

Validation of decision-making in artificial intelligence-based autonomous vehicles

Christopher Medrano-Berumen & Mustafa İlhan Akbaş

To cite this article: Christopher Medrano-Berumen & Mustafa İlhan Akbaş (2020): Validation of decision-making in artificial intelligence-based autonomous vehicles, Journal of Information and Telecommunication, DOI: [10.1080/24751839.2020.1824154](https://doi.org/10.1080/24751839.2020.1824154)

To link to this article: <https://doi.org/10.1080/24751839.2020.1824154>



© 2020 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 01 Oct 2020.



Submit your article to this journal [↗](#)



Article views: 192



View related articles [↗](#)



View Crossmark data [↗](#)

Validation of decision-making in artificial intelligence-based autonomous vehicles

Christopher Medrano-Berumen^a and Mustafa İlhan Akbaş ^b

^aComputer Science, Florida Polytechnic University, Lakeland, FL, USA; ^bElectrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA

ABSTRACT

The autonomous vehicle technology is considered as a significant market disruptor for multiple industries. However, to reach this potential and to be accepted by the public, autonomous vehicles must be proven to be reliable and safe. Therefore, validation is essential for improving the public trust for autonomous vehicles and deploying them for everyday transportation activities. There have been a number of significant efforts on validation of autonomous vehicles; and real-life testing and test tracks have been the major platforms for these activities. However, simulation has also been gaining popularity due to its advantages in cost, time and safety. In this paper, we present a simulation scenario generation methodology with pseudo-random test generation to validate the decision-making system of autonomous vehicles. The methodology separates the validation concerns and focuses on generating scenarios that test the decisions taken by the vehicle. The implementation demonstrates the capabilities and the efficiency of the approach.

ARTICLE HISTORY

Received 16 June 2020

Accepted 13 September 2020


Keywords

Autonomous vehicles;
validation; simulation;
testing; verifiable AI

1. Introduction

Successful autonomous vehicle (AV) technologies could fundamentally transform various industries such as automotive, transportation, energy, farming and so on. To achieve this potential, there is a significant investment in artificial intelligence (AI) technology, which is at the heart of these platforms. Consequently, autonomous capabilities such as those afforded by advanced driver assistance systems (ADAS) and other automation solutions are increasingly becoming available in the marketplace.

As the introduction of AVs becomes more inevitable, questions of their safety and viability also become more apparent. In order to gain the public trust for adopting the technology and to convince federal entities for allowing AVs on the streets, a reliable way to validate and verify AV competence needs to be established. An AV will not be accepted unless it is demonstrably as competent, at the very least, as its human counterpart. This can be achieved through validation and verification, the process by which a product is shown to satisfy the needs of the

CONTACT Mustafa İlhan Akbaş  akbas@erau.edu  Electrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114, USA

© 2020 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

stakeholders and regulatory bodies. This is actually a serious issue for not only AVs but all autonomous cyber-physical systems such as unmanned aerial vehicles or autonomous robots.

Due to the novelty of AV validation and verification, the requirements of this field are still unclear and there is yet to exist a regulatory body to define what specifications an AV must meet. There is a safety standard for electronics in regular vehicles, the International Organization of Standardization's ISO 26262 (ISO, 2018). ISO 26262 is a risk-based safety standard that provides a process to prove functional safety through a vehicle's life cycle for its electronic components and systems. Current methods of AV validation and verification such as shadow driving or annotated images-based testing are costly, slow, dangerous and resource intensive (Razdan et al., 2019). Hence, modelling and simulation is an indispensable asset to achieve validation goals for AVs.

This paper is an extended version of our ACIIDS 2020 paper (Medrano-Berumen & Akbaş, 2020), where we described our initial studies on the simulation development methodology for the validation of AV decision-making. The proposed approach follows a framework that is built according to the 'separation of concerns' principle (Alnaser et al., 2019; Medrano-Berumen & Akbaş, 2019) and separates the role of low-fidelity simulation testing for AV decision-making from other testing methods such as high-fidelity simulation testing for perception systems. In this paper, we extend our approach with a comprehensive literature review, a complete description of the approach with model generation details and simulations to describe additional example scenarios.

In our solution, a semantic language is designed to describe the scenarios for testing the decision-making component of an AV. To enable automatic test generation and control, we need to be able to define scenarios by using parameters. As all parameters are not independent of one another, they should be able to constrain each other to limit the input space to legal scenarios, which can be done automatically through the use of a constraint solver (George & Mohamed, 2011). Hence, details used to describe roads are parameterized, allowing us to both randomly and deterministically generate roads and street networks.

We focus on the topology of the region in consideration to verify an AV in the most efficient way possible. The roads within that region are broken down into their most elementary segments which are then turned into road pieces, single road blocks that are connected to make up more complex roads in the simulation. Thus, the input space is limited further while defining the parameters through which coverage-driven validation and verification methods will be searching.

The parameterization is done for the behaviour of actors in a driving scenario as well, giving us a full description of the scenario that the vehicle under test perceives. This gives us the means by which test scenarios can be generated with entirely random features. By designing the language to describe all possible scenarios, complete coverage of the AV's complex domain can be virtually achieved.

Focusing on the decision-making at AVs enables the utilization of low-fidelity simulation. The efficiency of this approach can be used with validation features such as assertions to classify scenarios and detect edge cases. The scenario generation capability allows positioning the AV under test within this simulation by hardware or software in the loop techniques using any existing system modelling environment, where the vehicle perceives its virtual environment and acts at each step. Data for further analysis is collected during and after the run as part of this framework.

2. Related work

The safe deployment of AVs in traffic is still an open challenge for both academia and industry. Hence, there are various initiatives in industry and research on testing AVs. However, most of these methods fall short by some measure. In this section, we describe the current state of the testing and validation for AVs with a focus on modelling and simulation.

The most common form of AV testing is known as ‘Shadow Driving,’ where a driver is ready to preemptively prevent an accident or to take over in case the AV decides to disengage (Favarò et al., 2017). To verify that AVs will be at least as safe as humans, this method is shown to take at least 275 million miles (Kalra & Paddock, 2016). Some if not all of those miles would also need to be repeated if there were to be any update to the AV under test. Using test tracks is also common for real-world testing, as this allows companies to test specific and sometimes extreme scenarios. Hardware-in-the-loop (HiL) testing is another option that allows connecting the AV’s brain into a simulation system and testing the interaction with specific hardware components simultaneously (Sarhadi & Yousefpour, 2015).

Several entities have been combining the common testing methods to develop their own framework for testing AVs. However, there is yet to be a single framework adopted as a standard. For instance, Waymo tests each of their primary subsystems, the vehicle, their hardware, and their software, individually as well as together using simulation, closed-course testing, and real-world driving (Waymo safety report: On the road to fully self-driving, 2018). The scenarios they focus on are based on training the vehicle behavioural competencies as designed by Berkeley’s PATH, such as detecting bikes and pedestrians or responding to emergency vehicles (Nowakowski et al., 2015), as well as behavioural competencies designed by Waymo themselves. PEGASUS (Hallerbach et al., 2018), a joint effort between multiple groups from science and industry, aims to develop a full toolchain for AV validation, looking at traditional vehicle validation and new innovations in the field for inspiration. A common trait with these is that they are still in progress of being designed and provide no hard definition for when an AV can be demonstrated to be sufficiently competent.

Test scenario generation is critical for the validation efforts. (Fremont et al., 2020) proposed a methodology for generating tests using a scenario description language (Fremont et al., 2019), identifying unsafe tests using runtime verification (Dreossi et al., 2019), and testing those scenarios at a higher fidelity in the form of a test track. However, rather than focusing on separation of concerns or domains, they focus on using a high-fidelity simulator that is tied closely to the test track in that it uses LiDAR data collected from the test track to generate the simulation environment. Rather than developing a testing regime using validation methodology, they also emphasize more on edge case testing. We follow a different approach and instead of verifying the AV through a single form of testing, we break down the components and testing methods in order to reduce complexity and gain coverage. The different components of the AVs to be broken down are the perception, object recognition, decision-making, vehicle dynamics. These components are verified at different levels in the system by using platforms such as simulation, hardware-in-the-loop, and real-world testing.

