In: 2024 IEEE Aerospace Conference, 02-09 March 2024, Big Sky, MT, USA
DOI: 10.1109/AERO58975.2024.10521333

A Study on Co-simulation Digital Twin with MATLAB and AirSim for Future Advanced Air Mobility

Lorenzo Turco, Junjie Zhao, Yan Xu, and Antonios Tsourdos
School of Aerospace, Transport and Manufacturing
Cranfield University
Bedfordshire, MK43 0AL, UK
Email: {lorenzo.turco.260, junjie.zhao, yanxu, a.tsourdos}@cranfield.ac.uk

***Abstract*—The exponential growth in Unmanned Aerial Vehicle (UAV) operations highlights the need for reliable and efficient airspace management. Ensuring the integrity and reliability of UAV Traffic Management (UTM) systems requires extensive testing, verification and validation. Overcoming these challenges is essential to guarantee that UAVs can be successfully integrated into operational airspace while maintaining the highest standards of safety and effectiveness. The development and improvement of virtual environment testing capabilities are critical to the advancement and rapid deployment of UTM services; however, building a high-fidelity digital environment that enables the development, verification and validation of UTM solutions in a compliant, scalable and sustainable manner remains demanding. By integrating MATLAB, Simulink, AirSim, Unreal Engine and Cesium, this study aims to extend the performance and functionality of a Digital Twin (DT) system. This integration leverages the sophisticated modelling capabilities of MATLAB and the advanced 3-dimensional (3D) simulation capabilities of AirSim. It also aligns with emerging trends in the aerospace sector, including autonomous flight and advanced sensing technologies. Firstly, co-simulation is explored as a potential technique to overcome the limitations of individual simulation software available on the market. The general principles are then applied to define and model a communication interface between the software selected to build the DT ecosystem. This interface is then evaluated by proposing practical and achievable test cases. Only after validating the proposed co-simulation framework, further experiments are introduced aiming at improving the vehicle and sensor model performance to obtain more accurate synthetic data. A series of experiments of increasing difficulty is proposed, starting from co-simulating a single sensor (GNSS, INS, LIDAR and camera) up to co-simulating the entire aircraft system for the development of intelligent algorithms such as waypoint following and collision avoidance. Ultimately, this research contributes to the practical application of co-simulation, providing insight into its capabilities and potential influence on the advancement of autonomous aircraft and DT systems.**

Keywords: MATLAB, AirSim, Unmanned Aerial Systems, Autonomous Flight, Vehicle Dynamics, Sensor Modelling, Flight Data Analysis

TABLE OF CONTENTS

1. INTRODUCTION 1
2. METHODOLOGY 4
3. TOOLS AND PLATFORMS 4
4. EXPERIMENTAL RESULTS AND VALIDATION 7
5. CONCLUSIONS 15
ACKNOWLEDGEMENTS 15
REFERENCES 16
BIOGRAPHY 18

1. INTRODUCTION

A. Background and Problem Statement

The exponential growth in UAV (Unmanned Aerial Vehicle) and Advanced Air Mobility (AAM) vehicle operations has raised concerns about safe and efficient airspace management. Testing, verifying and validating solutions related to UAV Traffic Management (UTM) become paramount to ensure the integrity and reliability of these systems. The complexity of integrating the cited vehicles into the existing airspace, along with the evolving regulatory landscape, poses significant challenges [1]-[2]. Overcoming these challenges is essential to enable the seamless integration of UAV and AAM operations while maintaining the highest standards of safety and efficiency in an airspace environment that has been developed to accommodate crewed aircraft. The proliferation of unmanned vehicles can be attributed to technological advancements, increasing demand for autonomous systems, and the potential for transformative applications across various industries. In the same way, AAM is an emerging concept that envisions the integration of innovative technologies and systems to enable on-demand, safe, and efficient air transportation. AAM expands the capabilities of traditional aviation by incorporating electric vertical take-off and landing (eVTOL) aircraft, Urban Air Mobility (UAM), and autonomous flight operations [3]. It aims to revolutionise the transportation industry by providing efficient point-to-point air travel, reducing road congestion, and addressing urbanisation challenges. By taking advantage of pioneering technologies, AAM systems have the potential to offer faster travel times, reduce carbon emissions, and enhance overall mobility.

Despite the numerous benefits, some issues must be tackled to adopt these novelty solutions successfully. One of the critical challenges is the development of infrastructures, including vertiports and charging stations, to support their operations. Additionally, regulations and policies need to be established to ensure the safe integration of UAV and AAM vehicles into existing airspace. Their deployment and integration require adherence to national and international regulations. Programmes such as SESAR (Single European

© 2024 IEEE. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/AERO58975.2024.10521333. Please refer to any applicable publisher terms of use.

Sky ATM Research) and NextGen (Next Generation Air Transportation System) play crucial roles in shaping the regulatory landscape [4], [5]. These frameworks aim to develop common standards, procedures and technologies to support UAVs' sustainable and reliable integration into the airspace system. National Civil Aviation Authorities (CAA) have also implemented specific regulations and guidelines to ensure compliance and manage the unique challenges associated with UTM services [6].

To deal with the testing, verification and validation problem in UTM, a proposed solution is a DT. A DT is a virtual representation of a physical system or process that enables real-time monitoring, analysis, and testing [7]. It serves as a bridge between the physical and digital worlds, providing a realistic and controlled environment to model, test, and certify the performance of UAV and AAM systems.

B. Related Work

A DT comprises three key components: the physical entity, the virtual model, and the communication interface that connects the two. The physical entity represents the real-world system, such as an aircraft or a network of UAVs. The virtual model is a digital representation that includes detailed information about the system's structure, behaviour, and operational characteristics. The communication interface facilitates data exchange between the physical entity and the virtual model, ensuring real-time synchronisation and feedback [8].

The DT concept encompasses a wide range of technologies, including data analytics, simulation, and advanced modelling techniques. Aerospace systems' design and development processes can take advantage of this technology. Enhanced simulations and investigations can be performed, enabling early detection of potential issues, optimisation of designs, and reduction of development cycles [9]. Furthermore, patterns, anomalies and performance deviations can be easily identified thanks to the DT's real-time monitoring capabilities [10]. By applying advanced analytics and Machine Learning (ML) algorithms, a DT can also provide insights for optimising the system's operations, improving efficiency, and reducing downtime. For example, in the case of UTM applications, a DT can monitor multiple UAVs simultaneously and provide real-time information on their locations, trajectories, and potential conflicts. By continuously monitoring and analysing the behaviour of physical systems, it can identify inefficiencies, suggest improvements, and support decision-making processes. Overall, it can optimise routing algorithms, airspace utilisation, and resource allocation [11]–[13].

Moreover, DT technology offers a powerful platform for training purposes. By replicating physical systems' behaviour, operators, pilots, and maintenance personnel can train in a virtual environment, reducing the need for costly and time-consuming physical training scenarios. To sum up, by leveraging a DT environment, stakeholders can assess the behaviour of a custom flying vehicle in various scenarios, evaluate the effectiveness of ATM and UTM solutions, and identify potential risks or issues before deployment in real-world operations [14]. This approach offers a cost-effective and efficient way to iterate and refine solutions, reducing the time and resources required for physical testing and certification processes. Despite its numerous benefits, this emerging technology also has certain limitations. One identified limitation is the complexity of developing and maintaining accurate virtual models that faithfully represent the physical system. Gathering and processing real-time data from sensors and integrating it into the virtual model can be demanding [8]. Additionally, the availability of high-quality data is crucial for accurate simulations and studies within the virtual environment. Another limitation is the computational resources required to run real-time simulations, especially in complex multi-agent operations. Addressing these challenges will pave the way for successfully developing and utilising a DT for aerospace applications.

Cranfield University, renowned for its pioneering research and advancement in emerging technologies, has achieved a significant milestone by successfully creating, testing, and assessing a DT prototype [15]. This noteworthy achievement was accomplished using state-of-the-art software such as Unreal Engine, AirSim, and Cesium. The outcome of this remarkable endeavour has stimulated subsequent research efforts with the objective of enhancing the existing capabilities and integrating novel functionalities, particularly in the domains of co-simulation, mixed reality, and intelligent solutions. With regards to mixed reality and intelligent solutions developments, these were enhanced in [16] through a robust software architecture and assessed through several multi-agent and UTM use cases. Intelligent solutions integration enables DTs to adapt and respond to dynamic and changing conditions, making them valuable tools for system monitoring, control, and optimisation. Notably, some examples in the literature have also achieved promising results in Control, Guidance and Navigation domains [17]–[22].

This work aims to push further the boundaries of the prototype developed, focusing on co-simulation features. Co-simulation is intended to integrate and coordinate multiple simulation models or software tools to create a unified and synchronised simulation environment. It enables the simulation of complex systems involving multiple components or subsystems, each simulated using specialised software or modelling techniques [23].

Co-simulation allows for the smooth exchange of data and information between different simulation platforms, enabling a holistic analysis of system behaviour and performance. In the context of DTs, co-simulation plays a decisive role in replicating real-world scenarios and interactions between different components or entities within a system. By combining models from diverse domains, such as physics, control systems, and data analytics, co-simulation enables a more accurate and comprehensive representation of the physical system being modelled [24].

Several studies in the literature have been conducted to provide a clear and schematic overview of the available simulators that can be used for UAS performance assessment as stand-alone or in a co-simulation framework. Not only UAV simulators but also network simulators, UAV emulators and some UAV frameworks are analysed in one of the most comprehensive studies [25]. The characteristics of the developed prototype and the features provided by AirSim have been carefully examined. According to various sources, the main identified limitation in AirSim is that simulations are limited to quadcopters in X configuration [25]–[27]. However, since it is an open-source project, adding or improving new functions is possible. In fact, the Project AirSim simulator follows a modular architecture designed to be extensible [27], as described in more detail in the following. It has also been designed to encourage the development of intelligent solutions for autonomous aerial vehicles. Therefore, to facilitate the immediate application of the algorithms and solutions identified in the real world, the sensor data synthetically generated in the virtual environment and used to train the ML algorithms must be as close as possible to the data generated by real sensors. For this reason, it was also decided to study the available sensors to identify and, if possible, propose possible improvements.

In summary, the literature research has been re-examined from a different perspective looking for simulation tools with features which potentially mitigate the AirSim's limitations highlighted:

- Simple vehicle physics and dynamics
- Basic sensor models
- A limited number of UAV models

The benefits and constraints that led to choosing to adopt MATLAB and Simulink to compensate for the limitations mentioned above are shown in Table 1 1. Rendering and visualisation capabilities are not considered as a factor in this analysis because AirSim and Unreal Engine are the only simulators that support Motion of Capture (MOTCAP), which allows the simulation of the UAV's natural movement [26], giving them an edge over their competitors in this domain.

