A STUDY OF DRIVING SIMULATION PLATFORMS FOR AUTOMATED VEHICLES

by

Vaishakh Gopalakrishnan Nair

Submitted to:

Dr. Jeffrey Wishart

ARIZONA STATE UNIVERSITY

November 2018

ABSTRACT

The following literature review talks about the driving simulation platforms commercially available for automated vehicle development. It is also a comparison of the simulation packages, their advantages and drawbacks, and an insight into what is missing in the simulators of today. Automated vehicle safety and reliability are the important requirements when developing automated vehicles. These requirements are guaranteed by extensive functional and performance tests. Conducting these tests on real vehicles is extremely expensive and time consuming, and thus it is necessary to develop a simulation platform to perform these tasks. In most cases, it is difficult for system or algorithm developers in the testing process to evaluate the massive design space. To test any algorithm change, developers need to test a functional module alone, and later setting up a whole physical testing environment that consists of several other modules, leading to enormous testing costs. Fortunately, many of the testing tasks can be accomplished by utilizing simulator. The key to the success of a simulation is how accurately the simulator can simulate the physical reality.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who provided me the possibility to complete this report. A special gratitude I give to my professor for EGR 598: Connected and Automated Vehicle course Dr. Jeffrey Wishart, whose contribution in stimulating suggestions and encouragement, helped me in writing this report.

Furthermore, I would like to acknowledge with much appreciation the role of Dr. Rob Gray at Arizona State University in giving me valuable suggestions and pushing me in the right direction for completing this report. I would also like to thank all my colleagues for their valuable help provided during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Automated driving represents an imminent challenge consisting of several domains including robotics, computer vision, motion planning, and simulation. Central to this challenge are the safety considerations of automated vehicles navigating the roads surrounded by an unpredictable crowd. Humans, whether drivers, pedestrians, or cyclists, often behave erratically, inconsistently, forcing other vehicles (including automated vehicles) to react quickly to avoid hazards. To provide acceptance and guarantee safety, vehicles must be tested not only in typical, relatively safe scenarios, but also in dangerous, less frequent scenarios.

Apart from safety concerns, costs pose an additional challenge to the testing of automated driving algorithms. Each new configuration of a vehicle or new sensor requires re-calibration of a vehicle, which is labor intensive. Furthermore, the vehicle can only be tested under conditions limited either by a testing track, or the current traffic conditions if a road test is being performed. This means that the vehicle can be tested no faster than real-time and without any speedups or parallel testing. Many recent approaches to automated driving rely on machine-learning via deep-learning to provide entity detection, entity prediction, and end-to-end control.

However, such approaches rely on large amounts of annotated data in safe and dangerous scenarios. The dataset must also contain varied weather and lighting conditions. In addition, not all automated vehicles are equipped with identical or equivalent sensing capability; training data must be available for the specific configuration or sensors of the vehicle being tested. Gathering such data by physical tests can be expensive, difficult and even dangerous. In contrast, a high-fidelity simulator can augment and improve training of algorithms and allow for testing safely and efficiently. Insights gained from simulation could provide important training data and information on algorithmic inaccuracies before actual vehicle testing.

One of the hardest problems in the development of road traffic simulation tools is the case of intersection. The common approach, used in most traffic simulation tools, involves a drastic simplification of the problem. A ‘‘solver’’, which must manage traffic inside the intersection, lets vehicles enter only when their trajectories are not in conflict [2]. Such solutions are sometimes enough but are not acceptable when the aim of the simulation is to mimic the actual behavior of real drivers. The accuracy of such simulations is more important because driver behavior in intersections influences the overall flow on the road network.

Sensor and motor control in three-dimensional environments remain an obstacle in machine learning and robotics. The development of automated ground vehicles is a long-studied instantiation of this problem. Its most difficult form is navigation in densely populated urban environments. This setting is particularly challenging due to complex multi-agent dynamics at traffic intersections; the necessity to track and respond to the motion of tens or hundreds of other actors that may be in view at any given time;

prescriptive traffic rules that necessitate recognizing street signs, street lights, and road markings and distinguishing between multiple types of other vehicles; several rare events such as road construction, a child running onto the road, an accident ahead, a rogue driver speeding on the wrong side; and the necessity to rapidly reconcile conflicting objectives, such as applying appropriate deceleration when a pedestrian strays onto the road ahead but another car is rapidly approaching from behind and may rear-end if one brakes too hard.

Research in automated urban driving is hindered by infrastructure costs and the logistical difficulties of training and testing systems in the physical world. Instrumenting and operating even one robotic car require significant funds and manpower. And a single vehicle is far from sufficient for collecting the requisite data that cover the multitude of corner cases that must be processed for both training and validation. Training and validation of driving control models for urban driving in the real world is beyond the scope of most research groups.

An alternative is to train and validate these control models in simulation. It is also necessary for system verification, since some scenarios are too dangerous to be recreated in the physical world (e.g., a child running onto the road ahead of the car). Simulation has been used for training driving models since the early days of automated driving research. More recently, racing simulators have been used to evaluate new approaches to automated driving [5]. Custom simulation setups are commonly used to train robotic vision systems. And commercial games have been used to acquire required data for training and benchmarking visual perception systems.

CHAPTER 2
HISTORY AND OVERVIEW OF DRIVING SIMULATORS

Simulation has been an integral tool in the development of controllers for automated vehicles. Many successful vehicle demonstrations of automation were first tested in simulation. Recent work in traffic modeling has sought to increase the accuracy of the modelled drivers and vehicles. The very first simulation systems started with the beginning of air force pilots training procedure. Training in real planes resulted in higher training cost and high danger rate for novice pilots. After the successful launch of pilot training simulators, other vehicles simulators rolled out quickly. The most widespread are the car driving simulators. The simulator systems used to be developed for use in Training, Research and Entertainment. Originally, different ways of use of the simulators in the above noted fields have merged during decades of their use and development.

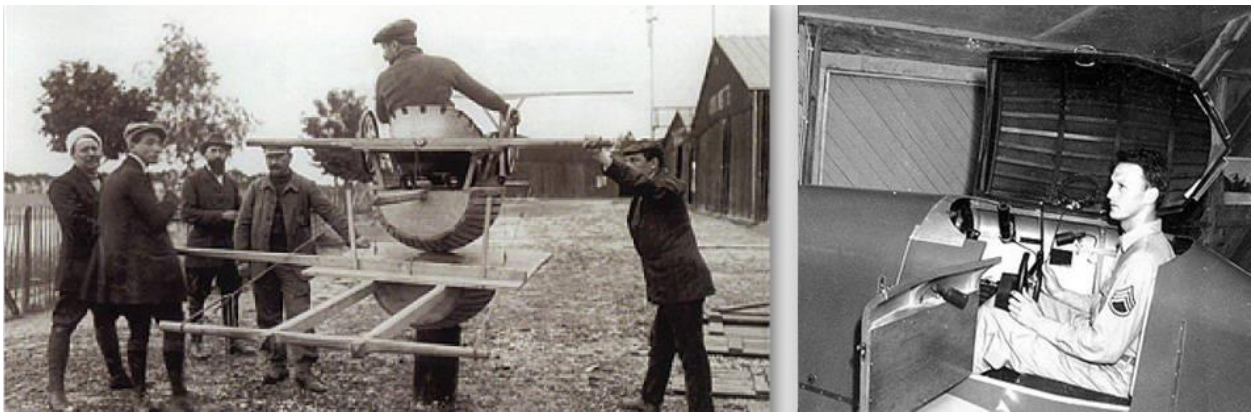


Fig. 2.1: First flight simulators used in 1910 (Source: [1])

The first traffic flow models were mathematical. Such a model allows, for instance, a highway traffic situation to be modeled using car-following laws, which are, in fact, differential equations that are obtained empirically through regression using data collected at currently operating road sections. Even now, most of the microscopic simulations use the car following law to model in-line driving [28].

To make up for the practical limitations of real-world driving, automotive companies are currently using software simulators to train automated vehicles. Simulators recreate in digital form the software and hardware components of CAV technology, as well as the virtual world in which such vehicles operate. Simulators help engineers test the effectiveness of an automated vehicle, as well as optimize the system. Software simulation has two main parts:

1) *The training agent*: the simulation of the vehicle to train;

2) *the environment*: the simulation of the dynamic environment in which the vehicle operates, including elements such as road textures, other cars and pedestrians, the road geometries, etc.

Simulators are not new technology. Engineers in aerospace, robotics, and automotive have used them for development and research for the last couple of decades to simulate how such machines act in the real world.

However, the thing that is recent in this market is the emergence of startup companies developing simulation products specifically tailored to meet the needs of automated driving. Some of the notable companies include Cognata, CARLA, METAMOTO, etc. Such companies provide an overall solution for AV simulation, including simulation of both the AV car undergoing training and the environment in which it operates. Other companies, such as ANSYS, also provide engineering simulation solutions but release their products across multiple sectors, from automotive, aerospace, consumer products, and so forth, as opposed to solely concentrating on the automotive market [26].

The lack of developed simulation solutions historically and even today caused most automotive companies to end up building AV simulators of their own, utilizing existing open-source and proprietary technologies as well. For example, an automotive company with their own simulation platform may use the Unity or Unreal physics engine to model how moving objects would interact in a certain environment; it may utilize the libraries of OpenScenario to run different scenarios in the simulation; and it may build its own software module for modeling the AV's path and behavioral planning. However, building simulators often falls far outside the core research of automotive companies. Automotive companies are better suited toward tackling the software and hardware that make it into the car rather than building the software tools for simulation of these components. Hence automotive companies have begun to use the simulation solutions provided by the startup companies. Automotive vehicles would have to drive hundreds of millions of miles or in some cases, billions of miles in order to achieve or satisfy the required reliability. The following table shows an example of the miles and years needed to satisfy automated vehicle reliability.

BENCHMARK FAILURE RATE				
STATISTICAL QUESTION	Miles (Years) automated vehicles must be driven	A) 1.09 fatalities per 100 million miles	B)77 reported injuries per 100 million miles	C) 190 reported crashes per 100 million miles
	To demonstrate at 95% confidence that failure rate is at most…	275 million miles (12.5 years)	3.9 million miles (2 months	1.6 million miles (1 month)
	To demonstrate at 95% confidence their failure rate to within 20% of the true rate of…	8.8 billion miles (400 years)	125 million miles (5.7 years)	51 million miles (2.3 years)
	To demonstrate at 95% confidence and 80% power that the failure rate is 20% better than the human driver failure rate of…	11 billion miles (500 years)	161 million miles (7.3 years)	65 million miles (3 years)

Table 2.1: Examples of miles and years needed to determine automated vehicle reliability (Source: [32])

The path for AV-simulation startups to earn big by focusing just on the automotive industry alone will be difficult. Automotive companies have already developed a lot of their simulation software using open-source and proprietary solutions. As such, they are unlikely to pursue all-inclusive platform solutions from simulator startups but only purchase certain software modules for areas like scenario-generation or the training environment for which they do not already have urbane solutions. Together with the limited number of automotive OEM's, Tier-1's, and even Tier-2's that would potentially use such software, the market opportunity for simulation startups is extremely small. The market is no larger than a few hundred million dollars.

Simulation startups set solely on the automated driving problem therefore face a difficult battle ahead. A company may still be able to become a big business in this space, but it will have to tackle other market verticals outside of automotive only. ANSYS is one multi-vertical simulation company well-positioned here. Further, companies that can provide the necessary technologies in line with the simulation market also have significant room to grow. Rescale is one such company. It provides high-performance computing set-up to run simulations for various applications across multiple industries. Other than such uncertain solutions, opportunities with the AV simulation space are otherwise limited.

OVERVIEW OF EFFICIENT VIRTUAL SIMULATION SOFTWARE

Whenever things become complex, they need to be divided into smaller pieces. Also, when the development of a product is complex, not only the product, but also the development process needs to be structured to become manageable. Both in the areas of software development and systems engineering, various such processes exist. An apt process can be chosen based on the needs of the project in terms of complexity, criticality, and urgency of the product. However, due to the different nature of virtual software and physically existing hardware, these development processes differ greatly. Due to its immaterial nature, software can more easily be modified and evolved than physically existing hardware. Therefore, software development processes can be iterative and incremental to allow an evolutionary process towards a stable and robust result. Iterations are necessary to continuously evaluate the current state of the user's needs on the one hand. On the other hand, iterations enforce to continuously incorporate existing artifacts to a functional system. Thus, iterations are a vital part to deal with the necessary improvement of existing software, changing requirements, and an incremental process that breaks up software into smaller pieces. The more innovative software products are, the less predictable a software development process becomes. In such a context many small iterations are preferred over a few larger ones. More and smaller iterations allow the software to adjust priorities and to lead the project more successfully [33].

