Seven Widely Used Open-Source Platforms

Subjects: Automation & Control Systems

Contributor: Ziming Chen , Jinjin Yan , Bing Ma , Kegong Shi , Qiang Yu , Weijie Yuan

Simulation platforms are critical and indispensable tools for application developments of unmanned aerial vehicles (UAVs) because the UAVs are generally costly, have certain requirements for the test environment, and need professional licensed operators. Thus, developers prefer (or have) to test their applications on simulation platforms before implementing them on real machines. Seven widely used open-source simulation platforms, including Webots, Gazebo, CoppeliaSim, ARGoS, MRDS, MORSE, and USARSim.

open-source simulation platform multi-copter UAV

1. Webots

Webots, developed by Cyberbotics Company [1], is an open-source simulation platform, which can simulate three-dimensional agents (models). This platform is highly compatible with the mainstream operation systems, such as Windows, Linux, MAC OS. It can support C, C++, Java, Python, MATLAB and other programming languages [2]. The simulation of a unmanned aerial vehicles (UAVs) in a jungle by Webots is shown in **Figure 1**.

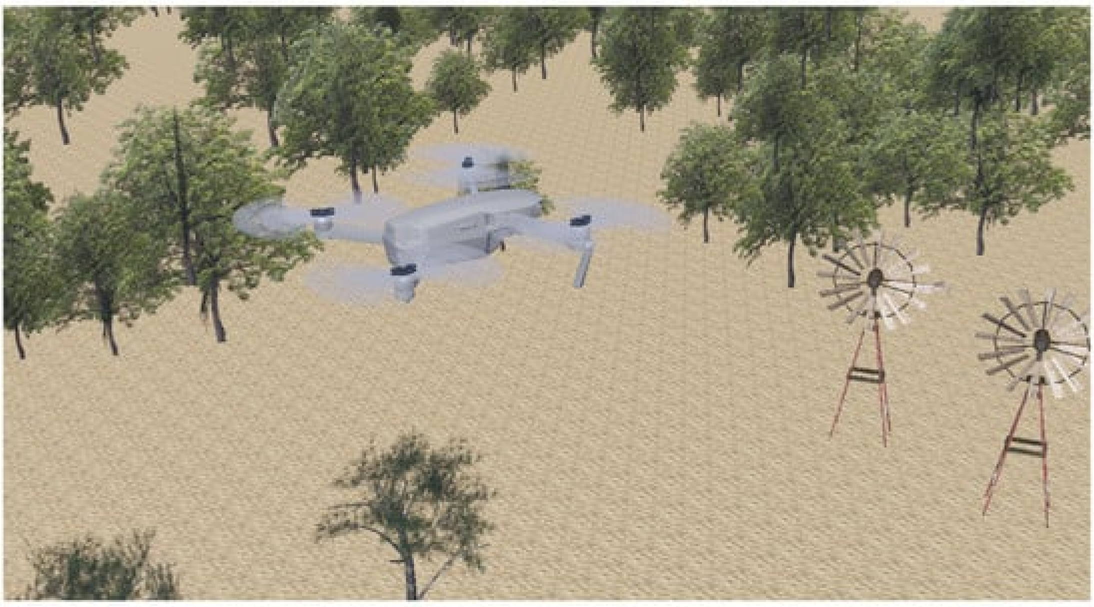


Figure 1. Simulation of a UAV by Webots.

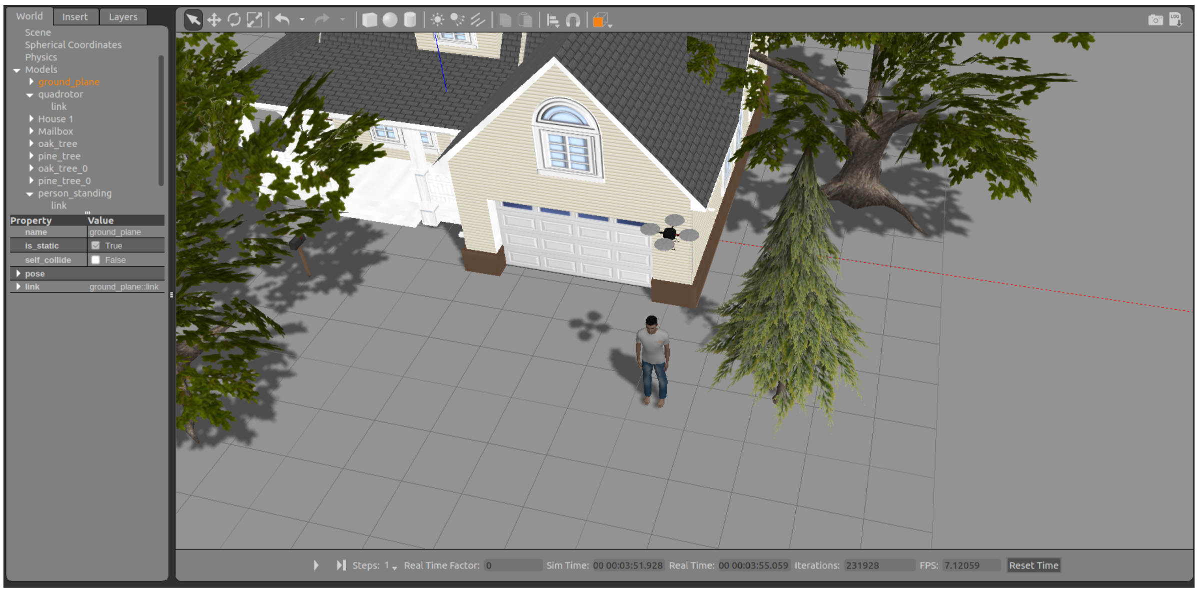
Webots includes a variety of built-in agents (models) such as wheeled robots, underwater robots, UAVs, pedestrians, humanoid robots, and bionic robots. These agents can be easily customized by developers with enriched attributes such as shape, material, and color. Additionally, developers can create their own models by using other tools. For instance, Webots can import files in VRML97 format, which can be produced using SolidWork [3]. Developers also can easily obtain the parameters of the agents, such as motor speed, accelerometer, camera screen, and other such parameters of the robot, via its interface. Webots also includes a range of sensors that are commonly used in robots, such as distance sensors, cameras, gyroscopes, and LIDARs. Moreover, Webots is capable of supporting the Robot Operating System (ROS) [4], an open-source framework that assists researchers and developers in creating and reusing code across various robotics applications. Additionally, Webots allows for the simulation of physical properties such as gravity, illumination, density, and friction coefficients. To simulate collision and dynamics, Webots uses an open-source physical simulation engine known as the Open Dynamics Engine (ODE). This engine facilitates the simulation of rigid body dynamics and collision systems, bringing the simulations closer to realism and greatly enhancing their reliability [5].

