

A Driver-Vehicle Model for ADS Scenario-based Testing

Rodrigo Queiroz, Divit Sharma, Ricardo Caldas, Krzysztof Czarnecki, Sergio Garcia, 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 Model (SDV) 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 layered architecture of the model leverages behavior trees to express high-level behaviors in terms of lower-level maneuvers, affording multiple driving styles and reuse. Further, optimization-based maneuver planner guides 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 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.

I. INTRODUCTION

Testing automated driving systems (ADS) requires simulating a wide range of operating scenarios to ensure their safety and conformity to traffic regulations and industry standards. As the responsibility for the driving task shifts from the human driver to the ADS with increasing levels of automation [1], the system is required to handle interactions with the other road users, in particular with human-operated vehicles (HVs). Scenarios for verification and validation must reflect how these dynamic interactions between humans and the subject system can unfold in real traffic.

Figure 1 shows an example based on the National Highway Traffic Safety Administration’s (NHTSA) pre-crash scenario catalog [2]. In this scenario, the vehicle operated by the subject ADS (aka Ego vehicle) moves in traffic when V_2 cuts in front of it, leading to a near-collision. This cut-in maneuver likely triggers a reaction by several other close-by vehicles, and the Ego’s reaction strongly influences how the scenario unfolds. Testing the ADS capabilities in collision avoidance in such scenarios requires models that are able to represent

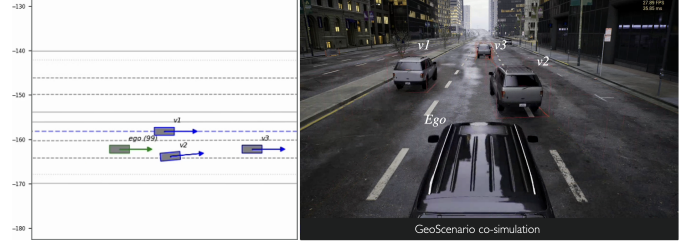


Fig. 1. A challenging interaction between Ego and HVs based on a pre-crash scenario from NHTSA [2] and using the SDV model in simulation: (left) the SDV Model as V_2 performs the cut-in maneuver targeting Ego, and (right) a high-fidelity co-simulator renders the scene.

and simulate the dynamics of the traffic, including the HVs and their interactions with Ego.

Many Domain-Specific Languages (DSLs) for scenario-based testing have emerged to support the scenario design and representation. As such, they include models for HVs and allow testers to define the HVs’ behavior and guide their interactions with Ego when executed by simulation tools during testing. However, these languages are typically limited to relatively simple models, such as using an event-based orchestration mechanism that directly manipulates the simulated vehicle’s primitive attributes [3], [4]. In such a model, a condition may trigger a direct assignment to the vehicle’s position and velocity. The focus of these languages is declaratively specify “what a vehicle must do” and “where it must be” in a particular scenario, but without detailing “how” it moves. Consequently, this approach relies on a simulation tool to implement the actual vehicle behavior, either by translating the high-level definition to the vehicle simulation model internal (and often implicit) to the tool or by forcing the change in vehicle state, while disregarding the limitations of a real vehicle in traffic. The potential mismatch between what is specified by the language and the actual vehicle behavior compromises test reproducibility across simulation environments and validity of the test results.

An alternative approach is to use catalogs of predefined trajectories (PDTs), either extracted from traffic or designed by testers using a variety of mathematical functions or polylines, and orchestrate them via triggers [5], [3], [6], [7]. This approach brings realistic trajectories to HVs. However, the traffic is a dynamic system with complex interactions amongst participants, as the example in Figure 1 illustrates. Further, ADS sub-systems often exhibit some non-determinism [8], which leads to non-determinism in the overall ADS behavior. A predefined behavior at the level of trajectories is unlikely to adequately account for the wide range of possible reactions

R. Queiroz, D. Sharma, and K. Czarnecki are with the University of Waterloo, Canada. R. Caldas, S. Garcia and T. Berger are with Chalmers | University of Gothenburg, Sweden. P. Pelliccione is with Gran Sasso Science Institute (GSSI), Italy

in a dynamic interaction between the ADS and the traffic and also achieve reproducibility. Further, such PDTs are typically specific and limited to certain road geometries.

To fill this gap between scenario design and execution, this paper contributes a model to specify and simulate realistic HV behavior in ADS scenario testing, while affording high expressiveness, execution accuracy, scalability, and reuse. We refer to our model as the GeoScenario Simulated Driver-Vehicle model (or simply SDV model). It extends the base scenario-definition language GeoScenario [5] with HVs as dynamic agents in both scenario representation and simulation execution. The SDV model encapsulates driver and vehicle as a single entity and is based on two main layers: Behavior and Maneuvers. The Behavior layer is a higher level of abstraction aimed at coordinating the vehicle behavior using an explicit, user-oriented DSL. The Maneuver layer generates the actual vehicle trajectories and is designed to approximate how real vehicles drive on the road and optimize for the scenario test objective. Both layers are highly configurable to allow multiple driving styles, subject to the physical limitations and nonholonomic properties [9] of a typical road vehicle operating on structured roads.

We evaluate the SDV model in terms of (i) scenario design effectiveness, which includes expressiveness, execution accuracy, and reuse, using NHTSA pre-crash scenarios; (ii) its motion realism in comparison to naturalistic urban traffic; (iii) its scalability with traffic density; and (iv) its practical applicability to test an actual ADS. The results show that the model is able to successfully express and accurately execute all eighteen NHTSA vehicle-to-vehicle pre-crash scenarios, except one scenario variant. In comparison, only four scenarios are effectively expressible using PDTs, which is our baseline. In particular, lane-change and crossing-path scenarios are highly unpractical with the baseline and benefit the most from the new model. The SDV model also results in high-levels of internal reuse, achieving over 80% on average for the NHTSA scenarios. Using naturalistic traffic data from a busy urban intersection, we show how the motion of the simulated vehicles is similar to that of the real vehicles when operating under the same conditions. In the best performing scenarios (first quartile), the synthetic trajectories from the model are almost indistinguishable from empirical vehicles, with an average spatio-temporal trajectory distance of less than 55 cm (while the average from all scenarios is 1.24 m). We also show that the model scales in scenarios with up to 10-20 simultaneous highly-interactive vehicles, while maintaining simulation quality and consistency in the driving task. Finally, we demonstrate the model’s applicability in ADS scenario-based testing with a real subject system, and its ability to reveal collision scenarios that cannot be expressed using the baseline.

A reference implementation of the model is available to the research community with a full scenario simulation tool that is ready to be integrated in co-simulation with any simulation environment. The tool-set and additional model documentation is available in the companion website.¹

¹<https://geosenario2.readthedocs.io>

II. BACKGROUND AND RELATED WORK

A. Scenario-Based Testing

The usage of the term *scenario* varies depending on the author or discipline [10]. In our work, we rely on Ulbrich et al., who analyzed the concept across multiple disciplines and proposed a definition based on requirements for ADS testing:

“A *scenario* describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions and events as well as goals and values may be specified to characterize this temporal development in a scenario.” [11]

The *scenario-based design paradigm* considers scenarios as a central concept to support the development of complex systems throughout the entire lifecycle, from helping to derive initial requirements to validating the system during the testing [12]. Kaner et al. [13] 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 [14] and ISO/PAS 21448 [15] guide the development of safety-critical electrical/electronic vehicle systems and mandate the use of scenarios as part of validation activities.

