AirTwinX: A High-Fidelity Digital Twin for Advanced Air Mobility with Ray Tracing

Annoy Dey[1], Maxwell McManus[1], Zhaoxi (Josh) Zhang[1], Guanying Sun[1],
Nicholas Mastronarde[1], Elizabeth Serena Bentley[2], and Zhangyu Guan[1]
[1]Department of Electrical Engineering, University at Buffalo, USA; [2]U.S. AFRL
Email: {annoydey, memcmanu, zhaoxizh, gsun4, nmastron, guan}@buffalo.edu, elizabeth.bentley.3@us.af.mil

***Abstract*—Ensuring the safety and reliability of emerging Advanced Aerial Mobility (AAM) systems requires wireless communication to provide real-time monitoring and control information to ground stations. This greatly depends on the quality of the wireless links during aerial transit. In this demonstration, we present AirTwinX, designed to emulate flight control of flying vehicles while generating high-fidelity, context-aware models for air-to-air (AA) and air-to-ground (AG) communication links. Using GPU-accelerated ray tracing, AirTwinX can predict the quality of wireless links with near real-time updates based on the environmental geometry observed during flight. This data is then used to guide autonomous control decision-making. Additionally, the vehicle control toolchain employed in this work is based on software-in-the-loop (SITL) emulation of a commercial flight controller, enabling seamless translation of control policies from simulation to real-world hardware.**

***Index Terms*—Advanced Air Mobility (AAM), Ray Tracing, NVIDIA GPU, Software-in-the-Loop (SITL).**

I. Introduction

Advanced Aerial Mobility (AAM) has been envisioned as a transformative solution for transporting people and cargo to areas under-served by existing transportation systems, promising increased flexibility, resilience, and efficiency. However, AAM technology remains in its early stages, with a critical need for standardization across device capabilities, communication protocols, and service requirements. In particular, there is a significant gap in protocol-specific evaluations, which are essential to identify the most effective communication systems for these emerging AAM applications [1].

Furthermore, evaluating the accuracy and relevance of current toolchains for designing AAM systems remains a challenge without adequate hardware validation. Although several studies have demonstrated autonomous control systems and rigorously modeled flight dynamics and performance [2], many do not address the real world translation of simulated system performance [3]. This highlights the need for high-fidelity simulation and emulation frameworks that can integrate with hardware to evaluate control policies, communication quality, and ultimately the safety and reliability of future AAM technologies.

This work was supported in part by the National Science Foundation (NSF) under Grant SWIFT-2229563, and the U.S. Air Force Research Laboratory under Contracts FA8750-21-F-1012 and FA8750-20-C-1021.
Distribution A. Approved for public release: Distribution Unlimited: AFRL-2024-5896 on 21 Oct 2024.

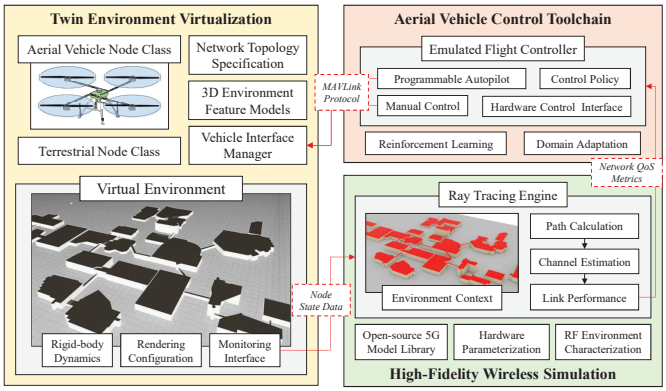

Fig. 1: Overview of Digital Twin Framework Architecture.

In this demonstration, we introduce AirTwinX, a novel digital twin framework designed to support the development and deployment of AAM applications. AirTwinX enables real-time virtualization of command and control (C2) functions within AAM systems. We showcase the integration of near real-time ray tracing capabilities within the digital twin, which provides precise network Quality of Service (QoS) estimates to inform C2 decision-making for both manual and autonomous flight control scenarios. *The main contribution of this work is the software-in-the-loop integration of high-fidelity, ray-tracing-based wireless communications modeling with commercial off-the-shelf aerial emulation software.* This integration facilitates the direct transfer of the resulting control policies from simulation environments to aerial hardware platforms, effectively reducing the sim-to-real gap.

II. AirTwinX Design

As outlined in Fig. 1, the proposed AirTwinX framework is comprised of three core components: Twin Environment Virtualization, Aerial Vehicle Control Toolchain, and High-Fidelity Wireless Simulation.

Twin Environment Virtualization. This component is designed to replicate real-world scenes with 3D structures and terrain for flying vehicle simulation, utilizing OpenStreetMap [4] for geographical data and storing the 3D scenario model in Extensible Markup Language (XML) format, while also enabling ray-tracing compatibility. The Virtual Vehicle Node Class serves as a comprehensive class that encapsulates the properties of the flying vehicle, including its rotors,

antennas, battery, and aerodynamic structure. It also specifies the Universal Scene Description (USD) file path for the flying vehicle model and the stage (i.e., the 3D virtual environment in NVIDIA Isaac Sim), facilitating the creation of a single vehicle. Aerial Fleet Specification sets the specific positions and orientations of the vehicle. The Vehicle Interface Manager enables realistic vehicle simulation, multirotor control, and sensor simulation through the Pegasus Simulator [5] Plug-in. The virtual environment is visualized in NVIDIA Isaac Sim by configuring rendering settings, including cameras and lighting. Finally, the monitoring interface transfers the flying vehicle sensor's data to the High-Fidelity Wireless Simulator to inform the physical and geometric configurations for ray-tracing-based channel modeling and estimation.