Simulations can duplicate not only the software and hardware dynamics of a given car, but also the environment in which an AV would drive (e.g. the road, other moving cars, pedestrians, etc.). Accordingly, simulators solve the biggest practical confines of real-world training in that 1) they are much cheaper and faster to run and 2) they permit the testing of scenario edge cases (e.g. car crashes, ethical problems, etc.) that would be impossible to run in real-life [7].

FIELDS OF APPLICATION

With the development of driving simulators, automated vehicle simulation was regarded as vital, through its influence on driving tasks (speed control, lane change maneuvers, etc.) and on mental load of the driver for ergonomics studies. Today its use is flared and spreads out to many fields.

The main fields for vehicle simulation are:

1) *Design and improvement of car equipment* - The simulation is a powerful tool for evaluation. It allows saving time between authentication and physical realization (driving aid systems, headlights, etc.).

2) *Training* - Real time simulation is progressively used to educate and train personnel (traffic control centers) or for vehicle driving training (trucks, buses, cars, etc.).

3) *Testing new structures* - Transportation facilities are costly reserves. Simulation can be applied to quantify traffic performance according to different design options before the dedication of resources to construction (roads, real estate, etc.).

4) *Security and environment* - Some applications in this field are: intelligent highways and intelligent vehicles studies, driver behavior analysis in dangerous situation, accident reconstruction and exhaust fumes emissions level evaluation.

5) *Research* - Traffic simulation is used for mathematical and statistical studies with an aim at improving traffic-flow models. It can also be used for the evaluation of alternative treatments by enabling to control the experimental environments and the range of conditions to be explored (signal control strategies, Advanced Traffic Management Systems, etc.) [31].

SIMULATION MODELS CLASSIFICATION

Simulators and simulation models can be classified in agreement with several factors. Various approaches can depend on the specificity of the application and its restrictions.

1) *Time* - Almost all the vehicle simulation models describe dynamical systems; time is always the essential independent variable. Discrete models, unlike continuous models, denote real world systems by declaring that their states change at points in time. There are two kinds of discrete models: discrete time models and discrete event models. Within each interval of time, discrete time models calculate the activities that change the states of system components. Discrete event models remain constant until a change of state occurs.

2) *Level of detail* - According to the level of detail selected to represent the system that is under study, which mainly depends on the available computation ability, a simulator is macroscopic, mesoscopic or microscopic. A macroscopic model describes objects and their activities and interactions at a low level of detail. The traffic stream may be represented in some collective manner or by scalar values of flow rate and density. For example, individual lane changes are not characterized; the model provides global quantitative or qualitative information. A mesoscopic model generally represents most objects at a high level of detail but describes their activities and interactions at a lower level. For example, lane change maneuvers are represented but could be performed as a rapid event. A microscopic model describes most objects and their interactions at a high level of detail. For example, a lane change could

summon a car-following law with respect to its current leader, then with respect to its assumed leader and follower in the target lane.

3) *Deterministic model and stochastic model* - When one wants to reproduce specific conditions or events, probability plays a major part. Two types of model are to be considered. Deterministic models have no random variables; all object interactions are defined by exact relationships (mathematical or logical). On the contrary, stochastic models have methods which include probability functions. Deterministic models are well suited to experiments whose situations are intended to be completely reproducible. On the other hand, the behavior of the vehicles may appear too mechanical and repetitious and realism is somehow lessened.

4) *Design methodology* - The design methodology used for the model may also be considered. A model can be programmed in a progressive way, as for some macroscopic simulators. It can also be designed according to object-oriented standards, as for many microscopic simulators. Today, another method is also used. This method, resulting from work carried out these last ten years in dispersed artificial intelligence, is called the agent-oriented method [31].

CHAPTER 3
STATE OF THE ART SIMULATION PLATFORMS

Achieving safe and correct automated driving systems is targeted from multiple perspectives. The first approach is to design planning and control functions such that they can cope with all situations. However, the simplifications of these methods applied for gaining efficiency can lead to undesired behavior in some situations. Therefore, some researchers try to prove their planning principle to work in all situations. For a good coverage, the simulation needs to consider nondeterministic events. Such events are automatically included in real world tests. The disadvantage of such tests is a worse cost efficiency. The rest of the section talks about some of the most common automated vehicle simulation platforms used in the industry today.

A. CARCRAFT (WAYMO)

In Alphabet's campus, there is a team working on a piece of software that may be the key to self-driving cars. The engineers call it Carcraft, after the popular game World of Warcraft. If Waymo can supply fully automated vehicles in the next few years, Carcraft should be recollected as a virtual world that had a massive role in reshaping the actual world on which it is based. Initially developed to recreate scenes that the cars experienced while driving on public roads, Carcraft, and simulation generally, have taken on an ever- greater role within the self-driving program. Originally, they primarily used the tool to see what their cars would have done in complicated situations in which human drivers have taken over control of the car. And they started creating scenarios from these moments. The spatial extent of Carcraft's abilities grew to include whole cities, the number of cars grew into a huge virtual fleet [3].

Fig. 3.1: Carcraft's recreation of Chandler, Arizona (Source: [3])

The car uses cameras, radar, and laser scanning to detect objects in its field of view and it represents them in the software as small wireframe shapes, outlines of the real world. With a press of a button, the objects on the screen begin to move. Cars act like real-life cars, driving in their lanes and turning, so do the cyclists. Their logic has been developed from the millions of miles of public-road driving the team has done. Underneath it all, there is that very detailed map of the world and models for the physics of the different objects in the scene. They have modeled both the tire and the road. Not surprisingly, the toughest thing to simulate is the behavior of the other people.

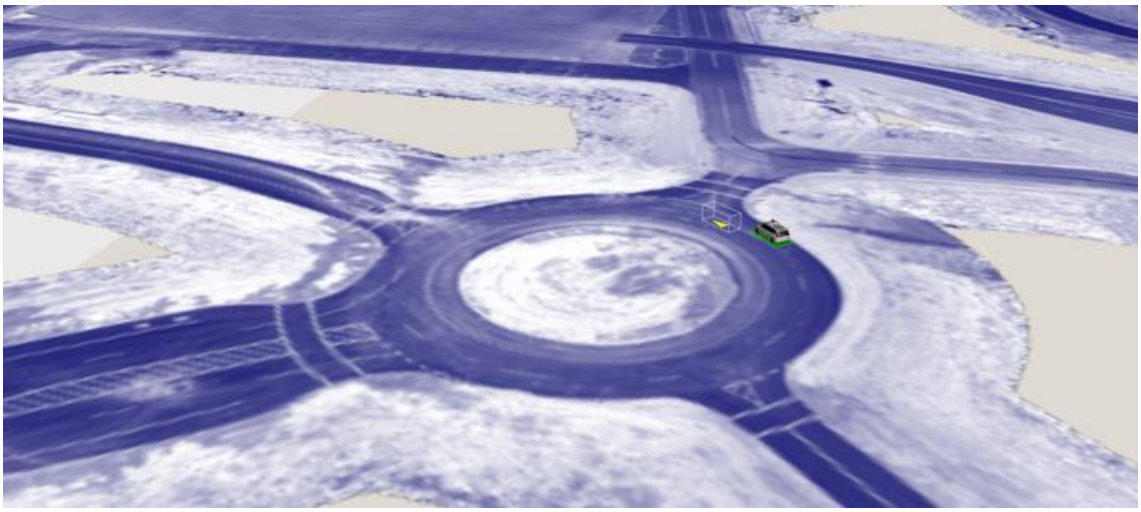


Fig.3.2: Carcraft's recreation of Waymo testing ground 'Castle' (Source: [3])

There is one important difference: In the real world, they must take in new, real-time data about the environment and convert it into an understanding of the scene, which they then navigate. But now, after years of work on the program, the Carcraft team feel assured that they can do that because they've run a bunch of tests that show that they can recognize a wide variety of pedestrians [3]. So, for most simulations, they skip the object-recognition step. Instead of feeding the car crude data it must identify as a pedestrian, they simply instruct the car: A pedestrian is here. The different objects can be instructed to follow the logic Waymo has modeled for them or the Carcraft scenario builder can program them to move in a specific way, in order to test specific behaviors.

Once they have the basic structure of a scenario, they can test all the important disparities it contains. So, imagine, for a four-way stop, one might want to test the arrival times of the several cars and pedestrians and bicyclists, how long they stop for, how fast they are moving, and so forth. They simply put in sensible ranges for those values and then the software creates and runs all the permutations of those scenarios.

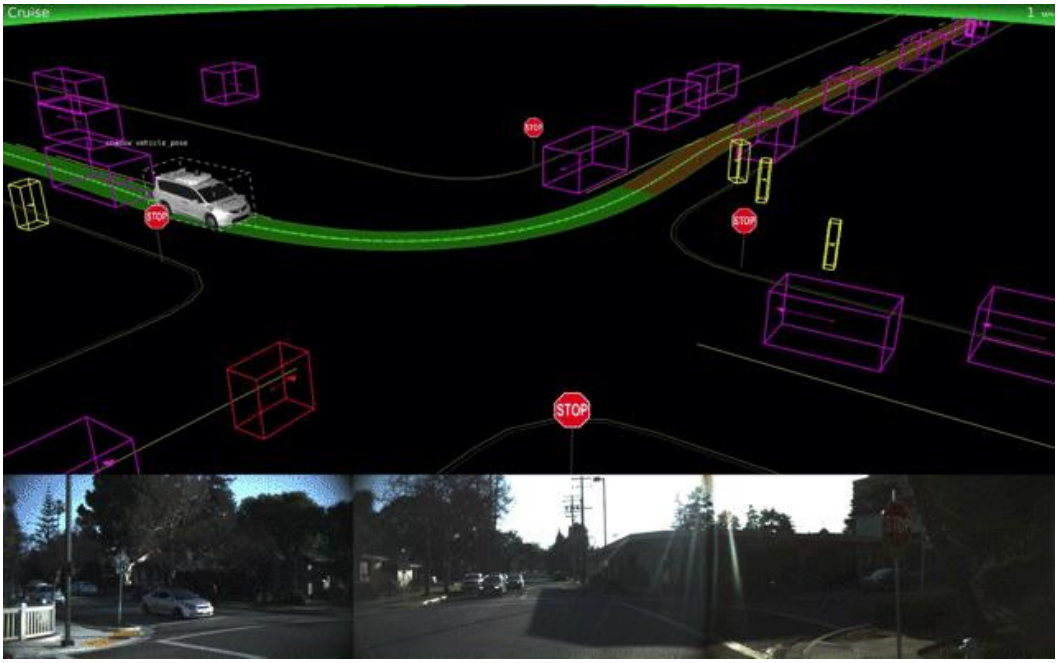


Fig. 3.3: Waymo's improved vehicle navigation using Carcraft (Source: [3])

It is called "fuzzing," for example, 800 scenarios are generated by a four-way stop. It creates a graph where engineers can see how different combinations of variables change the path that the car would decide to take [3].

The engineers might want to look for very long decision times or braking profiles outside the selected range. Anything that engineers are working on learning or adjusting, they simulate looking for problems. There are three core aspects to simulation. One is that they drive a lot more miles than would be possible with a physical fleet. Two, those miles emphasize on the interesting and difficult interactions for the cars rather than uninteresting miles. Finally, the development cycles for the software can be much faster.

The power of the computer-generated worlds of Carcraft is not that they are a realistic and attractive rendering of the real world. The power is that they reflect the real world in the ways that are significant to the self-driving car and allow it to get billions more miles than physical testing would permit. For the driving software running the simulation, it is not similar to making decisions out there in the real world, but it is exactly the same as making decisions out there in the real world.

B. METAMOTO

Silicon Valley startup Metamoto allows companies working on automated technology to confirm automated vehicle (AV) safety in the virtual world before they hit public roads.

