From Dashcam Videos to Driving Simulations: Stress Testing Automated Vehicles against Rare Events

Anonymous Author(s)
Affiliation
Address
email

Abstract

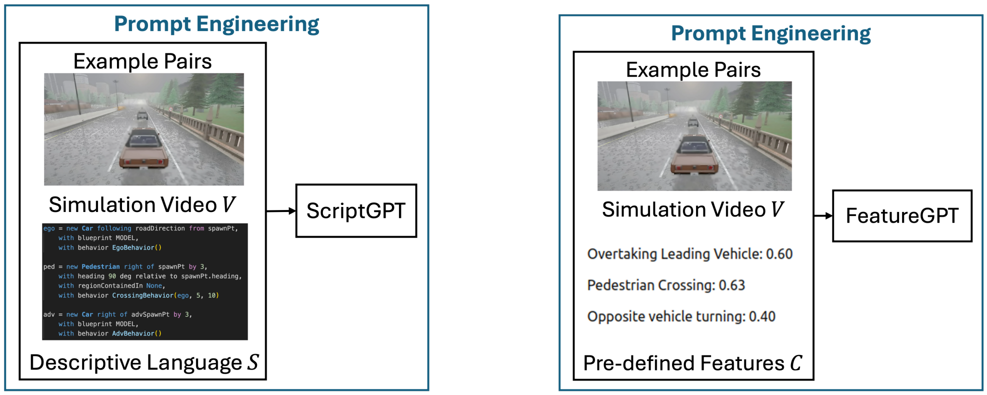
Testing Automated Driving Systems (ADS) in simulation with realistic driving scenarios is important for verifying their performance. However, converting real-world driving videos into simulation scenarios is a significant challenge due to the complexity of interpreting high-dimensional video data and the time-consuming nature of precise manual scenario reconstruction. In this work, we propose a novel framework that automates the conversion of real-world car crash videos into detailed simulation scenarios for ADS testing. Our approach leverages prompt-engineered Video Language Models (VLM) to transform dashcam footage into SCENIC scripts, which define the environment and driving behaviors in the CARLA simulator, and subsequently generate the simulation scenario. Additionally, we introduce a similarity metric that helps iteratively refine the generated scenario through feedback by comparing key features between the real and simulated videos. Our preliminary results demonstrate substantial time efficiency, finishing the real-to-sim conversion in minutes with full automation and no human intervention, while maintaining high fidelity to the original driving events.

1 Introduction

The rapid advancements in Automated Driving Systems (ADS) technology have created an urgent need for robust and realistic testing environments to assure reliability of ADS [1]. Real-world scenarios, such as crash videos and near-miss events, offer valuable insights into the diverse conditions ADS must navigate, making them an essential source for improving ADS testing. However, replicating these scenarios in real-world settings is both dangerous and impractical. Therefore, converting real-world driving videos into simulation scenarios is a necessary solution, but this process also poses several challenges: the high dimensional video data significantly limits the development of automated methods, while manually reconstructing these scenarios can take experts hours to complete. This time-consuming process underscores the need for a more efficient and automated approach.

In this paper, we propose a novel framework that automates the conversion of real-world driving videos into detailed simulation scenarios. Our approach utilizes prompt-engineered Video-Language Models to transform dashcam videos into SCENIC scripts, enabling the automatic generation of realistic simulations in CARLA. Furthermore, we introduce a similarity metric that iteratively refines the generated simulations by comparing key driving features between the real and simulated videos. The contributions of this work are fourfold: (1) an automated video-to-simulation pipeline that removes the need for manual scenario construction, (2) a similarity metric that bridge the gap between real and simulated scenarios, (3) an iterative feedback loop for scenario refinement using neural network-based feedback from Video-Language Models, and (4) a significant improvement in time efficiency, reducing scenario generation from hours to minutes while maintaining high fidelity to the original events.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.



(a) **ScriptGPT**: A Video Language Model derived from GPT-4o, developed through prompt engineering with paired examples of simulation videos and their corresponding descriptive SCENIC scripts.

(b) **FeatureGPT**: A Video Language Model derived from GPT-4o, developed through prompt engineering with paired examples of simulation videos and their corresponding pre-defined features.