The simulation has been an important tool for studying cyber-physical systems (Akbaş et al., 2015, 2016; Akbaş & Turgut, 2011; Rentrop & Akbaş, 2017). Simulators used for AV

testing vary by great degree in what their approach and focus are. Vehicle-in-the-Loop simulators place a physical representation of the vehicle under test (often an actual vehicle) on a platform surrounded by projected screens to create the full driving experience from the driver's perspective. Some of the testing approaches focus on using simulation to verify that newly learned manoeuvres (e.g. u-turns, merging) are tested until they can be performed at a satisfactory rate. Others use simulations based on scenarios that their vehicles in the real world encountered (Dolgov, 2016). In these systems, each important scenario from real life can be fuzzed into generating more scenarios based on the original to strengthen the coverage of that test. There are also several recent initiatives aiming for the standardization of AV validation by integrating different techniques. Intelligent Testing Framework (Li et al., 2016) and PEGASUS are two important examples of such approaches.

There are also simulators focusing on vehicle dynamics, to test the internal machinations and physics of the vehicle driving on the road. V-REP is a robotics testing environment for vehicle dynamics that allows the user to experiment with sensors, mechanics, and control algorithms (Freese et al., 2010). Another example, VTI, uses open-source formats, OpenSceneGraph and OpenDRIVE, to describe the road network and includes highly detailed built-in vehicle models (Jansson et al., 2014). PreScan is a well-known commercial product which allows the user to define scenarios and execute them in its runtime environment with different 3rd party integrations (Tideman & Van Noort, 2013). SynCity is a simulator with an emphasis on realistic scenarios for sensor testing and machine learning training, working at the communication protocol level for realistic sensor feedback (Keirstead et al., 2010).

Traffic simulators are used to implement the logic of transportation control systems and simulate traffic at the micro level, macro level, or both. SUMO is a popular open-source traffic simulator with a low-overhead for running city-size road networks at a microscopic level (Behrisch et al., 2011). It has been used by both academia and industry in various projects (Goss et al., 2019). CORSIM is also a microscopic traffic simulation system that includes implementations of more complex driving behaviours such as spillback changes and toll plazas (Halati et al., 1997). MATSIM is another open-source agent-based microscopic traffic simulator, which is designed to bring together the fields of traffic simulation, large-scale computation, choice modelling, and complex adaptive systems (Horni et al., 2016). Vires is involved in the development of several open-source standards to define scenarios, roads, and vehicle dynamics, as well as Virtual Test Drive, a complete tool-chain for driving simulation that uses those standards (Dupuis & Karl, 2017). Waymo's Carcraft combines virtual maps with sensor data from the real world to recreate scenarios that their shadow driving fleet have encountered, fuzzing them for variation (Kehrer et al., 2018). Uber's Advanced Technology Group created AVS, a web-based visualization toolkit that allows users to visualize an AV's environment when making decisions. In this system look aheads, predictions, and so on are visually represented through different colours on the map (ATG, n.d.).

There is a small group of simulators optimized as deep learning platforms, which focus on the training process. NVIDIA's DRIVE Constellation (Goodwin, 2019) focuses on using the GPU to efficiently train the AV using two servers, one to simulate realistic sensor data while the other simulates the entire AV software stack. Cognata (Nair & Wishart, 2018) creates a comprehensive AV feedback loop with a realistic environment to train AVs by combining PhysicsStudio with TrueLife 3D Mesh. PhysicsStudio adds a layer of dynamic traffic

models while TrueLife 3D Mesh uses computer vision and deep learning algorithms to create meshes of cities. rFpro creates realistic road in simulation for the same purpose by using data collected from the real world and matched with HD maps (Nair & Wishart, 2018).

The majority of the simulators target testing of the full vehicle stack, from scene perception and understanding to making a decision for action in that scene. In our approach, we focus mainly on the decision-making step. In other words, our approach is designed to test the decision-making of the AV under test with the assumption that perception (sensors and sensor fusion) and action (vehicle dynamics) are functioning perfectly. We chose MATLAB as our simulation platform, as it provides a full tool-chain testing platform, the Automated Driving (AD) toolbox (The MathWorks, Inc, Release R2018b), with the capability to eliminate the vehicle dynamics or realistic visuals for sensor testing, two things that would create unnecessary complexity at the abstracted level. By rendering all objects in the scenario, from the vehicles to the pedestrians as three-dimensional boxes, we obtain the necessary level of abstraction to make the testing vehicle agnostic.

To provide interoperability among simulators, a common language is necessary when describing scenarios that will often be read into or exported from the simulators. For driving scenarios, these mainly fall into two categories: road description language or scenario description language. A road description language describes the street network which includes the main details such as geometry, lanes, signage, lights, speed limit, and sometimes texture. Some of the road description languages include VIRES's OpenDRIVE (Dupuis et al., 2010) and CommonRoad (Koschi et al., n.d.), both of which use an XML format to define the street networks. Scenario description languages describe different actors and how they operate on the road. These include Foretellix's M-SDL (M-SDL, 2019), VIRES's OpenSCENARIO (Menzel et al., 2018), and Berkeley's Scenic (Fremont et al., 2019). OpenSCENARIO takes an XML approach similar to OpenDRIVE, while M-SDL and Scenic use a tabbed approach in the language similar to Python.

3. Scenario generation approach

The scenarios for AV verification is generally generated by using the scenarios from real-life situations. Most of the time, the data are collected from naturalistic driving experiments and replicated in simulation with some fuzzing to increase the coverage. Due to how long it would take to experience every possible scenario, this is an impossible task to tackle. To overcome this obstacle, our approach defines the possible properties and types of roads and actors in a specific operational domain and creates a system that generates a scenario from those properties' randomization. If this system manages to cover all possibilities in a well-defined input space, then not only have we defined a way to describe the scenarios but also a way to construct the scenarios without the need for any extra real-world data collection.

Our goal is to verify the decision-making part of the AV at a highly abstracted level which requires only the logical components of a scenario: the scenario elements and their positions relative to the AV under test in the scenario. Once we are done with the scenario testing of the decision-making component, these scenarios can then be translated to the existing languages meant to define all scenarios in a standard format such as OpenDRIVE (Dupuis et al., 2010), the scenario can then be moved to a higher fidelity simulator to be tested for varying platforms such as HiL, ViL and varying environmental or terrain conditions.

It is important to note that there is a need for a clear definition of separation of concerns at this point. The validation mechanism must determine where to draw lines to avoid unnecessary complexity. For instance, let us say we must decide on the significance of the width of a median for an AV's decision. If it plays a part in the decision of the AV, then its physical properties should be varied at this lower level of simulation. If only the presence of the median matter, then this can be simplified to a boolean value that inserts a median of constant width.

An AV's decisions are mainly based on the road, actors in the scenario, obstacles, traffic rules, and its goal. By constructing the scenarios based on individual road segments, the goal is automatically to get to the end of the generated road segment. The only other potential goal an AV could have would be parking, though this can be attached to road segments that specifically represent parking lots or other parking areas, with the intent to park in any spot or a specific one. Traffic rules and signals are determined by location and are specific to the road they are on, meaning they can be attached to the road as parameters or properties as well.

Actors in a driving scenario are most often used to describe other vehicles. However, actors in test scenarios can also be used to describe pedestrians, bikes, emergency vehicles, and other dynamic objects on or adjacent to the road. We consider obstacles as static actors, following all of the properties of the dynamic actors except the travelling path. This means that a scenario can be defined purely in terms of its road and its actors.