In addition to its reliable updates and maintenance, Webots has a thriving community and official technical support. The platform is widely used for simulation and control of UAV, offering numerous examples and tutorials for UAV simulation, as well as support for popular open-source flight control systems such as ArduPilot [6]. There is also an abundance of open-source tutorials available to developers who wish to learn from them. Because of its rich functionality and diverse development environments, more than two hundred universities and research centers use this platform to test applications and algorithms [7]. For example, it can be utilized to evaluate the performance and reliability of UAVs swarm algorithms, such as obstacle avoidance algorithms [8].

2. Gazebo

Gazebo, developed by Dr. Andrew Howard from the University of Southern California in 2002 [9], is a widely used open-source simulation platform in scientific research and engineering, particularly due to its close integration with ROS. Typically, it is used in conjunction with ROS, as Gazebo-ROS [10], to simulate complex robot systems, especially robot swarms [11].

Gazebo is compatible with Windows, Mac OS, Ubuntu, and other Linux distributions. It supports applications and algorithms developed in C, C++, and Python. Gazebo also offers a range of physical simulation engines, including ODE, Bullet [12], Simbody [13], and Dynamic Animation and Robotics Toolkit (DART) [14]. This enables the platform to simulate physical properties such as friction, gravity, and lighting [15]. Additionally, Gazebo can provide high-quality lighting and texture simulations by using its use of the Object-Oriented Graphics Rendering Engine (OGRE). **Figure 2** illustrates the main interface and a simulation example of Gazebo.



Property	Value
name	ground_plane
is_static	True
self_collide	False
pose	
link	ground_plane::link

Steps: 1 Real Time Factor: 0 Sim Time: 00 00:03:51.928 Real Time: 00 00:03:55.059 Iterations: 231928 FPS: 7.12059

Figure 2. The main interface and a simulation example of Gazebo.

Gazebo comes with various built-in robot models, mainly wheeled and flying robots such as TurtleBot, PR2, and iRobot Create. The default model is relatively simple but suitable for complex validation calculations [16]. With ROS integration, this platform supports motion path planning. Sensors such as laser radar, monocular, stereo, RGB-D, and infrared sensors can also be simulated. Developers can customize plug-ins for robots and sensors using plug-in functions, including GUI, system, sensor, world, model, and visual. The sensor plug-in is used to model laser radar and infrared sensors. Additionally, the internal data of Gazebo can be directly accessed by its application program interface (API). Gazebo also offers cloud simulation, enabling developers to run projects on an online server [17].

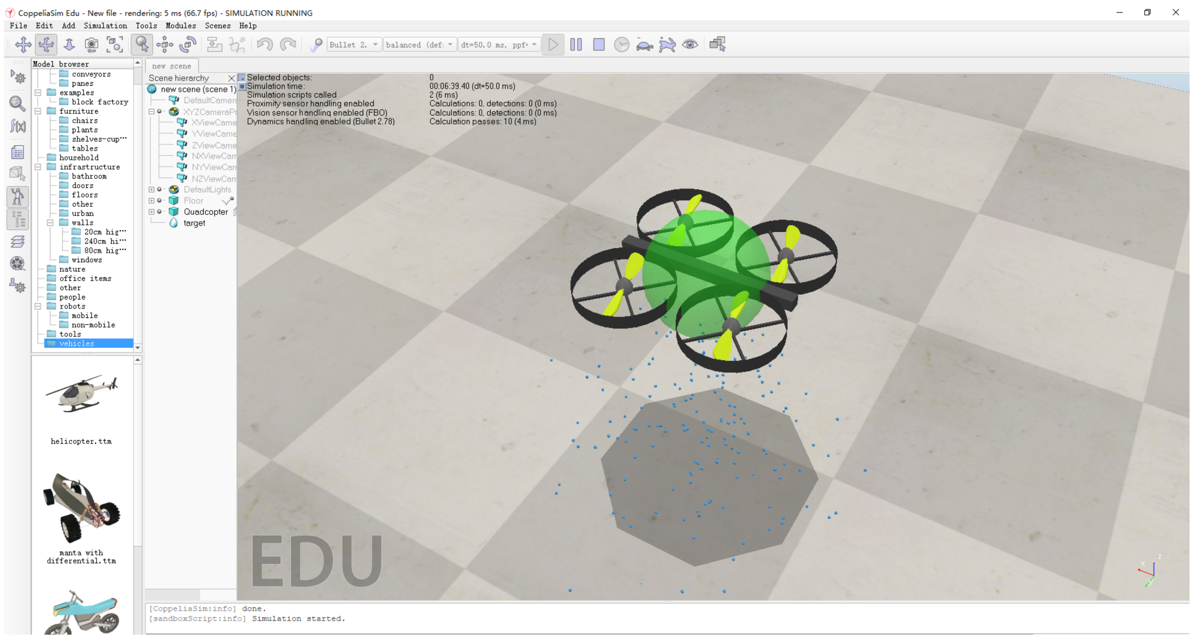
The user operability of Gazebo is relatively poor. It can be difficult to install, and has a high threshold for users [18]. Built-in models are accessed online instead of locally, resulting in longer loading times or failure to load when the internet is unstable. Moreover, the built-in models are not well classified, making it difficult for users to locate the desired models quickly. Developers cannot edit models using this platform; they need to use 3D modeling software (such as Blender, SketchUp) and learn an XML-based SDF specification [17]. However, Gazebo can support for mainstream open-source flight control systems, such as PX4 and ArduPilot, which are widely used on real UAVs, has attracted researchers to evaluate the performance and reliability of UAV control algorithms, such as UAVs swarm validation and visual target tracking [19][20].