Metamoto's offering consist of three components:

- *Director*: scalable, parameterized simulator allowing for tests to run across a range of environmental and hardware parameters and unique edge cases.
- *Designer*: a first-of-its-kind tool to custom build parameterized scenarios via a graphic editor. Scenarios describe the training and testing behaviors of traffic, pedestrians, sensor configurations, etc. within a virtual scene.
- *Analyzer*: a complete analysis tool to assess how a vehicle performed in specific simulations. This tool is often named as an automated systems debugger.

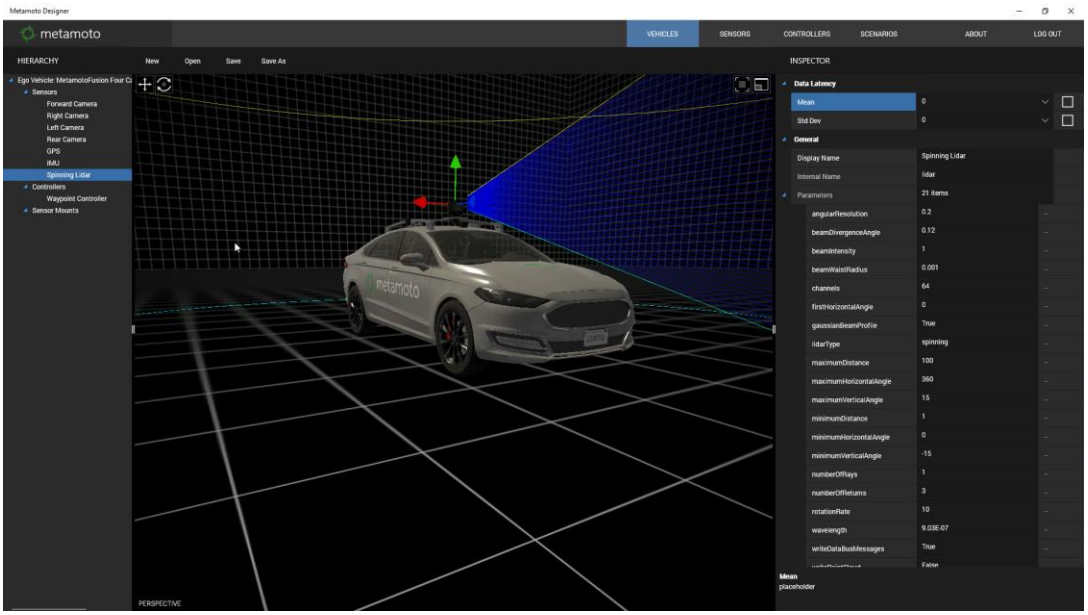


Fig. 3.4: Simulation of LiDAR sensor using Metamoto on Ford Fusion (Source: Metamoto, 2017)

Key features of the service include:

- *scalability*: the ability to run millions of tests in a single cycle, eventually driving billions of virtual test miles, to identify unique outcomes, performance boundaries and system acceptances.
- *parameterization*: simulations are executable across a range of environmental (weather, traffic, road conditions, pedestrians, etc.) and hardware (vehicle properties, sensor placement, latency, sensor settings, etc.) parameters.

- *continuous test and integration*: on-demand simulation is inserted into the AV product lifespan with tools like Jenkins and Jira, allowing for smooth regression testing, agile workflows, version control and more, every time vehicle software, sensors and/or infrastructure change.
- *accurate simulation*: accurate, integrated simulation of a variety of sensors is available including LiDAR, camera, radar, GPS, IMU and others.
- *interoperability*: Metamoto's offering is supplied as an add-on to other mobility solutions. For example, Metamoto and Renovo recently announced a partnership to make simulation solutions obtainable to Renovo's AWare ecosystem [4].

C. CARLA

Intel Labs, the Toyota Research Institute and the Computer Vision Center in Barcelona, Spain have created an open-source driving simulator that automotive manufacturers can use to test self-driving technologies under realistic driving conditions. The system, called CARLA (Car Learning to Act), is an open-source software that simulates a wide range of driving conditions and repeats unsafe situations endlessly to help learning. The team has already used it to assess the performance of several different approaches to automated driving.
It includes urban layouts, a variety of vehicle models, buildings, pedestrians, street signs, etc. The simulation platform supports flexible setup of sensor groups and provides signals that can be used to train driving strategies, such as GPS coordinates, speed, acceleration, and detailed data on collisions and other violations. A variety of environmental conditions can be specified, including weather and time of day. And these systems do not give the kind of technical feedback that automated driving systems need to learn.

CARLA offers a library of resources that can be arranged into towns under various weather and lighting conditions. The library includes 40 different buildings, 16 dynamic vehicle models, and 50 animated pedestrians. The team has used these to create two towns with several kilometers of traversable roads and then tested three different approaches to training self-driving systems. The scenarios are evaluated in cumulative difficulty [5].

Fig.3.5: Simulation of a vehicle under varied weather conditions (Source: CARLA, 2017)

CARLA uses three methodologies to study the performance of automated driving. The first is a classic integrated pipeline that comprises a vision-based perception module, a rule-based planner, and a maneuver controller. The second is a deep neural network that maps sensor inputs to driving commands, trained end-to-end via imitation learning. The third is also a deep network, trained end-to-end via emphasis learning. CARLA uses stage-controlled aim -directed navigation scenarios of growing difficulty. The user controls the complexity of the route that must be traversed, the presence of traffic, and the environmental conditions [33].

Fig. 3.6: Diversity of cars and pedestrians available in CARLA (Source: CARLA, 2017)

D. VRXPERIENCE by ANSYS

VRXPERIENCE offers an immersive driving simulation setting with SCANeR, including scenarios, traffic, and vehicle dynamics run time. The user can also create custom virtual road environments and testing scenarios. VRXPERIENCE is also interfaced to other vehicle dynamics (CarSim) and complete driver hardware simulator interfaces (SensoDrive) for the best and most accurate simulation.

It is a fully virtual driving lab for testing lighting systems in a regulated environment. It can reduce the night road tests and virtually evaluate headlamp performance, with real-time comparison and a connection to logical simulation. The user can clearly account on night-driving simulations, and virtually analyze the efficiency of headlamps compared to older headlamps or even to the headlamps of competitors, simply with a change of configuration. One can enhance the lamp light distribution design, the development of smart headlamps and test drive replacement. This way, one can validate the control law for higher quality, and strongly reduce the risk of finding problems too late in the development process. It has real time LED Matrix Beam, up to 500 pixels each side and real-time dynamics lighting strategy.

VRXPERIENCE allows testing and authentication of the full cockpit design for HMIs, including virtual displays and actuators, through visual simulation, eye and finger tracking, and touch feedback. VRXPERIENCE provides a full HMI (Human Machine Interface) evaluation for next-generation vehicles, using virtual reality. This tool reduces the time and cost of design, since the assessment of the design is mostly performed on virtual prototypes, reducing the number of expensive physical prototypes necessary to create the product. VRXPERIENCE offers cooperative driving scenarios based on virtual HMIs, considering human factors analyses and cognitive workloads. The test driver can interact directly with the virtual interfaces, from touchscreens to switches, by a fine finger tracking system. As the system records the behavior of the driver and displays driving and infotainment information, it identifies and understands the actions of the pilot and triggers the adapted HMI reaction automatically. One can thus easily evaluate the significance of the displayed information, in real time, for a safer drive.

VRXPERIENCE readily incorporates the simulation of ground-truth sensors and camera and lidar sensor types. It has strong graphical visualization capabilities that enable you to assess your complex ADAS systems and automated vehicles virtually, by connecting optical and functional operations in a single drive simulator. It uses the ground truth sensor (GTS) simulation for several sensors such as Camera, Lidar, Ultrasonic and Radar sensors.

Fig. 3.7: Simulation of vehicle sensors using VRXPERIENCE (Source: ANSYS, 2018)

It has powerful ray-tracing capabilities to recreate sensor behavior, and easily obtain sensor results through a dedicated interface. This solution provides a unique method to collect virtual sensor information during driving and use the information to develop autopilot code [6].

E. AUTONOVI-SIM by UNC GAMMA

AutonoVi-Sim, a novel high-fidelity simulation platform for automated driving data generation and driving strategy testing. AutonoVi-Sim is a collection of high-level extensible modules which allows the swift development and testing of vehicle configurations and enables construction of complex traffic scenarios. It supports numerous vehicles with unique steering or acceleration limits, as well as unique tire parameters and dynamics profiles. Engineers can specify the specific vehicle sensor systems and alter time of day and weather conditions to generate robust data and gain insight into how conditions affect the performance of an algorithm. In addition, AutonoVi-Sim supports navigation for non-vehicle traffic members such as cyclists and pedestrians, allowing engineers to specify routes for these artists, or to create scripted scenarios which place the vehicle in dangerous sensitive situations. Autonovi-Sim facilitates training of deep-learning algorithms by allowing data export from the vehicle's sensors, including camera data, LIDAR, relative positions of traffic participants, and detection and classification results. Thus, AutonoVi-Sim allows for the rapid prototyping, development and testing of automated driving algorithms under changing vehicle, road, traffic, and weather conditions.

AutonoVi-Sim is divided into eight extensible modules, each with several sub-components. The modules are Environment, Road Network, Road, Drivers, Infrastructure, Vehicles, Pedestrians, and Analysis. Each module captures some aspect of automated driving simulation and can be extended and modified to match the specific needs of an algorithm.

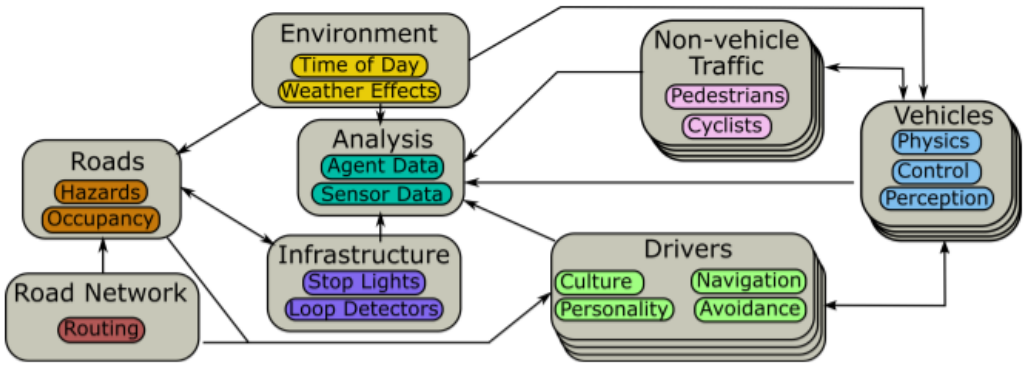


Fig. 3.8: AutonoVi-Sim platform overview and logic flow (Source: [7])

AutonoVi-Sim captures numerous automated driving phenomena and testing requirements including:

• Data Generation: AutonoVi-Sim enables data analysis by allowing exports of relevant data for traffic nearby to the automated vehicle as well as data from each virtual sensor on the vehicle. Sensor and local

traffic data can be used in training deep-learning methods by generating automatically labelled classification and decision data efficiently.

• Changing vehicle, cyclist, pedestrian, and traffic conditions: AutonoVi-Sim includes several vehicle and sensor models, pedestrians, and cyclists. Diversity of these traffic units allows for training classification on differing shapes, sizes, colors, and behaviors of cyclists, pedestrians, and other drivers.

• Dynamic Traffic, Weather and Lighting Conditions: AutonoVi-Sim delivers high fidelity traffic simulation, supporting dynamic changes in traffic density, time of day, lighting, and weather.

• Rapid Scenario Construction: Conventional road networks can be easily laid out using spline painting and are automatically connected for routing and navigation purposes. AutonoVi-Sim supports many lane configurations and unusual road geometry such as cloverleaf overpasses. In addition, other vehicles and entities can be scripted to generate repeatable inconsistent behavior, e.g. cutting in front of the ego-vehicle, walking into the road.

AutonoVi-Sim currently lacks calibration information to duplicate specific sensors and sensor configurations. In the future, specific sensing packages and algorithms to test real-world configurations will be developed. In addition, it will be beneficial to explore the movement between algorithms trained on AutonoVi-Sim and actual test vehicles. The present driver models are limited to hierarchical, rule-based driving approaches. The present simulator supports few hundreds of vehicles. By merging the simulator with macroscopic or hybrid traffic simulation approaches, the size of supported traffic conditions may be increased [7].

