Comparative analysis of UAV simulation platforms: challenges, opportunities, and future directions

Andrey Kupin, Yuriy Osadchuk and Maksym Kosei

Kryvyi Rih National University, 11 Vitalii Matusevych Str., Kryvyi Rih, 50027, Ukraine

Abstract

Unmanned aerial vehicles (UAVs) are deployed across agriculture, disaster response, infrastructure inspection, and military operations, but flight testing is expensive, hazardous, and slow, which has made simulation the default environment for developing and validating UAV algorithms. Dozens of simulation platforms exist, and a substantial secondary literature already compares them; what is missing is a consolidated view across those comparisons. This paper reports a meta-review of nine studies published between 2018 and 2025: six comparative surveys of simulators and agent-based frameworks, two simulator-development papers, and one review of UAV airframes and autonomy components. For each study we reconstruct its scope, the platforms it covers, and the evaluation criteria it applies, and consolidate the results in a single comparison. The surveys converge on a common set of evaluation axes – sensor support, physics fidelity, scalability, usability, and maintenance activity – but weight them differently and share no common benchmark, so their verdicts cannot be compared across publications. Coverage is highly uneven: Gazebo is assessed in five of the nine studies and most other platforms in only one. We argue that the principal gap in UAV simulation is therefore not platform capability but the absence of a shared benchmarking protocol, and set out what such a protocol would have to specify.

Keywords

UAV, drone simulators, simulation platforms, meta-review, evaluation criteria, benchmarking, swarm robotics, sensor integration, scalability

1. Introduction

Unmanned aerial vehicles (UAVs), commonly referred to as drones, have become increasingly popular and diverse in recent years, with a wide variety of designs, configurations, and manufacturers catering to both civilian and military applications [1, 2]. The membership of the Association for Unmanned Vehicle Systems International (AUVSI) alone exceeds 400 companies, among them Boeing, Lockheed Martin, Northrop Grumman, General Atomics, and DJI, alongside other manufacturers in the defense, aerospace, and commercial drone sectors [3]. Their product lines run from small consumer quadcopters to large, high-endurance systems built for military and industrial missions, and are applied to remote sensing, transportation, search and rescue, and military operations, whether flown autonomously or under remote control.

UAVs come in a variety of configurations, each tailored to specific applications and operational requirements. The most common designs include fixed-wing UAVs, rotary-wing UAVs, VTOL (vertical takeoff and landing) UAVs, and HTOL (horizontal takeoff and landing) UAVs [4].

Fixed-wing UAVs resemble conventional airplanes and offer long endurance and range, which suits surveillance, mapping, and environmental monitoring. Rotary-wing UAVs – quadcopters, hexacopters, and octocopters – are highly maneuverable and can hover, which suits urban inspection, search and rescue, and payload delivery. VTOL UAVs combine the two: they take off and land vertically, then transition to efficient forward flight, which makes them usable in confined spaces or rugged terrain

CS&SE@SW 2025: 8th Workshop for Young Scientists in Computer Science & Software Engineering, November 21, 2025, Kryvyi Rih, Ukraine
https://doi.org/10.55056/cssesw2025/paper19
kupin.andrew@gmail.com (A. Kupin); u.osadchuk@knu.edu.ua (Y. Osadchuk); kosei@knu.edu.ua (M. Kosei)
https://www.knu.edu.ua/fakultety/fakul-tet-informaciynyh-tehnolohiy/struktura/kafedra-komp-yuternyh-system-ta-merezh/zaviduvach-kafedry (A. Kupin); http://aespt.knu.edu.ua/team/osadchuk/ (Y. Osadchuk)
0000-0001-7569-1721 (A. Kupin); 0000-0001-6110-9534 (Y. Osadchuk); 0009-0008-3445-5675 (M. Kosei)

where no runway is available. HTOL UAVs, by contrast, require conventional runways and are optimized for long-range, high-speed missions such as military reconnaissance and large-scale mapping, where range matters more than maneuverability.

The diversity of UAV configurations highlights the need for simulation platforms that can accurately model the unique dynamics and capabilities of each type. For instance, simulating the aerodynamics of fixed-wing UAVs requires precise modeling of lift and drag forces, while rotary-wing UAVs demand accurate representations of rotor dynamics and hovering stability. VTOL and HTOL UAVs introduce additional complexities, such as the transition between vertical and horizontal flight modes, which must be captured in high-fidelity simulation environments (figure 1) [5].

