UNMANNED VEHICLES: A SURVEY OF MODERN SIMULATORS

M. I. Makarov*, N. A. Korgin**, and A. A. Pyzh'yanov***

*,**,***Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
*Moscow Institute of Physics and Technology, Moscow, Russia

*✉ maxim.i.makarov@gmail.com, **✉ nkorgin@ipu.ru, ***✉ ipu@isko.moe

Abstract. This survey is devoted to popular simulators supporting rough terrain for unmanned vehicles, namely, Gazebo, CARLA, AirSim, NVIDIA Isaac Sim, and Webots. Their main capabilities related to terrain modeling, motion physics, and support for sensors and weather conditions are described. Particular attention is paid to the creation of realistic rough terrain scenes, the complexity of importing real maps, and interaction with other software platforms, such as Robot Operating System (ROS) and artificial intelligence (AI) systems. The main drawbacks of each simulator are analyzed: the labor intensity of creating detailed terrain and vehicle models, the high complexity of integrating real maps, and the dependence on powerful hardware. The survey also notes the complexity of interaction with various software solutions and the required knowledge of 3D modeling. Gazebo and Webots are remarkable for their good integration with ROS but require more effort to work with rough terrain. CARLA and AirSim provide high-quality visualization but have higher requirements for creating landscapes. NVIDIA Isaac Sim stands out for AI simulation support but is resource-intensive. The authors' experience in mapping vehicle trajectories and orienting in some simulators is presented.

Keywords: unmanned vehicles, simulators, rough terrain, tracked platforms.

INTRODUCTION

The development and testing of unmanned vehicles is one of the most difficult and urgent tasks of modern robotics and the automotive industry. To ensure the safety and reliability of autonomous systems, it is necessary to conduct large-scale tests in various conditions, including complex scenarios of interaction with the environment and other road users. However, conducting such tests in the real world leads to several challenges, including high costs, safety risks, and limited repeatability of experiments [1–3]. Simulators have long been an important tool in engineering and computer science. The first full-fledged vehicle simulators appeared as early as the 1990s and were intended to study particular aspects and characteristics of vehicle motion. With the development of computing power and modeling algorithms, simulators gradually became more sophisticated and realistic, allowing one to reproduce road conditions and interactions with other vehicles, pedestrians, and infrastructure.

In recent years, the development of *open-source* (OS) software communities has further contributed to the availability and flexibility of simulators. Platforms such as CARLA, Gazebo, and AirSim have become widespread due to their open-source nature and strong support from the development community. OS simulators allow researchers and engineers to customize and modify the simulation environment in accordance with the particular requirements of their projects, greatly accelerating the development and testing processes. Before the extensive evolution of OS simulators, some researchers used relevant computer games such as GTA [4].

This paper aims to review modern simulators used for the development and testing of unmanned vehicles, with a special focus on OS solutions supporting rough terrain simulation. We discuss different types of simulators and the fields of their application in the context of unmanned vehicles, particularly unique off-road transport platforms with electric drive of different layouts (wheel-tracked and ski-tracked, see Fig. 1). Significant datasets have already been collected to analyze the motion models of such mobile platforms within validating the elements of a conceptual distributed network of testing grounds for practicing the application scenarios of heterogeneous groups of electrically driven vehicles in difficult climatic and landscape conditions [5]. Based on these datasets, we plan to design autonomous motion control algorithms using

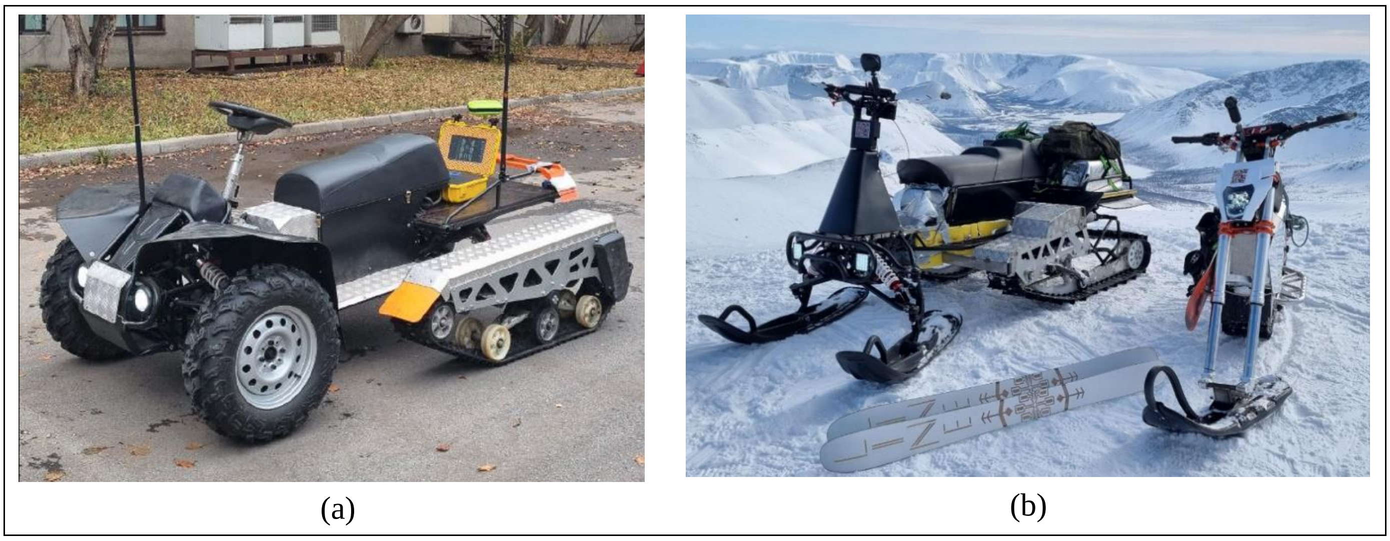

Fig. 1. Tracked platforms: (a) wheel-tracked and (b) ski-tracked.

the 2D path planning methods proposed in [6], with a suitable adaptation to rough terrain.

