SoVAR: Building Generalizable Scenarios from Accident Reports for Autonomous Driving Testing

An Guo
guoan218@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Yuan Zhou*
yuanzhou@zstu.edu.cn
School of Computer Science and Technology
Zhejiang Sci-Tech University
Hangzhou 310018, China

Haoxiang Tian
tianhaoxiang20@otcaix.iscas.ac.cn
College of Computing and Data Science
Nanyang Technological University
Singapore

Chunrong Fang*
fangchunrong@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Yunjian Sun
sunyunjian.syj@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Weisong Sun
weisong.sun@ntu.edu.sg
College of Computing and Data Science
Nanyang Technological University
Singapore

Xinyu Gao
xinyugao@smail.nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Anh Tuan Luu
anhtuan.luu@ntu.edu.sg
College of Computing and Data Science
Nanyang Technological University
Singapore

Yang Liu
yangliu@ntu.edu.sg
College of Computing and Data Science
Nanyang Technological University
Singapore

Zhenyu Chen*
zychen@nju.edu.cn
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China

Abstract

Autonomous driving systems (ADSs) have undergone remarkable development and are increasingly employed in safety-critical applications. However, recently reported data on fatal accidents involving ADSs suggests that the desired level of safety has not yet been fully achieved. Consequently, there is a growing need for more comprehensive and targeted testing approaches to ensure safe driving. Scenarios from real-world accident reports provide valuable resources for ADS testing, including critical scenarios and high-quality seeds. However, existing scenario reconstruction methods from accident reports often exhibit limited accuracy in information extraction. Moreover, due to the diversity and complexity of road environments, matching current accident information with the simulation map data for reconstruction poses significant challenges.

In this paper, we design and implement SoVAR, a tool for automatically generating road-generalizable scenarios from accident reports. SoVAR utilizes well-designed prompts with linguistic patterns to guide the large language model (LLM) in extracting accident information from textual data. Subsequently, it formulates and solves accident-related constraints in conjunction with the extracted accident information to generate accident trajectories. Finally, SoVAR reconstructs accident scenarios on various map structures and converts them into test scenarios to evaluate its capability to detect defects in industrial ADSs. We experiment with SoVAR, using the accident reports from the National Highway Traffic Safety Administration's (NHTSA) database to generate test scenarios for the industrial-grade ADS Apollo. The experimental findings demonstrate that SoVAR can effectively generate generalized accident scenarios across different road structures. Furthermore, the results confirm that SoVAR identified 5 distinct safety violation types that contributed to the crash of Baidu Apollo.

*Corresponding authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASE '24, October 27-November 1, 2024, Sacramento, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1248-7/24/10
https://doi.org/10.1145/3691620.3695037

CCS Concepts

• **Software and its engineering → Software testing and debugging.**

Keywords

Software testing, Automatic test generation, Constraint solving, Autonomous driving system

ACM Reference Format:
An Guo, Yuan Zhou, Haoxiang Tian, Chunrong Fang, Yunjian Sun, Weisong Sun, Xinyu Gao, Anh Tuan Luu, Yang Liu, and Zhenyu Chen. 2024. SoVAR: Building Generalizable Scenarios from Accident Reports for Autonomous Driving Testing. In *39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3691620.3695037

1 Introduction

The advent of autonomous driving technology has ushered in a new era of transportation, promising increased safety, efficiency, and convenience [27, 50]. However, the occurrence of crashes involving autonomous driving vehicles, including those that have resulted in fatalities [1, 4], serves as evidence that autonomous driving is not currently as safe as it is marketed to be. In many instances, these crashes can be attributed to defective software, highlighting the urgent requirement for an enhanced approach to testing autonomous driving software [33].

As one of the most critical quality assurance techniques, ADS testing has attracted the attention of both academia and industry [20, 30, 54]. The key testing techniques for ADSs can be classified into two categories: road testing and simulation testing. Road testing involves testing specific driving scenarios in closed autonomous vehicle proving grounds [33] or monitoring autonomous vehicles in real traffic [53], but this approach requires a long period and extensive resources. Additionally, the acquisition of diverse critical test data that encompasses a wide range of realistic usage scenarios presents significant challenges for testers. Therefore, high-fidelity simulation testing methods have become imperative to the development and validation of autonomous driving technologies [21, 43]. It conducts ADS testing in simulation platforms, such as LGSVL [38] and CARLA [15]. One essential aspect of ADS simulation testing is the identification and construction of critical scenarios that may lead to accidents.

Software engineering researchers have proposed utilizing real-life accident cases to generate critical test scenarios [9, 16–18, 51]. The main insight is that real car accidents present critical situations that pose significant challenges for self-driving cars. Consequently, recent research has mainly focused on scenario reconstruction from different driving data sources. This includes utilizing textual descriptions [17], accident sketches [18], sensor data [16], and video recordings [9, 51] to generate essential scenarios. Compared to sensor data and video recordings of critical cases, textual descriptions of crashes are more accessible and abundant [18, 24]. Accident report analysis approaches offer more comprehensive insights into collisions, including intricate details like weather, lighting, and road conditions, which are visually challenging to represent. However, the accuracy of information extracted through current scenario reconstruction methods based on accident reports is limited [17]. Furthermore, current methods can reproduce the accident scenarios only on the same road structure described in the accident report. It limits the application to simulation-based ADS testing because the road structures in the simulation may differ from those described in the accident reports. Reproducing accident scenarios on different roads is challenging and overwhelming [8, 45].

To bridge this gap, we propose an automatic and universal method, SoVAR, to reconstruct accident scenarios on different road structures from accident reports. The primary objective of SoVAR is to automatically reconstruct accident scenarios from accident reports on different roads within the simulation environment, which can then be used as initial seeds for ADS testing. SoVAR first leverages carefully designed linguistic prompt patterns to guide the LLM in accurately extracting environment, road, and object movement information. Based on the extracted information and the roads to reconstruct the scenarios, SoVAR formulates the constraints for the accident scenario. By solving these constraints with constraint solvers, we can generate the accident trajectories of the accident participants and finally reconstruct the accident scenarios. Therefore, the generated scenarios can be executed on the required map. Furthermore, to use these scenarios for ADS testing, SoVAR transforms them into testing scenarios by identifying the ego vehicle (i.e., the vehicle controlled by the ADS under testing) and the NPC (non-player character) vehicles. The NPC vehicles will follow the computed trajectories, while the ego vehicle is controlled by the ADS instead of following a predetermined trajectory.

To evaluate the effectiveness of SoVAR, we reconstruct scenarios based on well-known NHTSA's accident reports [37] and the San Francisco map provided by the LGSVL simulator. The generated scenarios are then used to test Baidu Apollo [2]. Our experimental findings demonstrate that the accident information extraction approach of SoVAR outperforms all baseline methods. Furthermore, subsequent results highlight that SoVAR successfully generated road-generalizable scenarios across different road structures compared to the existing textual accident report reconstruction method. Additionally, our experiments reveal that converting reconstructed scenarios into test cases effectively identified 5 distinct types of safety violation behaviors in the industrial-grade software Apollo.

The main contributions of this paper are summarized as follows:

- **Method.** We propose an innovative method for automatically reconstructing crash scenarios from accident reports and testing ADS. Our method enhances LLM's ability to accurately extract textual information by designing linguistic patterns for prompts. We then generate driving trajectories that align with desired road structures by solving a set of driving constraints.
- **Tool.** We implement the proposed approach into the automated testing tool, SoVAR. To support the open science community, we have made the source code available[1] and released the scenarios that led to the ADS collisions.
- **Study.** We utilize SoVAR to assess the industry-grade ADS Baidu Apollo and find fatal collisions. The results demonstrate that, compared with state-of-the-art scenario reconstruction techniques, our method can extract accident information more accurately and generate generalized accident scenarios that can adapt to

[1]https://github.com/meng2180/SoVAR

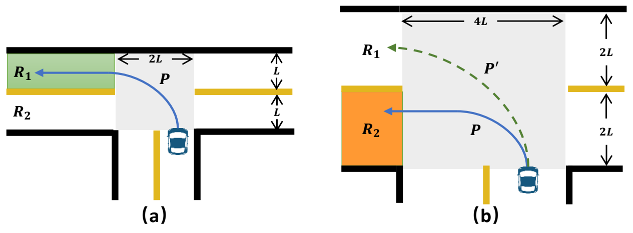

Figure 1: A motivating example of reconstructing an accident scenario on different road structures.

different road structures. Furthermore, our approach can be used for ADS testing and to identify various safety violations in ADS.

