Survey, Selection, and Integration of Aerial Vehicle Simulators

Cora A. Dimmig and Marin Kobilarov
Johns Hopkins University

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

ICRA LONDON · 2023

Abstract

Aerial vehicle testing can be highly inefficient and often costly due to the high propensity for crashes that is inherent with a flying object. This adds significant complexity to aerial vehicle research. Evaluating new control algorithms on hardware can be dangerous, costly, and ecologically unfriendly, due to the frequent replacement of components that break in a crash. Thus, high-fidelity simulators are a necessity and can expedite the development of novel controllers and control techniques. This paper looks to analyze existing aerial vehicle simulators and the decision factors that go into selecting a simulator. Additionally, we include a discussion of the integration of a simulator we are using for aerial grasping research and the advantages and disadvantages of our chosen simulator.

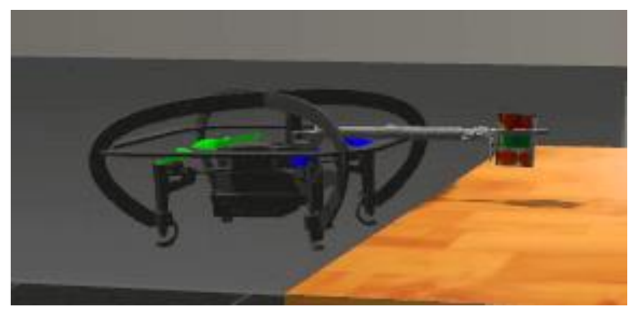
Introduction

Uncrewed Aerial Vehicles (UAVs) are being widely adopted for a variety of use cases and industries, such as for agriculture, inspection, mapping, and search and rescue.

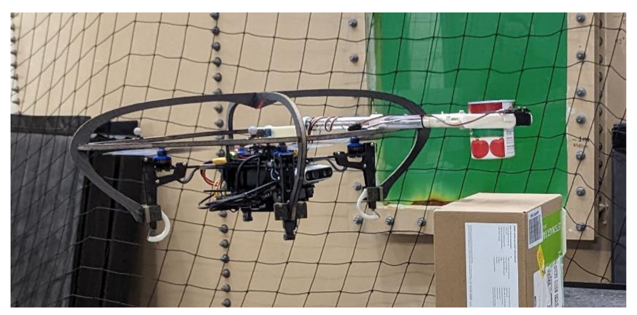
We need strong aerial vehicle simulators for a variety of reasons, including:

- Unexpected behaviors in hardware experiments can be highly dangerous.
- Simulators enable safe, rapid development of algorithms.
- Crashes can be costly, detrimental to development timelines, and harmful to the environment due to generating waste.
- Collecting data on hardware, such as for learning based approaches, can be highly inefficient and often impractical.

In this work, we analyze some of the prominent UAV simulators and key selection criteria and decision factors to consider when selecting a simulator. Furthermore, we describe our selection process and integration of a simulator we are using for aerial grasping research.



(a) Gazebo simulation



(b) Hardware experiment

Figure 1. Our autonomous aerial research platform in flight both in simulation and on hardware grasping a target object.

TABLE I
SELECTION CRITERIA FOR AERIAL VEHICLE SIMULATORS

Criteria	Decision Factors
Physics Engine	Required fidelity for the intended use case
Visual Fidelity	If realistic images are necessary, such as for computer vision or machine learning
ROS Integration	For compatibility with existing software infrastructure
RL API	Ease of integration for RL applications
Autopilots	Compatibility with common autopilots, e.g. PX4 and ArduPilot, such as for software-in-the-loop (SITL) testing
HITL	Infrastructure for performing hardware-in-the-loop (HITL) testing
Multiple Vehicles	Support for simulating multiple vehicles
Sensors	Integration with common sensors, such as cameras (RGB and RGBD), IMU, Magnetometer, GPS, barometer, LIDAR, and optical flow sensors
UAV Models	Support of common UAV models and ease of integrating new models
Simulation Speed	Real-time speed and ability to run in super real-time, such as for learning applications
Integration	Ease of getting started and development with the software

Aerial Vehicle Simulators

Table I outlines selection criteria and decision factors that are frequently considered when comparing aerial vehicle simulators.

Tables II and III compare the most widely used aerial vehicle simulators using some of the selection criteria from Table I.

- Table II compares notable features of the simulation environments.
  - The four physics engines supported by Gazebo (i.e. ODE, Bullet, DART, and Simbody) are labeled as "GazeboPhys".
- Table III compares sensors that are supported by each simulator.