Scenarios can be defined at different levels of abstraction. Menzel et al. [16] propose three such levels within the ISO 26262 systems engineering process: (i) functional scenarios, being high-level natural language descriptions in the concept phase, (ii) logical scenarios, being semi-formal models with state space parameters and their ranges in the development phase, and (iii) concrete scenarios, represented in an executable format with concrete values in the test phase. In this work we focus on the levels (ii) and particularly (iii), since they are closer to the level of detail required for simulation.

Researchers and engineers design scenarios based on expert knowledge and the common traffic situations the ADS must be able to cope with, or by reproducing and augmenting situations collected from traffic databases. For example, CommonRoad [6], a benchmark for motion planners, provides scenarios extracted from NGSIM data [17]. 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. [18], [19] use evolutionary optimization methods combined with surrogate model learning to find crash scenarios. Given a parameterized scenario space, the evolutionary search produces subsequent generations of parameter values with increasing criticality based on how the system performs under simulation. Similar methods are also used to test autonomous parking systems [20].

B. Scenario Representation and Driver Behavior

Multiple tool-independent DSLs have emerged in recent years, providing a formal definition of scenario structure, behavior, test conditions, and pass/fail criteria to support scenario-based design and 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. We focus our discussion on how some of the prominent languages specify this behavior.

OpenScenario [3] 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 [21], 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) [4] 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 represents the vehicle behavior at the logical abstraction level and supports generating concrete scenarios by picking random values while obeying the constraints.

Other formats are Scenic [22], Scenario Description Language (SDL) [23], and SceML [24]. 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 models running in simulation to handle the execution.

Finally, our language 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 PDTs specific to the road. The SDV model extends GeoScenario with interactive and flexible driver behavior.

C. Models for Traffic Simulation

Traffic simulation has a wide range of applications and can be used to generate the motion of vehicles at various levels of detail. 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 [25]. Since they are not suitable for street-level motion and interactions between vehicles, they cannot be used for ADS testing.

In contrast, microscopic traffic models can generate vehicle motion and interactions at the individual vehicle level at the cost of limited scalability [26]. 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 [27], [28], [29]. 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). The supported behavior is rigid, and it is hard or impossible to encode the fine-grained details that replicate human driving.

Some micro-models target a particular maneuver, for example, a lane-change model encoding the accelerating/decelerating behavior based on surrounding vehicles [30], or the driver’s decision and conditions that trigger the maneuver [31]. While these models better capture details at the maneuver level and allow testers to cover a range of parameters, they are suitable for testing specific functions and subsystems (for instance, testing the ADS emergency break) in a very constrained environment. They do not cover the complexity of the full driving task required for scenarios in system-level testing. Attempting to combine multiple maneuver-specific models into a simulated agent would be challenging since every model has its own set of assumptions and constraints.

A different approach is to learn models directly from data. Krajewski et al. [32] build a lane-change model by using unsupervised learning to extract primitive attributes from lane changes observed in the highD dataset [33]. The resulting model can then be used to generate synthetic lane change maneuver trajectories in new scenarios. The main limitation in a purely data-driven approach is the inherent bias in the data used to build the model. While most available datasets cover common situations, driver mistakes and safety-critical scenarios are rarely captured in such data sources. Also, they can capture the diversity of driving styles in one road environment, but are difficult to generalize to other environments. TrafficSim [34] uses a hybrid approach to build a model by learning from naturalistic data and also encoding common-sense rules to guide the driving task. This hybrid approach shows promising results in imitating the human-driving and its diversity of driving-styles, while still reacting to traffic. However, agents are not fully controllable and cannot be adapted to new scenarios by freely assigning new goals or styles based on a new scenario design.

Overall, traffic simulation models are built for simulated agents to drive independently without collisions. As a result, they tend to limit the controllability by the tester. For scenario-based testing, the simulation model must serve the scenario goals. If the evolution between scenes is not controllable, and the agents are not guaranteed to reach the target situation (as specified by parameters, such as a time gap for a maneuver), even the most realistic traffic simulation will not be suitable. Thus, scenario-based testing requires expressiveness, controllability, and realistic behavior.

D. Behavior Trees

Behavior Trees (BTs) is a control architecture that emerged from the gaming industry and plays a significant role in

robotics. Its design aims to address the shortcomings of finite state machines and their variations, and provide improved modularity, reusability, scalability, and readability [35]. The tree-like structure of BTs conveys a hierarchical understanding of how composition operators coordinate elemental behaviors, such as maneuvers, to perform the desired overall behavior. There are two types of tree nodes: control nodes and behavior nodes.

Control nodes, or operators, are responsible for coordinating the execution of their children nodes. In the classical architecture, there are three operators: the *fallback*, *sequence*, and *parallel* nodes. The *fallback* operator resembles the logic operator *or*, but with a short-circuit semantics. This node 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 succeed. The *sequence* operator resembles the short-circuit logic operator *and*. This node 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. Last, 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.

Behavior nodes are responsible for encoding domain-specific tasks, which BTs then compose into the overall desired behavior. These nodes are the interface to the concrete low-level behaviors. A behavior node returns success when its task succeeds, or ‘running’ while the task is under execution, or failure when the task fails. The expressive, modular, and interpretable representation of BTs makes them suitable for representing driver behavior in test scenarios.

III. SDV MODEL DESIGN AND ARCHITECTURE

We now introduce the structure of the GeoScenario Simulated Driver-Vehicle model (SDV) and its components. For simplicity and scalability, the model combines driver and vehicle as a single entity, abstracting away driver inputs, such as steering angle, braking, and throttle. The resulting behavior (model output) is the vehicle movement: position, velocity, acceleration, and heading at each point in time, referred to as *VehicleState*.

We design a layered architecture inspired by the seminal works of Michon [36] and Boer et al. [37], which propose a hierarchical structure of the driving task with strategic (e.g., route selection), tactical (maneuver selection), and control (maneuver execution) levels. According to Boer et al. [37], the driving task can be characterized as a goal-directed behavior, where the goal is typically composed of “a set of higher-level needs whose interaction affects how drivers orchestrate the set of observable low-level driving tasks.” Our model architecture targets the tactical and control levels, with a focus on the ease of use for testers to express the overall tactical behavior in the behavior layer, and the remaining two layers providing reusable maneuver planning and execution in (see Figure 2):

- The *Behavior Layer* structures the driver tactical behavior. It breaks down the complex driving task into smaller

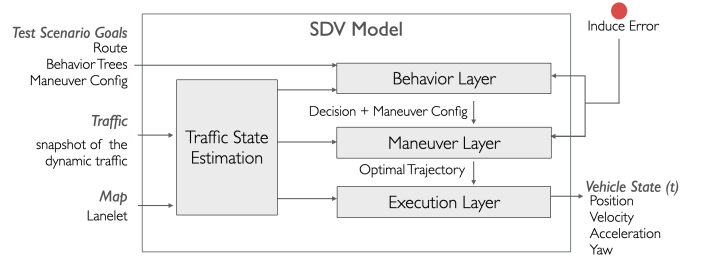


Fig. 2. GeoScenario SDV model overview. The model combines driver and vehicle into a single entity and outputs the resulting vehicle state in simulation.

tasks and coordinates maneuvers via a user-oriented DSL that consists of BTs and elemental maneuvers.