In short, the advantage of Gazebo is its close connection with the ROS, which maximizes the use of advantages of ROS. Additionally, its built-in models are accessible online, providing developers with unrestricted access. However, the downside is that poor network connectivity can make it difficult for developers to connect to the server and load the models.

3. CoppeliaSim

CoppeliaSim is a powerful simulation platform for robots, formerly known as V-rep [21]. It supports ODE, Bullet, Vortex, and Newton physics engines, allowing for the simulation of dynamic scenarios such as collisions, rolling, gravity, and other physical factors [22][23]. Other than that, this platform is able to simulate dynamic particles such as jet engines, water jets, and propellers [24]. Its distributed architecture allows for scene object association or an unlimited number of scripts [25], and it supports applications or algorithms developed using C, C++, Python, Java, LUA, Matlab, or Octave. The conventional API is written in C or LUA. CoppeliaSim is highly portable, performs well on Windows, Mac OS, and Linux systems, and has kinematics solver and data visualization functions [23][26].

CoppeliaSim offers a variety of built-in agent models, such as biped, wheeled, bionic, flying, hexapod, and others. The user-friendly interface (**Figure 3**) allows developers to easily select appropriate models. Additionally, the platform supports importing models in mainstream formats such as URDF, COLLADA, DXF, OBJ, STL, and glTF. There are six options for developing applications or algorithms, including embedded script, plugin, add-on, ROS node, remote API, and node talking TCP/IP. CoppeliaSim also provides various sensors for detecting distances, visuals, and pressures. Each type of sensor has different variants; for instance, proximity sensor includes ray-type, randomized ray-type, pyramid-type, cylinder-type, disk-type, and cone-type; vision sensors are divided into orthogonal projection-type and perspective projection-type [27][28]. These sensors can meet the needs of most developers. Furthermore, this platform has rich development resources such as a large number of tutorials, API documents for developers, and an official forum. It is still frequently updated and released, making it a great tool for developers working with robotics simulations.


CoppeliaSim Edu - New file - rendering: 5 ms (66.7 fps) - SIMULATION RUNNING

```
Selected objects:                          0
Simulation time:                           00:06:39.40 (dt=50.0 ms)
Simulation scripts called                  2 (6 ms)
Proximity sensor handling enabled          Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)       Calculations: 0, detections: 0 (0 ms)
Dynamics handling enabled (Bullet 2.78)    Calculation passes: 10 (4 ms)
```

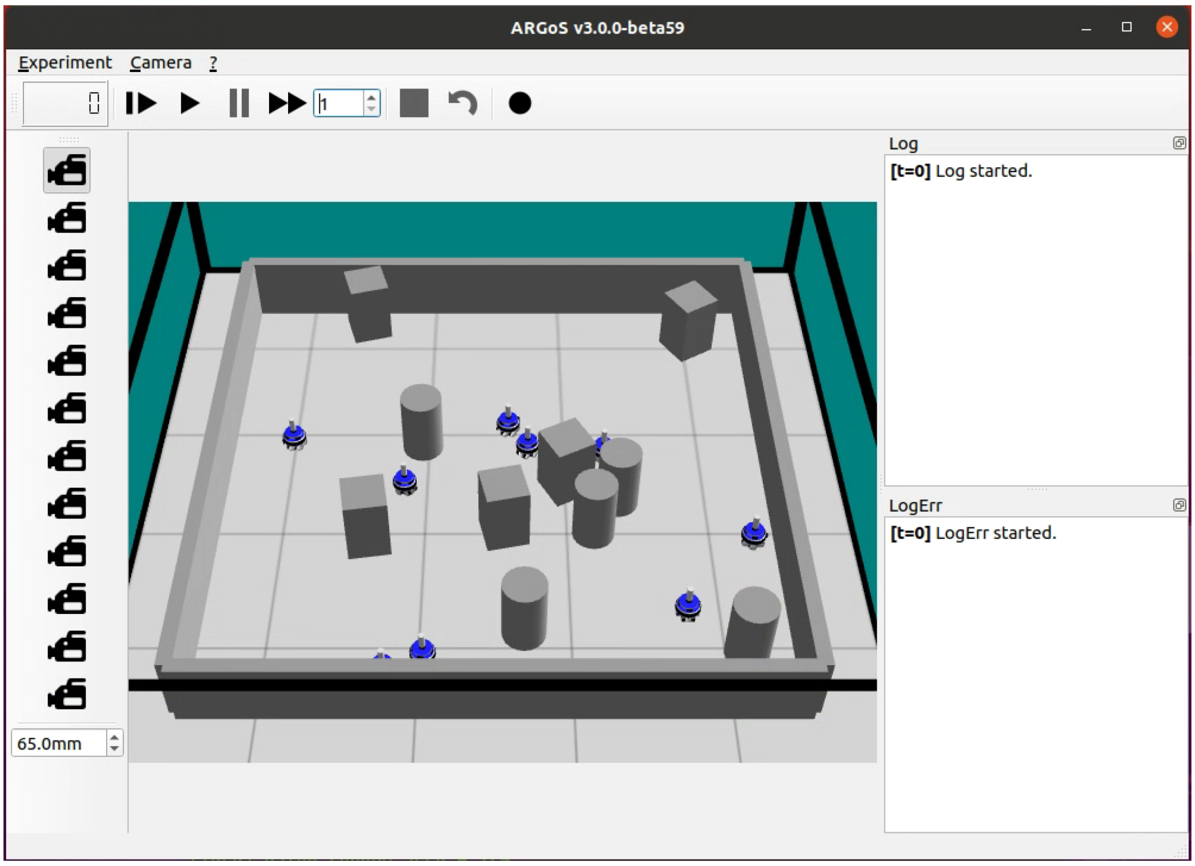
```
[CoppeliaSim:info] done.
[sandboxScript:info] Simulation started.
```

Figure 3. The main interface and a simulation example of CoppeliaSim.