Figure 1: Train *ScriptGPT* and *FeatureGPT* using Prompt Engineering

2 Related Work

Testing ADS in simulation environments has been the subject of extensive research [1]. Existing autonomous vehicle simulation platforms such as CARLA [2] and LGSVL [3] allow researchers to generate and manipulate driving scenarios in controlled environments. Efforts such as the SCENIC language [4] enables search-based testing(SBT) so that the generated scenario can be a seed for search based testing with respect to temporal logic requirements [5, 6], safe driving rules [7] and traffic laws [8]. However, accurately designing these scenarios to closely resemble real-world events remains a challenge.

Recently, efforts have shifted toward automatic real-to-simulation (real-to-sim) conversion approaches that use video data to guide scenario generation. For instance, Bai et al. [9] introduced a system that extracts key information from videos to create driving scenarios in simulation environments. In [10], the authors automatically generate driving scenarios from police crash reports. Similarly, [11] focus on learning realistic human behaviors in real-life scenarios and use learned models to improve simulations. NVIDIA's STRIVE [12] generates accident-prone driving scenarios by modifying 2D trajectories, but this method is based on controlled scenarios rather than real-world crash videos. Another approach, DEEPCRASHTEST [13], converts dashcam footage into crash tests by extracting 3D vehicle trajectories but lacks an iterative refinement process to improve simulation accuracy.

While these approaches represent meaningful progress, existing methods are either limited by a reliance on pre-defined trajectories or fail to incorporate iterative feedback to refine the generated scenarios. To address these gaps, we chose to use Video Language Models (VLM), as they allow for more flexible and scalable video-to-language translation, which can be enhanced through prompt engineering to generate detailed simulation scenarios.

3 Real-to-Sim Scenario Generation Framework

Given a real-life vehicle crash video (e.g., a dash camera recording), our objective is to generate a corresponding simulation scenario that accurately captures the core driving behaviors. Our framework consists of 4 components: (1) conversion of real-world video into SCENIC scripts, (2) generation of simulation videos from SCENIC scripts, (3) similarity analysis between the real and simulated videos, and (4) iterative refinement to ensure the simulated video's consistency with the original scenario.

3.1 Video-to-Text Generation

First, we convert the input video into a descriptive script. This is accomplished using a prompt-engineered version of the pre-trained GPT-4o model.

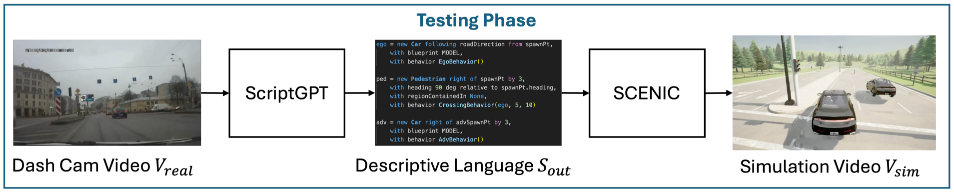


Figure 2: After prompt engineering, the dash cam video is fed into *ScriptGPT*, which synthesizes descriptive language in the SCENIC format. This SCENIC script can then be executed in CARLA to generate a corresponding testing scenario in simulation.

Prompt engineering involves designing and refining input prompts to guide large foundational models, such as GPT-4o [14], toward producing accurate and desired outputs. In this case, prompt engineering is especially effective for generating detailed scenario descriptions (e.g., SCENIC scripts) from real-world driving videos. During the prompt engineering process, we improve the pre-trained GPT-4o model by providing it with multiple "positive-example" pairs (V_i, S_i), where V_i is a simulation video generated in CARLA using the corresponding SCENIC script S_i that describes the scenario, as illustrated in Figure 1a. Through this process, the new Video-Language Model, referred to as *ScriptGPT*, learns to map key visual elements, such as weather, road conditions, and vehicle behaviors, into structured and accurate scenario description languages. After sufficient prompt engineering, ScriptGPT is capable of generating a SCENIC script S_{out} for real-world crash video V_{real}.

Although the initial prompt engineering process was conducted using simulation videos paired with their SCENIC scripts, this approach is extendable to real-world driving videos because the underlying visual and descriptive patterns (e.g., road layouts, traffic behaviors, and environmental factors) are consistent across both domains, allowing the model to effectively generalize its learned capabilities and accurately capture real-world scenarios.