- The *Maneuver Layer* is responsible for trajectory planning. It turns a maneuver decision from the BTs into a viable motion profile based on the road, the surrounding actors, and the maneuver parameters.
- The *Execution Layer* is responsible for trajectory execution in simulation. The result is the vehicle state as the output from the SDV model to the simulation.

A. World Model and Vehicle Representation

We assume that a simulation holds a ground-truth bird’s-eye-view representation of the world in two-dimensional Cartesian coordinates, including all the static elements of the scenario (road geometry and network, regulatory elements, static objects) and a simulation state for all dynamic elements (pedestrians, vehicles, and regulatory element state). Amongst the vehicles, Ego represents the vehicle under test, and its state is determined by an Ego vehicle model controlled by the ADS under test. The remaining vehicles can be SDV model instances, vehicles simulated by an unknown model, or vehicles controlled by a human during test. In the world representation, they are equal traffic participants with a body and a physical presence. All dynamic actors are perceived by each other through their type and the state over time:

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

B. Vehicle Motion and Traffic State Estimation

The vehicle *driving mission* is defined by a *start state* and a *route* assigned in GeoScenario as part of the scenario design. The *scenario route* is a sequence of points to be visited (in order), and its last point is the *goal*. From the Lanelet Map routing graph [38], we generate a sub-map of connected lanelets visiting each route point on a shortest path, if such a route exists in the road network. With all the lanelets in this route, a *global path* is formed by a sequence of points from the lane centre line. It is used to guide the vehicle motion and its progress along the route to the goal point. If the vehicle deviates from this route (after a scenario event), a new route is generated from the last state to the remaining route points.

The SDV parameterizes and plans its motion in its dynamic Frénet reference frame [39], rather than the global Cartesian

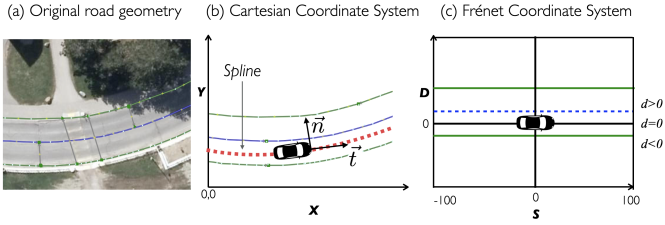


Fig. 3. Road Geometry and vehicle displacement from original coordinates are transformed into Frénet Frame using the tangential and normal vectors \vec{t} , \vec{n} from the lane centre line (shown in red).

coordinates of the simulation environment. This is motivated by the fact that safety requirements on the motion of an on-road vehicle are typically specified relative to its Frénet frame derived from the local lane geometry (e.g., see [40]). For a given global path segment surrounding the vehicle position (which aligns with the local lane centre line), we fit a spline, which we refer to as the *reference path* and use it for the frame transformation. The reference frame (*Frénet frame*) is given by the tangential \vec{t} and normal \vec{n} vectors at the point along the arc length of this path that is closest to the vehicle. The resulting frame's S axis represents the longitudinal displacement along this path, and the D axis represents the lateral displacement (Figure 3). The vehicle motion is represented by a trajectory that combines two independent polynomial functions $S(t)$ and $D(t)$ in the Frénet frame, and T as the total time (2). Velocity and acceleration are the first and second derivatives, respectively, yielding the longitudinal and lateral state (3):

$$\text{Trajectory} = [S, D, T] \quad (2)$$

$$\begin{aligned} \text{VehicleState}_{\text{Frénet}}(t) &= [S(t - t_0), \dot{S}(t - t_0), \ddot{S}(t - t_0), \\ &D(t - t_0), \dot{D}(t - t_0), \ddot{D}(t - t_0)] \\ &\text{for } 0 \leq t - t_0 \leq T \end{aligned} \quad (3)$$

The SDV trajectory is planned by the Maneuver Layer (Section III-D) in the Frénet frame, and the current SDV state is then translated to the Cartesian frame state (1) by the Execution Layer at each simulation cycle.

The *Traffic State Estimation* is a support task transforming the state of the static and dynamic elements, including Ego, that surround the reference path into the SDV's reference frame (see Figure 2). Since both the SDV and the traffic are moving, the task predicts the state of the world for the next point in time when the Maneuver Layer will generate a new trajectory for the SDV. This predicted traffic snapshot in Frénet frame, along with the map, represents a simplified representation of the world, which is then used for decision making and trajectory generation.

C. Behavior Layer

Given a *route* and the *estimated traffic state*, this layer performs the decision making, modeled using BTs. In each execution cycle, the SDV executes the main BT, which is a directed rooted tree with internal nodes being operators controlling the flow and leaf nodes being either (i) conditions to be evaluated (based on the traffic state), (ii) decisions that

start (or end) maneuvers, or (iii) references to sub-trees. Figure 4 shows a graphical representation of two sample trees, with the left one being the main tree, and the right one being a sub-tree referred to from the main one. The main tree first checks the sequence node, which tests whether the vehicle reached its goal; if this test succeeds, then the tree will issue a decision to stop (stop maneuver). Otherwise the vehicle continues driving by executing the sub-tree on the right, which first checks if there is a lead vehicle, in which case it issues the follow maneuver; otherwise it commands cruising at a set velocity.

The key motivation to use BTs is to provide test engineers with an easy-to-use means to specify scenario-specific SDV behavior. Rather than using a full-fledged behavior planner for an entire ODD (Operational Design Domain) to control an SDV, they can specify scenario-specific “micro-planners” by composing, parameterizing, and, if needed, customizing reusable BTs, expressed in a user-oriented DSL. This is possible since the driving task can be broken into smaller sub-tasks (e.g. road following, handling traffic lights, switching lanes), each encapsulated in a separate tree, and stored in a library. Test engineers can select the BTs representing the behaviors needed for a test scenario from the library and easily compose them through the sub-tree reference mechanism (as in Figure 4). They can also modify the driving style of an SDV by modifying BT parameters, and inject misbehaviors, such as dangerous cut-ins, by replacing normal maneuvers with BTs that represent such misbehaviors. Compared to using a full-fledged behavior planner, this approach shields the engineers from the decision logic needed to support other scenarios and thus eliminates unnecessary complexity. Section V shows a practical BT and how it captures a range of behaviors via parameters (Figure 13). Further examples are available in the online documentation.

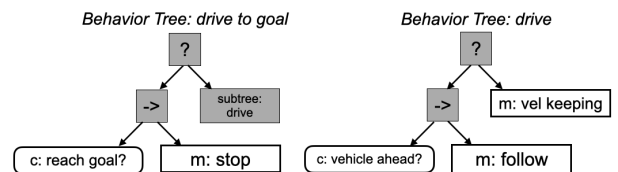


Fig. 4. Graphical representation of a sample SDV BT structuring the decision-making with conditions (c) and maneuvers (m). “?” is the *fallback* operator (short-circuit or), and \rightarrow is the *sequence* operator (short-circuit and).

D. Maneuver Layer

This layer is responsible for the actual vehicle motion on the road. It receives a maneuver decision from the Behavior Layer and implements it by generating a feasible trajectory, which can be performed by a real vehicle. To achieve this, the model is bounded by a set of feasibility constraints respecting the vehicle dynamics. This is an important distinction from the behavior models assumed by the scenario definition languages and traffic simulators discussed in Section II.

A maneuver is “*goal-oriented vehicle motion control behavior undertaken by a human driver or an ADS in order to achieve a specific result/outcome.*” [41] A key element of the result is the target state of the vehicle, such as reaching a

desired velocity or an adjacent lane and challenging Ego is a specific way. Furthermore, the maneuver must account for the road geometry, other traffic, including Ego, and the desired driving style, according to the test objectives.

