SCTrans: Constructing a Large Public Scenario Dataset for Simulation Testing of Autonomous Driving Systems

Jiarun Dai
Fudan University, China
jrdai14@fudan.edu.cn

Bufan Gao
Fudan University, China
22210240085@m.fudan.edu.cn

Mingyuan Luo
Fudan University, China
22210240096@m.fudan.edu.cn

Zongan Huang
Fudan University, China
21210240088@m.fudan.edu.cn

Zhongrui Li
Fudan University, China
21210240089@m.fudan.edu.cn

Yuan Zhang
Fudan University, China
yuanxzhang@fudan.edu.cn

Min Yang
Fudan University, China
m_yang@fudan.edu.cn

ABSTRACT

For the safety assessment of autonomous driving systems (ADS), simulation testing has become an important complementary technique to physical road testing. In essence, simulation testing is a scenario-driven approach, whose effectiveness is highly dependent on the quality of given simulation scenarios. Moreover, simulation scenarios should be encoded into well-formatted files, otherwise, ADS simulation platforms cannot take them as inputs. Without large public datasets of simulation scenario files, both industry and academic applications of ADS simulation testing are hindered.

To fill this gap, we propose a transformation-based approach SCTrans to construct simulation scenario files, utilizing existing traffic scenario datasets (i.e., naturalistic movement of road users recorded on public roads) as data sources. Specifically, we try to transform existing traffic scenario recording files into simulation scenario files that are compatible with the most advanced ADS simulation platforms, and this task is formalized as a Model Transformation Problem. Following this idea, we construct a dataset consisting of over 1,900 diverse simulation scenarios, each of which can be directly used to test the state-of-the-art ADSs (i.e., Apollo and Autoware) via high-fidelity simulators (i.e., Carla and LGSVL). To further demonstrate the utility of our dataset, we showcase that it can boost the collision-finding capability of existing simulation-based ADS fuzzers, helping identify about seven times more unique ADS-involved collisions within the same time period. By analyzing these collisions at the code level, we identify nine safety-critical bugs of Apollo and Autoware, each of which can be stably exploited to cause vehicle crashes. Till now, four of them have been confirmed.

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04…$15.00
https://doi.org/10.1145/3597503.3623350

CCS CONCEPTS

• **Software and its engineering** → **Software verification and validation**; • **Computer systems organization** → **Robotics**.

KEYWORDS

simulation scenario, autonomous driving, model transformation

ACM Reference Format:
Jiarun Dai, Bufan Gao, Mingyuan Luo, Zongan Huang, Zhongrui Li, Yuan Zhang, and Min Yang. 2024. SCTrans: Constructing a Large Public Scenario Dataset for Simulation Testing of Autonomous Driving Systems. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3623350

1 INTRODUCTION

Due to the potential to reshape mobility, autonomous driving systems (ADS) have attracted significant attention from investors [91] and companies [11]. Recently, autonomous vehicles (AV) have emerged from laboratories and are starting test runs on public roads. However, during these test runs, traffic accidents involving AVs continue to occur. To date, the California DMV [3] (Department of Motor Vehicles) has received more than 600 AV collision reports, showing varied property damages or human injuries.

In this state of confusion, safety has become the main obstacle to the widespread adoption of ADSs. Manufacturers are in severe need of safety testing methodologies to identify ADS flaws that may cause traffic accidents. Generally, there are two types of widely used testing methodologies, including road testing [81, 92] and simulation testing [39, 49, 79, 80]. In practice, physical road testing is usually costly and difficult to cover rare driving situations (e.g., surrounding vehicles cut in unexpectedly). Hence, simulation testing has emerged as an important complementary technique for the safety assessment of ADS, due to its sufficient flexibility. Generally, simulation testing is a scenario-driven testing methodology. A simulation scenario consists of various configurable settings to characterize each simulated object, agent, and scene during virtual driving. These configurations should be encoded into simulation scenario files (see §2.3) as inputs to the ADS simulation platform. In practice, the quality of the simulation scenarios directly determines the effectiveness of ADS simulation testing.

Given the importance of simulation scenarios, however, the public availability of simulation scenario files is quite limited, largely hindering the applications of ADS simulation testing. Existing datasets provide vehicle-side raw sensor data (e.g., camera, LiDAR, and radar) [43, 54, 70, 72, 84, 87–90] or naturalistic trajectories of traffic participants [41, 59], for the training and testing of standalone algorithms within ADS (e.g., object detection, object tracking, and motion planning). However, the formats of these traffic recording files are usually self-defined and are far from compatible with ADS simulation platforms. Although some groups [31] or companies [21] claimed that they have curated adequate well-formatted simulation scenario files to facilitate ADS simulation testing, these datasets are not freely accessible due to commercial considerations. Without a public dataset of simulation scenario files, both practitioners [64] from ADS companies and academic researchers [52, 53, 62, 93] have no choice but to manually curate desired ones, which is quite time-consuming and expertise-dependent.

In light of this, we are highly motivated to construct a large public dataset of ready-to-use simulation scenario files. However, this is quite a challenging task due to the following two reasons:

- *Hard to Ensure Diversity:* Diversity means that the simulation scenarios should cover as many realistic driving conditions (i.e., those can be seen during real-world driving) as possible, to ensure both the realism and the completeness of simulation testing. Obviously, it cannot be done simply by referring to expert knowledge, which is inevitably biased or inadequate.
- *Hard to Ensure Usability:* Usability means that the scenario configurations should be encoded into well-formatted files with domain-specific languages (e.g., OpenScenario [2]), otherwise current advanced simulators or ADSs cannot accept them as inputs. Unfortunately, these scenario specifications usually imply complex grammar models for unambiguous descriptions of every possible configuration parameter, consequently imposing extensive coding efforts. For example, it might take more than 200 lines of well-formatted code to instantiate a simple lane-cutting scenario with OpenScenario [2].

Our Work. To address the above challenges, our overall idea is to convert existing traffic scenario datasets (i.e., inD [41], CommonRoad [37], highD [59]) into simulation scenario datasets. Our approach is named SCTrans. The key rationale behind is that, these traffic scenario datasets are usually collections of naturalistic behaviors of road users (e.g., naturalistic vehicle movement recorded on highways), which contain very diverse and realistic traffic patterns as data sources for the parameterization of simulation scenarios. Although existing traffic scenario datasets have been encoded into machine-readable formats (see Table 1), these formats are not compatible with advanced ADSs (i.e., Apollo [25] and Autoware [4]) and simulators (i.e., LGSVL [79] and Carla [49]). To tackle this issue, we formalize the dataset transformation as a Model Transformation Problem [67]. With carefully implemented transformation rules, we ensure the generated scenario files can satisfy the specification requirements (see Table 2) of these advanced simulators and ADSs.

Note that, although these traffic scenario datasets have claimed that they can potentially be used for ADS testing, to the best of our knowledge, we are the first to make them truly capable of whole-system ADS testing in the most advanced simulation environment (e.g., LGSVL+Apollo and Carla+Autoware). Among all these traffic scenario datasets, CommonRoad [37, 86] takes a step further, combining with the lightweight simulator SUMO [61] to evaluate the ADS planning module. But, as a comparison, our curated scenario files can be used to expose the safety flaws of all ADS modules (i.e., perception, prediction, planning, and control) in high-fidelity simulators (i.e., LGSVL and Carla).

Results. From existing traffic scenario datasets, we can potentially construct over 13,000 simulation scenarios. Such a number is continuously increasing since these traffic scenario datasets are actively maintained. To evaluate the prototype of SCTrans, we try to construct an initial version of the simulation scenario dataset, based on 2,000 randomly selected traffic scenarios from CommonRoad [37], inD [41] and highD [59]. After ignoring 6/2,000 invalid traffic scenarios (e.g., incomplete files), we utilize SCTrans to transform the remaining 1,994/2,000 traffic scenarios into simulation scenarios. Results show that the scenario transformation has 100% syntactic correctness (i.e., the generated files are syntactically correct), introducing only about 0.3% semantic deviations compared to the input traffic scenarios (i.e., the generated scenarios are of high realism).

To evaluate the usability of our dataset, we run each generated simulation scenario file in the most advanced ADS simulation platforms, i.e., LGSVL+Apollo and Carla+Autoware. Results show that 100% of them can be properly parsed and loaded by these simulation platforms. Besides, the dataset, even in its initial version, has already shown high diversity according to the scenario labels (e.g., road shapes and weather conditions) defined in ISO 21448 [20] (guiding principles for ADS testing). Statistically, our dataset can over 78.5% of simulation-capable scenario labels suggested by ISO 21448, significantly outperforming existing datasets. To further demonstrate the utility of our dataset, we try to investigate whether our generated simulation scenarios can boost the collision-finding capability of existing simulation-based ADS fuzzers [52, 53, 62, 93]. Specifically, we randomly select 50 scenarios from our dataset and feed them to the ADS fuzzer as seed inputs. Compared to the manually curated seed scenarios utilized in existing works [52, 53, 62, 93], our scenarios help identify about 7 times more unique traffic collisions of Apollo and Autoware within the same time period. Through code-level root cause analysis, we identify 9 safety-critical bugs of Apollo and Autoware. After submitting the bug reports (i.e., bug locations and root causes) to the developers, till now, 4 of them have been confirmed.

Contributions. This paper makes the following contributions:

- We propose SCTrans to construct ready-to-use simulation scenario files for ADS simulation testing, utilizing existing traffic scenario datasets as data sources.
- We release a large public dataset of simulation scenario files, which are compatible with state-of-the-art driving simulators (i.e., LGSVL and Carla) and high-level ADSs (i.e., Apollo and Autoware). Each scenario has labels for ease of scenario selection (e.g., road types and driving tasks). Our dataset is available at https://seclab-fudan.github.io/SCTrans/.
- We conduct experiments to demonstrate the usability, diversity, and utility of our dataset. Results show that it can boost the collision-finding capability of existing ADS fuzzers, helping identify 9 safety-critical bugs of Apollo and Autoware.