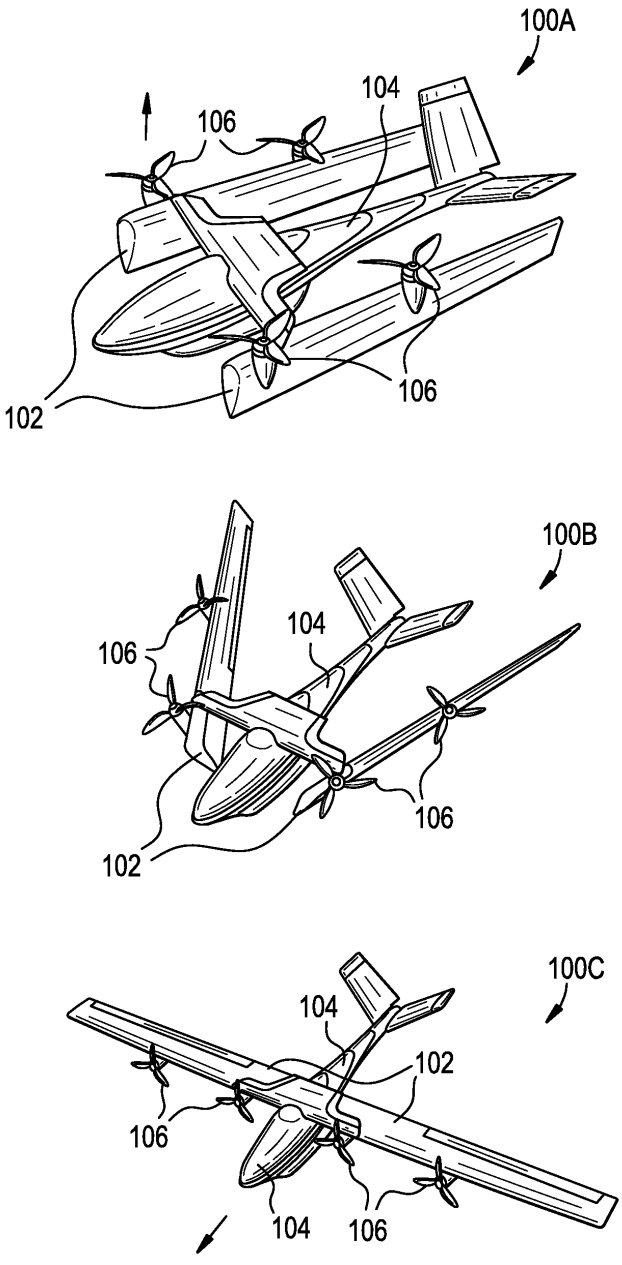

Figure 1: Schematic of the VTOL-to-cruise transition of the PteroDynamics Transwing airframe, from the manufacturer's U.S. patent [5].

A second demand on UAV simulation is the modeling of environmental conditions, above all weather. Wind, precipitation, and temperature affect flight stability, sensor accuracy, and energy consumption: turbulence destabilizes rotary-wing UAVs, strong crosswinds make it hard for fixed-wing UAVs to hold altitude and heading, rain and snow degrade cameras and LiDAR, and extreme temperatures shorten

battery life. A platform that omits these effects cannot establish the operating envelope of the system under test.

The sensor suite must be represented with the same care. Modern UAVs carry cameras, LiDAR, radar, GPS receivers, inertial measurement units (IMUs), ultrasonic rangefinders, and environmental sensors for temperature, humidity, and air pressure. Reproducing their nominal outputs is the easy part; the harder requirement is reproducing their degradation under the environmental conditions above, under electromagnetic interference, and under physical occlusion, since it is exactly that degradation which perception and navigation algorithms have to survive.

UAVs also operate in collaborative networks, particularly in swarm robotics and other multi-UAV systems, where vehicles exchange position, velocity, sensor readings, and task assignments in real time. Wi-Fi, radio frequency (RF) links, and ad-hoc mesh networks each impose their own limits on range, bandwidth, and interference tolerance, and are driven by protocols such as MAVLink, the Data Distribution Service (DDS), and swarm-specific alternatives; Hentati and Fourati [6] classify the routing protocols involved (figure 2).

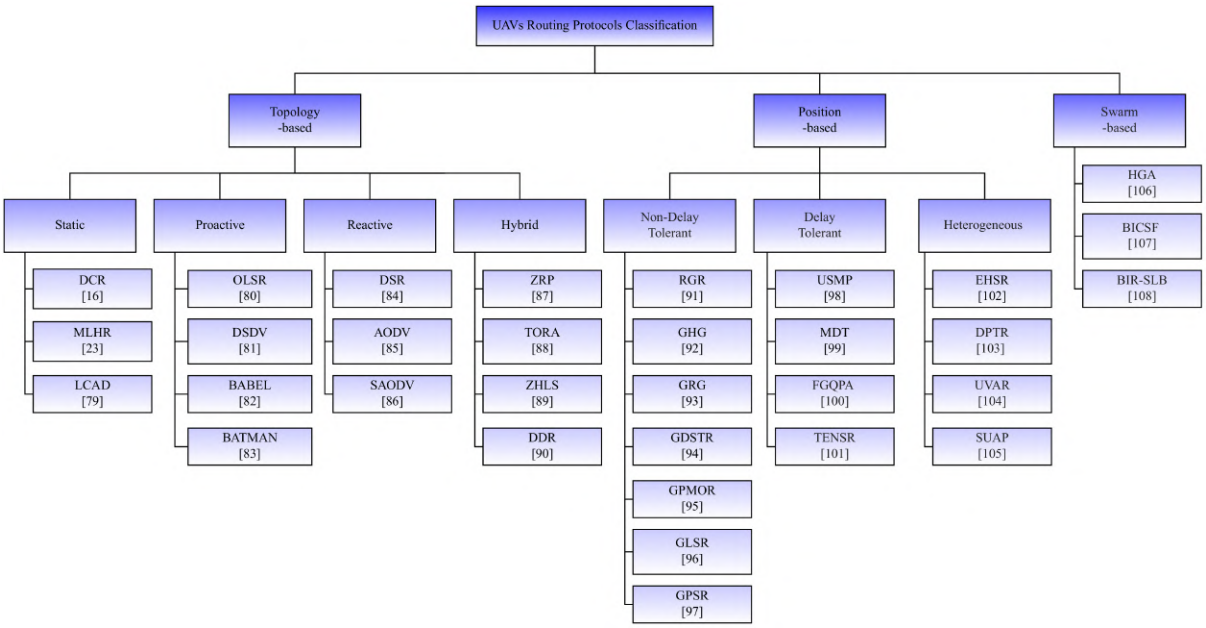

Figure 2: Classification of routing protocols for UAV networks [6].

Airframe dynamics, weather, sensors, and networking therefore all have to be present in a platform that is to serve as a realistic testbed. This is why simulation, rather than flight testing, has become the working environment for UAV development: physical trials are expensive, slow, and hazardous, and prohibitively so for swarm experiments, autonomous navigation in cluttered space, and any scenario in which failure destroys the vehicle. A simulator lets flight dynamics, sensor integration, decision-making, and environmental adaptability be exercised at no risk and at whatever scale the hardware allows.