2 Background

2.1 Motivation

In the real world, the autonomous vehicle industry typically employs simulation-based testing, evaluation, and validation as the first step, followed by testing in controlled environments (such as closed roads and test sites), and finally progressing to testing in open road configurations [26]. During the simulation testing phase, where autonomous vehicles encounter unknown scenarios, efficiently constructing challenging scenarios becomes a top priority. To address this need, several academic institutions [12, 21, 44, 55] and companies [22, 23] have developed testing platforms that generate possible combinations of scenarios in an initial configuration, using only the necessary initial parameters and variables. Reconstructing scenarios from real accident reports aims to generate initial scenarios capable of detecting ADS defects. However, there are currently two main challenges in obtaining key information and performing scenario reconstruction:

Challenge 1. Accident reports encompass substantial information regarding the road environment and vehicle trajectories amidst traffic accidents [17]. Nevertheless, the extraction of accident information encounters significant challenges. Off-the-shelf natural language processing (NLP) tools do not exhibit satisfactory performance due to the inclusion of traffic jargon and non-standard phrase structures in accident report [18].

Challenge 2. ADS simulation testing is usually conducted in a simulator with different maps. After extracting information from the accident report, reconstructing the accident scenario necessitates matching the road structures on the map to be tested in the simulator. However, this task poses challenges for ADS engineers. On the one hand, the map covers all the key static properties of complex roads [3, 32], including road type, road size (length and width), etc. It is extremely difficult to find roads that fully match the extracted information when reproducing the scenario within the map to be tested [8, 45].On the other hand, it is necessary to test multiple roads on the map because collisions often occur in similar road environments, thereby challenging the ability to adapt to a specific map. The current method [17] does not include a mechanism to adapt the generated accident trajectory to the map. As shown in Figure 1, the accident that occurred in Figure 1(a) can also occur on the road in Figure 1(b). However, directly reconstructing the original trajectory on the right road cannot replay the accident. The vehicle turns left along the fixed trajectory P to the road R_1 with a width of L. When the road width becomes $2L$, the vehicle turns left along the same trajectory P to the reverse road R_2 with a width of $2L$, which is inconsistent with the expected trajectory P' and alters the semantics of the accident scenario (turn left to R1 and go straight -> turn left to R2 and retrograde). Therefore, planning a trajectory compatible with the simulation map during the scenario reconstruction process is necessary.

Based on the aforementioned challenges, we leverage LLMs to extract accident information and implement a suitable linguistic pattern prompting strategy to enhance extraction effectiveness. Additionally, we employ a constraint-solving strategy to generate trajectories of traffic participants, ensuring reproducibility on roads with varying lengths and widths, as well as different road types such as intersections and T-junctions.

2.2 Large Language Models for Textual Understanding

Large Language Models [28, 52] (LLMs) have gained prominence in recent years, demonstrating comparable or superior performance to humans in various NLP tasks [29], highlighting their ability to comprehend, generate, and interpret text. LLMs are sophisticated language models characterized by their massive parameter sizes and exceptional learning capabilities. Currently, LLMs are primarily employed through the utilization of prompts, which serve as concise cues or instructions that direct the model's output. This approach is commonly referred to as prompt-based learning.

In prompt-based learning, a pre-trained language model is optimized for various tasks through priming on natural language prompts [34]. These prompts consist of text segments that are combined with an input and then used to generate an output for the given task. This approach has proven effective for few-shot and zero-shot learning in numerous general-domain tasks. Recent research has shown that large language models exhibit promising results in the few-shot setting, occasionally outperforming previous approaches that utilize fine-tuned models [11]. The ChatGPT [49] (Chat Generative Pre-trained Transformer) from OpenAI, has billions of parameters and is trained on a vast dataset encompassing textual understanding. Directly using GPT to understand complex text has shown mediocre performance; therefore, appropriate prompt patterns and rules need to be designed to achieve optimal performance in downstream tasks.

In this paper, we employ GPT-4 [5] to extract information regarding driving accidents. Furthermore, we design linguistic patterns for information extraction prompts to facilitate GPT-4's rapid adaptation to the task of extracting information from accident reports.

3 Approach

In this section, we present the design and implementation of SoVAR, a tool devised for automatically reconstructing crash scenarios from accident reports and testing ADS. SoVAR consists of three steps to obtain simulation-based tests: information extraction, trajectory planning, and simulation and test generation, as illustrated in Figure 2. According to the layer-based scenario definition [7], SoVAR initially abstracts the accident scenario into three layers: the road network and traffic guidance objects, the environmental conditions, and the dynamic objects. It then leverages an LLM to systematically

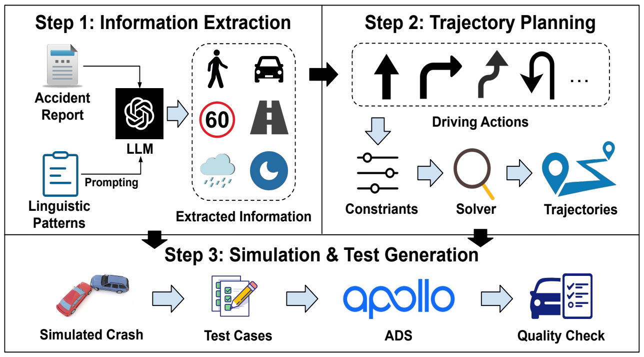

Figure 2: The overview of SoVAR.

extract information about accident-influencing factors from the accident report, organizing the information layer by layer. Following this, SoVAR establishes constraints on the pre-accident driving actions of traffic participants and employs a constraint solver to generate trajectories that comply with the specified constraints outlined in the accident report. Subsequently, the trajectories generated along with the extracted environment and road information are inputted into the driving simulator to reconstruct the car accident scenario. Finally, in the evaluation phase, SoVAR converts the generated simulation scenarios into test cases that include test oracles. These test cases are then inputted into the ADS. SoVAR checks whether the ADS under test successfully reaches its final expected position without encountering any crashes.

3.1 Information Extraction

According to the official general standards [47] for recording accident reports issued by the U.S. Department of Transportation, accident reports usually encompass valuable information regarding the road environment and vehicle trajectories. To understand complex sentence structures and traffic terminology in these texts, SoVAR leverages the LLM to extract this information. During information extraction, SoVAR incrementally parses the narrative of the car crash and accumulates the information about weather, lighting, roads, and vehicles into a data structure that forms the abstract of a car crash. While LLM demonstrates excellence in information extraction tasks, its performance can be significantly influenced by the quality of its prompt. Specifically, we need to design an appropriate prompt that precisely describes what needs to be queried or requested to enhance the extraction accuracy and effectiveness of the LLM. Section 3.1.1 outlines the information that will be extracted, while Section 3.1.2 details how we organize this information into a format that LLM can better comprehend.

3.1.1 *Layer-based Accident Abstract Representation.*

SoVAR only needs the descriptive text of the accident to extract information, without relying on additional data. Consistent with current work in scenario reconstruction [17, 51], SoVAR concentrates on the primary crash-contributing factors, namely lighting, weather, roads, and vehicle movements. It directly extracts information using LLM without introducing new information. To abstractly represent the accident information and organize it into a semantic structure that can be understood by the LLM, we present the extracted information in a hierarchical representation [40], divided into three layers: road, environment, and dynamic objects. Table 1 shows detailed descriptions and examples of the extracted attributes. If information is missing from an accident report, it indicates that the missing details are not crucial to the accident. SoVAR demonstrates strong versatility because it automatically assigns default values when necessary.

Environmental Conditions. The environmental condition layer encompasses weather and light conditions. Weather conditions (Weather) include factors such as rain, fog, snow, and others. Light conditions (Lighting) pertain to the lighting conditions on the road, typically brighter during the day and darker at night. Additionally, lighting may be enhanced by the street lights.

Road Network and Traffic Guidance. This layer describes the road network and the traffic signs used for guidance on the road. Road represents the geographical context of a crash, including the type of road where the accident occurred (CollisionLocation) and the number of lanes on the relevant roads (LaneNum). Additionally, SoVAR extracts information about the speed limit (SpeedLimit) of the road to reconstruct the speed constraints applicable to the location of the accident.

Dynamic Objects. This layer contains information about the striker and victim involved in the crash and the moving actions that led to the crash. SoVAR extracts information regarding the number of traffic participants (ParticipantsNumber) involved in simultaneous collisions and identifies the type of collision. The collision type information (CrashType) specifies the angle at which the traffic participants collided, including three types of collisions: rear-end collision, frontal collision, and front-to-side collision. For each car involved in the accident, SoVAR extracts the status of each vehicle, including the initial running lane of the vehicle (RunningLanes), the initial running direction of the vehicle (DrivingDirections), and the vehicle's behavior before the crash (DrivingActions). Specifically, vehicle behaviors describe the regular and abnormal actions [35] taken by the vehicle. Regular driving actions include U-turn, stop, drive into roads, vehicle cross, turn left, turn right, follow lane, and change lane. Abnormal driving actions include driving off the road and retrograde. Pedestrians involved in the accident are considered victims, and SoVAR constructs pedestrian cross and pedestrian walk actions for them.

3.1.2 *Linguistic Patterns of Information Extraction Prompt.*

With the information to be extracted, we design linguistic patterns to generate prompts for the LLM. To design the patterns, each of the three annotators is tasked with writing a prompt sentence following the regular prompt template [13, 19]. Subsequently, we assess the impact of accident information extraction. Using these prompt sentences, the three annotators conduct card sorting [41] and engage in discussions to derive linguistic patterns. As illustrated in Table 2, this process results in three linguistic patterns corresponding to the three sub-types of information outlined in Table 1. We show a simplified sample template for each linguistic pattern.