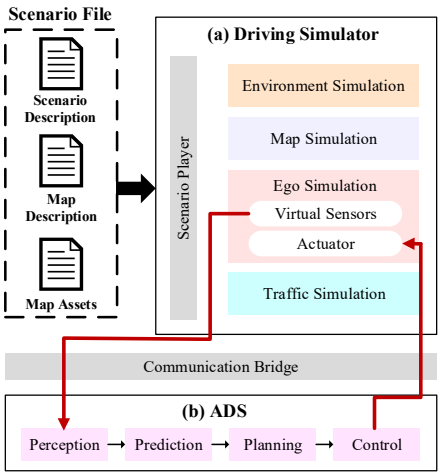


Figure 1: Architecture of ADS Simulation Testing.

2 BACKGROUND

2.1 Autonomous Driving System (ADS)

Levels of ADS. According to the levels of automation defined by the Society of Automotive Engineers (SAE) [71], ADSs can be categorized into 6 levels, ranging from L0 (no driving automation) to L5 (full driving automation). Nowadays, manufacturers are engaged to promote the deployment of high-level ADSs (e.g., L4). In this work, our goal is to prepare simulation scenario files for simulation testing of these high-level ADSs.

High-level ADS. High-level ADSs have multiple system modules, which work together to achieve automatic driving in a real-time fashion, mainly including *Perception Module* that perceives the driving environment via collected sensor data, *Prediction Module* that estimates the future status of surrounding objects, *Planning Module* that produces a feasible trajectory towards the destination and *Control Module* that generates vehicle control commands (e.g., throttle, brake and steering) along this trajectory. These modules work asynchronously and are managed through a high-performance middleware framework (e.g., Cyber-RT [13] and ROS [30]) during runtime.

2.2 ADS Simulation Testing

Architecture of ADS Simulation Platform. Figure 1 demonstrates the general architecture of the ADS simulation platform. Given a scenario file (detailed in §2.3) as input, the simulation platform can thus accordingly monitor the ADS misbehaviors in this scenario. Generally, built on top of the graphics rendering engine [33, 34], the driving simulator features 4 important components to process the ADS simulation testing:

- The *Ego Simulation* is responsible for simulating the motion of the ego vehicle. Specifically, the ego vehicle is equipped with various virtual sensors. These sensors can produce realistic sensor data inside the simulated environment, which is continuously fed to the ADS as input through a communication bridge. After the computation of the ADS, the generated vehicle control commands are sent back to the simulator to control the ego movement.
- The *Environment Simulation* allows customization of environmental attributes, mainly including weather conditions and lighting conditions. These attributes affect the way virtual sensors perceive the scene.
- The *Map Simulation* aims to create virtual maps, which include road networks and road infrastructures (e.g., traffic lights), etc.
- The *Traffic Simulation* aims to simulate the traffic actors (e.g., cars, bicycles, and pedestrians) around the ego vehicle.

2.3 Simulation Scenario

Definition. According to the literature [48], a simulation scenario can be regarded as the collection of all necessary configurations to characterize simulated objects, agents, and scenes at any instant of time during virtual autonomous driving. To be more specific, as discussed §2.2, these configurations can be classified into 4 categories, namely ego-related configuration (Config-ego), environment-related configuration (Config-env), map-related configuration (Config-map), and traffic-related configuration (Config-tfc).

- Config-env specifies the parameters related to weather conditions and lighting conditions (e.g., Carla simulator [49] features 14 pre-defined weather settings to choose from).
- Config-map determines the types and coordinates of road networks and road infrastructures (e.g., traffic lights).
- Config-ego specifies the basic calibration model [77] of the ego vehicle (e.g., maximum steering angle), as well as the target driving task (e.g., the destination and the initial speed).
- Config-tfc defines the behaviors of the traffic actors except for the ego vehicle (e.g., moving along a pre-defined trajectory).

Description Languages. To avoid ambiguously describing the simulation scenarios, researchers [65] have proposed various domain-specific languages (DSL) to concretely and formally express the scenario configurations mentioned above. For example, GeoScenario [76] or OpenScenario [2] are widely used to specify Config-env, Config-ego and Config-tfc. In addition, OpenDrive [1], OSM [28] and Lanelet [40] are used to specify Config-map.

Scenario File. Following the aforementioned DSLs, all configurations of a simulation scenario should be saved in a well-formatted system-independent file, a.k.a., a scenario file, so as to ensure its portability to the different testing environments. As shown in Figure 1, a scenario file usually has three sub-files, namely *Scenario Description File*, *Map Description File* and *Map Assets File*, each of which is indispensable for one run of the simulation. *Scenario Description File* specifies Config-env, Config-ego and Config-tfc. *Map Description File* specifies Config-map. Additionally, *Map Assets file* is usually a binary file for 3D map visualization (rendering), which stores meshes, materials, and textures of map elements.

Existing Datasets of Simulation Scenario Files. After a systematic review of the literature, we find that some existing works [46, 55, 56, 69, 73, 85] also aim to construct ready-to-use simulation files. However, among all these works, only D. Karunakaran *et al.* [56] released 9 scenario files to the community. Besides, there exist some third-party repositories [8, 16, 27] that provide open-source simulation scenarios, however, only dozens are available. Hence, we are highly motivated to construct and release a large public

dataset of simulation scenario files to facilitate the applications of ADS simulation testing. In particular, we also conducted extensive experiments (see §6.4) to compare our curated dataset with these existing ones in diversity.

2.4 Traffic Scenario

Traffic Scenario Datasets. There exists a line of research, namely traffic scenario construction [37, 41, 59]. These works aim to automatically extract the concrete meta-information (i.e. agent type and recording time) and moving trajectories (i.e., position, orientation, speed, and acceleration) of road users from real-world driving records (e.g., videos), and further save them in machine-readable file formats (e.g., CSV and XML). For example, the highD dataset [59] and the inD dataset [41] store the naturalistic trajectories of road users recorded on highways or at intersections; CommonRoad[37] dataset not only stores diverse traffic trajectories, but also accordingly design benchmarks to evaluate ADS planning algorithms, including target driving tasks and cost functions.

Description Languages. The authors of existing traffic scenario datasets [37, 41, 59] have discussed the potential to apply them in the (whole-system) simulation testing (see §2.2). However, these traffic scenarios are usually described using self-defined XML-based or CSV-based languages [10, 17, 19], which are largely different from those of simulation scenarios (see §2.3).

Traffic Scenario Recording File. Existing datasets describe each traffic scenario with a corresponding traffic scenario recording file. In practice, such a traffic scenario recording file usually has two subfiles, namely *Traffic Description File* and *Map Description File*. *Traffic Description File* stores concrete values about the vehicle trajectories captured, and other necessary meta information including vehicle type, recording time, etc. Similar to the *Map Description File* of a simulation scenario (see §2.3), *Map Description File* of a traffic scenario also formally specifies the road networks and infrastructures.

Table 1: Specification Formats of Traffic Scenarios.

Source Dataset	Traffic Scenario Recording File	
	Traffic Description Format	Map Description Format
CommonRoad [37]	CommonRoad Scenario [10]	Lanelets [40]
inD [41]	inD CSV [19]	Lanelet2 [74]
highD [59]	highD CSV [17]	Lanelets [40] [1]

[1] highD itself does not provide map description files, but we can obtain them in a third-party repository [18].

Table 2: Specification Formats of Simulation Scenarios.

Target ADS or Simulator	Simulation Scenario File		
	Scenario Description Format	Map Description Format	Map Assets Format
Apollo [25]	—	Apollo HD Map [51]	—
Autoware [4]	—	Autoware Vector Map [5]	—
LGSVL [79]	VSE Scenario [23]	OpenDrive [1] or Lanelet2 [74]	LGSVL AssetBundle [22]
Carla [49]	OpenScenario [2]	OpenDrive [1]	Carla Map Meshes [7] (optional) [1]

[1] Carla features OpenDRIVE standalone mode which allows running a full simulation using only an OpenDRIVE map file, without any additional map assets. Therefore, we currently do not generate map assets for Carla.

3 OVERVIEW

In this section, we first introduce our overall idea for simulation scenario construction in §3.1. After that, in §3.2, we clarify the data sources of scenario construction and usage scope of curated scenarios. In §3.3, we present a running example of our approach.

3.1 Overall Idea

To build diverse and realistic simulation scenarios, an appealing idea is to digitalize real-world driving scenes in the simulation world. However, the challenging point lies in that, to build a simulation scenario, one should specify very low-level parameters (e.g., coordinates, width, length, speed, and so on) for each simulated object, agent, or scene. These parameters are difficult to accurately determine, even for human experts.