The difficulty is choosing one. Simulators differ widely in physics accuracy, sensor modeling, scalability, and accessibility, and no single platform is strong on every axis: some offer high-fidelity dynamics but do not scale to swarms, others coordinate many vehicles but model sensors and environments only coarsely. This has produced a substantial secondary literature of surveys and comparative reviews, and with it a second-order problem – the surveys themselves adopt different criteria, cover overlapping but unequal sets of platforms, and reach verdicts that cannot readily be placed side by side.

This paper addresses that second-order problem. It reviews nine studies of UAV simulation published between 2018 and 2025, reconstructs the evaluation criteria each applies, and consolidates their scope and methodology in a single comparison (table 3). Section 2 presents the studies individually and then synthesizes them; section 3 states what the synthesis implies for the selection and future development of simulation platforms.

2. Drone simulation platforms: a literature review

The nine studies reviewed here [7, 8, 9, 10, 11, 12, 13, 14, 15] approach UAV simulation from different directions. Six are comparative surveys of simulators or agent-based frameworks; two report simulators built by their own authors and position them against existing tools; one reviews UAV airframes and autonomy components rather than simulation software, and is included because the component taxonomy it establishes is what simulators are expected to reproduce. Taken together they document a fragmented landscape in which each platform caters to particular vehicle types and applications, and in which the selection of a tool is a matter of trading simulation speed against physics fidelity, sensor realism, and accessibility. The sections below summarize each study in turn; section 2.1 then compares them.

Chen et al. [7] restrict their attention to open-source platforms, on the grounds that community-driven development, unrestricted customization, and the absence of licensing costs make them particularly suited to UAV swarm research. They assess seven such platforms – Webots [16], Gazebo [17], CoppeliaSim [18], ARGoS [19], MRDS [20], MORSE [21], and USARSim [22] – for their strengths, limitations, and suitability for multi-copter swarm simulation. Table 1 summarizes their main characteristics.

The platforms are evaluated against eight criteria:

- **Multi-sensor support**: Ability to simulate sensors like GPS, cameras, IMUs, and radars.
- **Built-in models**: Availability of pre-designed UAVs, robots, and environments.
- **Realism**: Accuracy of the physics engine in reproducing physical environments and aerodynamic effects.
- **Computation performance**: Efficiency in handling large-scale swarm simulations.
- **Programming language and OS compatibility**: Support for multiple languages and operating systems.
- **Usability**: Availability of documentation, tutorials, and user-friendly interfaces.
- **Stability and maintenance**: Frequency of updates and duration of support.
- **Commonly used functions**: Support for ROS, AI algorithms, and path planning.

Table 2 gives the resulting ratings. It reports nine rated columns rather than eight, because language support and operating-system support are scored separately there.

The overall scores are equal-weight sums of the symbolic ratings, converted as '+++' = 3, '++' = 2, '+' = 1, and '–' = 0 [7]; no criterion is weighted above the others.

On these scores the authors rank Webots, CoppeliaSim, and Gazebo as the three strongest platforms for multi-copter swarm simulation: Webots as the best-balanced option across functionality, usability, and accuracy; CoppeliaSim where high precision and extensive sensor integration are needed; and Gazebo where ROS (Robot Operating System) integration matters and computational demands are moderate.

Casado and Bermúdez [8] take a different starting point: rather than surveying platforms, they build one, aimed at engineering students learning to develop autonomous drone navigation systems (figure 3). Their framework abstracts away low-level control so that drone behavior can be defined graphically as a state machine.

They utilized Gazebo [17] for realistic drone modeling and low-level control, integrating a plugin that manages motor control and stabilization. Simulink [23] and Stateflow [24] were employed for implementing the autonomous navigation system, processing sensor data (such as an ideal IMU for drone position and orientation, and a built-in camera for vision-based object detection). Communication between Gazebo and Simulink is established via ROS topics, enabling the exchange of information like drone position and commanded velocities.

The framework was used in the "ESII Drone Challenge", a student competition run by the School of Computer Science and Engineering of the University of Castilla-La Mancha, in which quadcopters must be programmed to fly through colored frames and return to base. The authors position their framework against fuller open-source stacks such as Aerostack: the deliberate reduction in functionality is the

Table 1
Main characteristics of the seven simulation platforms [7].

Simulation platform	Programming languages	OS	Physics engines	Main sensors
Webots	C, C++, Java, Python, MATLAB	Windows, Linux, MAC OS	ODE	Accelerometer, altimeter, compass, GPS, gyroscope, distance sensor, inertial unit, position sensor, receiver, touch sensor, camera, LIDAR, radar, rangefinder
Gazebo	C, C++, Python	Mac OS, Linux, Windows	ODE, Bullet, Simbody, DART	Laser sensor, camera, inertial measurement, contact sensor, force sensor, torque sensor
CoppeliaSim	C, C++, Python, Java, LUA, Matlab, Octave	Windows, Mac OS, Linux	ODE, Bullet, Vortex, Newton	Proximity sensors, force sensors, vision sensors, cameras
ARGoS	Lua, C++	Ubuntu, KUbuntu, OpenSuse, Mac OS	ODE, 3D particle engine, 2D-dynamics open-source physics engine library Chipmunk, 2D-kinematics engine	—
MRDS	VPL, C#, Visual Basic, JavaScript, Iron-Python	Windows	PhysX engine	Analog sensor, array, GPS, contact sensors, depth camera, encoder, sonar, webcam, SICK laser range finder
MORSE	Python	Linux, Windows, Mac OS	Blender engine, Bullet	Accelerometer, airspeed, armature pose sensor, barometer, attitude sensor, collision, battery sensor, depth camera, generic camera, gps, gyroscope, infrared proximity sensor, laser scanner sensors, magnetometer, odometry, inertial measurement unit, radar altimeter, thermometer sensor, proximity sensor, velocity
USARSim	C, C++, Java	Linux, Windows, Mac OS	Unreal engine	State sensor, range sensor, range scanner sensor, odometry sensor, GPS, INS, encoder sensor, touch sensor, RFID, sound sensor, human-motion sensor, robot camera, omnidirectional camera

Table 2
Overall comparisons of the seven simulation platforms [7].