SoVAR extracts accident information layer by layer using the prompt patterns designed in Table 2. For each linguistic pattern, it first explains the meaning of each attribute to help LLM understand the extracted information. Additionally, the pattern includes heuristic rules to guide LLM in producing accurate results. For example, if a car intends to perform a certain action but a collision occurs

Table 1: Extracted accident information description and examples.

Id	Attribute	Description	Examples
		Accident context- Environment Conditions information	
1	Weather	Weather conditions at the time of the accident	Weather="Cloudy"
2	Lighting	Lighting conditions at the time of the accident	Lighting="Dark"
		Accident context- RoadNetwork and Traffic Guidance information	
3	LaneNum	Total number of lanes on the road where the accident occurred	LaneNum="4"
4	CollisionLocation	Type of road on which the accident occurred	CollisionLocation="T-junction"
5	SpeedLimit	Speed limit for the road on which the car is travelling	SpeedLimit="60"
		Accident context- Dynamic Object information	
6	DrivingActions	Actions of traffic participants before the accident	DrivingActions=["P1:[turn right,...]","P2:[follow lane"...]",...]
7	CrashType	Type of collision between the attacker and the victim at the time of the accident	CrashType="Rear-End"
8	DrivingDirections	Initial direction of travel for each traffic participant	DrivingDirections=["P1:westbound","P2:southbound"...]
9	RunningLanes	The initial lane position of each traffic participant	RunningLanes=["P1:1","P2:2"...]
10	ParticipantsNumber	Total number of traffic participants	ParticipantsNumber="2"

Table 2: The example of linguistic patterns for information extraction prompts.

Id	Pattern type	Sample of linguistic patterns/rules
1	Environment conditions information	You should help me extract environmental conditions. The answer includes <Weather> and <Lighting>. $\cdots$ For the <Weather>, it means the weather conditions when the accident happened$\cdots$. $\cdots$ If it's rainy when the accident happened, "rainy" should be added into the <Weather>$\cdots$.
2	RoadNetwork and Traffic Guidance	You should help me extract roadnetwork and traffic guidance information. The answer includes <CollisionLocation>,<LaneNum>, and <SpeedLimit>. $\cdots$ For the <CollisionLocation>, it means the type of road on which the accident occurred$\cdots$. $\cdots$ If the accident happened near or at an intersection or intersecting roadway, the answer of <CollisionLocation> is "intersection"$\cdots$.
3	Dynamic Object	You should help me extract dynamic object information. The answer includes <ParticipantsNumber>, <CrashType>, <DrivingDirections>, <RunningLanes> and <DrivingActions>. $\cdots$ For the <DrivingActions>$\cdots$, if the car proceeds to do an intended action but does not do actually, such as intending to turn right, this intended action must not be added to the <DrivingActions>$\cdots$.

before the action is executed, the intended driving action should not be extracted. Finally, we utilize few-shot learning to ensure the LLM's output conforms to our expected standards. Therefore, taking the linguistic patterns as references, the trajectory planning module can directly use the output to build trajectory constraints, which will be described in the next section.

3.2 Trajectory Planning

Accurately planning the trajectories of traffic accident participants that satisfy the extracted information and match the target roads is an essential process in accident scenario reconstruction [17, 18]. A trajectory is an ordered sequence of waypoints, i.e., positions and velocities a traffic participant must follow. For example, the i^{th} waypoint of the b^{th} action of vehicle X can be represented as $X_i^b = (x, y, v)$, where x, y, v are the x-coordinate, y coordinate, and velocity, respectively. In addition, we introduce $pos = (x, y)$ to represent the position of the waypoint in the floor plan.

SoVAR utilizes extracted driving actions and road information (including the roads from the accident report and the tested roads to construct and execute the scenarios) to simulate crashes and compute trajectories for simulated traffic participants. This approach first builds constraints on each action based on the road information and then generates trajectories by resolving the constraints. Therefore, it can adapt to different roads on various maps. Algorithm 1 outlines the generation of waypoints. To facilitate the display in the algorithm, the extracted attributes are represented by letter abbreviations. For example, driving action is abbreviated as DA. The algorithm takes road information $\mathbb{R}$ extracted from accident reports using LLM, extracted dynamic object information $\mathbb{D}$, the given map $\mathbb{MAP}$ in the simulator, and defined driving action constraints $\mathbb{C}$ as inputs. The algorithm initially parses the given map into a set of candidate roads, noting the road type (e.g., crossroads) for each road (Line 3). It then iterates through the set of candidate roads, sequentially selecting lanes of varying lengths and widths (Lines 4-5). If the current road type matches the type where the accident occurred and if the maximum number of lanes occupied by all traffic participants during their movement is less than or equal to the number of lanes on the selected road, the waypoint generation process begins (Lines 6-7). Subsequently, following the adjustment of the driving direction and initial lane position of the traffic participant to match the selected road (Lines 8-11), the constraint solver resolves constraints based on driving actions to generate waypoints (Line 12). The following describes the design of driving constraints $\mathbb{C}$ in detail.

To make participants perform corresponding actions and drives into the crash site, we define a group of trajectory constraints for each action from five aspects. Then, SoVAR leverages a constraint solver to automatically generate trajectories for each participant. Besides, traffic participants must reach the collision location simultaneously while executing their actions. To accomplish this, SoVAR introduces a collision area C_A, which is automatically calculated by the system.

Algorithm 1 Waypoints generation process of trajectory planning

```
Input :  The extracted dynamic object information $\mathbb{D}$, the extracted road information $\mathbb{R}$, the given map $\mathbb{MAP}$, the set of driving constraints $\mathbb{C}$
Output :  Generated waypoints $genWPoints$
1:  $CandidateRoad \leftarrow \emptyset$
2:  $SelectRoad \leftarrow \emptyset$
3:  $CandidateRoad = ParseMap(\mathbb{MAP})$
4:  for $Cand$ in $CandidateRoad$ do
5:      $SelectRoad = [Cand.len, Cand.wid, Cand.lnum, Cand.dir]$
6:      if $Cand.type == \mathbb{R}.CL$ then
7:          if $CalMaxLanes(\mathbb{R}.LN, \mathbb{D}) \leq Cand.lnum$ then
8:              $Recal \leftarrow \emptyset$
9:              $\mathbb{D}.RL.\mathbb{D}.DD = ConvertInfo(\mathbb{D}, SelectRoad)$
10:             $Recal.append(\mathbb{D}.RL)$
11:             $Recal.append(\mathbb{D}.DD)$
12:             $GenWP = SolveTraj(\mathbb{C}, \mathbb{D}.DA, Recal, SelectRoad)$
13:         end if
14:     end if
15: end for
16: return $GenWP$
```

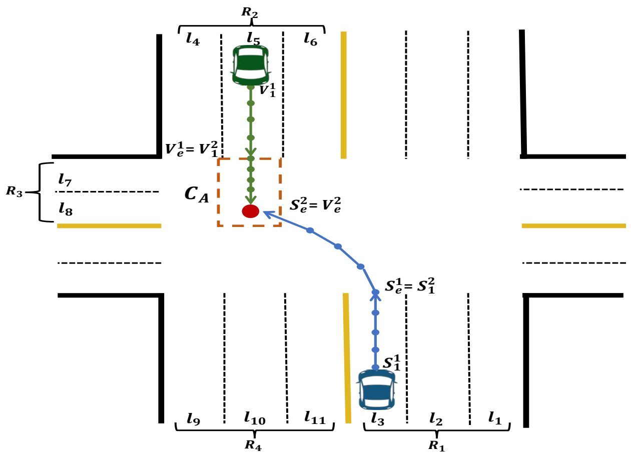

Figure 3: An example of accident trajectory planning (NHTSA report #2006048103067)

Due to space limitations, we have chosen an example to showcase these constraints. As shown in Figure 3, the collision takes place at a regulated intersection. A blue vehicle is observed traveling northbound in lane l_3, making a left turn under a circular green signal, while a green vehicle is traveling south in lane l_5. During the blue vehicle's left turn, its front collided with the front of the green vehicle. The striker vehicle is depicted as following the lane and performing a left turn, while the victim vehicle is shown following the lane and performing a crossing maneuver. The blue vehicle may turn into lane l_7 or lane l_8. SoVAR calculates the intersection of the drivable areas between the green vehicle and the blue vehicle, determining the collision area C_A. Next, we introduce the constraint design of the driving actions involved in the example. The constraints for all accident driving actions can be found in https://github.com/meng2180/SoVAR/blob/main/constraints.pdf.

3.2.1 Constraints of Participant Basic Driving Actions. To generate trajectories of participants that make them perform basic driving actions, we define a group of trajectory specifications for each behavior from three aspects:1) initial position and destination, 2) position to perform the action, and 3) velocities of waypoints.

