CHALMERS
UNIVERSITY OF TECHNOLOGY

A Driver-Vehicle Model for ADS Scenario-Based Testing

Citation for the original published paper (version of record):
Queiroz, R., Sharma, D., Diniz Caldas, R. et al (2024). A Driver-Vehicle Model for ADS Scenario-Based Testing. IEEE Transactions on Intelligent Transportation Systems, 25(8): 8641-8654. http://dx.doi.org/10.1109/TITS.2024.3373531

(article starts on next page)

A Driver-Vehicle Model for ADS Scenario-Based Testing

Rodrigo Queiroz, Divit Sharma, Ricardo Caldas, Krzysztof Czarnecki, Sergio García, Thorsten Berger, and Patrizio Pelliccione

***Abstract*— Scenario-based testing for automated driving systems (ADS) must be able to simulate traffic scenarios that rely on interactions with other vehicles. Although many languages for high-level scenario modelling have been proposed, they lack the features to precisely and reliably control the required micro-simulation, while also supporting behavior reuse and test reproducibility for a wide range of interactive scenarios. To fill this gap between scenario design and execution, we propose the Simulated Driver-Vehicle (SDV) model to represent and simulate vehicles as dynamic entities with their behavior being constrained by scenario design and goals set by testers. The model combines driver and vehicle as a single entity. It is based on human-like driving and the mechanical limitations of real vehicles for realistic simulation. The model leverages behavior trees to express high-level behaviors in terms of lower-level maneuvers, affording multiple driving styles and reuse. Furthermore, optimization-based maneuver planners guide the simulated vehicles towards the desired behavior. Our extensive evaluation shows the model's design effectiveness using NHTSA pre-crash scenarios, its motion realism in comparison to naturalistic urban traffic, and its scalability with traffic density. Finally, we show the applicability of our SDV model to test a real ADS and to identify crash scenarios, which are impractical to represent using predefined vehicle trajectories. The SDV model instances can be injected into existing simulation environments via co-simulation.**

***Index Terms*— Intelligent vehicles, autonomous vehicles, autonomous driving, system testing, simulation, road traffic.**

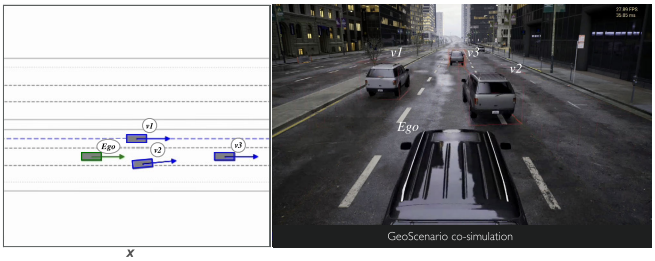

Fig. 1. A challenging interaction between ego and human-operated vehicles based on a pre-crash scenario from NHTSA [2] and using our SDV model in simulation: (left) in the map frame, the SDV Model as v_2 performs the cut-in maneuver targeting ego, and (right) a high-fidelity co-simulator renders the scene.

Manuscript received 4 May 2022; revised 14 June 2023 and 27 January 2024; accepted 11 February 2024. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada; in part by Renesas Electronics Corporation; in part by Japan Science and Technology Agency Exploratory Research for Advanced Technology (ERATO) Project HASUO Metamathematics for Systems; in part by Piano Nazionale di Ripresa e Resilienza (PNRR) Ministero dell'universit'a e della ricerca (MUR) Project VITALITY, Spoke 2 ASTRA–Advanced Space Technologies and Research Alliance, under Grant ECS00000041; in part by PNRR MUR Project CHANGES, Spoke 5 Science and Technologies for Sustainable Diagnostics of Cultural Heritage, under Grant PE0000020; in part by Progetti di Rilevante Interesse Nazionale (PRIN) Project–RoboChor: Robot Choreography under Grant P2022RSW5W; in part by PRIN Project–HALO: etHical-aware AdjustabLe autOnomous systems under Grant 2022JKA4SL; in part by MUR (Italy) Department of Excellence 2023–2027 for Gran Sasso Science Institute (GSSI); and in part by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation. The work of Patrizio Pelliccione was supported in part by the Centre of Excellence on Connected, Geo-Localized and Cybersecure Vehicles (Excellence (EX)-Emerge), funded by Italian Government under Comitato Interministeriale per la Programmazione Economica (CIPE) Resolution under Grant 70/2017 (August 7, 2017). The Associate Editor for this article was G. Ostermayer. *(Corresponding author: Rodrigo Queiroz.)*

Rodrigo Queiroz, Divit Sharma, and Krzysztof Czarnecki are with the Department of Electrical & Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: rqueiroz@uwaterloo.ca).

Ricardo Caldas and Sergio García are with the Department of Computer Science and Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden.

Thorsten Berger is with Faculty of Computer Science, Ruhr University Bochum, 44801 Bochum, Germany, and also with the Department of Computer Science and Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden.

Patrizio Pelliccione is with the Gran Sasso Science Institute (GSSI), 67100 L'Aquila, Italy.

Digital Object Identifier 10.1109/TITS.2024.3373531

I. Introduction

TESTING automated driving systems (ADS) requires simulating a wide range of operating scenarios to ensure an ADS's safety and conformity to traffic regulations and industry standards. As the responsibility for the driving task shifts from the human driver to the ADS [1], the system is required to handle interactions with other road users, especially human-operated vehicles. Test scenarios must reflect how these dynamic interactions between the subject system (a.k.a. *ego* vehicle) and other vehicles can unfold in real traffic.

Figure 1 shows a near-collision of ego with v_2 cutting-in before, taken from the National Highway Traffic Safety Administration's (NHTSA) pre-crash scenario catalog [2]. The cut-in maneuver of v_2 triggers reactions by other close vehicles, with ego's reaction strongly influencing how the scenario unfolds. Testing collision avoidance in such scenarios requires models able to represent and simulate traffic dynamics, including the interactions between ego and other human-operated vehicles.

Many domain-specific languages (DSLs) [3], [4] for scenario-based testing have emerged. These DSLs include models for representing test scenarios. Testers design such scenarios by defining behaviors of human-operated vehicles, and then executing them in simulation tools. However, these DSLs are often limited to relatively simple models, for instance, replay of pre-recorded trajectories [5], event-based orchestration to directly manipulate the vehicle state [6], [7], and narrow

behavior models (e.g., vehicle following [8]). As a result, testers may have limited control over the precise movement of the simulated vehicles and the resulting behaviors may vary between simulation tools, hurting test reproducibility and the validity of test results.

To bridge the gap between scenario design and execution, we contribute the *GeoScenario Simulated Driver-Vehicle (SDV) model* to specify and simulate realistic behavior of human-operated vehicles in ADS scenario testing. SDV offers high expressiveness, execution accuracy, scalability, and reuse. It extends the scenario-definition language GeoScenario [5] with human-operated vehicles as dynamic agents in both scenario representation and simulation execution. It implements a driver behavior model inspired by Michon [9], including (i) route selection as a strategic decision, (ii) maneuver selection as a tactical decision, and (iii) maneuver implementation as an operational decision. Specifically, the maneuver selection logic is expressed using behavior trees [10], [11], offering modularity and reuse. The maneuvers are implemented using an optimization-based trajectory planner, which guides the simulation towards achieving the scenario's test objectives. The executed maneuvers can be configured by simulation engineers to reflect different driving styles, subject to the human and physical limitations of actual vehicles.

We evaluate our model's (i) scenario design effectiveness, which includes expressiveness, execution accuracy, and reuse, using NHTSA pre-crash scenarios; (ii) motion realism in comparison to naturalistic urban traffic; (iii) scalability with traffic density; and (iv) practical applicability to test an actual ADS. The results show that our model can successfully express, achieving levels of model reuse of over 80 %, and accurately execute all eighteen NHTSA vehicle-to-vehicle pre-crash scenarios (except one variant), while only four scenarios are effectively expressible using predefined trajectories, which is our baseline. We also show that, after calibration, the model is capable of producing maneuver decisions and trajectories that closely resemble those from recorded real-world traffic. The model also scales in scenarios with up to 10–20 simultaneous and highly interactive vehicles in real-time simulation. Finally, we demonstrate the model's applicability to test an ADS software stack in simulation, which has been tested on public roads, and reveal collision scenarios that cannot be expressed using the baseline.

In summary, our paper contributes:

- a novel simulation model for human-operated vehicles, that combines behavior trees with an optimization-based trajectory planner to provide a highly-expressive, controllable, realistic, reusable, and scalable scenario representation for ADS testing;
- a set of experiments to support our claims about the qualities of the model;
- an open-source reference implementation of the model, which can be integrated with any simulation environment in co-simulation mode.[1]

[1]https://github.com/rodrigoqueiroz/geoscenarioserver

II. Background and Related Work

A. Scenario-Based Testing

Kaner et al. [12] define *scenario-based testing* as the dominant paradigm of black-box testing, where scenarios are used to check how the system copes with both nominal and off-nominal situations. In the automotive context, ISO 26262 [13] and ISO 21448 [14] guide the development of safety-critical electrical/electronic vehicle systems and mandate the use of scenarios in validation activities.

Scenarios are designed based on expert knowledge and on the traffic situations the ADS must be able to cope with, or by reproducing and augmenting situations collected from traffic databases. For example, CommonRoad [15], a benchmark for motion planners, provides scenarios extracted from NGSIM data [16]. A scenario can also be systematically generated to achieve specific test goals, e.g., lead the system to trigger a certain behavior, such as an emergency maneuver, or find a critical situation leading to a crash. For example, Abdessalem et al. [17], [18] use evolutionary optimization methods combined with surrogate model learning to find crash scenarios.

