Survey, Selection, and Integration of Aerial Vehicle Simulators

Cora A. Dimmig[1] and Marin Kobilarov[1]

***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.**

I. 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. In particular, aerial manipulation applications have been on the rise, as discussed in [1], such as for parcel delivery, warehouse management, and sample collection.

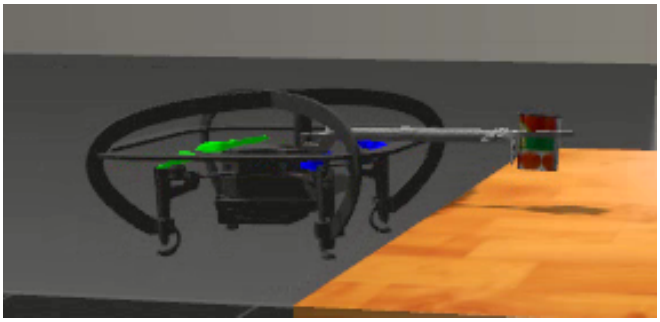
Testing experimental algorithms directly on hardware can be highly dangerous as unexpected behaviors arise. Furthermore, crashes can be costly, detrimental to development timelines, and harmful to the environment due to the waste created from frequent replacement of vehicle components that break in crashes. Additionally, with the rise of Machine Learning (ML) based techniques, collecting data on hardware can be highly inefficient and often impractical. Thus, a strong robotic simulator for UAVs can be essential for rapid development and progression of the field.

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. Fig. 1 shows our aerial grasping research platform in flight both in simulation and on hardware grasping a target object.

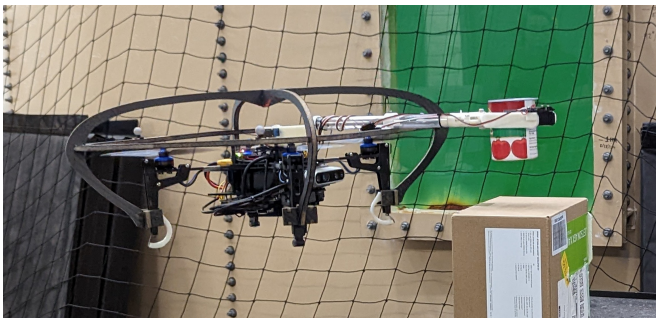
II. AERIAL VEHICLE SIMULATORS

Many options for physics-based simulators have arisen for the field of robotics as a whole as well as specific to aerial vehicles, all with unique advantages and disadvantages. Additionally, as the field evolves the latest innovations in simulators are constantly developing. Such a large set of options and frequent innovations ultimately makes selecting a simulator that is best suited for a particular researcher's application very challenging.

[1]Department of Mechanical Engineering and the Laboratory for Computational Sensing and Robotics (LCSR), Johns Hopkins University, Baltimore, MD 21218, USA. Email: cdimmig@jhu.edu, marin@jhu.edu



(a) Gazebo simulation



(b) Hardware experiment

Fig. 1. Autonomous aerial platform in flight grasping a target object.

A. Survey of Aerial Vehicle Robotics Simulators

The survey in [2] reviews a wide selection of physics simulators for robotic applications. The authors analyze many different application spaces, such as soft robotics, medical robotics, manipulation, legged locomotion, underwater vehicles, and aerial vehicles. The authors present tables comparing features for many different domains. For aerial robots, the following simulators are compared: AirSim [3], Flightmare [4], Gazebo [5], and Webots [6].

The authors in [7] provide a comparison of four simulators that are commonly used across many domains: Gazebo [5], MuJoCo [8], PyBullet [9], and Webots [6]. Specifically, the authors analyze these simulators with respect to stability, speed, and hardware utilization for reinforcement learning (RL) applications. They discuss the trade-off between real-time factor and precision; the simulation speed can be increased by increasing the simulation time step, however this directly reduces the resulting precision. Additionally, the authors analyze the ease of adopting each simulator by discussing the challenges users may encounter when first interfacing with the software.

Unfortunatley, not every simulator has easy integration for aerial vehicles. The dynamics considerations for a manipula-

tor or ground vehicle can vary significantly compared to an aerial vehicle, in particular for research that hopes to consider aerodynamic effects. The authors in [10] analyze a broad spectrum of considerations for aerial delivery vehicles, including the selection of simulators. The authors compare the following simulation packages: RotorS [11] (which is built on Gazebo), AirSim [3], Flightmare [4], FlightGoggles [12], and Gym-pybullet-drones [13] (which is built on PyBullet). Similarly, the authors in [14] analyze aerial vehicle specific simulators, including many less frequented simulators, and discuss some simulator selection criteria.

Also of note are extensions to widely used simulators, such as PRL4AirSim [15] which is a package built on AirSim for efficient parallel training for RL applications.

B. Aerial Vehicle Simulator Selection Criteria

The primary factor to consider when selecting a simulator is often the intended application space and which simulators have the features and sensors required for that application space. In most cases, the perfect simulator, that encompasses all desired criteria for a particular project, does not exist. This is often the impetus for the development of new simulators.

From our experience and the works we have cited, we have compiled a list of selection criteria and decision factors that are frequently considered when comparing aerial vehicle simulators. These comparison points are expressed in Table I.

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

C. Comparison of Aerial Simulators

Using information compiled from [2], [10], and the respective documentation for each simulator, we present Tables II and III for comparing aerial vehicle simulators using some of the selection criteria we discussed in Table I. We include in Tables II and III the most widely used aerial vehicle simulators. Not included in this analysis, but also worth noting, as mentioned in [10], are: jMAVSim [16], JSBSim [17], and the UAV Toolbox in Matlab [18].