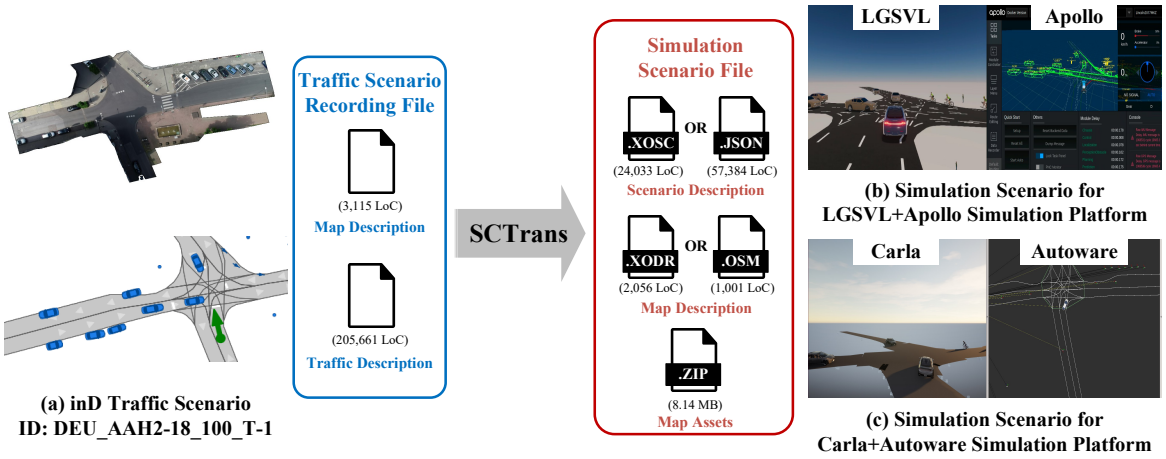
Key Insight. We observe that existing traffic scenario recording files (see §2.4) use various detailed and concrete values to describe real-world driving scenes (e.g., meta-information and moving trajectories of road users), which should be valuable data sources for parameterization of simulation scenario files. But the file formats of such data sources are usually self-defined by their authors and are not standardized in the field of ADS simulation testing. Hence, the ADS simulation platform cannot directly take them as inputs.

Transformation-based Approach: SCTrans. In light of this, we propose to achieve a format transformation, i.e., to transform traffic scenario recording files into simulation scenario files while preserving the scenario semantics. To ensure the correctness of the format transformation, we formalize this task as a Model Transformation Problem [67]. Specifically, we first create *Meta-Models* for both traffic scenario recording files and simulation scenario files, namely *source Meta-Model* and *target Meta-Model*. Here, a *Meta-Model* describes the abstract syntax of a given language, including its structure, elements, attributes, attribute constraints, etc. After manually teasing out the fine-grained mappings between the elements and attributes of *source Meta-Model* and *target Meta-Model*, we can then implement concrete *transformation rules* to automatically convert a traffic scenario recording file into a simulation scenario file. Our approach is named SCTrans.

3.2 Data Sources and Target Scope

In the following, we first clarify the data sources of SCTrans, and then the target scope (i.e., target simulators and ADSs) of the curated simulation scenario files. Finally, we summarize the specification formats of both source files and target files, providing guidance for the format transformation of scenario files.

Data Sources. Our overall idea is to take advantage of real-world traffic trajectories to parameterize simulation scenarios. Therefore, some datasets (e.g., CityScapes [47] and CamVid [42]) are



(a) inD Traffic Scenario ID: DEU_AAH2-18_100_T-1

(b) Simulation Scenario for LGSVL+Apollo Simulation Platform

(c) Simulation Scenario for Carla+Autoware Simulation Platform

Figure 2: Example of using SCTrans to transform a traffic scenario file from the inD dataset [41] into simulation scenario files that are compatible with the advanced LGSVL+Apollo and Carla+Autoware ADS simulation platforms.

not considered due to the lack of GPS trajectories of road participants. Finally, we chose three representative traffic scenario datasets as data sources, i.e., CommonRoad [37], inD [41] and highD [59]. The reasons are two-fold: Firstly, these datasets contain more than 13,000 concrete traffic scenario recording files, based on which we can build diverse simulation scenarios; Secondly, these datasets all provide detailed documents about their data formats, which are essential for precise file parsing.

Target ADSs. SCTrans aims to create simulation scenarios to test high-level ADSs. Recently, Apollo [25] and Autoware [4], two industry-grade ADSs, have emerged as frontrunners in the open-source community, and have been widely adopted for commercial usage. Beisdes, these two ADSs have been chosen as evaluation targets in various research works [44, 52, 53, 62, 83, 93]. Except for Apollo and Autoware, other ADSs are either of low-level automation (e.g., Level-2 OpenPilot [26]), or closed-source due to commercial restrictions (e.g., Waymo [35] and Uber [32]). Hence, we finally chose Apollo and Autoware as our target ADSs.

Target Simulators. A recent study [57] has extensively compared multiple simulators for ADS testing, from the perspectives of fidelity, flexibility, stability, and so on. Results show that Carla [49] and LGSVL [79] are the current state-of-the-art open-source simulators for the end-to-end testing of ADSs. In particular, Carla and LGSVL also provide user-friendly API interfaces which support stable connections to our target ADSs (i.e., Apollo and Autoware). Therefore, we chose Carla and LGSVL as our target simulators.

Source-to-Target Format Transformation. Table 1 summarizes the specification formats of selected traffic scenario recording files, and Table 2 illustrates the specification formats of simulation scenarios that are compatible with our target ADSs (i.e., Apollo and Autoware) and target simulators (i.e., Carla and LGSVL). In this work, we aim to curate well-formatted scenario files that satisfy these specification requirements. To be more specific, given a traffic scenario recording file from Commonroad [37], inD [41] or highD [59], we should accordingly construct 2 scenario description files (i.e., VSE Scenario [23] and OpenScenario [2]), 4 map description files (i.e., Apollo HD Map [51], Autoware Vector Map [5], OpenDrive [1] and Lanelet2 [74]) and 1 map assets file (i.e., LGSVL AssestBundle [22]), so as to flexibly support diverse simulation environments like LGSVL+Apollo and Carla+Autoware.

3.3 Running Example

Following the transformation-based idea, Figure 2 illustrates a running example of SCTrans. The generated simulation scenario files can be directly used by the most advanced ADS simulation platforms, without any further manual modifications.

4 APPROACH

SCTrans features a transformation-based approach, which converts traffic scenario recording files into ready-to-use simulation scenario files. In §4.1, we first introduce our design considerations for carrying out the transformation task. After that, we formulate the task as a Model Transformation Problem [67] in §4.2. Finally, §4.3 and §4.4 introduce how to perform model transformation in our problem domain.

4.1 Design Considerations

Decomposition of the Transformation Task. Conceptually, a traffic scenario recording file (Source for short, see §2.4) consists of 2 sub-files, namely *Map Description File* and *Traffic Description File*; a simulation scenario file (Target for short, see §2.3) consists of 3 sub-files, namely *Map Description File*, *Scenario Description File* and *Map Assets File*. In general, the transformation task from Source to Target can be divided into 3 sub-tasks: ❶ transforming the *Map Description File* of Source to the *Map Description File* of Target; ❷ transforming the *Map Description File* of Source to the *Map Assets File* of Target; ❸ transforming the *Traffic Description File* of Source to the *Scenario Description File* of Target.

Design Choice. Theoretically, to achieve the above 3 sub-tasks, SCTrans should support the transformation of each file format listed in Table 1 and Table 2 (i.e., more than 10 file formats in total), requiring extensive efforts. Here, fortunately, we find that

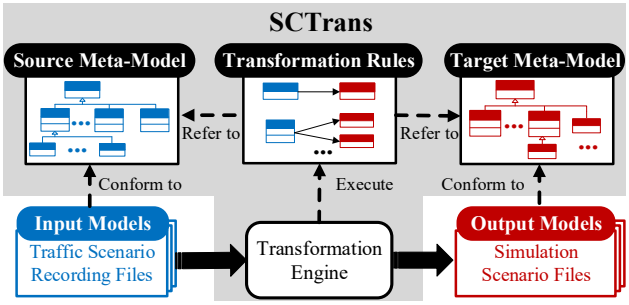


Figure 3: Architecture of SCTrans.

the community has provided various open-source tools [6, 38, 66] for converting formats of *Map Description File*, as well as a GUI tool [24] that assists the manual creation of *Map Assets File* based on given *Map Description File*. Therefore, we do not need to build transformation tools for these map-related files from scratch. But, in real practice, we find that existing tools are in an early stage of development, which frequently return error results. As detailed in §5, we invest non-trivial efforts to improve these tools (e.g., fix bugs and automate the creation of Map Assets), so as to ensure the reliability of SCTrans.

To sum up, in this work, we focus on transforming the *Traffic Description File* of Source into the *Scenario Description File* of Target. As summarized in Table 1 and Table 2, we mainly consider 3 formats for the *Traffic Description File* (i.e., CommonRoad Scenario [10], inD CSV [19] and highD CSV [17]), and 2 formats for the *Scenario Description File* (i.e., VSE Scenario [23] and OpenScenario [2]). Since open-source toolkits [14] are available to convert inD CSV [19] or high CSV [17] to CommonRoad Scenario [10], we finally target two types of format transformation tasks (i.e., to accomplish these tasks from scratch), including ❶ transforming CommonRoad Scenario [10] into OpenScenario [2]; ❷ transforming CommonRoad Scenario [10] into VSE Scenario [23].

4.2 Problem Formulation

Here, we first introduce the basic concepts of model transformation, and then the general architecture of SCTrans

Basic Concepts. A *Meta-Model* specifies the syntactic grammar of a language, mainly including its structures, elements, and relations between elements. A *Model* refers to a concrete instance conforming to a *Meta-Model*. The *Transformation Rules* determine how a given *Model* that conforms to *Source Meta-Model* can be converted into a new *Model* that conforms to *Target Meta-Model*.

Architecture of SCTrans. Based on the aforementioned concepts, Figure 3 demonstrates the architecture of SCTrans. In §4.3 and §4.4, we will, respectively, introduce how to build *Meta-Models* and *Transformation Rules* in our problem domain.