Group 1: Constraints on initial position and destination. There are four constraints in Group 1. Equation 1 restricts that the driving direction from X_1^b to X_e^b is not opposite to the direction of lane l_m. The calculation of $fd\left(w_i, w_j, l\right)$ is defined as $\left(w_j.x - w_i.x\right)\left(l_{ex}.x - l_{en}.x\right) > 0 \wedge \left(w_j.y - w_i.y\right)\left(l_{ex}.y - l_{en}.y\right) > 0$. Here, l_{en} is the entrance point of lane l and l_{ex} is the exit point of the lane, which are both known after parsing the map. When fd is true, it guarantees waypoint w_j is ahead of w_i in the direction of lane l. Equation 2 restricts the road position where the ***Follow Lane*** behavior begins and ends. $X_1^b.pos$ and $X_e^b.pos$ are on the same lane l_m of the road R_i. Equations 3 and 4 limit the initial and the end road positions of the ***Turn Left*** and ***Vehicle Across*** actions: Equation 3 is applied when the driving action avoids a collision, and $X_1^b.pos$ and $X_e^b.pos$ are on the lanes of different roads; Equation 4 is employed when the action leads to a collision, and the position of destination $w_e^X.pos$ is in C_A.

$$fd\left(X_1^b.\,pos, X_e^b.pos, l_m\right) = 1 \tag{1}$$

$$X_1^b.pos \in l_m, X_e^b.pos \in l_m, l_m \in R_i \tag{2}$$

$$X_1^b.pos \in l_m \wedge l_m \in R_i, X_e^b.pos \in l_n \wedge l_n \in R_j, i \neq j \tag{3}$$

$$X_1^b.pos \in l_m \wedge l_m \in R_i, X_e^b.pos \in C_A \tag{4}$$

Group 2: Constraints on positions to perform the action. In this group, Equation 5 limits the position of the waypoint during the execution of the ***Turn Left*** action, ensuring that the current waypoint is on the right side of the line connecting two adjacent waypoints. As shown in Figure 4, $\overrightarrow{s_i s_{i+1}}$ represents the direction vector composed of the position of the i-th waypoint and the position of the $(i+1)$-th waypoint. The vector $\overrightarrow{s_i s_{i+2}'}$ represents the right normal vector formed by the positions of the i-th and $(i+2)$-th waypoints. Equations 6 and 7 respectively impose limits on the positional relationship between the roads accessed through the execution of the ***Turn Left*** behavior and the ***Vehicle Across*** behavior. k_1 and k_2 represent the slopes of roads l_m and l_n.

$$\forall i \in [1, e-1), \overrightarrow{s_i s_{i+1}} \cdot \overrightarrow{s_i s_{i+2}'} > 0 \tag{5}$$

$$\begin{cases} k_1 * k2 = -1, l_{en}^m \in l_m, l_{ex}^n, l_{en}^n \in l_n \\ k_1 = \left(l_{ex}^m.y - l_{en}^m.y\right) / \left(l_{ex}^m.x - l_{en}^m.x\right) \\ k_2 = \left(l_{ex}^n.y - l_{en}^n.y\right) / \left(l_{ex}^n.x - l_{en}^n.x\right) \end{cases} \tag{6}$$

$$\begin{cases} \arctan\left|\frac{k_1 - k_2}{1 + k_1 k_2}\right| < \frac{\pi}{2}, l_{en}^m \in l_m, l_{ex}^n, l_{en}^n \in l_n \\ k_1 = \left(l_{ex}^m.y - l_{en}^m.y\right) / \left(l_{ex}^m.x - l_{en}^m.x\right) \\ k_2 = \left(l_{ex}^n.y - l_{en}^n.y\right) / \left(l_{ex}^n.x - l_{en}^n.x\right) \end{cases} \tag{7}$$

Group 3: Constraints on the velocities of waypoints. Finally, group 3 restricts the velocities of all intermediate waypoints. Specifically, Equation 8 restricts velocities of waypoints from X_1^b to X_e^b, and Equation 9 restricts the limits of speeds during driving. $DX(i, j)$ and $DY(i, j)$ respectively compute the distance that the vehicle traveled from the i^{th} waypoint to the j^{th} waypoint along X axis and Y axis. $DX(i, j) = \sum_{c=i}^{j-1}\left(\frac{v_c.x + v_{c+1}.x}{2} * \Delta t_c\right), DY(i, j) =$

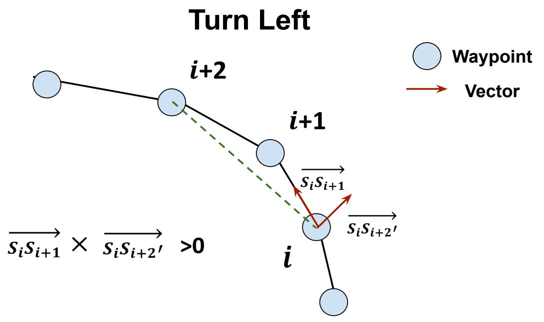
Turn Left
Waypoint
Vector
$i+2$
$i+1$
$\overrightarrow{s_i s_{i+1}}$
$\overrightarrow{s_i s_{i+2'}}$
i
$\overrightarrow{s_i s_{i+1}} \times \overrightarrow{s_i s_{i+2'}}$ >0

Figure 4: A graphic explanation of Equation 5 in Group 2.

$\sum_{c=i}^{j-1}\left(\frac{v_c.y+v_{c+1}.y}{2} * \Delta t_c\right)$. $v_c.x$ is the c^{th} waypoint's speed along X axis and $v_c.y$ is its speed along Y axis. spd_i is the speed of v_i. spd^{limit} is the speed limit for the driving road.

$$\left(X_1^b.x - X_e^b.x\right) = DX(1,e), \left(X_1^b.y - X_e^b.y\right) = DY(1,e) \tag{8}$$

$$\forall i \in (1,\mathrm{e}], 0 < spd_{i-1}^X = spd_i^X \leq spd^{limit} \tag{9}$$

3.2.2 Constraints between Multiple Driving Actions of Participants. To ensure that the striking and victim vehicles execute a sequence of driving actions and reach the accident site simultaneously, we define trajectory specifications from two aspects: 1) trajectory combinations of multiple basic driving actions, and 2) constraints on vehicle crashes.

Group 4: Constraints on trajectory combinations of multiple basic driving actions. The fourth group constrains the relationship between waypoints when multiple actions are connected. Equations 10 and 11 represent the trajectory constraints for multiple actions exhibited by the striker and victim, respectively. The ending position of a traffic participant's current action should align with the starting position of the subsequent driving action. The striker performed n actions, while the victim performed m actions before the collision.

$$\forall p \in (1,\mathrm{n}], S_1^p.pos = S_{\mathrm{e}}^{p-1}.pos \tag{10}$$

$$\forall q \in (1,\mathrm{m}], V_1^q.pos = V_{\mathrm{e}}^{q-1}.pos \tag{11}$$

Group 5: Constraints on vehicle crash. The fifth group of constraints ensures that the collision information is consistent with the report description. Equation 12 limits the occurrence of the accident collision to the location stated in the collision report, while Equation 13 describes the striker and victim simultaneously arriving at the collision location.

$$S_{\mathrm{e}}^n.pos = V_{\mathrm{e}}^m.pos \land S_{\mathrm{e}}^n.pos \in C_A \land V_{\mathrm{e}}^m.pos \in C_A \tag{12}$$

$$\sum_{i=1}^{n} S_{t_i} = \sum_{i=1}^{m} V_{t_i} \tag{13}$$

3.3 Simulation and Test Generation

Simulation generation. SoVAR is not limited to a specific simulator; it requires a simulation environment that encompasses precise soft-body physics and authentic 3D textures representing roads, vehicles, weather, and lighting conditions. Additionally, the simulator must have the capability to control the virtual car using external software, as it serves as a fundamental requirement for conducting autonomous driving simulation testing [15, 38]. The simulator receives the waypoints generated by the trajectory planning module alongside the environment conditions information extracted by LLM and conducts a simulation to reconstruct the crash detailed in the accident report.

Test generation. The simulations generated by SoVAR contain the essential information needed to reconstruct the car crashes described in the accident reports. However, these simulated crashes do not meet the specifications of the test cases because they lack proper test oracles and do not permit any exogenous interference. Specifically, the generated simulations cannot verify if the behavior of the ego car is acceptable, and the ego car cannot freely interact with the simulated environment. To address this, SoVAR automatically derives system-level test cases from the output of the trajectory planning module, allowing the ego car to choose a trajectory different from the one described in the accident report to avoid collisions. For the simulation scenarios generated based on the reports, SoVAR creates multiple test scenarios by designating each traffic participant as the ego vehicle. The starting point of each ego vehicle is set to the initial point of the generated waypoints, while the NPC vehicles follow the waypoints provided by the SoVAR trajectory planning module. Collision scenarios involving ego vehicles are recorded during these runs. However, some collisions are not caused by the ego vehicle, such as when an NPC vehicle crashes into a legally parked ego vehicle. To minimize false positives, SoVAR counts collision scenarios in which the ego vehicle's speed is not near zero.