Simulation platform	Multi-sensors	Built-in model library	Physics engines	Computation performance and accuracy	Programming language	Cross-platform	Usability	Being updated	ROS and AI algorithms	Overall scores
Webots	++	++	+	+++	+	+	+	+	+	13
Gazebo	++	+	+	+	+	+	+	+	+	10
CoppeliaSim	++	++	+	++	+	+	+	+	+	12
ARGoS	−	−	+	+	−	−	−	+	+	4
MRDS	+	−	+	−	+	−	−	−	−	3
MORSE	+	+	+	+	−	+	−	−	+	6
USARSim	−	−	+	+	+	+	−	−	+	5

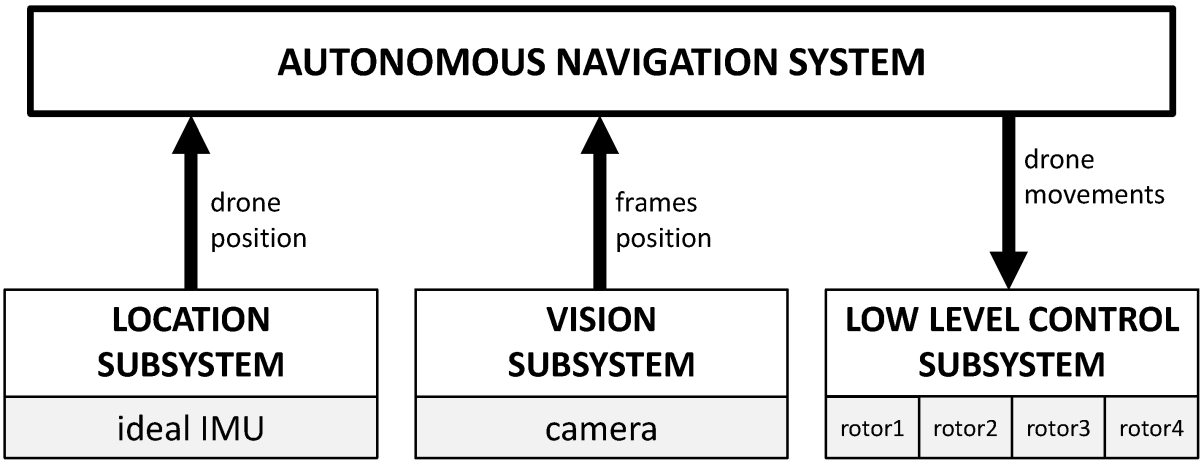

Figure 3: Dependencies between the subsystems composing the framework [8].

point, since the target users are new to aerial robotics. Educational accessibility is thus treated as a design criterion in its own right, and it is one that none of the criteria sets used by the comparative surveys discussed below captures directly.

Herich and Vaščák [9] likewise present a simulator of their own, this one for drone swarms, built to model and optimize cooperative behavior under realistic physics, communication constraints, and decentralized decision-making.

They frame swarm simulation around four difficulties: maintaining communication between members, designing decentralized decision-making, building fault-tolerant vehicles, and capturing the dynamics of a swarm whose collective behavior emerges from individual interactions. Their system responds with realistic inter-drone physics, real-time control and monitoring, and a testbed for control algorithms and communication strategies under varying conditions.

Technically, the system is built on Blender [25] for 3D modeling, texturing, and animation, which supplies both the vehicles and dynamic laboratory environments containing moving obstacles. Control logic is written in Python, with a custom low-latency interface over Python sockets, while the Blender Game Engine applies gravity, momentum, and aerodynamic drag and resolves collisions.

They contrast their system with SwarmLab [26, 27], which offers comparable simulation features

inside MATLAB: the Blender-plus-Python combination buys richer real-time interaction and more responsive environmental dynamics. The system scales from small groups to swarms of several hundred vehicles, using polygon reduction and Level of Detail (LOD) rendering to hold real-time performance, and supports reproducibility through scenario logging and replay.

The authors identify urban cityscape environments, reinforcement-learning-based control, and scaling to larger swarms without losing real-time responsiveness as future work, and single out latency under AI-driven control as the binding constraint.

The broadest of the surveys is that of Dimmig et al. [10], who catalog 44 UAV simulators and compare 14 of them in depth. Their concern is the selection problem itself: which simulator fits a given use case, given its limitations and how readily it can be customized.

To that end they establish nine decision factors:

- **Physics fidelity**: The required accuracy of physics and dynamics models for the intended use case.
- **Visual fidelity**: The level of realism in generated images, particularly for computer vision or machine learning applications.
- **Autopilot compatibility**: Support for common autopilots like PX4 and ArduPilot, crucial for software-in-the-loop (SITL) and hardware-in-the-loop (HITL) testing.
- **Multiple vehicles and heterogeneity**: The capability to concurrently simulate multiple vehicles and integrate with other platforms.
- **Sensors**: Support for common sensors such as cameras, IMU, GPS, and LiDAR.
- **UAV models**: Ease of integrating new UAV models and support for common types.
- **Simulation speed**: Ability to run in real-time or super real-time, important for learning applications.
- **APIs**: Compatibility with various programming languages, middleware like ROS, and packages such as OpenAI Gym.
- **Integration and ease of use**: Factors like ease of getting started, development, license type, and maintenance status.