3.2 Text-to-Video Generation

We convert the descriptive language S_{out} to simulation video V_{sim} using SCENIC [4, 15]. SCENIC is a programming language tool designed for specifying driving scenarios through environmental factors, vehicle behaviors, and road conditions. Once we have the SCENIC script S_{out} generated from the real crash video using the prompted-engineered model, we feed it into the SCENIC framework to synthesize a simulation scenario in CARLA, as shown in Figure 2. The script S_{out} serves as the textual representation of the scene, encoding environmental conditions (e.g., weather, traffic), vehicle dynamics, and road types. The output is a new simulation video V_{sim}, which visually represents the scenario described in S_{out}.

3.3 Similarity Check

Next, we perform a similarity check on the simulated scenario V_{sim} and the original V_{real}. Ideally, they should closely match, allowing us to seamlessly replace the ego vehicle with any ADS (e.g. Baidu's Apollo planner [16] and controller) for testing. However, due to the complexity of real-world scenarios, even with prompt engineering, discrepancies often arise, where the generated simulation may miss key features or introduce extra ones.

To address this, we introduce a similarity metric to ensure that the generated video captures the most important features from the original video. We predefine a set of crucial feature categories, such as the most critical and frequently encountered driving behaviors and environmental conditions, to examine the original crash video V_{real} with the generated simulation V_{sim}. Then, we use another prompt-engineered transformer model, *FeatureGPT* (depicted in Figure 1b), to output a predicted probability (from 0 to 1) for each predefined feature category for a given video. Next, the similarity score, $Sim(V_{real}, V_{sim})$, is calculated as a vector of differences between the predicted probabilities across the predefined categories:

$$Sim(V_{real}, V_{sim}) = [C_{real_1} - C_{sim_1}, C_{real_2} - C_{sim_2}, \ldots, C_{real_n} - C_{sim_n}],$$

where C_{real_i} and C_{sim_i} represent the predicted probabilities of V_{real} and V_{sim} for feature i.

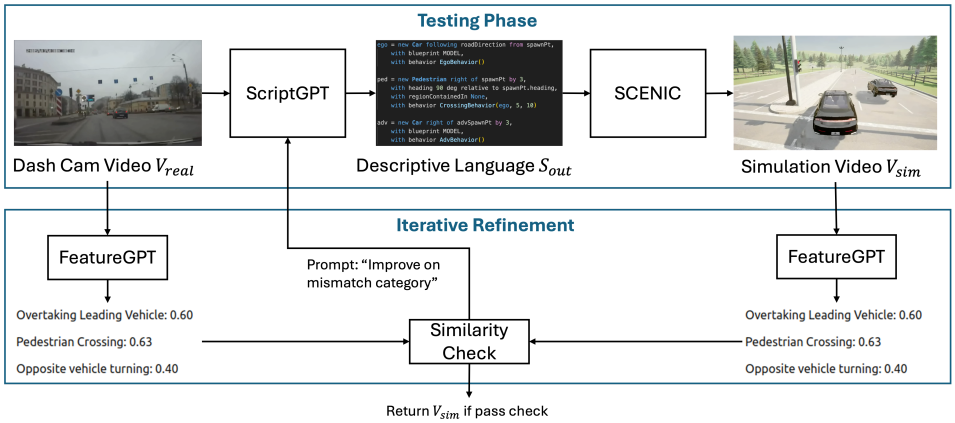


Figure 3: **Iterative Refinement Process**: After obtaining the simulated video V_{sim} from *ScriptGPT* and SCENIC, both the original and simulated videos are fed into *FeatureGPT* to evaluate the probabilities of predefined features. If the difference of any feature between the original and simulated videos exceeds a certain threshold, we iteratively refine *ScriptGPT* by incorporating additional feedback into the SCENIC script, guiding further scenario adjustments until the similarity improves.

3.4 Iterative Refinement

Finally, we perform iterative refinement on the *ScriptGPT* with the help of similarity check, as illustrated in Figure 3. If the absolute difference for a given category i is above a predefined threshold τ_i, we refine the SCENIC script $Sout$ by feeding the discrepancy for that category back into *ScriptGPT* as an additional prompt (e.g., "there shouldn't be a leading vehicle overtaking behavior, please improve on that"). This feedback allows *ScriptGPT* to adjust the SCENIC script S_{out} accordingly, generating a new version of the script and producing a new simulation video $V_{sim'}$.