1. THE MAIN TYPES OF SIMULATORS

Modern autopilots are extremely complex systems consisting of many interconnected modules, each performing definite functions. These systems include modules for localization, obstacle detection and tracking, traffic flow analysis, path planning, and path following. Each module requires a separate approach to development, testing, and optimization, making the use of simulators an integral part of the process of creating and improving autonomous vehicles. Different types of simulators are used to effectively model and test different operational aspects of autopilot systems, each focusing on particular tasks. Traffic flow simulators are intended to model and analyze vehicle interactions in urban and suburban environments; vehicle dynamics simulators focus on the physical behavior of a vehicle; sensor and perception simulators serve to model data from cameras, lidars, and other sensors. In addition, there are simulators for creating complex motion scenarios; they allow testing decision algorithms in various road situations.

In this section, we discuss the main types of simulators used to develop and test autopilot systems, their key features, and examples of the most popular solutions in each category.

1.1. Traffic Flow Simulators

Traffic flow simulators are intended to model the motion of multiple vehicles on roads, including their interactions, in order to analyze and optimize transportation systems. These simulators help to study the dynamics of traffic flows, the impact of different scenarios on congestion, the effectiveness of various road infrastructures, and the behavior of road users in different situations [7].

Key features:

• motion modeling for a large number of vehicles;

• support for different types of road networks and scenarios;

• the ability to integrate with transportation management systems (e.g., traffic lights);

• the analysis and visualization of traffic flows and congestion.

Examples of simulators:

• SUMO (Simulation of Urban Mobility) is one of the most widespread simulators for modeling traffic flows in urban environments. It supports the modeling of large urban networks and integration with other simulators [8].

• AIMSUN is a commercial simulator used for traffic flow analysis and management, with support for complex vehicle-to-vehicle interaction models [9].

1.2. Vehicle Dynamics Simulators

These simulators focus on modeling the dynamic performance characteristics of individual vehicles. They are used to analyze vehicle behavior in different conditions (acceleration, braking, steering on slippery surfaces, and interaction with road irregularities). Such simulators are important for the development and testing of control systems, particularly stabilization and autonomous driving systems [10].

Key features:

• the high-accuracy modeling of physical processes (chassis, suspension, engine, and brake system dynamics);

• modeling of vehicle-road surface interaction;

• support for different types of vehicles, including cars, trucks, and motorcycles;

• the ability to simulate extreme conditions (accidents).

Examples of simulators:

• CarMaker is an industry standard for vehicle dynamics modeling. It supports the testing of *advanced driver-assistance systems* (ADAS) [11].

• TruckSim is a special-purpose simulator for modeling the dynamics of heavy vehicles (trucks and buses) [12].

1.3. Sensor and Perception Simulators

These simulators focus on modeling the sensors used in autonomous vehicles (cameras, lidars, radars, and ultrasonic sensors). The main purpose of such simulators is to reproduce sensor data realistically, which can be used to develop and test perception and decision algorithms [13].

Key features:

• realistic modeling of sensor data in different environmental conditions;

• support for multiple sensor types and combinations;

• the ability to integrate with image and signal processing algorithms;

• testing and debugging perception systems in complex scenarios (poor weather conditions or poor lighting).

Examples of simulators [14]:

• CARLA provides a wide range of sensors and models for testing perception systems in urban environments.

• AirSim is a Microsoft simulator that supports the realistic modeling of sensor data and is used to develop autonomous drones and ground vehicles.

1.4. Complex Simulators

These simulators are intended to create and test complex traffic scenarios involving autonomous vehicles. They allow the modeling of various traffic situations and the interaction of an autonomous vehicle with other road users, which is especially important for developing decision systems. As a rule, they include a certain implementation of other types of simulators or have the possibility of integrating third-party solutions [15, 16].

Key features:

• support for creating complex scenarios with multiple vehicles and pedestrians;

• integration with decision and traffic control algorithms;

• the ability to simulate rare and extreme situations such as accidents or sudden obstacles;

• analysis and visualization of decisions made and their consequences.

Examples of simulators:

• CARLA is a flexible and extensible platform for training, testing, and validating autonomous driving systems; built on the Unreal Engine visualizer, CARLA offers highly accurate environments, realistic physics, and a full suite of sensors, including cameras, lidars, and radars [17].

• PreScan is used for the development and testing of ADAS systems and autonomous vehicle systems, including the modeling of complex scenarios and traffic situations.

• LGSVL Simulator supports the modeling of various scenarios and integration with autonomous control development platforms (Apollo and Autoware) [14].

• AutoDRIVE Simulator is a high-accuracy simulation platform developed using the Unity game engine; it includes a vehicle model equipped with a comprehensive set of sensors and actuators to facilitate research and training in autonomous vehicle technologies [18].