C. Contributions

This paper presents a novel Modelling & Simulation (M&S) approach proposing a co-simulation methodology aimed at enhancing the existing DT prototype. More accurately, the ambition is to integrate the functionalities offered by MathWorks toolboxes, starting from the conceptual co-simulation DT framework developed in [33]. The primary objective is to strengthen the capabilities of the DT for the long-term development, testing and validation of UTM solutions. Specifically, this work focuses on augmenting the simulation capabilities of the DT from the perspective of the vehicle and the scenario in which it is deployed. Its functionalities will be demonstrated and validated through a meticulous analysis of the experimental results data coming from the demonstrative use cases designed using MATLAB, Simulink, AirSim, Unreal Engine and Cesium. While the initial application is targeted at a single vehicle, the modular and scalable nature of the proposed co-simulation architecture implies its potential to accommodate more complex and challenging scenarios. It is important to note that at this stage of development, air traffic control (ATC) systems and human actors are not addressed. Instead, the focus is on the vehicle dynamics, environmental factors, physics and sensors inherent to the modelled vehicle.

The development also covers some specific emerging trends that have been identified. One prominent trend is related to autonomous flight operations, including obstacle detection and avoidance and advanced data analytics [34]. A test case has been performed which shows that the proposed implementation can accomplish autonomous tasks when appropriately tuned. On the other hand, integrating advanced sensing technologies, such as LiDAR (Light Detection and Ranging) and multispectral imaging, is gaining traction in the UAS market. These sensing technologies promise to enable high-precision mapping, 3D modelling, and remote sensing applications [35]. In the current evolution a case study based on LiDAR point cloud data demonstrates that, with the right effort, this technology can be handled in a co-simulation environment.

The updated co-simulation framework significantly elevates the DT prototype's simulation capacities. The key advantage of the proposed architecture is its ability to exploit the 3D visualization modules, UAS dynamics, and sensor models available in different simulation tools. This integration represents a significant step forward, creating a high-fidelity simulation environment that streamlines the modelling, testing and validation of AAM and UAV systems using DT technology. It not only enhances current capabilities, but also paves the way for continued refinements and enhancements in the future.
The following research questions will be addressed to achieve the stated goal:

- Extending the Co-Simulation DT Framework:
  - Presenting a prototype integrating AirSim and MATLAB/Simulink to cover aerodynamic models, control systems, and synthetic sensor models.
  - Describing the capabilities and functionalities of the proposed co-simulation DT framework.
  - Evaluating the effectiveness of co-simulation in enhancing the DT system's functionality and performance.
- Demonstrating Practical Solutions:
  - Explaining how AirSim's 3D simulation capabilities and MATLAB's vehicle modelling tools are effectively integrated.
  - Illustrating the benefits of using MathWorks' libraries for path planning, obstacle avoidance and sensor modelling
- Facilitating Future Autonomous Aerial Systems:
  - Exploring how the integration of AirSim and MATLAB contributes to the advancement of UAS operations in line with the AAM concept.

By addressing these research questions, this study seeks to demonstrate the practicality and effectiveness of this co-simulation approach and its potential impact on future autonomous aerial systems.

D. Paper Structure

The remaining part of the paper is organised as follows: Section 2 presents the proposed methodology to extend the co-simulation framework; Section 3 briefly reviews the architecture of AirSim and the communication interface used to enable the co-simulation. Then, Section 4 presents and discusses the experiments and results on the enhanced DT capacities based on co-simulation with AirSim and MATLAB/Simulink. Finally, in Section 5, the main conclusions of this work are summarised, and the future direction of this research is proposed.

2. METHODOLOGY

The co-simulation process involves several essential steps that collectively contribute to its success. Firstly, the complex system is broken down into subsystems based on the underlying physics, leading to the creation of functional models for each subsystem within individual simulators. Subsequently, connections or signals between the subsystems are established to enable the exchange of relevant information. In [33], a generic DT co-simulation framework for autonomous UAS operations is presented and tested, combining BlueSky's traffic simulation with AirSim. Furthermore, in [36] an integrated simulator is presented based on three simulation tools: Gazebo, ArduCopter, and ns-3. The proposed solution introduces a software middleware that coordinates their functionalities and ensures a common notion of simulated time during the simulation. An in-depth examination of the common features of these remarkable examples reveals that, in order to realise the full potential of the DT framework, an iterative process must be adopted for the development of complex systems and the design of co-simulations. This process involves three key steps: modelling, integration and testing.

In the modelling step, distinct elements representing various aspects of the physical system are selected from the existing suite or developed for use in subsequent experiments. Then, in the integration step, the focus moves on transforming these heterogeneous models into interoperable components, ensuring that data exchange between M&S software is continuous and that simulators' dynamics are compatible. Specific methods of coupling and communication interfaces need to be defined. Choosing the appropriate mechanisms for inter-process communication, such as shared memory or TCP/IP, further ensures efficient data exchange between interconnected simulators during the combined simulation. Overall, this approach allows for utilising models from diverse domains and creating a comprehensive representation of the modelled physical system. It is implicit that an accurate sampling and signal transmission strategy must be in place for this methodology to be effective. A well-planned sampling strategy allows each solver to sample connected signals at specific intervals, effectively depositing the relevant information into a Co-Simulation Buffer. This buffer then facilitates the orderly data exchange between the different simulators, allowing multiple simulators to be effectively coupled.

After a brief introduction to the foundational concepts of co-simulation, it is imperative to delve deeper into these notions in order to establish a tailored methodology specific to the intended purpose. While the chosen simulation tools and their rationale have already been defined, a more complex integration strategy deserves to be articulated. A refined architecture, going beyond the elementary co-simulation steps, is shown in Figure 1. This diagram meticulously outlines the integration of the selected simulation tools to enhance a specific layer of the DT prototype, the layer associated with unmanned agents. Complementing this layer are four additional layers within the DT prototype that delineate the digital environment and the framework for UTM and UAS operations, as described in detail in [37]. AirSim and MATLAB/Simulink jointly provide a UAV systems characterisation within the defined layer. In particular, three models are displayed using gradient colours to indicate the potential for dual design and resolution by both simulators. Vehicle, Sensor and Environment models are ably linked via the predefined co-simulation buffer whenever the configured scenario involves both simulation engines. Figure 2 provides an illustrative graphical outline of how events flow within the aforementioned communication interface. In this particular instance, the MATLAB and AirSim execution flows are highlighted, each teeming with instructions that converge into the general flow before being channelled through the server module to the rendering and visualisation module. The example provided demonstrates the implementation of a sensor module through MATLAB. The demarcated time intervals underline the temporal progression inherent to co-simulation. Obviously, to avoid communication latency and ensure real-time execution, the processing of the commands within MATLAB and AirSim must occur at an increased frequency.

3. TOOLS AND PLATFORMS

A. Project AirSim Architecture Exploration

The successful DT prototype, developed in [16], relies on the Unreal 4.27 graphical engine for 3D scene visualisation, while AirSim simulated UASs and their operations within the digital world. AirSim, an open-source simulator developed by Microsoft Research, proved instrumental in creating realistic and dynamic UAS simulation platforms, supporting a wide range of testing scenarios [38]. It is built on AirLib, a library which comprises essential elements like the Physics Engine, Sensor Models (Barometer, IMU, GPS, and Magnetometer), Vehicle Models, and API-related files. These components determine the foundation for comprehensive simulations and testing in the virtual environment. Vehicle, environment and sensor models, physics engine, rendering engine, and public API layer collaborate to accurately calculate forces, torques, and vehicle data.

Table 1: Simulation Tools Survey

Tool	Ref	Relevant Features	Benefits	Constraints
MATLAB/ Simulink	[28]	Dynamic system modelling and simulation capabilities, including UAV, ROS, and Computer Vision toolboxes	• Well-suited for control system design and analysis • Extensive toolboxes and libraries for control algorithms and signal processing	Limited 3D visualisation and environment interaction compared to dedicated UAV simulators
Gazebo	[29]	Powerful 3D robot simulator with a physics engine, widely used in robotics and UAV research	• Realistic physics-based simulation and accurate UAV dynamics modelling. • Supports sensor simulation and testing for perception algorithms. • Extensive community-contributed plugins and models available.	Overwhelming initial configuration and learning
XPlane	[30]	Commercial flight simulator with a focus on realistic flight dynamics for various aircraft, including UAVs.	• Offers a realistic and accurate flight model. • Large community and extensive add-ons available. • Modelling and simulation capabilities	Baseline lacks UAV-specific features Additional purchases required
JSBSim	[31]	Open-source flight dynamics model that can be integrated into various simulation frameworks.	• Highly customisable and physics-based flight dynamics model. • Accurate UAV control algorithm design • Flexible to custom dynamics and control algorithms.	Complex initial setup and configuration
jMAVSim	[32]	Lightweight multirotor UAVs simulator integrated with the MAVLink protocol, commonly used in the PX4 ecosystem	• Well-suited for testing and developing PX4-based UAV control algorithms. • Quick prototyping and iterative development. • Integrated MAVLink communication	Limited to UAVs compatible with the PX4 flight controller.