TABLE II
COMPARISON OF FEATURES FOR WIDELY USED AERIAL VEHICLE SIMULATORS: INCLUDED (✓) AND NOT INCLUDED (✗)

Simulator	Physics Engine	Rendering	Visual Fidelity	ROS	RL API	PX4	ArduPilot	HITL	Multiple Vehicles	Ref.
Gazebo	GazeboPhys	OpenGL	Low	✓	✓	✓	✓	✓	✓	[5]
AirSim	Fast Physics / PhysX	Unreal, Unity	High	✓	✓	✓	✓	✓	✓	[3]
Flightmare	Ad hoc, GazeboPhys	Unity	High	✓	✓	✗	✗	✗	✓	[4]
Webots	ODE	OpenGL	Low	✓	✓	✗	✓	✗	✓	[6]
RotorS	GazeboPhys	OpenGL	Low	✓	✗	✗	✗	✓	✗	[11]
FlightGoggles	Ad hoc	Unity3D	High	✓	✗	✗	✗	✓	✗	[12]
Gym-pybullet-drones	PyBullet	OpenGL	Low	✓	✓	✗	✗	✗	✓	[13]

TABLE III
COMPARISON OF INCLUDED SENSORS FOR WIDELY USED AERIAL VEHICLE SIMULATORS: INCLUDED (✓) AND NOT INCLUDED (✗)

Simulator	RGB	Depth	Seg.	Point Cloud	IMU	Mag.	GPS	Barometer	LIDAR	Optical Flow	Ref.
Gazebo	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	[5]
AirSim	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	[3]
Flightmare	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	[4]
Webots	✓	✓	✗	✗	✓	✓	✓	✗	✓	✗	[6]
RotorS	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	[11]
FlightGoggles	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓	[12]
Gym-pybullet-drones	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	[13]

Aerial Grasping Simulator Selection

We aimed to simulate a system that can autonomously detect a target, navigate to that object and grasp it and then detect a destination and place the object, all in an unknown environment.

Our Priorities

- Allow for seamlessly swapping between simulation and hardware.
- Integration with our flight controller, PX4, which highly recommends Gazebo.
- Strong physics engine and collision handling.
- Ease of integration.
- Available sensors.

Simulator Selection: Gazebo

Figure 2. Autonomously grasping an object in clutter.

We are using a modified Uvify IFO-SX quadrotor with a custom collision tolerant carbon fiber foam cage and modular gripper extension package, as seen in Figure 2.

Aerial Grasping Gazebo Integration

Figure 3 shows our aerial grasping research platform in our Gazebo simulation. The vehicle is positioned in front of a target object on the table. Projected from the vehicle's RGB camera is an image of the simulated camera's view.

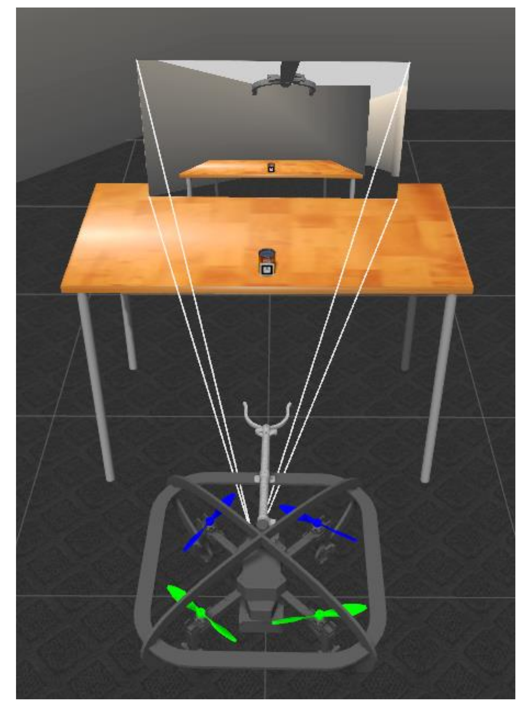


Figure 3. Gazebo simulation.

Gazebo Simulation Advantages

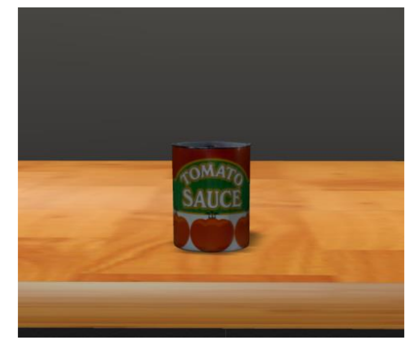
- Essential for integrating and tuning our controller with the PX4 software stack.
- Enabled evaluating performance and debugging issues rapidly and then seamlessly transitioning to hardware.
- After tuning our controller gains in the simulated environment, we found that only minor adjustments were required on the real vehicle.
- Dramatically reduced hardware testing time.
- Efficient transition from simulation to hardware.

Gazebo Simulation Disadvantages

- Images generated in our Gazebo simulation environment have low visual fidelity and the environments have minimal visual features, as seen in Figure 3 and Figure 4(a).
- This was prohibitive to running visual odometry and object detection in our simulated environment.
- Additionally, vision and learning based methods would not transfer well from simulation to hardware.