2. WORKING WITH ROUGH TERRAIN SCENES

The tasks set before modern unmanned vehicles require them to move on smooth, asphalted, and marked roads and, moreover, on rough terrain with different types of surfaces, ranging from dirt to snow [19]. To practice control, navigation, and localization algorithms, an appropriate simulator is required to implement motion over challenging terrain with a set of typical obstacles. Since the use of unmanned vehicles is not yet widespread, the number of appropriate simulators is significantly limited. Note the additional requirements for such simulators compared to those involving asphalted roads:

• The physical terrain model. It must support terrain with unevenness, different types of ground (gravel, sand, rocks), water, and other natural obstacles. This allows one to test the vehicle's response to slippery or loose surfaces as well as uphill and downhill slopes.

• Detailed vehicle models. It is important to consider the characteristics of different types of vehicles, whether they are autonomous cars, tracked vehicles, wheeled robots, or even drones capable of moving over difficult terrain. Simulators must accurately model suspension, grip, dynamometry, and other important aspects.

• Environment. Simulators must include various terrain types and simulate weather conditions (snow, fog, and wind) that can strongly affect vehicle handling. In the simulators discussed below, different weather conditions affect only the visual series acquired by sensors; the impact of these factors on grip, if possible, is explicitly shown in the simulator specifications.

The next subsections describe in detail several simulators that can be used to model motion over rough terrain.

2.1. Gazebo

This is one of the most popular simulators of robots and unmanned vehicles, supporting realistic physics and a high degree of customization. Due to its open-source code, the simulator is used in research and educational projects, as well as in commercial developments [20, 21]. It represents an open platform to integrate various physics engines (ODE, Bullet, Simbody, and others).

Advantages:

• Realistic physics. Gazebo supports challenging terrain with bumps, hills, rocks, and even water, allowing one to simulate scenarios of autonomous vehicle motion over rough terrain.

• Grip and suspension simulation. It is important for tracked or wheeled vehicles operating on rough terrain; one can accurately simulate how the vehicle will behave over challenging terrain.

• Support for different surface types. The simulator can simulate different types of ground, including slippery or loose surfaces (sand, gravel, and mud).

• Environment. One can add effects (rain, wind, and snow) that influence motion conditions over rough terrain.

• Integration with Robot Operating System (ROS). Gazebo is actively used in conjunction with ROS, which makes it convenient for developing and testing autonomous systems in real-world conditions.

Drawbacks:

• Scene creation. Although Gazebo provides tools for creating scenes, developing detailed rough landscapes can be time-consuming. Users often face the need to manually model challenging terrain elements.

• Map import. Maps can be automatically imported from real data (e.g., satellite data) via additional plugins, but this requires additional customization and does not necessarily provide sufficient detail for challenging terrain.

• The complexity of model creation. Creating detailed 3D vehicle models requires knowledge of 3D modeling and physical simulation. This process can be challenging for users unfamiliar with such tools.

• Integration with other systems. While Gazebo integrates well with ROS, integration with other frameworks may require additional effort. For example, the use of other physics engines or control interfaces may be limited without manual configuration.

Complexity: medium. Gazebo is oriented toward researchers, so a good understanding of ROS and simulation basics will be required to create and set up complex scenes.

2.2. CARLA

This is an open-source simulator developed for testing autonomous vehicles in urban environments [22]. However, it can be adapted to rough terrain, as it provides sufficient capabilities to create nonstandard landscapes [23].

Advantages:

• Environment customization. CARLA provides tools to create custom maps, allowing one to build challenging terrain with rough elements.

• Support for realistic motion physics. CARLA can simulate vehicle dynamics over rough terrain, including speed, grip, and stability control.

• The ability to work with different types of surfaces. Despite its urban orientation, CARLA can simulate grass, dirt, sand, and other types of surfaces.

• Weather conditions. Various weather conditions affecting grip and visibility can be simulated.

Drawbacks:

• Scene creation. CARLA is intended primarily for urban environments, and creating rough terrain scenes may require manual customization. The built-in maps do not include rough terrain, so it is necessary to import user maps and manually customize terrain.

• Map import. Importing real maps requires additional tools and modules, as well as skills in working with 3D graphics and geospatial data. Despite the declared support for rough terrain [24], no examples or publications on the application of this functionality have been found so far, making adaptation to rough terrain difficult.

• The complexity of model creation. CARLA contains ready-made vehicle models, but third-party tools such as Blender or Maya have to be used to create unique models. This can be challenging, especially if one needs to detail suspension and motion dynamics.

• Integration with other software solutions. This simulator integrates well with Python API for scripting, but interaction with other systems requires additional customization. There is no built-in ROS support, which can be disadvantageous for ROS projects.

Complexity: high for rough terrain. Considerable effort will be required to create detailed natural scenes and integrate real maps.

2.3. AirSim

This simulator was developed by Microsoft for drones and ground vehicles [25]. Its main advantage is integration with Unreal Engine, which allows one to create highly detailed 3D scenes, including rough terrain [26].

Advantages:

• Highly detailed terrain. Owing to Unreal Engine, AirSim can accurately simulate various challenging terrain types, from mountainous landscapes to dense forests.

• Support for realistic motion physics. Different types of vehicles can be simulated in AirSim, including wheeled and tracked platforms, allowing one to test motion over challenging terrain.

• Sensor customization. The simulator enables one to model various sensors (cameras, lidars, and GPS trackers), which is especially useful for testing operation in rough terrain conditions with low-level signals.