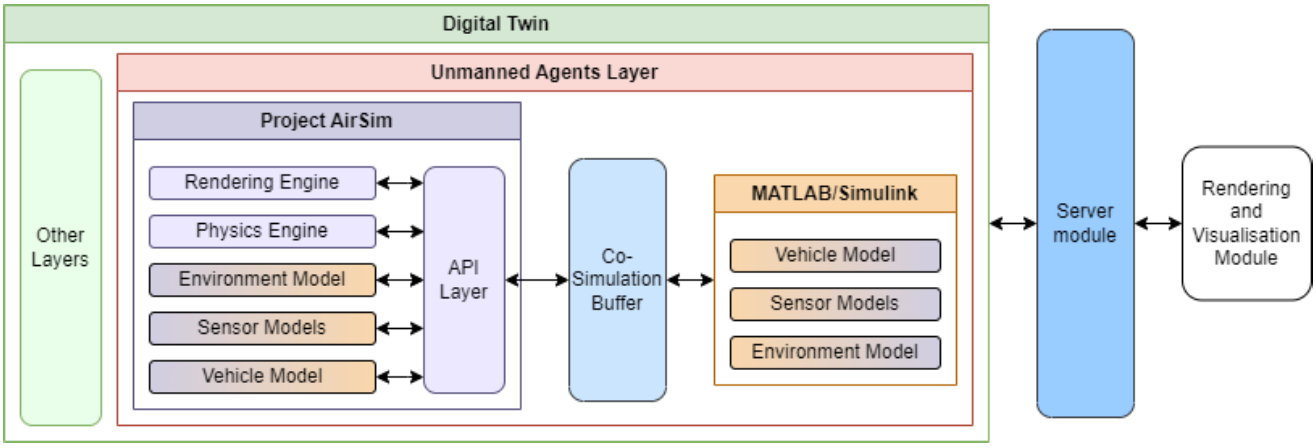


Figure 1. Detailed co-simulation architecture (inspired by [37])

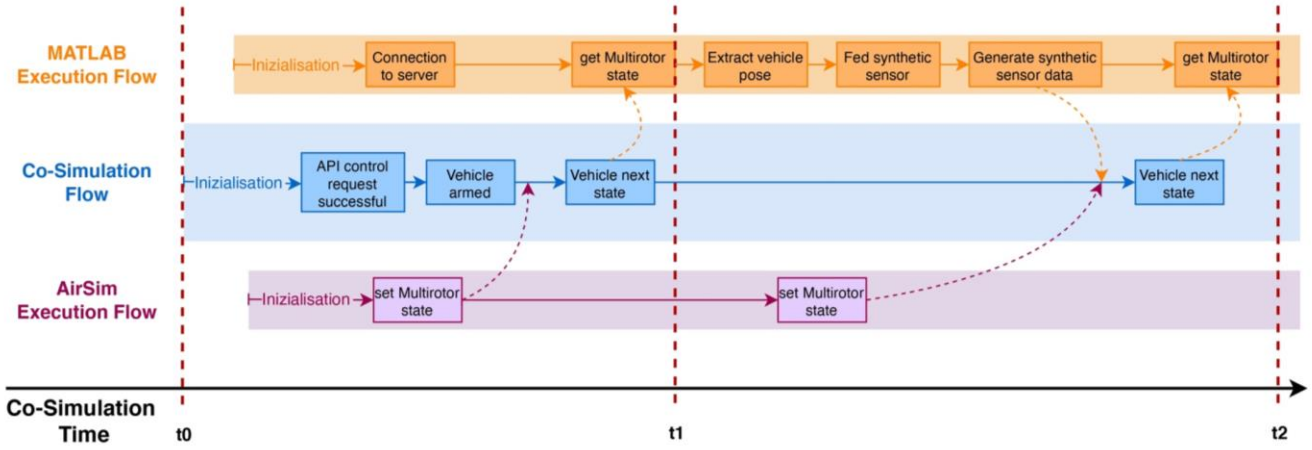


Figure 2. Detailed co-simulation buffer instructions flow

The inclusion of environmental models, accounting for gravity, air density, air pressure, magnetic field, and global coordinates, further enhances simulation accuracy. The result is a precise representation of the kinematic states of vehicles and other bodies within the simulated scene, which the sensor models then process to generate sensor data. The default API layer provides substantial functionality through a high-level interface and facilitates interaction with the Unreal environment through Remote Procedure Call (RPC). Starting from the analysis of the architecture in Figure 3, all the depicted modules were studied and evaluated. The aim was to overcome the identified limitations using the UAV toolbox available in MATLAB/Simulink [39]. The leading role of the API Layer as a bottleneck for information and data exchange with all other modules was also identified. The investigation included the following: Sensor, Vehicle and Environment Models.

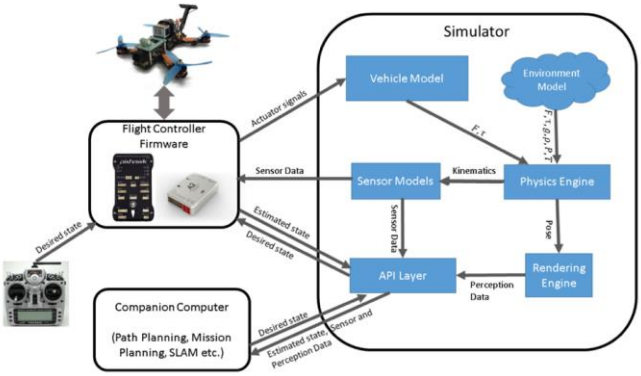


Figure 3. AirSim architecture illustrating the core components and their interactions [38]

Starting with the Sensor Module, the first perceived limitation is that the quality and accuracy of AirSim's built-in sensor data depend on analytical models. Therefore, they do not behave according to the neutral or hostile scenario in which they are simulated. In particular, if we consider the navigation system as consisting of the IMU and the GPS, both are modelled from ground truth data estimated by the physics engine to which two components are added, white noise and a bias drift additive term, to reproduce the noise that affects real sensors. In contrast to navigation systems for aerospace applications, where fault-tolerant design and redundancy play a crucial role, the navigation sensors are coupled and share the same set of inputs.

Moving on to the Vehicle Model, only vehicles based on rotary wings, such as quadrotors, hexacopters and octocopters, are available. In addition, it does not have the flexibility required for the control of individual motors. This limitation makes it unsuitable for a custom and adaptive implementation of a flight controller.

B. MATLAB/Simulink Toolboxes

Specific products for designing, analysing, and certifying complex aerospace systems demonstrate MathWorks' commitment to the aerospace industry. Critical to the development and delivery of this work have been the Aerospace Blockset, Aerospace and UAV toolboxes. Their use has enabled high-fidelity modelling, simulation and analysis of airborne platforms.

Out-of-the-box, default architectures are provided, offering the design of reusable vehicle models. These serve as indispensable tools for various tasks in the aerospace domain. They adapt for purposes ranging from flight and mission analysis, intricate mission design, to the development of innovative Guidance, Navigation and Control (GNC) algorithms. Furthermore, they find application in software integration and hardware-in-the-loop (HIL) testing for a wide spectrum of autonomous flight scenarios, including advanced radar and communications applications. UAV Toolbox similarly enables such capabilities for specific UAV or drone applications. Some of the blocks used for the improvement of DT capabilities through MATLAB/Simulink and AirSim co-simulation are shown in Table 2.

By integrating MATLAB/Simulink and Project AirSim, users can benefit from the strengths of both software solutions and get the best of both worlds. Simulink provides model-based development workflows and a comprehensive

suite of robust tools tailored to control system and physical model development. Integrating them provides greater flexibility and accuracy than Project AirSim's built-in, simple, fast physics model. It also allows designers to use their own custom Simulink dynamics models.

Table 2. Blockset Implemented Models

Block	Ref	Description
6DOF	[40]	Six degrees of freedom equations of motion are represented using Euler angles.
Guidance Model	[41]	Represents a small UAV guidance model that estimates the UAV state based on control and environmental inputs. The model approximates the behaviour of a closed-loop system consisting of an autopilot controller and a fixed-wing or multirotor kinematic model for 3-D motion.
Path Manager	[42]	Sequentially navigates between mission points specified as inputs to calculate mission parameters for a UAV.
Obstacle Avoidance	[43]	Uses distance sensor data and target position to calculate an obstacle-free path
Orbit follower	[44]	Based on the current pose, it generates course and yaw controls to follow a circular path around a point of interest.
INS sensor	[45]	Simulates an INS sensor replicating the generation of position, velocity, and orientation data while introducing noise interference aligned with the inputs provided
GPS sensor	[46]	Based on input position and speed, it outputs noise-corrupted GPS measurements.

C. Integration Framework: Co-Simulation Tools Interface

Following the strategy outlined in Section 2 for realising a successful DT co-simulation framework, after defining the simulation tools to be used and identifying the models that can be developed with them, the next step is to define a communication interface that allows effective integration between the two.

In order to achieve the latter, different options have been considered and examined, and the main characteristics of each are presented in Table 3. From the survey conducted, it was concluded that the best method of demonstrating full co-simulation capabilities, dealing with project time constrain, albeit at the expense of theoretically better performance, was to import the Python API layer of AirSim into MATLAB. Therefore, the first two methods listed in Table 3 were exploited for extending the co-simulation capabilities from MATLAB to Simulink as well.

This has resulted in the development and integration of new capabilities into the Digital Twin prototype, such as:

- Use of MATLAB as a co-simulation tool for data pre-processing, visualisation, and high-level control logic
- Near real-time simulation in Simulink with custom control algorithms and dynamics models
- Smooth integration with Simulink models, enabling co-simulation among AirSim and other Simulink blocks.

More specifically, communication between MATLAB and AirSim's Python API layer is handled by Inter-Process Communication (IPC), a key enabler for seamless collaboration between co-simulations running on the same or different machines/platforms. IPC uses TCP sockets to ensure efficient communication between processes running solvers, promoting compatibility and scalability across different platforms. By enabling data exchange and synchronisation, IPC facilitates the integration of different simulation tools.

In contrast, a different approach was introduced in [47] where an S-Function was employed to represent the AirSim quadrotor model. This approach empowers Simulink to interface effectively with the AirSim quadrotor model by accommodating general systems of equations with inputs, outputs and states, thus facilitating the communication and harmonisation of data between the two simulation tools.

4. Experimental Results and Validation

A. Co-Simulation Framework Validation

Validating the co-simulation process poses demanding challenges due to the complexity of the models involved. Extensive testing was required to ensure accuracy and compatibility between AirSim and MATLAB/Simulink. Comparing co-simulation results with full AirSim simulations serves to establish their consistency and reliability, guaranteeing the fidelity of the DT system. Solutions to the equations of motion, i.e., vehicle position and orientation, sensor and communication data, were given particular emphasis. The first step to validate the proposed framework and verify the correct functioning of the chosen integration methodology was to use AirSim as the main executor and MATLAB as the observer. Specifically, AirSim was employed as a plugin for Unreal, and the flight commands to define the trajectory to be followed were sent using the Python APIs and, thus, the Python Client provided by the Project developers. In accordance with the chosen methodology, MATLAB was connected to the same server in order to read the simulated vehicle states in real-time. A schematic representation of this setup is shown in Figure 4.

The secondary objective of this experiment was to verify that MATLAB could be used as a co-simulation tool for collecting, analysing and plotting data, as described in Section 3.3. The structure of the MATLAB code used to run the experiment is briefly illustrated by a flowchart in Figure 5. The code used as input commands for the quadrotor simulation was orbit.py [48]. It includes the take-off, the reaching of the target altitude, the execution of three circular orbits with a given radius and the landing.

