Simulation-Guided Testing for Autonomous Aerial Robotics Applications

Rafael Perez-Segui
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0002-9508-1055

Pedro Arias-Perez
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0001-7166-9367

Javier Melero-Deza
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0002-1420-2960

David Perez-Saura
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0003-2571-3165

Miguel Fernandez-Cortizas
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0002-3822-075X

Pascual Campoy
Computer Vision and Aerial Robotics
Universidad Politécnica de Madrid
Madrid, España
0000-0002-9894-2009

***Abstract*—The deployment of autonomous unmanned aerial vehicles (UAVs) is rapidly increasing due to their ability to perform a variety of tasks, including industrial inspection, search and rescue, and transportation. To ensure the safe and efficient operation of UAVs, simulation testing is essential for identifying and mitigating potential risks. In some applications, such as industrial inspections, performing real flight experiments on the final installations is complicated or even risky, such as wind turbines or photovoltaic panels. In this case, being able to perform real flights using simulated data from the final application allows more thorough testing of the robotic system without putting expensive industrial devices at risk. In this paper, we show how the use of the Aerostack2 framework eases the step between simulation and final application using simulation data for real flight experiments.**

***Index Terms*—Aerial Robotics, ROS 2, Gazebo, Simulation, Aerostack2**

I. Introduction

The use of simulation in aerial robotics has proven to be crucial in the development of autonomous algorithms, particularly in drones. One of the significant benefits of simulation is the ability to test and validate algorithms before deployment. Engineers can identify and resolve any issues in a virtual environment without incurring high costs or risking damage to the drone. The simulation also allows engineers to evaluate the performance of the algorithms under various conditions, such as different weather patterns, lighting conditions, and obstacles. Another advantage of simulation in autonomous algorithms is the ability to anticipate and address potential safety issues before deployment. Autonomous drones involve potential safety hazards, including collisions with objects and people. Simulation allows researchers and engineers to anticipate and address potential safety issues, ensuring that the algorithm operates within safe parameters.

Once a design has been optimized in simulation, it can be tested in real flight experiments to validate its performance and safety. But carrying out experiments in certain environments can be risky. For example, industrial installations contain expensive and sensitive equipment, and accidents can result in costly damage. In such cases, using simulation data to inform and guide real-world experiments allows testing the behavior of the robotic system far from the final installations, reducing the risk of costly accidents and increasing safety during development.

This work presents two main contributions. First, it explains the development process of a real industrial application using the Aerostack2[1] framework [1], which provides a comprehensive and flexible solution for the design and implementation of autonomous UAVs. Second, it demonstrates the use of simulation data to guide and validate real flight experiments. This approach allows for a safer and more efficient testing process, as potential issues and risks can be identified and mitigated in the simulation environment before moving on to final environment experiments.

II. Related Work

Related work in research for aerial robotics shows a big number of simulators. We mention only a few of the existing

This work has been supported by the project COPILOT ref. Y2020\EMT6368 ”Control, Monitoring and Operation of Photovoltaic Solar Power Plants by means of synergic integration of Drones, IoT, and advanced communication technologies”, funded by Madrid Government under the R&D Synergic Projects Program. We acknowledge the support of the European Union through the Horizon Europe Project No. 101070254 CORESENSE. This work has also been supported by the project INSERTION ref. ID2021-127648OBC32, ”UAV Perception, Control and Operation in Harsh Environments”, funded by the Spanish Ministry of Science and Innovation under the program ”Projects for Knowledge Generating”. The work of the second author is supported by the Grant FPU20/07198 of the Spanish Ministry for Universities. The work of the fourth author is supported by the Spanish Ministry of Science and Innovation under its Program for Technical Assistants PTA2021-020671.

[1]https://github.com/aerostack2/aerostack2

open-source simulators that have been widely used by aerial robotics, highlighting important features and limitations of each one.

RotorS [2]: Is a modular Micro Aerial Vehicle (MAV) simulation framework built on Gazebo [3], which allows a quick start to perform research on MAVs. The simulator was designed in a modular way so that different controllers and state estimators can be used interchangeably, while incorporating new MAVs is reduced to a few steps. The provided controllers can be adapted to a custom vehicle by simply changing a parameter file. Different controllers and state estimators can be compared with the provided evaluation framework. All components were designed to be analogous to their real-world counterparts. This allows the usage of the same controllers and state estimators, including their parameters, in the simulation as on the real MAV.