• Weather and lighting conditions. It is possible to simulate various weather conditions (rain, snow, and fog), which considerably complicate navigation over rough terrain.

Drawbacks:

• Scene creation. Since AirSim involves Unreal Engine, creating a scene requires working with the game engine's tools. Despite its visual power, Unreal Engine has a fairly high entry threshold for beginners. Creating complex landscapes and environments can be time-consuming and require serious 3D modeling skills.

• Map import. AirSim has no direct option for importing real geodata. Third-party tools are available, but their setup is complicated.

• The complexity of model creation. One has to use Unreal Engine or third-party 3D modeling programs to create detailed vehicle models. The process of integrating new models with physical simulation can be labor-intensive.

• Integration with other software solutions. AirSim provides API for working with Python and C++, but interaction with ROS or other systems will require additional effort. Integration with other frameworks is limited compared to Gazebo.

Complexity: high. Despite powerful visualization capabilities, the complexity of creating scenes and working with real maps makes AirSim more time-consuming for academic or commercial projects with rough terrain.

2.4. NVIDIA Isaac Sim

This robot simulation platform from NVIDIA is intended to support complex tasks of robotics and autonomous transportation. Based on the capabilities of the graphics processing unit (GPU), it allows one to simulate complex scenarios, including rough terrain. Also, custom extensions can be implemented with flexible functionality [27–29].

Advantages:

• Photorealistic environment. The PhysX engine and GPU acceleration allow accurately simulating vehicle motion physics over challenging terrain and various surfaces.

• Integration with real-world algorithms. Isaac Sim supports integration with deep learning and path planning algorithms to model and test complex scenarios over rough terrain.

• Support for different robot types. The simulator can model both wheeled vehicles and tracked robots or drones, thereby being versatile for autonomous motion tasks over rough terrain.

• Sensors. The simulation of complex sensor systems, including cameras, lidars, and GPS trackers, is supported to test robot performance in rough terrain conditions.

Drawbacks:

• Scene creation. Although Isaac Sim provides tools for creating complex scenes, modeling rough terrain will require considerable effort. The platform is oriented toward high-performance computing using GPUs, which can make development difficult for users without powerful hardware.

• Map import. It is possible to integrate real map models into Isaac Sim through 3D modeling, but the process involves third-party tools and can be quite complex, especially for the realistic simulation of terrain and natural conditions.

• The complexity of model creation. Creating new vehicle models and adapting them to motion physics requires extensive training and knowledge of 3D graphics. Incorporating suspension dynamics and complex mechanisms can be challenging for non-professionals.

• Integration with other software solutions. The platform integrates well with NVIDIA AI tools, but integration with ROS or other autonomous systems may require additional modules and customizations. This can be difficult for projects expecting quick and easy integration.

2.5. Webots

This free open-source robot simulator is used for educational and research purposes [30, 31]. It supports a wide range of robotic systems, including unmanned vehicles.

Advantages:

• Flexible terrain modeling. In Webots one can create custom models of challenging terrain, including mountains, hills, canyons, and other natural objects.

• Support for various vehicles. Webots can simulate the operation of both wheeled and tracked vehicles, allowing one to test motion algorithms over challenging terrain.

• Sensor modeling. Webots supports a wide range of sensors, therefore being suitable for testing autonomous systems over rough terrain using cameras, lidars, and GPS trackers.

• Weather conditions. One can simulate different weather conditions (rain, fog, etc.) that affect visibility and grip.

Drawbacks:

• Scene creation. Webots has a user-friendly interface for creating simple scenes; however, when it comes to complex rough terrain, one has to customize reliefs and surfaces manually. This can be a limitation compared to other (more advanced) simulators.

• Map import. Importing real maps is not directly supported: one has to use third-party tools to create complex reliefs and challenging terrain. Therefore, rough terrain modeling will be much more difficult compared to more powerful simulators.

• The complexity of model creation. Webots has a library of standard robots, but creating custom models requires using third-party 3D modeling tools. Built-in modeling tools are limited, which complicates work with unique vehicles.

• Integration with other software solutions. Webots supports integration with ROS and other popular frameworks. However, more complex tasks, such as deep integration with external AI systems, may require the development of additional modules.

2.6. Blender with OpenDroneMap photogrammetry

Blender is an editor for creating 3D computer graphics, including modeling, sculpting, animation, simulation, rendering, post-processing, and sound video editing. OpenDroneMap is a set of photogrammetry tools based on aerial images, which generates 3D maps of the captured terrain [32].

Advantages:

• Modeling. The platform provides almost unlimited capabilities for modeling and simulation of processes: there is a rich set of tools, from basic to highly specialized, to build a custom simulator [33].

• Map import. Plug-ins are available to import maps and project satellite images onto public elevation data.

• Animation. This platform has flexible options for creating animation and access to the Python API, where any properties can be changed.

• User-friendly interface. After creating the necessary auxiliary tools, the tasks of importing sensor records are reduced to one-button solutions with quick debugging of the results.

Drawbacks:

• Modularity. It is necessary to search (create, assemble) all parts of the simulation.

• GNSS support. There is no built-in binding of the 3D-editor base space to geographic coordinates: all work with real data requires the careful recalculation of coordinate frames and the preliminary preparation (processing) of records.

• Interaction physics. Note the complexity of simulating the physical interaction between vehicle and terrain. Blender is primarily intended for animation; its built-in physics emulation tools are often used to simplify the realization of artistic intent rather than to calculate physical loads.