Table 3. Integration Methodologies Review

Methodology	Description	Benefits	Limitations
Using the Python API in MATLAB	Use MATLAB for high-level data analysis and visualisation and take advantage of the full AirSim Python APIs.	• Direct access to the full capabilities of the AirSim Python APIs. • Flexibility to use MATLAB for high-level data analysis and visualisation while leveraging Python for low-level control and simulation tasks. • MATLAB can act as a bridge between AirSim and Simulink, enabling integration with both tools.	• Requires setting up a Python environment within MATLAB. • Performance may not be as efficient as using custom C++ S-Functions for real-time simulations. • Interfacing between MATLAB and Python may introduce communication overhead.
MATLAB Functions in Simulink	Create custom MATLAB functions for Simulink models to set vehicle poses, control vehicles, or collect simulation data using the AirSim Python APIs to interact with the simulation environment.	• Easy to implement and integrate MATLAB scripts into Simulink models. • Can leverage the full capabilities of the AirSim Python APIs from MATLAB. • MATLAB functions can be used to control vehicles and perform data analysis in real time.	• Performance might be less efficient than using a custom C++ S-Function or direct API access. • Limited control over the simulation environment compared to C++ S-Functions.
Simulink External Mode	The External Mode feature enables real-time data exchange between Simulink and external environments, including Python scripts, connecting Simulink models to Project AirSim.	• Easy integration with Simulink models. • Real-time data exchange with Project AirSim through External Mode. • Simulink's user-friendly graphical interface for control system design and simulation. • MATLAB and Simulink can be used for data analysis and visualisation.	• Limited control over the simulation environment compared to direct API access. • External Mode may introduce communication overhead. • Performance may be affected if the simulation is complex or requires high-frequency updates.
Custom C++ S-Function	The S function can encapsulate the AirSim C++ API and provide a direct interface between Simulink and AirSim, providing fine-grained control over the simulation while achieving real-time performance.	• High performance and low-latency interaction with Project AirSim. • Deep control over the simulation environment. • Possibility to integrate custom physics models and control algorithms directly into the simulation.	• Requires expertise in C++ and Simulink S-Functions for implementation. • Increased complexity compared to other integration methods.
ROS Interface	MATLAB and AirSim have Robot Operating System (ROS) message communication capabilities. MATLAB Robotics System Toolbox can send and receive messages to and from AirSim to control the simulation.	• Integration with Project AirSim using ROS messages. • Can take advantage of the ROS ecosystem for additional functionalities.	• Requires setting up a ROS environment and understanding ROS message communication. • Additional overhead due to ROS message passing.

The output obtained from the implemented MATLAB code is shown in Figure 6.

As a result, it was possible to verify in the digital simulation environment that the commands sent and the path planned were correctly executed, as the position and orientation of the drone were captured and plotted in real time.

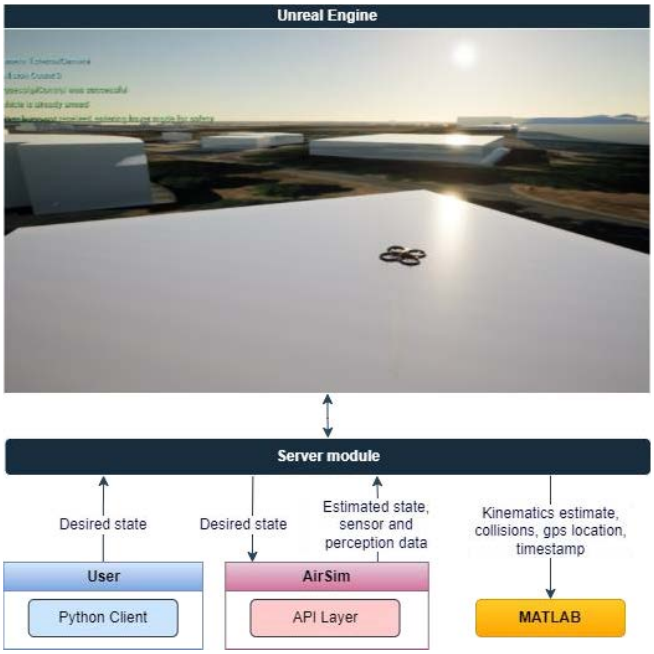


Figure 4. Co-simulation experimental setup

The same approach was then applied to the sensor modules: GPS, INS, camera and LIDAR. Similarly, real-time data acquisition and plotting were possible, as depicted in Figure 7. This approach proved particularly useful for the camera and LIDAR sensors.

As far as the LIDAR is concerned, it is implemented in AirSim, but only for the purpose of graphical visualisation. With the current capabilities, storing and using data directly in AirSim for intelligent algorithms such as collision avoidance or SLAM (Simultaneous Localisation and Mapping) is impossible. However, by co-simulation with the MATLAB/Simulink environment, storing and processing such data in the format the user chooses while the simulation is running is possible. Point Cloud, commonly used to process and display data in 3D space, is the format used in the actual configuration.

Similarly, for the camera sensor, it is possible to acquire images using Python APIs, but these are then stored in jpg format in a default directory, which does not allow the data to be used for immediate processing. On the other hand, in the MATLAB workspace, the acquisition is in RGB format, generating arrays at each time stamp that can be manipulated to reconstruct the original image and apply specific image processing techniques. The latter include segmentation, filtering or the application of intelligent algorithms for detection and tracking if further elaborations are applied after the acquisition. Thanks also to the application of another specific toolbox dedicated to image processing, all these advancements can be accomplished in the MATLAB/Simulink environment. Thus, the proposed co-simulation architecture can go beyond the basic capabilities of a single software simulation.

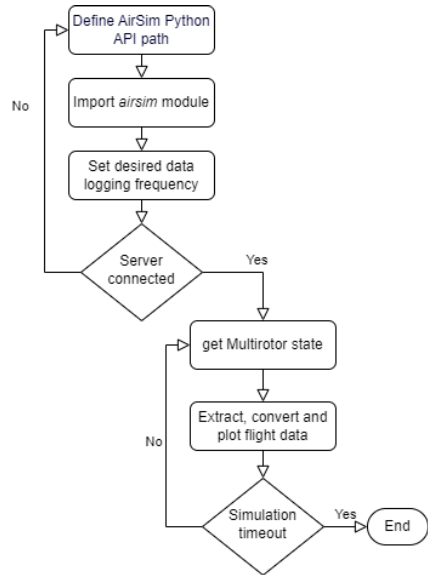


Figure 5. Flowchart for data acquisition and plotting

B. Sensor Modules Evaluation

The validation experiment reported in the previous section helped verify that the proposed methodology was correct and could be implemented, satisfying the requirements for a DT prototype. However, in the architecture proposed in Figure 4, only the reading capabilities of MATLAB/Simulink from the server module were tested. Ideally, to fully exploit the co-simulation framework and improve the functionality of the DT, the demonstrated capabilities of this software should not only be used for data acquisition and processing but should also completely replace one of the modules of the AirSim core library.

In this context, it was decided to perform another experiment concerning the sensor modules, specifically in the navigation domain, starting with the GPS sensor. The architecture used is almost the same. The only difference is that, in this case, the option of writing to the server module in MATLAB has been enabled. The code used is shown in Figure 8, highlighting the changes made with respect to the previous experiment and detailing the inputs and outputs used by the GPS sensor module, which this time was implemented in MATLAB instead of using the standard one. The digital simulated scene and trajectory remain unchanged as in the previous simulation.

MATLAB allows the customisation of the GPS sensor by tuning seven different parameters, including Sampling Rate, Horizontal Position Accuracy, Vertical Position Accuracy, Velocity Accuracy and Decay Factor. These were set to the default configuration to allow a comparison with the AirSim supplied GPS sensor, whose parameters remained

unchanged. The results obtained by performing a simulation for approximately two minutes are shown in Figure 9.

When the sensor implemented in MATLAB is used as the primary sensor, the vehicle pose is modified accordingly, as shown in the middle plot of Figure 9. In both graphs, the outputs of the two sensors are shown for comparison purposes. However, to make the deviation between them easier to spot, the error of the MATLAB sensor compared to the reference AirSim one has been determined and plotted. The trend of this error is reported and the horizontal blue line highlights its mean value.