Each maneuver is defined through a set of trajectory characteristics relative to the road environment. A maneuver exposes a set of parameters to control it according to scenario objectives. We use existing maneuver catalogs [41], [42] and implement a subset to support the evaluation in Section V: velocity keeping, vehicle following, lane swerving (used for lane change and swerve-in-lane), merge-in-front, stop, and reverse. Note that these are elemental maneuvers, and composite maneuvers are implemented as BTs over the elemental maneuvers. For instance, lane maintenance composes velocity keeping, vehicle following, and stop. The maneuvers instantiate a general model (see Figure 5), which has three steps: (i) finding the target states for the maneuver, (ii) generating candidate trajectories, (iii) selecting an optimal trajectory. Each of these steps is controlled by a set of configurable parameters, allowing testers to realize a particular driving style or misbehavior. The generated trajectories are kept short (2 to 5 seconds), but some maneuvers, e.g., vehicle following, are performed over extended periods of time and, therefore, consist of a sequence of trajectories. The Behavior layer decides when to start, finish, or abort a maneuver.

1) *Target Finding*: Each maneuver has its own criterion to define a target state and a time to reach it. Target finding requires evaluating the road structure, traffic, and other objects. For example, the target for velocity keeping is to reach and keep a target velocity, while in the same lane; and the target for vehicle following is to reach and keep a certain target gap. The maneuver configuration is used to adjust the desired behavior according to the scenario goals by assigning target ranges to these parameters. Any lateral position relative to the current lane can be used, but if the maneuver is a lane swerve, the position is relative to the target lane. In the merge-in-front maneuver, the goal is to reach the same lane as the target vehicle, while achieving the target differences in position, velocity, and acceleration (δS). These target parameters allow simulating a dangerous cut-in maneuver by setting the gap to be small and closing. Our online documentation has a detailed description of maneuvers and target configuration options.

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. During execution, our model 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. The target state set is used to generate multiple trajectory candidates and select the best trajectory, filtering out configurations that may be infeasible or suboptimal.

2) *Trajectory Generation*: 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. We use an approach that plans each trajectory as a pair of quintic polynomials, in longitudinal and lateral direction, respectively [39], which minimizes jerk to reflect

smooth and comfortable driving. A quintic polynomial is a jerk-minimal connection between two points P_0 and P_T , in a one-dimensional problem with $p(t)$ as location and T as the motion duration [43]. The total accumulated jerk over the one-dimensional trajectory is given by 4:

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

Trajectory generation creates 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 $VehicleState_{Frénet}(t_0)$ and each of the target states $VehicleState_{Frénet}(t_0 + T)$ from the target-finding step. This results in a candidate set that respects the target constraints.

3) *Optimal Trajectory Selection*: This step selects a feasible and optimal trajectory from the candidate set, based on feasibility constraints and cost functions. *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 in Figure 6.

The remaining candidate set is ranked using a weighted sum of *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 BT 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 must be lowered. The resulting trajectory respects realistic vehicle motion, balances conflicting qualities such as progress and comfort, while implementing the scenario goals.

E. Execution Layer

The selected trajectory is executed as a function of time. At each new planning cycle, the BTs either continue the trajectory or switch between maneuvers if a new condition is triggered. Figure 7 shows an example of trajectory planning for a cut-in maneuver to the right lane. The grey lines are the candidate trajectories eliminated by feasibility constraints or higher cost. The blue line is the best cut-in trajectory based on scenario goals (target and weight values) and motion constraints. The green line is the target vehicle (Ego) trajectory.

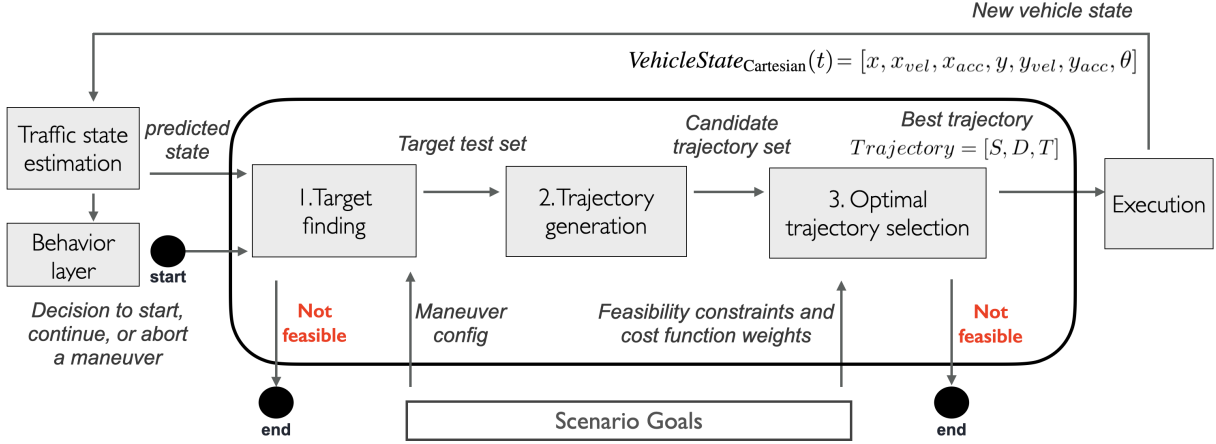


Fig. 5. SDV general maneuver model

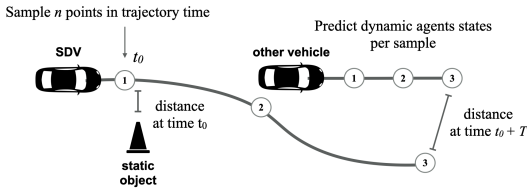


Fig. 6. Checking for collisions with static objects and dynamic obstacles

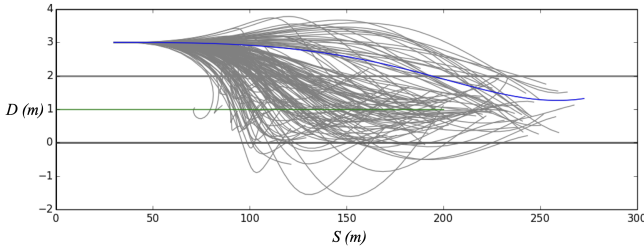


Fig. 7. Trajectory planning by the SDV during a cut-in maneuver

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 [38] and GeoScenario language [5] extended with the SDV BT definition format and creates a traffic simulation with the SDV model instances running concurrently. Vehicles that do not require complex reactive behaviors can use predefined trajectories rather than the SDV model, which improves performance. 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 (see Figure 8). The GSClient component provides a shared memory interface between the GeoScenario Server and WISE Sim, which runs within the Unreal game engine 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) [44] 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 (shown in the cut-in example in Figure 1). We also provide an experimental integration to run scenarios in co-simulation with Carla [29]. The implementation provides a collection of sample scenarios, BTs, and maps covering different traffic situations. All tools are available to the research community and can run scenarios out-of-the-box. More technical details are available in the online documentation.

