Unified Co-Simulation Framework for Autonomous UAVs

Sri Ranganathan Palaniappan*
Georgia Institute of Technology
Atlanta, GA, USA
spalaniappan9@gatech.edu

Varun Pateel*
Georgia Institute of Technology
Atlanta, GA, USA
vpateel@gatech.edu

Sam Jijina
Georgia Institute of Technology
Atlanta, GA, USA
sam.jijina@gatech.edu

Jeffrey Young
Georgia Institute of Technology
Atlanta, GA, USA
jyoung9@gatech.edu

Hyesoon Kim
Georgia Institute of Technology
Atlanta, GA, USA
hyesoon@cc.gatech.edu

ABSTRACT

Autonomous drones (UAVs) have rapidly grown in popularity due to their form factor, agility, and ability to operate in harsh or hostile environments. Drone systems come in various form factors and configurations and operate under tight physical parameters. Further, it has been a significant challenge for architects and researchers to develop optimal drone designs as open-source simulation frameworks either lack the necessary capabilities to simulate a full drone flight stack or they are extremely tedious to setup with little or no maintenance or support. In this paper, we develop and present UniUAVSim, our fully open-source co-simulation framework capable of running software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations concurrently. The paper also provides insights into the abstraction of a drone flight stack and details how these abstractions aid in creating a simulation framework which can accurately provide an optimal drone design given physical parameters and constraints. The framework was validated with real-world hardware and is available to the research community to aid in future architecture research for autonomous systems.

KEYWORDS

Drones, UAV, Distributed Systems, Simulation, Hardware-in-the-loop, Software-in-the-loop, FPGA

ACM Reference Format:
Sri Ranganathan Palaniappan, Varun Pateel, Sam Jijina, Jeffrey Young, and Hyesoon Kim. 2023. Unified Co-Simulation Framework for Autonomous UAVs. In *Practice and Experience in Advanced Research Computing (PEARC '23), July 23–27, 2023, Portland, OR, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3569951.3597544

1 INTRODUCTION & MOTIVATION

Autonomous drones, also known as Unmanned Aerial Vehicles UAVs, have become increasingly popular in recent years due to their ability to perform a wide range of tasks without human intervention. Drone systems come in various form factors to suit different missions and environments such as multi-rotor, fixed-wing, and hybrid VTOL versions which combine the advantages of both multi-rotor and fixed-wing drones.

UAV architecture is a rapidly evolving domain. Initial designs focused on high performance purpose-specialized drones with the lowest latencies for a specific task such as *object tracking*. These drones were well suited for their purpose and the technology made its way to the recreational sector with popular models such as DJI Mavic [3] and Skydio [19]. Enhanced drone workloads and missions have led to the proliferation of *general-purpose* drones in both the commercial and recreational markets. General-purpose drones are built on a compute platform which closely relates to the general-purpose model of computing, hence the use of Raspberry Pis [7], Intel NUCs [10], and Nvidia Jetson [16] is seen in such drones. These drones are able to adapt to changing workloads and environments which has shifted the primary focus of drone design from fully performant to a combination of performant and resource efficient.

Efficient development of drones has become increasingly crucial due to the vast choice of components. For example, a common question facing architects is *whether to attach a GPU or an FPGA to a drone for accelerating computations?* The answer lies in weighing several characteristics until an optimal design is reached. This is where simulation and modeling play a critical role. While there has been a lot of research effort in designing intricate models [1, 9, 11, 12], not all are integrated into a simulation platform. For those which are integrated into a simulation platform, they either perform software-in-the-loop (SITL) or hardware-in-the-loop (HITL) simulations and to the best of our knowledge, almost no open-source platform integrates both into one unified framework. Hence we propose UniUAVSim, our own unified co-simulation framework which can model and simulate across the full UAV flight stack.