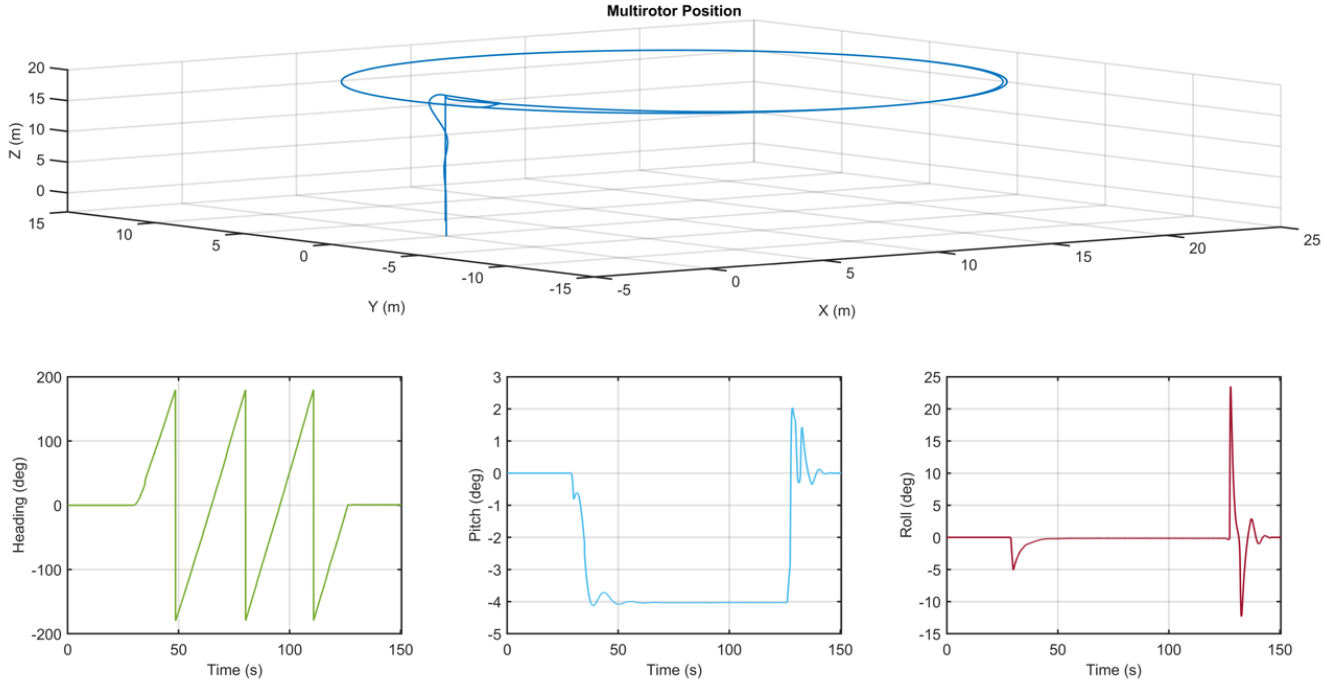


Figure 6. MATLAB kinematics position and orientation output

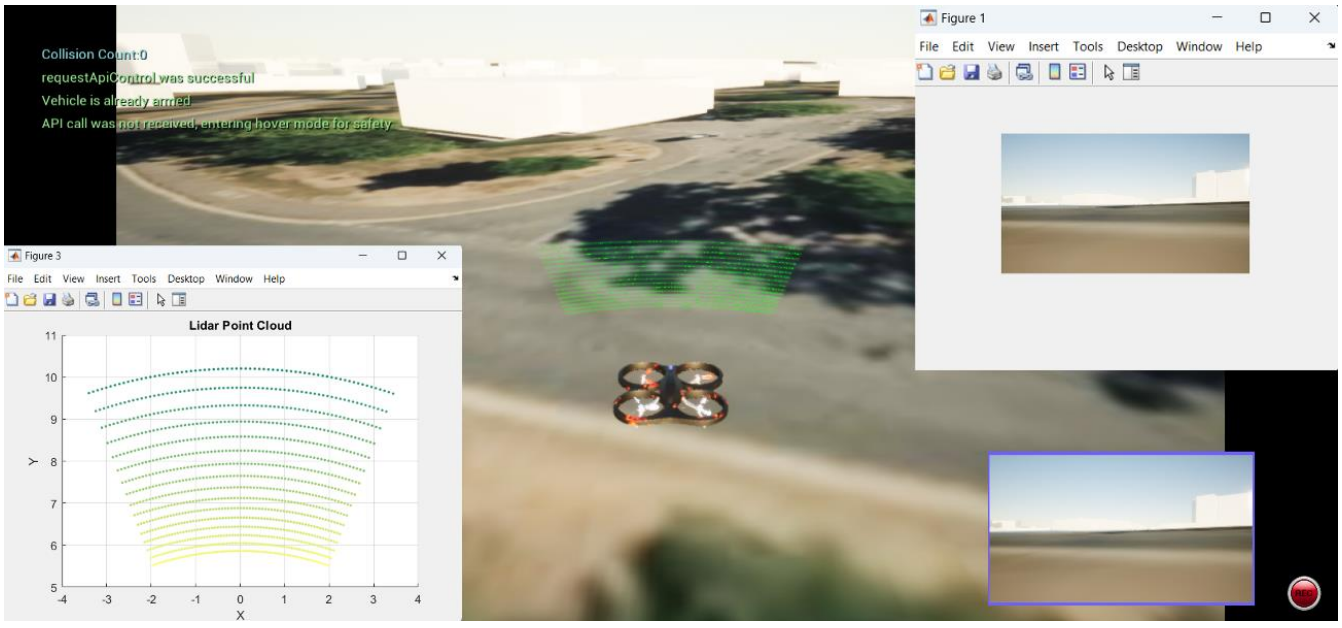


Figure 7. Sensor modules co-simulation sample

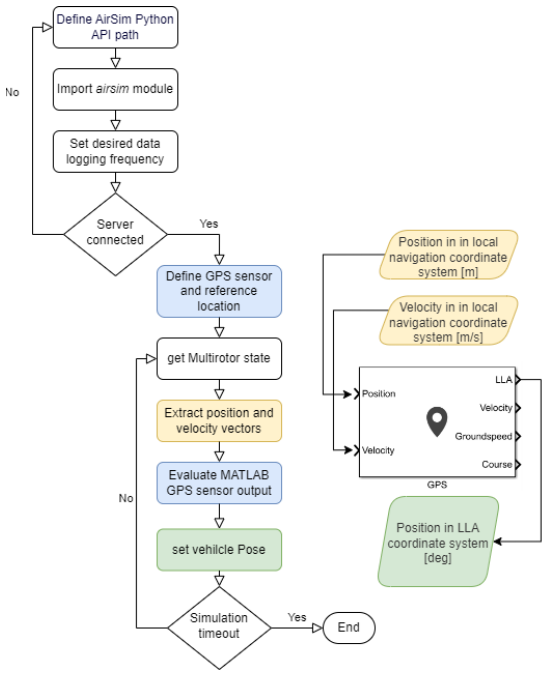


Figure 8. Flowchart for GPS sensor co-simulation

Since the accuracy of a commercial GPS sensor is between three and five metres under open sky conditions, it can be stated that the average deviation of about one metre for all directions of the determined reference system is negligible. It would be necessary to replicate the same experimental setup in the physical world, record the data from the physical GPS sensor and use the latter as a reference in order to determine which of the two sensors best emulates the behaviour of a real sensor and is, therefore, able to generate more accurate synthetic data. However, thanks to the flexibility and customisation features of the sensor module in MATLAB, it is expected to achieve better results.

Should this prove insufficient for the accuracy requirements of the DT prototype, a further step could be taken by fusing the GPS data with the inertial sensor measurements and implementing a Kalman filter. Simple solutions in this direction have also been under investigation, and the results are promising. This illustrates an additional aspect of how the co-simulation of different tools can enrich the existing functionality and models.

C. Vehicle Dynamic Model Comparison

The experiments presented in the following aims to address the identified limitations of the vehicle dynamics model. Modifying this model requires understanding how it is designed, how it interacts with other models in the simulation architecture, and which data it processes and shares. A simplified architecture showing the data flow from sending a command to the dynamic response of the vehicle is shown in Figure 10. The same figure then shows two variants in which certain blocks have been redesigned so that they can be realised by employing MATLAB/Simulink. In the first variant, a MATLAB/Simulink 6DOF model is proposed to replace the equations of motion solver implemented in AirSim with the aim of obtaining more accurate kinematic estimates. The second model, however, proposes an even more integrated architecture that takes full advantage of the modelling capabilities of MATLAB/Simulink from all perspectives. Indeed, in such a configuration, not only are the equations of motion modified, but the entire dynamic model of the airborne platform is replaced by a vehicle dynamic system built in Simulink that can be easily integrated with guidance and control systems. This was then translated into a Simulink architecture, shown in Figure 11.

The last configuration depicted is chosen as the best solution because it allows the modelling and simulation of different types of UAVs. This solves the limitations associated with the rotorcraft proposed in AirSim. Furthermore, designing the behaviour of the vehicle from scratch in MATLAB allows a better analysis of the vehicle plant so that the flight control system can be designed and tuned in the best possible way. This opens up unlimited possibilities from a vehicle design point of view; from fixed wing to eVTOL models, all conceptual AAM design models can be implemented using the combined capabilities of these simulation tools.

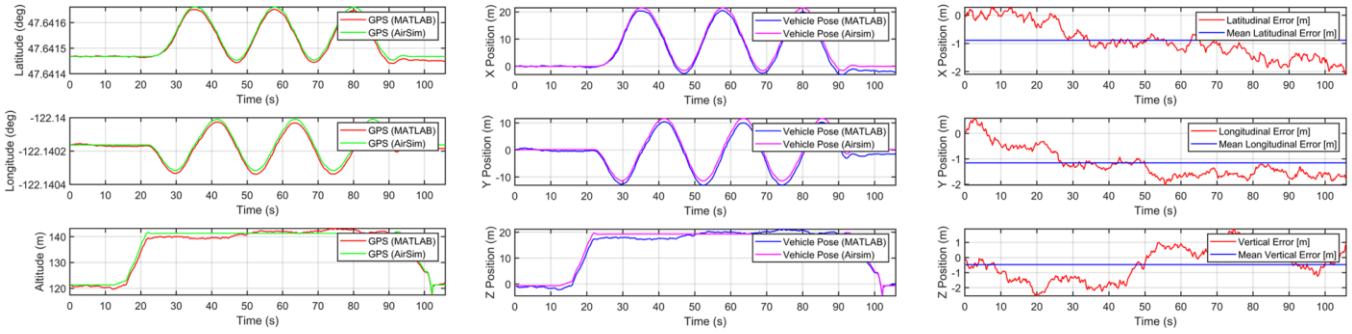


Figure 9. GPS sensor module comparison (left), Vehicle Pose comparison with and without co-simulation (middle), GPS sensor error propagation and mean value (right)