To put it simply, this platform offers a rich selection of built-in models that can fulfill most simulation requirements. It also provides various UAV models, including multi-copter UAVs, with high realism and flexibility, enabling users to simulate different UAV behaviors and motion modes. To enhance UAV behavior simulation, CoppeliaSim allows users to incorporate different sensors such as cameras, LIDAR, GPS, and inertial navigation systems that provide perception and navigation capabilities to help UAVs perform tasks better. With the powerful physics engine, users can simulate various physical effects and model UAV behavior more realistically. In simulations of UAVs swarm, communication and coordination between UAVs are crucial. This platform enables users to set communication protocols and frequencies between UAVs, simulating information exchange and collaborative actions. On the basis of network simulation, researchers can better understand and study UAVs swarm behavior. Many researchers currently use this platform to simulate and verify UAVs swarm algorithms such as entrapping multiple targets [29], formation control [30], and collaborative transportation [31]. However, due to the high accuracy and precision of the built-in models, they may not be suitable for simulating large numbers of swarms [16].

4. ARGoS

ARGoS is a simulation platform developed by the European Union as part of the Swarmanoid project. It is designed to support large-scale robot simulations, and provides a flexible and modular architecture that enables researchers to experiment with a wide range of algorithms and scenarios. ARGoS integrates several engines, including ODE-based 3D dynamics engine, 3D particle engine, Chipmunk-based 2D dynamics engine, and 2D dynamics engine [32]. It also comes with a built-in LUA script editor that allows researchers to easily create and modify simulation scenarios. While ARGoS lacks a scene editor, its modular architecture and built-in scripting capabilities enable researchers to customize and manipulate the simulation environment through custom scripts. The platform supports Ubuntu/Kubuntu, OpenSuse, and Mac OS, but does not currently support Windows [33]. **Figure 4** illustrates the main interface of this platform and a simulation example based on it.



[t=0] Log started.

[t=0] LogErr started.

Figure 4. The main interface and a simulation example of by ARGoS.

The most significant feature of ARGoS is its ability to divide the simulation space into independent subspaces, and each is controlled by a separate physics engine. This enables concurrent execution of multiple simulations, resulting in significant reductions in processing time [32]. Additionally, the platform has a multi-threaded structure that includes master and slave threads. The master thread assigns the task of updating a single plug-in, such as a sensor, actuator, or physics engine, to the slave thread, which effectively employs multi-core processors to significantly improve computing performance, particularly for multi-robot or swarm simulations. ARGoS can support the simulation of thousands of robots simultaneously, saving 40% of time compared to the two-dimensional dynamic verification of 10,000 robots [32].

The robot library comprised ARGoS relatively limited and includes only a few simple models such as e-puck, eye-bot, Kilobot, marXbot, and spiri robots. These robots can be programmed using LUA script or C++, but mesh importing is not supported, and the object representation is encoded with OpenGL [34]. While ARGoS does provide an official technical support forum and manuals for developers, the update frequency is not very high, and the development tutorial resources are not extensive.

In summary, ARGoS is a powerful simulation platform that is particularly well-suited for simulating robot swarms, especially when large numbers of robot models need to be generated and involved. It has been widely adopted for simulating swarm foraging robots due to its ability to simulate large-scale swarms of robots quickly [35]. This feature is also applicable to simulating UAVs swarms, where ARGoS is better suited for simulating large-scale swarms of UAVs compared to high-precision simulation of individual UAVs. However, ARGoS still has room for improvement. Its robot library needs to be enriched, and its poor cross-platform performance cannot be ignored. Additionally, 3D mesh models are currently not supported, which limits its use in certain applications. Despite these limitations, nonetheless, ARGoS remains a valuable tool for researchers in swarm robotics and related fields.

5. MRDS

The full name of MRDS is Microsoft Robotics Developer Studio, which is a 3D robot simulation platform designed and developed by Microsoft [36]. **Figure 5** shows an outdoor scene simulated by MRDS. In addition to CPU, MRDS can also leverage other independent floating-point processors such as GPU and PPU for computation, making it capable of completing intensive computing-based simulations [37]. The platform is based on Physx Engine, which is commonly used on various simulation platforms for modeling rigid bodies and liquids [38][39][40].