AirSim [4]: Is a photo-realistic simulator built on Unreal Engine that offers physically and visually realistic simulations. It includes a physics engine that can operate at a high frequency for real-time hardware-in-the-loop (HITL) simulations with support for popular protocols (e.g. MavLink). The simulator is designed from the ground up to be extensible to accommodate new types of vehicles, hardware platforms and software protocols.

FlightGoggles [5]: Is an open-source photo-realistic sensor simulator for perception-driven robotic vehicles. It consists of two separate components, the photo-realistic rendering engine built on Unity and a quadrotor dynamics simulation engine. It also provides an interface with real-world aircrafts for image and data processing.

Flightmare [6]: Like FlightGoggles, is a flexible modular quadrotor simulator composed of two main components: a configurable rendering engine built on Unity and a flexible physics engine for dynamics simulation. Those two components are totally decoupled and can run independently from each other. In addition, it also provides an interface with Gazebo simulator.

In conclusion, after examining several simulators for autonomous drone flights, it can be concluded that they do not provide a direct pathway from simulation to real-world drone operations. Rather, they focus on specific components or aspects of drone flights, such as simulation training environment, testing of specific platform or image processing algorithms. These simulators are valuable tools for training and testing autonomous drone systems in a controlled environment, but they do not necessarily prepare them for the complex and unpredictable realities of real-world drone operations. In addition, they do not facilitate the simulation for any platform, specializing in several with specific characteristics.

III. Methodology

Aerostack2 framework interaction with the Platform interface facilitates the integration of both physical and simulated interfaces, without requiring the rest of the framework to distinguish between the two. This platform agnostic feature is a fundamental pillar that strengthens the sim2real capabilities of the algorithms developed [1].

For simulation purposes, Aerostack2 provides a platform based on Gazebo Simulator [3], which sends control commands to the simulated UAV. In order to do so, the platform receives the control commands from the motion controller. As the platform is using a common interface, it can be easily decoupled for a real UAV's platform, which will receive the control commands to be sent to the UAV exactly the same way.

In order to launch the simulation, aerial robots and external objects are defined in a configuration file. This configuration file is then loaded by the Aerostack2's component that launches the simulation with every asset of each model that is defined in the configuration file, alongside every ROS 2 to Gazebo Bridge that is unrelated to aerial robots. Aerostack2 Gazebo platform is in charge of launching every ROS 2 to Gazebo Bridge related to aerial robots and their sensors that are defined in the configuration file. This scheme is shown in Fig. 1.

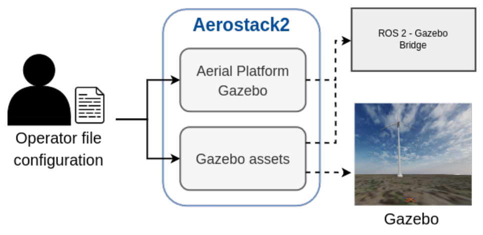

Fig. 1. Gazebo simulator launch in Aerostack2

After this, through ROS 2 - Gazebo bridge, Aerostack2 can communicate with the simulator using the ROS 2 common communication, that is, topic, services, actions, and transformation frames (tf2).

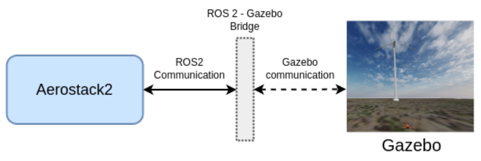

Fig. 2. Gazebo simulator communication with Aerostack2

The development of autonomous systems in aerial robotics typically involves three stages of simulation: simulate the aircraft and its environment in a simulation station, simulate them using the final hardware, and perform the real implementation. For this, the Aerostack2 framework proposes the following structure, shown in Fig. 3

A. Simulate in Simulation Station

The first stage of development involves the simulation of the aircraft, environment, and algorithms in a simulation station. In this, engineers design and develop a simulation environment that includes the virtual representation of the aircraft and the surrounding environment. The simulation station allows

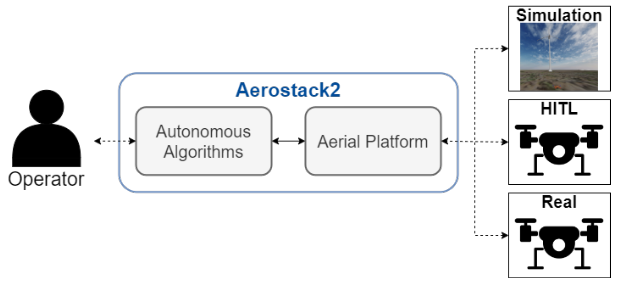

Fig. 3. Simulation scheme in Aerostack2, where the aircraft can be simulated in different ways or be the real one, keeping the same algorithms and only modifying the aerial platform

the engineers to test and validate the performance of the algorithms in a virtual environment before deploying them on a physical aircraft.