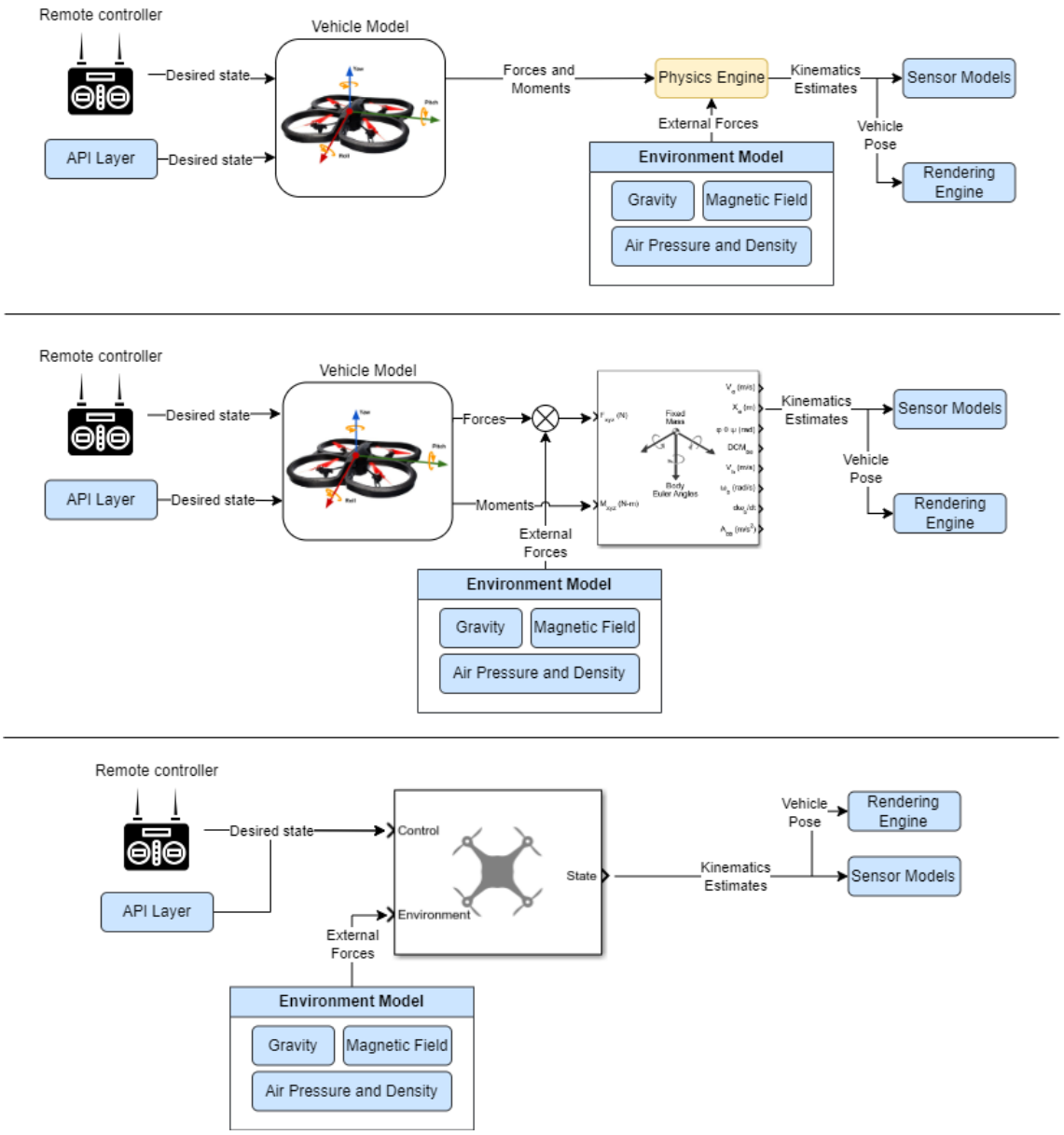


Figure 10. Simulator architecture configurations combining two co-simulation tools: AirSim and MATLAB/Simulink

Furthermore, this methodology enables users to ensure that the vehicle plant model synchronises with the latest technologies. Adopting an incremental approach, the model's fidelity can be progressively increased as the design progresses from the conceptual to the detailed phase. At the same time, the development of guidance and control algorithms can be started with a low-fidelity model, and the plant model can be adapted as additional model elements become available. This again emphasises an incremental strategy that systematically integrates control and dynamics layers into the simulation process.

Two experiments were performed using vehicle models pre-built in MATLAB to demonstrate how the proposed integration was performed. This choice was dictated by the development timeframe of this project. The development of an aerodynamic model from scratch would have been prohibitive in terms of time and resources, but this leaves potential for future development and improvement. The first of these was used to validate the architecture concept presented in Figure 10, by completing a mission based on the standard drone operational concept. This proved useful in understanding the behaviour of the MATLAB vehicle model, particularly in terms of the dynamic stability response of the drone receiving commands from an external platform as the AirSim API layer.

A) Infrastructure Inspection Mission Using Co-Simulated Waypoint Follower Algorithm

MATLAB/Simulink includes two predefined models, a quadrotor and a fixed-wing unmanned aircraft. Without

developing additional code or components, all the functionalities, translated into blocks in the Simulink environment, can be modified according to the chosen vehicle type. In the diagram shown in Figure 11, it is possible to identify three domains represented by distinct colours. The first domain, in green, enables co-simulation between MATLAB and Simulink. A low-fidelity scenario is defined in MATLAB by specifying the number of vehicles, their initial position and orientation, and the coordinate system used. The blue area represents the guidance and control systems of the UAV. Specifically, the waypoint follower block calculates the trajectory the drone must follow to satisfy the input parameters, i.e., defined waypoints and lookahead distance. The same block then provides the controller with the following desired state and instructions on how to reach it by indicating the heading variation. The controller used, and some data conversion blocks are provided by a MathWorks library [49]. The control system's output, together with the variations due to external factors (gravity, magnetic field, pressure), becomes the input to the dynamic model that ultimately determines the next state of the drone. As can be seen, the plant and control system are arranged in a classical closed-loop configuration. Then, in the area shown in red, the necessary conversions are made to the vehicle state to make it compatible with the notation used in AirSim, after which the kinematic estimates are forwarded via the communication interface already described to be processed and used by all the other modules that make up the DT.

It should be noted that changes could also be made to the environment model. As shown in Figure 10, the AirSim environment model considers only three factors: gravity, pressure and magnetic field, whereas, in Simulink, it is also possible to model: atmosphere, celestial phenomena and wind. Standard atmospheric profiles such as ISA and COESA are given, while non-standard daily simulations and lapse rate atmospheres can also be modelled if required. Instead, the wind section allows the implementation of wind-related simulations like turbulence, gust, shear, and horizontal wind.

It was decided to simulate an infrastructure inspection mission to verify the validity of the proposed architecture and methodology. Specifically, four waypoints were defined around the Kings Norton Library building in the digital environment developed at Cranfield University, employing Cesium as a plugin for Unreal Engine. This made it easy to plan a comprehensive circuit of the building at a constant altitude. Figure 12 shows a scenario rendering where the defined waypoints are marked in green. The time taken to complete the defined path was 62.5 s.

Figure 14 shows a sample of the ongoing simulation, with Simulink vehicle position state generation at the top and AirSim and Unreal 3D rendering at the bottom, highlighting the DT's high-fidelity simulation capabilities. As has already been shown for the sensor suite, it has also been possible to collect flight data via the communication interface that has been designed. The trajectories of four different simulations are shown in Figure 13. The varied parameter is the Lookahead Distance, considered one of the controller's most critical tuning properties. It defines how far along the trajectory the UAV should look from its current position in order to calculate the angular velocity commands. The effect of changing this parameter is to alter the UAV's path tracking capabilities, and two main objectives can be addressed: regaining the path and maintaining the path. To quickly retrieve the path between waypoints, a small Lookahead Distance will cause the drone to move quickly towards the path. However, as seen in the orange, yellow and purple plots, the drone overshoots the path and oscillates along the desired one. A larger Lookahead Distance can be chosen to reduce the oscillations, which may result in larger curvatures near the corners, as shown by the green plot.

In brief, the conducted infrastructure inspection mission underlines the practical applicability and effectiveness of the proposed architecture. Through the integration of MATLAB/Simulink and AirSim, the successful execution of the mission validates the high-fidelity simulation capabilities of the developed DT framework. This experiment serves as a critical step in confirming the potential of co-simulation to enable autonomous aircraft systems to navigate complex scenarios with unprecedented accuracy, setting the stage for further advances in unmanned aircraft operations.

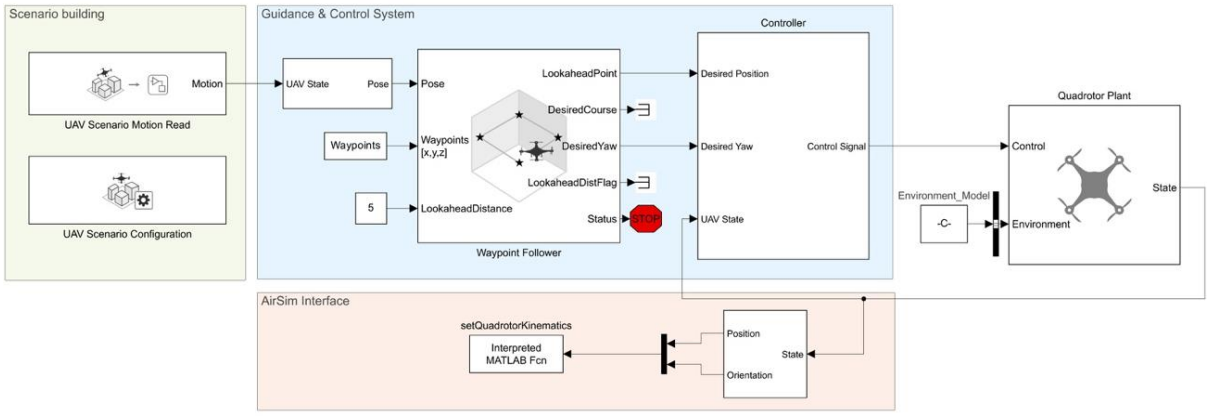


Figure 11. Simulink Dynamic Model, Guidance and Control systems block diagram (inspired by [49])

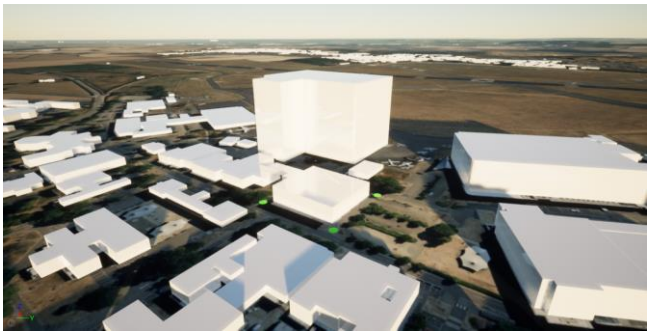


Figure 12. Cranfield waypoint definition scenario rendered in Unreal Engine

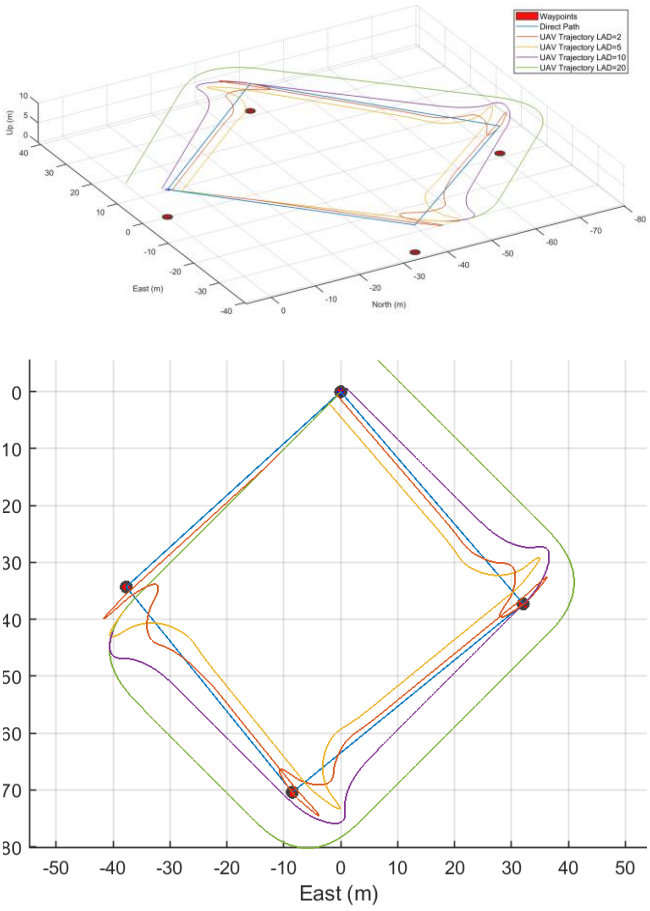


Figure 13. Co-simulated UAV trajectory varying Lookahead Distance parameter

b) Obstacle Avoidance Co-Simulated Solution

The final proposed experiment aims to go one step further than the current demonstrated capabilities, by simultaneously using the co-simulation capabilities developed for both the sensor module and the vehicle model. This was possible thanks to the validation data obtained while performing the previous test case. It was crucial in synchronising the data handover behaviour between the two simulation software, allowing the co-simulation of a more complex scenario. Specifically, two waypoints were defined on opposite sides of Cranfield University's Building 88. This building is flanked by Building 62, creating a corridor approximately two metres wide.