4.3 Source/Target Meta-Models

Symbol Definition. The *Meta-Model* $\mathbb{MM} := (\mathbb{MC}, \mathbb{MA})$ describes the format syntax, based on a set of meta-classes $\mathbb{MC}$ and a set of meta-associations $\mathbb{MA}$. Each meta-class $mc \in \mathbb{MC}$ has a set of data attributes $A(mc)$. Furthermore, each attribute $a_i^{mc}(id, T, R) \in A(mc)$ is specified with the identifier name id, the attribute type T, and the attribute sanity R (i.e., valid value range). Each meta-association $ma(mc_i, mc_j, Q) \in \mathbb{MA}$ indicates that mc_i associates with mc_j with association type Q (e.g., one-to-many reference).

Overview. Among the three considered formats of scenario description files (see §4.1), CommonRoad Scenario [10] and OpenScenario [2] all provide scheme files (i.e., XSD files) to precisely describe the language syntax. These scheme files can be automatically converted into semantically equivalent meta-model files using the model transformation framework [1] (see §5). Hence, we only need to manually construct the meta-model for VSE Scenario [23] from scratch. Our key observation is that a scenario description language, if lacking a scheme file to define its syntax, should necessarily come with a hand-written serialization/deserialization program to ensure its usability. Such a program should contain implicit meta-models in its object-oriented design. In the following, we introduce the formalized steps to manually construct meta-classes and meta-associations from the serialization/deserialization program [2] of VSE Scenario [23] (written in C#).

Meta-class Construction. We first enumerate all data objects $\mathbb{O} := \{o_1, o_2, ..., o_n\}$ that have a chance to be accessed (i.e., read-/write) during the scenario serialization/deserialization. Given each data object $o \in \mathbb{O}$, we accordingly construct a meta-class mc_o. After that, the set of data attributes A_{mc_o} is constructed by collecting serializable/deserializable data fields of o. Specifically, to construct a data attribute $a(id, T, R) \in A(mc_o)$ from a given field, the field name is set as $a.id$, the field property (e.g., data type, default value, required or not, etc) is set as $a.type$, and $a.R$ is set by merging its value constraints implemented in the initialization/serialization/deserialization functions (i.e., from if-checks, loop-conditions, assertions, etc).

Meta-association Establishment. Given an object $o \in \mathbb{O}$, each of its object-typed fields (the object type is denoted as o_f) uniquely represents a meta-association (m_o, m_{o_f}, Q) between meta-classes m_o and m_{o_f}. Specifically, under the model transformation framework [82], the association type Q is usually specified by the lower-bound and upper-bound of object reference times. For instance, if both lower-bound and upper-bound are set to 1, it represents a one-to-one reference relation between meta-classes.

Running Example. For ease of understanding, Figure 4 illustrates a running example of meta-model construction for VSE Scenario [23]. According to the deserialization program shown in Figure 4(a), we can accordingly build 3 meta-classes and their meta-associations. Besides, for each data attribute, we carefully collect its name and sanity. Finally, Figure 4(b) and Figure 4(c) formally describe the constructed meta-classes and meta-associations, respectively based on the UML class diagram (graph-based representation) and the BNF formulas (textual representation).

4.4 Source-to-Target Transformation

Symbol Definition. The model transformation $\mathbb{MT} : \mathbb{MM}_s \rightarrow \mathbb{MM}_t$ consists of a finite set of transformation rules $\mathbb{F}_{s \rightarrow t}$. Each transformation rule $f \in \mathbb{F}_{s \rightarrow t}$ can be denoted as a three-tuple

[1] https://www.eclipse.org/modeling/emf/docs/overviews/XMLSchemaToEcoreMapping.pdf
[2] https://github.com/lgsvl/simulator/tree/release-2021.1/Assets/Scripts/ScenarioEditor/Data

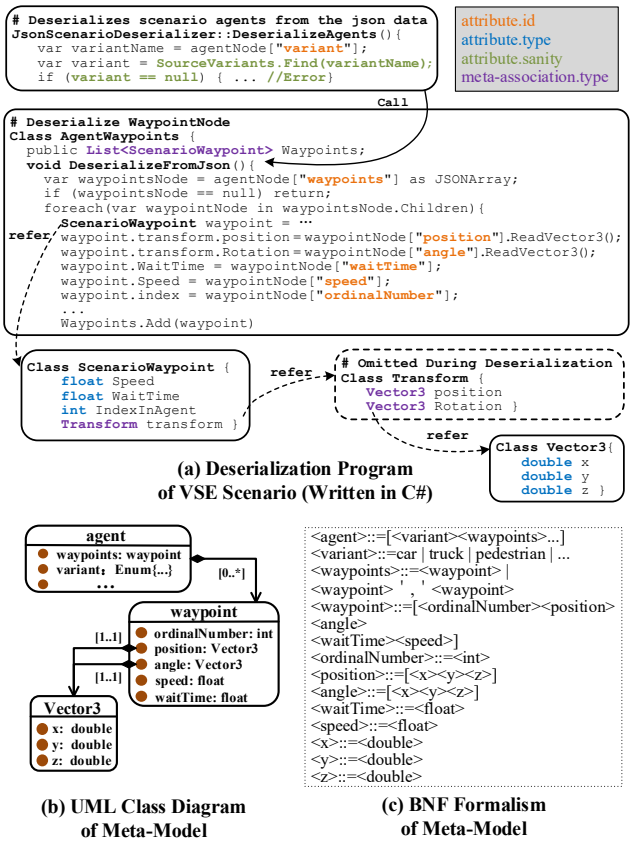


Figure 4: Example of Meta-model Construction (for VSE Scenario [23]).

$(\mathbb{MC}_s^f, \mathbb{MC}_t^f, \Phi_{s\to t}^f)$, $\mathbb{MC}_s^f \subset \mathbb{MM}_s.\mathbb{MC}$, $\mathbb{MC}_t^f \subset \mathbb{MM}_t.\mathbb{MC}$. Besides, $\Phi_{s\to t}^f$ denotes the source-to-target attribute mapping between $\mathbb{MC}_s^f$ and $\mathbb{MC}_t^f$, i.e., $\Phi_{s\to t}^f : \{a_s(id,T,R)|mc \in \mathbb{MC}_s^f, a_s \in A(mc)\} \to \{a_t(id,T,R)|mc \in \mathbb{MC}_t^f, a_t \in A(mc)\}$, which also implies the matching relations between the attribute sanity $a_s.R$ and $a_t.R$ (i.e., valid value range).

Overview. In this work, we aim to manually construct two sets of transformation rules, respectively for ❶ transforming CommonRoad Scenario [37] into OpenScenario [2], and ❷ transforming CommonRoad Scenario [37] into VSE Scenario [23]. Given a source meta-model and a target meta-model, we first match their meta-classes. After that, we further match and transform data attributes among each matched pair of meta-classes. Details are as follows.

Meta-class Matching. The meta-class matching aims to identify all possible semantic correspondences between meta-classes. To practically reduce the matching scope of meta-classes, we first classify meta-classes into 3 general categories (i.e., Config-env, Config-tfc and Config-ego, see §2.3), and only consider matching meta-classes within the same category. After that, given a set of source meta-classes $\mathbb{MC}_s$ and a set of target meta-classes $\mathbb{MC}_t$, we utilize a matrix $P_{mc} \in \{0,1\}^{|\mathbb{MC}_s|\times|\mathbb{MC}_t|}$ to record their pairwise semantic correspondences, where $P_{mc}[i,j] = 1$ only if the meta-class $mc_s^i \in \mathbb{MC}_s$ has shared semantics with the meta-class $mc_t^j \in \mathbb{MC}_t$ (otherwise, $P_{mc}[i,j] = 0$). To be specific, the semantics of a meta-class mc, denoted as $sem(mc)$, which are featured by its class name, class descriptions in the document (if available), its attribute names, and corresponding attribute descriptions in the document (if available). Based on the matrix P_{mc}, we can obtain all matched pairs of meta-classes $(\{mc_s\},\{mc_t\})$, s.t., $\forall mc_s^i \in \{mc_s\}, \exists mc_t^j \in \{mc_t\}, P_{mc}[i,j] = 1$. Obviously, the matched pair $(\{mc_s\},\{mc_t\})$ forms a $|\{mc_s\}|$-to-$|\{mc_t\}|$ matching relation, which could be one-to-one, one-to-many or many-to-many.

Attribute Matching and Transformation. Given a matched pair of meta-classes $(\{mc_s\},\{mc_t\})$, we denote all their non-object attributes as A_s and A_t respectively. Similar to the way mentioned above, we utilize another matrix $P_{attr} \in \{0,1\}^{|A_s|\times|A_t|}$ to enumerate the pairwise semantic correspondences between their data attributes. Based on the matrix P_{attr}, we then extract all matched pairs of data attributes $(\{a_s\},\{a_t\})$, which could be one-to-one, one-to-many or many-to-many.