They then group the simulators into seven classes by primary focus:

- **Universal simulators**: Platforms like Gazebo Classic, the newer Gazebo, Isaac Sim/Gym, Webots, CoppeliaSim, and MuJoCo, designed to simulate a wide variety of platforms including ground and aerial vehicles.
- **Sensor-focused simulators**: Examples include AirSim, Flightmare, FlightGoggles, and FastSim, which prioritize realistic sensor performance (e.g., photorealism or accurate LiDAR measurements).
- **Learning-focused simulators**: These include Isaac Gym (Aerial Gym), MuJoCo, gym-pybullet-drones, and PyFlyt, optimized for compatibility with Machine Learning architectures for training behaviors or learning dynamics models.
- **Dynamics-focused simulators**: Such as RotorTM, MATLAB UAV Toolbox, and PyFly, which emphasize thorough dynamics models for specific systems like aerial manipulation or fixed-wings.
- **Swarming simulators**: Platforms like gym-pybullet-drones, QuadSwarm, and Potato, specifically designed for simulating groups of vehicles.
- **Part of flight stacks**: Simulators integrated directly with flight control stacks, including Agilicious, MRS UAV System, and Aerostack2.
- **Flight simulators**: General flight emulators like X-Plane and FlightGear, some of which have been adapted for robotics applications due to their realistic flight experience.

Hentati et al. [11] narrow the question to system performance analysis, identifying the simulation tools, environments, and frameworks best suited to it and setting out the requirements, goals, strengths, and weaknesses of each.

They treat a UAV system as three parts – the vehicle, a ground control station (GCS), and the UAV-UAV and UAV-GCS communication links – and argue for SITL simulation as the practical way to exercise autopilot code without risking hardware. Their classification of flight simulators turns on Motion Capture (MOCAP) support. Earlier simulators like X-Plane, FlightGear, Gazebo, and JMavSim often focused on simulating UAV physics for pilot training but typically lacked MOCAP support and used limited assets. More recently, simulators such as Microsoft AirSim and UE4Sim, built on Unreal Engine 4, have emerged to provide photo-realistic simulations with MOCAP support, enabling realistic UAV motion planning.

The simulators they analyze are:

- **X-Plane**: A commercial simulator known for its realistic aerodynamics and FAA certification. It uses a geometric shape-based engineering mechanism to calculate forces on UAVs and supports UDP for communication.
- **FlightGear**: A free, open-source, and multi-platform flight simulator. It allows for multi-aircraft environments suitable for air traffic control or formation flight simulation and supports various Flight Dynamics Models (FDMs). FlightGear can run in SITL and Hardware-In-The-Loop (HITL) modes and is often used in academic projects.
- **JMavSim**: A lightweight multirotor simulator developed by PX4. It supports SITL and HITL, integrates with PX4 firmware via a MAVLink interface, and can be used with ROS. However, it has limited capabilities for integrating extra sensors or obstacles.
- **Gazebo**: A versatile simulator widely used with ROS, supporting various robots beyond just quadrotors. It offers flexibility with multiple physics engines (ODE, Simbody, DART, Bullet) and facilitates easy creation of worlds and models.
- **Microsoft AirSim**: Developed for machine learning, this simulator leverages Unreal Engine 4 for high visual fidelity and supports MAVLink, SITL, and HITL with Pixhawk firmware. It provides Python and C++ APIs but primarily supports quadrotors and requires powerful GPUs due to its demanding visuals.
- **UE4Sim**: Also built on Unreal Engine 4, UE4Sim generates photo-realistic simulations for autonomous cars and UAVs. It is used in computer vision applications such as object tracking, detection, and autonomous navigation, providing access to both visual and semantic data.

The paper also analyzes Ground Control Station (GCS) software, which handles mission planning, navigation control, payload management, and communication with the vehicle:

- **QGroundControl**: An open-source, graphical GCS compatible with multiple platforms. It supports PX4 Pro and ArduPilot firmware, the MAVLink protocol, and offers features like 2D map display, video streaming, and mission creation for autonomous flights.
- **Mission planner (MP)**: A Python-based graphical GCS primarily for Windows. It displays UAV information, and allows downloading and examining mission log files.
- **Universal ground control software (UGCS)**: A desktop software with a graphical interface that supports various autopilots and MAVLink-compatible drones. A free version with limited capabilities is available for beginners.
- **MAVProxy**: An extensible, command-line GCS written in Python, compatible with multiple operating systems. It manages MAVLink-supported UAVs and is commonly used by developers for SITL interaction and even for analyzing MAVLink protocol vulnerabilities.

Hentati et al. [11] conclude their paper with a case study demonstrating the exchange of MAVLink messages between FlightGear as the UAV simulator and QGroundControl as the GCS, highlighting their suitability for analyzing UAV system performance and communication protocols.

Tahir et al. [12] stand apart from the other studies reviewed here in that their subject is UAV platforms rather than simulation software. They are included because the review defines what an autonomous UAV consists of, and therefore what a simulator has to reproduce. The stated gap they address is the scarcity of near-implementation detail in the literature on autonomy strategies.

The review is organized around five aspects of autonomous operation:

1. **UAV classification**: The configurations introduced in section 1, extended with flapping-wing and hybrid designs, each assessed for its fitness for autonomous missions.
2. **Key components for autonomy**: The hardware and software elements that make autonomy possible:
    - **Sensors**: The sensor suite listed in section 1, treated by function – GPS for localization, IMUs for attitude and velocity, LiDAR for 3D mapping and obstacle avoidance, monocular, stereo, and depth cameras for vision-based navigation and object recognition, and ultrasonic and thermal sensors for specific environmental interactions.
    - **Processing units**: The onboard compute required for real-time data processing and decision-making.
    - **Communication systems**: The technologies and protocols carrying data between UAVs and ground control stations and between swarm members, including Wi-Fi, 4G and 5G cellular links, satellite communication, and MAVLink.
3. **Control algorithms**: The strategies available for stable and precise autonomous flight, from PID controllers, linear quadratic regulators (LQR), and model predictive control (MPC) to adaptive control, fuzzy logic, and neural network-based approaches.
4. **Navigation and path planning**: Simultaneous localization and mapping (SLAM), waypoint navigation, global and local path planning, and dynamic obstacle avoidance, on which independent operation in cluttered environments depends.
5. **Perception systems**: The fusion of data from multiple sensors into environmental models usable for autonomous decision-making.