The objective of the experiment was to acquire data from the LIDAR sensor generated by AirSim, convert it to point cloud format as previously demonstrated and use them as input to the obstacle avoidance block presented in Table 3 1. This block allows the vehicle's guidance system to calculate an obstacle-free trajectory based on the distance measured by the LIDAR sensor. In order to avoid collision with other elements in the scenario, it transmits the desired heading angle change to the control system.

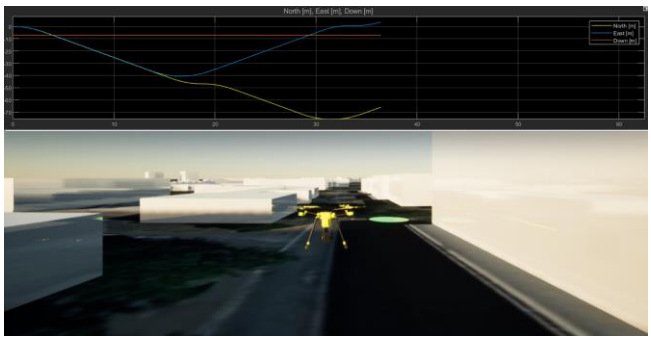


Figure 14. Run co-simulation with MATLAB, Simulink, AirSim and Unreal Engine

The simulation time for the aircraft to reach the second defined waypoint was 30 s. The trajectory travelled compared to the straight line connecting the two points is shown in Figure 15. As can be easily seen, the path followed by the UAV was modified according to the scenario to avoid collisions with buildings. The only input, as in the previous experiment, was to determine the correct tuning parameters of the control system; no other commands were supplied during the execution of the simulation. The achieved performance is the result of the proposed co-simulation framework, where the guidance system implemented in MATLAB, together with the control system and the vehicle plant, modified the kinematic state of the UAV, calculating the obstacle-free direction thanks to the data from the AirSim LIDAR sensor, which in parallel acquired the structural characteristics of the scenario constructed in Unreal.

In conclusion, the extended experiment performed represents a significant step beyond the capabilities initially demonstrated. By coordinating the co-simulation functionalities of both the sensor and vehicle model, the experiment achieved an impressive level of integration. A challenging corridor scenario was designed using strategically placed waypoints, demonstrating the system's adaptability to complex environments. Central to the experiment was the use of LIDAR sensor data captured by the AirSim simulation and its effective use in obstacle avoidance tasks. Sharp interaction between the Simulink model and the AirSim LIDAR sensor output enabled the autonomous navigation of the platform through the corridor while avoiding buildings.

This achievement again highlights the co-simulation framework's potential and emphasises its role in enabling sophisticated autonomous manoeuvring, further consolidating its applicability in real-world scenarios.

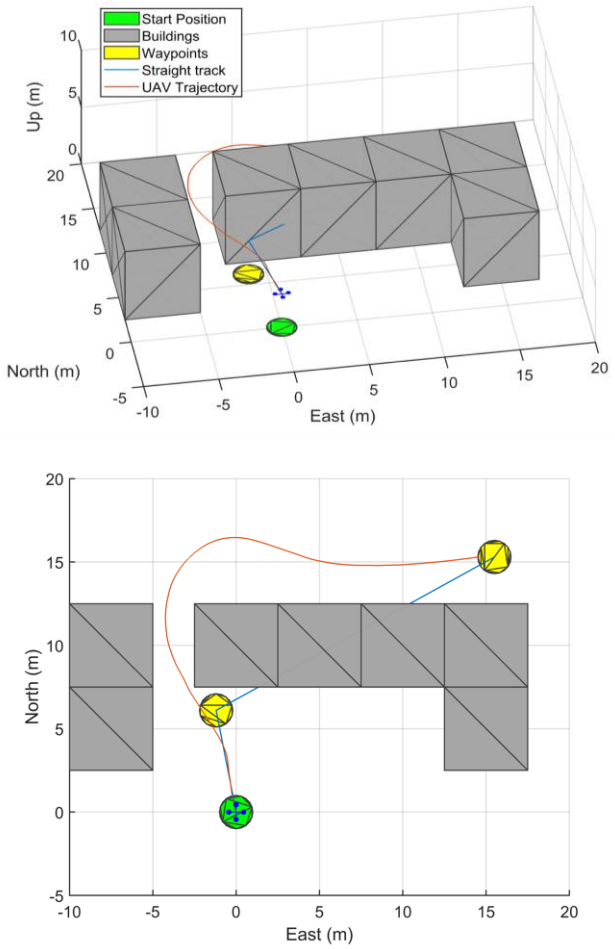


Figure 15. Co-simulated UAV trajectory varying heading according to LIDAR measurements

5. CONCLUSIONS

Overall, this paper demonstrates a step towards DT technology for AAM systems through an innovative co-simulation approach. Theoretical concepts are translated into practical implementation by effectively combining multiple simulation tools, paving the way for the advancement of autonomous aircraft systems in line with the AAM paradigm. The main results and contributions can be summarised as follows:

A novel co-simulation architecture is presented, effectively fusing the functionality of MATLAB/Simulink and AirSim, thereby enhancing DT's simulation capabilities for AAM concepts. It also incorporates cutting-edge sensor technologies such as LIDAR, opening the door to high-precision mapping and remote sensing applications.

- Emerging trends have been addressed, including autonomous flight operations and advanced data analytics.
- A high-fidelity simulation arena has been tested through an incremental approach. The experiments performed started with a basic implementation to validate the proposed architecture and then progressed to more ambitious objectives. This was critical for thoroughly testing, refining, and validating DT capabilities across multiple scenarios.
- The development of autonomous systems has been enabled, providing a robust foundation for vehicle modelling, guidance algorithm optimisation, and comprehensive flight data evaluation.

The co-simulation methodology explored in this paper offers a solid foundation for future research and development. Several directions for future exploration include:

- Enhancing existing Sensors' Models and adding new ones: further refining the integration of sensor models within the co-simulation framework, such as improving accuracy, fidelity and exploring sensor fusion techniques. Model cutting-edge sensors as multispectral imaging systems.
- Advanced Control Algorithms: Investigating advanced control algorithms to enhance autonomous flight operations, obstacle avoidance, and path planning. Use Machine Learning techniques to incorporate intelligent solutions such as Reinforcement Learning Control Laws.
- Real-world Validation: extending the validation process to real-world scenarios by comparing the co-simulation results with physical experimentation to ensure the framework's reliability and accuracy.
- Evolving Vehicle Dynamics Models: continuously evolving the vehicle dynamics models to encompass a broader range of vehicle types, enabling the simulation of various advanced air mobility concepts.
- Enable large-scale co-simulation activities: addressing software latency and synchronisation issues.

In conclusion, this paper's contributions are poised to shape the landscape of DT development for future AAM systems, establishing a versatile and effective co-simulation framework that opens new horizons for research, innovation, and practical implementation in the aviation industry.

ACKNOWLEDGEMENT

This research is funded by the UKRI Future Flight Challenge Phase 3 project HADO (High-intensity Autonomous Drone Operations), grant number 10024815. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising.

REFERENCES

[1] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, 'Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends', Intelligent Service Robotics, vol. 16, no. 1. Springer Science and Business Media Deutschland GmbH, pp. 109–137, Mar. 01, 2023. doi: 10.1007/s11370-022-00452-4.

[2] M. Grote et al., 'Sharing airspace with Uncrewed Aerial Vehicles (UAVs): Views of the General Aviation (GA) community', J Air Transp Manag, vol. 102, Jul. 2022, doi: 10.1016/j.jairtraman.2022.102218.

[3] J. Coykendall, M. Metcalfe, A. Hussain, and T. Dronamraju, 'Advanced air mobility Disrupting the future of mobility', 2021. Accessed: Aug. 16, 2023. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/energy-resources/eri-advanced-air-mobility.pdf

[4] R. Horizons Council, 'The Regulation of Drones An exploratory study The Regulatory Horizons Council', 2021.

[5] ICAO, 'Development of UAS Regulations', UAS Toolkit, Chicago Convention Doc 7300. https://www.icao.int/safety/UA/UASToolkit/Pages/Narrative-Regulation.aspx (accessed Jun. 13, 2023).

[6] Civil Aviation Authority UK, 'Safety and Airspace Regulation Group Unmanned Aircraft System Operations in UK Airspace-Policy and Guidance CAP 722 | Ninth Edition Amendment 1'. [Online]. Available: www.caa.co.uk/cap722

[7] D. M. Botín-Sanabria, S. Mihaita, R. E. Peimbert-García, M. A. Ramírez-Moreno, R. A. Ramírez-Mendoza, and J. de J. Lozoya-Santos, 'Digital Twin Technology Challenges and Applications: A Comprehensive Review', Remote Sensing, vol. 14, no. 6. MDPI, Mar. 01, 2022. doi: 10.3390/rs14061335.

[8] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, and A. Calinescu, 'Digital Twins: State of the art theory and practice, challenges, and open research questions', J Ind Inf Integr, vol. 30, p. 100383, Nov. 2022, doi: 10.1016/J.JII.2022.100383.

[9] D. Lee et al., 'Digital Twin-Based Analysis and Optimization for Design and Planning of Production Lines', Machines, vol. 10, no. 12, Dec. 2022, doi: 10.3390/machines10121147.

[10] M. Attaran and B. G. Celik, 'Digital Twin: Benefits, use cases, challenges, and opportunities', Decision Analytics Journal, vol. 6, p. 100165, Mar. 2023, doi: 10.1016/J.DAJOUR.2023.100165.

