)`, . . .

`velo(i: OID, arg: str)`	`(1, stop)`, `(2, fast)`, `(3, slow)`, `(2, turn left)`, . . .
`acce(i: OID, arg: str)`	`(1, break)`, `(2, faster)`, `(3, slower)`, . . .

`ori(ori: str,`
`i1: OID, i2: OID)`
	`(forward, 1, 6)`,
	`(backward, 2, 6)`, . . .

Table 7: Interface for spatio-temporal specification.