F. COGNATA

Audi is directing to Israeli startup Cognata to help the car manufacturer validate its automated vehicles in the virtual world before they head out on the road for testing. Automated Intelligent Driving, Audi's self-driving unit led by a team of former Microsoft, Tesla and internal Audi experts uses Cognata's automated vehicle simulation platform to test and develop its technology.

The deal also highlights the rising ecosystem of Israeli startups, many of which developed technology initially designed for military use, such as drones and other defense applications, only to find a eager customer base within the automated vehicle industry.

Cognata raised $5 million in 2017 from Airbus Ventures, Emerge and Maniv Mobility, recreates cities in its 3D simulation platform to give customers numerous testing scenarios. The platform drags in layers of data to help build these virtual environments. It begins with recreating real cities, then adds AI-based traffic models to simulate real-world conditions, as well as data from the vehicle's sensors [8].

Cognata Ltd. launched the first cloud-based simulation engine for automated vehicle validation powered with technologies from NVIDIA and Microsoft. The Cognata platform uses artificial intelligence, deep learning, and computer vision to provide the only solution capable of validating automated vehicles with unlimited scalability. The NVIDIA DRIVE platform supports automated applications with software that helps developers and researchers optimize, validate, and implement their work. Cognata runs the combined technologies on the Microsoft Azure cloud-based platform. The cooperative solution represents a major milestone for the automated vehicle industry and promises to bring safer self-driving vehicles to market much faster than anticipated. NVIDIA DRIVE is a cloud-to-car scalable AI platform that enables automated vehicles to recognize their environment and drive safely. By training and testing deep neural networks in the datacenter, and then executing real-time, low-latency inferencing in the vehicle, it enables Level 2 through Level 5 driving systems. The association with Microsoft's Azure cloud platform helps Cognata bring the power of scaled cloud computing infrastructure to the automotive world, along with NVIDIA's cloud-based GPU technology [9]. Car makers are required to gather 10 billion miles worth of test drives, which can take years to complete. Cognata's simulation platform can advance that process by enabling automated vehicle manufacturers to log extremely realistic virtual test drives on the virtual roadways of its simulated environment, thereby cutting years off the road-testing process. Further, the Cognata simulation engine cloud-based services allows auto makers to avoid the expense of significant upgrades to their infrastructure that the validation process would otherwise require [10].

Fig. 3.9: COGNATA's simulated city (left) and 3D map (right) (Source: [10])

G. CARSIM

CarSim delivers an accurate, detailed, and efficient method for simulating the performance of passenger vehicles and light-duty trucks. It has twenty years of real-world validation by automotive engineers and is universally preferred for analyzing vehicle dynamics, developing active controllers, calculating a car's performance characteristics, and engineering next-generation active safety systems.

CarSim, TruckSim, and BikeSim are used globally by over 110 OEMs and Tier 1 suppliers and over 200 universities and government research labs. (There are more than 1500 active CarSim licences, not

including driving simulators or students). CarSim is used widely by 7 of the 10 largest automotive OEMs. OEM users reliably find close agreement between CarSim predictions and test results. It works as a standalone application; it does not require any other software to carry out simulations and has a standard interface to MATLAB/Simulink. It allows users to build complicated scenarios and test event sequences.

CarSim supports:

Software-in-the-loop

Model-in-the-loop

Hardware-in-the-loop

Driver-in-the-loop

CarSim supports vehicle sensors and interactive traffic for V2V and ADAS development. It includes example datasets for 20 vehicle types (with about 80 variants in total), used in about 200 example simulations. It is also cheap in comparison to other commercial vehicle dynamics software tools.

High Fidelity Vehicle Models: CarSim includes detailed math models for all combinations of vehicles with independent, solid axle, and twist beam suspensions. Optional features in the math models add degrees of freedom to handle one and two axle trailers, flexible chassis frames, and flexible powertrain mounts.

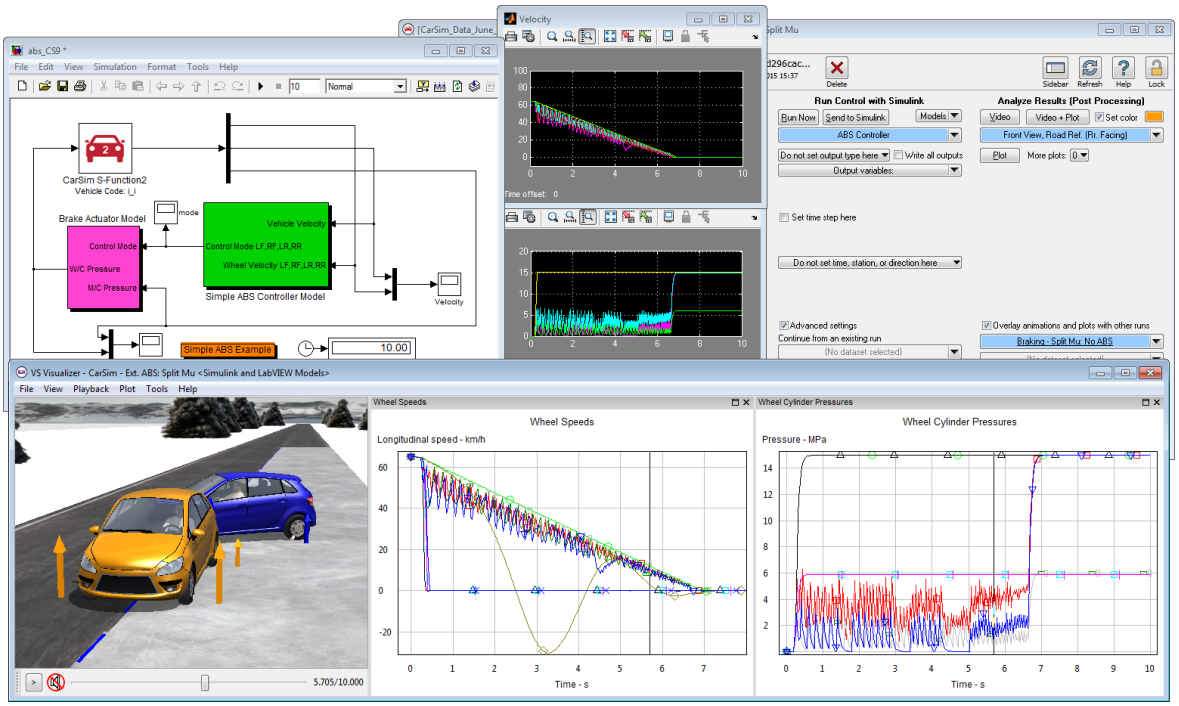


Fig. 3.10: Simulation plots in CarSim and co-simulation with MATLAB/Simulink® (Source: [11])

The features of CarSim include:

Modular Vehicle Definition: automobile sub-systems are described with parameters and tables that can be obtained from published data, engineering tools, and test rigs. When data is not available, you can use the real-world data provided in our sample datasets. CarSim's modular, parameter-based design approach lets you modify parameters and run simulations any time during the design cycle.

Automobile Performance Metrics: CarSim provides open-loop and closed loop driver models with superior features to help engineers quickly discover a vehicle's limit capabilities or its optimal path through a complex maneuver. These technologies are demanded by manufactures who must certify compliance with worldwide ISO and ECE stability control regulations.

Integrate technologies using standard design tools: Mechanical Simulation provides uniform interfaces to other standard simulation and design tools such as Simulink and LabVIEW. Advanced users can develop standalone technologies using Visual Studio and CarSim's API.

VS Commands: this strong scripting language provides tools to automatically control test runs, extend the vehicle model, control complex driving maneuvers, and model auxiliary sensors [11].

H. APOLLO

Apollo provides an open, dependable and secure software platform for its partners to develop their own automated driving systems through on-vehicle and hardware platforms.

Apollo has a leading HD map service, the only open Automated Driving simulation engine, End-to-End, a deep learning algorithm. Apollo accelerates the development, testing, and deployment of Automated Vehicles. As participation grows, more amassed data becomes available. Compared to a closed ecosystem, Apollo can evolve quicker, bring greater benefits to members, and continually grow [12][14].

The components of Apollo are:

Accurate Perception: Different sensors, such as LiDAR, cameras and radar collect environmental data surrounding the vehicle. Using sensor fusion technology perception algorithms can decide in real time the type, location, velocity and orientation of objects on the road.

This automated perception system is backed by both Baidu's big data and deep learning technologies, as well as a large collection of real-world labeled driving data. The large-scale deep-learning platform and GPU clusters drastically.

Simulation: Simulation provides the ability to virtually drive millions of kilometers daily using an collection of real-world traffic and automated driving data. Through the simulation service, partners gain access to many automated driving scenes to quickly test, validate, and optimize models with thorough coverage in a way that is safe and efficient.

Fig. 3.11: Obstacle detection and avoidance in Apollo simulation (Source: [14])

HD Map and Localization: Baidu established the extensive application of deep learning and artificial intelligence technology to map creation and is one of the few Chinese firms capable of producing HD mapping data on a large scale.

The localization system is a comprehensive positioning solution with centimeter level accuracy based on GPS, IMU, HD map, and a variety of sensor inputs. Developers can reduce costs and adjust precision using varied usage scenarios.

End to End: End-to-End automated solutions are appealing because of the low cost and low engineering complexity. By using real road data, the horizontal and latitude driving models are based entirely on deep learning. This allows for the quick and efficient application onto automated test vehicles. Currently, horizontal, and latitude model source code with 10,000 km of data is accessible on Apollo.

Planning: Apollo vehicles are equipped with a planning system containing prediction, behavior, and motion logic. The planning system adapts to real time traffic conditions, resulting in accurate trajectories that are both safe and comfortable. Currently, the planning system functions on a fixed route in both night/day conditions.

Intelligent Control: The Apollo intelligent vehicle control and CAN bus-proxy modules are accurate, broadly applicable and adaptive to different environments. The modules can handle different road conditions, speeds, vehicle types and CAN bus protocols. Apollo provides waypoint following capability with a control accuracy of ~10 cm.

Open Data Platform: By opening the automated driving source code, capabilities, and data, Apollo forms a thorough “vehicle and cloud” open ecosystem. Apollo offers developers and partners lacking data and computing power an array of fast and flexible services. Through this, Apollo is building an open automated driving ecosystem that empowers each participant and broadens the widespread adoption of automated driving.

Fig. 3.12: Automatic Grading System to detect speed limit, traffic light, etc. in Apollo (Source: [14])

Reference Hardware: Apollo provides partners with a broad hardware reference design, including vehicle selection, key hardware components, peripherals, and a multifaceted hardware installation guide.

MAP Engine: The MAP Engine manages and protects the HD-Map data, as well as provides a combined data query interface. The MAP Engine provides a modular, hierarchical, cross platform, and a very customizable programming interface.

DuerOS for Apollo: Based on DuerOS, the world’s first human-vehicle devoted AI system, it provides solutions including human-vehicle dialogue, facial log-in, fatigue driving monitoring, AR navigation, intelligent security, vehicle to home (V2H) and personalized services & contents.

Security: Apollo provides the 4S solution – Scan, Shield, See, and Save. This covers the full life-cycle of a vehicle's cyber security requirements. For Shield, Apollo's security products are currently implemented in mass production vehicles, including IDPS, Car FireWall, Secured OTA Kits, in order to protect user privacy and vehicle information from network security breaches [13].

I. AUTOWARE

Autoware is ROS-based open-source software, enabling self-driving mobility to be deployed in open city areas [16]. Localization is achieved by 3D maps and SLAM algorithms in combination with GNSS and IMUsensors. Detection uses cameras and LiDARs with sensor fusion algorithms and deep neural networks. Prediction and Planning are based on probabilistic robotics and rule-based systems, partly using deep neural networks as well. The output of Autoware to the vehicle is a twist of velocity and angular velocity (also curvature). This is a part of Control, though the major part of Control is supposed to reside in the by-wire controller of the vehicle, where PID and MPCalgorithms are often adopted [12][15].