A matrix-based language was designed in our earlier work to describe scenario properties (Medrano-Berumen & Akbaş, 2020). We extend this language in this paper with a matrix definition for the road network and another for the actors in the scenario. Each row of the road matrix represents a single road piece that makes up the road network or path in the scenario, and each column represents a property of the road. Each row of the actor matrix represents an actor in the scenario, and each column represents a property of the actor. Each value in the matrix is either a numeric value or a string, allowing for more complex properties with potentially variable lengths to be described. Changing any cell in either matrix then changes the scenario, creating an efficient way to randomly generate individual scenarios.

3.1. Road piece generation

The road on which the ego vehicle and other actors drive is generated by defining road pieces, the basic road building blocks. Each road piece is the smallest identifiable layout of a road segment that can be encountered while driving, each with their own individual properties, such as roundabouts, intersections, turns and so on. By defining these road pieces and all possible variations, a significant coverage is attained both by the pieces alone and combined.

Since the ego vehicle only depends on information within a certain local vicinity to properly manoeuvre, more complex road networks are not within our scope. In other words, the ego vehicle only cares about its immediate surroundings and not what is across the block if it is not heading there. Therefore, generating a road far away from the ego vehicle is not of our concern. The road is therefore generated from where the ego vehicle will begin and continues piece by piece, creating with it the path that the ego vehicle will follow.

While the pieces constrain themselves to roads technically possible in the real world, the validation for the road network is performed as it is being generated. With each new piece about to be placed, the 3D space around this potential piece is checked to ensure it will not conflict with a previous piece, as there would then be no reason to test the scenario.

Performing the validity check for new road pieces during the road network generation instead of when defining the road matrix saves computation time. This also saves time that would have been wasted running illegal scenarios that are not logically possible. The advantage of random scenario generation is the ability to batch run several of these scenarios at a time to more quickly identify interesting scenarios.

To generate the road piece in simulation, the parameters of the scenario that the road will be placed in, the points that make up the centre line of the road, the width, and optional lane markers are used. Clothoid curves are used to smoothly connect the points, allowing one to describe the road. Each road piece has its own script to generate the necessary information for inserting it into the scenario since each has its own unique layout.

[Figure 1](#) shows the road generation loop that is used in our design. The roads are created in the simulation by feeding the road matrix into a loop that iterates over the rows and using a switch statement to pass along the parameters to the appropriate script according to the road type. The generated pieces are then validated before the road function is called. When a road piece fails the validation, it is not placed in the scenario and the system moves on to the next row.

Each added road piece is a part of the ego vehicle's path meaning each piece is placed in a sequence to make a single route. To do this, each new road piece's coordinates, the points that make up the centre line for the road function, are rotated to align the road's tangent line at its first point with the tangent line of previous piece's last point and then translated to that point. If two pieces are different in widths for a reason such as differing number of lanes, a transition piece is placed in between, which gradually changes widths from one piece to the other. This transition piece also takes care of phasing in or out of lanes when the number of lanes changes.

Parameters for roads are defined as they apply to existing road pieces. Whenever a new road piece is designed, it is examined for any details that cannot be described by existing parameters. If there are details that cannot be described by the existing parameters, new parameters are added and the matrix is expanded. [Table 1](#) shows our current parameters. This parameter list grows as we analyse more operational domains and their critical parameters.

3.2. Actor generation

Actors are defined in our scenario generation method as any component of the scenario that is not a road piece or the ego vehicle. Hence, we include stationary obstacles as well in our actor definition. The types of actors an AV can expect to encounter are limited by its operational domain.

The common actors are different types of cars, trucks, motorcycles, and people, making their dimensions easy to randomize, as they are all usually within a certain range of sizes. The important factor is their behaviour, as it directly impacts the decisions of the ego vehicle. While it is important to be able to simulate normal behaviour, in order to maximize bug finding, behaviours that deviate from the norm demand more effort.

In simulation, actors are created using the dimensions of the actor, and the trajectory along with the parameters in [Table 2](#). Translating the continuous range of possible behaviours of different actor types into discrete values to be able to define them in terms of a

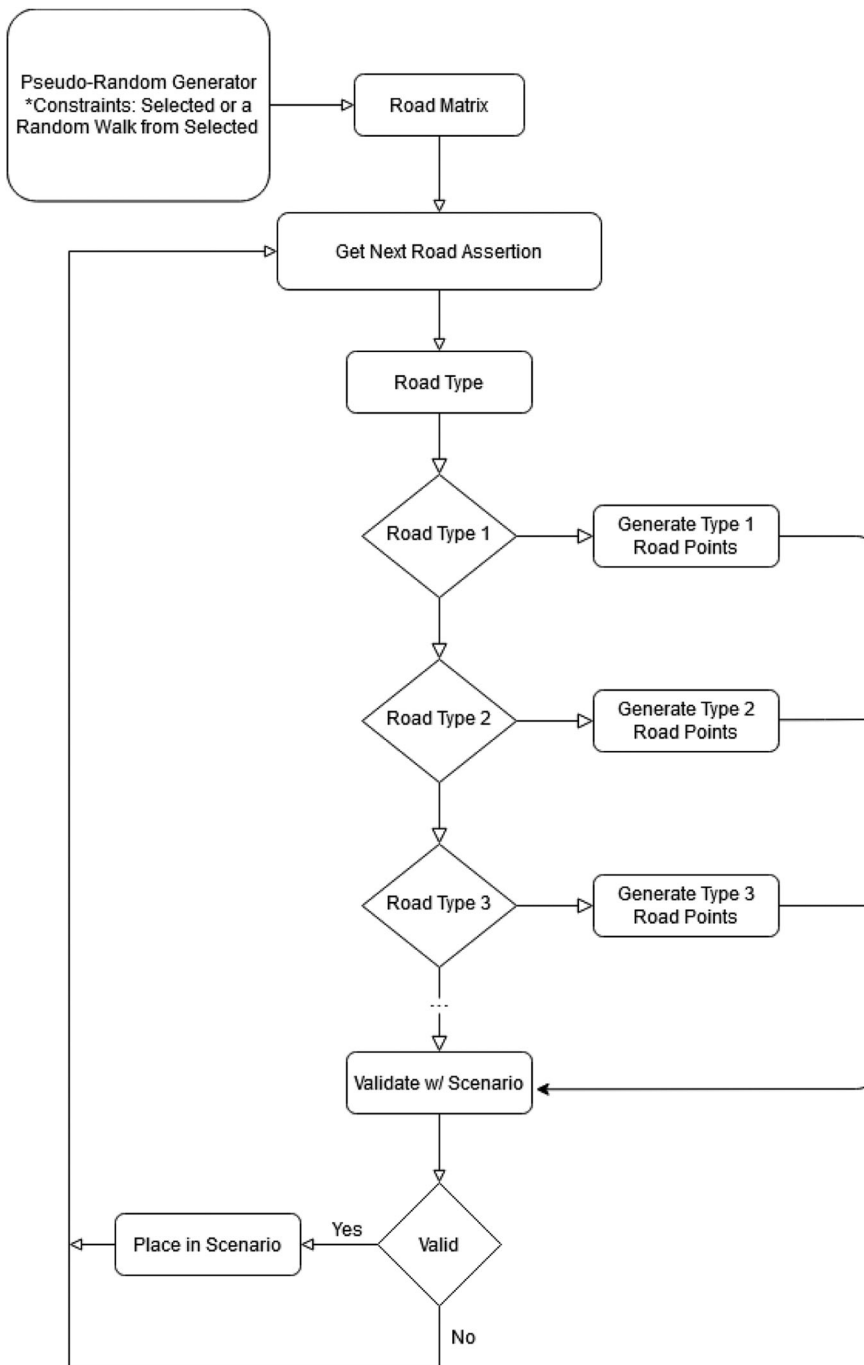


Figure 1. Road generation loop.

limited number of parameters is a challenging task. We decided to start with some easily identifiable behaviours that could be generated as some path with extra parameters for variation in terms of speed or basic constant values such as horizontal offset as seen in the offset parameter in [Table 2](#). The behaviours we started off with were cutting the

Table 1. Road parameters in road matrix.