The review closes on open problems: deeper integration of machine learning into decision-making and adaptive control, swarm intelligence for collaborative missions, human-UAV interaction, energy efficiency, cybersecurity, and the unsettled regulatory position of autonomous flight.

Mairaj et al. [13] review application-specific simulators, on the argument that general-purpose modeling is insufficient once applications diverge. Their particular concern is the UAV network (UAVNet), whose characteristics differ from those of other ad-hoc networks and which therefore needs dedicated modeling. Evaluating a design, on their account, means evaluating the vehicle's working mechanism, the geographical and weather conditions, and UAVNet communication together.

They categorize drone simulators into two main types:

1. **Generic simulators**: These are general-purpose platforms like Gazebo, AirSim, ArduPilot, V-Rep, X-Plane, and FlightGear, designed to simulate a broad range of UAV behaviors and environments.
2. **Application-specific simulators**: These are specialized tools tailored for particular scenarios or research objectives, and they are further subdivided:
    - **UAV network specific simulators**: Platforms such as NS-3, OMNeT++, QualNet, and NetSim, which primarily focus on modeling and evaluating network communication protocols and performance within UAV networks.
    - **Environmental specific simulators**: Including tools like AirSim, RotorS (often used with Gazebo), MATLAB UAV Toolbox, Quad-Sim, and ArduPilot, designed to simulate realistic environmental factors, including complex terrains, weather conditions, and physical interactions.
    - **Control algorithm specific simulators**: Such as Gym-PyBullet-Drones, Sim-Mecanum, and JSBSim, which are optimized for testing and validating specific control algorithms, often in the context of artificial intelligence (AI) and machine learning (ML) applications.
    - **Power specific simulators**: An example being Simu-drone, which focuses on the analysis and management of power consumption for extended flight durations.
    - **Swarm specific simulators**: Including platforms like SwarmLab and Multi-Agent Drone Simulator, dedicated to modeling and optimizing the coordinated behaviors of multiple UAVs in swarm configurations.

Nikolaiev and Novotarskyi [14] survey current UAV simulators, set out the factors that should govern selection, and argue that the field must balance diversity against standardization. Their case for simulation rests on two grounds: the risk of real-world testing, where unpredictable behavior costs money, time, and equipment; and the data requirements of machine learning for UAV operations, which cannot practically be met through hardware trials.

Their central hypothesis is the one this paper takes up. A more standardized approach to UAV simulation, they argue, would yield comparability between studies, easier collaboration, and a unified framework for benchmarking simulator performance; they add that better aerodynamics modeling, particularly for multirotor and fixed-wing vehicles, is needed for simulation accuracy and realism.

Mualla et al. [15] address civil aerial transportation and contribute what none of the other studies offers: an externally grounded comparison method. Their evaluation of agent-based simulation frameworks derives its criteria from the ISO software quality model and combines them through a weighted sum, so that the weights are declared rather than implicit. The criteria cover general agent-based simulation features and are then adapted to the demands of unmanned aerial transportation.

2.1. Synthesis across the reviewed surveys

Table 3 consolidates the nine studies along the three dimensions that determine whether their results can be read together: what each one set out to cover, which platforms it actually assessed, and on what basis it judged them.

Table 3
Scope, coverage, and evaluation basis of the nine reviewed studies, in order of presentation.

Study	Year	Scope	Platforms assessed	Evaluation basis
Chen et al. [7]	2023	Open-source platforms for multi-copter swarms	7: Webots, Gazebo, CoppeliaSim, ARGoS, MRDS, MORSE, USARSim	8 criteria; symbolic ratings summed with equal weights
Casado and Bermúdez [8]	2021	Educational framework for autonomous navigation	1 (own, on Gazebo, Simulink and Stateflow)	Demonstration in competition scenarios; single comparator
Herich and Vaščák [9]	2024	Swarm simulator with real-time control	1 (own, on Blender and Python), against SwarmLab	Qualitative; scaling to several hundred vehicles
Dimmig et al. [10]	2025	Systematic survey of aerial-robot simulators	44 catalogued, 14 compared in depth	9 decision factors; taxonomy of 7 classes
Hentati et al. [11]	2018	Tools and frameworks for system performance analysis	6 simulators and 4 ground control stations	Requirements and MOCAP support; MAVLink case study
Tahir et al. [12]	2023	UAV airframes and autonomy components, not simulators	—	Narrative review across 5 aspects of autonomy
Mairaj et al. [13]	2019	Application-specific simulators	Generic platforms plus 5 application-specific classes	Taxonomy by application domain
Nikolaiev and Novotarskyi [14]	2024	Comparative review of drone simulators	Numerous current simulators	Selection factors; case for standardization
Mualla et al. [15]	2018	Agent-based frameworks for aerial transportation	Widely used agent-based frameworks	ISO software quality model; weighted sum scoring

Three observations follow from the table. The first is that the surveys converge on a common set of evaluation axes without agreeing on them. Sensor support, physics fidelity, multi-vehicle scalability, programming interfaces, usability, and maintenance activity recur in the criteria of Chen et al. [7], Dimmig et al. [10], and Mualla et al. [15] alike, and the last of these – whether a platform is still actively updated – deserves more weight than it usually receives, since a simulator that stops tracking autopilot firmware, sensor hardware, and regulatory change becomes unusable regardless of how it once scored. But the criteria sets are not the same. Dimmig et al. [10] rate visual fidelity and autopilot compatibility for SITL and HITL testing, neither of which appears in Chen et al. [7]; Chen et al. [7] score update frequency as a criterion in its own right, whereas Dimmig et al. [10] fold it into ease of use. A platform can therefore score well in one survey and poorly in another without either being wrong.