Complexity: high. Despite powerful visualization capabilities, the primary complexity of creating scenes and working with real maps makes the simulator difficult to use in academic or commercial projects with rough terrain.

2.7. NVO73

This is a 3D simulator developed at the Trapeznikov Institute of Control Sciences, the Russian Academy of Sciences, based on Unreal Engine 5.2 and AirSim to simulate the joint operation of a group of unmanned ground, underwater, surface, and aerial vehicles [34]. Using AirSim as the core inherits all its advantages and drawbacks. There is an adapted and simplified assembly for viewing the records of electric vehicle paths, along with the operational characteristics of the power unit.

Advantages:

• Easy viewing control. There are a convenient slider with a timeline and start/stop buttons.

• Support for heterogeneous vehicles. The platform supports four types of unmanned vehicles: ground, aerial, surface, and underwater.

• The variety of animations. One can visualize engine revolutions through wheel rotation animation as well as change lighting and weather conditions.

• Simplified run and import of records. The entire environment runs with a single executable file and reads the prepared record file from a predefined folder.

• A built-in algorithm corrects the path record by elevation, linking the motion to the map surface.

Drawbacks:

• No ability to add terrain models. New maps can be added only with developer involvement.

• No display of the entire path. For each time instant, only the current position of the vehicle is shown, making it difficult to assess the whole record.

• Surface referencing ignores recorded elevation data, making the simulation highly dependent on the quality of the terrain model.

• No documentation. All information about the simulator can be obtained only from several papers or directly from its developers, which causes difficulties during the installation procedure and further use.

2.8. Comparison of the Simulators Considered

Each of the above simulators provides some rough terrain modeling capabilities. Which simulator should be chosen? The answer depends on the specifics of the simulation project. Gazebo and AirSim are suitable for the highly detailed simulation of physical conditions; CARLA can be adapted for nonstandard tasks; NVIDIA Isaac Sim offers powerful GPU simulation capabilities; Webots is a simple and flexible tool for educational purposes. Gazebo and Webots are easier to integrate with ROS but may require more manual work to create complex natural landscapes. In turn, CARLA and AirSim are good for high-quality visualization but require significant effort to create rough terrain scenes and import real data. NVIDIA Isaac Sim offers strong support for AI simulations but requires high-performance resources and complex customization to create a scene. Building a terrain model with OpenDroneMap and simulating motion records in Blender are suitable for visualizing terrain and paths but provide no ready-made tools for simulating physical interactions. Most of these simulators support interaction with external tools, making it possible to reproduce the behavior of a real object in a simulation environment.

To select the simulator, we define the main parameters under comparison.

• RTF (*Real-Time Factor*), an index to evaluate the speed of simulation relative to real time. It is widely used in robotics and autonomous transportation systems to analyze performance and computational efficiency.

• Sensor refresh rate, representing the frequency of updating the readings of sensors (lidars, cameras, and GPS trackers). The higher this rate is, the more accurate the simulation process will be (of course, at the cost of increasing the load on the processor and graphics units).

• Graphics engine. It is responsible for rendering images. Unreal Engine and Omniverse RTX provide high-quality graphics but require powerful hardware. OGRE and OpenGL are simpler and more lightweight.

• Physics engine. It is responsible for simulating motion dynamics, collisions, friction force, suspension reactions, and other physical effects. The more powerful the engine is, the more realistically the objects will behave.

• The maximum number of vehicles (robots). It shows how many objects can be modeled simultaneously. This number is limited by the computing power of the processor and video adapter.

The simulators considered in the survey are compared by these parameters in Table 1.

The final expert assessment of the capabilities and technical specifications of the simulators is given in Table 2; scores 1, 2, and 3 correspond to the least complicated, medium, and most complicated ones. According to Table 2, it seems impossible to identify a simulator excelling than the others by all parameters.

3. PRACTICAL APPLICATION OF SIMULATORS

We have to simulate the motion of wheel-tracked and ski-tracked platforms (Fig. 1), primarily to analyze their application scenarios in the sections of a distributed network of testing grounds located in the mountainous areas of the Murmansk region and the Caucasus [5]. To solve scientific and applied tasks, the simulator must satisfy the following requirements:

• support for rough terrain motion, including various types of surfaces (dirt, snow, and sand);

• the availability of ready-made vehicle models, close by characteristics to those in Fig. 1, or the possibility of adding a custom model;

• Python integration;

• shareware (no need to purchase a license);

• the ability to display motion based on real vehicle data.

Table 1

Simulators for unmanned vehicles: a comparison of characteristics

Characteristics	Gazebo	CARLA	AirSim/NVO73	NVIDIA Isaac Sim	Webots
Physics engine	ODE, Bullet, Simbody, DART	Unreal Engine PhysX	Unreal Engine PhysX	NVIDIA PhysX	ODE
RTF	1.0–2.0	~1.0 (depends on the video adapter)	~1.5 (depends on settings)	1.0 (high GPU demands)	1.0–3.0 (lightweight)
Sensor refresh rate	100–1000 Hz	~100 Hz (cameras, lidars)	120 Hz (lidars), 30–60 Hz (cameras)	240 Hz (lidars), 60 Hz (cameras)	~100 Hz
Graphics engine	OGRE (basic)	Unreal Engine 4	Unreal Engine 4 (5.2 in the case of NVO73)	Omniverse RTX (high detail)	Built-in OpenGL-based
The maximum number of vehicles	50+ (optimized)	~20–50 (depends on settings)	10–30 (depends on GPU)	100+ (with RTX acceleration)	10–50 (optimized for mobile robots)