4 Experimental Design

In this section, we introduce the experimental design, including the experiment settings, evaluation metric and baseline in the experiments. All experiments are performed on Ubuntu 21.10 desktop with GeForce RTX 4070, one 16-core processor at 3.80GHz, and 32GB RAM.

SoVAR aims to use accident reports as information sources to reconstruct accident scenarios and convert them into test cases for evaluating ADS. To this end, we empirically explore the following three research questions (RQ):

- RQ1: How effective is information extraction from accident reports with SoVAR?
- RQ2: What is the effectiveness of SoVAR in producing generalizable simulations with the intended impact?
- RQ3: Do tests derived from generalizable scenarios find crashes in autonomous driving systems?

4.1 Experiment Settings

Dataset and LLM. To investigate the research questions, we acquired data from the National Motor Vehicle Crash Causation Survey database, maintained by the National Highway Traffic Safety Administration (NHTSA) [37]. The NHTSA adheres to the Model Minimum Uniform Crash Criteria (MMUCC) guidelines [36], which outline the primary factors contributing to crashes. To obtain the necessary accident reports, we divide the NHTSA police reports according to crash type and then randomly select a sample of reports from each category. We obtained NHTSA reports, enabling

Table 3: Results of different methods for extracting information from accident reports.

Attributes	Environment Conditions		RoadNetwork and Traffic Guidance			Dynamic Objects				
	Weather	Light	LaneNum	SpeedLimit	CollisionLocation	DrivingActions	CrashType	DrivingDirections	RunningLanes	ParticipantsNumber
SoVAR	77.33%	93.33%	93.33%	100%	96.00%	76.33%	90.67%	91.66%	77.00%	100%
SoVAR_N	69.33%	81.33%	76.00%	91.33%	95.33%	55.00%	90.00%	91.33%	73.00%	100%
AC3R	54.67%	76.00%	70.67%	82.67%	78.67%	43.67%	11.67%	39.30%	25.33%	94.67%

comprehensive coverage of diverse environmental conditions such as daytime and nighttime scenarios, various road types, as well as different crash types. We adopted GPT-4 [5] as the information extraction model. We chose GPT-4 because it is the current state-of-the-art LLM, widely known and easily accessible.

Simulation Platform. We connected the scene recovery outcomes with the leading virtual testing platform, LGSVL, widely used in academia and industry. LGSVL simulator is a high-fidelity simulation tool specifically designed for autonomous driving [38]. Its advanced engine enables comprehensive end-to-end simulation and seamless integration with Apollo, allowing numerous virtual testing scenarios to be generated. In our study, we implemented SoVAR in the simulation environment built with Baidu Apollo 6.0 [2] (the ADS under test) and SORA-SVL simulator [25].

Constraints Solver. We employ Z3 [10], an efficient Satisfiability Modulo Theories (SMT) solver, to compute the trajectories of participants. Z3 is renowned for its efficiency and incorporates state-of-the-art algorithms that allow it to tackle large-scale problems swiftly and accurately. Z3 has garnered widespread adoption in both academia and industry for various applications, including software analysis, cyber-physical systems, and beyond.

4.2 Evaluation Metric and Baseline Comparison

Metric. To evaluate the success rate of accident scenario reconstruction, we propose using the "Scenario Reconstruction Rate" metric to measure the effectiveness of SoVAR in reconstructing scenarios from accident reports. Here an accident that is successfully reconstructed if the participants move along the generated trajectories can replay the same collision. And we express it as follows:

$$SRR = \frac{1}{N_r} \sum_{i=1}^{N_r} \mathbb{1} \left[\bigwedge_{\forall tr \in TR_i} SIM(tr, \mathbb{MAP}) == 1 \right] \tag{14}$$

where N_r represents the total number of accident reports, $\bigwedge$ means logical AND, TR_i represents the set of accident-related trajectories generated by the i-th report, $\mathbb{MAP}$ represents the map used to reconstruct the scenario, and SIM is a function that evaluates whether the trajectory, derived from correctly parsed collision actions, is correct when executed in the simulator. Specifically, SIM outputs 1 if the generated trajectory does not illegally cross any lines and the collision angle matches the report description; otherwise, it outputs 0. Additionally, $\mathbb{1}$ is an indicator function mapping a boolean condition to a value in {0, 1}: If the condition is true, it returns 1; otherwise, it returns 0.

Baseline. To evaluate ADS, there are various research works focusing on scenario reconstruction using accident information. Some methods leverage video recordings [9, 51] and accident sketches [18] as information sources to directly identify and extract trajectories of traffic participants. In contrast, our approach, similar to AC3R [17], automatically plans and generates trajectories based on the textual description provided in accident reports.

To assess the effectiveness of SoVAR in accident information extraction and its capability to generate generalized scenarios, we employ AC3R as a baseline for comparison. AC3R integrates NLP techniques with a domain-specific ontology to extract pertinent information from accident reports and subsequently generates trajectories using hardcoded rules with fixed patterns. It should be noted that the AC3R method requires accident reports to be segmented before inputting into the information extraction module. To demonstrate the best performance of AC3R, we segment all reports for evaluation. In contrast, to showcase SoVAR's ability to handle complex text, we use raw, unsegmented accident reports as input to SoVAR. Additionally, to further validate the effectiveness of linguistic pattern prompting in information extraction, we use SoVAR_N as another baseline. The SoVAR_N approach leverages GPT-4 to directly extract information from accident reports without employing any linguistic pattern prompting.

5 Result Analysis and Discussion

5.1 Answer to RQ1

To evaluate the effectiveness of information extraction from accident reports with SoVAR, we randomly select 150 reports encompassing diverse environmental conditions, road structures, and vehicle behaviors. Then, two of the authors independently analyze each accident report to establish the ground truth of extracted information. A third author is involved in a group discussion to resolve conflicts and reach agreements. Next, we execute the information extraction modules of SoVAR, SoVAR_N, and AC3R, respectively, and automatically calculate the average extraction accuracy of each attribute across all selected reports.

Results. Table 3 illustrates the accuracy of various methods for extracting information from accident reports. The extraction accuracy of SoVAR for the environment conditions layer, road network layer, and dynamic object layer are 85.33%, 96.44%, and 87.13%, respectively. Compared to the SoVAR_N and AC3R methods, employing GPT-4 with linguistic patterns prompting can significantly enhance accuracy across all attribute information extractions. Specifically, the average accuracy of the SoVAR method surpassed that of the SoVAR_N and AC3R methods by 7.3% and 31.9%, respectively. We observe that SoVAR and AC3R exhibit higher accuracy in extracting straightforward information such as SpeedLimit from the accident report. However, they demonstrate relatively lower effectiveness in extracting complex driving actions with contextual dependencies.

Discussion. According to the experimental analysis presented in Table 3, it is evident that utilizing LLMs, such as GPT-4, to extract information from accident reports is a highly effective method. Additionally, employing well-designed linguistic patterns for prompting

Table 4: The number of successfully reconstructed scenarios and scenario reconstruction rate under different road types.

Type		Intersection			T-junction			Straight Road		
		I1	I2	I3	T1	T2	T3	S1	S2	S3
SoVAR	Num.	47	45	48	35	36	38	42	41	40
	SRR	94%	90%	96%	70%	72%	76%	84%	82%	80%

*Note that the scenario reconstruction rate for AC3R is 0.

can significantly enhance the LLM's ability to extract information. While our method achieves good extraction results, there is still room for improvement in the accuracy of attributes such as driving actions before a collision. Upon reviewing reports containing extraction errors, we identified that many inaccuracies stem from implicit actions described within accident report descriptions. For instance, a retrograde action might be described simply as "crossing the double yellow line." Additionally, there is no consistent pattern for how each accident report from the NHTSA database is structured. By applying a large language model with stronger generalization and inference capabilities, or by fine-tuning the model after manually labeling the accident information, the information extraction effect of SoVAR can be further optimized.

5.2 Answer to RQ2

To ascertain the generalizability of the SoVAR method, i.e., its capacity to replicate accidents across roads of varying lengths and widths depicted on maps, we reconstruct scenarios based on accident location reports from the NHTSA database. These reports encompass accidents occurring on straight roads, T-junctions, and intersection locations. Specifically, we randomly select 50 accident reports for each road type for experimental purposes, and SoVAR is executed three times for each report, ensuring that each run covers different roads on the San Francisco map. We tally the number of successful reconstruction scenarios for each type of road accident. Additionally, for scenarios that are not fully successfully reconstructed, we analyze the causes and classify them into three categories: trajectory planning program crashes, crash type mismatches during simulation generation, and instances of driving actions crossing the line during simulation generation.