In summary, this paper makes the following contributions:

- Provides an abstraction of the fundamentals of UAV architecture which can be extended to ***any*** UAV design.
- Develops an ***open-source*** co-simulation framework that can simulate the entire UAV flight stack using models for both SITL and HITL simulations.
- Showcases the usefulness of the developed framework by ***validating*** it against real-world UAV hardware.
- Provides researchers a packaged and setup ***toolchain*** for modelling and developing UAV systems and algorithms.

*Both authors contributed equally to this research.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
PEARC '23, July 23–27, 2023, Portland, OR, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9985-2/23/07.
https://doi.org/10.1145/3569951.3597544

The rest of the paper is organized as follows. Section 2 provides details on how drone sub-systems and architecture are abstracted for our simulation framework. Section 3 discusses the simulation issues which affect current frameworks and discusses how to overcome them. Section 4 details the development of our framework along with validation. The related work and conclusions are shown in Sections 5 and 6 respectively.

2 AUTONOMOUS DRONE ARCHITECTURE

The success rate and performance of autonomous drone missions is directly linked to the underlying software and hardware flight stacks. A few studies have thoroughly investigated the inner workings of these flight stacks for specific UAV designs or applications [9, 11, 12]. However, in the context of unified software and hardware co-simulations, there is a substantial lack of abstraction which impedes the development of such a simulation framework. In this section, we detail the abstraction of a drone system which allows for the creation of a generic simulation framework which can be quickly adapted to a specific use case, if required. Figure 1 shows a drone system abstracted as three discreet layers: *Compute, Flight Controller, and Control Surfaces.*

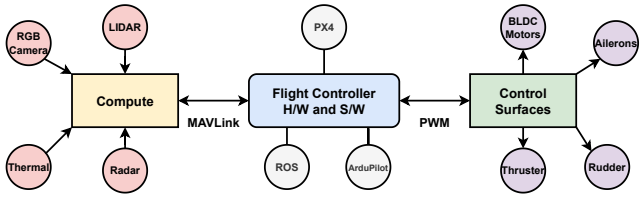

Figure 1: Abstracted Drone Architecture Overview.

2.1 Compute

The *compute* sub-system of a drone consists of various input sensors (LIDAR, camera, radar, etc.) interfacing with a compute unit such as a Raspberry Pi [7] or Nvidia Jetson [16]. These single-board-computers (SBCs) are a popular choice due to their high performance-to-weight and performance-per-watt ratios, along with a high degree of programmability and extensibility for distributed workloads. They interface with the flight controller via an open source protocol named MAVLink [21]. However, when a very high degree of performance or *real-time* processing is required, many drone architects opt for an ASIC design as it minimizes data movement costs while providing higher throughput. However, these designs are specialized for a single use case and lack the adaptability several drone workloads require.

A common approach to this tradeoff, is the use of accelerators such as GPUs and Field Programmable Gate Array (FPGA) devices, both of which allow for high performance and high programmability, albeit at a high power draw. For drone architects to accurately assess the performance vs. power tradeoff of adding such devices to their drone model, an accurate simulation framework is needed which can quantify the compute requirements of various workloads along with the specific characteristics of the underlying hardware and software stacks.

2.2 Flight Controller

The *flight controller* (FC) consists of software and hardware components. The hardware component is necessary as regular compute units lack features such as PID controllers, barometers, redundant IMUs (Inertial Measurement Units), and GPS/GLONASS receivers. It serves as the interface between the higher level compute units and the control surfaces of the drone by decoding MAVLink data packets and issuing PWM commands as shown in Figure 1. An example of such a flight controller is the Navio2 HAT [5] for Raspberry Pi pictured in Figure 2. The Navio2 is a popular choice as it is based on the open-source PX4 [17] platform and comes in a small form factor.

Figure 2: Navio2 FC [5]

The role of the flight controller software is to run *inner-loop* code [9] consisting of several control sequences such as weight imbalance correction, motor imperfection adjustments, wind gust mitigation etc. This inner-loop code is black-boxed away from the higher level compute algorithm to simplify the task of developing workloads for drones. In our simulation framework, we allow the user to choose whether to follow the same black-box methodology or to expose the control code to the compute algorithm as it will be useful for simulating heterogeneous compute-control chip designs.