Figure 4(b) shows an object being detected in the Flightmare simulator using Deep Object Pose Estimation (DOPE) from [28]. The comparable image in our Gazebo simulation, Fig. 4(a), did not yield detections when running DOPE, due to the low visual fidelity.

Moving forward, requiring higher visual fidelity may motivate switching simulators or integrating with a second simulator for different use cases.



(a) Gazebo simulation

(b) Flightmare simulation

Figure 4. Toy can in Gazebo and Flightmare simulation environments. DOPE detection (green bounding box) in the Flightmare simulation.

Conclusions

Selecting a simulator that is best for a particular application space can be very challenging, but rewarding when it increases safety and reduces testing time and cost. In this work, we discussed some of the prominent robotic simulators for aerial vehicles. We enumerate possible decision factors to consider when selecting a simulator and we compare features and integrated sensors across many widely used simulation packages. Pertaining to our recent aerial grasping research, we discussed our considerations when selecting a simulator and our software integration. Finally, we detailed the main advantages and disadvantages of our selected simulator, specific to our research. We hope that this analysis will be valuable to the community when embarking on aerial vehicle research and selecting a simulation environment.

Contact

Cora Dimmig
Johns Hopkins University
cdimmig@jhu.edu

References

[1] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," IEEE Transactions on Robotics, vol. 38, no. 1, pp. 626–645, Feb 2022.
[2] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," IEEE Access, vol. 9, pp. 51 416–51 431, 2021.
[3] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in Field and Service Robotics. Springer, 2018, pp. 621–635.
[4] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in Conference on Robot Learning. PMLR, 2021, pp. 1147–1157.
[5] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat.No.04CH37566), vol. 3, Sep. 2004, pp. 2149–2154 vol.3.
[6] O. Michel, "Cyberbotics Ltd. Webots™: Professional mobile robot simulation," Int. J. of Adv. Robotic Sys., vol. 1, no. 1, p. 5, 2004.
[7] M. Körber, J. Lange, S. Rediske, S. Steinmann, and R. Glück, "Comparing popular simulation environments in the scope of robotics and reinforcement learning," arXiv preprint arXiv:2103.04616, 2021.
[8] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2012, pp. 5026–5033.
[9] E. Coumans and Y. Bai, "PyBullet, a python module for physics simulation for games, robotics and machine learning," 2016. [Online]. Available: https://pybullet.org/
[10] J. Saunders, S. Saeedi, and W. Li, "Autonomous aerial delivery vehicles, a survey of techniques on how aerial package delivery is achieved," arXiv preprint arXiv:2110.02429, 2022.
[11] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS — a modular gazebo MAV simulator framework," Robot Operating System (ROS) The Complete Reference (Volume 1), pp. 595–625, 2016.
[12] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Nov 2019, pp. 6941–6948.
[13] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with PyBullet physics for reinforcement learning of multi-agent quadcopter control," in IEEE/RSJ Int. Conf. on Intell. Robots and Sys. (IROS), Sep. 2021, pp. 7512–7519.
[14] A. Mairaj, A. I. Baba, and A. Y. Javaid, "Application specific drone simulators: Recent advances and challenges," Simulation Modelling Practice and Theory, vol. 94, pp. 100–117, 2019.
[15] J. Saunders, S. Saeedi, and W. Li, "Parallel reinforcement learning simulation for visual quadrotor navigation," arXiv preprint arXiv:2209.11094, 2022.
[16] "jMAVSim." [Online]. Available: https://github.com/PX4/jMAVSim
[17] J. Berndt, "JSBSim: An open source flight dynamics model in C++," in AIAA Modeling and Sim. Tech. Conf. and Exhibit, 2004, p. 4923.
[18] MATLAB, "UAV toolbox." [Online]. Available: https://www.mathworks.com/products/uav.html
[19] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep. 2021, pp. 1205–1212.
[20] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," Science Robotics, vol. 6, no. 59, p. eabg5810, 2021.
[21] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," arXiv preprint arXiv:2006.05768, 2020.
[22] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot," Applied Sciences, vol. 9, no. 24, 2019.
[23] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing," IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 8631–8638, Oct 2021.
[24] S. Gronauer, M. Kissel, L. Sacchetto, M. Korte, and K. Diepold, "Using simulation optimization to improve zero-shot policy transfer of quadrotors," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct 2022, pp. 10 170–10 176.
[25] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for visionbased flight," Science Robotics, vol. 7, no. 67, p. eabl6259, 2022.
[26] G. Li, X. Liu, and G. Loianno, "RotorTM: A flexible simulator for aerial transportation and manipulation," arXiv preprint arXiv:2205.05140, 2023.
[27] ROS, "Solidworks to URDF exporter." [Online]. Available: http://wiki.ros.org/sw urdf exporter
[28] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in Conference on Robot Learning (CoRL), 2018.