The process is repeated iteratively until the difference for each category falls below the predefined threshold, i.e., $\|Sim(V_{real}, V_{sim'})_i\| \leq \tau_i$. Once the similarity across all categories pass the threshold check, the final simulation scenario is ready for testing ADS.

4 Experiments & Analysis

4.1 System Setup

Dataset We obtain the collision videos from the Car Crash Dataset (CCD) [17]. CCD is chosen because it contains real traffic accident videos captured by dashcams mounted on driving vehicles, potentially providing a rich source for developing and testing ADS. Moreover, our framework is also relevant for near misses events.

Simulator We use CARLA [2], an open-source platform designed to support the development and validation of autonomous driving systems. CARLA is selected for its realistic physics engine and high-fidelity environmental rendering, making it ideal for generating the simulation scenarios needed in our framework.

Description Language SCENIC We use SCENIC as the description language for driving scenarios due to its similarity to Python, which aligns well with GPT-4o's input data [14], making prompt engineering doable and more straightforward. Furthermore, SCENIC integrates seamlessly with the CARLA simulator and it is highly effective at specifying complex driving scenarios, making it well-suited for our application.

Prompt-Engineered *ScriptGPT* To construct the *ScriptGPT* model, we begin by writing 20 SCENIC scripts that cover diverse driving scenarios, such as overtaking, cruising, sudden stops due to obstacles, and turns in varying road and weather conditions. Using the SCENIC library and CARLA, we generate corresponding videos for each scenario and pair them with their respective SCENIC scripts, forming 20 (V_i, S_i) pairs. These pairs are then used as input data for prompt engineering GPT-4o, selected for its ability to learn and generalize from a wide range of examples.

The empirical design choice we made is to design only 20 pairs because 20 scenarios are sufficient to cover most of the common interesting scenarios. Moreover, in practice, since GPT-4o API does not natively accept video formats like .mp4, we preprocess the videos by sampling frames and concatenating them into an n-dimensional array. The same preprocessing technique is applied during testing phases to ensure consistency between training and testing phases, and we also apply the same to the *FeatureGPT*.

Prompt-Engineered *FeatureGPT* We use *FeatureGPT* to enhance our framework's ability to recognize specific driving behaviors. First, we predefine 10 driving feature categories, as shown in Table 1. Next, we create 20 SCENIC scripts representing scenario videos where these features may or may not appear. Each video is paired with a corresponding 10-dimensional feature vector (e.g., [parallel vehicle overtaking: 0, ... , leading vehicle stopped: 1], where 0 indicates absence and 1 indicates presence). The input data is used to prompt-engineer GPT-4o into *FeatureGPT*, which outputs a 10-dimensional probability vector for each video during inference. This allows us to compare and categorize driving behaviors between the original video V_{real} and the generated video V_{sim}. Again we made the empirical design choice of using 10 feature categories and 20 samples to cover most frequently encountered interesting driving behaviors through trial and error.

Iterative Refinement using Similarity Check Once *FeatureGPT* produces feature vectors for both the generated and original videos, we compare them to detect discrepancies. A large discrepancy in any feature indicates that the generated video is either missing a key behavior (negative gap) or introducing an unintended one (positive gap), and then we map the gap into natural language feedback (e.g., "there should be a leading vehicle overtaking behavior, please improve on that") and feed back into *ScriptGPT* to refine the SCENIC script. Gap thresholds τ_i for each feature i, as shown in Table 1, are customized through empirical testing.

Table 1: Pre-defined feature Category and Threshold

Pre-define Feature	Gap Threshold τ
Sunny / Rainy	0.3
Urban / Highway	0.3
Random Object on Road	0.1
Leading Vehicle Cruising	0.2
Leading Vehicle Stopped	0.2
Parallel Vehicle Cutting in	0.2
Parallel Vehicle Cruising	0.2
Parallel Vehicle Stopped	0.2
Behind Vehicle Overtaking	0.2
Opposite Vehicle Turning	0.2

4.2 Case Study

We show 5 interesting dashcam video from the CCD dataset and generate the corresponding simulation scenario using our framework, as shown from in Figure 4, 5, 6, 7, 8.