Results. Table 4 shows the number of successfully reconstructed scenarios and SRR metrics under different road types. The average scenario reconstruction rate (SRR) of accidents occurring at intersections, T-junction roads, and straight roads are 93.3%, 72.7%, and 82.0% respectively. Since the length and width of all roads chosen for reproduction in the experiment are inconsistent with those of AC3R, the experiments on AC3R found that none of the scenarios can be successfully reconstructed for simulation testing. To further analyze the recurrence results, Table 5 presents the fault localization analysis of scenarios during their reconstruction. The proportion of scenarios generated by SoVAR that cannot be completely reconstructed is 17.33%. Most of these scenarios fail due to Crash Type simulation errors, accounting for 80.8% of all failed reconstruction attempts. Since AC3R does not generate adaptive waypoints according to lane width and length during operation and lacks a mechanism to map waypoints to a map, we don't collect statistics on AC3R vehicles crossing the line, as shown in Table 5. It can be seen from the table that SoVAR has a 60% lower average accident reconstruction failure rate compared to AC3R. These results demonstrate that, in comparison to AC3R, SoVAR is more capable of generating generalized scenarios.

Discussion. Experiments demonstrate that SoVAR can generate generalized scenarios on lanes of varying lengths and widths. It is noted that 2% of the scenarios reconstructed by SoVAR involve crossing-the-line situations. This issue arises due to slight errors in the lane width of the same road when the map creators manually construct the experimental simulation map. Most of the errors in the AC3R method stem from the trajectory planning module. Besides, the method for generating waypoints in AC3R is hard-coded. Given the complex combinatorial logic between adjacent actions, it is inevitable that some waypoints cannot be generated due to insufficient consideration of the code logic during execution.

5.3 Answer to RQ3

Here, we assess the effectiveness of test cases derived from simulations generated by SoVAR in detecting crashes involving ADS and identifying safety concerns. We randomly selected 200 accident reports from the NHTSA database for scenario reconstruction, of which 39.5%, 26.5%, and 34% occurred at intersections, T-junctions, and straight roads, respectively. We then convert the reconstructed scenarios into test cases and evaluate the performance of the Apollo ADS. We compare SoVAR with a random scenario generation method. Specifically, the random scenario generation method utilizes the motion task of the ego vehicle generated by SoVAR and randomly creates reasonable NPC trajectories. We then separately quantify the number of collisions experienced by autonomous vehicles using both SoVAR and the random method during simulated test scenarios. Additionally, to further analyze the cause of the collisions, we capture the trajectories of the ego vehicle and the NPC vehicle when collisions occurred during test execution. Our thorough examination of ego and NPC actions aims to identify and classify various safety-violation scenarios that emerged.

Results. Table 6 presents the number of collisions caused by self-driving cars in generated test scenarios using different methods. The table reveals that the number of collisions identified by SoVAR across various road types exceeds those found by the random method. Overall, the total number of collisions detected by SoVAR is more than six times higher than that detected by the random method. In our comprehensive analysis, we identify and summarize 5 types of safety violations in the Apollo ADS. Due to space limitations, we present four safety violation scenario types as shown in Figure 5. The left sub-figure of each group of safety violation types describes the location and driving route of the NPC and ego car in the initial scene, and the right sub-figure is a schematic diagram of the scene when a collision occurs. Illustrations for other types of safety violations can be found in the open-source repositories we provide. From the safety violation types illustrated in Figure 5, it is evident that various regular NPC actions, such as vehicle across, turn left, and drive into roads, can cause collisions involving the vehicle equipped with Apollo ADS. In addition, as shown in Figure 6, we summarize another two types of collisions, which are caused by irregular NPC actions.

Table 5: Fault localization analysis of not successfully reconstructed scenarios.

Method	Trajectory Planning				Simulation Crash Type				Simulation Crossing				Total			
	I	T	S	Percentage	I	T	S	Percentage	I	T	S	Percentage	I	T	S	Percentage
SoVAR	0.00%	0.00%	1.33%	1.33%	1.78%	8.00%	4.22%	14.00%	0.44%	1.11%	0.44%	2.00%	2.22%	9.11%	6.00%	17.33%
AC3R	21.62%	24.32%	22.52%	68.47%	2.70%	2.70%	3.60%	9.01%	—	—	—	—	24.32%	27.03%	26.13%	77.48%

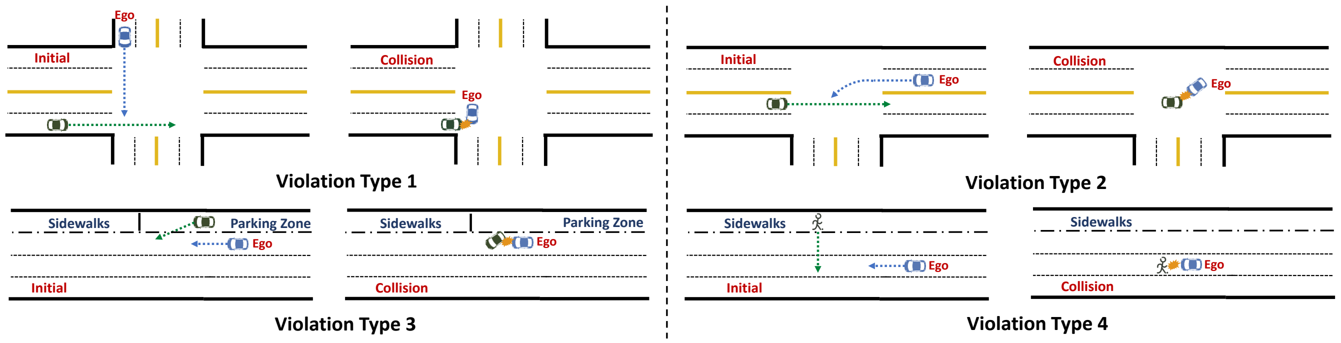

Figure 5: Visualization samples showing the safety violations of Apollo detected by SoVAR on different road types.

Table 6: The number of collisions caused by ADS in generated test scenarios using different methods.

Collision	Intersection	T-junction	Straight Road	Total
SoVAR	15	4	6	25
Random	1	1	2	4

Discussion. Through an in-depth analysis of the results presented in Table 6, it becomes apparent that scene reconstruction utilizing information extracted from accident reports proves to be an effective method for testing the safety of ADS. This effectiveness stems from the fact that test scenarios generated from scenario descriptions provided in accident reports, which include driving actions among other details, are indeed crucial for ensuring safety. Furthermore, the safety violations we identify encompass erroneous behaviors in Apollo's operational capabilities. Specifically, violation type 1 demonstrates that when the NPC vehicle crosses an intersection at high speed, the ego vehicle fails to decelerate in time. Violation type 2 shows that when the NPC vehicle goes straight through an intersection, the ego vehicle turns without yielding, as required by traffic rules, resulting in a collision. Violation type 3 and violation type 4 indicate that the ego vehicle does not slow down in time when the NPC enters the road from the parking zone or sidewalks. The causes of the accidents in collision type 1 and collision type 2 are not attributed to Apollo. Collision type 1 shows that when an NPC turns left from the right lane at an intersection without traffic lights, the ego vehicle results in a collision with the NPC. Collision type 2 demonstrates that when an NPC crosses the double yellow line and drives in the opposite direction, the ego vehicle does not stop but continues to move forward, leading to a collision. Although in these cases, the NPCs violate traffic rules, we believe that ADS should have emergency avoidance capabilities when there is sufficient time to react. In such scenarios, the ADS's ability to issue early warnings and take countermeasures is crucial to ensuring safety.

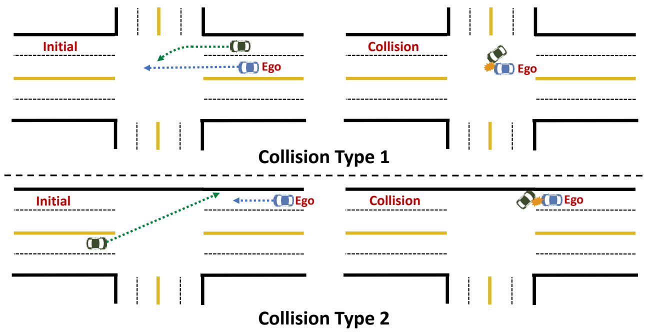

Figure 6: Visualization samples showing another two collision types detected by SoVAR.

5.4 Threats to Validity

Data Selection. The selection of accident report data poses a primary threat to validity. We randomly selected a small number of police reports from the NHTSA database for experiments, because evaluating the effect of GPT-4 on extracting accident information requires time-consuming and laborious manual annotation. Nonetheless, the NHTSA database is a widely utilized dataset in academia, and we took measures to ensure that the selected reports encompass a range of environmental conditions, crash types, road geometries, and driving actions. We believe that our approach can be readily applied to other accident report datasets containing detailed descriptions of accident scenarios.

Environmental Simulation. One of the validity threats arises from limitations in the environmental simulation within the simulator. the simulator lacks the simulation of LiDAR point cloud data during adverse weather conditions. To address these challenges, we directly send ground-truth perception data to Apollo in the form of messages to confirm that Apollo correctly perceives information, and then test downstream modules of perception in Apollo, including prediction, planning, and control [2]. Actually, many ADS testing

[2]https://github.com/ApolloAuto/apollo?tab=readme-ov-file#architecture

works use a similar simulation environment without considering the perception module [14, 42, 46]. Although we don't apply the impact of environmental factors in LGSVL, our experimental results demonstrate that our method remains effective in detecting flaws in Apollo. Moving forward, we plan to validate our approach on other simulators that support ADS perception capabilities.