V. EVALUATION

We evaluate the SDV model in terms of design effectiveness, realistic vehicle motion, practical applicability for scenario-based ADS testing, and finally 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 using 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*: the breadth of scenarios that can be represented with the model. Given a set of scenarios, we classify each scenario according to whether the required behaviors for all vehicles in the scenario could be modeled. We assign *success* (S) when all behaviors are successfully expressed with no limitations, *partial*

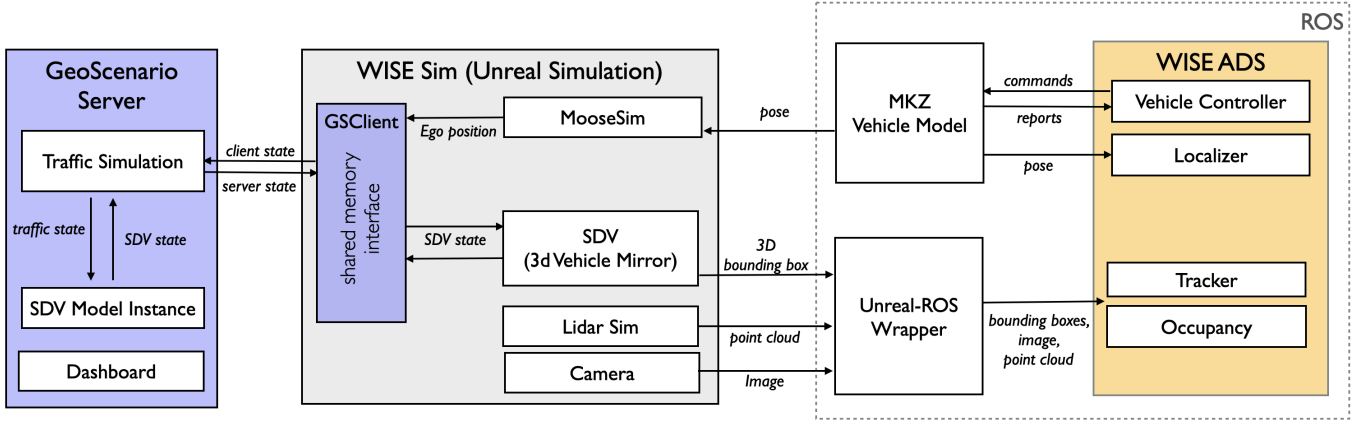


Fig. 8. SDV Model integration with the co-simulation environment (GeoScenario Server + WISE Sim) and the subject system under test (WISE ADS).

(P) when the behaviors for at least one variation of the scenario can be expressed, or *failure* (F) when the minimum behavior required for a scenario cannot be expressed.

- (ii) *execution accuracy*: accuracy of the scenario and vehicle behavior during simulation with respect to scenario objective. Even when a scenario can be represented with the language, the intent in design may not always translate to correct execution. After running a simulation, we classify the degree to which scenarios are correctly executed: *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) when vehicles deviate from the design intent and thus the scenario execution fails.
- (iii) *reuse*: the ability to reuse behavior across scenarios to reduce design effort. Behavior is defined using BTs, which can be reused by importing from a shared library, composing using sub-tree references, and configuring using parameter values. We quantify reuse in a scenario based on the *internal reuse level* from Frakes and Terry [45]. When applying this metrics, BTs are considered as higher-level items, which consist of nodes as lower-level items. Given a scenario containing a set of BTs (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 BTs of other scenarios) and L is the total number of nodes in the set of BTs. This metric assumes values between 0 and 1 and represents the percentage of internal reuse. The remaining percentage represents custom BT code, such as custom sub-trees, required to implement behavior that is specific to the particular scenario. Note that each BT in the library is used by at least two scenarios. Since a scenario may not use all the nodes of the BTs it imports from the library, we also compute the internal reuse level for a given scenario 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. In the original language, the vehicle behavior is composed of predefined

trajectories (PDT) with speed profiles and triggers to change them at run-time [5]. This common approach is also supported by other simulation tools, including PreScan [46] and VTD [7].

SDV models can be used in a wide variety of scenario designs and test cases requiring Ego-to-HV interactions, which naturally leads to a large design space to explore. In order to make the evaluation feasible, we focus on safety-critical scenarios that account for the majority of crashes in traffic. We use the Pre-Crash Scenario Typology from NHTSA [2] to compose this evaluation set. This scenario catalog provides interactive and realistic scenarios that can challenge the ADS capabilities in crash avoidance and are commonly used as a reference for ADS validation in other projects [47], [29]. 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. Each instance is based on a collection of BTs and maneuver configurations representing the behavior of one or more vehicles interacting with Ego. The original NHTSA set is based on reported events between HVs, but we assume that one of the HVs is Ego, operated the ADS (similar to how Waymo adapts NHTSA scenarios as tests [47]). A test scenario must also have goals and a clear success/fail criteria. 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 execute them in simulation using the reference implementation (Section 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 PDTs 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 PDTs with reasonable effort. 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 PDTs, 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 and provide the full list of scenarios and observations in the companion website.

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 are thus 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, 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 PDTs 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 required an adaptation to execute the scenario correctly.

Reuse: The composable nature of BTs allows us to reuse most of them, i.e., use each BT 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 or vehicle following 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 BT 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 BT 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. Considering only the nodes executed during a simulation, the average is 0.81.

The experience modelling 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 PDT baseline. 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. Based on the NHTSA statistics, the scenarios expressed and executed successfully with the SDV model represent about 49% of all light-vehicle crashes in traffic.

B. Vehicle Motion (RQ2)

As the primary goal is to simulate human controlled 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

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 is computed with all Behavior tree nodes (IRL), and only for nodes that are executed in the simulation (IRL exec). *Scenario #16 required a map adaptation to perform correctly.

mid-day traffic in Waterloo, Canada, which is part of the Waterloo Multi-Agent Traffic Dataset [48]. The “birds-eye” image was collected using a drone and processed to label and track pedestrians and vehicles (Figure 9).

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 where a simulated trajectory can trivially match the empirical vehicle. 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 repro-

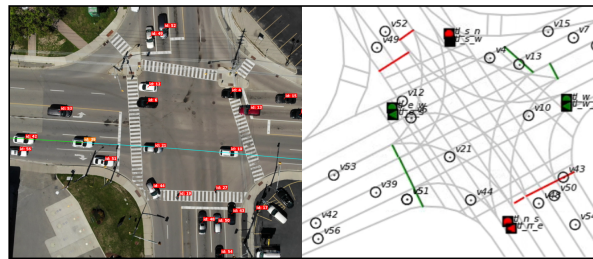


Fig. 9. A snapshot of the signalized intersection used for experiments and its corresponding simulation on the right.

duced 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 BT 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 BT 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 PDTs, 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 BT, it needs a BT 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 [49].

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) [50], 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:

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

Results: Figure 10 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. Further note that the improvement from calibration is most pronounced for (i) free driving by matching the average speed of the empirical vehicle, and (ii) signal light handling by matching the delay to resume driving on green.

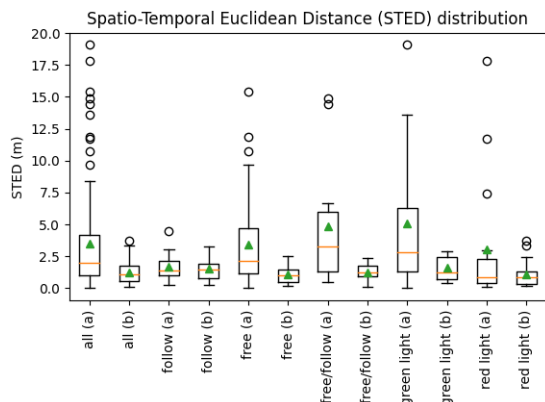


Fig. 10. Performance for all scenarios and per type, before (a) and after (b) calibration, measured using STED in meters. Orange lines represent medians, and green triangles represent averages.