Table 2

Simulators for unmanned vehicles with rough terrain support: expert assessments (a lower score corresponds to a higher priority)

Criterion	Gazebo	CARLA	AirSim/NVO73	NVIDIA Isaac Sim	Webots	ODM+Blender
Scene creation	2	3	3	2	1	2
Map import	2	3	3	3	3	2
The complexity of model creation	2	3	3	3	1	2
Integration with other software solutions	1	1	1	2	2	1
Rough terrain support	1	3	1	1	2	1
Display of real object's logs	1	1	3	2	1	2
Equipment requirements	1	2	3	3	1	2

According to the survey results and Table 2, it is impossible to identify one simulator excelling the others in all parameters. The most suitable simulators are Webots, Isaac Sim, and Gazebo. However, Isaac Sim requires high-performance video adapters; Gazebo leads by all parameters, except for the complexity of creating scenes (being inferior to Webots). We do not care so much about the ability to create custom scenes, so further experiments and testing will be done using Gazebo. This simulator offers tools for modeling the physical behavior of a vehicle, working with sensors, and visualizing the collected data. The tracked platforms under consideration have complex dynamics, especially when moving on rough terrain. Therefore, Gazebo with its realistic physics engine allows testing important aspects such as track grip, suspension behavior, and the effects of different types of ground (sand, dirt, and rocks) [35].

We tried two different tools to display real data recorded on a motorcycle or tracked platform:

– the NVO 73 simulator based on Unreal Engine [34, 36];

– Blender with OpenDroneMap photogrammetry.

The functionality of NVO 73 is still very limited, which prevents it from satisfying all the requirements for visualization tools. However, it is possible to reproduce motion from a pre-recorded log file. The interface of this simulator is shown in Fig. 2.

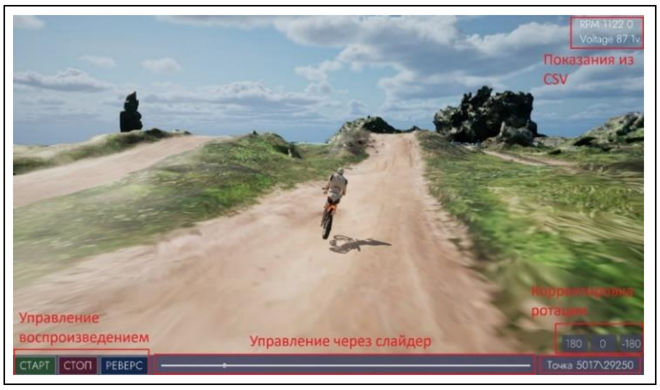

Fig. 2. The interface of the NVO 73 simulator.

Rendering motion paths on terrain models from OpenDroneMap in Blender turns out to be quite visual (Fig. 3), also demonstrating the record of body movements. However, the synchronization of records is very labor-intensive, as it requires manual corrections for all initial positions (the terrain model and satellite navigation coordinates, and the mutual location of the vehicle and body) [36]. Without developing a physical motion model and binding the vehicle to the ground surface and the body to the vehicle, the reconstruction process yields periodic significant discrepancies with the control video record: the vehicle flies into the sky or falls under the ground, whereas the body turns and shifts in different directions.

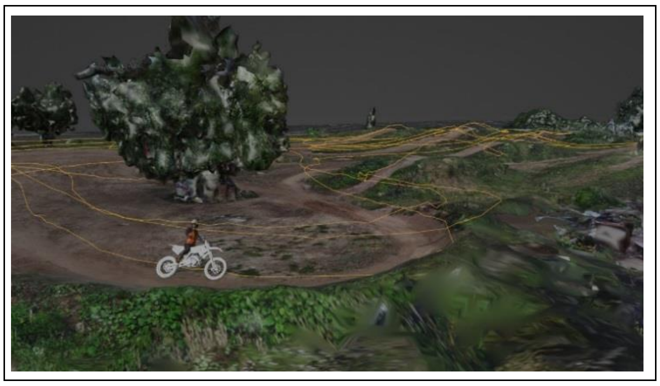

Fig. 3. Rendering motion in Blender with ODM.

CONCLUSIONS

This survey has been devoted to popular simulators for modeling various aspects of autonomous vehicles. Special attention has been paid to simulators supporting rough terrain: Gazebo, CARLA, AirSim, NVIDIA Isaac Sim, and Webots. Each of these simulators has unique capabilities, suitable for different applications, and some limitations as well.

For example, Gazebo and Webots stand out for easy integration with ROS and low hardware requirements but, at the same time, need significant effort to create complex rough terrain scenes. CARLA and AirSim offer high-quality visualization and flexibility for custom scenarios; however, the complexity of setting up rough terrain and high hardware load can be limiting factors for these platforms. In turn, NVIDIA Isaac Sim demonstrates outstanding AI and GPU-accelerated simulation capabilities but has high computational demands and complexity of customization. The final choice of an appropriate simulator depends on project goals, available hardware, and the level of simulation detail required. Regardless of the choice, the use of simulators significantly accelerates the development and testing of autonomous systems, minimizing the risks and costs of real-world testing. For tasks involving rough terrain, one should consider the simulator's capabilities and, moreover, the complexity of integrating real data (relief and motion records), which is especially relevant for research and optimization of autonomous control algorithms.