The second observation concerns coverage. Gazebo is named in five of the nine studies – surveyed by Chen et al. [7], Dimmig et al. [10], Hentati et al. [11], and Mairaj et al. [13], and used as the substrate of the framework of Casado and Bermúdez [8]. AirSim, X-Plane, and FlightGear appear in three each. Almost everything else is assessed once or twice, which means that for most platforms the literature offers a single opinion rather than a comparison. Naming drift compounds the problem: Mairaj et al. [13] list V-Rep among generic simulators while Chen et al. [7] and Dimmig et al. [10] assess the same tool under its current name, CoppeliaSim; a reader aggregating across surveys would count it twice.

The third observation is that the taxonomies do not commute. Dimmig et al. [10] classify simulators by primary focus – universal, sensor-focused, learning-focused, dynamics-focused, swarming, part of a flight stack, or general flight simulator – while Mairaj et al. [13] classify by application domain, separating network, environmental, control-algorithm, power, and swarm simulators. The same tool lands in different classes depending on which scheme is applied, and gym-pybullet-drones is placed in two classes at once even within a single scheme. Nor are the scoring methods commensurable: the equal-weight symbolic sum of Chen et al. [7] and the ISO-derived weighted sum of Mualla et al. [15] produce numbers on unrelated scales, and the two simulator-development papers [8, 9] evaluate against one comparator each, Aerostack and SwarmLab respectively, which is positioning rather than benchmarking. This is the situation that leads Nikolaiev and Novotarskyi [14] to argue for standardization, and it is the point on which the present review concurs: the constraint on comparing UAV simulators is not a shortage of comparisons but the absence of a shared protocol under which they could be conducted.

3. Conclusion

This paper has reviewed nine studies of UAV simulation published between 2018 and 2025 and consolidated their scope, coverage, and evaluation methods in table 3. Three results follow, set out in section 2.1: the studies share a recurring vocabulary of evaluation axes but neither name nor weight them alike, so a platform's standing shifts with the survey consulted; coverage is concentrated on a handful of tools, with Gazebo assessed five times and most platforms only once; and neither the classification schemes nor the scoring scales translate between studies. What this makes concrete is the standardization argued for by Nikolaiev and Novotarskyi [14]. The obstacle to choosing a simulator on evidence is not that too few comparisons have been published, but that those published cannot be read together.

Several difficulties will persist whatever protocol is adopted. The reality gap remains the central one: policies learned in simulation transfer imperfectly to hardware, and no reviewed study measures the discrepancy in a way another study could reproduce. Scaling to large swarms while holding both real-time performance and physical fidelity is a standing trade-off rather than a solved problem, as the latency constraint identified by Herich and Vaščák [9] illustrates. Communication behavior – latency, bandwidth limits, interference – is modeled in detail only by the network-specific simulators identified by Mairaj et al. [13], and rarely in the same tool that models flight dynamics well. One further factor cuts across all of these and is easy to overlook when comparing feature lists: a platform's long-term usefulness depends on the activity of its developer community and the frequency of its releases, because UAV hardware, autopilot firmware, and aviation regulation all change faster than an unmaintained

simulator can follow. The point is not hypothetical: of the seven platforms compared by Chen et al. [7], the most recent documentation available for MRDS [20], USARSim [22], and MORSE [21] dates from 2012, 2015, and 2016 respectively.

A benchmarking protocol adequate to this field would need to specify at least three things: reference scenarios covering single-vehicle navigation, swarm coordination, and sensor degradation under adverse weather; reported metrics that include wall-clock cost per simulated second and the measured sim-to-real discrepancy, not only qualitative ratings; and a declared weighting over criteria, in the manner of the ISO-derived scheme of Mualla et al. [15], so that scores state what they assume. Around such a protocol, the developments already visible in the literature would become measurable rather than merely asserted: digital twins that mirror specific vehicles and environments closely enough for predictive maintenance and rapid prototyping, learning-based control evaluated on identical scenarios across platforms, and explicit modeling of decentralized decision-making in swarms. The computational cost of that last item is substantial, and the elastic-scaling techniques developed for microservice architectures – machine-learning-driven autoscaling of the kind examined in [28] – offer a plausible route to running high-fidelity swarm simulations at the scale a shared benchmark would demand.

Author contributions

Conceptualization, Andrey Kupin, Yuriy Osadchuk, and Maksym Kosei; methodology, Andrey Kupin, Yuriy Osadchuk, and Maksym Kosei; writing—original draft, Andrey Kupin, Yuriy Osadchuk, and Maksym Kosei; writing—review and editing, Andrey Kupin, Yuriy Osadchuk, and Maksym Kosei. All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data availability statement

No new data were created or analyzed during this study. Data sharing is not applicable.

Conflicts of interest

The authors declare no conflict of interest.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4o and Gemini in order to: grammar and spelling check; formatting assistance; paraphrase and reword. Elicit (https://elicit.com/) was used to identify relevant research papers. In preparing the final version, the authors used an AI assistant to: remove redundant and repetitive passages; correct the attribution of claims to their sources; and check the internal consistency of reported figures, criteria, and cross-references. Every claim so produced was verified by the authors against the cited sources. After using these tools and services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. V. Riabko, T. A. Vakaliuk, O. V. Zaika, R. P. Kukharchuk, V. V. Kontsedailo, Edge computing applications: using a linear MEMS microphone array for UAV position detection through sound

source localization, in: T. A. Vakaliuk, S. O. Semerikov (Eds.), Proceedings of the 4th Edge Computing Workshop (doors 2024), Zhytomyr, Ukraine, April 5, 2024, volume 3666 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 14–36. URL: https://ceur-ws.org/Vol-3666/paper02.pdf.