Figure 11 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 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. After waiting for the green light, the reference vehicle can resume driving but needs to keep a following distance from the lead vehicle. The simulated vehicles resume with a smaller delay compared to the empirical one.

In summary, SDV models can closely reproduce the behavior of human-driven vehicles under the same traffic conditions. Overall, the model calibration can address varying driving styles and significantly increase the similarities in the trajectories. In some scenarios, such as in Figure 11 (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 BTs as additional maneuvers. All results, trajectory logs, speed and trajectory plots are available in the project website.

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. After the maneuver, the lane-changing vehicle becomes the lead of the Ego. Cut-in maneuvers from other drivers pose challenges to the ADS, and if not handled properly can lead to crashes. In fact, this scenario accounts for 338 000 crashes or 6.69% of all the light-vehicle crashes in the NHTSA report [2]. Avoiding or mitigating them is an important goal for any ADS development program.

The goal of this test is to evaluate the ADS capabilities to handle cut-ins from other vehicles. Testers want 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 [51]. 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 (Figure 12), in automated

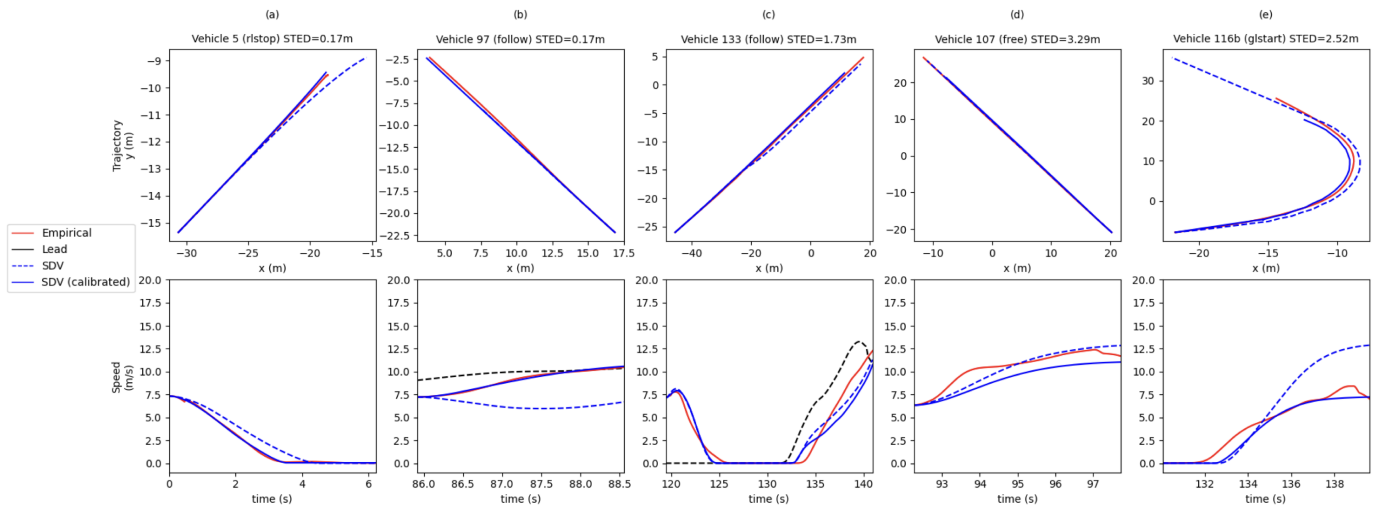


Fig. 11. Paths and speed profiles for five sample scenarios. Empirical vehicles in red; SDV models in dashed blue (before calibration) and solid blue (after calibration).



Fig. 12. The research platform

mode at SAE level 3. We focus on testing the ADS software in simulation, using WISE Sim with the GeoScenario Server implementing the SDV model (see Figure 8). GeoScenario server simulates the SDV model instances and injects them into WISE Sim, with both environments operating in co-simulation. Then WISE Sim uses the ROS publish-subscribe mechanism to publish simulated sensor data from (GPS, IMU, cameras, and lidar) for the ADS and receives the Ego pose from the high-fidelity dynamics model of the Ego.

2) *Test scenario*: The cut-in behavior is expressed as a BT, such as the one in Fig. 13, and assigned to an SDV model instance. According to this BT, 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 BT calls the standard drive BT (line 8) 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 to allow the vehicle to pick up pace (line 4), it starts checking for the acceptance distance gap (range) of 5 m (+10%) for a lane change to the right (`target_lane_id=-1`), on which Ego drives at a speed matching the road speed limit (line 6). Once the acceptance gap is satisfied, the lane change is triggered (line 7), with a target distance gap of 5 m and a relative velocity of -3 m/s (`delta_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 to lead to a crash.

```

1 behaviortree cutin:
2   ?
3
4   -> condition c_trigger( sim_time(t=4) )
5
6     -> condition c_gap(gap( target_lane=-1, range=MP(5.0,10) ))
7       maneuver m_swerve( MCutInConfig( target_lane=-1, delta_s=(5, -3) ) )
8       subtree drive_tree(m_vel_keep=MVelKeepConfig( vel=MP(14.0,10.6) ) )
9

```

Fig. 13. Cut-in Behavior Tree using our DSL

Results: As expected, more aggressive cut-ins 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 $\Delta d_a = 2m$ and the same target relative velocity $\Delta v_t = -5m/s$, but #8 has a smaller target distance gap, $\Delta d_t = -5m$, compared to $\Delta d_t = -2m$ 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 $\Delta d_a = 5m$. 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 aggressiveness in #11 results in a collision, however. Figure 14 shows scenario #8 with the SDV's trajectory generation (a-b), its ground-truth perspective (c), and the ADS's 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 the Ego initiates an emergency stop, the rear-end collision is not avoided.

This experiment demonstrates how the 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. We found that using another SDV instance as placeholder for Ego

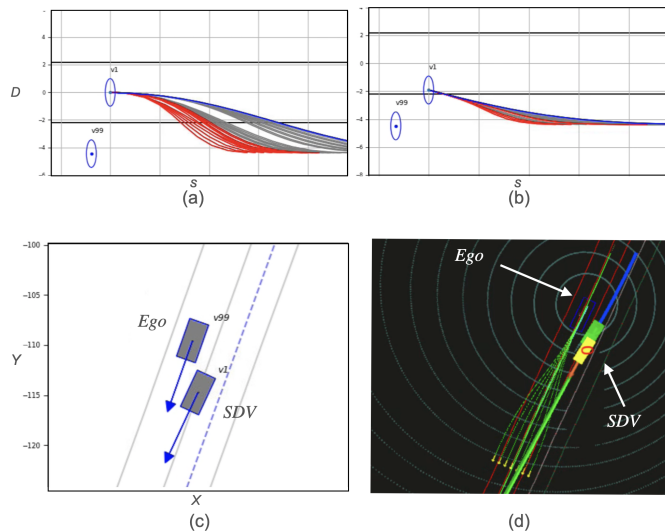


Fig. 14. One of the simulation scenarios that results in a crash; in (a) and (b), the SDV trajectory generation in Frénet Frame targeting Ego at two different moments (optimal trajectory in blue and infeasible ones in red); in (c) the SDV simulation view in Cartesian coordinates; and in (d) the ADS perception (circles represent the lidar simulation, with the Ego located at their center)