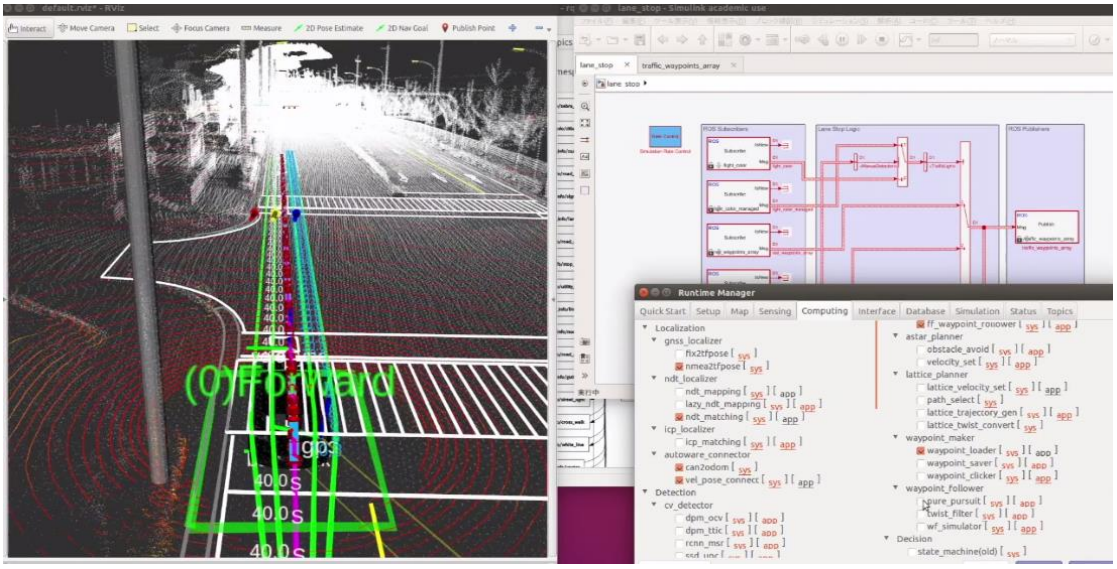


Fig. 3.13: Intersection model and control in Autoware using MATLAB® (Source: [15])

J. EB ROBINOS

EB Robinos is a comprehensive, hardware agnostic software solution for highly automated driving systems. Control and manage the increasing complexity of HAD systems and bring them to market quickly. Highly automated driving (HAD) means many pieces of software and hardware need to work together seamlessly, efficiently, and safely. HAD requires creating, coordinating and combining sensors, actuators and components—with all the different parties involved (carmakers, suppliers etc.). The complexity of HAD systems is a huge challenge to the industry [12].

EB Robinos is a software framework for automated driving that helps for overcome this challenge by controlling and managing the complexity professionally. EB Robinos is an application-layer functional software architecture with open interfaces and software modules. Its holistic approach provides control and management of complex HAD systems to accelerate, enhance and optimize automated driving system development.

The functional architecture and the open interfaces enable cooperation among different parties and pave the way for future mobility. EB robinos is available for development, embedded prototyping, and for series use. With EB robinos, carmakers and Tier 1 suppliers can shift their focus from architecture effort to differentiation: bringing cars to market quickly and delighting consumers with distinctive, highly automated driving features that provide competitive advantages and strong market positions.

Fig. 3.14: Simulation of lane-change maneuver in EB Robinos (Source: [17])

EB Robinos defines an application-layer architecture for ADAS up to SAE level 5 of automated driving and runs on Embedded Linux systems or integrates into AUTOSAR ECUs. It can be integrated into central ADAS ECU as well as into distributed systems of several ADAS ECU. It can incorporate existing or new, customer or third-party subsystems. EB Robinos Is available for development and prototyping in your environment or within EB Assist ADTF. It is an integrated approach – available from development and embedded prototyping through series production [17].

K. NVIDIA DRIVEWORKS

Advances in automotive technology will continue to drive more computing power, more sensors, more functionality, more driver's assistance and more autonomy into cars. NVIDIA® DriveWorks is a Software Development Kit (SDK) that contains reference applications, tools and library modules. It also includes a run-time pipeline framework that goes from detection to localization to planning to visualization. It is designed to be educational to use and open, so you can enhance it with your own code [12].

NVIDIA DriveWorks is available as part of the NVIDIA DRIVE Software provided to select Automakers, Tier 1 Suppliers, and Research Institutions working on developing systems that enable cars to drive themselves. Developers working on automated driving will find the DriveWorks SDK helpful in accelerating their development on NVIDIA DRIVE PX platforms.

The key features include:

- Detection libraries include sensor processing, sensor correlation, sensor fusion, segmentation and deep learning detection & classification
- Localization libraries include map localization, HD-Map interfacing, and ego motion (structure from motion and odometry)
- Planning libraries include vehicle control, scene understanding and path planning problem solvers
- Visualization libraries include streaming to a cluster display, ADAS rendering, and debug rendering
- Tools that include sensor calibration, rig calibration, camera intrinsic & extrinsic and lidar extrinsics
- NVIDIA DriveWorks is architected to be modular, educational, optimized and open [18].

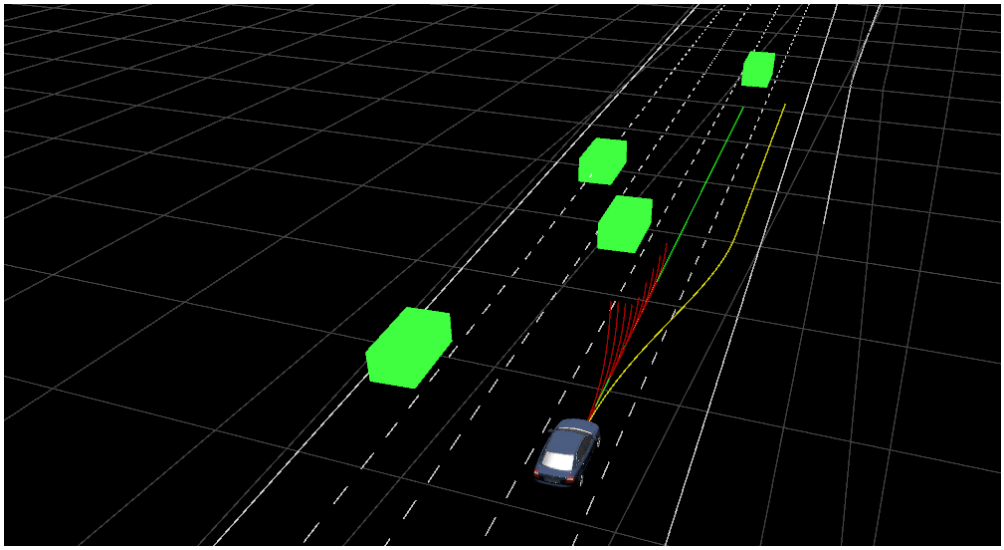


Fig. 3.15: Path planning problem solving using DriveWorks (Source: [18])

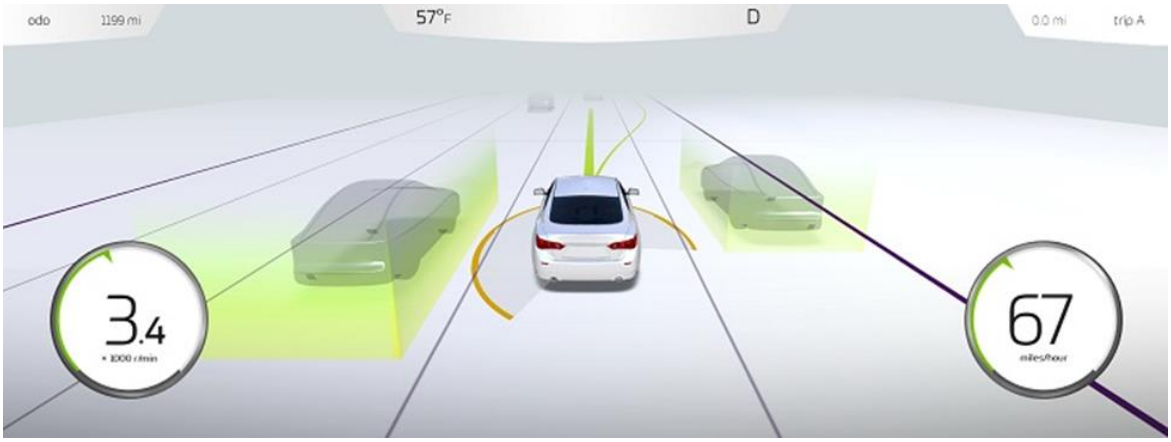


Fig. 3.16: Sensor and rig calibration using DriveWorks (Source: [18])

L. CESIUM

Cesium is a horizontal 3D map platform and has been used across verticals including drones, building information modeling, etc. From massive mixed dataset streaming to high-precision rendering to time-dynamic simulation, Cesium has built up a feature set and developer ecosystem that forms a flexible foundation for the unique automated driving challenges outlined below.

1. Massive amounts of data

An automated vehicle can easily gather 4 TB of data per day, including 40 MB of camera frames per second, 70 MB of LIDAR points per second, and time-varying telemetry from GPS, gyroscopes, and

accelerometers. Real-time visualization and log playback of this data requires efficient streaming of time-dynamic data. Cesium's open time-dynamic format, CZML, is designed for this, including batching several frames into windows like streaming video, originally created for aerospace use cases.

Recreating a virtual environment from the collected photogrammetry (from cameras) and LIDAR data requires the ability to stream massive heterogeneous 3D geospatial datasets. Cesium's open 3D Tiles is designed for exactly this—streaming only the part of the 3D model needed for the current virtual view and keeping the rest of the data in the cloud.

Given the massive amount of telemetry and environment data collected, users do not want to have to download an entire log database to play back an automated vehicle road test or, even worse, imagine doing this for a fleet of 20 vehicles at 4 TB each. Since Cesium is 100% web-based, it runs in a browser on any device using WebGL, from phones to tablets to laptops, and streams from the cloud only the data currently needed for the simulation.

2. Flexibility and interoperability

Cesium is uniquely focused on visualization, which allows others in the automated driving industry to focus on AI and other areas and easily leverage Cesium for visualization. Car manufacturers use a combination of software and hardware that is developed in-house and by third-parties. All these components must interoperate with each other via well-defined—and ideally open—APIs and formats.

CesiumJS has a very adaptable API with more than 450 top-level entry points that let developers control every detail from the brightness of the map to the orientation of a vehicle to the shade of the atmosphere. Since CesiumJS is open-source, developers have access to modify any part of the code and even contribute back code as more than 100 developers have already done. The formats Cesium uses are open, widely supported, and optimized for 3D runtime. The Cesium team is the creator of 3D Tiles, a candidate OGC Community Standard, and co-creators of glTF, an open Khronos standard used by Google, Microsoft, Facebook, and others. These formats form the foundation for streaming individual 3D models, such as cars, as well as massive environments, such as point clouds and photogrammetry, on the web. The commercial Cesium ion content pipeline provides 3D tiling for point clouds, photogrammetry, 3D buildings, and other massive environments using 3D Tiles allowing developers to mix and match interoperable components.

3. High precision rendering

Automated driving requires accuracy—ideally sub-cm accuracy—for the safest and most precise measurements. Originally built for aerospace, Cesium was designed ground-up for high-precision rendering both for individual vertex position transforms and for massive view distances. For example,

Cesium has been used to track every satellite in space and for sub-cm 6.4 billion-point point clouds. With Cesium you could see the view from inside a car to a distant mountain to the International Space Station.