Count for Randomness. Another potential threat stems from the stochastic nature of constraint solving used in SoVAR, where Z3 is employed to generate scenarios that satisfy constraints. Due to the inherent multiple feasible solutions in constraint solving, the results of each run may exhibit slight variations. However, this characteristic also serves as an advantage for SoVAR, as it can produce diverse scenarios that meet the specified constraints. To mitigate this threat, we conducted each experiment three times to validate the method. Remarkably, the results of each SoVAR experiment consistently outperformed other baseline methods, underscoring the robustness and effectiveness of our approach.

6 Related Work

6.1 Driving Accident Scenario Reconstruction

Traffic accidents inherently encompass a vast amount of valuable semantic information, thereby facilitating the comprehension of the accident process by reconstructing the accident scenario [6, 31]. Recent research efforts have primarily concentrated on scenario reconstruction across various driving data sources, including leveraging sensor data collected during actual crashes [16], textual descriptions [17], accident sketches [18], and video recordings [9, 51] to generate critical scenarios. Erbsmehl [16] recreates crashes by replaying the sensory data collected during actual crashes. However, this approach has limited applicability as it relies on naturalistic field operational data, which is not generally available. Gambi et al. implement an automatic crash construction tool AC3R, which automatically extracts information leveraging natural language processing and reproduces crashes in simulated environments [17]. Subsequently, Gambi et al. present CRISCE, a semi-automated approach for generating simulations of critical scenarios from accident sketches that commonly complement crash reports [18]. Zhang et al. propose a panoptic segmentation model, M-CPS, designed to extract accident information from images or video recordings [51].

The key differences between the related work mentioned above and SoVAR are twofold: (1): Trajectories of traffic participants can be directly extracted from accident sketches or videos. Accident reconstruction from textual descriptions may result in multiple trajectories that satisfy the given text descriptions. (2): SoVAR can accurately extract accident information and generate generalized scenarios adaptable to different map structures.

6.2 ADS Simulation Testing

Reconstructing core road scenarios from original accident information holds significant value for ADS simulation testing [43]. The reconstructed scenarios can serve as initial test scenarios for ADS fuzz testing [39, 48]. Numerous studies have examined and unveiled crashes exhibited by ADS through the reconstruction of accident scenarios [9, 17, 39, 51]. Gambi et al. present a crash construction tool, which generates test cases by incorporating test oracles to evaluate the performance of the DeepDriving self-driving car software based on simulations [17]. Bashetty et al. adopt a framework that aims to extract 3D vehicle trajectories from dashcam videos. The framework then recreates these extracted scenarios to facilitate the testing of collision avoidance systems [9]. Tian et al. extract behaviors from collision-related trajectories and perform behavior pattern mining to generate critical scenarios for testing Baidu Apollo [48]. Zhang et al. propose a mutation algorithm based on the original accident scenario set to facilitate testing of Apollo ADS [51]. Scheuer et al. implement the avoidable collision scenarios generation tool by extending focused collision descriptions using a multi-objective optimization algorithm [39].

7 Conclusion

This paper introduces and assesses SoVAR, a tool designed for automatically reconstructing crash scenarios from accident reports and testing ADS. SoVAR leverages LLM to extract detailed accident information, significantly enhancing LLM's text comprehension and parsing capabilities through the design of specialized linguistic patterns for extraction prompts. To enhance the method's generalization capability in reconstructing accident scenarios, SoVAR generates accident-related trajectories by solving a predefined set of trajectory specifications. Experimental results demonstrate that our method can successfully replicate accident reports on different map structures and the reconstructed simulation scenarios can identify various types of safety violations based on industrial-grade ADS. These findings underscore the crucial role of SoVAR in upholding the quality and dependability of ADS.

Acknowledgement

We would like to thank anonymous reviewers for their insightful comments. This work is supported partially by the National Natural Science Foundation of China (61932012, 62372228), and Science, Technology and Innovation Commission of Shenzhen Municipality (CJGJZD20200617103001003), National Research Foundation, Singapore, and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-008), Program of China Scholarship Council (Grant No. 202306190132), and Science Foundation of Zhejiang Sci-Tech University (ZSTU) under Grant No. 24232204-Y.

References

[1] 2019. Feds Say Self-Driving Uber SUV Did Not Recognize Jaywalking Pedestrian In Fatal Crash. https://www.npr.org/2019/11/07/777438412/feds-say-self-driving-uber-suv-did-not-recognize-jaywalking-pedestrian-in-fatal-.
[2] 2020. Apollo 6.0. https://github.com/ApolloAuto/apollo. Accessed on May 12, 2024.
[3] 2021. ASAM OpenDRIVE. https://www.asam.net/index.php?eID=dumpFile&t=f&f=4422&token=e590561f3c39aa2260e5442e29e93f6693d1cccd.
[4] 2023. San Francisco self-driving car involved in serious accident. https://www.straitstimes.com/world/united-states/san-francisco-self-driving-car-involved-in-serious-accident.
[5] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
[6] Farman Ali, Amjad Ali, Muhammad Imran, Rizwan Ali Naqvi, Muhammad Hameed Siddiqi, and Kyung-Sup Kwak. 2021. Traffic accident detection and condition analysis based on social networking data. *Accident Analysis & Prevention* 151 (2021), 105973.
[7] Gerrit Bagschik, Till Menzel, and Markus Maurer. 2018. Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1813–1820.

[8] Zhibin Bao, Sabir Hossain, Haoxiang Lang, and Xianke Lin. 2023. A review of high-definition map creation methods for autonomous driving. *Eng. Appl. Artif. Intell.* 122 (2023), 106125. https://doi.org/10.1016/J.ENGAPPAI.2023.106125

[9] Sai Krishna Bashetty, Heni Ben Amor, and Georgios Fainekos. 2020. DeepCrashTest: Turning Dashcam Videos into Virtual Crash Tests for Automated Driving Systems. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020.* IEEE, 11353–11360. https://doi.org/10.1109/ICRA40945.2020.9197053

[10] Nikolaj S. Bjørner, Leonardo de Moura, Lev Nachmanson, and Christoph M. Wintersteiger. 2018. Programming Z3. In *Engineering Trustworthy Software Systems - 4th International School, SETSS 2018, Chongqing, China, April 7-12, 2018, Tutorial Lectures (Lecture Notes in Computer Science, Vol. 11430)*, Jonathan P. Bowen, Zhiming Liu, and Zili Zhang (Eds.). Springer, 148–201. https://doi.org/10.1007/978-3-030-17601-3_4

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[12] Nacer Eddine Chelbi, Denis Gingras, and Claude Sauvageau. 2022. Worst-case scenarios identification approach for the evaluation of advanced driver assistance systems in intelligent/autonomous vehicles under multiple conditions. *J. Intell. Transp. Syst.* 26, 3 (2022), 284–310. https://doi.org/10.1080/15472450.2020.1853538

[13] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 2778–2788. https://doi.org/10.1145/3485447.3511998

[14] Mingfei Cheng, Yuan Zhou, and Xiaofei Xie. 2023. BehAVExplor: Behavior Diversity Guided Testing for Autonomous Driving Systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, René Just and Gordon Fraser (Eds.). ACM, 488–500. https://doi.org/10.1145/3597926.3598072

[15] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings (Proceedings of Machine Learning Research, Vol. 78)*. PMLR, 1–16.

[16] Christian Erbsmehl. 2009. Simulation of real crashes as a method for estimating the potential benefits of advanced safety technologies. In *Technical Conference on the Enhanced Safety of Vehicles.*

[17] Alessio Gambi, Tri Huynh, and Gordon Fraser. 2019. Generating effective test cases for self-driving cars from police reports. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*, Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (Eds.). ACM, 257–267. https://doi.org/10.1145/3338906.3338942

[18] Alessio Gambi, Vuong Nguyen, Jasim Ahmed, and Gordon Fraser. 2022. Generating Critical Driving Scenarios from Accident Sketches. In *IEEE International Conference On Artificial Intelligence Testing, AITest 2022, Newark, CA, USA, August 15-18, 2022.* IEEE, 95–102. https://doi.org/10.1109/AITest55621.2022.00022

[19] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 8410–8423. https://doi.org/10.18653/V1/2022.ACL-LONG.576

[20] An Guo, Yang Feng, and Zhenyu Chen. 2022. LiRTest: augmenting LiDAR point clouds for automated testing of autonomous driving systems. In *ISSTA '22: 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, South Korea, July 18 - 22, 2022*, Sukyoung Ryu and Yannis Smaragdakis (Eds.). ACM, 480–492. https://doi.org/10.1145/3533767.3534397

[21] An Guo, Yang Feng, Yizhen Cheng, and Zhenyu Chen. 2024. Semantic-guided fuzzing for virtual testing of autonomous driving systems. *J. Syst. Softw.* 212 (2024), 112017. https://doi.org/10.1016/J.JSS.2024.112017

[22] Mokrane Hadj-Bachir, Erik Abenius, Jean-Claude Kedzia, and Philippe de Souza. 2019. Full Virtual ADAS Testing. Application to the Typical Emergency Braking EuroNCAP Scenario. (2019).