Figure 4: **Vehicle Cutting In with Pedestrian Crossing Scenario**: in the original dash camera video (top row), the vehicle on the right performs an emergency lane change to the left due to a jaywalking pedestrian in red. In the generated scenario (bottom row) produced by our framework, the vehicle on the right exhibited a similar lane change behavior to the left to avoid a jaywalking pedestrian.

Figure 5: **Opposite Vehicle Invading Lane Scenario**: in the original dash camera video (top row), the vehicle on the opposite lane gradually swifts to ego's lane probably due to loss of focus. In the generated scenario (bottom row) produced by our framework, the vehicle on the opposite lane exhibited a similar lane change behavior to switch to our lane and caused collision.

Figure 6: **Vehicle Spin Scenario**: in the original dash camera video (top row), the vehicle in front of the ego first spins to the left and then collided into the right vehicle. In the generated scenario (bottom row) produced by our framework, the front vehicle exhibited a similar spin and collision behavior

Figure 7: **Animal Crossing Scenario**: in the original dash camera video (top row), an animal attempted to cross the road, prompting the ego vehicle to perform an emergency lane change to the left. In the generated scenario (bottom row) produced by our framework, the ego vehicle exhibited a similar lane change behavior to the left to avoid a jaywalking pedestrian (since CARLA does not have an animal model, the animal is replaced by a pedestrian).

Figure 8: **Vehicle Cutting In with Stopped Object Scenario**: in the original dash camera video (top row), the vehicle on the right perform an emergency brake and tried lane change to the left due to a front parked vehicle. In the generated scenario (bottom row) produced by our framework, the vehicle on the right exhibited a similar lane change behavior to the left to avoid the stopped vehicle and cause a collision.

4.3 Preliminary Qualitative Result

Automated Pipeline After completing the prompt engineering process, our framework is capable of automatically generating the 5 scenarios mentioned in Section 4.2 without any human intervention during the testing phase. We also extended our framework to evaluate 50 randomly selected accidents from the CCD dataset, and found that 32 out of 50 (64%) scenarios generation can be fully automated without any human involvement, while 18 scenarios (36%) encountered syntax errors that required human debugging.

Time Efficiency Our framework significantly reduces the time required for real-to-simulation scenario generation. For the 32 videos that can be automatically generated, it takes 1.5 minutes per scenario on average during the testing phase, including iterative refinement, to produce a SCENIC script of approximately 70 lines of code. In contrast, manually coding and debugging a similar real-to-simulation scenario could take an experienced engineer several hours.

Advantage of Iterative Refinement The 5 scenarios in Section 4.2 underwent 1-2 iterations of refinement, resulting in notable improvements in both accuracy and realism. In the extended evaluation of 50 accidents, iterative refinement happened in 17 scenarios (34%), further showcasing its potential to enhance scenario quality and be generalized to more crash scenarios.

Framework Accuracy While we have not yet conducted a formal quantitative analysis on the framework's timing efficiency or objectively measure benefits of iterative refinement, the preliminary results provide strong evidence of the concept's validity. Our framework consistently captures the core driving behaviors from the original videos, indicating its effectiveness in generating accurate and realistic driving scenarios.

5 Limitations & Future Work

Figure 9: Scenarios from the Car Crash Dataset where our current framework cannot handle due to heavy traffic with multiple cars or compromised lighting environmental conditions at night

The current framework faces several challenges. It struggles with scenarios involving poor perception conditions or high complexity, as illustrated in Figure 9. Additionally, the framework's performance may degrade when testing scenarios (e.g. multi-car complicated driving scenario at night like shown in Figure 9) have not been encountered by *ScriptGPT* and *FeatureGPT* during the prompt engineering process. Therefore, the performance of our framework heavily relies on carefully and manually selecting diverse "positive-examples" for prompt engineering.

Another limitation is that both *ScriptGPT* and *FeatureGPT* operate as black-box neural networks affected by our prompt engineering with input data as a form of "training", providing two levels of indirection without any theoretical guarantees of performance or correctness. This lack of transparency can complicate the validation and debugging of generated scenarios.