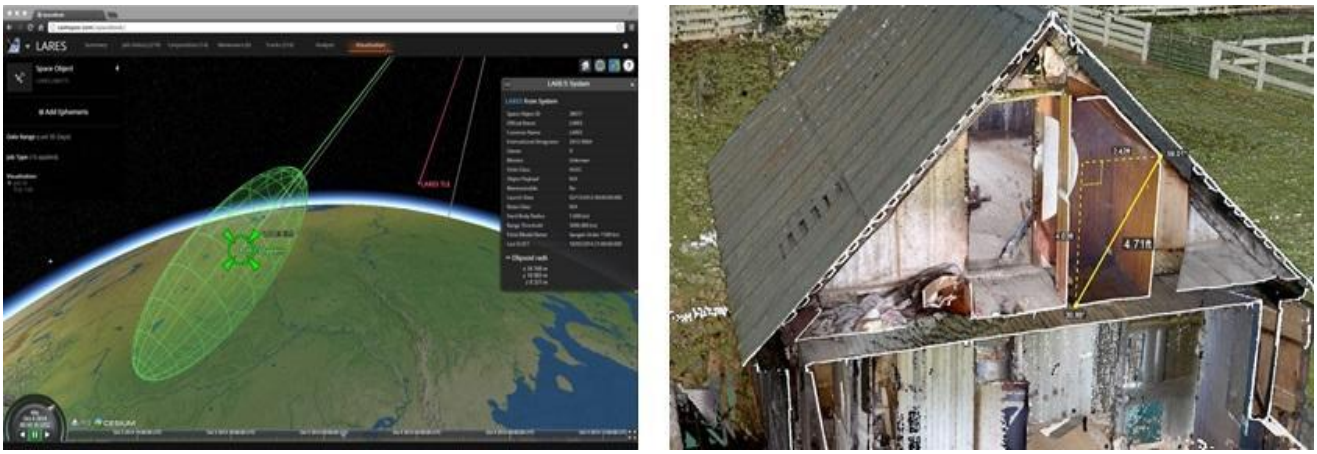


Fig. 3.17: High level of detailing represented in Cesium (Source: [21])

4. Geospatial data fusion

One automated vehicle may only capture camera and LIDAR data for a subset of the environment, whereas a simulation or even log playback may need to show a more complete environment.

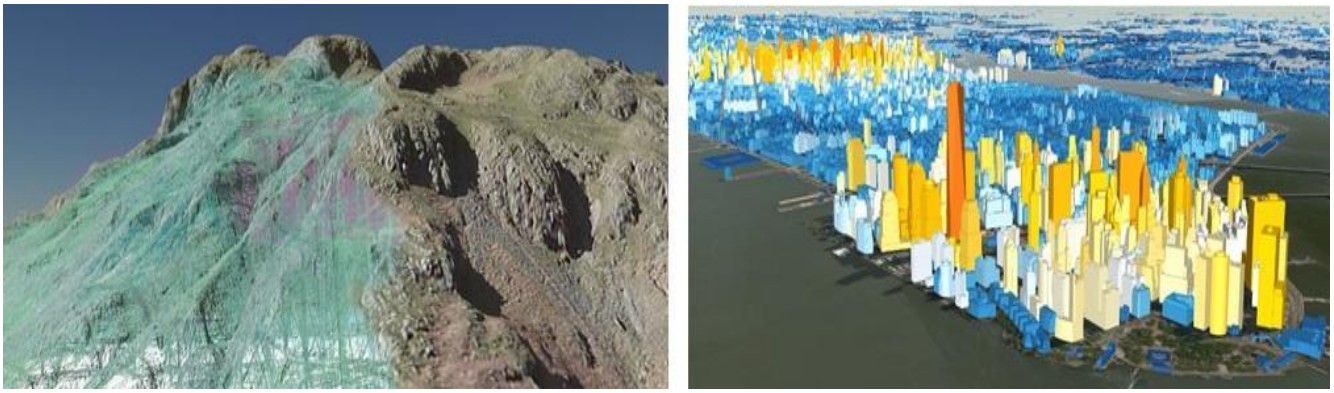


Fig. 3.18: Terrain and 3D buildings from geospatial data sources (Source: [21])

With Cesium, you can fuse data collected from multiple automated vehicles into one virtual environment and augment that with geospatial data such as terrain, imagery, 3D buildings, and vector data. These data may be captured by drones, aircrafts, satellites, surveys, or provided as open data from governments.

5. Simulating vehicle and sensor performance

Adding a new sensor or reconfiguring a sensor array can significantly impact what a vehicle can “see.” Likewise, different traffic and pedestrian patterns and weather and lighting conditions further impact visibility. Simulation provides an efficient means of determining this visibility for a given route and environment.

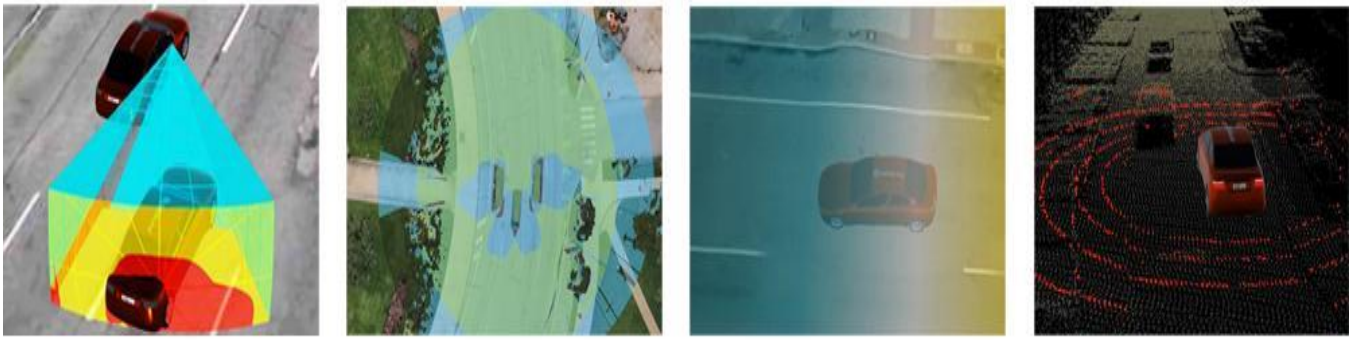


Fig. 3.19: Visibility analytics in Cesium (Source: [21])

The commercial Cesium ion SDK provides GPU-accelerated line of sight, viewshed, and visibility analytics to efficiently and easily understand the impact of sensor visibility [21].

M. rfPRO

There are now road car models on sale today, from Top 10 OEMs who were early adopters of rFpro technology, whose testing began in rFpro in order to reduce costs and time delays in the engineering process. rFpro provides driving simulation software, and 3D content, for Deep Learning Automated Driving, ADAS and Vehicle Dynamics testing and validation.

rFpro is being used to train, test and validate Deep Learning systems for ADAS and Automated applications. rFpro’s weather and physically modeled atmosphere, delivering real-time reflections, shadows and lighting mean rFpro can save years from ADAS, sensor and Automated development projects. Hundreds of kilometers of digital models of public roads are available off-the-shelf, from rFpro, spanning North America, Asia and Europe, including multi-lane highways, urban, rural, mountain routes, all copied faithfully from the real world.

Fig. 3.20: Recreation of a street in Paris in an rFpro simulation (Source: [23])

The unique feature of rFpro, compared to traditional driving simulators, is that it allows driving simulation to be used to test the vehicle dynamics of road vehicles. By delivering a high-resolution road surface in real time, while generating accurate realistic graphics without lag, professional test drivers may contribute to the engineering process while the car design is still model based.

This means that an rFpro driving simulator, or engineer's Workstation can allow one to conduct testing and calibration with professional test drivers and engineers without the delays and cost associated with waiting for the testing of real cars or prototypes.

By testing earlier in the engineering process on an rFpro driving simulator, the changes identified through test driving can be made while the cost of implementing those changes is still relatively low; before prototypes are required and before production. The cost savings can be substantial. rFpro is being used to test and calibrate passive chassis designs, steering systems, chassis control systems, ADAS and Automated sensor models, algorithms and control systems, drivetrain control systems, traction control, stability control, torque vectoring and engine control systems [22].

Fig. 3.21: Co-simulation of rFpro with Simulink® blockset (Source: [22])

rFpro provides both a Simulink® blockset library and an API for customers to write their own function blocks for communicating between their Simulink model and rFpro's external applications (e.g. human-in-the-loop rFpro PHYSICS around Simulink plant models, Simulink sensor models and detection algorithms inside rFpro ADAS_WORKSTATION). A Simulink blockset provides complete real-time control over environmental, atmospheric, weather, and lighting conditions so that tests and deep learning may span DOE runs over multiple permutations and combinations of conditions affecting cameras, camera sensors, and algorithms.

Customers can use rFpro to communicate with remote Simulink models by connecting to rFpro's remote blocksets, which implement UDP over Ethernet. These blocksets have been developed to work from both the MATLAB® and Simulink desktop environment and from real-time platforms where Simulink models are compiled to run (e.g. dSPACE, Speedgoat, Concurrent). rFpro provides example integration Simulink models for customers who run CarSim for Simulink and CarMaker for Simulink models as well as those who run real-time Simulink models that encompass Dymola compiled blocksets, C-code chassis vehicle models, and purely Simulink block-built models. rFpro also offers real-time sensor feeds, with an example Simulink project for customers to try out with MATLAB® Coder™ and Simulink® Coder™. The compiled DLL files can be hosted by rFpro, allowing customers to develop their own models using MATLAB and Simulink to run rFpro's photo-realistic 3D environments [23].

N. PRESCAN

PreScan is a simulation platform consisting of a GUI-based preprocessor to define scenarios and a run-time environment to execute them. The engineer's prime interface for making and testing algorithms includes MATLAB and Simulink. PreScan can be used from model-based controller design (MIL) to real-time tests with software-in-the-loop (SIL) and hardware-in-the-loop (HIL) systems. PreScan can operate in open-loop & closed-loop, and offline & online mode. It is an open software platform which has flexible interfaces to link to 3rd party vehicle dynamics model (e.g. CarSIM, dSPACE ASM, etc.) and 3rd party HIL simulators/hardware (e.g. ETAS, dSPACE, Vector).

PreScan comprises several modules that together provide everything an ADAS system developer needs. An intuitive graphical user interface (GUI) allows you to build your scenario and model your sensors, while the MATLAB/Simulink interface enables you to add a control system. This interface can also be used to import existing MATLAB/Simulink models such as vehicle dynamics models. When running your experiment, the visualization viewer gives a realistic 3D representation of the scenario. Optionally, tools such as dSPACE ControlDesk and NI LabVIEW can be used for activities such as ‘data acquisition’ and ‘test automation’.

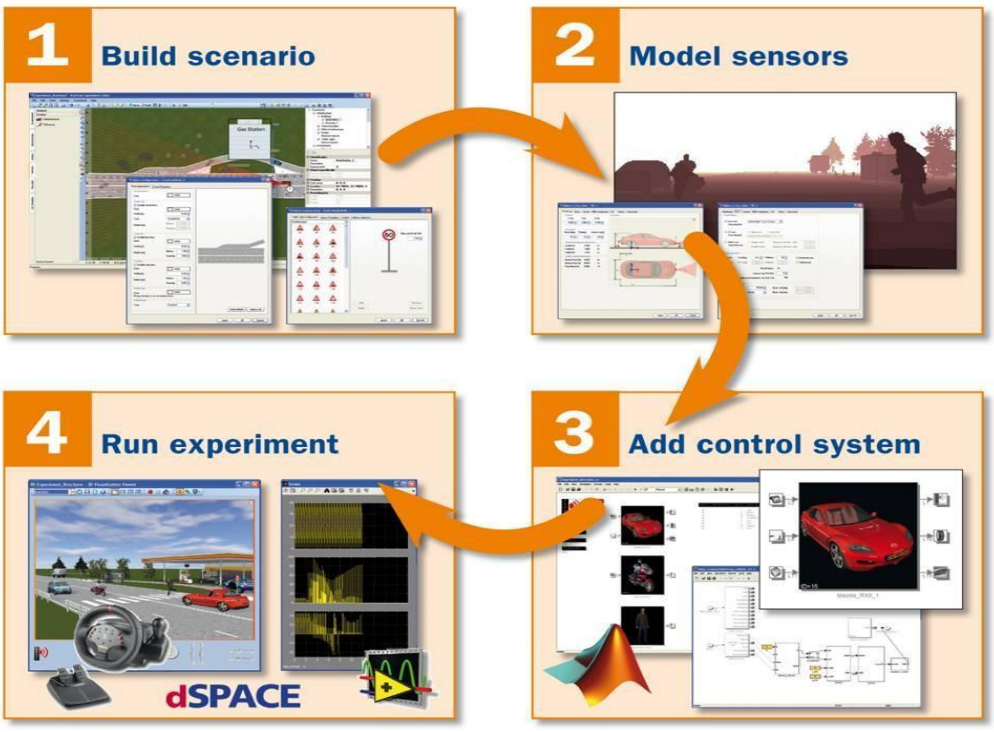


Fig. 3.22: Simulation flow/control flow diagram in PreScan using dSPACE (Source: [24])

The program works using four easy steps:

1. Build scenario
   A dedicated pre-processor (GUI) allows users to build and modify traffic scenarios within minutes using a database of road sections, infrastructure components (trees, buildings, traffic signs), actors (cars, trucks, bikes and pedestrians), weather conditions (such as rain, snow and fog) and light sources (such as the sun, headlights and lampposts). Representations of real roads can be quickly made by reading in information from OpenStreetMap, Google Earth, Google 3D Warehouse and/or a GPS navigation device.
2. Model sensors
   Vehicle models can be equipped with different sensor types, including radar, laser, camera, ultrasonic, infrared, GPS and antennas for vehicle-to-X (V2X) communication. Sensor design and benchmarking is facilitated by easy exchange and modification of sensor type and sensor characteristics.
3. Add control system
   A MATLAB/Simulink interface enables users to design and verify algorithms for data processing, sensor fusion, decision making and control as well as the re-use of existing Simulink models such as vehicle dynamics models from CarSim, Dyna4 or ASM.
4. Run experiment
   A 3D visualization viewer allows users to analyze the results of the experiment. It provides multiple viewpoints, intuitive navigation controls, and picture and movie generation capabilities. Also, interfaces with ControlDesk and LabView can be used to automatically run an experiment batch of scenarios as well as to run hardware-in-the-loop (HIL) simulations [24].