B. Scenario Representation and Driver Behavior

Multiple tool-independent DSLs have emerged recently, providing a formal definition of scenario structure, behavior, test conditions, and pass/fail criteria to support scenario-based testing in simulation. The goal is to offer a uniform representation and semantics across methods and tools. The scope and structure of each language vary, but fundamentally they all define how vehicles behave in traffic and orchestrate interactions with ego that must be executed by a simulation tool during the test.

OpenScenario [6] is a standard managed by the Association for Standardization of Automation and Measuring Systems (ASAM). The format describes dynamic content in driving simulation applications in combination with OpenDRIVE [19], which specifies the road structure. It covers traffic and driver behavior, weather, environmental events, and other features. It includes the description of a driver, but there is no model for driver behavior in any form other than "road following." The standard also does not contain maneuver models or a vehicle model. Maneuvers are described in terms of *actions* (e.g., change the vehicle's position or speed), and trajectories (defined as a polyline, clothoid, or spline).

The Measurable Scenario Description Language (MSDL) [7] expands the concepts of OpenScenario. The language uses *modifiers* to change the behavior of the agents similarly to *actions* from OpenScenario. It introduces parameter variability (a range instead of a single value) along with constraints to narrow down values and connect multiple parameters (e.g., velocity of vehicle A is between 10 and 20 m/s and less than vehicle B). The language supports generating concrete scenarios by picking random values while obeying the constraints.

Other formats are Scenic [20], Scenario Description Language (SDL) [21], and SceML [22]. A common trait amongst them is that they are primarily declarative languages. They

define "what" must happen in a scenario during key events without specifying "how." Their approach relies on external simulation models to handle the execution.

Finally, GeoScenario [5] provides mechanisms to represent road users and an orchestration system to allow testers' control of how they interact with ego. The language tackles the multi-agent orchestration via triggers, but is limited at the individual vehicle behavior to select among predefined trajectories specific to the road. Our SDV model extends GeoScenario with interactive and flexible driver behavior.

C. Models for Traffic Simulation

Macroscopic traffic models describe vehicle motion and interaction in terms of flow and density. They are mainly used for large scale simulation over a road network [23]. They are not suitable for street-level vehicle motion and interactions and thus ADS testing.

In contrast, microscopic traffic models can generate vehicle motion and interactions at the individual vehicle level at the cost of limited scalability [24]. They are able to encode simple rules that allow a vehicle to follow waypoints or the structure of the road, avoid frontal collisions by alternating between driving and stopping, and perform maneuvers triggered by conditions [25], [26], [27]. However, while capturing this reactive behavior, they usually lack enough detail to simulate complex interactions between the vehicle under test and other road users in realistic conditions. For example, they often use simplistic motion limited to a constant velocity throughout a maneuver and disregard the physical limitation of a real vehicle. They also cannot represent complex interactions, such as vehicles responding to merge attempts, using the available road space to navigate around obstacles, or skillfully navigating an intersection with multiple influencing factors (e.g., vehicles, pedestrians, and traffic regulation).

Established microscopic models target a particular maneuver, for example, vehicle following [8], [25], [28], [29], decisions to perform lane changes [26], [30], and the execution of lane changes [31]. While these models capture details of speed regulation during vehicle following or the parameters of deciding lane changes, they are suitable for testing specific functions and subsystems (for instance, testing adaptive cruise control) in a very constrained environment. They do not cover the complexity of the full driving task required for scenarios in system-level testing of an ADS, including complex decision making among multiple maneuvers and trajectory generation. They can be used to inform the design and parameter setting of the behavior trees in our model, however.

A different approach is to learn models directly from data. For example, a trajectory prediction model trained on recorded traffic data can be run in closed loop as a simulator [32], [33], [34], [35], [36], [37]. Recent approaches allow a degree of controllability of the road users during simulation, e.g., by using conditional models [36] or diffusion models with cost functions [35], [37] to guide trajectory sampling during inference. While helping to automate scenario creation, the main limitation of purely data-driven approaches is the inherent bias in the data used to build the models. In particular, driver mistakes and safety-critical scenarios are rare in traffic and thus usually absent from or rare in existing datasets. In fact, programmable behavior models provide an opportunity to generate such rare scenarios and use them to augment the training for data-driven approaches. Finally, while the models can capture the diversity of driving styles in road environments they were trained on, they are difficult to generalize to other environments [38].

Thus, scenario-based testing requires executable models that offer high expressiveness, controllability, and realistic behavior—a combination that existing work currently lacks.

D. Behavior Trees

Behavior trees is a discrete control architecture, which aims to address the shortcomings of finite state machines and their variations, and provide improved modularity, reusability, scalability, and readability [10], [11], [39]. These user-oriented qualities motivate their use to express driving behavior, which has been explored in the past. Several works have proposed using behavior trees to make maneuver decisions within an ADS [41], [42]. Perhaps the closest is BTScenario [43], which uses them to control maneuvers of vehicles in simulation testing. However, BTScenario uses behavior trees to issue driving control inputs directly to a longitudinal and lateral controller. The lack of a trajectory planner makes it impossible to plan flexible and realistic trajectories to avoid static and dynamic obstacles. The work also lacks a systematic evaluation of expressiveness, reusability, motion realism, and scalability. In another work, we used behavior trees to control pedestrians in simulation, where behavior trees set motion objectives for pedestrians moving according to the social force model [44]. To the best of our knowledge, we are not aware of other work that (i) combines behvaior trees with an optimization-based planner to provide a highly-expressive, controllable, realistic, reusable, and scalable scenario representation for ADS testing, and (ii) systematically evaluates such an approach.

III. The SDV Model

We now introduce the concepts and the algorithm of the SDV model (see Fig. 2). The overall simulation consists of (i) a set of simulated road users, each run in a separate process (SDV Planner) that plans its future trajectory, and (ii) a single traffic simulation process (Traffic Simulation) that executes these trajectories. For simplicity and scalability, the model combines driver and vehicle as a single entity (SDV Planner), abstracting away driver inputs, such as steering angle, braking, and throttle.

An SDV Planner executes its behavior tree and communicates with the Traffic Simulation using two shared variables: an SDV Planner p_i reads the traffic state (TS) and writes the traffic plan (TP), and the Traffic Simulation reads TP and writes TS. The latter includes the current state of all vehicles, including their coordinates x, y in the global Cartesian frame of the simulation, their first and second time derivatives, and heading θ:

$$VehicleState_{\text{Cartesian}}(t) = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}, \theta]_t \qquad (1)$$

The traffic plan includes the future trajectories for all SDVs. Each trajectory is represented in the Frénet reference frame [45] of the respective SDV (Fig. 3). This is motivated by the fact that safety requirements on the motion

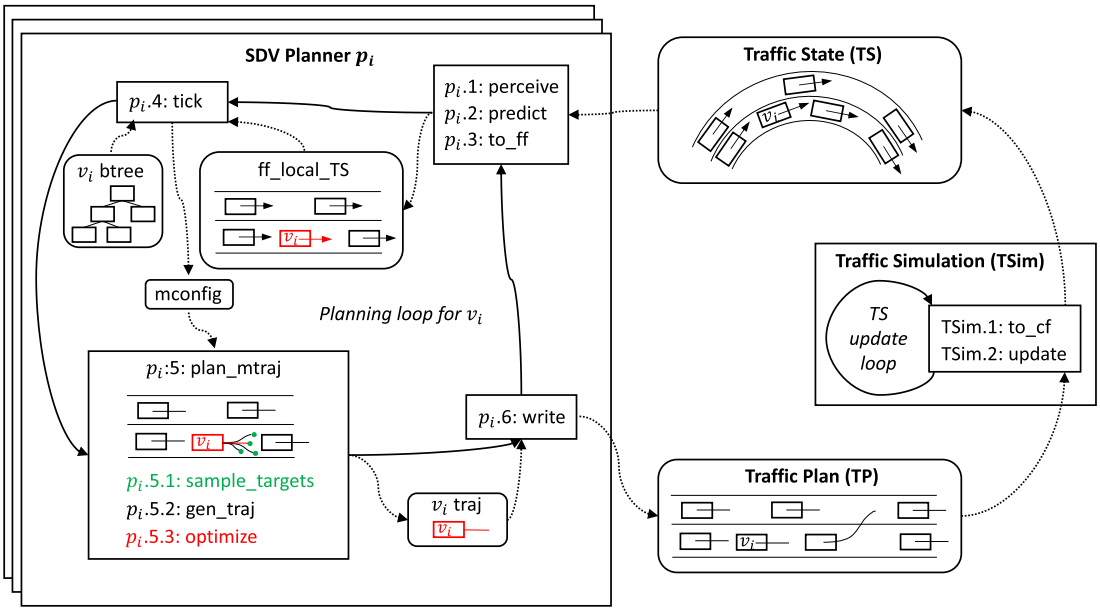

Fig. 2. Flow diagram of the simulation. Top-level sharp-cornered rectangles represent processes; nested ones represent procedures. Round-cornered rectangles denote data. Solid arrows represent control flow; dashed ones represent data flow. Arrows attached to vehicles denote velocities; curves denote trajectories.

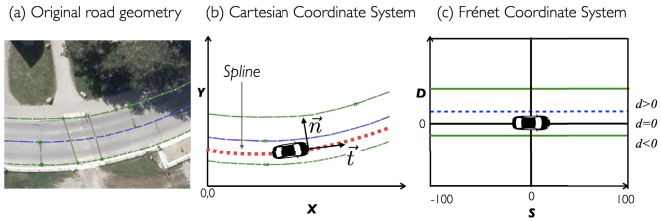

Fig. 3. Road Geometry and vehicle displacement from original coordinates are transformed into Frénet Frame using the tangential and normal vectors $\vec{t}$, $\vec{n}$ from the lane centre line (shown in red).