Parameter	Values	Description
roadType	1–5	Determines the piece to be generated
roadLength	80–200	In m, how long the road is; 1–10 m for a pedestrian crosswalk
lanes	1–5	How many lanes, repeated for number of roads in road piece ('###...')
bidirectional	0–2	0 = one-way, 1 = bidirectional w/ double solid yellow lines, 2 = bidirectional w/ dashed yellow line
midLane	0–3	0 = nothing, 1 = mid-turn-lane, 2 = median, 3 = large median
speedLimit	30–80	In mph, goes by 5's, converted to mps when passed into function
intersectionPattern	0–2 0–2 0–2	For fourway intersection, calculates a valid pattern for the bidirectional parameter for other roads
curvature1	–0.25 – +0.25	Starting curvature for multilane road
curvature2	–0.25 – +0.25	Finishing curvature for multilane road
pedPathWays	pos1_side1_freq1	Composite parameter of variable length placing pedestrian spawning points somewhere along the road
outlets	pos1_side1_size1	Composite parameter of variable length placing an empty piece of road besides the road being implemented
showMarkers	0, 1	Boolean value determining whether to show lane markers

Table 2. Actor parameters or columns in actor matrix.

Parameter	Values	Type	Description
actorType	1–2	int	Determines whether actor is vehicle or pedestrian
carType	1–3	int	For vehicles, determines whether car, motorcycle, or truck; affects dimensions
pathType	1–6	int	Determines what path the vehicle or pedestrian will take (i.e. cut-off)
movSpeed	–3–3	int	Determines mps offset from speed limit or standard walking speed
x	0.67–1	float	Offset factor for max width
y	0.67–1	float	Offset factor for max length
z	0.67–1	float	Offset factor for max height
startLoc	0–1	float	Where along the road the actor will start from 0 at the start to 1 at the end
forward	0–1	int	Determines whether vehicle will be going in same or opposite direction for ego vehicle; left or right across the road for pedestrians
offset	0–1	float	Offset for certain behaviour types (i.e. how far off a vehicle will veer from the centre for that behaviour)
cutOffPoint	0–1	float	For vehicle cut off behaviour, how far until it cuts off the ego vehicle
stopPoint	0–1	float	For vehicle stopping behaviour, how far until it stops suddenly

ego vehicle off, veering outside of the lanes, and platooning for vehicle actors, and crossing the road straight and with a stop for pedestrian actors.

The trajectory of an actor consists of the points that make up its path and the speeds at each of those points. Other optional parameters are also available such as pitch and yaw. Points along the trajectory are connected using clothoid curves as the roads are to maintain continuity. The speeds are interpolated linearly so that the car accelerates or decelerates at a constant acceleration between them to reach the given speeds at their respective points. Together, the parameters given by the actor matrix help define the trajectory once passed into the appropriate path function. This is then validated as in [Figure 2](#) before the actor is placed in the scenario to avoid illegal scenarios where actors cross into each other, as most simulation platforms lack this check.

3.3. Scenario matrix

The model must be capable of creating scenarios to reflect the set of all possible situations. Our matrix-based semantic language is designed for breaking down the factors and it defines a scenario including roads, actors, and traffic logic. Therefore, the creation of

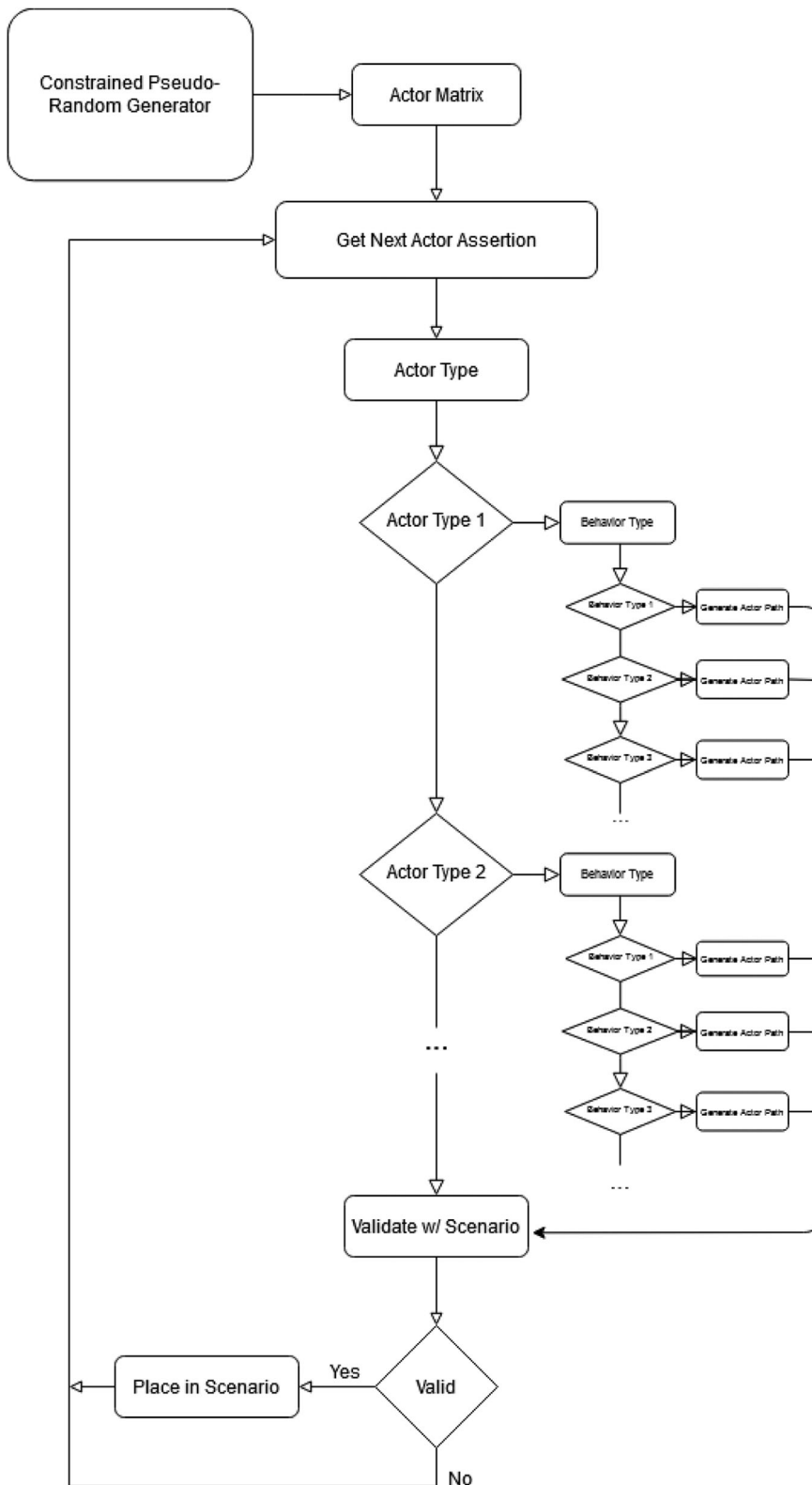


Figure 2. Actor generation loop.

test scenarios in our matrix-based system generalizes scenario characteristics and create an efficient system of labelling and sorting.