B. Simulate in Deployment Hardware

The second stage of development involves the simulation of the environment in the simulation station, the processing of algorithms on the on-board computer, and the simulation of the platform using hardware in the loop technique. In this, engineers use the simulation station to simulate the environment and test the algorithms on the on-board computer. They then use hardware in the loop to simulate the behavior of the platform in response to the algorithms processed by the on-board computer. This stage allows engineers to test and validate the performance of the system under realistic hardware limitations and identify any issues that may arise.

C. Testing on Deployment Hardware

The final stage of development involves the testing of the system with the on-board computer, real hardware, and real environment. In this stage, engineers deploy the system on a physical aircraft and test its performance in a real-world environment. This stage allows engineers to validate the performance of the system under realistic conditions and check if the execution is faithful to the simulation.

IV. Use of case for wind turbine inspection

This use of case consists of the inspection of a wind turbine's blades while these are rotating. The nacelle connected to the rotor that makes the blades rotate is also rotating along its z-axis, located at the point where it is connected to the tower.

The mission consists of following the inspection points related to the wind turbine rotor. These points are calculated depending on certain parameters such as the blade's length, the intrinsic camera parameters, or the security inspection distance.

The prior data received from the wind turbine is the position in WGS84 coordinates of the base of the wind turbine, blades length, and rotor height. The real-time data received from the wind turbine is the orientation of the rotor given in azimuth.

A. Simulation of the wind turbine

For the simulation of the wind turbine, a model of a wind turbine has been created in Blender tool. This model has been divided into three main parts. These parts are then turned into separate SDFormat models and connected by rotatory joints. These parts are:

- Tower: static part of the wind turbine. The base of the tower has been set as the origin of our wind turbine coordinate system.
- Nacelle: it consists of a box that connects the tower with the rotor. The top of the tower is connected by a rotation joint that rotates in yaw.
- Blades: this model contains the rotor with the blades. The rotor is connected to the front of the nacelle with another rotation joint that rotates in roll.

In order to move the blades and the nacelle within the simulation, Gazebo's joint speed controller plugins has been used for each of the joints within the model. These plugins have then been bridged to ROS 2.

A simulated GPS sensor has been integrated into the nacelle so we can receive the WGS84 coordinates. The cartesian orientation of this sensor is used to calculate the azimuth which is then sent through a ROS 2 topic. It primordial to get real-like data structures and values from the simulation.

As the mission is planned with poses relative to the wind turbine rotors, every component of the wind turbine has been added to the global transformation tree, we address in this way the problem of navigating in the global coordinate system with the given relative poses.

B. Simulation of the UAV

The UAV simulation is mainly based on the Gazebo *Multicopter Velocity Control* plugin. This velocity controller for multicopters allows control over the linear velocity and the yaw angular velocity of the vehicle. It requires a vehicle with at least four rotors (*Multicopter Motor Model* plugin) for the controller to function.

In addition, other Gazebo plugins are added to the quadrotor, a magnetometer, an IMU and a GPS receiver.

C. Methodology

The methodology used to address this problem consists on using a fully simulated environment first, and then start detaching simulated components for real components gradually in order to finally deploy the mission in a real environment in a secure and efficient way. This process has taken three steps before deploying every component in a real environment.

1) Simulation in Gazebo: Every hardware component is simulated in Gazebo simulator. Aerostack2 receives the data bridged from Gazebo coming from the UAV and the wind turbine. The control commands for both the wind turbine and the UAV are bridged from ROS 2 and sent to Gazebo, as shown in Fig. 4.