Acknowledgments. *The research of N.A. Korgin was supported in part by the Russian Science Foundation, project no. 23-29-00681, https://rscf.ru/project/23-29-00681.*

REFERENCES

1. Karunakaran, D., Berrio, J.S., and Worrall, S., Challenges of Testing Highly Automated Vehicles: a Literature Review, *Proceedings of 2022 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, Tainan, 2022, pp. 1–8. DOI: 10.1109/RASSE54974.2022.9989562
2. Beringhoff, F., Greenyer, J., and Roesener, C., Thirty-One Challenges in Testing Automated Vehicles: Interviews with Experts from Industry and Research, *Proceedings of 2022 IEEE Intelligent Vehicles Symposium (IV)*, Aachen, 2022, pp. 360–366. DOI: 10.1109/IV51971.2022.9827097
3. Lou, G., Deng, Y., Zheng, X., et al., Testing of Autonomous Driving Systems: Where Are We and Where Should We Go?, *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Singapore, 2022, pp. 31–43.
4. Martinez, M. and Sitawarin, C., Beyond Grand Theft Auto V for Training, Testing and Enhancing Deep Learning in Self-Driving Cars, *arXiv:1712.01397*, 2017.
5. Korgin, N.A. and Meshcheryakov, R.V., The Conceptual Project on Creating a Distributed Network of Testing Grounds for Practicing the Application Scenarios of Heterogeneous Groups of Electrically Driven Vehicles in Difficult Climatic and Landscape Conditions, *Trudy 11-oi Vserossiiskoi nauchnoi konferentsii "Sistemnyi sintez i prikladnaya sinergetika"* (Proceedings of the 11th All-Russian Scientific Conference "System Synthesis and Applied Synergetics"), Nizhnii Arkhyz, Rostov-on-Don: Southern Federal University, 2022, pp. 197–202. (In Russian.)
6. Makarov, M.I., A Local Path Planning Algorithm for Avoiding Obstacles in the Frenet Frame, *Control Sciences*, 2024, no. 3, pp. 56–61.
7. Mitrohin, M.A., Alyaev, A.O., Lobanov, R.I., and Semenkin, M.V., Investigation of the Influence of Lighting Objects

Control Algorithms on the Characteristics of Road Traffic at Intersections, *Transport Automation Research*, 2024, no. 3, pp. 282–295.