The actor matrix and road matrix define a scenario with the ego vehicle automatically generated at the start of the path. A workflow to develop the language-to-scenario methods was made that passed parameters for the number of road and actor assertions or rows to the road and actor generation methods. The numerical matrix is read as input where each row is a different assertion describing a single road piece or actor that can then be parsed to generate the scenario. To do this, road network and actors are reduced to their most basic elements in terms of Newtonian physics such as centre of mass and dimensions. These elements are then parameterized according to real-life conditions. It is important to note that the model contains no environmental factors. The overall scenario generation flowchart is given in [Figure 3](#).

An example for randomly generated road and actor matrices can be seen in [Figure 4](#). Each of these matrices was made with 10 assertions. The parameter values are selected from a uniform distribution of the valid range for each. The road starts at location (0,0) with the road piece described by the first row in the road matrix which in this case is a multilane road piece. This piece is seen at the top of the Bird’s Eye View in [Figure 5](#) and continues downward with each subsequent row that is placed. All 10 roads were placed in this scenario, as none created any conflicts of legality.

4. Implementation study

The scenario generation using the designed semantic language was implemented as a collection of MATLAB scripts. The workflow illustrated in [Figure 6](#) is used to test the scenarios.

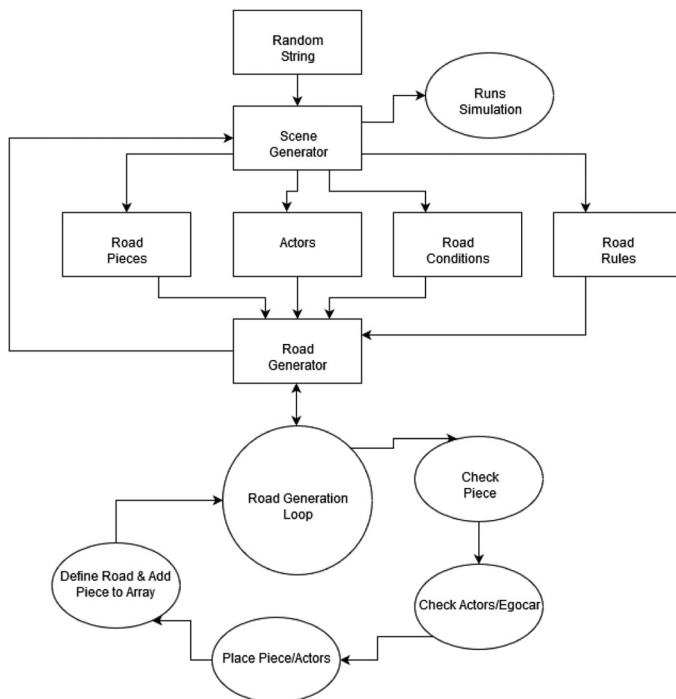


Figure 3. The flowchart for creating a test scenario.

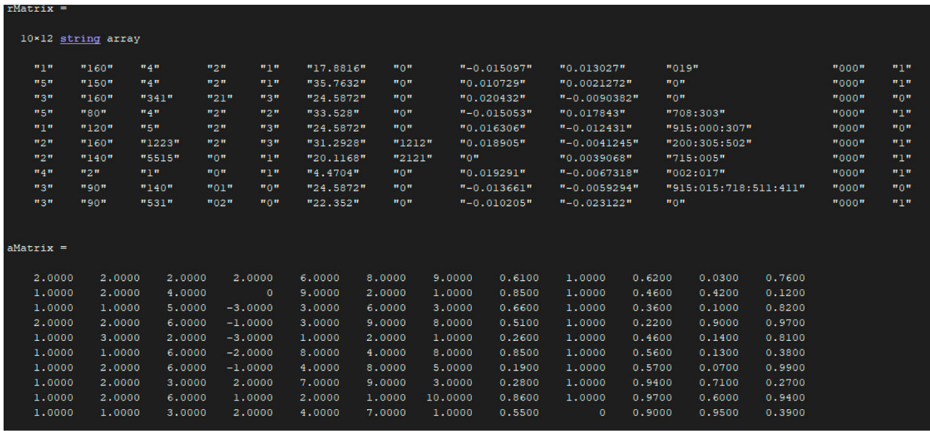


Figure 4. Randomly generated road and actor matrices.

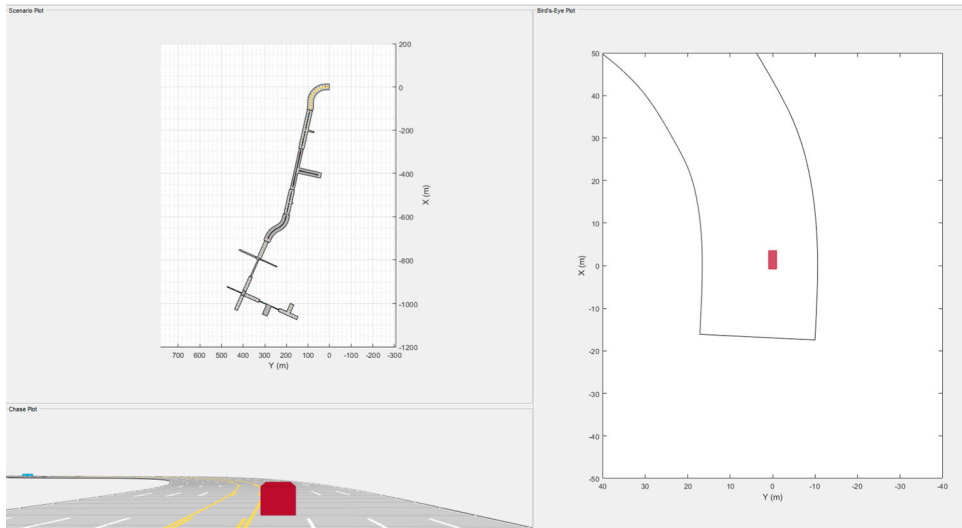


Figure 5. Sample scenario from randomly generated matrices.

This consists of calling a function *simpleRun* that takes parameters for the number of road pieces and actors desired, which then gets passed to the *runSimulations* function. This calls the function *getRandMatrix* to initialize the configuration for a scenario using random values for each of the road matrix and actor matrix's parameters from within their ranges. These matrices are then passed into a *matrix2scn* function, which constructs the scenario object. It then passes the matrices to their respective functions *road2scn* and *actor2scn* which follow Figures 1 and 2 to add the respective objects to the scenario. Finally, the simulation loop starts along with the visualization of an automatically generated ego vehicle that follows the path.

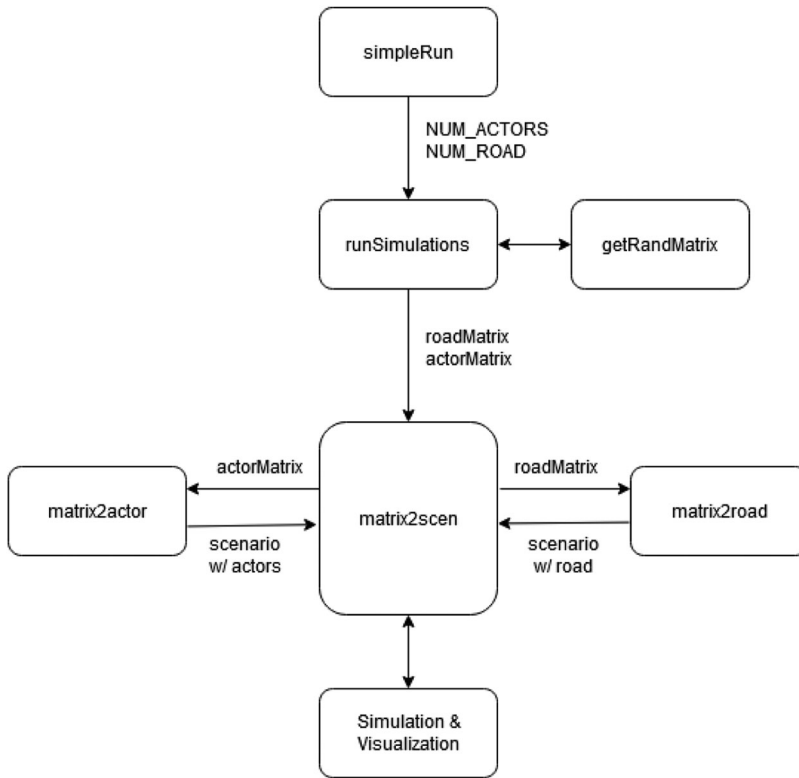


Figure 6. Scenario generation workflow.