Given a matched pair of attributes $(\{a_s\},\{a_t\})$, we should correlate their semantic-equivalent attribute values to achieve attribute transformation, i.e., $a_s^1.R \times a_s^2.R \times \ldots \times a_s^n.R \to a_t^1.R \times a_t^2.R \times \ldots \times a_t^m.R$. After transforming all the matched pairs of attributes of $(\{mc_s\},\{mc_t\})$, we can finally get a transformation rule $f := (\{mc_s\},\{mc_t\},\Phi_{s\to t}^f)$ (i.e., the symbol $\Phi_{s\to t}^f$ is defined at the beginning of this subsection). However, since the source meta-model $\mathbb{MM}_s$ and the target meta-model $\mathbb{MM}_t$ naturally express different semantics, it may require additional semantic completion or mutation to ensure the completeness of transformation. ❶ *Semantic Mutation.* For each mapping between semantically equivalent attribute values from $\Phi_{s\to t}^f$, denoted as $g := a_s.R \to a_t.R$, if there exists a source attribute value $v \in a_s.R$ such that $g(v) = NULL$, we apply the semantic mutation to update g to $g \cup \{v \to x\}$ where $x \in a_t.R$ shares the closest semantic with v. For example, CommonRoad Scenario [10] allows specifying the agent type as `Motocycle`, which is not allowed by OpenScenario [2]. During the transformation, we would intentionally mutate it to `Bicycle` (i.e., the most similar agent type allowed by OpenScenario [2]). ❷ *Semantic Completion.* If there exists a required target attribute a_t, such that for any $a_s \in \{a|mc \in \{mc_s\}, a \in A(mc)\}$, $\Phi_{s\to t}^f(a_s) \neq a_t$, we set $a.t$ to a default/random value $x \in a_t.R$ (i.e., x is within the valid value range of $a.t$). For example, VSE Scenario [23] requires an attribute to describe the weather, which, however, is not supported by CommonRoad Scenario [10]. During the transformation, we randomly set the weather conditions (e.g., sunny, foggy, etc).

Running Example. For ease of understanding, Figure 5 illustrates a running example of transform rule construction. Following the 3-step procedure, we can precisely convert the source meta-classes {state, position, point} into the target meta-classes {waypoint, angle, position}.

5 IMPLEMENTATION

Model Transformation Framework. The implementation of SCTrans is based on the Eclipse Modeling Framework (EMF) [82]. EMF is a well-established model transformation framework, offering a great set of tools or techniques, including the meta-modeling

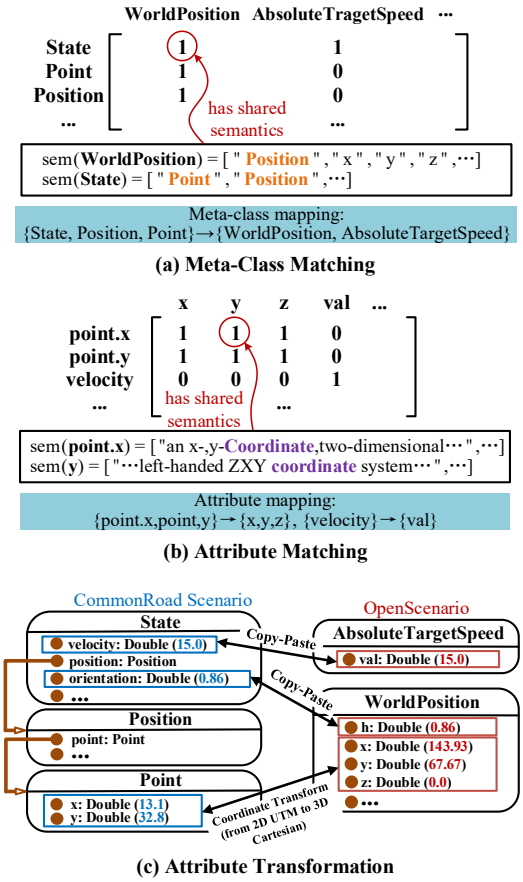


Figure 5: Example of Transformation Rule Construction (from CommonRoad Scenario [10] to OpenScenario [2]).

language Ecore to encode *Meta-Models*, the ATL transformation language to implement *Transformation Rules*, etc.

Meta-Models. With the help of EMF, we specify *Meta-Models* (see §4.3) for CommonRoad Scenario [10], OpenScenario [2] and VSE Scenario [23], with Ecore files that contain 2,898, 667 and 4,212 lines of code. To be more specific, CommonRoad Scenario [10] and OpenScenario [2] are XML-based description languages. They all provide XML schema files (i.e., XSD files), which define the structure and data elements of XML files. We follow the official documents [12] of EMF, to automatically convert these XSD files into Ecore files. Differently, VSE Scenario [23] is a JSON-based language specific to the LGSVL simulator [79]. We manually audit its parsing code [23] to create an Ecore file from scratch.

Transformation Rules. We implement two sets of transformation rules (see §4.4): one for the transformation from CommonRoad Scenario [10] to OpenScenario [2], and the other for the transformation from CommonRoad Scenario [10] to VSE Scenario [23]. These two sets of rules are implemented via ATL files that contain 272 and 642 lines of code, using several ATL-specific language mechanisms (e.g., rule-nesting[3]) to ensure the correctness of model transformation.

[3]https://wiki.eclipse.org/Henshin/Transformation_Meta-Model

Map Converter and Map Assets Generator. Similarly, we do not create *Meta-Models* for map-related files (i.e., *Map Description File* and *Map Assets File*) from scratch, since the community has provided various tools [6, 24, 38, 66] for converting map formats and generating map assets. However, existing tools [6, 38, 66] might produce incorrect or incomplete map transformation results due to implementation issues. We carefully fix these issues to ensure the correctness of conversion results. For instance, the CommonRoad Map Toolbox [66], which we used for converting maps formats from Lanelets [40] to Lanelet2 [74], would ignore the conversation of traffic lights and traffic signs. To address this issue, we add 290 lines of Python code to implement this desired functionality. For map assets generation, the existing tool [24] is only a GUI-based one that assists manual assets generation. We further leverage xdotool [36] to automate the process of GUI interaction, with a script file that contains 228 lines of code. As such, we can generate map asset files automatically in a scalable manner.

6 EVALUATION

In this section, we first elaborate on the experiment setup (§6.1) and report the general results (§6.2) of dataset transformation (e.g., syntactic/semantic correctness and time cost). For all curated simulation scenarios, we carried out extensive experiments to demonstrate their usability (§6.3), diversity (§6.4), and utility (§6.5).

6.1 Experiment Setup

Inputs of SCTrans. We considered 3 representative traffic scenario datasets as inputs of SCTrans, i.e., CommonRoad [37], inD [41] and highD [59]. These traffic scenario datasets are actively maintained and have indexed more than 1,3000 traffic scenario recording files. Here, to evaluate the prototype of SCTrans, we randomly selected 2,000 traffic scenario recording files as its inputs, marked as $Dataset_{TFC}$, with 1,000/2,000 from CommonRoad [37], 500/2,000 from inD [41] and 500/2,000 from highD [59]. These 2,000 concrete traffic scenarios are built upon 410 different maps.

ADS Simulation Platforms. We set up stable versions of the target ADSs and simulators (i.e., Apollo 6.0, Autoware 1.15+openplanner 2.5, LGSVL 2021.2.2 and Carla 0.9.13) on our server, so as to convincingly verify whether our curated simulation scenario files are compatible with advanced LGSVL+Apollo and Carla+Autoware simulation platforms. Besides, a simulator should rely on a scenario player (see Figure 1) to parse the simulation scenario files, and accordingly build the virtual world. However, we found that the scenario player of Carla 0.9.13 cannot correctly instantiate NPC vehicles with pre-defined trajectory waypoints. We fixed this issue with 374 lines of Python code to avoid biased evaluation results.

Server Environment. All our experiments were conducted on a Ubuntu 18.04 server with 64 GB memory, 32 CPU cores (Intel(R) Xeon(R) Silver 4215R) running at 3.20 GHz, and a single NVIDIA GeForce RTX 3070 GPU. The GPU is utilized to support the runtime computation of ADSs, and also the 3D rendering of simulators.

6.2 General Transformation Results

Experiment Design. According to the specification formats (see Table 2) of our two target simulation platforms, we should utilize

Table 3: General Transformation Results of SCTrans.

Inputs ($Dataset_{TFC}$)	Syntactic Accuracy	Semantic Perseverance	Avg. Time Cost	
			SF [1]	MF [2]
CommonRoad [37]	994/994 (100%)	99.77%	2.3s	97.9s
inD [41]	500/500 (100%)	99.88%	3.2s	107.4s
highD [59]	500/500 (100%)	99.90%	2.2s	86.7s

[1] **SF**: Scenario Description File;
[2] **MF**: Map Description File and Map Assets File;

SCTrans to convert each input from $Dataset_{TFC}$ into 2 scenario description files (i.e., VSE Scenario [23] and OpenScenario [2]), 4 map description files (i.e., Apollo HD Map [51], Autoware Vector Map [5], OpenDrive [1] and Lanelet2 [74]) and 1 map assets file (i.e., LGSVL AssestBundle [22]). Here, we mainly evaluate the following aspects of scenario transformation:

- *Syntactic Accuracy*, i.e., whether the output simulation scenarios are syntactically correct. Here, we use EMF Validation Tool [15] to automatically check whether each generated simulation scenario file conforms to its corresponding *Meta-Model*.
- *Semantic Perseverance*, i.e., whether the input traffic scenarios and the output simulation scenarios express the same semantics. Specifically, we count the percentage of data attributes that go through semantic mutation or completion, i.e., $1 - \frac{|\{Mutated\ Attributes\}| + |\{Completed\ Attributes\}|}{|\{All\ Target\ Attributes\}|}$, so as to quantify the semantic deviations.
- *Time Cost*, i.e., how much time is required to generate the desired simulation scenario files and map description files.

Experiment Results. SCTrans, as a model-transformation approach, can work properly only when the given inputs have correct syntax. Hence, we first utilized the EMF Validation Tool [15] to check the syntactic correctness of input traffic scenarios. Results show that 6 CommonRoad scenario files have format errors (e.g., incomplete files or duplicated data elements). Thus, we chose the remaining 1,994/2,000 traffic scenarios as inputs of SCTrans.