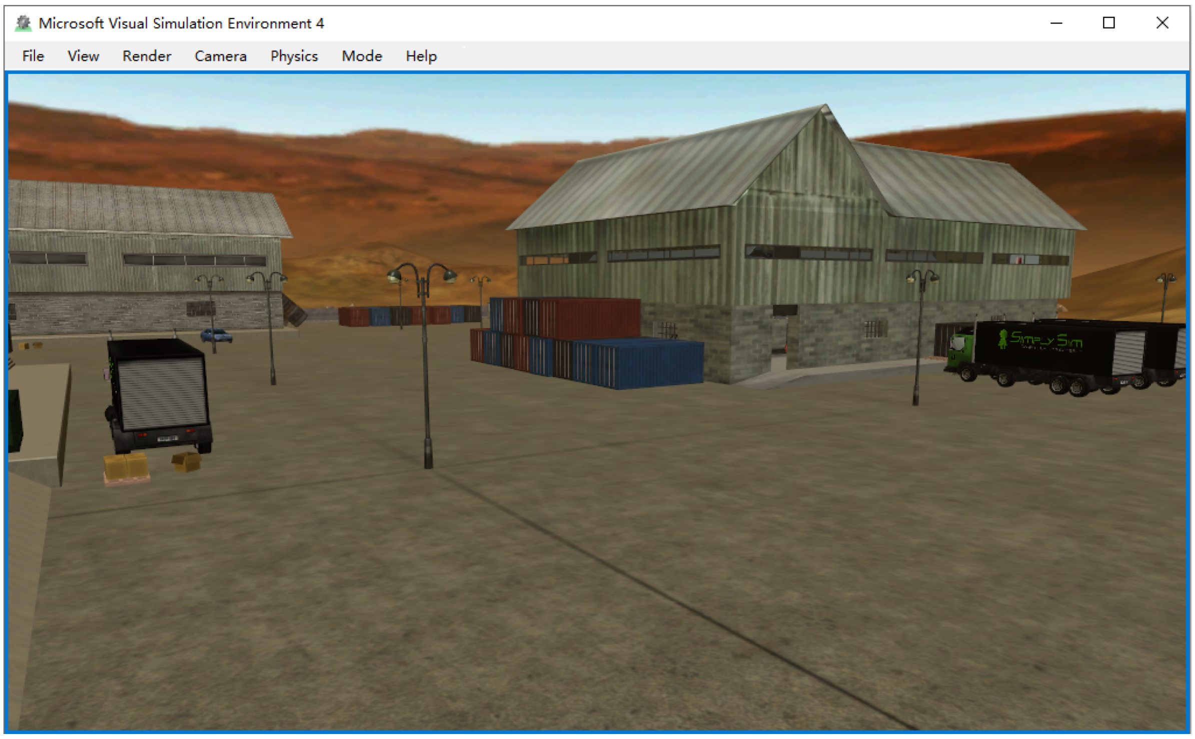


Figure 5. An outdoor scene simulated by MRDS.

MRDS can only be used on the Windows operating system, and the applications or algorithms for MRDS can be developed by VPL, C#, Visual Basic, Jscript, or Python [2]. Although MRDS is not an open-source platform, it can be used free of charge for non-commercial purposes. It uses decentralized software services (DSS) and concurrency and coordination runtime (CRR) to improve the performance of multi-line programming capabilities and to reuse the multi-core capabilities of processors [41]. This means that each service component can operate independently without synchronizing, allowing visual and acoustic services to be delivered independently without interruption. Even if one of the sensors fails, other functions will continue to operate normally. Unfortunately, this platform only provides limited robot models, such as iRobot, Lego NXT, and MobileRobotics, and it cannot support path planning [42]. The tutorials and development resources for MRDS are also very limited. Furthermore, the platform has not been updated for a long time, and its maintenance is becoming more and more difficult.

To be brief, the Physx Engine integrated into MRDS provides significant advantages to this platform, particularly in simulating harsh environments, such as sand. Additionally, the decentralized framework enhances its robustness and fault tolerance; even if one sensor fails, other robot functions can continue to operate normally. Users can leverage the Visual Simulation Environment (VSE) tool in MRDS to create and modify UAV models to suit their needs. Within VSE, users can design and adjust the physical entities, sensors, behaviors, and more of the UAV before controlling them with MRDS services. However, the platform has shortcomings such as limited built-in models, variable cross-platform performance, and maintenance challenges.

6. MORSE

MORSE (Modular Open Robots Simulation Engine) is a simulation platform developed based on Blender engine [13], which offers a realistic graphics environment that can be used for testing and evaluating robots without any further modifications in real-life scenarios [43]. The simulation engine utilizes the Bullet library to define physical properties such as object shapes, mass, friction, boundary collision, and other physical factors. This platform provides three types of robot components, namely, sensors, actuators, and robots. Robots carry two components: sensors and actuators. The former is used for data collection, while the latter is responsible for executing instructions. In addition, there are three components related to environment interaction, including scenes, middlewares, and modifiers. Scenes are simulated environments such as buildings, indoor environments, and jungles. Middlewares are responsible for binding sensors and actuators. As the data generated by sensors in simulation do not contain any noise, the modifier is used to simulate the noise data to achieve an effect that is much closer to the real environment.

MORSE can be deployed on Linux, Windows, and Mac OS, but with a better support for Linux. This platform generates simulation scenes using a set of Python classes called Builder API, which provides a special internal language called domain-specific language (DSL) [44]. This allows developers to use Python to configure MORSE, even if they are not familiar with Blender. The built-in models contain agents such as iRobot ATRV unmanned ground vehicle (UGV) and Yamaha RMAX UAV, which can be enriched by importing Collada, DXF, 3DS Max, VRML, and other formats. MORSE also has built-in sensors such as cameras, gyroscopes, GPS, accelerometers, thermometers, and lasers [43], making it suitable for simulating realistic graphics. MORSE has unique advantages in human-computer interaction and UAVs attitude simulation. It provides UAV models such as multi-copter UAV in the model library. Users can define parameters such as the number of UAVs, their positions, sensors, controllers, etc. by editing Python scripts. The platform also provides a variety of simulation scenarios, such as the outdoor (environment), which meets the needs of UAV simulation, and users can also model the environment based on it. Researchers use MORSE to simulate UAVs, such as multi-UAV communication [45] and path planning [46][47]. The simulation of an UAVs swarm using MORSE is shown in **Figure 6**.