For future work, we plan to conduct a human study to quantitatively assess the framework's time efficiency and accuracy. This will involve timing how long experts take to manually write real-to-simulation conversion scripts and asking them to rate the accuracy of our automated conversions. We also aim to improve the diversity of data samples used in the prompt engineering process to extend our framework's applicability across the entire Car Crash Dataset. Finally, we envision extending this video-to-video conversion framework to other domains, such as flying tasks or other robotics applications, broadening its use cases and potential impact.

6 Conclusion

In this paper, we have presented a novel framework for automatically converting real-world vehicle crash videos into simulation scenarios using prompt-engineered Video-Language Models. We have deploy multiple techniques including the similarity score metric and the iterative refinement process to ensure the generated scenarios closely align with the original videos. Despite the framework's current limitations, such as reliance on data diversity and the challenges of handling complex or unseen scenarios, through multiple examples, it demonstrates clear potential for improving ADS testing. Future work will focus on expanding data diversity, conducting quantitative human studies, and extending the framework to other robotics domains.

References

[1] WuLing Huang et al. "Autonomous vehicles testing methods review". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 163–168. DOI: 10.1109/ITSC.2016.7795548.

[2] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator". In: *CoRR* abs/1711.03938 (2017). arXiv: 1711.03938. URL: http://arxiv.org/abs/1711.03938.

[3] Guodong Rong et al. "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294422.

[4] Daniel J. Fremont et al. "Scenic: A Language for Scenario Specification and Scene Generation". In: *Proceedings of the 40th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*. 2019.

[5] Tommaso Dreossi et al. "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems". In: *International Conference on Computer Aided Verification*. Vol. 11561. LNCS. Springer, 2019, pp. 432–442.

[6] Cumhur Erkan Tuncali et al. "Requirements-driven Test Generation for Autonomous Vehicles with Machine Learning Components". In: *IEEE Transactions on Intelligent Vehicles* 5 (2 2020), pp. 265–280.

[7] Mohammad Hekmatnejad, Bardh Hoxha, and Georgios Fainekos. "Search-based Test-Case Generation by Monitoring Responsibility Safety Rules". In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. 2020.

[8] Yang Sun et al. "LawBreaker: An Approach for Specifying Traffic Laws andFuzzing Autonomous Vehicles". In: *Automated Software Engineering (ASE)*. 2022.

[9] Xiangyu Bai et al. "Bridging the Domain Gap between Synthetic and Real-World Data for Autonomous Driving". In: *ACM J. Auton. Transport. Syst.* 1.2 (2024). DOI: 10.1145/3633463. URL: https://doi.org/10.1145/3633463.

[10] Karim Elmaaroufi et al. *Generating Probabilistic Scenario Programs from Natural Language*. 2024. arXiv: 2405.03709 [cs.SE]. URL: https://arxiv.org/abs/2405.03709.

[11] J. Wang et al. "Learning Human Dynamics in Autonomous Driving Scenarios". In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 20739–20749. DOI: 10.1109/ICCV51070.2023.01901. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01901.

[12] Davis Rempe et al. *Generating Useful Accident-Prone Driving Scenarios via a Learned Traffic Prior*. 2022. arXiv: 2112.05077 [cs.CV]. URL: https://arxiv.org/abs/2112.05077.

[13] Sai Krishna Bashetty, Heni Ben Amor, and Georgios Fainekos. "DeepCrashTest: Turning Dashcam Videos into Virtual Crash Tests for Automated Driving Systems". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 11353–11360. DOI: 10.1109/ICRA40945.2020.9197053.

[14] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[15] Daniel J. Fremont et al. "Scenic: Language-Based Scene Generation". In: *CoRR* abs/1809.09310 (2018). arXiv: 1809.09310. URL: http://arxiv.org/abs/1809.09310.

[16] Haoyang Fan et al. "Baidu Apollo EM Motion Planner". In: *CoRR* abs/1807.08048 (2018). arXiv: 1807.08048. URL: http://arxiv.org/abs/1807.08048.

[17] Wentao Bao, Qi Yu, and Yu Kong. "Uncertainty-based Traffic Accident Anticipation with Spatio-Temporal Relational Learning". In: *ACM Multimedia Conference*. 2020.