As shown in Table 3, SCTrans has 100%(=1,994/1,994) syntactic accuracy when transforming traffic scenario recording files into simulation scenario files. That is, all the data attributes in simulation scenarios satisfy the constraints (e.g., value sanity and name specification) encoded in the meta-model, indicating the reliability of our constructed transformation rules.

In total, the generated simulation files (i.e., 1,994 VSE Scenario [23] files and 1,994 OpenScenario [2] files) have 40,967,668 data attributes. To ensure the syntactic correctness of generated scenario files, only 4,119 data attributes are semantically mutated and 18,0323 data attributes are semantically completed, i.e., keep over 99.7% semantics of the input scenario files. This result indicates the high realism of our curated simulation scenarios.

Table 3 also shows that, given a traffic scenario recording file, it cost about 99.9s to generate all needed files for our target ADS simulation platforms. The generation of map-related files costs most of the time, mainly due to the heavy-weight 3D rendering process for map assets preparation.

Table 4: Usability Results of Curated Simulation Scenarios.

Simulation Platform	Scenario Format	# of failed runs Caused by			Scenario Usable Rate
		Simulator Flaws	ADS Flaws	SCTrans Flaws	
LGSVL 2021.2.2 + Apollo 6.0	**VSE Scenario**	0	59	0	1,994/1,994 (100%)
Carla 0.9.13 + Autoware 1.15 (openplanner 2.5)	**OpenScenario**	59	600	0	1,994/1,994 (100%)

6.3 Usability of Curated Simulation Scenarios

Experiment Design. As mentioned above, we successfully curated 1,994 simulation scenarios. We further conducted experiments to verify the usability of these scenarios, i.e., whether state-of-the-art ADS simulation platforms can indeed take them as inputs to run testing. Here, we chose LGSVL+Apollo and Carla+Autoware as our target ADS simulation platforms, which respectively take VSE Scenario [23] files and OpenScenario [2] files as inputs.

Intuitively, the bidirectional communication (see §2.2) between the simulator and the ADS can be considered as the key evidence of one successful run of ADS simulation testing (i.e., the given simulation scenario file is usable). But note that, except for the format errors of simulation scenario files, flaws or incapabilities of ADSs or simulators would also cause failed runs of simulation testing (e.g., ADS crashes caused by memory bugs). To avoid biased results when evaluating the scenario validity, we inserted logging statements to simulators and ADSs to collect necessary runtime information, based on which we can determine whether a failed run is caused by the format errors of the simulation scenario file or not. To be more specific, advanced ADSs or simulators all provide high-level APIs to load and parse scenarios or maps (e.g., `lgsvl.Simulator.load()`). We can monitor the runtime status of such APIs to verify whether our curated scenario files are usable. If so, we further analyze runtime error logs to confirm whether a failed run originates from ADS flaws or simulator flaws.

Experiment Results. In total, it cost 68.2 CPU hours to dynamically run all of the 1,994 simulation scenarios on our two target simulation platforms. As shown in Table 4, the simulation scenario files generated by SCTrans have a 100% usable rate on these state-of-the-art ADS simulation platforms. This result is consistent with the 100% syntactic accuracy of our curated scenarios (i.e., reported in §6.2). Thus, we believe our dataset can drive practical value for the safety testing of these two advanced high-level ADSs.

It is also surprising that Carla, Apollo, and Autoware, even emerging as front-runners in the open-source community, still have various implementation flaws that cause 59+59+600 failed runs during this experiment. Here, we also invested extensive efforts to confirm these failed runs are indeed caused by flaws of simulators and ADSs, rather than those of SCTrans (detailed as follows).

- *59 failed runs on LGSVL+Apollo.* After careful diagnosis, we found that these failed runs are all caused by unsynchronized data between LGSVL and Apollo. Specifically, LGSVL would intentionally standardize the path of a given map (i.e., modify the folder name) without synchronizing with the map loader of Apollo. Consequently, Apollo cannot correctly load necessary maps.
- *59+600 failed runs on Carla+Autoware.* Through manual case studies, we found that these failed runs are either caused by the

Table 5: Diversity Results of Curated Simulation Scenarios.

Dataset	# of Scenarios	Scenario Format	Diversity Measures	
			Label Coverage	Vendi Score
Esmini [16]	41	OpenX [2]	55.4% (+23.4%)	13.3 (×2.47)
OSC-ALKS [27]	15	OpenX [2]	21.4% (+57.1%)	2.86 (×11.5)
Carla [8]	9	OpenX [2]	33.9% (+44.6%)	5.15 (×6.39)
D. Karunakaran [56]	9	OpenX [2]	14.3% (+64.2%)	2.03 (×16.2)
Ours	1,994	OpenX [2] VSE [23]	**78.5%**	**32.9**

rendering errors of the simulator, or the functional insufficiencies of the ADS (e.g., cannot plan the trajectory in complicated driving conditions). To confirm this, we chose LGSVL+Apollo as a baseline and fed the 59+600 scenario files to it. After 6.9 CPU hours of testing, we confirmed the success of all the 59+600 runs. This evidence supports the view that the 59+600 failed runs are not caused by incorrect scenario files.

6.4 Diversity of Curated Simulation Scenarios

Experiment Design. The 1,994 usable simulation scenarios make up the initial version of our public scenario dataset. Here, we carried out experiments to demonstrate the diversity of our dataset and also made comparisons with existing simulation scenario datasets.

- *Existing Datasets.* To conduct this experiment, we tried our best to collect open-source simulation scenario datasets from either research works or third-party repositories. Notably, we only considered datasets whose file formats are compatible with the most advanced ADS simulation platforms (e.g., LGSVL+Apollo and Carla+Autoware). During the data collection process, we found that the public availability of simulation scenarios is quite limited (shown in Table 5). Among existing works [46, 55, 56, 69, 73, 85] that aim to construct ready-to-use simulation scenario files, only D. Karunakaran *et al.* [56] released 9 scenario files to the community. Similarly, the number of simulation scenario files in public repositories is also limited.
- *Diversity Measures.* We mainly consider two diversity measures. ❶ *Label Coverage*: ISO 21448 [20] (guiding principles for ADS testing) suggests 13 categories of scenario factors (e.g., road shapes), consisting of 119 concrete scenario labels. Specifically, 56/119 labels can be used to specify simulation scenarios [4] (i.e., can be expressed by standard scenario DSLs and can be supported by our target simulators), while the others are commonly for physical scenarios. *Label Coverage* measures how many of these 56 labels can be covered. ❷ *Vendi Score* [50] is the SOTA metric to quantify the effective number of unique or dissimilar samples in a dataset. The calculation of *Vendi Score* requires a pairwise sample similarity function, which is set as the conventional Jaccard similarity between scenario labels in our problem domain.

Experiment Results. To compare our dataset with existing ones in diversity, we first spent 17.3 manual hours labeling all these simulation scenarios, considering a complete set of 56 simulation-capable scenario labels suggested by ISO 21448 [20]. After scenario labeling, we can then accordingly calculate the diversity measures. Table 5 presents the diversity measures of different datasets. Results show that our dataset significantly outperforms existing ones, either in *Label Coverage* or *Vendi Score* [50].

[4] We list these labels at https://seclab-fudan.github.io/SCTrans/.

Table 6: ADS Fuzzing Results with Different Seeds.

Target ADS	Results (# of Collisions / # of Unqiue&Reproducible Collisions) [1]								
	AV-Fuzzer [63]			AutoFuzz [93]					Ours
	S1	S2	Avg.	S3	S4	S5	S6	Avg.	Avg.
Apollo 6.0	15/2	38/1	26.5/1.5	21/1	5/0	24/0	21/0	17.8/0.25	86.7/3.7
Autoware 1.15 (openplanner 2.5)	0/0	1/0	0.5/0	42/0	47/2	17/0	0/0	26.5/0.5	84.6/3.5

[1] **Unique Collisions**: Enhance existing heuristics [93] to filter similar collision scenarios.
[2] **Reproducible Collisions**: Ignore ego collisions that can not be stably reproduced.

6.5 Utility of Curated Simulation Scenarios

Experiment Design. In previous experiments (see §6.3 and §6.4), we have demonstrated that our curated simulation scenarios have high usability and diversity. To further showcase their utility, we fed them as inputs to advanced simulation-based ADS testing tools, to see whether the effectiveness of testing can be improved. Finally, we chose simulation-based ADS fuzzers [52, 53, 58, 62, 93] as our evaluation targets. This is because we observe that, without publicly available simulation scenario files, these fuzzers commonly use simple hand-crafted scenarios as seed inputs. But unfortunately, it has long been proved [78, 95] that low-quality seed inputs would largely hurt the fuzzing performance. Hence, we are highly motivated to investigate whether our curated simulation scenarios can boost the collision-finding capabilities of ADS fuzzers.

We randomly selected 50 simulation scenarios from our dataset as seeds to be evaluated, and we collected 6 open-source seed scenarios utilized in existing ADS fuzzers (i.e., 2 seeds from AV-Fuzzer [63] and 4 seeds from AutoFuzz [93]) as comparison targets. Specifically, inspired by existing practice [78], we ran ADS fuzzer each time with only one seed scenario for 3 hours and deactivated all runtime seed scheduling mechanisms. After finishing 50+6 fuzzing tasks, we can faithfully compare the collision-finding capability of each seed. We implemented the fuzzing framework on LGSVL 2021.2.2 with over 4,000 lines of Python code, according to the mutation strategies of DriveFuzz [58]. The ADSs under test are Apollo 6.0 and Autoware 1.15 (with openplanner 2.5).