[11] F. Fabra, W. Zamora, J. Sangüesa, C. T. Calafate, J. C. Cano, and P. Manzoni, 'A distributed approach for collision avoidance between multirotor UAVs following planned missions', Sensors (Switzerland), vol. 19, no. 10, May 2019, doi: 10.3390/s19102404.

[12] S. Goel, M. Kumar Sharma, and D. Garg, 'Analysis of Reinforcement Learning Application in Autonomous Drone Control: A Survey'. [Online]. Available: http://www.proteusresearch.org/

[13] M. Ramezani, H. Habibi, H. Voos, and J. Luis Sanchez-Lopez, 'UAV Path Planning Employing MPC-Reinforcement Learning Method Considering Collision Avoidance', Feb. 2023, doi: 10.48550.

[14] M. Masson, 'Use and Benefits of Simulators', EASA Community Network, Mar. 2021. https://www.easa.europa.eu/community/topics/use-and-benefits-simulators (accessed Jun. 13, 2023).

[15] C. Conrad, Q. Delezenne, A. Mukherjee, A. A. Mhowwala, and M. Ahmed, 'Developing a Digital Twin System to Test Intelligent Solutions for Advanced Air Mobility Operations', Group Design Project, Advanced Air Mobility Systems MSc, Cranfield University, Cranfield, 2023.

[16] C. Conrad et al., 'Developing a Digital Twin for Testing Multi-Agent Systems in Advanced Air Mobility: A Case Study of Cranfield University and Airport', unpublished, Cranfield University, 2023.

[17] X. L. Wei, X. L. Huang, T. Lu, and G. G. Song, 'An Improved Method Based on Deep Reinforcement Learning for Target Searching', in 2019 4th International Conference on Robotics and Automation Engineering, ICRAE 2019, Institute of Electrical and Electronics Engineers Inc., Nov. 2019, pp. 130–134. doi: 10.1109/ICRAE48301.2019.9043821.

[18] V. Mnih et al., 'Human-level control through deep reinforcement learning', Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[19] R. Polvara et al., 'Autonomous Quadrotor Landing using Deep Reinforcement Learning', Sep. 2017, [Online]. Available: http://arxiv.org/abs/1709.03339

[20] L. He, N. Aouf, J. F. Whidborne, and B. Song, 'Deep Reinforcement Learning based Local Planner for UAV Obstacle Avoidance using Demonstration Data', Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.02521

[21] T. C. Wu, S. Y. Tseng, C. F. Lai, C. Y. Ho, and Y. H. Lai, 'Navigating assistance system for quadcopter with deep reinforcement learning', in Proceedings - 2018 1st International Cognitive Cities Conference, IC3 2018, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 16–19. doi: 10.1109/IC3.2018.00013.

[22] H. Bou-Ammar, H. Voos, and W. Ertel, 'Controller design for quadrotor UAVs using reinforcement learning', in 2010 IEEE International Conference on Control Applications, IEEE, Sep. 2010, pp. 2130–2135. doi: 10.1109/CCA.2010.5611206.

[23] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, 'Co-simulation: State of the art', Feb. 2017, [Online]. Available: http://arxiv.org/abs/1702.00686

[24] G. Schweiger et al., 'An empirical survey on co-simulation: Promising standards, challenges and research needs', Simul Model Pract Theory, vol. 95, pp. 148–163, Sep. 2019, doi: 10.1016/J.SIMPAT.2019.05.001.

[25] A. I. Hentati, L. C. Fourati, E. Elgharbi, and S. Tayeb, 'Simulation tools, environments and frameworks for UAVs and multi-UAV-based systems performance analysis (version 2.0)', International Journal of Modelling and Simulation, 2022, doi: 10.1080/02286203.2022.2092257.

[26] Institute of Electrical and Electronics Engineers, IWCMC 2018 : the 14th International Wireless Communications & Mobile Computing Conference : June 25-29, 2018, Limassol, Cyprus.

[27] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, 'A survey of Open-Source UAV flight controllers and flight simulators', Microprocess Microsyst, vol. 61, pp. 11–20, Sep. 2018, doi: 10.1016/j.micpro.2018.05.002.

[28] The MathWorks Inc, 'MATLAB version 9.14.0 (R2023a)', mathworks.com, 2023. https://uk.mathworks.com/products/matlab.html (accessed Aug. 03, 2023).

[29] Open Source Robotics Foundation Inc., 'Gazebo', gazebosim.org. http://gazebosim.org. (accessed Aug. 03, 2023).

[30] A. Meyer, 'X-Plane 12'. http://www.x-plane.com. (accessed Aug. 03, 2023).

[31] J. S. Berndt, T. Peden, and D. Megginson, 'JSBSim', GitHub, 2011. https://github.com/JSBSim-Team/jsbsim (accessed Aug. 03, 2023).

[32] A. Babushkin, 'jMAVSim', GitHub, 2013. https://github.com/PX4/jMAVSim#readme (accessed Aug. 03, 2023).

[33] J. Zhao et al., 'Co-simulation Digital Twin Framework for Testing Future Advanced Air Mobility Concepts: A Study with BlueSky and AirSim', unpublished, Cranfield University, 2023.

[34] A. Alharbi, I. Petrunin, and D. Panagiotakopoulos, 'Assuring Safe and Efficient Operation of UAV Using Explainable Machine Learning', Drones, vol. 7, no. 5, May 2023, doi: 10.3390/drones7050327.

[35] A. Arfaoui, 'Unmanned Aerial Vehicle: Review of Onboard Sensors, Application Fields, Open Problems and Research Issues', 2017. [Online]. Available: https://www.researchgate.net/publication/315076314

[36] F. D'Urso, C. Santoro, and F. F. Santoro, 'An integrated framework for the realistic simulation of multi-UAV applications', Computers & Electrical Engineering, vol. 74, pp. 196–209, Mar. 2019, doi: 10.1016/j.compeleceng.2019.01.016.

[37] J. Zhao, Y. Xu, C. Conrad, A. Tsourdos, and Q. Delezenne, 'A Digital Twin Mixed-reality System for Testing Future Advanced Air Mobility Concepts: A Prototype', unpublished, vol. Cranfield University, 2023.

[38] S. Shah, D. Dey, C. Lovett, and A. Kapoor, 'AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles', in Springer Proceedings in Advanced Robotics, Springer Science and Business Media B.V., 2018, pp. 621–635. doi: 10.1007/978-3-319-67361-5_40.

[39] The MathWorks Inc., 'UAV Toolbox User's Guide R2023a', Natick, Massachusetts, 2020. [Online]. Available: www.mathworks.com

[40] The MathWorks Inc., '6DOF (Euler Angles)', mathworks.com, 2023. https://it.mathworks.com/help/aeroblks/6dofeulerangles.html (accessed Aug. 04, 2023).

[41] The MathWorks Inc., 'Guidance Model', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/guidancemodel.html (accessed Aug. 04, 2023).

[42] The MathWorks Inc., 'Path Manager', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/pathmanager.html (accessed Aug. 04, 2023).

[43] The MathWorks Inc., 'Obstacle Avoidance', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/obstacleavoidance.html (accessed Aug. 04, 2023).

[44] The MathWorks Inc., 'Orbit Follower', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/orbitfollower.html (accessed Aug. 04, 2023).

[45] The MathWorks Inc., 'INS', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/ins.html (accessed Aug. 04, 2023).

[46] The MathWorks Inc., 'GPS', mathworks.com, 2023. https://it.mathworks.com/help/uav/ref/gps.html (accessed Aug. 04, 2023).

[47] E. Dufford, M. B. Malhi, and F. Noto, 'Accelerate Aerial Autonomy with Simulink and Microsoft Project AirSim'. Accessed: Aug. 04, 2023. [Online]. Available: https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/company/events/conferences/matlab-expo-2023/ww-2023-expo-accelerate-aerial-autonomy-with-simulink-and-microsoft-project-airsim.pdf

[48] Microsoft Research, 'Orbit Trajectory'. https://microsoft.github.io/AirSim/orbit/ (accessed Aug. 05, 2023).

[49] The MathWorks Inc., 'UAV Obstacle Avoidance in Simulink', mathworks.com, 2023. https://it.mathworks.com/help/uav/ug/uav-obstacle-avoidance-in-simulink.html (accessed Aug. 09, 2023).

BIOGRAPHY



Lorenzo Turco *received a BSc. in Electronic Engineering from University "Federico II", Naples in 2019 and an MSc in Electronic Engineering with a focus on Digital Electronics from the same University in 2021. He joined the Italian Air Force Academy in 2016 and is actually employed at the Italian Air Force Flight Test Centre in Pratica di Mare as Flight Test Engineer with the rank of Lieutenant. He has been studying Avionic Systems Design at Cranfield University in 2023 obtaining an additional MSc.*



Junjie Zhao *received an MSc and his PhD degrees in Aerospace from Cranfield University. He is currently a Research Fellow in Intelligent Systems Engineering for Digital Aviation with the Centre for Autonomous and Cyber-Physical Systems at Cranfield University. He conducts research in the fields of Air Traffic Management (ATM) / Unmanned Traffic Management (UTM) Integration, Advanced Air Mobility, and Digital Twin.*



Dr. Yan Xu *is a Senior Lecturer in ATM/CNS, with the School of Aerospace, Transport and Manufacturing at Cranfield University. He is the Course Director of newly-established MSc in Advanced Air Mobility Systems (AAMS). He is also leading the Air Traffic Management Laboratory based in the Digital Aviation Research and Technology Centre (DARTeC). Before taking the academic role, he conducted his postdoc research with the same school as a Research Fellow in UAS Traffic Management. Prior to joining Cranfield University, he received his Ph.D. in Aerospace Science and Technology from the Technical University of Catalonia, and received his M.Sc. and B.Eng. in Traffic Engineering from Nanjing University of Aeronautics and Astronautics. He performed a visiting study in early 2017 at the Ecole Nationale de l'Aviation Civile. His main research interests include air traffic management, UAS traffic management and Urban Air Mobility. He has been involved in various research projects in these areas funded by EU SESAR and UK Future Flight Programmes.*



https://dspace.org/

A study on co-simulation digital twin with MATLAB and AirSim for future advanced air mobility

Turco, Lorenzo; Zhao, Junjie; Xu, Yan; Tsourdos, Antonios

2024-03-02
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/AERO58975.2024.10521333
Turco L, Zhao J, Xu Y, Tsourdos A. (2024) A study on co-simulation digital twin with MATLAB and AirSim for future advanced air mobility. In: 2024 IEEE Aerospace Conference, 2-9 March 2024, Big Sky, MT, USA

Downloaded from a DSpace repository. https://dspace.org/