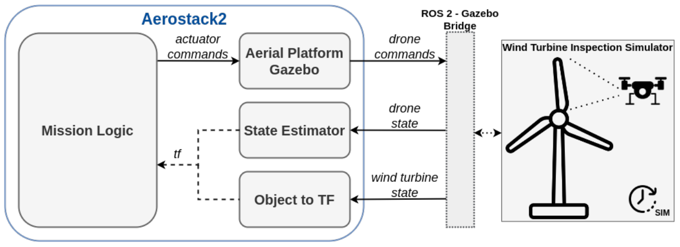

Fig. 4. Fully simulated wind turbine inspection

2) Hybrid simulation, Gazebo and DJI's HITL: Gazebo's basic quadrotor is detached from the simulation and substituted for DJI's Hardware In The Loop (HITL) as seen in image 5. This step ensures that the platform that is going to be used in the real flight works with the initial mission planning and data regarding the wind turbine.

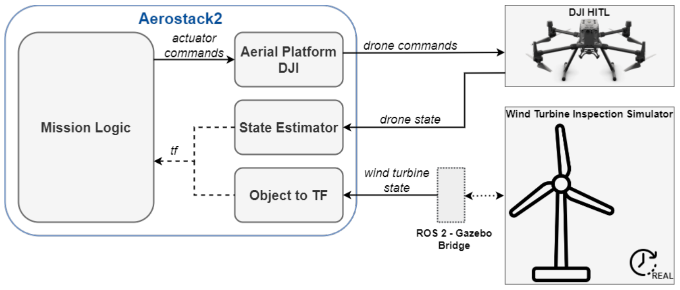

Fig. 5. Integrating Gazebo simulation and DJI's HITL

3) Simulation in Gazebo with real flight: The last step before real deployment consists of substituting DJI's HITL for the real hardware as shown in Fig. 6. The wind turbine simulation allows testing the aerial system in real flights without endangering a wind turbine.

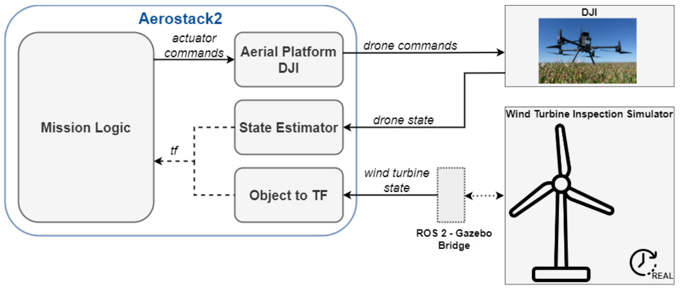

Fig. 6. Integrating Gazebo simulation and Real flight

In order to monitor the mission planning and execution, all the information coming from every component used in this step has been integrated into RViz2 visualizer. The information integrated can be seen in Fig. 7.

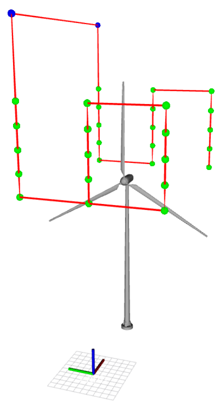

Fig. 7. Integrating Gazebo simulation and real flight. The inspection waypoints are in green, the trajectory in red and the axes represent the drone pose.

V. Conclusions and Future Work

We have highlight the importance of simulation for safer and more efficient testing of autonomous unmanned aerial vehicles algorithms. The experiment presented showcases the use of simulation data for wind turbine inspection, using a hybrid testing architecture to evaluate the algorithms safely. The use of the Aerostack2 framework is shown to be beneficial in facilitating the simulation stages of UAV testing, resulting in more efficient and reliable testing processes. Overall, the framework's implementation can significantly improve the safety and efficiency of testing processes, making it a valuable asset in the development of UAV algorithms.

Moving forward, the next step is to test the algorithms used for simulation process in real-world scenarios with a wind turbine. It will be crucial to compare the results obtained from simulation testing with those obtained from the real-world experiments, in order to assess the accuracy and reliability of the simulation model. Overall, this future work will provide a more comprehensive understanding of the efficacy of the simulation model and the Aerostack2 framework in the development of autonomous UAVs for industrial inspection applications.

References

[1] M. Fernandez-Cortizas, M. Molina, P. Arias-Perez, R. Perez-Segui, D. Perez-Saura, and P. Campoy, "Aerostack2: A software framework for developing multi-robot aerial systems," 2023.

[2] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[3] 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. IEEE, 2004, pp. 2149–2154.

[4] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[5] 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 *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6941–6948.

[6] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.