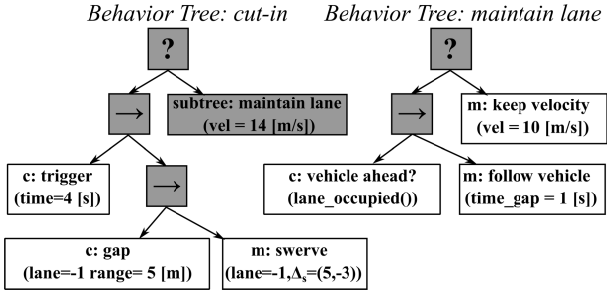

Fig. 4. Graphical representation of an example SDV behavior tree structuring the decision-making with conditions (c) and maneuvers (m). '?' is the *fallback* operator (short-circuit or), and → is the *sequence* operator (short-circuit and).

of an on-road vehicle are typically specified relative to its Frénet frame derived from the local lane geometry (see, e.g., Shalev-Shwartz et al. [46]).

A. SDV Planner

An SDV Planner process is instantiated for each SDV, as indicated by the stacked boxes in Fig. 2. The SDV Planner p_i for vehicle v_i is given a route, represented as a sequence of lane segments that can be legally traversed by the vehicle, and a behavior tree (btree), and it performs a maneuver planning loop with six steps (p_i.1–6). Maneuver planning starts with the procedure p_i.1:perceive, which retrieves the current traffic state from the perspective of v_i and simulates perception, including sensor range. The next procedure, p_i.2:predict, projects the perceived local traffic state forward to the future simulation time targeted by the current maneuver planning iteration. Prediction uses the previously planned trajectory for v_i but assumes constant velocity for all other vehicles, including externally-simulated ones for which planned trajectories are not observable, such as ego under test. The next step, p_i.1:to_ff, transforms the local traffic state TS into the Frénet frame of v_i, resulting in ff_local_TS. The frame is defined w.r.t. the center line of the lane (red in Fig. 3(b)) that v_i is traveling on as part of its route (Fig. 3(a)). Its origin is the point along this line that is closest to the vehicle. The resulting frame's S axis represents the longitudinal displacement along this center line, and the D axis represents the lateral displacement.

B. Maneuver Selection

Maneuver selection is expressed using behavior trees. Their leaf nodes are either (i) conditions to be evaluated (based on the traffic state), (ii) decisions that start (or end) maneuvers, or (iii) references to sub-trees. The inner nodes are control nodes, a.k.a. operators, which are responsible for coordinating the execution of their child nodes. There are three operators. The *fallback* operator commands a sequential execution of its children, left-to-right, and returns success immediately when a child succeeds; otherwise, it executes the next child. It returns failure when none of the children succeeds. The *sequence* operator also commands a sequential execution of its children, left-to-right, but returns failure immediately when a child fails; otherwise, it executes the next child. It returns success when all of the children succeed. The *parallel* operator commands the execution of all children at the same time. The rule for success or failure of the parallel operator is user-defined.

Figure 4 shows a graphical representation of two example behavior trees, with the left one being the main tree planning cut-in behavior, and the right one being a sub-tree referred

to from the main one and performing lane maintenance. The main behavior tree would be assigned to an SDV, e.g., v_2 in Fig. 1. In each maneuver planning cycle of v_2, the main tree is "ticked" (p_i.4:tick in Fig. 2), i.e., executed, with the local traffic state as context. The execution starts with the root of the main behavior tree and traverses the nodes according to the operator semantics. In our example (Fig. 4), the execution starts from the fallback operator at the root and proceeds to its child sequence node and then to condition (c:trigger), which tests whether the simulation has been running for 4 s. If the condition is satisfied, the execution proceeds to the deepest sequence node and then to the condition (c:gap) checking the acceptance distance gap of 5 m ($+-10\%$) for a lane change in front of another vehicle to the right (lane_id= -1). If the gap condition is satisfied, the maneuver node (m:swerve) executes the lane change with a target distance gap of 5 m and a relative velocity of -3 m/s (Δ_s=(5,$-$3)). If any of the two conditions fails, the reference node is executed, triggering the execution of the sub-tree on the right, which implements a simple lane maintenance behavior.

A maneuver exposes a set of parameters to control it according to scenario objectives. We use existing maneuver catalogs [47], [48] and implement a subset to support the evaluation in Sec. V: keep velocity, follow vehicle, swerve (used for lane change and swerve-in-lane), merge-in-front, stop, and reverse. Note that these are elemental maneuvers; composite maneuvers are implemented as behavior trees over the elemental maneuvers. For instance, lane maintenance composes velocity keeping, vehicle following, and stopping (for more complex examples see Queiroz [49]).

A maneuver node (e.g., m:swerve in Fig. 4) is represented by a *maneuver configuration* (mconfig in Fig. 2), consisting of the maneuver type (e.g., swerve) and a set of maneuver-specific parameter values, such as the target gap distance and velocity delta for swerve. The behavior tree execution ('tick') is expected to return a maneuver configuration, which is passed to maneuver trajectory planning. A given maneuver ends when a condition for a new maneuver is triggered in btree.

C. Maneuver Trajectory Planning

The maneuver trajectory is planned by p_i.5:plan_mtraj in Fig. 2 using the maneuver configuration (mconfig) and local traffic (ff_local_TS) as inputs. A trajectory is represented by longitudinal $S(t)$ and lateral $D(t)$ position in Frénet frame as functions of time and the trajectory duration T (2). Velocity and acceleration are the first and second derivatives, respectively, yielding the longitudinal and lateral state (3):

$$Trajectory = [S, D, T] \tag{2}$$

$$\begin{aligned} VehicleState_{\text{Frénet}}(t) = [&S(t-t_0), \dot{S}(t-t_0), \ddot{S}(t-t_0),\\ &D(t-t_0), \dot{D}(t-t_0), \ddot{D}(t-t_0)]\\ &\text{for } 0 \le t-t_0 \le T \end{aligned} \tag{3}$$

The maneuver trajectory planning has three steps: (i) sampling the target states for the maneuver, (ii) generating candidate trajectories, and (iii) selecting an optimal trajectory. Each of these steps is controlled by a set of parameters

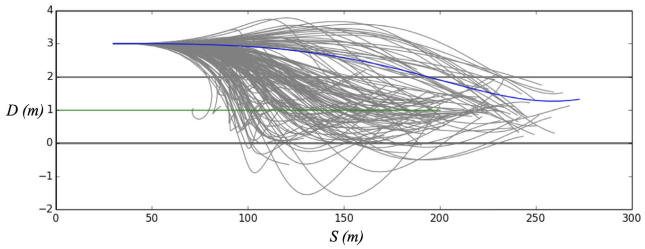

Fig. 5. Trajectory planning by the SDV during a cut-in maneuver.

accessible through the maneuver configuration and allowing testers to realize a particular driving style or misbehavior. The generated trajectories are kept short (2 to 5 s), but some maneuvers, e.g., vehicle following, are performed over extended periods of time and thus consist of a sequence of trajectories. A behavior tree decides when to start, finish, or abort a maneuver.

Figure 5 shows an example of trajectory planning for a cut-in maneuver to the right lane. The grey lines are the candidate trajectories eliminated by the optimization step due to feasibility constraints or higher cost. The blue line is the best cut-in trajectory based on motion constraints and the scenario goals specified in mconfig, such as the target gap to ego. The green line is the ego trajectory. The remainder of this section describes each of the three planning steps in more detail.

1) Maneuver Target Sampling (p_i.5.1:sample_targets): Each maneuver has its own configurable criteria to define its target state and a time to reach it (T). Target sampling requires evaluating the road structure, traffic, and other objects. For example, in the NHTSA pre-crash scenario 'following vehicle making maneuver' scenario, a leading vehicle decelerates to turn right that may end up in a crash with an inattentive following vehicle.

In such scenario, the leading vehicle's target for velocity keeping is to comfortably accelerate to and maintain a specified target velocity, e.g., 16 m/s. The following vehicle's target for vehicle following is to reach and keep a certain target time gap, e.g., 10 s. While defining the maneuver configuration, parameters can be set as a single value or a value range, e.g., a vehicle target speed of exactly 14 m/s, or within 20 % from 14 m/s. The target sampling step samples multiple values for each range parameter independently and creates a target state set as a Cartesian product over the parameter value sets. The sampling method of choice and the number of samples per parameter are configurable through mconfig. The target state set corresponds to the end points of the trajectories in Fig. 5.

2) Trajectory Generation (p_i.5.2:gen_traj): Given a target state set, trajectory generation computes a smooth motion profile between the current vehicle state and each target state in the Frénet frame (Fig. 5). We use an approach that plans each trajectory as a pair of quintic polynomials, in longitudinal and lateral direction, respectively, which minimizes jerk to reflect smooth and comfortable driving [45]. A quintic polynomial is a jerk-minimal connection between two points P_0 and P_T, with $p(t)$ as location and T as the motion duration [50]. More precisely, such a quintic polynomial minimizes the total accumulated jerk over the one-dimensional trajectory:

$$J_{p,T} := \int_{t=0}^{t=T} \dddot{p}^2(t)dt \tag{4}$$

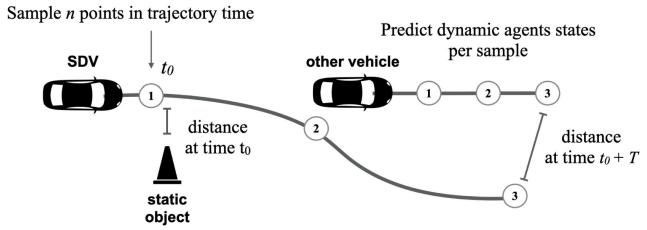

Fig. 6. Checking for collisions with static objects and dynamic obstacles.