O. APPLIED INTUITION

Applied Intuition provides simulation tools for all parts of an AV stack including sensors, prediction, planning/drive policy, and controls. The team also develops sophisticated, intelligent behavior models for real-world traffic and pedestrians. All this functionality can be applied at scale for thousands of simulations to test the software thoroughly.

Their products are being used by engineers both inside and outside Silicon Valley. They are actively developed by a team with years of collective experience and the simulator is designed from the ground up for advanced autonomy. It streamlines in-vehicle driving experience and analysis of previously collected drive data. This makes it easy to identify edge cases in recorded drives, so one can play back and even edit critical moments from all the vehicle data. This means that as the research fleet grows, simulation capabilities grow too [29].

The key features of this software are:

Easy integration

- The clean and well documented interfaces are inspired by developer-first products like Stripe and GitHub. The team handles almost all the integration work.

Modular

- Every AV team's development process is different, and the product finds tailored solutions that meet specific needs. Applied Intuition follows global automotive standards [25].

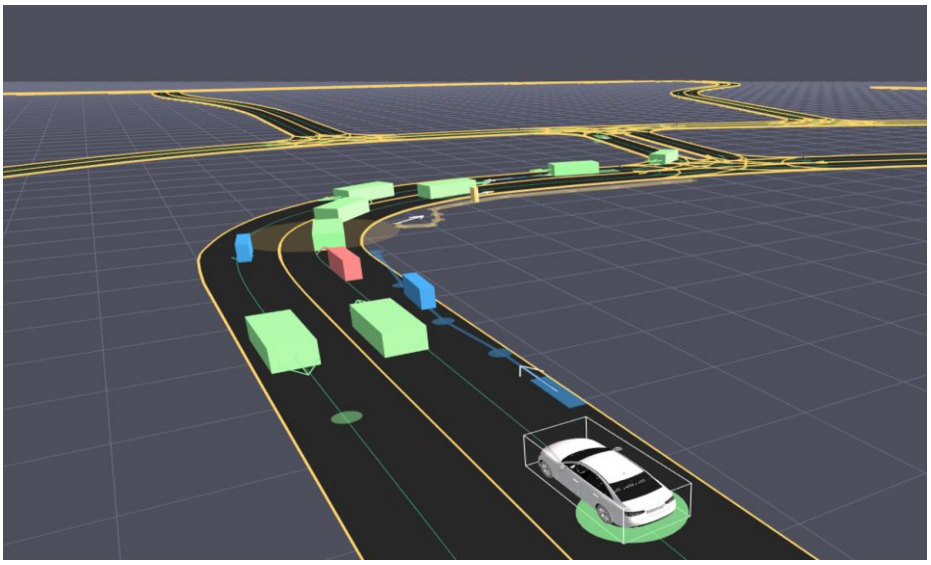


Fig. 3.23: Applied Intuition simulation of automated vehicle interacting with cars and bicycles (Source: [25])

CHAPTER 4 LIMITATIONS OF AUTOMATED DRIVING SIMULATORS

The central problem of the simulator lies in how realistic we can simulate the actual driving environment. No matter how accurate and powerful the simulator is, the artificial simulation of the scene and the real scene still have some differences. There are still many unanticipated events in the real scene that cannot be simulated in a simulator. Therefore, if one can use the real traffic data to reproduce the real scene, the test results will be better compared to the artificial simulation of the scene. However, the major problem of replaying real-world data is the computing power essential to process the massive amount of real-world data. To reproduce the scene of every segment of the real world on the simulator, the automated vehicles must gather the information of each section of the road. This amount of information cannot be handled on single machines. Furthermore, each scene can be broken down down into basic fragments and the combinations rearranged so that these fragments generate more test cases. However, this would only generate even more data and add more load to the already stressed simulation platform.

The simulation platforms being used in the industry has certain failings including real-time feedback and latency issues, as well as inaccurate vehicle, tire and road models. These will lead to inaccurate and optimistic results which can affect the safety of the passengers and pedestrians in real life situations. The lack of motion simulators in these platforms are also a drawback as motion cues can have a significant effect on the models and the automated driving systems.

Perhaps the most noticeable challenge in a fully automated vehicle is that the whole point is for the driver to no longer be driving the vehicle. That means that the driver can no longer be considered to provide control inputs to the vehicle during operation.

In order to assess the response of the automated vehicle in a risky situation, it is necessary to virtually represent a wide number of elements:

- Various obstacles, such as pedestrians or other vehicles
- The road, its state, the rain-drenched tarmac
- The vehicle, its behavior, mechanical characteristics, state and location
- The environment, the level of visibility, weather conditions and various road signs
- The driver, his or her attention, glance and fatigue

The two main weaknesses of the simulation platforms are a lack of emergency scenarios, and potential significances of differences between real and simulated data.

Emergency situations are still hard to simulate as each real-world accident is distinctive. Although vehicles can absorb general driving operations through simulation, it is impossible to predict every single emergency. For example, the May 7th, 2016 fatal accident involving a Tesla on autopilot occurred because the system failed to differentiate a white truck against a bright sky. Although this is not a rare scenario in the real world, the simulator could not cover it.

This differences between real and simulated data is another issue that could negatively affect system performance, as the full effects of these differences remain unclear. Engineers are struggling to determine what type of data leads to an accident due to unexplainable features of the algorithms. For example, real world pedestrians with unusual clothing and posture abnormalities cannot all be reflected in the simulator, and this might reduce a self-driving vehicle's ability to identify pedestrians.

In order to overcome these disadvantages, it is important to simulate more scenarios to make anomalous data traceable. For example, SynCity is a Dutch company developing an advanced simulator that aims to present a very wide range of scenarios, including other vehicle misconduct and various emergency situations, in order to optimize algorithms [27].

CHAPTER 5 CONCLUSIONS

The challenges in developing a suitable automated driving simulation platform is significant. This paper has reviewed some of the most prominent and powerful software available in the market today. The main objective of this study is to compare the simulation platforms based on their different characteristics. Almost all the simulation platforms have certain things that are common to them such as the simulation engine upon which it is based, the logic flow and the control models for vehicle and their interaction with the environment [32].

Based on the study, it has been concluded that the open-source software such as Cognata, Metamoto, CARLA and so forth are better in terms of fidelity and accuracy. Such platforms are developed by startup companies which only make one type of product. Hence, the companies are bound to spend more time and effort tuning the software and making it as perfect as possible. This can be observed as soon as one opens the software and see the renderings of the models and the environment generated for simulation. Software such as Cognata have renderings which are almost life-like when compared to the wireframe boxes and the basic detailing of the roads in Carcraft used by Waymo for the simulation of their automated vehicles. Platforms like Cesium also has renderings and models which are of high precision which allow for the results to be more accurate rather than optimistic. The detailing of the road, tire models, etc. are crucial to how well the simulation results turn out.

Products such as PreScan, CarSim, Autoware, rFpro, and others allow engineers to simulate sensors, conditions, mechanical builds, and so forth. Each of them has their own advantages but they miss in areas crucial to simulation for automated vehicles, including an oversimplification of existing sensor outputs, as well as a lack of complex understanding of how environments impact the vehicle models.

Some of the platforms studied here have cloud capabilities which can be useful. Cloud-based services allows automotive manufacturers to avoid the expense of substantial upgrades to their infrastructure that the validation process would otherwise require. Cognata, Apollo and Cesium have advanced cloud services where all the simulation data is stored in cloud including vehicle data, the maps of the environment and other information. Only that information which are critical to the specific simulation is extracted from the cloud. This saves a lot of space and reduces the computing power required to carry out that simulation.

Most of the simulation software that have emerged in the past decade all have powerful processing and rendering capabilities. Therefore, it is difficult to compare the platforms based on these criteria. Also,

every software is unique in the way it runs the simulation although some ways such as SLAM algorithms used in Autoware are better than others. Platforms like Carcraft program the pedestrians, bicyclists, trees, etc. into the simulation predefining its position in the environment so that the sensors do not have to be configured in order to detect them. This reduces the computing power and processing time required for simulation although this method makes the behavior more unrealistic compared to the real world.

Apollo has one of the best and most detailed HD maps used for simulation. It also provides the only open-source automated driving simulation engine. EB robinos is a platform that can accommodate changes more easily. It can be modified and developed according to the user's needs which is want most automotive OEMs and Tier 1 suppliers want. NVIDIA Driveworks is intended to be more for educational use and can be enhanced with the user's own code. This software is based on NVIDIA Drive which is being used at Arizona State University as part of human factors in driving study. Metamoto has no requirement to be integrated into Simulink and can work on any OS. It also has test scripting and post test analysis integrated into the software. The time to run 1000 simulations is also low of about 5 minutes.

One cannot argue that open-source software is better than in-house software developed by automotive OEMs but ranking these simulators is sometimes not very useful as one software may be more suited to an automated vehicle development by a specific OEM while another software may be more suited to another OEM. Overall, COGNATA and Apollo seems to have the best overall package followed by CARLA, Metamoto and rFpro. ANSYS and NVIDIA have powerful simulation packages and are more robust and reliable owing to the big companies that have developed the software. Carcraft is tailored to Waymo's needs and thus is the most efficient for Waymo's self-driving vehicles.

REFERENCES

1. “The History of Flight Simulators.” SimulatorflÃ¼ge Im Full Flight Flugsimulator Von Lufthansa Aviation Training, www.proflight.com/en/full-flight-simulatoren/historie.php.
2. “A Behavioral Multi-Agent Model for Road Traffic Simulation.” NeuroImage, Academic Press, 23 May 2008, www.sciencedirect.com/science/article/abs/pii/S0952197608000456.
3. Madrigal, Alexis C. “Inside Waymo's Secret World for Training Self-Driving Cars.” *The Atlantic*, Atlantic Media Company, 23 Aug. 2017, www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/.
4. “Metamoto Launches Simulation-as-a-Service Offering.” Smart Cities World, www.smartcitiesworld.net/news/news/metamoto-launches-simulation-as-a-service-offering-3081.
5. arXiv, Emerging Technology from the. “Need a Driving Instructor for Your Automated Vehicle? Try CARLA.” *MIT Technology Review*, MIT Technology Review, 16 Nov. 2017, www.technologyreview.com/s/609503/the-open-source-driving-simulator-that-trains-automated-vehicles/.
6. “VRXPERIENCE Capabilities.” Simulation Capabilities | ANSYS Icepak, ANSYS Inc., www.ansys.com/products/systems/ansys-vrxperience/vrxperience-capabilities.
7. Andrew, et al. “AutonoVi: Automated Vehicle Planning with Dynamic Maneuvers and Traffic Constraints.” [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 29 Mar. 2017, arxiv.org/abs/1703.08561.
8. Korosec, Kirsten. “Audi Taps Israeli Startup Cognata to Accelerate AV Ambitions.” TechCrunch, TechCrunch, 26 June 2018, techcrunch.com/2018/06/26/audi-israeli-cognata-automated-vehicle/.
9. Cognata. “Cognata Builds Cloud-Based Autonomous Vehicle Simulation Platform with NVIDIA and Microsoft.” Medium.com, Medium, 9 Jan. 2018, medium.com/@cognataAI/cognata-builds-cloud-based-autonomous-vehicle-simulation-platform-with-nvidia-and-microsoft-accd46b160e6.
10. Bloomberg.com, Bloomberg, www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=433576158.
11. Mechanical Simulation Corporation, www.carsim.com/products/carsim/.
12. Schiavullo, Raffaele. “The 5 Most Used Open Source Software to Develop Self-Driving Platforms for ADAS (Advanced Driver Assistance System).” Twitter Tech News, Twitter Tech News - Http://Twittertechnews.com, 8 Oct. 2018, twittertechnews.com/software-reviews/the-5-most-used-open-source-software-to-develop-self-driving-platforms-for-adas-autoware-apollo-eb-robinos-eb-robinos-predictor-nvidia-driveworks-and-openpilot/.