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

D. Applications of Aerial Simulators

It is valuable to analyze applications of aerial simulators to see how they have been utilized in the past. Flightmare is extensively used for learning applications, such as for autonomous drone racing [19], and outdoor high-speed flight [20]. In [20], the authors use Flightmare with the RotorS Gazebo plugin, for physics modeling, and Unity, for rendering. In [21], Gazebo with RotorS is used for training policies for drone acrobatics.

AirSim is used in [22] for obstacle avoidance with drones using deep RL and in [23] for autonomous drone racing. The authors in [23] note that they selected AirSim due to AirSim's debugging API and support of more sensors and control modes compared to other simulators such as Flightmare and FlightGoggles. Additionally, they note AirSim's detailed documentation and the ease of changing the environment for building unique drone racing tacks.

The authors of [24] use PyBullet, as in the library gym-pybullet-drones, in their work for zero-shot policy transfer of quadrotors.

E. Specialized Simulators

Many simulators can generalize across a wide-variety of use cases. However, there will always be application spaces that do not fit within the bounds of existing simulators and require the development of specialized software. We have included a couple such examples here for reference.

The Agilicious open-source and open-hardware platform for vision-based agile flight [25] introduces a specialized simulator to model varying levels of fidelity of the quadrotor dynamics. However, the Agilicious software stack also includes interfaces to RotorS and Flightmare.

The field of aerial manipulation is rapidly growing [1], though few simulators specific to these types of applications have arisen. In [26], the authors report RotorTM, a new simulator for aerial transportation and manipulation. In particular, they investigate simulating cable-suspended loads and passive connection mechanisms between multiple vehicles. These equipping mechanisms are not modeled in other common simulators.

III. AERIAL GRASPING SIMULATOR SELECTION AND INTEGRATION

We assessed the aerial vehicle simulators from Tables II and III for aerial grasping research. We aimed to simulate a system that can autonomously detect a target object, navigate to that object and grasp it and then detect a destination and place the object, all in an unknown environment.

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]

A. Aerial Grasping Simulator Selection

One of our highest priorities when selecting a simulator was the ability to seamlessly swap between the simulation and hardware vehicle for rapid testing and prototyping of algorithms. Thus, we wanted a simulator that had integration with our intended flight controller, PX4. This would enable running our full software stack in simulation for integration of new features, testing functionality, and debugging. Then when we were ready, and confident in our algorithms, we could run the same software stack on the physical hardware platform. PX4 highly recommends Gazebo for SITL simulation. Our other priorities were a strong physics engine (and handling of collisions), ease of integration, and available sensors, which were also satisfied by Gazebo. Thus, we decided to use Gazebo with PX4's SITL simulation architecture as a starting point.

B. Aerial Grasping Gazebo Integration

For our research, 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 Fig. 1. We modeled our vehicle in SolidWorks and then exported an Unified Robotics Description Format (URDF) file using the SolidWorks to URDF exporter [27]. We then converted this file to a Gazebo SDF model. Ultimately this allowed us to have accurate collision geometries and form-factors in our simulation.

We created a plugin for our simulated gripper that matches the software interface of our hardware gripper. Then we built a simulated environment similar to our real-world testing area and imported models for our target objects.

Fig. 2 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.

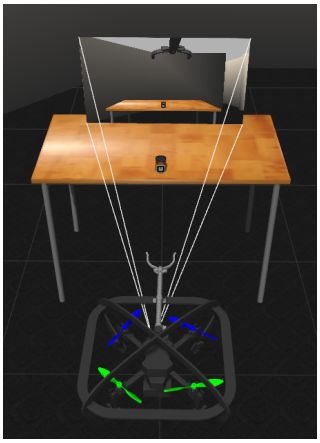


Fig. 2. Gazebo simulation.

1) Advantages: Our simulation was essential for integrating and tuning our controller with the PX4 software stack. It allowed us to evaluate performance and debug issues rapidly and then run the same software seamlessly on the hardware vehicle. Additionally, after tuning our controller gains in the simulated environment, we found that only minor adjustments were required on the real vehicle. Ultimately, this dramatically reduced hardware testing time. Overall, the shift from simulation to hardware for our control algorithms was highly efficient and allowed us to quickly replicate our simulation results, as seen in Fig. 1.

2) Disadvantages: The images generated in our Gazebo simulation environment have low visual fidelity and the environments have minimal visual features, as seen in Fig. 3(a). This ended up being prohibitive to running visual odometry and object detection in our simulated environment, including

(a) Gazebo simulation (b) Flightmare simulation

Fig. 3. Toy can in Gazebo and Flightmare simulation environments. DOPE detection, in the Flightmare simulation, indicated by green bounding box.

when more textures were added to the environment. Ultimately, this necessitated using alternative methods for odometry and AR tags for object detection in the simulation, which is contrary to our goal of running identical software stacks in simulation and hardware. Furthermore, as our research moves towards vision and learning based methods, we will require higher fidelity rendering. For example, Fig. 3(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. 3(a), did not yield detections when running DOPE. Moving forward, requiring higher visual fidelity may motivate switching simulators or integrating with a second simulator for different use cases.

IV. CONCLUSION

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.

ACKNOWLEDGMENT

We gratefully acknowledge the support of the National Science Foundation under grant #1925189.

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 $\mathrm{SE}(3)$ 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 vision-based 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.