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. Although most scenarios rely on a small set of vehicles interacting with Ego, such as between one and three in the NHTSA pre-crash scenarios, complex scenarios may require simulating heavy traffic. For example, we observed that a single vehicle in our intersection dataset may interact with up to six other vehicles. In order 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 (Section IV). To provide a sufficient simulation update rate, the

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	-

Behavior and Maneuver Layers target a planning rate of 3 Hz, and the Execution Layer 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. 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 Execution Layer executes the trajectory of each vehicle from the previous planning cycle by (i) transforming the current target state in the planned trajectory from the Frénet frame to the Cartesian frame and (ii) updating the vehicle's position, velocity, acceleration, and yaw. The state transformation and update must be completed for all vehicles within 33 ms. A small exceedance, if consistent, may be acceptable, as it would slightly reduce the update frequency below 30 Hz without destroying the actual vehicle motion. The experiment is executed on an Intel Core i7-6800K at 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 vehicles, up to 20. In each scenario, the vehicles travel in one lane and form a virtual platoon, simulating heavy traffic. In scenario A, the vehicles travel without any disturbance, and in scenario B, they need to steer to avoid a static obstacle in their lane. When running scenario A, collision checking is inactive; and it is activated when running scenario B. The purpose of scenario B is to show the impact of 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). On the other hand, 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 (Figure 15). 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 even more vehicles, the traffic can consist of a mix of vehicles, with the more expensive SDV instances used for interactions with Ego,

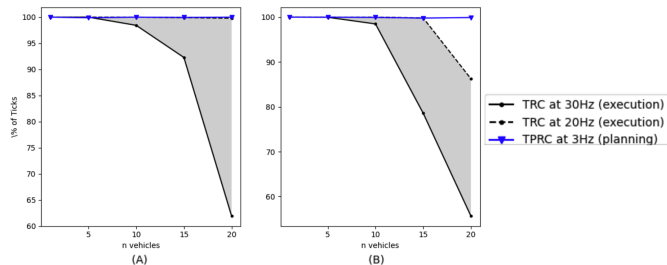


Fig. 15. Performance with increasing number of vehicles. (A) is without obstacle avoidance, and (B) is with obstacle avoidance. The grey area represents the performance range between 20Hz to 30Hz.

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

and the remaining vehicles following PDTs, which have a negligible computing cost.

VI. CONCLUSION

Scenario-based testing is one of the primary methods to verify and validate the behavioral safety of automated vehicles, as mandated by industry standards (e.g., ISO26262 and ISO21448) and safety frameworks (e.g., Waymo’s [47]). Supporting such testing requires tooling to express and execute focused tests, such as the NHTSA pre-crash scenarios, which is in contrast to broad, exploratory test runs, such as roaming in dense urban traffic. To be effective, such tooling must allow for sufficient behavior controllability, realistic movements, and interactive planning of the participating road users to achieve the test objectives. Further, the scenario representation should aid understandability and reuse, and the execution should scale to at least a dozen interacting vehicles.

The proposed SDV model addresses these needs through a combination of BTs and dynamic trajectory planning. The model encapsulates driver and vehicle as a single entity with a layered architecture that provides a user-oriented language to coordinate the vehicle behavior, and vehicle motion planning that optimizes for realism and achieving the scenario test objective. In particular, BTs provide a high-level description of discrete decisions, with a high-level of abstraction and parameterization to support controllability and reuse. Further, 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 the proposed approach supports effective test scenario development and execution. All eighteen NHTSA vehicle-to-vehicle pre-crash scenarios are successfully expressed and executed using the SDV model, except for one variant due to unsupported U-turns. The scenario

analysis also shows that their majority (78%) require dynamic trajectory planning, and thus cannot be effectively handled using the predefined-trajectory agent baseline. The dynamic trajectory planning also allows for easy adaptation of the tests to different road geometries. The ability to reuse sub-trees and override parameters support high levels of internal reuse, achieving over 80% on average for the NHTSA scenarios. In other words, on average, over 80% of a scenario’s content is also used in other scenarios.

The evaluation also shows the ability of the SDV model to reproduce real-world vehicle behavior and scale sufficiently. In one of the experiments, the average STED between the simulation and the real trajectories for 100 traversals through a busy urban intersection is 4.27 m before calibration, and it improves to 1.24 m after calibration. In particular, the simulation faithfully reproduces different driving styles by adjusting parameters and can accommodate custom behaviors, including misbehaviors, as additional 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.

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. Further, 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 [48] and the multi-country INTERACTION dataset [52]. We plan to improve the auto-calibration process and further automate creation of BTs and their parameterization to approximate the naturalistic traffic. We will also expand the BTs and maneuvers for interaction with pedestrians [53]. Finally, we plan to exploit the model in generating new scenarios by injecting road-user misbehaviors into BTs, such as simulating distraction [54] and ignoring occlusions [55]. 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] SAE, “Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems (SAE J3016),” SAE International, Tech. Rep., 2014.
- [2] W. G. Najm, J. D. Smith, and M. Yanagisawa, “Pre-Crash Scenario Topology for Crash Avoidance Research,” U.S. Department of Transportation, NHTSA, Tech. Rep., April 2007.
- [3] OpenScenario. <https://www.asam.net/standards/detail/openscenario>.
- [4] Measurable Scenario Description Language (M-SDL). <https://www.foretellix.com/open-language/>.