[23] Mokrane Hadj-Bachir, Philippe de Souza, Javed Shaik, and Gruyer Dominique. 2020. Evaluating autonomous functions performance through simulation using interoperable sensor, vehicle, and environment models. In *FISITA Web Congress 2020 & World Congress 2021.*

[24] H Hizal Hanis and SMR Sharifah Allyana. 2009. The construction of road accident analysis and database system in Malaysia. In *14th IRTAD Conference*. Citeseer, 16–17.

[25] Yuqi Huai. 2023. SORA-SVL. https://ics.uci.edu/~yhuai/SORA-SVL/. Accessed on May 30, 2024.

[26] Wuling Huang, Kunfeng Wang, Yisheng Lv, and Fenghua Zhu. 2016. Autonomous vehicles testing methods review. In *19th IEEE International Conference on Intelligent Transportation Systems, ITSC 2016, Rio de Janeiro, Brazil, November 1-4, 2016.* IEEE, 163–168. https://doi.org/10.1109/ITSC.2016.7795548

[27] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. 2020. Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art. *Found. Trends Comput. Graph. Vis.* 12, 1-3 (2020), 1–308. https://doi.org/10.1561/0600000079

[28] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[29] Md. Tahmid Rahman Laskar, M. Saiful Bari, Mizanur Rahman, Md Amran Hossen Bhuiyan, Shafiq Joty, and Jimmy X. Huang. 2023. A Systematic Study and Comprehensive Evaluation of ChatGPT on Benchmark Datasets. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, 431–469. https://doi.org/10.18653/V1/2023.FINDINGS-ACL.29

[30] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael B. Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. 2020. AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems. In *31st IEEE International Symposium on Software Reliability Engineering, ISSRE 2020, Coimbra, Portugal, October 12-15, 2020*, Marco Vieira, Henrique Madeira, Nuno Antunes, and Zheng Zheng (Eds.). IEEE, 25–36. https://doi.org/10.1109/ISSRE5003.2020.00012

[31] Yancheng Ling, Zhenliang Ma, Xiaoxian Dong, and Xiaoxiong Weng. 2024. A deep learning approach for robust traffic accident information extraction from online chinese news. *IET Intelligent Transport Systems* (2024).

[32] Rong Liu, Jinling Wang, and Bingqi Zhang. 2020. High definition map for automated driving: Overview and analysis. *The Journal of Navigation* 73, 2 (2020), 324–341.

[33] Guannan Lou, Yao Deng, Xi Zheng, Mengshi Zhang, and Tianyi Zhang. 2022. Testing of autonomous driving systems: where are we and where should we go?. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, Abhik Roychoudhury, Cristian Cadar, and Miryung Kim (Eds.). ACM, 31–43. https://doi.org/10.1145/3540250.3549111

[34] Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118* (2021).

[35] Wassim G Najm, John D Smith, Mikio Yanagisawa, et al. 2007. *Pre-crash scenario typology for crash avoidance research.* Technical Report. United States. Department of Transportation. National Highway Traffic Safety ....

[36] National Highway Traffic Safety Administration (NHTSA). 2003. MMUCC model minimum uniform crash criteria second edition. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/809577. Accessed on May 30, 2024.

[37] National Highway Traffic Safety Administration (NHTSA). 2024. National motor vehicle crash causation survey. https://crashviewer.nhtsa.dot.gov/LegacyNMVCCS/Search. Accessed on May 12, 2024.

[38] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Martins Mozeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In *23rd IEEE International Conference on Intelligent Transportation Systems, ITSC 2020, Rhodes, Greece, September 20-23, 2020.* IEEE, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294422

[39] Franz Scheuer, Alessio Gambi, and Paolo Arcaini. 2023. STRETCH: Generating Challenging Scenarios for Testing Collision Avoidance Systems. In *IEEE Intelligent Vehicles Symposium, IV 2023, Anchorage, AK, USA, June 4-7, 2023.* IEEE, 1–6. https://doi.org/10.1109/IV55152.2023.10186634

[40] Maike Scholtes, Lukas Westhofen, Lara Ruth Turner, Katrin Lotto, Michael Schuldes, Hendrik Weber, Nicolas Wagener, Christian Neurohr, Martin Herbert Bollmann, Franziska Körtke, et al. 2021. 6-layer model for a structured description and categorization of urban traffic and environment. *IEEE Access* 9 (2021), 59131–59147.

[41] Donna Spencer. 2009. *Card sorting: Designing usable categories.* Rosenfeld Media.

[42] Yang Sun, Christopher M. Poskitt, Jun Sun, Yuqi Chen, and Zijiang Yang. 2022. LawBreaker: An Approach for Specifying Traffic Laws and Fuzzing Autonomous Vehicles. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022.* ACM, 62:1–62:12. https://doi.org/10.1145/3551349.3556897

[43] Shuncheng Tang, Zhenya Zhang, Yi Zhang, Jixiang Zhou, Yan Guo, Shuang Liu, Shengjian Guo, Yan-Fu Li, Lei Ma, Yinxing Xue, and Yang Liu. 2023. A Survey on Automated Driving System Testing: Landscapes and Trends. *ACM Trans. Softw. Eng. Methodol.* 32, 5 (2023), 124:1–124:62. https://doi.org/10.1145/3579642

[44] Shuncheng Tang, Zhenya Zhang, Jixiang Zhou, Yuan Zhou, Yan-Fu Li, and Yinxing Xue. 2023. EvoScenario: Integrating Road Structures into Critical Scenario Generation for Autonomous Driving System Testing. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 309–320.

[45] Yun Tang, Yuan Zhou, Kairui Yang, Ziyuan Zhong, Baishakhi Ray, Yang Liu, Ping Zhang, and Junbo Chen. 2022. Automatic Map Generation for Autonomous Driving System Testing. *CoRR* abs/2206.09357 (2022). https://doi.org/10.48550/ARXIV.2206.09357

[46] Yun Tang, Yuan Zhou, Tianwei Zhang, Fenghua Wu, Yang Liu, and Gang Wang. 2021. Systematic Testing of Autonomous Driving Systems Using Map Topology-Based Scenario Classification. In *36th IEEE/ACM International Conference on Automated Software Engineering, ASE 2021, Melbourne, Australia, November 15-19, 2021*. IEEE, 1342–1346. https://doi.org/10.1109/ASE51524.2021.9678735

[47] the U.S. Department of Transportation. 2022. ANSI-D20 Traffic Records Systems Data Dictionary: Release 6.0. https://highways.dot.gov/safety/data-analysis-tools/rsdp/rsdp-tools/ansi-d20-traffic-records-systems-data-dictionary-release. Accessed on August 17, 2024.

[48] Haoxiang Tian, Guoquan Wu, Jiren Yan, Yan Jiang, Jun Wei, Wei Chen, Shuo Li, and Dan Ye. 2022. Generating Critical Test Scenarios for Autonomous Driving Systems via Influential Behavior Patterns. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*. ACM, 46:1–46:12. https://doi.org/10.1145/3551349.3560430

[49] Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. *IEEE CAA J. Autom. Sinica* 10, 5 (2023), 1122–1136. https://doi.org/10.1109/JAS.2023.123618

[50] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* 8 (2020), 58443–58469. https://doi.org/10.1109/ACCESS.2020.2983149

[51] Xudong Zhang and Yan Cai. 2023. Building Critical Testing Scenarios for Autonomous Driving from Real Accidents. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, René Just and Gordon Fraser (Eds.). ACM, 462–474. https://doi.org/10.1145/3597926.3598070

[52] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *CoRR* abs/2303.18223 (2023). https://doi.org/10.48550/ARXIV.2303.18223

[53] Xingyu Zhao, Valentin Robu, David Flynn, Kizito Salako, and Lorenzo Strigini. 2019. Assessing the Safety and Reliability of Autonomous Vehicles from Road Testing. In *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*, Katinka Wolter, Ina Schieferdecker, Barbara Gallina, Michel Cukier, Roberto Natella, Naghmeh Ramezani Ivaki, and Nuno Laranjeiro (Eds.). IEEE, 13–23. https://doi.org/10.1109/ISSRE.2019.00012

[54] Ziyuan Zhong, Gail E. Kaiser, and Baishakhi Ray. 2023. Neural Network Guided Evolutionary Fuzzing for Finding Traffic Violations of Autonomous Vehicles. *IEEE Trans. Software Eng.* 49, 4 (2023), 1860–1875. https://doi.org/10.1109/TSE.2022.3195640

[55] Yuan Zhou, Yang Sun, Yun Tang, Yuqi Chen, Jun Sun, Christopher M Poskitt, Yang Liu, and Zijiang Yang. 2023. Specification-based autonomous driving system testing. *IEEE Transactions on Software Engineering* (2023).