8. Lim, K.G., Lee, C.H., Chin, R.K., et al., SUMO Enhancement for Vehicular Ad Hoc Network (VANET) Simulation, *Proceedings of 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, Kota Kinabalu, 2017, pp. 86–91.
9. Barceló, J., Barceló, P., Casas, J., and Ferrer, J.L., AIMSUN: New ITS Capabilities, *Proc. Eur. ITS Conf.*, Bilbao, Spain, 2001, pp. 1–10.
10. Ghafarian, M., Watson, N., Mohajer, N., et al., A Review of Dynamic Vehicular Motion Simulators: Systems and Algorithms, *IEEE Access*, 2023, vol. 11, pp. 36331–36348.
11. Ziegler, S. and Höpler, R., Extending the IPG CarMaker by FMI Compliant Units, *Proceedings of 8th International Modelica Conference*, Dresden, 2011, pp. 779–784.
12. *TruckSim Overview*. URL: https://www.carsim.com/products/trucksim/index.php. (Accessed September 30, 2024.)
13. Silva, I., Silva, H., Botelho, F., and Pendao, C., Realistic 3D Simulators for Automotive: a Review of Main Applications and Features, *Sensors*, 2024, vol. 24, no. 18, art. no. 5880.
14. Li, Y., Yuan, W., Zhang, S., et al., Choose Your Simulator Wisely: a Review on Open-Source Simulators for Autonomous Driving, *IEEE Transactions on Intelligent Vehicles*, 2024, vol. 9, no. 5, pp. 4861–4876.
15. Cantas, M.R. and Guvenc, L., Customized Co-simulation Environment for Autonomous Driving Algorithm Development and Evaluation, *arXiv:2306.00223*, 2023.
16. Holen, M., Knausgard, K., and Goodwin, M., An Evaluation of Autonomous Car Simulators and Their Applicability for Supervised and Reinforcement Learning, *Proceedings of International Conference on Intelligent Technologies and Applications*, Grimstad, 2021, pp. 367–379.
17. May, J., Poudel, S., Amdan, S., et al., Using the CARLA Simulator to Train a Deep Q Self-Driving Car to Control a Real-World Counterpart on a College Campus, *Proceedings of 2023 IEEE International Conference on Big Data*, Sorrento, 2023, pp. 2206–2210.
18. Tanmay, V.S., Chinmay, V.S., and Ming, X., AutoDRIVE Simulator: a Simulator for Scaled Autonomous Vehicle Research and Education, *Proceedings of the 2021 2nd International Conference on Control, Robotics and Intelligent System (CCRIS '21)*, Qingdao, 2021, pp. 1–5.
19. Hanevold, M., Path Following Model Predictive Control of a Differential Drive UGV in Off-Road Terrain, *Master of Informatics Thesis*, Oslo: University of Oslo, 2022.
20. Zheng, H., Smereka, J.M., Mikulski, D., et al., Bayesian Optimization Based Trust Model for Human Multi-Robot Collaborative Motion Tasks in Offroad Environments, *International Journal of Social Robotics*, 2023, vol. 15, no. 7, pp. 1181–1201.
21. Koenig, N. and Howard, A., Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator, *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, 2004, vol. 3, pp. 2149–2154.
22. Dosovitskiy, A., Ros, G., Codevilla, F., et al.. CARLA: An Open Urban Driving Simulator, *Proceedings of Conference on Robot Learning*, Mountain View, 2017, pp. 1–16.
23. Han, I., Park, D.H., and Kim, K.J., A New Open-Source Off-Road Environment for Benchmark Generalization of Autonomous Driving, *IEEE Access*, 2021, vol. 9, pp. 136 071–136 082.
24. *Let's go off-road!* URL: https://carla.org/2023/04/21/avl-off-road-simulation. (Accessed September 30, 2024.)
25. Shah, S., Dey, D., Lovett, C., and Kapoor, A., Airsim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, *Proceedings of the 11th International Conference on Field and Service Robotics*, Cham: Springer, 2018, pp. 621–635.
26. Jansen, W., Verreycken, E., Schenck, A., et al., COSYS-AIRSIM: a Real-Time Simulation Framework Expanded for Complex Industrial Applications, *Proceedings of 2023 Annual Modeling and Simulation Conference (ANNSIM)*, Hamilton, 2023, pp. 37–48.
27. Richard, A., Kamohara, J., Uno, K., et al., Omnilrs: A Photorealistic Simulator for Lunar Robotics, *Proceedings of 2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, 2024, pp. 16901–16907.
28. Jacinto, M., Pinto, J., Patrikar, J., et al., Pegasus Simulator: an Isaac Sim Framework for Multiple Aerial Vehicles Simulation, *Proceedings of 2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, Chania, 2024, pp. 917–922.
29. Ellis, K., Zhang, H., Stoyanov, D., and Kanoulas, D., Navigation among Movable Obstacles with Object Localization Using Photorealistic Simulation, *Proceedings of 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Huntington Place, 2022, pp. 1711–1716.
30. Zea, D., Toapanta, A., and Perez, V.H., Intelligent and Autonomous Guidance through a Geometric Model for Conventional Vehicles, in *Innovation and Research:* A *Driving Force for Socio-Econo-Technological Development*, Cham: Springer, 2020, pp. 78–93.
31. Couceiro, M.S., Vargas, P.A., and Rocha, R.P., Bridging the Reality Gap between the Webots Simulator and E-puck Robots, *Robotics and Autonomous Systems*, 2014, vol. 62, no. 10, pp. 1549–1567.
32. Zhang, C. and Maga, A.M., An Open-Source Photogrammetry Workflow for Reconstructing 3D Models, *Integrative Organismal Biology*, 2023, vol. 5, no. 1, art no. obad024.
33. *OpenDroneMap + blender = Fun*. URL: https://smathermather.com/2019/12/30/opendronemap-blender-fun-part-2/. (Accessed September 30, 2024.)
34. Amosov, O.S., Amosova, S.G., and Kulagin, K.A., Modeling of a Virtual Testing Ground for Practicing Joint Navigation of a Group of Heterogeneous Unmanned Vehicles, *Trudy 34-oi konferentsii pamyati vydayushchegosya konstruktora giroskopicheskikh priborov N.N. Ostryakova* (Proceedings of the 34th Conference in Memory of Outstanding Designer of Gyroscopic Devices N.N. Ostryakov), St. Petersburg, 2024, pp. 117–120. (In Russian.)
35. Dobrokvashina, A., Lavrenov, R., Bai, Y., et al., Servosila Engineer Crawler Robot Modelling in Webots Simulator, *International Journal of Mechanical Engineering and Robotics Research*, 2022, vol. 11, no. 6, pp. 417–421.
36. Trefilov, P., Kulagin, K., and Mamchenko, M., Developing a Flight Mission Simulator in the Context of UAVs Group Control, *Proceedings of 2020 13th International Conference "Management of Large-Scale System Development" (MLSD)*, Moscow, 2020, pp. 1–4. DOI:10.1109/MLSD49919.2020.9247692
37. Bazenkov N.I. and Pyzh'yanov, A.A., A Data Acquisition System for Reconstructing Motorcyclist Movements, *Trudy 14-go Vserossiiskogo soveshchaniya po problemam upravleniya* (Proceedings of the 14th All-Russian Meeting on Control Problems), Moscow, 2024, pp. 1776–1780. (In Russian.)

This paper was recommended for publication by R.V. Meshcheryakov, a member of the Editorial Board.

Received December 3, 2024,
and revised February 11, 2025.
Accepted February 12, 2025.

Author information

Makarov, Maxim Igorevich. Junior researcher, Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ maxim.i.makarov@gmail.com
ORCID iD: https://orcid.org/0000-0002-4854-5910

Korgin, Nikolai Andreevich. Dr. Sci. (Eng.), Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ nkorgin@ipu.ru
ORCID iD: https://orcid.org/0000-0003-1744-3011

Pyzh'yanov, Andrei Alexandrovich. Programming engineer, Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ ipu@isko.moe
ORCID iD: https://orcid.org/0009-0000-4539-6190

Cite this paper

Makarov, M.I., Korgin, N.A., and Pyzh'yanov, A.A., Unmanned Ground Vehicles: a Survey of Modern Simulators. *Control Sciences* 1, 2–12 (2025).

Translated into English by *Alexander Yu. Mazurov*,
Cand. Sci. (Phys.–Math.),
Trapeznikov Institute of Control Sciences,
Russian Academy of Sciences, Moscow, Russia
✉ alexander.mazurov08@gmail.com