4.1. Road pieces

Geometric centres are used for placing the roads in simulation models with the width of the road at each of those centres, the banking angles, and the details of the lanes. Then, generating the roads is a matter of taking an input and converting it into a series of points that the road follows within the context of the scene. The first value in the input defines what type of road piece will be created. As each road piece is generated, it is stitched to the previous piece by rotating it so that the tangent line along the first point is lined up with the tangent line of the last point of the previous pieces and shifting it to that coordinate in the driving scenario. Between two consecutive pieces, there is also an intermediary piece to smooth transitions between two pieces with different number of lanes. [Figure 7](#) shows a randomly created street network with multiple road types. In the following sections, we present several road piece examples and show how the parameters are used to generate these pieces.

4.1.1. Multi-lane road

One of the main goals in designing the multilane road piece was creating a building block from which all other road pieces would be built on top of. To be able to build up the other road pieces, the multilane road piece had to be as customizable as possible. Therefore, it uses most of the properties that exist as columns in the road matrix.

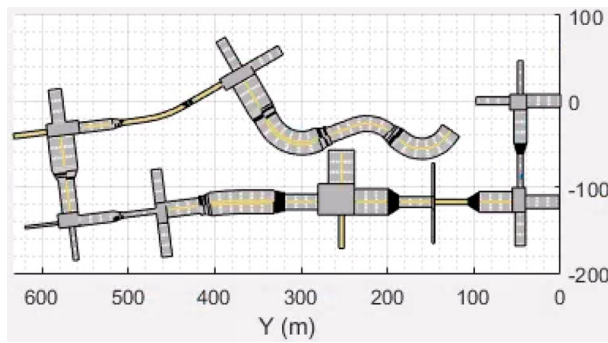


Figure 7. Example of a street network with multiple road types.

Road designers aim for making the curves G2 continuous to minimize jerk or sudden change in acceleration (McCrae & Singh, 2009). A road being G2 continuous means that it is made up of curves smoothly connected by up to two differentiations. The multi-lane road piece has variable length and width that can take on virtually all geometries with G2 continuity, which is a desired property when designing real roads. The parameters determining the geometry of the road are curvature1 and curvature2. If either of them are zero, the road is made of a line to a clothoid to an arc, where the arc is the non-zero value. If both are zero, a straight line is made. And if both are non-zero values, then either a Clothoid-Arc-Clothoid (0 to curvature1 to curvature2) or an Arc-Clothoid-Arc (curvature1 to curvature2) is generated. By composing the roads using these three primitives, the permutations should make up all possible road types. The length of each of the three parts is one third of the total length given from the determining matrix row. Figure 8 shows a multilane road example.

4.1.2. Four-way intersection

The four-way intersection creates a perpendicular intersection, where each road can have its own number of lanes and direction of travel as given in Figure 9. They are placed across from each other around a central rectangle calculated by using the widths of all four roads. Once placed, the paths are determined by distributing the possible directions to each lane, starting with left turns if possible, and following with right, and then forward. This is for the

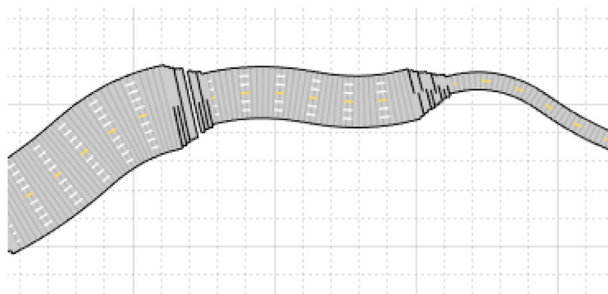


Figure 8. Multilane road example.

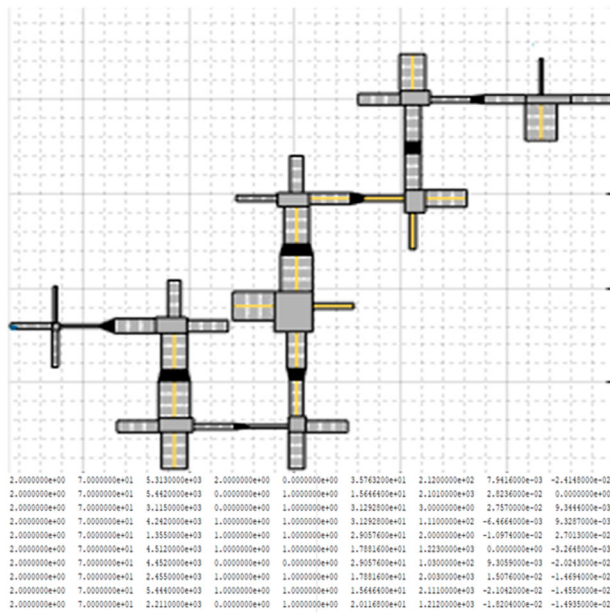


Figure 9. Four-way intersection example.

direction on the intersection that connects with the previous road piece, as that is the critical one for the vehicle under test.

4.1.3. Side entrance

The side entrance is a road piece that splits the road with a median. It also has a side entrance in the middle to a space as given in [Figure 10](#). This piece is deliberately created to encompass the logic of a real-life path. A side entrance can be highly complex for validation scenarios since multiple vehicles and actors may interact with each other at this location.

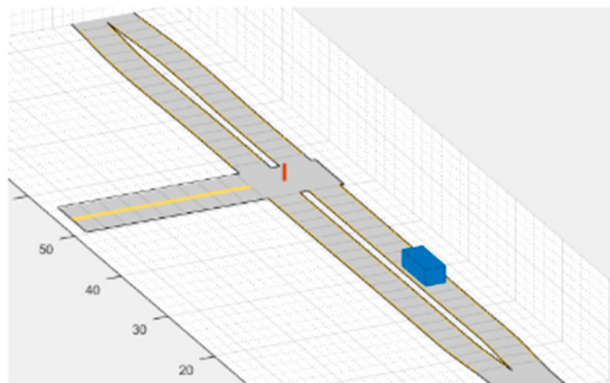


Figure 10. Side lot entry road piece example.

4.2. Actors

The actors in the scenarios can be randomly generated for both testing and demonstration purposes. Actors can follow a straight path along the road with the option to have variability in several actions. To generate actors in the scenarios with the required variability, we use the aforementioned parameters such as type, speed and dimensions. The main two actor types are given in the following sections.

4.2.1. Vehicle

The vehicle actor is generated along the path (*StartLocation* parameter) and moves forward along a lane or in the opposite direction given the *Forward* parameter and the availability of an opposite direction (the road is bidirectional). Its size is determined based on its vehicle type (*Car*, *Truck*, *Motorcycle*) and varies according to its dimensions. In MATLAB, this sets the *x*, *y*, and *z* dimensions of the actor as all actors are represented by boxes. Its path can be set to follow the lanes exactly or offset to varying degrees from the lane which also uses an offset parameter. It moves at the road's speed limit, set in the road pieces parameters, with variation according to the *MoveSpeed* parameter.

4.2.2. Pedestrian

Pedestrians are set on either side of the road somewhere along the scenario based on their initial location and their directions. Then the actors move across at an average walking speed varied by using a speed parameter. Two path options are walking straight across and walking across with a pause at some point in the middle.