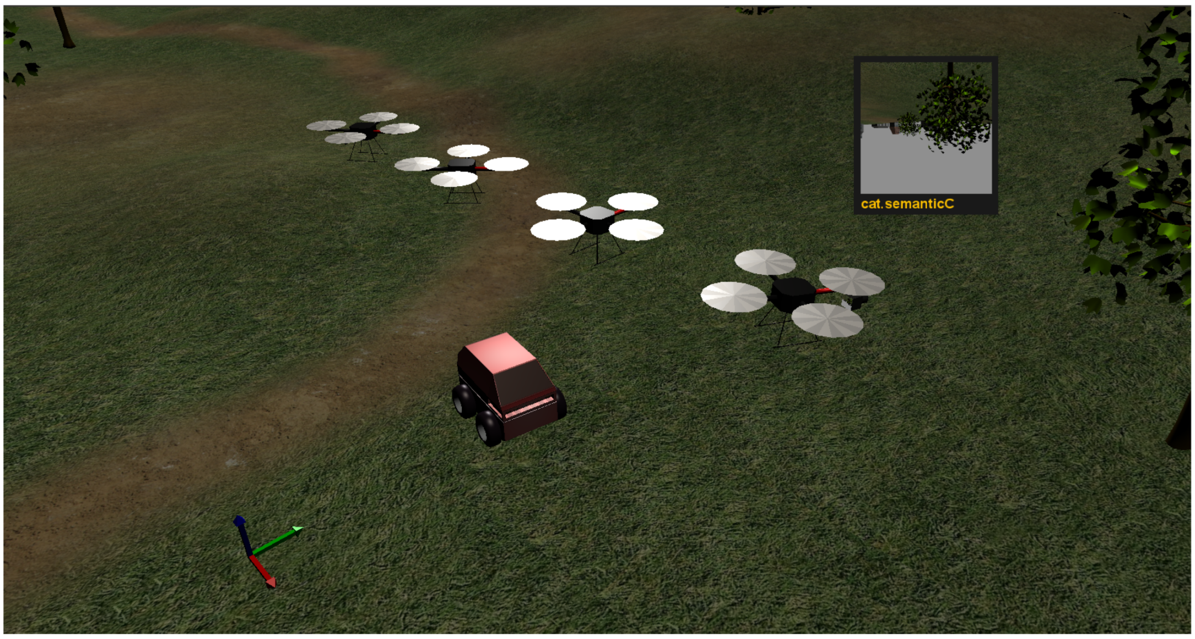


Figure 6. An UAVs swarm simulated by MORSE.

Unfortunately, the lack of updates since 2016 and limited community resources and development tutorials make it challenging for developers to use MORSE effectively. Furthermore, the absence of a graphical user interface and the requirement for developers to be familiar with command line tools and the Blender engine can be a barrier to entry for beginners. Additionally, MORSE lacks built-in algorithms such as path planning, which can make it challenging for developers to use it for more complex projects. Overall, while MORSE is not a good choice for entry-level developers due to its technical requirements and limited support resources.

7. USARSim

USARSim is a simulation platform developed by the National Institute of Standards and Technology in the USA for researching rescue robots and intelligence agents [48]. The platform is based on the Unreal Engine 2.0 developed by Epic Games [49], which provides high-quality 3D rendering capabilities using the PhysX physics engine. Applications and algorithms can be developed using C, C++, Java [2], or the built-in Unreal Script language. Control codes are developed using the GameBot interface, which is a communication interface exclusive to the phantom engine. USARSim is compatible with Windows, Linux, and Mac OS. An example of USARSim simulating UAVs can be seen in **Figure 7**.

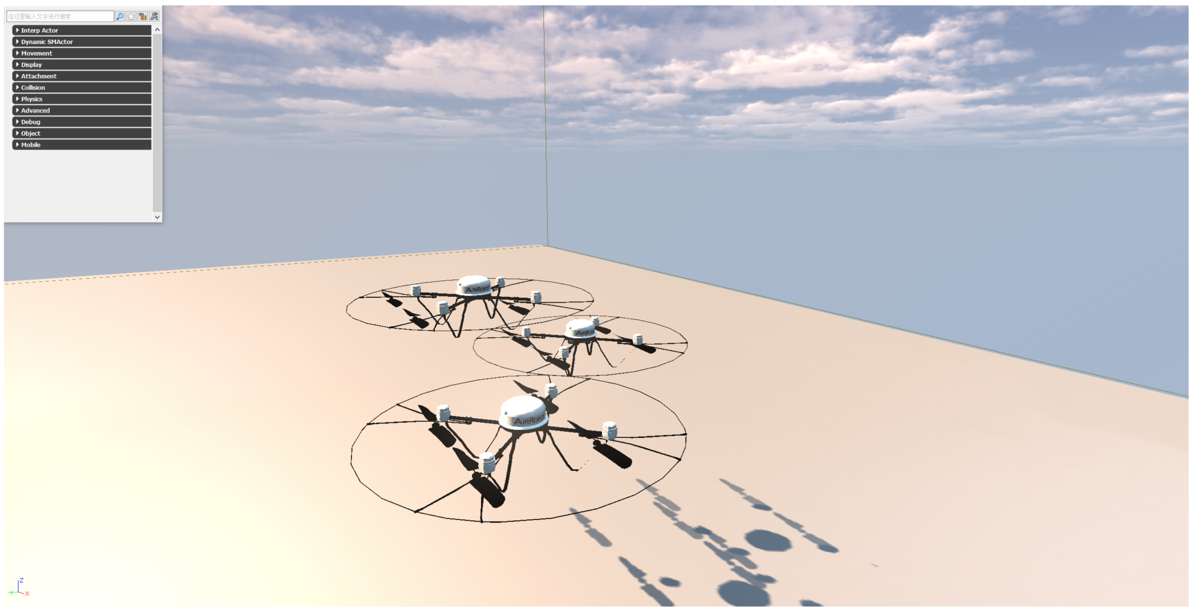


Figure 7. Simulation of UAVs by USARSim.

The Mobility Open Architecture Simulation and Tools (MOAST) framework of USARSim offers a range of hierarchical and modular controllers, interfaces, and tools that provide motion generation, world modeling, and sensor information processing [50]. In addition, USARSim includes two controllers, Pyro and Player [51]. Pyro is a controller for AI exploration and robot development, which includes Python libraries, environments, a GUI, and drivers. Player is a robot service that can control the robot and sensors, such as touch sensors, lasers, and acoustics, in the simulation environment. Users can easily and fully control the controller and actuator carried by a robot using Player.

USARSim supports eight types of ground robots, including P2AT, P2DX, ATRV-Jr, Zerg, Tarantula, Talon, Telemax, and Soryu; four types of vehicles, namely, Hummer, Sedan, SnowStorm, and Cooper; two bipedal robots, QRIO and ERS, a helicopter, and a submarine. As USARSim is developed by using Unreal Engine 2.0, it has high-quality 3D capabilities. Furthermore, it supports multiple types of UAVs, including multi-copter UAV, as well as different sensors and controllers. It provides predefined scenarios such as buildings, earthquake zones, and tunnels, while also allowing users to create custom scenarios. Users can write their algorithms to control the UAV. Although USARSim was primarily designed for search and rescue missions, it has also been used by many researchers for UAV simulation, such as obstacle avoidance and navigation [52][53]. However, the robot library is limited, and similar to MRDS, this platform has not been updated or maintained, resulting in poor support for new types of robots and sensors. While there are some official communities and manuals, community activity is weak, and tutorials are inadequate [13].