This step generates a trajectory by computing the coefficients of two quintic polynomials, $S(t)$ for the longitudinal dimension as $p(t)$, and $D(t)$ for the lateral direction as $p(t)$, to fit the boundary conditions: the initial state $VehicleState_{\text{Frénet}}(t_0)$ and each of the target states $VehicleState_{\text{Frénet}}(t_0 + T)$ from the target-sampling step. This results in a candidate set that respect the maneuver target constraints.

3) Optimal Trajectory Selection (p_i.5.3:optimize): This step selects a trajectory that is feasible and optimal with respect to a set of maneuver feasibility constraints and cost functions, which are configured in the maneuver configuration to suit the needs of the test scenario. Feasibility constraints reject trajectories with any collision, direction inversion, lane departure, and exceedance of maximum lateral/longitudinal jerk and acceleration. These are checked by sampling points over the planned and predicted trajectories (e.g., ego), as illustrated Fig. 6. Note that the optimization step predicts the motion of other vehicles by assuming constant longitudinal velocity in Frénet frame. Furthermore, the constraints can be configured to fit the scenario objectives. For example, the behavior tree of a v_2 (Fig. 1) may issue a swerve maneuver with a configuration that disables its collision check to simulate a reckless cut-in.

The remaining candidate set is ranked using a weighted sum of configurable cost functions:

- *Time cost* penalizes trajectories longer or shorter than the target time T.
- *Efficiency cost* penalizes low average velocity.
- *Lane-offset cost* penalizes distance from lane center during the entire trajectory.
- *Jerk cost* penalizes high longitudinal and lateral jerk over the entire trajectory ($J_{S,T}$ and $J_{D,T}$).
- *Acceleration cost* penalizes high longitudinal and lateral acceleration over the entire trajectory.
- *Proximity cost* penalizes proximity to obstacles (vehicles, pedestrians, or other objects).

The best trajectory is the lowest-cost feasible one. Weights can be adjusted per behavior tree node according to scenario goals. For example, if a given scenario requires the vehicle to drive too close to ego, the proximity cost weight for ego should be lowered. The resulting trajectory respects realistic vehicle motion, balances conflicting qualities such as progress and comfort, while implementing the scenario goals.

a) Traffic simulation execution: Traffic plans are executed in Traffic Simulation, a process that sets the traffic state of each SDV in TS according to its planned trajectory. It runs at a fixed frequency that is typically an order of magnitude higher than that of an SDVPlanner. The new trajectories produced by the SDVPlanner processes arrive asynchronously in the traffic plan TP (e.g., p_i.6:write). Traffic Simulation retrieves the state of each SDV for the current simulation time from TP, transforms it to the global Cartesian frame of the simulation (TSim.1:ro_cf), and updates the state of the corresponding SDV in TS (TSim.2:update). Note that updates to TP (p_i.6:write) and TS (TSim.2:update) are atomic.

IV. Model Implementation

A reference implementation for the SDV model and tools for running scenarios in simulation are available as part of the open-source project GeoScenario Server. The server parses scenario definitions expressed using Lanelet2 map [51] and the GeoScenario language [5] extended with the SDV behavior-tree definition format and creates a traffic simulation with the SDV model instances running concurrently. The server is implemented in Python and operates as a co-simulator to be interfaced with the simulation of the ego vehicle, its sensors, and the ADS under test. The implementation also provides a sample integration with an existing simulator, WISE Sim, and an ADS software stack, WISE ADS. The *GSClient* component provides a shared memory interface between the GeoScenario Server and WISE Sim, which runs within the game engine Unreal and provides LiDAR and camera simulation. The high-fidelity dynamics model of the ego vehicle, a Lincoln MKZ, runs as a Robot Operating System (ROS) [52] module along with the WISE ADS. The GeoScenario Server can be integrated into any other simulation environment, simply by customizing GSClient for the new environment.

V. Evaluation

We evaluate the SDV model in terms of design effectiveness, realistic vehicle motion, practical applicability for scenario-based ADS testing, and scalability. The following research questions guide our evaluation:

- **RQ1**: Can realistic and interactive scenarios for ADS testing be effectively modeled and executed via SDV models?
- **RQ2**: Can SDV models generate realistic vehicle motion?
- **RQ3**: Can the use of SDV models improve the effectiveness of scenario-based testing of a real ADS?
- **RQ4**: How does the model performance scale with traffic density?

A. Effective Scenario Development (RQ1)

We evaluate the effectiveness of scenario development using the SDV model by analyzing how the model improves GeoScenario as the baseline DSL to design and execute test scenarios from a catalog using three metrics:

(i) *Expressiveness*: Given a set of scenarios, we classify them as follows: we assign *success* (S) when all behaviors are successfully expressed with no limitations, *partial* (P) when the behaviors for at least one variation of the scenario can be expressed, or *failure* (F) otherwise.

(ii) *Execution accuracy*: After running a simulation, we classify the degree to which scenarios are correctly executed according to NHTSA description: *success* (S) when all vehicles behave as expected and the scenario objective is achieved; *partial* (P) when at least one variation of the scenario succeeds; and *failure* (F) otherwise.

(iii) *Reuse*: We quantify reuse in a scenario based on the *internal reuse level* [53]. Given a scenario containing a set of behavior trees (higher-level items), the metric is defined as M/L, where M is the number of nodes (lower-level items) that are used more than once (i.e., used also in behavior trees of other scenarios) and L is the total number of nodes in the set of behavior trees. This metric assumes values between 0 and 1 and represents the percentage of internal reuse. We also compute the internal reuse level accounting for only the nodes that are actually executed in a successful simulation.

Since the SDV model extends the capabilities of GeoScenario, we use the latter as the baseline [5]. We focus on safety-critical scenarios and, specifically, we use the Pre-Crash Scenario Typology from NHTSA [2]. These interactive and realistic scenarios can challenge the ADS capabilities in crash avoidance and they are commonly used as a reference for ADS validation in other projects [27], [54]. We filter the original set for scenarios with vehicle-to-vehicle interactions, resulting in 18 scenarios (Table I).

We design each scenario using a combination of the original GeoScenario and multiple instances of SDV models with their respective behavior trees and maneuver configurations. The original NHTSA set is based on reported events between human-operated vehicles, but we assume that one of the vehicles is ego, operated by the ADS (similar to how Waymo adapts NHTSA scenarios as tests [54]). Ego's goal is to drive through the scenario (from start to goal point) and avoid a collision. The goal of an SDV is to interact with ego using target parameters defined by the tester, e.g., achieving a certain time gap before braking. The overall scenario goal is to replicate the pre-crash events as described by NHTSA, leading to a crash or a near-crash. If execution differs by either a safe outcome (vehicles never interact or interact differently than intended) or another type of crash, the scenario execution fails. After modeling the scenarios, we simulate them in the reference implementation (Sec. IV).

As part of the comparison of expressiveness with the baseline, we classify the type of SDV behavior required in each scenario as *static* or *dynamic* with respect to three elements: *path shapes*, *speed profiles*, and *behavior triggers*. Behavior triggers are conditions triggering the required changes in paths and speed profiles during the scenario (Table I). Scenarios that involve static behavior for all three elements, i.e., fixed paths and speed profiles for each SDV and their starting triggers, can be easily designed with predefined trajectories from start to finish and do not benefit significantly from a dynamic model (stat,stat,stat in Table I). Scenarios that require dynamic behaviors, but the behaviors can be expressed as sets of static paths and velocity profiles with dynamic triggers to select among them (stat,stat,dyn in Table I), can still be modeled using predefined trajectories with reasonable effort.

TABLE I
SCENARIOS AND PERFORMANCE

ID	Group	Scenario	Path Shape	Speed Profile	Behavior Trigger	Expressiveness	Execution	IRL	IRL exec
4	CP	Running Red Light	stat	dyn	stat	S	S	0.83	0.60
5	CP	Running Stop Sign	stat	dyn	dyn	S	S	1.00	1.00
15	B	Backing Up Into Another Vehicle	stat	stat	dyn	S	S	0.91	0.60
16	LC	Turning SD	dyn	dyn	dyn	S	S*	0.87	0.63
17	LC	Parking SD	dyn	dyn	dyn	P	P	0.84	0.71
18	LC	Changing Lanes SD	dyn	dyn	dyn	S	S	0.89	0.79
19	LC	Drifting SD	dyn	dyn	dyn	S	S	0.79	0.71
20	OD	Making Maneuver OD	dyn	dyn	dyn	S	S	0.90	0.84
21	OD	Not Making Maneuver OD	dyn	dyn	dyn	S	S	0.76	0.50
22	RE	Following Vehicle Making Maneuver	dyn	dyn	dyn	S	S	1.00	1.00
23	RE	Lead Vehicle Accelerating	stat	stat	dyn	S	S	0.90	0.75
24	RE	Lead Vehicle at Lower Speed	stat	stat	stat	S	S	1.00	1.00
25	RE	Lead Vehicle Decelerating	stat	stat	dyn	S	S	0.90	0.75
27	CP	Left-Turn Across Path/OD at SJ	stat	dyn	dyn	S	S	0.90	0.75
28	CP	Vehicle Turning Right at SJ	stat	dyn	dyn	S	S	0.99	0.94
29	CP	Left-Turn Across Path/OD at NSJ	stat	dyn	dyn	S	S	0.98	0.93
30	CP	Straight Crossing Paths at NSJ	stat	dyn	dyn	S	S	0.94	0.81
31	CP	Vehicle Turning at NSJ	stat	dyn	dyn	S	S	0.94	0.81

Acronyms: B: Backing up, CP = Crossing Paths, LC = Lane Change, OD = Opposite Direction, RE = Rear-end, SD = Same Direction, SJ = Signalized Junction, NSJ = Non-Signalized Junction. Path Shape, Speed Profile, and Behavior Trigger are requirements for vehicle behavior that can be static (stat) or dynamic (dyn). Expressiveness and Execution show the degree in which a scenario is modeled and correctly executed, respectively (S=successfully, P=partially, F=Failed). The Internal Reuse Level (IRL) is computed with all Behavior tree nodes, and only for nodes that are executed in the simulation (IRL exec). *Scenario #16 required a map adaptation to perform correctly.