[2] O. Kucherenko, T. Vakaliuk, Comparison of Separate Path Planning Algorithms to UAV Control, in: 2024 IEEE 6th International Conference on Modern Electrical and Energy System, MEES 2024, 2024. doi:10.1109/MEES64070.2024.11405272.

[3] AUVSI, Home, 2025. URL: https://www.auvsi.org.

[4] N. Elmeseiry, N. Alshaer, T. Ismail, A Detailed Survey and Future Directions of Unmanned Aerial Vehicles (UAVs) with Potential Applications, Aerospace 8 (2021) 363. URL: https://www.mdpi.com/2226-4310/8/12/363. doi:10.3390/aerospace8120363.

[5] V. Petrov, Vertical takeoff and landing airframe, U.S. Patent US10556679B2, 2020. URL: https://patents.google.com/patent/US10556679B2/, assignee: PteroDynamics Inc.

[6] A. I. Hentati, L. C. Fourati, Comprehensive survey of UAVs communication networks, Computer Standards & Interfaces 72 (2020) 103451. doi:10.1016/j.csi.2020.103451.

[7] Z. Chen, J. Yan, B. Ma, K. Shi, Q. Yu, W. Yuan, A Survey on Open-Source Simulation Platforms for Multi-Copter UAV Swarms, Robotics 12 (2023) 53. URL: https://www.mdpi.com/2218-6581/12/2/53. doi:10.3390/robotics12020053.

[8] R. Casado, A. Bermúdez, A Simulation Framework for Developing Autonomous Drone Navigation Systems, Electronics 10 (2021) 7. URL: https://www.mdpi.com/2079-9292/10/1/7. doi:10.3390/electronics10010007.

[9] D. Herich, J. Vaščák, Advancing Drone Swarm Simulation: A Comprehensive System for Real-Time Control and Dynamic Interaction, Acta Electrotechnica et Informatica 24 (2024) 16–22. URL: https://doi.org/10.2478/aei-2024-0003. doi:10.2478/aei-2024-0003.

[10] C. A. Dimmig, G. Silano, K. McGuire, C. Gabellieri, W. Hönig, J. Moore, M. Kobilarov, Survey of Simulators for Aerial Robots: An Overview and In-Depth Systematic Comparisons [Survey], IEEE Robotics & Automation Magazine 32 (2025) 153–166. doi:10.1109/MRA.2024.3433171.

[11] A. I. Hentati, L. Krichen, M. Fourati, L. C. Fourati, Simulation Tools, Environments and Frameworks for UAV Systems Performance Analysis, in: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018, pp. 1495–1500. doi:10.1109/IWCMC.2018.8450505.

[12] M. A. Tahir, I. Mir, T. U. Islam, A Review of UAV Platforms for Autonomous Applications: Comprehensive Analysis and Future Directions, IEEE Access 11 (2023) 52540–52554. doi:10.1109/ACCESS.2023.3273780.

[13] A. Mairaj, A. I. Baba, A. Y. Javaid, Application specific drone simulators: Recent advances and challenges, Simulation Modelling Practice and Theory 94 (2019) 100–117. doi:10.1016/j.simpat.2019.01.004.

[14] M. Nikolaiev, M. Novotarskyi, Comparative Review of Drone Simulators, Information, Computing and Intelligent Systems 4 (2024) 79–98. doi:10.20535/2786-8729.4.2024.300614.

[15] Y. Mualla, W. Bai, S. Galland, C. Nicolle, Comparison of Agent-based Simulation Frameworks for Unmanned Aerial Transportation Applications, Procedia Computer Science 130 (2018) 791–796. doi:10.1016/j.procs.2018.04.137, The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

[16] Cyberbotics, Robotics simulation with Webots, 2025. URL: https://cyberbotics.com/.

[17] Open Robotics, Gazebo, 2025. URL: https://gazebosim.org/.

[18] Coppelia Robotics, Robot simulator CoppeliaSim: create, compose, simulate, any robot, 2025. URL: https://www.coppeliarobotics.com/.

[19] ARGoS Project, The ARGoS Website, 2022. URL: https://www.argos-sim.info/.

[20] Microsoft Corporation, Microsoft Robotics Developer Studio, 2012. URL: https://learn.microsoft.com/en-us/previous-versions/microsoft-robotics/bb483024(v=msdn.10).

[21] morse-simulator, Modular OpenRobots Simulation Engine, 2016. URL: https://morse-simulator.

github.io/.
[22] USARSim Project, USARSim (Unified System for Automation and Robot Simulation), 2015. URL: https://sourceforge.net/projects/usarsim/.
[23] The MathWorks, Inc., Simulink - Simulation and Model-Based Design, 2025. URL: https://www.mathworks.com/products/simulink.html.
[24] The MathWorks, Inc., Stateflow, 2025. URL: https://www.mathworks.com/products/stateflow.html.
[25] Blender Foundation, Blender – the free and open source 3D creation software, 2025. URL: https://www.blender.org/.
[26] Laboratory of Intelligent Systems, EPFL, lis-epfl/swarmlab: SwarmLab: a versatile Matlab package for drone swarm simulation, 2022. URL: https://github.com/lis-epfl/swarmlab.
[27] E. Soria, F. Schiano, D. Floreano, SwarmLab: a Matlab Drone Swarm Simulator, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 8005–8011. doi:10.1109/IROS45743.2020.9340854.
[28] S. Semerikov, D. Zubov, A. Kupin, M. Kosei, V. Holiver, Models and Technologies for Autoscaling Based on Machine Learning for Microservices Architecture, in: V. Lytvyn, A. Kowalska-Styczen, V. Vysotska (Eds.), Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Systems. Volume I: Machine Learning Workshop, Lviv, Ukraine, April 12-13, 2024, volume 3664 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 316–330. URL: https://ceur-ws.org/Vol-3664/paper22.pdf.