References

1. Michel, O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. Int. J. Adv. Robot. Syst. 2004, 1, 5.
2. Staranowicz, A.; Mariottini, G.L. A survey and comparison of commercial and open-source robotic simulator software. In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, New York, NY, USA, 25–27 May 2011; pp. 1–8.
3. Guo, W.; Gao, Y.; Wang, Y. Design and realization of the interactive virtual laboratory based on VRML. In Proceedings of the 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China, 21–23 April 2012; pp. 2510–2513.
4. Collins, J.; Chand, S.; Vanderkop, A.; Howard, D. A Review of Physics Simulators for Robotic Applications. IEEE Access 2021, 9, 51416–51431.
5. Erez, T.; Tassa, Y.; Todorov, E. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4397–4404.
6. ArduPilot Dev Team. SITL with Webots. Available online: https://ardupilot.org/dev/docs/sitl-with-webots.html (accessed on 12 March 2023).
7. Vajta, L.; Juhasz, T. 3D Simulation in the advanced robotic design, test and control. Int. J. Simul. Model. 2015, 4, 105–117.

8. Singh, A.; Jha, S.S. Learning safe cooperative policies in autonomous multi-uav navigation. In Proceedings of the 2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India, 19–21 December 2021; pp. 1–6.
9. Rivera, Z.B.; De Simone, M.C.; Guida, D. Unmanned ground vehicle modelling in Gazebo/ROS-based environments. Machines 2019, 7, 42.
10. Lei, Z.; Hao, L.; Yu-fei, L.; Shuai, Z. Study on Simulation Optimization of Gazebo Based on Asynchronous Mechanism. Comput. Sci. 2020, 47, 593–598.
11. Kumar, A.S.; Manikutty, G.; Bhavani, R.R.; Couceiro, M.S. Search and rescue operations using robotic darwinian particle swarm optimization. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1839–1843.
12. He, H.; Zheng, J.; Sun, Q.; Li, Z. Simulation of realistic particles with bullet physics engine. In E3S Web of Conferences; EDP Sciences: Les Ulis, France, 2019; Volume 92, p. 14004.
13. Cook, D.; Vardy, A.; Lewis, R. A survey of AUV and robot simulators for multi-vehicle operations. In Proceedings of the 2014 IEEE/OES Autonomous Underwater Vehicles (AUV), Oxford, MS, USA, 6–9 October 2014; pp. 1–8.
14. Lee, J.; Grey, M.X.; Ha, S.; Kunz, T.; Jain, S.; Ye, Y.; Srinivasa, S.S.; Stilman, M.; Liu, C.K. Dart: Dynamic animation and robotics toolkit. J. Open Source Softw. 2018, 3, 500.
15. Mingo Hoffman, E.; Traversaro, S.; Rocchi, A.; Ferrati, M.; Settimi, A.; Romano, F.; Natale, L.; Bicchi, A.; Nori, F.; Tsagarakis, N.G. Yarp based plugins for gazebo simulator. In Modelling and Simulation for Autonomous Systems. MESAS 2014; Hodicky, J., Ed.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8906, pp. 371–382.
16. Pitonakova, L.; Giuliani, M.; Pipe, A.; Winfield, A. Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. In Proceedings of the Annual Conference Towards Autonomous Robotic Systems, Bristol, UK, 22 July 2018; pp. 357–368.
17. Comparative analysis between gazebo and v-rep robotic simulators. Semin. Interno Cognicao Artif.-SICA 2014, 2014, 2.
18. Castillo-Pizarro, P.; Arredondo, T.V.; Torres-Torriti, M. Introductory survey to open-source mobile robot simulation software. In Proceedings of the 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting, Sao Bernardo do Campo, Brazil, 23–28 October 2010; pp. 150–155.
19. Bernardeschi, C.; Fagiolini, A.; Palmieri, M.; Scrima, G.; Sofia, F. Ros/gazebo based simulation of co-operative uavs. In Proceedings of the Modelling and Simulation for Autonomous Systems: 5th International Conference, MESAS 2018, Prague, Czech Republic, 17–19 October 2018; pp. 321–334.

20. Nguyen, K.D.; Nguyen, T.T. Vision-based software-in-the-loop-simulation for Unmanned Aerial Vehicles using gazebo and PX4 open source. In Proceedings of the 2019 International Conference on System Science and Engineering (ICSSE), Dong Hoi, Vietnam, 20–21 July 2019; pp. 429–432.
21. Rohmer, E.; Singh, S.P.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1321–1326.
22. Hummel, J.; Wolff, R.; Stein, T.; Gerndt, A.; Kuhlen, T. An evaluation of open source physics engines for use in virtual reality assembly simulations. In Proceedings of the International Symposium on Visual Computing, San Diego, CA, USA, 3–5 October 2012; pp. 346–357.
23. Tursynbek, I.; Shintemirov, A. Modeling and simulation of spherical parallel manipulators in CoppeliaSim (V-REP) robot simulator software. In Proceedings of the 2020 International Conference Nonlinearity, Information and Robotics (NIR), Innopolis, Russia, 3–6 December 2020; pp. 1–6.
24. Obdržálek, Z. Mobile agents and their use in a group of cooperating autonomous robots. In Proceedings of the 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 28–31 August 2017; pp. 125–130.
25. Freese, M.; Singh, S.; Ozaki, F.; Matsuhira, N. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. In Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Darmstadt, Germany, 15–18 November 2010; pp. 51–62.
26. Coppelia Robotics, L. Kinematics. Available online: https://www.coppeliarobotics.com/helpFiles/en/kinematics.htm (accessed on 22 March 2022).
27. Coppelia Robotics, L. Proximity Sensor Types and Mode of Operation. Available online: https://www.coppeliarobotics.com/helpFiles/index.html (accessed on 22 March 2022).
28. Coppelia Robotics, L. Vision Sensor Types and Mode of Operation. Available online: https://www.coppeliarobotics.com/helpFiles/index.html (accessed on 22 March 2022).
29. Wang, C.; Shi, Z.; Gu, M.; Luo, W.; Zhu, X.; Fan, Z. Revolutionary entrapment model of uniformly distributed swarm robots in morphogenetic formation. Def. Technol. 2022.
30. Virbora, N.; Sokoeun, U.; Saran, M.; Channareth, S.; Saravuth, S. Implementation of Matrix Drone Show Using Automatic Path Generator with DJI Tello Drones. In Proceedings of the 2022 International Conference on Engineering and Emerging Technologies (ICEET), Kuala Lumpur, Malaysia, 27–28 October 2022; pp. 1–5.
31. Huang, K.; Chen, J.; Oyekan, J. Decentralised aerial swarm for adaptive and energy efficient transport of unknown loads. Swarm Evol. Comput. 2021, 67, 100957.