2.3 Control Surfaces

The *control surfaces* of a drone consist of any device or object on the drone that helps in maintaining flight. Since our simulation framework is abstract yet adaptable, the control surfaces could be motors (for quadcopters) or ailerons, thrusters, and rudder (for fixed-wing drones). We use the detailed physics models created by Hadidi et. al [9] in our simulation framework to accommodate quadcopters, fixed-wing drones, and hybrid versions.

3 SIMULATION CHALLENGES

There are two types of simulations: Hardware-in-the-loop (HITL) and Software-in-the-loop (SITL). SITL allows for cost efficient software testing and reduces barriers between iterations, making it easier to test edge cases. In the case of UAV systems, SITL simulations generally run on systems that are vastly more powerful than the hardware systems the software will actually run on. Reduced cost combined with ease of development make SITL ideal for early stages of development. HITL, on the other hand enables software changes based on integrated hardware functionality. With the emulation of electrical inputs to sensors, HITL simulation provides a more accurate representation of a system's performance through the introduction of hardware limitations and characteristics, making it useful for testing complex missions.

Though HITL and SITL simulations are popular for drone navigation, researchers often encounter various challenges while setting up these simulations. Simulation software often require many disparate dependencies for baseline functionality and have minimal documentation and community support for outlying bugs.

In addition, connectivity issues may arise such as the simulation software failing to detect the drone due to protocol mismatches or OS specific technicalities. Several simulation software specifically use Unreal Engine [20], a large game engine code-base, for the underlying physics and overlaying graphics. These challenges affect simulation efficacy and efficiency and require time consuming troubleshooting.

However, not all simulation challenges are development based. Some are fundamental issues which plague the very premise of first developing a drone in a simulator and then prototyping it, as summarized by Mairaj et. al [13]. Most simulators are built by first taking a physics engine used for aircraft simulation and then adding UAV specific algorithms and control sequences on top of it [6]. While this approach allows for rapid creation of a simulator, it starts to fall apart once the autonomous compute component of a drone is added to the simulation, as the physics engine of an aircraft simulator is not adapted for a direct link to the compute unit. However, current state-of-art simulators struggle to simulate these configurations [13].

By building our framework with the abstraction created in Section 2 along with autonomous compute specific models, we overcome the above-mentioned bottlenecks and challenges.

4 DEVELOPING A UNIFIED FRAMEWORK

In this section, we detail how UniUAVSim was developed along with validation details and distribution plans.

4.1 Development

The framework is split into two parts: a simulation engine and a graphics engine. This was done so that simulation is no longer dependent on graphics as it can be performed using just the command line interface.

Figure 3: Drone simulated with graphics engine.

4.1.1 ***Graphics Engine.*** Since the primary focus of our work is the simulation component of our framework, we decided to use the current state-of-art graphics engine supplied by AirSim, specifically the same Unreal Engine environment that is used by PEDRA [1]. The graphics engine provides a visual aid of the current simulation status to the user as shown in Figure 3. It is important to note that the graphics engine is optional and the simulation can run without it as shown in Figure 4. This was done so that our distributed framework can be used by a wider audience who may not have access to powerful GPUs to run the graphics engine.

4.1.2 ***Simulation Engine.*** The simulation engine incorporates the most popular of the different physics and compute models [9, 11–13] and provides a starting ground to the drone architect to fine tune each parameter or change the major workings. Each model also incorporates its own physics model i.e. description of how the drone's weight, drag coefficient, thrust-to-weight ratio (TWR), etc. affect its flight characteristics.