- [5] R. Queiroz, T. Berger, and K. Czarnecki, "GeoScenario: An open DSL for autonomous driving scenario representation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.
- [6] M. Althoff, M. Koschi, and S. Manzing, "Commonroad: Composable benchmarks for motion planning on roads," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 719–726.
- [7] Virtual Test Drive (VTD). <https://vires.com/vtd-vires-virtual-test-drive>.
- [8] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Trans. Safety*, vol. 4, pp. 15–24, 04 2016.
- [9] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Int. J. Robot. Res.*, pp. 583–602, 2003.
- [10] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weissgerber, K. Bengler, R. Bruder, F. Flemisch, and H. Winner, "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance," *IET Intelligent Transport Systems*, vol. 8, no. 3, pp. 183–189, 2014.
- [11] S. Ulbrich, T. Menzel, A. Reschka, F. Schultdt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *IEEE 18th International Conference on Intelligent Transportation Systems*, Sept 2015, pp. 982–988.
- [12] K. Go and J. M. Carroll, "The blind men and the elephant: Views of scenario-based system design," *Interactions*, vol. 11, no. 6, pp. 44–53, Nov. 2004.
- [13] C. Kaner, J. Bach, and B. Pettichord, *Lessons Learned in Software Testing*. New York, NY, USA: John Wiley & Sons, Inc., 2001.
- [14] ISO/FDIS 26262:1994, *Road vehicles – Functional safety*. ISO, Geneva, Switzerland, 2011.
- [15] ISO/PAS-21448:2019, *Road vehicles – Safety of the intended functionality*. ISO, Geneva, Switzerland, 2019.
- [16] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium, IV 2018, Changshu, Suzhou, China, June 26-30, 2018*, 2018, pp. 1821–1827.
- [17] 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*, vol. 19, no. 6, pp. 1243 – 1262, 2011.
- [18] R. B. Abdessaleem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *31st IEEE/ACM Int. Conference on Automated Software Engineering (ASE)*, Sep 2016, pp. 63–74.
- [19] —, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. 40th Int. Conf. on Software Engineering*. ACM, 2018, pp. 1016–1026.
- [20] O. Bühler and J. Wegener, "Automatic testing of an autonomous parking system using evolutionary computation," *SAE Technical Papers*, 2004.
- [21] OpenDRIVE. <https://www.opendrive.com>.
- [22] 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. on Programming Language Design and Implementation*. New York, USA: ACM, 2019, p. 63–78.
- [23] X. Zhang, S. Khastgir, and P. Jennings, "Scenario description language for automated driving systems: A two level abstraction approach," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 973–980.
- [24] 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 International Conference on Model Driven Engineering Languages and Systems*. New York, NY, USA: ACM, 2020, p. 114–120.
- [25] J. Sewall, D. Wilkie, P. C. Merrell, and M. C. Lin, "Continuum traffic simulation," *Computer Graphics Forum*, vol. 29, 2010.
- [26] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," *Computer Graphics Forum*, vol. 39, 2020.
- [27] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.
- [28] A. Kesting, M. Treiber, and D. Helbing, "General Lane-Changing Model MOBIL for Car-Following Models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.
- [29] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [30] S. Moridpour, M. sarvi, and G. Rose, "Modeling the lane changing execution of multi class vehicles under heavy traffic conditions," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2161, 12 2010.
- [31] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, March 2017.
- [32] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein, "Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2383–2390.
- [33] 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 *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2118–2125.
- [34] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to simulate realistic multi-agent behaviors," 2021.
- [35] M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.
- [36] J. A. Michon, "A critical view of driver behavior models: what do we know, what should we do?" in *Human behavior and traffic safety*. Springer, 1985, pp. 485–524.
- [37] E. R. Boer, M. Hoedemaeker *et al.*, "Modeling driver behavior with different degrees of automation: A hierarchical decision framework of interacting mental models," in *Proc. 17th European annual conference on human decision making and manual control*, 1998, pp. 63–72.
- [38] 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 *Proc. IEEE Intell. Trans. Syst. Conf.*, Hawaii, USA, November 2018.
- [39] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," *IEEE Int. Conference on Robotics and Automation*, pp. 987–993, 2010.
- [40] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2018.
- [41] SAE, "Taxonomy and Definitions for Terms Related to Automated Driving System Behaviors and Maneuvers for On-Road Motor Vehicles (SAE J3164)," SAE International, Tech. Rep., 2018.
- [42] K. Czarnecki, "Automated Driving System (ADS) Task Analysis - Part 2: Structured Road Maneuvers," Tech. Rep., 07 2018.
- [43] 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 on Intelligent Robots and Systems*, 1989, pp. 392–397.
- [44] Robot Operating System (ROS). <https://www.ros.org/>.
- [45] W. Frakes and C. Terry, "Software reuse: Metrics and models." *ACM Comput. Surv.*, vol. 28, pp. 415–435, 06 1996.
- [46] MathWorks PreScan. .
- [47] "Waymo safety report," Tech. Rep., 09 2020. [Online]. Available: <https://waymo.com/safety/>
- [48] Waterloo Multi-Agent Traffic Dataset. <http://wiselab.uwaterloo.ca/waterloo-multi-agent-traffic-dataset>.
- [49] K. Czarnecki, "Automated Driving System (ADS) Task Analysis - Part 1: Basic Motion Control Tasks," Tech. Rep., 07 2018.
- [50] M. Nanni and D. Pedreschi, "Time-focused clustering of trajectories of moving objects," *J. Intell. Inf. Syst.*, vol. 27, pp. 267–289, 11 2006.
- [51] WISE ADS. <https://uwaterloo.ca/waterloo-intelligent-systems-engineering-lab/projects/wise-automated-driving-system>.
- [52] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clause, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, and M. Tomizuka, "INTERACTION Dataset: An INTERNATIONAL, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," *arXiv:1910.03088 [cs, eess]*, 2019.
- [53] S. Larter, R. Queiroz, S. Sedwards, A. Sarkar, and K. Czarnecki, "A hierarchical pedestrian behaviour model to generate realistic human behaviour in traffic simulation," in *IEEE Intelligent Vehicles Symposium (IV22)*. IEEE, 2022.
- [54] J. van Lint and S. Calvert, "A generic multi-level framework for microscopic traffic simulation—theory and an example case in modelling driver distraction," *Transportation Research Part B: Methodological*, vol. 117, pp. 63–86, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0191261518302704>
- [55] 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 *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.



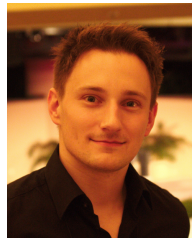
Rodrigo Queiroz is a Ph.D. candidate at the University of Waterloo Faculty of Engineering (Canada) and is part of the Waterloo Intelligent Systems Engineering (WISE) Lab. His research interest focuses on validation & verification of autonomous driving systems, safety-critical systems, motion planning, vehicle simulation, and traffic simulation. He received his Master's degree in Computer Science from the Federal University of Minas Gerais (UFMG) in Brazil.



Sergio García is a Ph.D. student at the University of Gothenburg in Sweden. His research interest focuses on robotics software engineering, striving to understand the complexity of the domain and analyzing its characteristics and challenges to develop solutions based on them. He received his master's degree in electronics from the University of Alcalá in Spain in 2016 and is expected to receive his Ph.D. degree from the University of Gothenburg in September 2021.



Divit Sharma is a graduate from the University of Waterloo in Canada with a Bachelor's degree in Computer Science. His interest and experience lie in robotics simulation and game development. His past work includes researching high-definition context maps for autonomous vehicles and developing interactive multi-agent simulation systems at the Waterloo Intelligent Systems Engineering Lab. He also has internship experience at Ike Robotics, a California-based self-driving startup, and Behaviour Interactive, a Montreal-based game development studio. Divit graduated in 2021 and is looking to continue his involvement in the autonomous vehicles industry.



Thorsten Berger is a Professor in Computer Science at Ruhr University Bochum in Germany. His research focuses on 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 PhD degree in computer science from the University of Leipzig in Germany in 2013. Thereafter, he worked as a Postdoctoral Fellow at the University of Waterloo in Canada and the IT University of Copenhagen in Denmark, and as an Associate Professor at Chalmers | University of Gothenburg in Sweden.



Ricardo Caldas is a Ph.D. student at the Chalmers University of Technology in Sweden. His research interest focuses on verification of autonomous systems. 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. He received his master's degree in Computer Science from the University of Brasília in Brazil in 2019.



Krzysztof Czarnecki is a Professor of Electrical and Computer Engineering at 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 focus on assuring the safety of driving behavior and machine-learned functions. He co-lead the development of the first ADS tested on public roads in Canada in 2018. As a member of standardization committees, he has contributed to ISO 21448 (2nd 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.



Patrizio Pelliccione is a Professor in Computer Science at Gran Sasso Science Institute (GSSI). His research topics are mainly in software engineering, software architecture modeling and verification, autonomous systems, and formal methods. He received his PhD in computer science from the University of L'Aquila in Italy. Thereafter, he worked as a senior researcher at the University of Luxembourg in Luxembourg, then assistant professor in the University of L'Aquila in Italy, then Associate Professor at Chalmers | University of Gothenburg in Sweden and University of L'Aquila. He has been on the organization and program committees for several top conferences and he is a reviewer for top journals in the software engineering domain. He is very active in European and National projects. In his research activity, he has collaborated with several companies. More information is available at <http://www.patriziopelliccione.com>.