4.3. Example scenario-1

Figure 11 gives an example scenario created in our system. Here, a curving multilane road with a median is constructed as well as a pedestrian actor that crosses the road. The road's geometry follows an Arc-Clothoid-Arc pattern starting with a curvature of 0.009 and

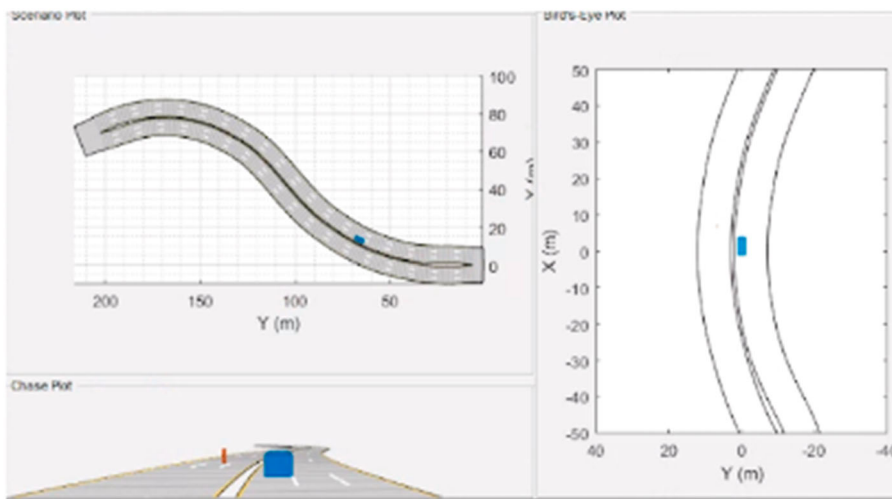


Figure 11. Sample scenario including a multilane road with a median and a pedestrian.

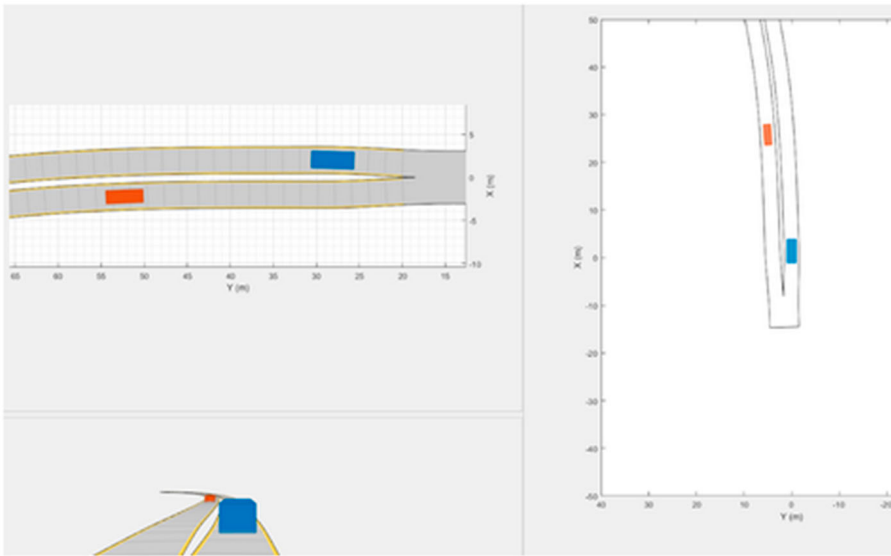


Figure 12. Ego vehicle and other actor in multilane road with median.

ending with a curvature of -0.014 , meaning the road starts off with an arc with a curvature of 0.009 for a third of the length, which is 50 m , transitions the curvature to -0.014 with a clothoid curve for a third of the length, and finishes by continuing with that curvature in an arc for a third of the length. For the pedestrian actor, a random point along the road is selected, and the point directly to the left is calculated, its path moving across to the right as the forward parameter was given as false. This actor starts moving once the scenario begins, and by the time the ego vehicle arrives, the actor has almost gotten halfway across. The ego vehicle was generated automatically.

The simulation data is collected during and after each run. The current collected data points demonstrate safety of decisions and legality of the scenario’s definition. Since the scenarios are generated by a certain input, they are available to be recreated whenever it is required.

For the actors, in order to minimize their box representation unrealistically occupying the same space, roads store information regarding the actors and at what point in the scene they are going over that lane. If two actors are at the same lane at the same point in the simulation, a different lane is used or the actor is made to slow down for a random but small period of time before moving forward. This does not apply to pedestrians as how they interact with each other is less critical in simulations.

4.4. Example scenario-2

The ego vehicle moves up a multilane road while another vehicle drives in the other direction. The other vehicle starts halfway through the road piece, and a metre-wide median separates the actors.

Road Piece: [1 130 1 2 2 22.352 0121 -0.0047779 -0.01018 '000' '000' 1]

- Road Type: 1 (Multilane Road)

- Length: 130 m
- Lanes: 1
- Bidirectional: 2 (Yes, Dashed Yellow Line)
- MidLane: 2 (Small Median)
- Speed Limit: 50 mph (from mps speed)
- Intersection Pattern: n/a
- Curvature 1: -0.0048
- Curvature 2: -0.0102
- Pedestrian PathWays: n/a
- Outlets: n/a
- Show Markers: 1 (Yes)

Actor : [1 2 1 -3 6 3 2 0.23 0 0.47]

- Actor Type: 1 (Vehicle)
- Vehicle Type: 2 (Truck)
- Path Type: 1 (Normal)
- Move Speed: -3 (-3 mps from regular speed)
- Dimensions: 6, 3, 2 (x, y, z proportions to regular truck size)
- Start Location: 0.23 (Starts 23% along path)
- Forward: 0 (Drives on reverse/opposite direction)
- Offset: 0.47 (Offset from main path)

5. Conclusion

The validation and verification is the most important tool to fill the gap between the current AV technology and its pervasive use in the market. In this paper, we present a test scenario generation method for AV simulation testing. Our method targets the validation of decision-making component of AVs. Therefore, we create a semantic language and use it to propose road network and actor generation mechanisms for random and constrained test scenario generation on a low-fidelity simulator with no consideration of environmental conditions. The current system is capable of taking a particular street network and producing the testing plan for it.

As the future work, we plan to integrate our solution with testing methods of other AV functionality layers such as perception. We also would like to enable scenario export and import features to exchange scenarios automatically with existing scenario generation tools.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Dr. M. İlhan Akbas is an Assistant Professor at the Electrical Engineering and Computer Science Department of Embry-Riddle Aeronautical University. He received his PhD degree in Computer

Engineering from the University of Central Florida, and BS and MS degrees in Electrical and Electronics Engineering from the Middle East Technical University. He has research interests in connected and autonomous cyber-physical systems, Internet of Things, modeling and simulation. He is a member of IEEE, ACM, SAE, IEEE Communications, IEEE Internet of Things and Complex Systems Societies.

Christopher Medrano-Berumen received his Master of Science and Bachelor of Science degrees in Computer Science at Florida Polytechnic University. His master thesis was on developing a methodology for autonomous vehicle validation and verification in simulation. He has research interests in validation and verification of connected and autonomous cyber-physical systems and modeling and simulation.