13. “Apollo Auto.” GitHub, github.com/apolloauto.
14. Apollo, apollo.auto/devcenter/devcenter.html.
15. IV, Tier. “Autoware.” Autoware, autoware.ai/.
16. CPFL. “CPFL/Autoware.” GitHub, github.com/CPFL/Autoware/wiki.
17. “EB Robinos.” Elektrobit, www.elektrobit.com/products/eb-robinos/.
18. “NVIDIA DRIVE.” NVIDIA Developer, 5 Oct. 2018, developer.nvidia.com/driveworks.
19. Ai, Comma. “How Does Openpilot Work? – Comma Ai – Medium.” Medium.com, Medium, 26 June 2017, medium.com/@comma_ai/how-does-openpilot-work-c7076d4407b3.
20. Ai, Comma. “Our Road to Self-Driving Victory – Comma Ai – Medium.” Medium.com, Medium, 7 June 2017, medium.com/@comma_ai/our-road-to-self-driving-victory-603a9ed20204.
21. Cozzi, Patrick. “Patrick Cozzi.” Documentation, 3 Apr. 2018, cesium.com/blog/2018/04/03/cesium-for-automated-driving-simulation/.
22. “Driving Simulation | Deep Learning Automated Driving | Vehicle Dynamics.” RFpro, www.rfpro.com/.
23. “RFpro.” Reconstructing an Image from Projection Data - MATLAB & Simulink Example, www.mathworks.com/products/connections/product_detail/rfpro.html.
24. “PreScan.” TASS International, 8 Nov. 2018, tass.plm.automation.siemens.com/prescan.
25. Applied Intuition, Inc. “Applied Intuition | Products.” Applied, www.appliedintuition.com/products/.
26. Huang, Sam. “The Companies You've Never Heard of Training Your Automated Driving Car.” Medium.com, Medium, 23 Jan. 2018, medium.com/@thewordofsam/the-companies-youve-never-heard-of-training-your-automated-driving-car-61fda80e2ee5.
27. Synced. “Simulations Pave the Road for Self-Driving Technologies.” Medium.com, Medium, 8 Apr. 2018, medium.com/syncedreview/simulations-pave-the-road-for-self-driving-technologies-78b696227383.
28. Bouchner, Petr. Interactive Driving Simulators – History, Design and Their Utilization in Area of HMI Research. International Journal of Systems Applications, Engineering and Development, May 2016, www.naun.org/main/UPress/saed/2016/a482005-279.pdf.
29. Bloomberg.com, Bloomberg, www.bloomberg.com/news/articles/2018-09-12/how-self-driving-cars-can-get-past-the-learning-permit-stage-without-any-risk.
30. Koopman, Philip, and Michael Wagner. “Challenges in Automated Vehicle Testing and Validation.” SAE International Journal of Transportation Safety, vol. 4, no. 1, 2016, pp. 15–24., doi:10.4271/2016-01-0128.
31. Blana, Eva. A Survey of Driving Research Simulators around the World. Institute for Transport Studies, University of Leeds, 1996.

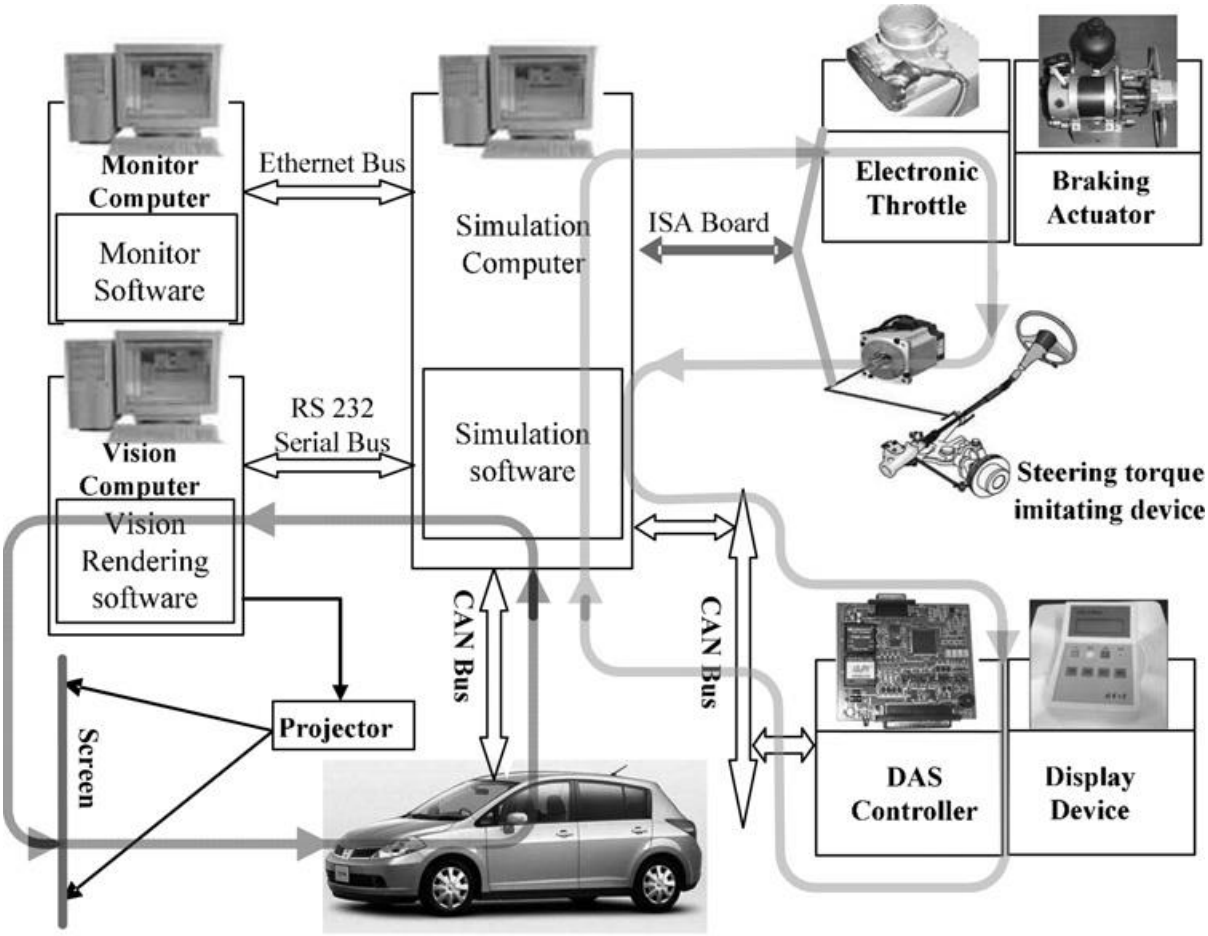
32. Metamoto. “How Legacy Automotive Simulation Tools Differ from Metamoto's Scalable Cloud-Based Simulation...” Medium.com, Medium, 30 Oct. 2018, medium.com/@metamoto/how-legacy-automotive-simulation-tools-differ-from-metamotos-scalable-cloud-based-simulation-c9a1e07b9c2c.
33. Team, CARLA. “CARLA.” CARLA Simulator, carla.org/.

Uncited Reference (Could not obtain due to high cost of report):

1. Lakrintis, Angelos. “2018: The Year of the Simulation for Autonomous Vehicles.” Strategy Analytics, 17 Sept. 2018, www.strategyanalytics.com/access-services/automotive/autonomous-vehicles/reports/report-detail/simulation-for-autonomous-vehicles.

APPENDIX A

SYSTEM CONFIGURATION OF DRIVING SIMULATION PLATFORM (FIGURE)



ISA-Industrial Standard Architecture bus, RS 232- Recommended Standard 232 bus, CAN- Control Area Network

Fig. 1: System configuration of driving simulation platform (Source: Jianqiang, Wang, et al. “A Driving Simulation Platform Applied to Develop Driver Assistance Systems.” 2009 IEEE Vehicle Power and Propulsion Conference, 2009, doi:10.1109/vppc.2009.5289755.)

APPENDIX B

AUTONOVI-SIM PERFORMANCE GRAPH

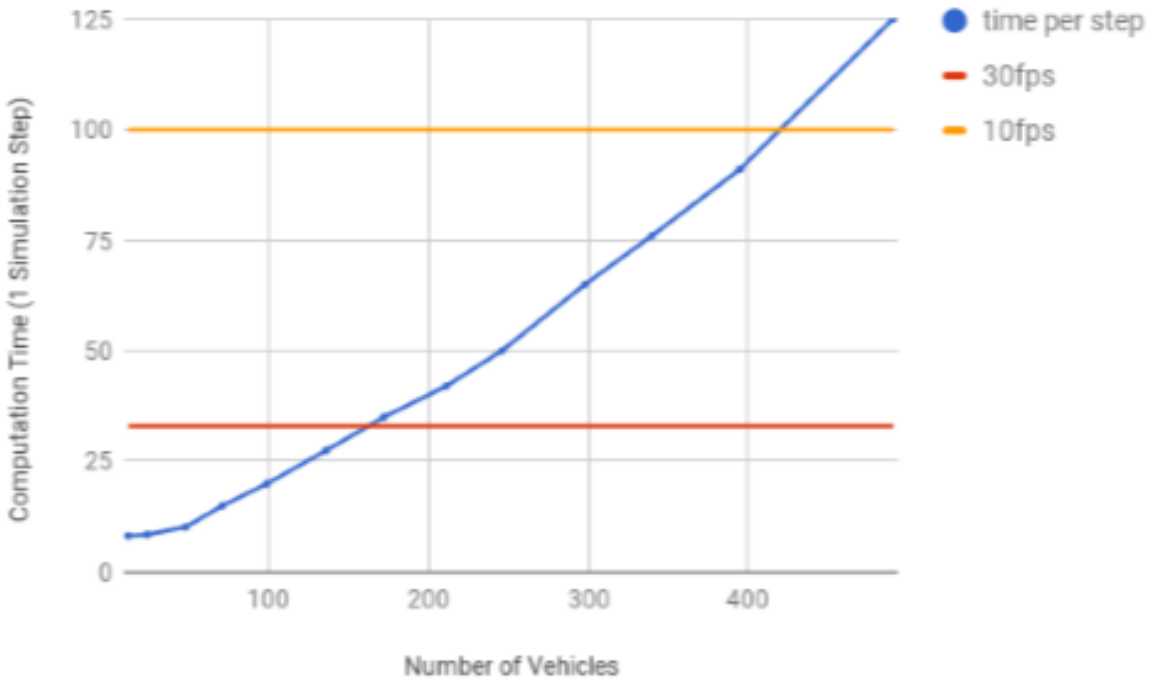


Fig. 2: AutonoVi-Sim Performance Graph

Repeated simulations were conducted on a traffic network, increasing the number of vehicles in each situation. This graph specifies the computation time for each simulation step as a function of the number of vehicles simulated. The boundary of 30 frames per second and 10 frames per second are shown for reference. In this scenario, each vehicle is provided with two basic ray-cast sensors with faultless accuracy. The computation time scales linearly in the number of vehicles simulated, with the ability to simulate 160 vehicles at 30 frames per second and up to 420 vehicles at 10 frames per second.

Source: Andrew, et al. “AutonoVi: Autonomous Vehicle Planning with Dynamic Maneuvers and Traffic Constraints.” [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 29 Mar. 2017, arxiv.org/abs/1703.08561.