Experiment Results. After 336 CPU hours (=(50+6)*3*2) of fuzzing, Table 6 summarizes the collision-finding capability of each seed scenario. Results show that our curated simulation scenarios can significantly boost the collision-finding capability of advanced simulation-based ADS fuzzer. Compared to simple hand-crafted seed scenarios utilized in existing works [63, 93], our simulation scenarios help identify over 3.2 times more ego-involved collisions, and over 7.0 times more unique ego-involved collisions within the same time period (3h). The key reason lies in that, our simulation scenarios are digitalization of real-world driving scenes, which capture complex and realistic moving behaviors of various traffic agents. These surrounding agents pose great challenges to the robustness of ADSs under test. Hence, utilizing such simulation scenarios as seed inputs, ADS fuzzers can quickly identify ego-involved accidents with fewer mutation efforts. As summarized in Table 7, after carefully analyzing all ego-involved unique collisions, we identified 4 bugs of Apollo 6.0 and 5 bugs of Autoware 1.15 (openplanner 2.5), each of which can be stably exploited to cause

Table 7: Identified Bugs on Apollo and Autoware.

ADS	ID	Module	Bug Description	Confirmed
Apollo	R1	Planning	Imprecise calculation of drivable areas	
	R2	Planning	Insufficient braking force in emergencies	
	R3	Prediction	Incorrect prediction of obstacle trajectories at intersections	
	R4	Planning	Fail to avoid rear-end vehicles	
Autoware	R5	Control	Keep removing when destination reached	✓
	R6	Planning	Fail to avoid rear-end vehicles	
	R7	Planning	Follow driving paths that may run off the road	✓
	R8	Planning	Incorrect prediction of collision points	✓
	R9	Control	Fail to handle sharp turns	✓

traffic accidents. After submitting the bug reports (including bug locations and root causes) to the developers of Apollo and Autoware, till now, 4 of them have been confirmed. Through this experiment, we demonstrate that our curated simulation scenarios have large practical value for the safety assessment of high-level ADSs.

7 DISCUSSION

Portability of Our Dataset. At present, the OpenX standards (e.g., OpenScenario [2] and OpenDrive [1]) are the most widely used specification formats in the field of ADS simulation testing. According to a recent survey [94], popular commercial driving simulators (e.g., CarMaker [9] and PreScan [29]) can also take OpenX [1, 2] files as inputs. Hence, we believe that our curated simulation scenario files can be smoothly migrated to other ADS simulation platforms.

Need for Diverse Real-world Data. As a transformation-based approach to curate simulation scenarios, SCTrans is highly dependent on the diversity of the input real-world data. To curate more realistic simulation scenarios with SCTrans, a practical solution is to involve new data sources (i.e., other traffic scenario datasets [45, 60, 68, 75, 96] which record concrete vehicle trajectories). We plan to explore the possibility of transforming these data sources into simulation scenarios in the future.

Limitations. Though the experiment results show that SCTrans can effectively transform traffic scenario recording files into ready-to-use simulation scenario files, it still has several limitations: ❶ Generally, the construction of meta-models and transformation rules requires non-trivial manual efforts, which might hinder the direct application of SCTrans on other data sources or target scenario formats. However, we believe that our formalized approach can offer practical guidance for different scenario transformation tasks, or inspire the design of automated transformation tools. ❷ As shown in Figure 2, our curated simulation scenarios only have road meshes for traffic actors to navigate through. Other additional assets such as buildings or vegetation will not be created. This is mainly because our data sources (i.e., traffic scenario datasets) do not describe such information. Although this issue would to some extent hurt the realism of curated simulation scenarios, however, as proved in §6.5, our scenario dataset can still help convincingly identify safety-critical ADS flaws.

8 RELATED WORKS

Data-driven Simulation Scenario Construction. After a systematic literature review, we find that some existing works [55, 56, 69, 73, 85] exploit a similar idea with SCTrans, i.e., to transform real-world traffic recording files into ready-to-use simulation scenario files. However, most of existing works [56, 69, 73] take vehicle-side raw sensor data (e.g., LiDAR data recorded by a sensor-equipped vehicle on public roads) as inputs. Raw sensor data is usually noisy, thus cannot provide a precise estimate of the surroundings. AC3R [55] elaborates police reports as data sources, which are high-level abstraction forms of driving scenarios. It heavily depends on expert knowledge to specify low-level scenario configurations. Besides, existing works [56, 69, 73, 85] commonly leverage a maneuver-based methodology to build simulation scenarios. They rely on heuristic-based rules to identify maneuvers of surrounding vehicles (e.g., accelerate, cut-in, etc) from real-world data, and then randomly specify low-level scenario configurations. However, heuristic-based maneuver identification cannot precisely digitalize complex driving conditions. Due to these limitations, it is quite hard to leverage existing works to generate diverse and realistic simulation scenarios. Besides, among all these works, only [56] releases 9 usable simulation scenario files to the community.

Mutation-based Simulation Scenario Generation. Some recent works [52, 53, 58, 62, 93] try to leverage mutation-based techniques to automatically build previously unknown simulation scenarios for ADS testing. These techniques work by mutating the configurations of initial scenario files to generate new ones (e.g., adding a new traffic actor). As demonstrated in §6.5, SCTrans is orthogonal to this line of research, having the capability to prepare high-quality initial inputs for ADS fuzzers.

9 CONCLUSION

In this work, we propose SCTrans to construct diverse simulation scenario files, which can be directly used to test state-of-the-art ADSs (i.e., Apollo and Autoware) with high-fidelity driving simulators (i.e., LGSVL and Carla). The key idea of SCTrans is to transform the file formats of existing traffic scenario datasets (i.e., datasets that record the naturalistic movement of road users) into the ones that can be accepted by popular ADSs and simulators. With the help of SCTrans, we construct and release a large public scenario dataset, consisting of over 1900 diverse ready-to-use simulation scenario files. We also conducted extensive experiments to demonstrate the utility of our dataset. Results show that it can significantly boost the collision-finding capability of existing ADS fuzzers, and help identify 9 safety-critical bugs of Apollo and Autoware. Till now, 4 of them have been confirmed.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200), National Natural Science Foundation of China (62172105,62172104,62102091, 62102093) and the Funding of Ministry of Industry and Information Technology of the People's Republic of China under Grant TC220H079. Yuan Zhang was supported in part by the Shanghai Rising-Star Program 210A1400700 and the Shanghai Pilot Program for Basic Research-Fudan University 21TO1400100 (21TQ012). Min Yang is the corresponding author, and a faculty of Shanghai Institute of Intelligent Electronics & Systems and Engineering Research Center of Cyber Security Auditing and Monitoring.