Finally, scenarios that require dynamic path or velocity profile or both (dyn,stat,*; stat,dyn,*; and dyn,dyn,* in Table I) are impractical to be modeled using predefined trajectories, but are enabled by the proposed SDV model. For example, the cut-in scenario has a continuous space of paths and speed profiles, and a dynamic trajectory needs to be planned based on the ego behavior, which may vary from execution to execution. We note that using the NHTSA descriptions of the scenarios as a source, many scenario variants are possible. Our classification is based on the minimal behavior required to reproduce the critical event occurring immediately prior to a crash as described by NHTSA; however, added elements, such as additional vehicles, might change the static classification to a dynamic one, but not the other way.

Results: Due to limited space, we focus on the main findings here. The full list of scenarios is in the online repository.[2]

Expressiveness: All 18 scenarios except for one variant of #17 are successfully expressed using the SDV model. We identify 14 scenarios (78%) that depend on dynamic path or velocity profile, or both, and thus are impractical for the baseline. For instance, a vehicle leaving a parking position in scenario #17 must start this maneuver only when ego is approaching and adjust its trajectory, in one of the variants,

[2]https://github.com/rodrigoqueiroz/geoscenarioserver

to merge ahead of ego. While the vehicle must challenge the ADS, an unavoidable lateral crash into ego would not be useful as a test scenario. To achieve the scenario goal, the vehicle must be able to generate a trajectory relative to ego's motion at run time. The same requirement applies to all lane-change scenarios (#16-#19). For crossing-path scenarios #30 and #31, the velocity profile must be dynamically planned. The SDV models enable us to successfully express these dynamic behaviors, which are infeasible with the baseline, resulting in a higher expressiveness. One variant of Scenario #17 "Parked Vehicle SD" requires the parked vehicle to join traffic by making a U-turn, and this maneuver is currently not supported by the implementation of trajectory generation.

A total of four scenarios (22%) require only static trajectories (stat,stat,* in Table I) and thus can be designed with the baseline. For instance, in the rear-end scenario #25 both path shape and speed profile can be generated offline and expressed as predefined trajectories with only a trigger to activate the deceleration as ego approaches. In such examples, the SDV model does not increase expressiveness. However, it adds two advantages: (i) conciseness, by defining the scenario at a higher level of abstraction using target parameters instead of detailed trajectories, and (ii) flexibility, by allowing the scenario to be replicated in different road geometries without changing the behavior definition.

Execution: In 17 scenarios, vehicles perform as expected, and the scenario ends with a crash or near-crash as described in the NHTSA report. The performance deviates from the design in the scenario #16 "Vehicle(s) Turning - Same Direction." The assigned behavior requires that vehicles perform a maneuver that violates the legal road-network connectivity. Since the current implementation relies on the Lanelet map to constrain the driving space, the map requires an adaptation to execute the scenario correctly.

Reuse: The composable nature of behavior trees allows us to reuse most of them, i.e., use each tree in two or more scenarios, since there is significant commonality in the driving task for the different scenarios. In most scenarios, vehicles start by performing normal lane maintenance until an unexpected event occurs, such as a risky behavior of another vehicle. The differences among scenarios emerge in such events and are usually modeled at the highest levels of the main behavior tree for the given scenario. We call them the "*scenario-trees*." The remaining tasks are reusable and performed using "*sub-trees*" (e.g., performing a lane-change). This reuse pattern is not part of the original behavior-tree concept, but it has emerged during this experiment when trying to maximize reuse. In some instances, a simple overriding of parameters for conditions or maneuvers during the sub-tree composition is sufficient to adapt the behavior from one scenario to another and achieve the scenario objective with 100% reuse (see *Internal Reuse Level* in Table I). Overall, the average internal reuse level (weighted by the size of behavior trees in each scenario) is 0.93 for all nodes, and 0.81 for executed nodes.

The experience modeling and running NHTSA scenarios reveals how effective the SDV model can be in ADS scenario development. The model enables expressing highly dynamic behaviors, fosters reuse, and can successfully execute most scenarios in simulation. Vehicle interactions involving lane changing, merging, and crossing paths are severely limited or impractical using the baseline of predefined trajectories. Thus, such interactive scenarios benefit most from the SDV model. The limitations we identify are due to missing underlying maneuvers (such as a U-turn) or the map constraints that prevent certain vehicle movements. We will address them in future work.

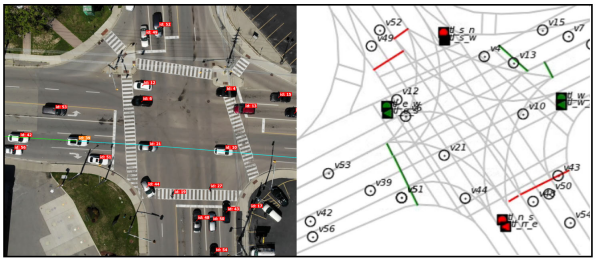

Fig. 7. A snapshot of the signalized intersection used for experiments and its corresponding simulation on the right.

B. Vehicle Motion (RQ2)

As the primary goal is to simulate human-operated vehicles, a good model must reflect the human-driving behavior and how vehicles move in naturalistic traffic conditions. To evaluate the motion realism, we use SDV models to replicate scenarios collected from urban traffic and compare their behavior with real vehicles. It is unreasonable to expect SDV models to drive exactly like the empirical vehicle, since not even humans drive equally. However, our model is designed to be highly configurable and adapt to different driving styles. With the proper configuration in the calibration process, we expect that SDV models can approximate the behavior of the empirical vehicles to a high degree given the same environment conditions. We use data from a busy signalized intersection during mid-day traffic in Waterloo, Canada, which is part of the Waterloo Multi-Agent Traffic Dataset [55]. The "birds-eye" video was collected using a drone and processed to label and track pedestrians and vehicles (Fig. 7).

This experiment follows four steps:

1) *Data preparation:* We classify the vehicle trajectories in the dataset into five scenario types based on the main maneuver they represent: (i) vehicle crossing intersection unconstrained (free), (ii) vehicle stopping (red light), (iii) vehicle resuming driving (green light), (iv) vehicle following a lead through the intersection (follow), and (v) vehicle partly following a lead when the lead merges or leaves mid-scenario (free/follow). In cases where a vehicle stops at a signal light, we split the trajectory into two scenarios, namely (ii) and (iii), in order to eliminate the waiting state. Each such classified vehicle trajectory represents an individual experimental trial.
2) *Test generation*: For each classified vehicle trajectory, we identify the traffic conditions that may affect how the vehicle is driving, e.g., signal light states and all other

vehicles and pedestrians that may affect it, to be reproduced in simulation. Each classified vehicle trajectory is used as a reference vehicle for a single test. We generate a new GeoScenario test replacing the reference vehicle with an SDV model instance with a standard-driver behavior tree and using the same start state (velocity and position in the intersection), and replicate the traffic conditions to ensure the driving task is influenced by the same factors. The standard-driver behavior tree is capable of performing each of the five maneuvers. We also assign a route goal to the model based on the last known position of the empirical reference vehicle to ensure the simulated vehicle will navigate the intersection towards the same exit lane. All other relevant empirical vehicles and pedestrians are included in the test as agents with predefined trajectories, and the signal light phases are also replicated. We generate 100 test scenarios and manually review the correctness of the identified traffic conditions.

3) *Calibration*: While each simulated reference uses the same standard-driver behavior tree, it needs a behavior-tree configuration to replicate the driving style of its empirical counterpart. We use a set of rules to automatically analyze each empirical reference trajectory and generate a configuration for it by extracting a set of high-level driving-style parameter values and value ranges, including maximum and average velocities, lateral displacement on the lane, stopping distance to target, reaction times, and time gap to other vehicles. We adjust the SDV parameter ranges to target similar values.
4) *Simulation*: We run two simulations per scenario using the SDV model, one with a default configuration before the calibration and another one after the calibration, and export the resulting trajectories as a discrete set of the vehicle states in the simulation frequency at 30 Hz. The default configuration uses nominal naturalistic driving parameters, such as zero offset from the lane centerline and a time gap range of 1.8..2.2 s [56].

The SDV performance is assessed using a measure of distance between the simulated trajectory T_1 and the empirical reference trajectory T_2, which takes into account both their spatial and temporal characteristics. The shorter the distance, the more similar the motion behavior of the simulated and the empirical vehicle. We use the spatio-temporal Euclidean distance (STED) [57], which represents the average Euclidean distance between positions of the respective vehicles, $T_1(t)$ and $T_2(t)$, along their respective trajectories T_1 and T_2, over the interval l in which both trajectories exist:

$$d_{STED}(T_1, T_2) = \frac{\int_l d(T_1(t), T_2(t))\ dt}{|l|} \tag{5}$$

Results: Figure 8 shows the distribution of STED before and after calibration per scenario type. The majority of simulated trajectories are already fairly similar to their empirical reference even before the calibration with an average STED of 4.27 m. A review of the simulated trajectories shows a similar decision making patterns, such as reacting to traffic lights and vehicles ahead, to the empirical ones. However, the main differences are observed in the speed profiles, lateral placement on the lane, time gaps, and various delays and reaction times, all indicative of different driving styles. The calibration brings the simulated trajectories significantly closer to their empirical counterparts: average STED for all 100 scenarios reduces from 4.27 m to 1.24 m. At an individual level, calibration improves the performance in 82 scenarios. Although the performance is worse for 18 scenarios, it is only slightly worse for 16 of them, with less than 1 m deterioration. Only two scenarios deteriorated more significantly, by 1.4 m and 1.9 m. The latter deviation is due to an erratic driving style of the empirical reference vehicle, which accelerates hard when resuming driving on green and then decelerates for no apparent reason. Such erratic behavior could be replicated by a dedicated maneuver.