32. Pinciroli, C.; Trianni, V.; O'Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.; Ducatelle, F.; et al. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell. 2012, 6, 271–295.
33. Pinciroli, C. ARGoS Core. Available online: https://www.argos-sim.info/core.php (accessed on 22 March 2022).
34. Allwright, M.; Bhalla, N.; Pinciroli, C.; Dorigo, M. Simulating multi-robot construction in ARGoS. In Proceedings of the International Conference on Swarm Intelligence, Shanghai, China, 17–22 June 2018; pp. 188–200.
35. Lu, Q.; Fricke, G.M.; Ericksen, J.C.; Moses, M.E. Swarm foraging review: Closing the gap between proof and practice. Curr. Robot. Rep. 2020, 1, 215–225.
36. Jackson, J. Microsoft robotics studio: A technical introduction. IEEE Robot. Autom. Mag. 2007, 14, 82–87.
37. Matta-Gómez, A.; Del Cerro, J.; Barrientos, A. Multi-robot data mapping simulation by using microsoft robotics developer studio. Simul. Model. Pract. Theory 2014, 49, 305–319.
38. Glette, K.; Hovin, M. Evolution of artificial muscle-based robotic locomotion in PhysX. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 1114–1119.
39. Wang, D.; Zhang, L.; Wang, M.; Xiao, T.; Hou, Z.; Zou, F. A simulation system based on ogre and physx for flexible aircraft assembly. In Proceedings of the 2012 ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation, Zhangjiajie, China, 15–19 July 2012; pp. 171–173.
40. Gechter, F.; Contet, J.M.; Galland, S.; Lamotte, O.; Koukam, A. Virtual intelligent vehicle urban simulator: Application to vehicle platoon evaluation. Simul. Model. Pract. Theory 2012, 24, 103–114.
41. Cepeda, J.S.; Chaimowicz, L.; Soto, R. Exploring Microsoft Robotics Studio as a mechanism for service-oriented robotics. In Proceedings of the 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting, Sao Bernardo do Campo, Brazil, 23–28 October 2010; pp. 7–12.
42. Michal, D.S.; Etzkorn, L. A comparison of player/stage/gazebo and microsoft robotics developer studio. In Proceedings of the 49th Annual Southeast Regional Conference, Kennesaw, GA, USA, 24–26 March 2011; pp. 60–66.
43. Echeverria, G.; Lassabe, N.; Degroote, A.; Lemaignan, S. Modular open robots simulation engine: Morse. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 46–51.

44. Echeverria, G.; Lemaignan, S.; Degroote, A.; Lacroix, S.; Karg, M.; Koch, P.; Lesire, C.; Stinckwich, S. Simulating complex robotic scenarios with MORSE. In Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Tsukuba, Japan, 5–8 November 2012; pp. 197–208.
45. Casas, V.; Mitschele-Thiel, A. On the impact of communication delays on UAVs flocking behavior. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Barcelona, Spain, 15–18 April 2018; pp. 67–72.
46. Casas, V.; Mitschele-Thiel, A. From simulation to reality: A implementable self-organized collision avoidance algorithm for autonomous UAVs. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020; pp. 822–831.
47. Dias, A.; Fernandes, T.; Almeida, J.; Martins, A.; Silva, E. 3D path planning methods for unmanned aerial vehicles in search and rescue scenarios. In Human-Centric Robotics: Proceedings of the CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, Porto, Portugal, 11–13 September 2017; World Scientific: Singapore, 2018; pp. 213–220.
48. Zhibao, S.; Haojie, Z.; Sen, Z. A robotic simulation system combined USARSim and RCS library. In Proceedings of the 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Wuhan, China, 16–18 June 2017; pp. 240–243.
49. Balakirsky, S.; Scrapper, C.; Carpin, S.; Lewis, M. USARSim: Providing a framework for multi-robot performance evaluation. In Proceedings of the Performance Metrics for Intelligent Systems (PerMIS) Workshop, Gaithersburg, MD, USA, 14–16 August 2006; pp. 98–102.
50. Carpin, S.; Lewis, M.; Wang, J.; Balakirsky, S.; Scrapper, C. USARSim: A robot simulator for research and education. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Rome, Italy, 10–14 April 2007; pp. 1400–1405.
51. Lewis, M.; Wang, J.; Hughes, S. USARSim: Simulation for the study of human-robot interaction. J. Cogn. Eng. Decis. Mak. 2007, 1, 98–120.
52. Mendes, J.; Ventura, R. Safe teleoperation of a quadrotor using FastSLAM. In Proceedings of the 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), College Station, TX, USA, 5–8 November 2012; pp. 1–6.
53. Drews, S.; Lange, S.; Protzel, P. Validating an active stereo system using USARSim. In Proceedings of the Simulation, Modeling, and Programming for Autonomous Robots: Second International Conference, SIMPAR 2010, Darmstadt, Germany, 15–18 November 2010; pp. 387–398.