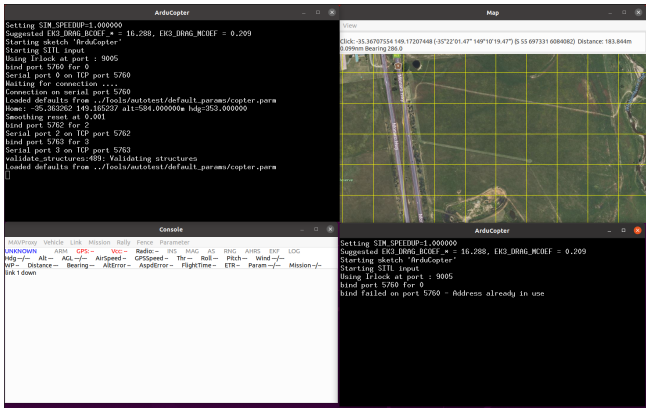

Figure 4: CLI only simulation mode.

These models are selected by the user prior to executing the simulator but the simulator is also capable of switching models mid-flight. This may be useful if a user wants to simulate a drone that was executing an object search workload but after finding the mobile object, now has to execute the object tracking workload. This ability to switch models enables our novel framework to be both generic and specialized.

4.2 Validation

To validate our drone's performance, we conducted both SITL and HITL tests, using a quadcopter drone which was assembled with open-source components pictured in Figure 5. The flight controller is a Navio2 board based off the PX4 [17] framework, running Emlid Raspberry Pi OS pre-installed with the ArduPilot flight controller. We connected this drone to a workstation based on x86_64 architecture and through tests, we ensured the drone's stability, reliability, and suitability for use in HITL or SITL simulation of drone workloads.

Figure 5: Drone HITL Validation.

4.3 Distribution

UniUAVSim is deployed through Docker [4] so that it is easily accessible to a wide audience. Docker containers provide a standard

environment that is easy to setup across contexts. Our goal is to streamline the setup process and allow for easy local development.

5 RELATED WORKS

There are several existing frameworks for the simulation of UAVs including AirSim, ArduPilot [2], PEDRA, UAV Toolbox [14], QGroundControl [18], and Gazebo [8]. However, only a few of them support SITL and HITL simulation modes while none support running both modes concurrently. Some of them such as UAV Toolbox are closed-source paid products which are more suited for commercial development.

AirSim is an open-source framework developed by Microsoft to support the simulation of drones, cars, and other vehicles. Its physics is supported by Unreal Engine and it currently supports SITL simulation for two flight controllers and HITL simulation for one. However, AirSim is unable to run both SITL and HITL simulations concurrently. Therefore, a full system simulation and validation is not feasible. However, our framework allows for unified co-simulation and enables simulation with validation. Further, the open-source version of AirSim is no longer being maintained as of 2022 [15].

ArduPilot is an open-source autonomous vehicle system that supports various types of UAVs. ArduPilot offers native SITL simulation for Linux and Windows with support for UAVs and specific documentation for a number of specific flight controllers. ArduPilot is also actively maintained [2]. However, the support for HITL simulations is lacking.

PEDRA is an open-source engine, built on top of AirSim and Unreal that allows for ease in developing reinforcement learning models. PEDRA allows for Python development and supports multi-drone environments and detailed environments for inference learning [1]. However, it does not simulate different control sequences or flight controller models.

6 CONCLUSIONS

Simulating autonomous drones remains a *hard* problem due to the tight physical and power constraints that drones operate with. In this paper, we propose and develop a fully open-source unified co-simulation framework which can simulate the entire drone flight stack by using concurrent software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations. The framework can utilize various compute and physics models of different drone designs and can produce the optimal drone configuration for a given set of constraints which was validated with real-world hardware. Lastly, the framework and toolchain is in the process of being made available to the research community to aid in future drone architecture research.

7 FUTURE WORK

We plan on further expanding the framework to support simulation of ML drone algorithms. We are also exploring the feasibility of hosting the framework as a cloud service. This would enable researchers to use less demanding systems (such as laptops) to run the simulation framework as the compute heavy tasks would be running on servers.

ACKNOWLEDGMENTS

This work is primarily supported by NSF OAC 2103951 and in part by the Rogues Gallery testbed [22] hosted by the Center for Research into Novel Computing Hierarchies (CRNCH) at Georgia Tech. The Rogues Gallery testbed is primarily supported by NSF Award #2016701. We also thank Jaewon Lee for GPGPU driver support.