Aerial Vehicle Control Toolchain. At the core of the toolchain is an Emulated Flight Controller (EFC) to manage flight operations and user input. The EFC is designed to provide both manual and autonomous navigational control through four sub-modules: Programmable Autopilot, Manual Control, Hardware Control Interface, and Control Policy.

The Programmable Autopilot, which leverages the PX4 autopilot firmware [6] for flying vehicles, manages all autopilot algorithms and is facilitated by the PyMAVLink Python module. Manual Control is jointly facilitated by QGroundControl (QGC) [7] and Pegasus Simulator Plug-in. The former enables users to manually control the flying vehicle, upload flight missions, set waypoints, and monitor sensor data, while the latter allows users to define custom software-defined controllers or connect hardware controllers compatible with the PX4 SITL firmware. The Hardware Control Interface provided by the Pegasus simulator, enables users to write scripts that control the backend, receive the current state of the vehicle and its sensors, and compute control inputs for the vehicle's rotors.

Autonomous control policies can be trained in AirTwinX's Control Policy Module by using various reinforcement learning (RL) algorithms to enhance decision-making. These policies can be evaluated in various virtual environments through domain adaptation techniques, such as robust learning, which can improve the performance of policy transfer from virtual to real world scenarios [8].

High-Fidelity Wireless Simulation. This is an advanced simulator for modeling wireless communication links between the flying vehicles and EFC. The simulator has four components that collectively enhance its fidelity: the Ray Tracing Engine, the Open-Source 5G Model Library, Hardware Parameterization, and RF Environment Characterization. It simulates the wireless communication links between the flying vehicle and ground stations, taking into account signal propagation, channel characteristics, and real-time interactions. The simulation utilizes the Sionna Ray Tracer [9] API for high-fidelity wireless channel modeling. Specifically, channel impulse responses (CIRs) are computed using the ray tracer for given networking scenarios. These CIRs are then converted into channel coefficients, along with Doppler shift and other parameters for channel estimation and link performance evaluation. The resulting channel information is subsequently forwarded to the Aerial Vehicle Control Toolchain, enabling the implementation of QoS-aware navigational control policies.

III. Demonstration

We will demonstrate the capabilities of the proposed digital twin framework by navigating a 3D model of the North Campus of the University at Buffalo. We consider three scenarios with different navigational control methods to highlight the flexibility of AirTwinX for AAM research, as follows.

Manual Control using QGC. In this demo, we simulate the wireless link between a single aerial node with a single ground station, tracking network QoS in real time for the duration of the flight. The flying vehicle will be operated under manual control, using QGC to demonstrate compatibility with well-known aerial control frameworks while simultaneously estimating near real-time network QoS through ray tracing.

Autonomous Navigation. We showcase the use of AirTwinX to determine aerial network QoS in the case of a single terrestrial transmitter. In this demo, we deploy several aerial nodes that seek to minimize flight time between several landmark locations throughout the environment. Each node is guided by a navigational control policy which has been trained offline in AirTwinX using an Echo State Network (ESN)-based RL algorithm. The policy aims to minimize flight time between landmark locations and avoid collision among aerial nodes. We adopt ESN because of its computational efficiency and excellent generalization of time-domain features.

QoS-Aware Flight Planning. We further showcase the safety evaluation in practical AAM scenarios using AirTwinX. To this end, we consider a network QoS-aware autonomous control policy, which seeks to maintain uplink bit error rate (BER) above a pre-configured threshold estimated based on existing cellular network standards. In this demo, we repeat the previous experiment, replacing its control policy with a pre-trained one designed to maximize network QoS for the duration of each flight.

References

[1] S. Chen, A. D. Evans, M. Brittain, and P. Wei, "Integrated Conflict Management for UAM With Strategic Demand Capacity Balancing and Learning-Based Tactical Deconfliction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 10 049–10 061, Aug. 2024.

[2] C. Park, G. S. Kim, S. Park, S. Jung, and J. Kim, "Multi-Agent Reinforcement Learning for Cooperative Air Transportation Services in City-Wide Autonomous Urban Air Mobility," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4016–4030, Jun. 2023.

[3] M. McManus, Y. Cui, J. Hu, S. K. Moorthy, Z. Guan, N. Mastronarde, E. S. Bentley, and M. Medley, "Digital Twin-Enabled Domain Adaptation for Zero-Touch UAV Networks: Survey and Challenges," *Computer Networks*, vol. 236, p. 110000, Nov. 2023.

[4] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[5] M. Jacinto, J. Pinto, J. Patrikar, J. Keller, R. Cunha, S. Scherer, and A. Pascoal, "Pegasus simulator: An isaac sim framework for multiple aerial vehicles simulation," in *Proc. of International Conference on Unmanned Aircraft Systems (ICUAS)*, Crete, Greece, June 2024.

[6] Dronecode Project. PX4 Autopilot. [Online]. Available: https://docs.px4.io/main/en/

[7] ——. QGroundControl. [Online]. Available: https://qgroundcontrol.com

[8] M. McManus, Z. Guan, N. Mastronarde, and S. Zou, "On the Source-to-Target Gap of Robust Double Deep Q Learning in Digital Twin Enabled Wireless Networks," in *Proc. of SPIE Big Data IV: Learning, Analytics, and Applications*, Orlando, Florida, United States, April 2022.

[9] J. Hoydis, F. A. Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling," in *Proc. of Globecom Workshop on Artificial Intelligence Enabled Next Generation Wireless Networks*, Kuala Lumpur, Malaysia, Dec. 2023.