Figure 9 shows the paths and speed profiles of sample individual scenarios. Plot (a) shows the reference vehicle 5 reacting to a red light. The path before calibration shows the simulated vehicle stop at the stop line, but the empirical

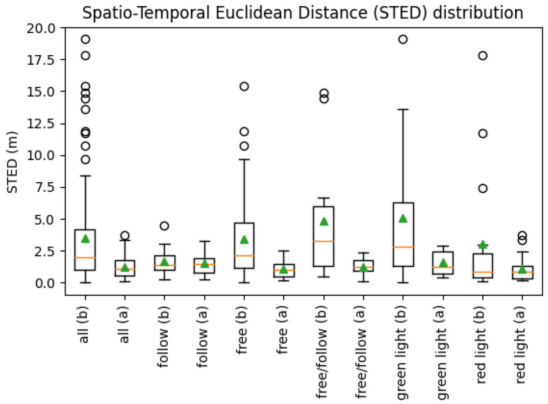

Fig. 8. Performance (Eq. 5) for all scenarios and per type, before (b) and after (a) calibration, measured using STED in meters. Orange lines represent medians, and green triangles represent averages.

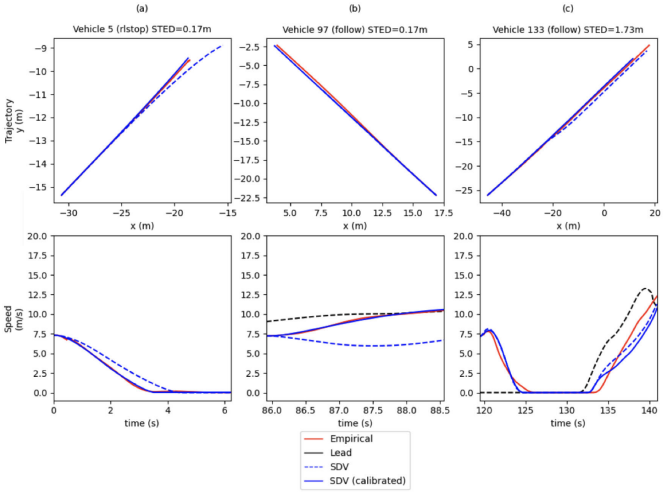

Fig. 9. Paths and speed profiles for three sample scenarios. Empirical vehicles in red; SDV models in dashed blue (before calibration) and solid blue (after calibration). Eq. 5 defines performance using the spatio-temporal Euclidean distance (STED).

vehicle stops about 2.5 m before the line. After calibration, both the simulated and empirical paths match up almost perfectly, with an STED of 17 cm, and a maximum distance of 31 cm. The calibrated speed profile also closely matches the empirical one. Plot (b) shows vehicle 97 crossing the intersection southwards, while already following a lead vehicle. The black dashed line shows the lead vehicle's speed profile, which is fairly constant throughout the scenario. The initially slower reference vehicle accelerates to match the lead's speed. The calibration improves the default configuration to match the more aggressive time-gap of the empirical vehicle, resulting in closely matched speed profile and reducing the STED from 2.37 m to 17 cm. In rare cases, the calibration does not improve performance, as shown in plot (c). A vehicle approaches the intersection with a red light and an already stopped vehicle ahead. The reference vehicle can resume driving on the green light, but needs to keep distance from the lead vehicle. The simulated vehicle resumes with a smaller delay compared to the empirical one.

In summary, SDV models can closely reproduce the behavior of human-operated vehicles under the same traffic conditions. The model calibration can address varying driving styles and significantly increase the similarities in the trajectories. In some scenarios, such as in Fig. 9 (a), the simulated trajectory after calibration is in essence indistinguishable from the empirical one, with maximum difference of 31 cm. In some scenarios the human behaves unexpectedly, however, and the current automatic calibration process cannot replicate such behaviors, but they could be modeled in the behavior trees as additional maneuvers.

C. Application (RQ3)

We run an in-depth case study to evaluate how the model performs in a real ADS testing environment and answer RQ3. We choose the cut-in lane change NHTSA scenario (#18 in Table I) to test an actual ADS software as the subject system. In this scenario, a vehicle changes lanes at a non-junction and merges closely in front of the ego traveling in a adjacent lane in the same direction. Cut-in maneuvers from other drivers pose challenges to the ADS and if not handled properly can lead to crashes. Thus, they represent an important test case.

The test aims to evaluate the impact of key vehicle interaction parameters, such as relative velocity and gap, on the likelihood and crash severity. The non-deterministic behavior of the subject ADS makes simulating this type of scenario challenging, however. Reaching the desired test parameter values while performing realistic vehicle interactions requires a reactive model, capable of adapting and re-planning trajectories as the scenario unfolds.

The case study has an explorative nature, with the objective to generate practical insights of applying the SDV model to test a real ADS, including identifying potential limitations.

1) System Under Test: We test *WISE ADS*, developed at the University of Waterloo.[3] The ADS software consists of a set of ROS modules implementing object-detection and tracking, occupancy and high-definition mapping, localization and state estimation, maneuver and trajectory planning, and control. The software can operate a Lincoln MKZ Hybrid, equipped with a drive-by-wire interface and a suite of LiDAR, camera, GPS, and inertial sensors, in automated mode at SAE level 3 on public roads in Waterloo. We test the ADS software in simulation, using WISE Sim with the GeoScenario Server implementing the SDV model (see Sec. IV).

2) Test Scenario: The cut-in behavior is expressed as a behavior tree similar to Fig. 4 and assigned to an SDV model instance. According to this behavior tree, the vehicle must reach a certain acceptance (rear) gap before performing the cut-in maneuver and then achieve a certain target (rear) gap to ego. The behavior tree calls the standard-driver behavior tree to maintain its current lane, parameterized with a target speed of 14 m/s ($+-10\%$), which is slightly higher than the road speed limit. The simulation plans candidate trajectories by sampling 6 target velocities from this target range (uniformly, by default). After a delay of 4 s to allow the vehicle to pick up pace, it starts checking for the acceptance distance gap of 5 m ($+-10\%$) for a lane change to the right (lane$=-1$), on which ego drives at the road speed limit. Once the acceptance gap is satisfied, the lane change is triggered, with a target distance gap of 5 m and a relative velocity of -3 m/s (Δ_s=(5,$-$3)). The experiment repeats the scenario with different combinations of parameters to evaluate how ego handles a variety of cut-in trajectories and find configurations that are more likely cause a crash.

Results: As expected, more aggressive cut-ins (shorter acceptance distance gap, shorter target distance gap, and lower target velocity) are more likely to cause collisions, but the response of the ADS to different parameter combinations of the cut-in maneuver is non-obvious (see Table II). Scenarios #7 and #8 are parameterized with the same short acceptance gap Δd_a=2 m and the same target relative velocity $\Delta v_t = -5$ m/s, but #8 has a smaller target distance gap, $\Delta d_t = -5$ m, compared to Δd_t=$-$2 m for #7. As a result, #8 ends in a collision. Note that Δd_t and Δv_t are planned relative to the predicted ego location at the end of the cut-in maneuver, assuming ego continues at a constant velocity. Thus, although a negative Δd_t would guarantee a collision if ego maintained its velocity, ego is likely to brake and thus a negative Δd_t does not necessarily result in a collision. Scenarios #9-11 use a larger acceptance gap, with Δd_a=5 m. As a result, although #9 has the same target parameters as #8, a collision is avoided, since the larger acceptance gap gives ego more time to react. Increasing the target deltas in #11 results in a collision, however. Figure 10 shows scenario #8 with the SDV's trajectory generation (a-b), its ground-truth perspective (c), and the ADS's internal perception of the scenario (d). The ADS detects the SDV (yellow bounding box), and the ADS's tracker predicts the SDV's future trajectory (bold green line) as in conflict with the ego's lane. Although ego initiates an emergency stop, the rear-end collision is not avoided.

This experiment demonstrates how the SDV model can be used with a real ADS to search for scenarios and parameters where the system may not be able avoid a collision.

[3]https://uwaterloo.ca/waterloo-intelligent-systems-engineering-lab/projects/wise-automated-driving-system

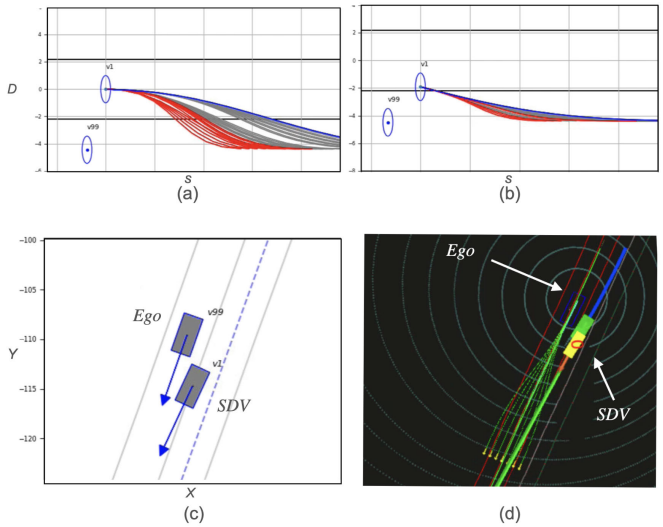

Fig. 10. One of the simulation scenarios that results in a crash; in (a) and (b), the SDV trajectory generation in Frénet Frame targeting ego at two different moments (optimal trajectory in blue and infeasible ones in red); in (c) the SDV simulation view in Cartesian coordinates; and in (d) the ADS perception (circles represent the lidar simulation, with the ego located at their center).