ORCID

Mustafa İlhan Akbaş  <http://orcid.org/0000-0002-5450-3522>

References

- Akbaş, M. İ., Brust, M. R., Turgut, D., & Ribeiro, C. H. (2015). A preferential attachment model for primate social networks. *Computer Networks*, 76, 207–226. <https://doi.org/10.1016/j.comnet.2014.11.009>
- Akbaş, M. İ., Solmaz, G., & Turgut, D. (2016). Molecular geometry inspired positioning for aerial networks. *Computer Networks*, 98, 72–88. <https://doi.org/10.1016/j.comnet.2016.02.001>
- Akbaş, M. İ., & Turgut, D. (2011). APAWSAN: Actor positioning for aerial wireless sensor and actor networks. In *IEEE Local Computer Networks (LCN)* (pp. 567–574). IEEE.
- Alnaser, A. J., Akbaş, M. İ., Sargolzaei, A., & Razdan, R. (2019, December). Autonomous vehicles scenario testing framework and model of computation. *SAE International Journal of Connected and Automated Vehicles*, 2(4), 205–218. <https://doi.org/10.4271/12-02-04-0015>
- ATG, U. (n.d.). *streetscape.gl*. Retrieved April 10, 2020, from <https://avs.auto/#/streetscape/gl/overview/introduction>.
- Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). SUMO—simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, the Third International Conference on Advances in System Simulation*. ThinkMind.
- Goodwin, A. (2019). Nvidia Drive Constellation is an online training ground for autonomous vehicles. Retrieved from <https://www.cnet.com/roadshow/news/nvidia-drive-constellation-is-an-online-training-ground-for-autonomous-vehicles/>
- Dolgov, D. (2016). *Google self-driving car project-monthly report-September 2016-on the road* (Tech. Rep.). Google.
- Dreossi, T., Fremont, D. J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., Seshia, S. A. (2019). Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification* (pp. 432–442). Springer.
- Dupuis, M., & Karl, W. (2017). Vtd-vires virtual test drive. Retrieved from <https://www.mscsoftware.com/product/virtual-test-drive>.
- Dupuis, M., Strobl, M., & Grezlikowski, H. (2010). OpenDRIVE 2010 and Beyond—Status and Future of the de facto Standard for the Description of Road Networks. In *Proceedings of the Driving Simulation Conference Europe* (pp. 231–242). INRETS, Arcueil.
- Favarò, F. M., Nader, N., Eurich, S. O., Tripp, M., & Varadaraju, N. (2017). Examining accident reports involving autonomous vehicles in California. *PLoS One*, 12(9), e0184952. <https://doi.org/10.1371/journal.pone.0184952>
- Freese, M., Singh, S., Ozaki, F., & Matsuhira, N. (2010). Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots* (pp. 51–62). Springer.

- Fremont, D. J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., & Seshia, S. A. (2019). Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th ACM Sigplan Conference on Programming Language Design and Implementation* (pp. 63–78). ACM.
- Fremont, D. J., Kim, E., Pant, Y. V., Seshia, S. A., Acharya, A., Bruso, X., Wells, P., Lemke, S., Lu, Q., Mehta, S. (2020). Formal scenario-based testing of autonomous vehicles: From simulation to the real world. *arXiv preprint arXiv:2003.07739*.
- George, M. J., & Mohamed, O. A. (2011). Performance analysis of constraint solvers for coverage directed test generation. In *ICM 2011 Proceeding* (pp. 1–5). IEEE.
- Goss, Q., Akbaş, M. İ., Jaimes, L. G., & Sanchez-Arias, R. (2019). Street network generation with adjustable complexity using k-means clustering. In *2019 Southeastcon* (pp. 1–6). IEEE.
- Halati, A., Lieu, H., & Walker, S. (1997). CORSIM-corridor traffic simulation model. In *Traffic congestion and traffic safety in the 21st century: Challenges, innovations, and opportunities urban transportation division, asce; highway division, asce; federal highway administration, usdot; and national highway traffic safety administration, usdot*.
- Hallerbach, S., Xia, Y., Eberle, U., & Koester, F. (2018). *Simulation-based identification of critical scenarios for cooperative and automated vehicles* (Tech. Rep.). SAE Technical Paper: 01-1066.
- Horni, A., Nagel, K., & Axhausen, K. W. (2016). *The multi-agent transport simulation matsim*. Ubiquity Press.
- ISO, I. (2018). 26262-1: 2018. Road vehicles—Functional safety—Part, 1.
- Jansson, J., Sandin, J., Augusto, B., Fischer, M., Blissing, B., & Källgren, L. (2014). Design and performance of the VTI Sim IV. In *Driving Simulation Conference* (pp. 128–138). VINNOVA.
- Kalra, N., & Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94, 182–193. <https://doi.org/10.1016/J.TRA.2016.09.010>
- Kehrer, M., Pitz, J., Rothermel, T., & Reuss, H. C. (2018). Framework for interactive testing and development of highly automated driving functions. In *18. Internationales Stuttgarter Symposium* (pp. 659–669). Wiesbaden: Springer Vieweg.
- Keirstead, J., Samsatli, N., & Shah, N. (2010). SynCity: An integrated tool kit for urban energy systems modelling. *Energy Efficient Cities: Assessment Tools and Benchmarking Practices*, 29, 21–42. <https://doi.org/10.1596/978-0-8213-8104-5>
- Koschi, M., Manzinger, S., & Althoff, M. (n.d.). CommonRoad: Documentation of the XML Format.
- Li, L., Huang, W. L., Liu, Y., Zheng, N. N., & Wang, F. Y. (2016). Intelligence testing for autonomous vehicles: A new approach. *IEEE Transactions on Intelligent Vehicles*, 1(2), 158–166. <https://doi.org/10.1109/TIV.2016.2608003>
- The MathWorks, Inc. *MATLAB and automated driving system toolbox* [Computer software manual]. (Release R2018b).
- McCrae, J., & Singh, K. (2009). Sketching piecewise clothoid curves. *Computers & Graphics*, 33(4), 452–461. <https://doi.org/10.1016/j.cag.2009.05.006>
- Medrano-Berumen, C., & Akbaş, M. İ. (2019, April). Abstract Simulation Scenario Generation for Autonomous Vehicle Verification. In *Proceedings of the IEEE SoutheastCon* (pp. 1–6). IEEE.
- Medrano-Berumen, C., & Akbaş, M. İ. (2020). Scenario generation for validating artificial intelligence based autonomous vehicles. In *Asian Conference on Intelligent Information and Database Systems* (pp. 481–492). Cham: Springer.
- Menzel, T., Bagschik, G., Isensee, L., Schomburg, A., & Maurer, M. (2018). Detaillierung einer stichwortbasierten Szenariobeschreibung für die Durchführung in der Simulation am Beispiel von Szenarien auf deutschen Autobahnen-english title: Detailing a Keyword Based Scenario Description for Execution in a Simulation Environment Using the Example of Scenarios on German Highways. In *Workshop Fahrerassistenzsysteme und Automatisiertes Fahren* (Vol. 12, pp. 15–26). Technische Universität Braunschweig.
- M-SDL (2019, September). [Computer software manual]. Version 0.9.
- Nair, V. G., & Wishart, J. (2018). A study of driving simulation platforms for automated vehicles. CAV Final Report, Arizona State University.
- Nowakowski, C., Shladover, S., & Chan, C. (2015). Behavioral competency requirements methodology project background regulatory issues and potential regulatory strategies for highly Automated Vehicles (AVs) how to ensure safety prior to deployment? In *Automated Vehicles Symposium*. AVS.

- Razdan, R., Akbas, M. I., Sargolzaei, A., Alnaser, A. J., Sahawneh, S., Alswiss, S., & Vargas, J. (2019, June). *Unsettled Technology Areas in Autonomous Vehicle Test and Verification*. SAE (Society of Automotive Engineers) EDGE™ Research Report EPR2019001.
- Rentrepe, J., & Akbaş, M. I. (2017). Spatially adaptive positioning for molecular geometry inspired aerial networks. In *Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications* (pp. 1–8). ACM.
- Sarhadi, P., & Yousefpour, S. (2015). State of the art: Hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software. *International Journal of Dynamics and Control*, 3(4), 470–479. <https://doi.org/10.1007/s40435-014-0108-3>
- Tideman, M., & Van Noort, M. (2013). A simulation tool suite for developing connected vehicle systems. In *2013 IEEE Intelligent Vehicles Symposium (iv)* (pp. 713–718). IEEE.
- Waymo safety report: On the road to fully self-driving (2018).