REFERENCES

[1] ASAM OpenDRIVE. https://www.asam.net/standards/detail/opendrive, 2023.
[2] ASAM OpenSCENARIO. https://www.asam.net/standards/detail/openscenario, 2023.
[3] Autonomous Vehicle Collision Reports. https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/autonomous-vehicle-collision-reports/, 2023.
[4] Autoware-AI. https://github.com/autowarefoundation/autoware, 2023.
[5] Autoware Vector Map. https://github.com/autowarefoundation/autoware_common/blob/main/tmp/lanelet2_extension/docs/lanelet2_format_extension.md, 2023.
[6] Carla Map Converter. https://gitlab.com/autowarefoundation/autoware.ai/utilities/-/tree/master/lanelet_aisan_converter, 2023.
[7] Carla Map Meshes. https://carla.readthedocs.io/en/0.9.7/how_to_make_a_new_map/, 2023.
[8] Carla Scenario Runner. https://github.com/carla-simulator/scenario_runner, 2023.
[9] CarMaker. https://ipg-automotive.com/en/products-solutions/software/carmaker/, 2023.
[10] Commonroad XML. https://gitlab.lrz.de/tum-cps/commonroad-scenarios/-/blob/master/documentation/XML_commonRoad_2020a.pdf, 2023.
[11] Companies that Develop Autonomous Driving. https://aimagazine.com/technology/top-10-companies-developing-autonomous-vehicle-technology, 2023.
[12] Convert XML Schema into Ecore Meta-Model. https://www.eclipse.org/modeling/emf/docs/1.x/tutorials/xlibmod/xlibmod_emf1.1.html, 2023.
[13] Cyber-RT. https://cyber-rt.readthedocs.io/en/latest/, 2023.
[14] Dataset Converter. https://commonroad.in.tum.de/tools/dataset-converters, 2023.
[15] EMF Validation. https://www.eclipse.org/emf-validation, 2023.
[16] Esmini. https://github.com/esmini/esmini, 2023.
[17] HighD CSV. https://www.highd-dataset.com/format, 2023.
[18] HighD Map. https://gitlab.lrz.de/tum-cps/dataset-converters/-/blob/master/src/highD/map_utils.py, 2023.
[19] InD CSV. https://www.ind-dataset.com/format, 2023.
[20] ISO 21448. https://www.iso.org/standard/77490.html, 2023.
[21] Levelxdata. https://levelxdata.com/, 2023.
[22] LGSVL Map AssetBundle. https://www.svlsimulator.com/docs/user-interface/web/library/#maps, 2023.
[23] LGSVL VSE Scenario. https://www.svlsimulator.com/docs/visual-scenario-editor/vse-inspector/, 2023.
[24] Map Asset Generation Tool. https://www.svlsimulator.com/docs/archive/2020.06/unity-help/, 2023.
[25] Open-sourced Version of Baidu Apollo. https://github.com/ApolloAuto/apollo, 2023.
[26] OpenPilot. https://comma.ai/openpilot/, 2023.
[27] OSC-ASKS. https://github.com/asam-oss/OSC-ALKS-scenarios, 2023.
[28] OSM. :https://www.openstreetmap.org, 2023.
[29] PreScan. https://m.tass.plm.automation.siemens.com/cn/prescan-2, 2023.
[30] ROS. https://www.ros.org/, 2023.
[31] Safety Pool. https://www.safetypool.ai/, 2023.
[32] Uber. https://www.uber.com/, 2023.
[33] Unity Engine. https://unity.com/, 2023.
[34] Unreal Engine. https://www.unrealengine.com/, 2023.
[35] Waymo. https://waymo.com/, 2023.
[36] XdoTool. https://github.com/jordansissel/xdotool, 2023.
[37] M. Althoff, M. Koschi, and S. Manzinger. CommonRoad: Composable Benchmarks for Motion Planning on Roads. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2017.
[38] M. Althoff, S. Urban, and M. Koschi. Automatic Conversion of Road Networks from Opendrive to Lanelets. In *Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2018.
[39] S. Baltodano, S. Sibi, N. Martelaro, N. Gowda, and W. Ju. The RRADS Platform: A Real Road Autonomous Driving Simulator. In *Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI)*, 2015.
[40] P. Bender, J. Ziegler, and C. Stiller. Lanelets: Efficient Map Representation for Autonomous Driving. In *Proceedings of the IEEE Intelligent Vehicles Symposium Proceedings*, 2014.
[41] J. Bock, R. Krajewski, T. Moers, S. Runde, L. Vater, and L. Eckstein. The Ind Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020.
[42] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic Object Classes in Video: A High-definition Ground Truth Database. *Pattern Recognition Letters*, 2009.
[43] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. Nuscenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[44] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. Li. Invisible for Both Camera and Lidar: Security of Multi-sensor Fusion Based Perception in Autonomous Driving under Physical-world Attacks. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (SP)*, 2021.
[45] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d Tracking And Forecasting with Rich Maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, 2019.
[46] H. Chen, H. Ren, R. Li, G. Yang, and S. Ma. Generating Autonomous Driving Test Scenarios Based on OpenSCENARIO. In *Proceedings of the 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022.
[47] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2016.
[48] E. De Gelder, J.-P. Paardekooper, A. K. Saberi, H. Elrofai, O. O. den Camp, S. Kraines, J. Ploeg, and B. De Schutter. Towards An Ontology for Scenario Definition for The Assessment of Automated Vehicles: An Object-Oriented Framework. *IEEE Transactions on Intelligent Vehicles*, 2022.
[49] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Conference on Robot Learning*, 2017.
[50] D. Friedman and A. B. Dieng. The Vendi Score: A Diversity Evaluation Metric for Machine Learning. *arXiv preprint arXiv:2210.02410*, 2022.
[51] C. W. Gran. HD-maps in Autonomous Driving. *M.S. thesis*, 2019.
[52] J. C. Han and Z. Q. Zhou. Metamorphic Fuzz Testing of Autonomous Vehicles. In *Proceedings of the 42nd IEEE/ACM International Conference on Software Engineering Workshops*, 2020.
[53] Z. Hu, S. Guo, Z. Zhong, and K. Li. Coverage-based Scene Fuzzing for Virtual Autonomous Driving Testing. *arXiv preprint arXiv:2106.00873*, 2021.
[54] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang. The ApolloScape Dataset for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018.
[55] T. Huynh, A. Gambi, and G. Fraser. AC3R: Automatically Reconstructing Car Crashes from Police Reports. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2019.
[56] D. Karunakaran, J. S. Berrio, S. Worrall, and E. Nebot. Automatic Lane Change Scenario Extraction and Generation of Scenarios in OpenX Format from Real-world Data. *arXiv preprint arXiv:2203.07521*, 2022.
[57] P. Kaur, S. Taghavi, Z. Tian, and W. Shi. A Survey on Simulators for Testing Self-driving Cars. In *Proceedings of the 4th International Conference on Connected and Autonomous Driving (MetroCAD)*, 2021.
[58] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim. DriveFuzz: Discovering Autonomous Driving Bugs through Driving Quality-Guided Fuzzing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.
[59] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The Highd Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.
[60] R. Krajewski, T. Moers, J. Bock, L. Vater, and L. Eckstein. The RounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany. In *Proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.
[61] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner. SUMO (Simulation of Urban MObility)-An Open-source Traffic Simulation. In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM)*, 2002.
[62] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer. AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems. In *Proceedings of the 31st IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2020.
[63] G. Li, Y. Li, S. Jha, T. Tsai, M. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. Iyer. AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems. In *Proceedings of the 31st IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2020.
[64] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang. 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.
[65] J. Ma, X. Che, Y. Li, and E. M.-K. Lai. Traffic Scenarios for Automated Vehicle Testing: A Review of Description Languages And Systems. *Machines*, 2021.
[66] S. Maierhofer, M. Klischat, and M. Althoff. Commonroad Scenario Designer: An Open-source Toolbox for Map Conversion and Scenario Creation for Autonomous Vehicles. In *Proceedings of the 24th IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021.
[67] T. Mens and P. Van Gorp. A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 2006.

[68] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki, and L. Eckstein. The ExiD Dataset: A Real-World Trajectory Dataset of Highly Interactive Highway Scenarios in Germany. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[69] F. Montanari, C. Stadler, J. Sichermann, R. German, and A. Djanatliev. Maneuver-based Resimulation of Driving Scenarios Based on Real Driving Data. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[70] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes. Drivetruth: Automated Autonomous Driving Dataset Generation for Security Applications. In *Proceedings of the 4th International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[71] S. of Automotive Engineers (SAE). J3016 - Taxonomy and Definitions for Terms Related to On-road Motor Vehicle Automated Driving Systems, 2016.

[72] A. Ouaknine, A. Newson, J. Rebut, F. Tupin, and P. Pérez. Carrada Dataset: Camera and Automotive Radar with Range-Angle-Doppler Annotations. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[73] S. Pathrudkar, S. Venkataraman, D. Kanade, A. Ajayan, P. Gupta, S. Khatib, V. S. Indla, and S. Mukherjee. SAFR-AV: Safety Analysis of Autonomous Vehicles Using Real World Data–An End-to-End Solution for Real World Data Driven Scenario-based Testing for Pre-certification of AV Stacks. *arXiv preprint arXiv:2302.14601*, 2023.

[74] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. Lanelet2: A High-Definition Map Framework for The Future of Automated Driving. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[75] V. Punzo, M. T. Borzacchiello, and B. Ciuffo. On The Assessment of Vehicle Trajectory Data Accuracy and Application to The Next Generation SIMulation (NGSIM) Program Data. *Transportation Research Part C: Emerging Technologies*, 2011.

[76] R. Queiroz, T. Berger, and K. Czarnecki. GeoScenario: An Open DSL for Autonomous Driving Scenario Representation. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[77] R. Rajamani. *Vehicle Dynamics and Control*. 2011.

[78] A. Rebert, S. K. Cha, T. Avgerinos, J. Foote, D. Warren, G. Grieco, and D. Brumley. Optimizing Seed Selection for Fuzzing. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*, 2014.

[79] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, et al. Lgsvl Simulator: A High Fidelity Simulator for Autonomous Driving. In *Proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[80] T. D. Son, A. Bhave, and H. Van der Auweraer. Simulation-based Testing Framework for Autonomous Driving Development. In *Proceedings of the IEEE International Conference on Mechatronics (ICM)*, 2019.

[81] D. Sportillo, A. Paljic, and L. Ojeda. On-road Evaluation of Autonomous Driving Training. In *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019.

[82] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro. *EMF: Eclipse Modeling Framework*. 2008.

[83] J. S. Sun, Y. C. Cao, Q. A. Chen, and Z. M. Mao. Towards Robust Lidar-based Perception in Autonomous Driving: General Black-Box Adversarial Sensor Attack and Countermeasures. In *Proceedings of the 29th USENIX Security Symposium (Usenix Security)*, 2020.

[84] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[85] A. Tenbrock, A. König, T. Keutgens, and H. Weber. The ConScenD Dataset: Concrete Scenarios from The HighD Dataset According to ALKS Regulation UNECE R157 in OpenX. In *Proceedings of the IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, 2021.

[86] X. Wang, A.-K. Rettinger, M. T. B. Waez, and M. Althoff. Coupling Apollo with The CommonRoad Motion Planning Framework. In *Proceedings of the FISITA Web Congress*, 2020.

[87] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. *arXiv preprint arXiv:2301.00493*, 2023.

[88] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, et al. Pandaset: Advanced Sensor Suite Dataset for Autonomous Driving. In *Proceedings of the 24th IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021.

[89] H. Yin and C. Berger. When to Use What Data Set for Your Self-driving Car Algorithm: An Overview of Publicly Available Driving Datasets. In *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[90] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[91] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access*, 2020.

[92] X. Zhao, V. Robu, D. Flynn, K. Salako, and L. Strigini. Assessing the Safety and Reliability of Autonomous Vehicles from Road Testing. In *Proceedings of the 30th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2019.

[93] Z. Zhong, G. Kaiser, and B. Ray. Neural Network Guided Evolutionary Fuzzing for Finding Traffic Violations of Autonomous Vehicles. *IEEE Transactions on Software Engineering (TSE)*, 2022.

[94] J. Zhou, Y. Zhang, S. Guo, and Y. Guo. A Survey on Autonomous Driving System Simulators. In *Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2022.

[95] X. Zhu, S. Wen, S. Camtepe, and Y. Xiang. Fuzzing: A Survey for Roadmap. *ACM Computing Surveys (CSUR)*, 2022.

[96] A. Zyner, S. Worrall, and E. M. Nebot. Acfr Five Roundabouts Dataset: Naturalistic Driving at Unsignalized Intersections. *IEEE Intelligent Transportation Systems Magazine*, 2019.