TABLE II
SIMULATION PARAMETERS FOR SDV BEHAVIOR AND RESULTS

	SDV Config			Observed				
#	Δd_a	Δd_t	Δv_t	Δd_a	Coll.	v_{SDV}	v_{Ego}	maneuver
7	2	-2	-5	2.07	n	-	-	-
8	2	-5	-5	2.05	y	10.89	13.16	emergency stop
9	5	-5	-5	5.49	n	-	-	-
10	5	-5	-10	5.50	n	-	-	stop
11	5	-10	-10	5.60	y	7.60	12.15	-

We found that using another SDV instance as placeholder for ego enables a rapid iterative development of test scenarios. The iterations are needed to ensure the correct behavior of the cutting-in vehicle and select reasonable ranges of test parameters, before running the more time-consuming simulation with ego controlled by the ADS. Finally, the experiment results also highlight the importance of being able to plan the SDV maneuver trajectories dynamically and influence their shape via parameters.

D. Scalability (RQ4)

We evaluate the SDV model scalability to see if it can support scenarios with heavy traffic. To support such scenarios, the model must be able to scale traffic density without any significant degradation of the simulation performance or the quality of the planned trajectories.

1) Reference Implementation and Performance Requirements: The experiment uses the reference implementation (Sec. IV). To provide a sufficient simulation update rate, the SDVPlanner instances target a planning rate of 3 Hz, and the TrafficSim process targets updating the position of all vehicles at 30 Hz. Planning is a highly time-critical task, which needs to be executed within its target period of 333 ms (3 Hz). If a vehicle misses the target time to generate its plan, it likely affects the quality of its trajectory and the resulting motion.

TABLE III
PERFORMANCE WITH MULTIPLE SCENARIO CONFIGURATIONS

id	vehicles	obstacle	coll.	TRC	max tick	TPRC	max plan
3	10	inactive	0	98.44%	0.042s	100.00%	0.333s
4	15	inactive	0	92.28%	0.055s	99.94%	0.338s
5	20	inactive	0	61.90%	0.052s	100.00%	0.333s
8	10	active	0	98.49%	0.041s	99.94%	0.338s
9	15	active	0	78.58%	0.052s	99.80%	0.340s
10	20	active	0	55.65%	0.065s	99.91%	0.343s

Furthermore, a long overrun can affect the SDV model's ability to predict the traffic state, resulting in sub-optimal trajectories and even unintended collisions. The state transformation and update is performed by TrafficSim and must be completed for all vehicles within 33 ms (30 Hz). A small exceedance, if consistent, may be acceptable, as it would slightly reduce the update frequency below 30 Hz without compromising the actual vehicle motion. The experiment is executed on an Intel Core i7-6800K (3.40 GHz), with 32 GB RAM and Ubuntu 18.04.5.

2) Scenarios: We use two long-running scenarios, each with a two-minute duration, and vary the number of SDVs, up to 20. In each scenario, the SDVs travel in one lane and form a virtual platoon, simulating heavy traffic. In scenario A, the SDVs travel without any disturbance, and in scenario B, they need to steer to avoid a static obstacle in their lane. When running scenario B, the object collision checking for obstacle avoidance is activated. The purpose of scenario B is to show the impact of object collision checking on scalability, since it is computationally expensive. Each vehicle travelling behind another one is expected to observe a safe following distance.

3) Metrics: We evaluate the adherence to the target rates using the following metrics: *Target Rate Compliance* (TRC), defined as the % of simulation (execution) ticks from all vehicles that adhere to the target tick time of 33 ms (30 Hz); the *maximum tick time*; the *Target Planning Rate Compliance* (TPRC), defined as the % of planning cycles from all vehicles that adhere to the target time of 333 ms (3 Hz); and the *maximum planning time*.

Results: Both scenarios with up to 20 vehicles execute successfully, without any collisions or lane boundary violations. The planning adheres to the target rate with almost 100%, with 99.8% being the worst case (Table III). However, execution deteriorates significantly between 10 and 15 vehicles, especially when the collision checking is active, plunging from 98.49% to 78.58%. Such a deterioration of the target rate to update the state of all vehicles may introduce inconsistencies and confuse the ADS under test, such as inducing significant errors in its object tracking system. However, reducing the update rate from 30 Hz to 20 Hz results in near perfect adherence for up to 20 vehicles when no collision checking is used and up to 15 vehicles with the collision checking active. Thus, scenarios with up to 10 SDV instances are easily handled by the reference implementation, and scaling to 20 instances requires reducing the update rate. For scenarios requiring more vehicles, the traffic can mix SDV instances for interactions with ego and vehicles based on predefined trajectories, with negligible computing cost.

VI. Conclusion

We presented the SDV model to express and execute scenarios for ADS scenario-based testing in simulation. The model encapsulates driver and vehicle as a single entity with an architecture that provides a user-oriented language to coordinate the vehicle behavior and motion planning that optimizes for realism and achieving the scenario test objective. In particular, behavior trees provide a high-level description of discrete decisions, with a high-level of abstraction and parameterization to support controllability and reuse. Furthermore, dynamic trajectory planning allows for flexible adaptation of the SDV trajectories to different road geometries and achieving the test objective despite varying and unpredictable ego behaviors.

The evaluation shows that our proposed approach supports effective test scenario development and execution using the NHTSA vehicle-to-vehicle pre-crash scenarios, with high internal reuse of over 80 %. The analysis also shows that the majority of scenarios require dynamic trajectory planning, benefiting from the SDV model compared to the baseline. The evaluation demonstrates the ability of the SDV model to faithfully reproduce real-world vehicle behavior, including different driving styles by adjusting parameters, with an average spatio-temporal trajectory distance of 1.24 m. It also allows custom behaviors and misbehaviors by adding dedicated conditions and maneuvers. The reference implementation demonstrates that the SDV model scales to execute scenarios with 10-20 highly interactive vehicles, and additional optimizations, such as reducing the number of sampled trajectories for vehicles farther away from ego, allow for further scaling. Finally, the application of the SDV model to test WISE ADS in the cut-in scenario confirms the usefulness of the model and offers practical insights. Among others, the ability to control the shape of the cut-in trajectories uncovers the varied response of the ADS to different trajectories, showing that not only the target gap and velocity, but also the acceptance gap impact the likelihood of a collision. Furthermore, using an SDV model instance in place of ego helps accelerate the development of the test scenario and parameter selection to tune the trajectories of the agent that challenges ego.

In future work, we plan several model extensions and new capabilities that exploit the model. First, we plan to expand the model with new maneuvers and configuration options based on additional scenarios, harvested from a wider range of naturalistic data, such as the additional locations in the Waterloo dataset [55] and the multi-country INTERACTION dataset [58]. We plan to improve the auto-calibration process and further automate creation of behavior trees and their parameterization to approximate the naturalistic traffic. We will also expand the behavior trees and maneuvers for interaction with pedestrians [44]. Finally, we plan to exploit the model in generating new scenarios by injecting road-user misbehaviors into behavior trees, such as simulating distraction [59] and ignoring occlusions [60]. The SDV model implementation and toolset to design and run scenarios is publicly available and can be integrated with any simulation environment via co-simulation.

References

[1] *Taxonomy and Definitions for Terms Related to on-Road Motor Vehicle Automated Driving Systems*, Standard SAE J3016, SAE, 2014.

[2] W. G. Najm, J. D. Smith, and M. Yanagisawa, "Pre-crash scenario topology for crash avoidance research," Dept. U.S. Dept. Transp., NHTSA, Washington, DC, USA, Tech. Rep. DOTHS810767, 2007.

[3] A. Wasowski and T. Berger, *Domain-Specific Languages: Effective Modeling, Automation, and Reuse*. Cham, Switzerland: Springer, 2023.

[4] R. Lämmel, *Software Languages*. Cham, Switzerland: Springer, 2018.

[5] R. Queiroz, T. Berger, and K. Czarnecki, "GeoScenario: An open DSL for autonomous driving scenario representation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 287–294.

[6] *OpenScenario*. Accessed: May 1, 2022. [Online]. Available: https://www.asam.net/standards/detail/openscenario

[7] *Measurable Scenario Description Language (M-SDL)*. Accessed: May 1, 2022. [Online]. Available: https://www.foretellix.com/open-language/

[8] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 368, no. 1928, pp. 4585–4605, Oct. 2010.

[9] J. A. Michon, *A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?* Boston, MA, USA: Springer, 1985, pp. 485–524.

[10] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. Boca Raton, FL, USA: CRC Press, 2018.

[11] R. Ghzouli, T. Berger, E. B. Johnsen, A. Wasowski, and S. Dragule, "Behavior trees and state machines in robotics applications," *IEEE Trans. Softw. Eng.*, vol. 49, no. 9, pp. 4243–4267, Jan. 2023.

[12] C. Kaner, J. Bach, and B. Pettichord, *Lessons Learned in Software Testing*. New York, NY, USA: Wiley, 2001.

[13] *Road Vehicles—Functional Safety*, Standard ISO/FDIS 26262:1994, ISO, Geneva, Switzerland, 1994.

[14] *Road Vehicles—Safety of the Intended Functionality*, Standard ISO-21448:2022, ISO, Geneva, Switzerland, 2022.

[15] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 719–726.

[16] 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," *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 6, pp. 1243–1262, Dec. 2011.

[17] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2016, pp. 63–74.

[18] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, pp. 1016–1026.

[19] *OpenDRIVE*. Accessed: May 1, 2022. [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[20] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, 2019, pp. 63–78.

[21] X. Zhang, S. Khastgir, and P. Jennings, "Scenario description language for automated driving systems: A two level abstraction approach," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2020, pp. 973–980.