REFERENCES

[1] Aqeel Anwar and Arijit Raychowdhury. 2019. Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes using Transfer Learning. *arXiv e-prints*, Article arXiv:1910.05547 (Oct 2019), arXiv:1910.05547 pages. arXiv:1910.05547 [cs.LG]
[2]ArduPilot. [n. d.]. ArduPilot. https://ardupilot.org/copter/.
[3]DJI. [n. d.]. DJI MAVIC. https://www.dji.com/mavic-3-pro.
[4] Docker. 2023. Docker Overview. https://docs.docker.com/get-started/overview/.
[5]Emlid. [n. d.]. Emlid Navio2 HAT for Raspberry Pi. https://emlid.com/navio/.
[6] Epic. 2023. The most powerful real-time 3D creation tool. https://www.unrealengine.com/en-US/.
[7] Raspberry PI Foundation. [n. d.]. Raspberry Pi 4. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. [Online; accessed 04/10/23].
[8] Kanishke Gamagedara, Mahdis Bisheban, Evan Kaufman, and Taeyoung Lee. 2019. Geometric controls of a quadrotor uav with decoupled yaw control. In *2019 American Control Conference (ACC)*. IEEE, 3285–3290.
[9] Ramyad Hadidi, Bahar Asgari, Sam Jijina, Adriana Amyette, Nima Shoghi, and Hyesoon Kim. 2021. Quantifying the Design-Space Tradeoffs in Autonomous Drones. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 661–673. https://doi.org/10.1145/3445814.3446721
[10] Intel. [n. d.]. Intel NUC. https://www.intel.com/content/www/us/en/products/details/nuc/kits/products.html.
[11] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Ninad Jadhav, Aleksandra Faust, and Vijay Janapa Reddi. 2022. Roofline Model for UAVs: A Bottleneck Analysis Tool for Onboard Compute Characterization of Autonomous Unmanned Aerial Vehicles. In *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 162–174. https://doi.org/10.1109/ISPASS55109.2022.00023
[12] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. 2020. The Sky Is Not the Limit: A Visual Performance Model for Cyber-Physical Co-Design in Autonomous Machines. *IEEE Computer Architecture Letters* 19, 1 (2020), 38–42. https://doi.org/10.1109/LCA.2020.2981022
[13] Aakif Mairaj, Asif I. Baba, and Ahmad Y. Javaid. 2019. Application specific drone simulators: Recent advances and challenges. *Simulation Modelling Practice and Theory* 94 (2019), 100–117. https://doi.org/10.1016/j.simpat.2019.01.004
[14] Mathworks. 2023. UAV Toolbox. https://www.mathworks.com/products/uav.html.
[15]Microsoft. 2021. AirSim. https://microsoft.github.io/AirSim/. https://microsoft.github.io/AirSim/
[16] NVIDIA. 2017. NVIDIA Jetson TX. https://developer.nvidia.com/embedded/jetson-tx2. [Online; accessed 04/10/23].
[17] Pixhawk. [n. d.]. Pixhawk 4. https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk4.html.
[18] QGroundControl. 2019. QGroundControl Overview. https://docs.qgroundcontrol.com/master/en/index.html.
[19]SkyDio. [n. d.]. SkyDio. https://www.skydio.com/.
[20] Unreal. [n. d.]. Unreal Engine. https://www.unrealengine.com/. [Online; accessed 04/10/23].
[21]Wikipedia. [n. d.]. MAVLink. https://en.wikipedia.org/wiki/MAVLink.
[22] Jeffrey S. Young, Jason Riedy, Thomas M. Conte, Vivek Sarkar, Prasanth Chatarasi, and Sriseshan Srikanth. 2019. Experimental Insights from the Rogues Gallery. In *2019 IEEE International Conference on Rebooting Computing (ICRC)*. 1–8. https://doi.org/10.1109/ICRC.2019.8914707