[22] B. Schütt, T. Braun, S. Otten, and E. Sax, "SceML: A graphical modeling framework for scenario-based testing of autonomous vehicles," in *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst.* New York, NY, USA: ACM, Oct. 2020, pp. 114–120.

[23] J. Sewall, D. Wilkie, P. Merrell, and M. C. Lin, "Continuum traffic simulation," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 439–448, May 2010.

[24] Q. Chao et al., "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 287–308, Feb. 2020.

[25] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transp. Res. B, Methodol.*, vol. 15, no. 2, pp. 105–111, 1981.

[26] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model MOBIL for car-following models," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1999, no. 1, pp. 86–94, Jan. 2007.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.

[28] D. C. Gazis, R. Herman, and R. W. Rothery, "Nonlinear follow-the-leader models of traffic flow," *Operations Res.*, vol. 9, no. 4, pp. 545–567, Aug. 1961. [Online]. Available: https://www.jstor.org/stable/167126
[29] V. Milanés and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," *Transp. Res. C, Emerg. Technol.*, vol. 48, pp. 285–300, Nov. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0968090X14002447
[30] P. G. Gipps, "A model for the structure of lane-changing decisions," *Transp. Res. B, Methodol.*, vol. 20, no. 5, pp. 403–414, Oct. 1986. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0191261586900123
[31] S. Moridpour, M. Sarvi, and G. Rose, "Modeling the lane-changing execution of multiclass vehicles under heavy traffic conditions," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2161, no. 1, pp. 11–19, Jan. 2010.
[32] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to simulate realistic multi-agent behaviors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10395–10404.
[33] L. Bergamini et al., "SimNet: Learning reactive self-driving simulations from real-world observations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 5119–5125.
[34] M. Igl et al., "Symphony: Learning realistic and diverse agents for autonomous driving simulation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 2445–2451.
[35] Z. Zhong et al., "Guided conditional diffusion for controllable traffic simulation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 3560–3566.
[36] S. Suo et al., "MixSim: A hierarchical framework for mixed reality traffic simulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 9622–9631.
[37] C. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov, "MotionDiffuser: Controllable multi-agent motion prediction using diffusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 9644–9653.
[38] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Uncertainty estimation for cross-dataset performance in trajectory prediction," in *Proc. ICRA Fresh Perspect. Future Auto. Driving Workshop*, 2022.
[39] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wąsowski, "Behavior trees in action: A study of robotics applications," in *Proc. 13th ACM SIGPLAN Int. Conf. Softw. Lang. Eng.*, Nov. 2020, pp. 196–209.
[40] M. Olsson, "Behavior trees for decision-making in autonomous driving," M.S. thesis, Dept. Comput. Sci., Royal Inst. Technol., Stockholm, Sweden, 2016.
[41] T. G. Tadewos, L. Shamgah, and A. Karimoddini, "Automatic safe behaviour tree synthesis for autonomous agents," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 2776–2781.
[42] M. Jamal and A. Panov, "Adaptive maneuver planning for autonomous vehicles using behavior tree on Apollo platform," in *Artificial Intelligence XXXVIII*, M. Bramer and R. Ellis, Eds. M. Bramer and R. Ellis, Switzerland: Springer, 2021, pp. 327–340.
[43] S. Kang, H. Hao, Q. Dong, L. Meng, Y. Xue, and Y. Wu, "Behavior-tree based scenario specification and test case generation for autonomous driving simulation," in *Proc. 2nd Int. Conf. Intell. Technol. Embedded Syst. (ICITES)*, Sep. 2022, pp. 125–131.
[44] S. Larter, R. Queiroz, S. Sedwards, A. Sarkar, and K. Czarnecki, "A hierarchical pedestrian behavior model to generate realistic human behavior in traffic simulation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2022.
[45] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *Proc. IEEE Int. Conf. Robot. Autom.*, Jul. 2010, pp. 987–993.
[46] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017, *arXiv:1708.06374*.
[47] *Taxonomy and Definitions for Terms Related to Automated Driving System Behaviors and Maneuvers for On-Road Motor Vehicles*, Standard SAE J3164, SAE Int., 2018.
[48] K. Czarnecki, "Automated driving system (ADS) task analysis—Part 2: Structured road maneuvers," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., 2018.
[49] R. Queiroz, "Scenario modeling and execution for simulation testing of automated-driving systems," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2022.
[50] A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto, "Local path planning and motion control for AGV in positioning," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots Syst.*, Sep. 1989, pp. 392–397.
[51] F. Poggenhans et al., "Lanelet2: A high-definition map framework for the future of automated driving," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1672–1679.
[52] *Robot Operating System (ROS)*. Accessed: May 1, 2022. [Online]. Available: https://www.ros.org/
[53] W. Frakes and C. Terry, "Software reuse: Metrics and models," *ACM Comput. Surv.*, vol. 28, pp. 415–435, Jun. 1996.
[54] (2020). *Waymo Safety Report*. [Online]. Available: https://waymo.com/safety/
[55] *Waterloo Multi-Agent Traffic Dataset*. Accessed: May 1, 2022. [Online]. Available: http://wiselab.uwaterloo.ca/waterloo-multi-agent-traffic-dataset
[56] K. Czarnecki, "Automated driving system (ADS) task analysis—Part 1: Basic motion control tasks," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., 2018.
[57] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, no. 3, pp. 267–289, Nov. 2006.
[58] W. Zhan et al., "INTERACTION dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," 2019, *arXiv:1910.03088*.
[59] J. van Lint and S. Calvert, "A generic multi-level framework for microscopic traffic simulation—Theory and an example case in modelling driver distraction," *Transp. Res. B, Methodol.*, vol. 117, pp. 63–86, Nov. 2018.
[60] M. Kahn, A. Sarkar, and K. Czarnecki, "I know you can't see me: Dynamic occlusion-aware safety validation of strategic planners for autonomous vehicles using hypergames," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 11202–11208.

Rodrigo Queiroz received the master's degree in computer science from Federal University of Minas Gerais (UFMG), Brazil. He is currently pursuing the Ph.D. degree with the Faculty of Engineering, University of Waterloo, Canada. He is part of Waterloo Intelligent Systems Engineering (WISE) Laboratory. His research interests include validation and verification of autonomous driving systems, safety-critical systems, motion planning, vehicle simulation, and traffic simulation.

Divit Sharma is currently pursuing the bachelor's degree in computer science with the University of Waterloo, Canada. His past work includes researching high-definition context maps for autonomous vehicles and developing interactive multi-agent simulation systems with WISE Laboratory. He also has internship experience with Ike Robotics, a California-based self-driving startup, and Behaviour Interactive, a Montreal-based game development studio. His research interests and experience include robotics simulation and game development.

Ricardo Caldas received the master's degree in computer science from the University of Brasília, Brazil, in 2019. He is currently pursuing the Ph.D. degree with the Chalmers University of Technology, Sweden. Lately, he has been investigating the interplay between control theory and software engineering principles for the engineering of resilient autonomous systems, including mobile robots and self-driving vehicles. His research interests include verification of autonomous systems.

Krzysztof Czarnecki co-lead the development of the first ADS tested on public roads in Canada in 2018. He is currently a Professor of electrical and computer engineering and a University Research Chair with the University of Waterloo, where he heads the Waterloo Intelligent Systems Engineering (WISE) Laboratory. He is a Leading Expert in the safety of automated driving systems (ADS), with a focus on assuring the safety of driving behavior and machine-learned functions. As a member of standardization committees, he has contributed to ISO 21448 (second edition), ISO 8800 (under development), and SAE J3164. He received the Premier's Research Excellence Award in 2004 and the British Computing Society in Upper Canada Award for Outstanding Contributions to IT Industry in 2008. He has also received eight best paper awards, two ACM Distinguished Paper Awards, and four most influential paper awards.

Sergio García received the master's degree in electronics from the University of Alcalá, Spain, in 2016, and the Ph.D. degree from the University of Gothenburg, Sweden, in September 2021. His research interests include robotics software engineering, striving to understand the complexity of the domain and analyzing its characteristics, and challenges to develop solutions based on them.

Thorsten Berger received the Ph.D. degree in computer science from the University of Leipzig, Germany, in 2013. Thereafter, he worked as a Post-Doctoral Fellow with the University of Waterloo, Canada, and the IT University of Copenhagen, Denmark, and an Associate Professor with the Chalmers University of Technology, Gothenburg, Sweden. He is currently a Professor of computer science with Ruhr University Bochum, Germany. His research interests include automating software engineering for the next generation of intelligent, autonomous, and variant-rich software systems, exploring new ways of software creation, analysis, and evolution. He received the Fellowship from the Royal Swedish Academy of Sciences and the Wallenberg Foundation, one of the highest recognitions for researchers in Sweden. He received two best-paper and two most influential paper awards, and his service was recognized with distinguished reviewer awards at A* conferences.

Patrizio Pelliccione received the Ph.D. degree in computer science from the University of L'Aquila, Italy. Thereafter, he worked as a Senior Researcher with the University of Luxembourg, Luxembourg; an Assistant Professor with the University of L'Aquila; and an Associate Professor with the Chalmers University of Technology, Gothenburg, Sweden, and the University of L'Aquila. He is currently a Professor of computer science and the Director of the computer science area of the Gran Sasso Science Institute (GSSI), Italy. He is also an Adjunct Professor with the University of Bergen, Norway. He is very active in European and national projects. In his research activity, he has collaborated with several companies. His main research interests include software engineering, software architecture modeling and verification, autonomous systems, and formal methods. He has been on the organization and program committees for several top conferences and he is a reviewer of top journals in the software engineering domain. More information is available at http://www.patriziopelliccione.com.