In: IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC) 2023, 1-5 October 2023, Barcelona, Spain
DOI: 10.1109/DASC58513.2023.10311333

Developing a Digital Twin for Testing Multi-Agent Systems in Advanced Air Mobility: A Case Study of Cranfield University and Airport

Christopher Conrad, Quentin Delezenne, Anurag Mukherjee, Ali Asgher Mhowwala,
Mohammad Ahmed, Junjie Zhao, Yan Xu, Antonios Tsourdos
Centre for Autonomous and Cyberphysical Systems
School of Aerospace, Transport and Manufacturing (SATM), Cranfield University
Cranfield, UK
Email: {christopher.conrad, quentin.delezenne.464, anurag.mukherjee.436, aliasgher.mhowwala.253,
mohammad.ahmed.954, junjie.zhao, yanxu, a.tsourdos}@cranfield.ac.uk

***Abstract*—Emerging unmanned aircraft system (UAS) and advanced air mobility (AAM) ecosystems rely on the development, certification and deployment of new and potentially intelligent technologies and algorithms. To promote a more efficient development life cycle, this work presents a digital twin architecture and environment to support the rapid prototyping and testing of multi-agent solutions for UAS and AAM applications. It leverages the capabilities of Microsoft AirSim and Cesium as plugins within the Unreal Engine 3D visualisation tool, and consolidates the digital environment with a flexible and scalable Python-based architecture. Moreover, the architecture supports hardware-in-the-loop (HIL) and mixed-reality features for enhanced testing capabilities. The system is comprehensively documented and demonstrated through a series of use cases, deployed within a custom digital environment, comprising both indoor and outdoor areas at Cranfield University and Airport. These include collaborative surveillance, UTM flight authorisation and UTM conformance monitoring experiments, that showcase the modularity, scalability and functionality of the proposed architecture. All 3D models and experimental observations are critically evaluated and shown to exhibit promising results. This thereby represents a critical step forward in the development of a robust digital twin for UAS and AAM applications.**

***Index Terms*—Advanced air mobility, AirSim, digital twin, mixed-reality, multi-agent, UAS**

I. Introduction

A. Context

Global demographic and economic growth is expected to increase conventional air traffic by an average of 6% per year, with the International Civil Aviation Organisation (ICAO) anticipating passenger traffic and freight volume to double by 2035 [1]. Additionally, the advanced air mobility (AAM) and unmanned aircraft systems (UAS) markets are forecast to experience large compound annual growth rates (CAGRs) over the next decade [2]. The introduction of a large quantity of new heterogeneous vehicles in an already saturated airspace, however, cannot be reliably handled by the existing air traffic management (ATM) infrastructure. Consequently, considerable research is underway to develop new and innovative solutions for UAS traffic management (UTM).

Testing and certifying new technologies for a conservative and safety-critical aviation industry remains a time-consuming and costly task. Various procedures have been proposed to facilitate this development process, including the specific operations risk assessment (SORA) methodology [3]. This emphasizes the importance of collecting evidence to support safety claims within a risk assessment report. Nonetheless, securing the permissions needed for in-field tests of new technologies and algorithms remains a daunting task. Moreover, physical tests are often restricted to strictly controlled airspace volumes that are not conducive to the complex environments in which the systems will be deployed, limiting the accuracy and reliability of the associated results.

Digital twin (DT) environments offer a unique solution to expedite product development, by supporting preliminary system evaluation in virtual or semi-virtual test environments. These have been widely employed in several industries [4], but have yet to be adopted for UAS and AAM applications. A market gap thereby exists for a DT that facilitates the development and testing of multi-agent frameworks and algorithms for UAS and AAM operations. Cranfield University aims to address this gap through the development and deployment of a robust DT platform for UAS and AAM applications with enhanced mixed-reality capabilities. This represents the main deliverable of the synthetic test environment work package within project HADO (High-intensity Autonomous Drone Operations), which aims to develop, evaluate, standardise, and operationally deploy fully automated UASs at London Heathrow Airport.

This work further develops the framework and methodologies proposed in [5] into a functional DT prototype, validated through several use cases within indoor and outdoor environments at Cranfield University and Airport. It leverages the capabilities of Microsoft AirSim as a plugin within the Unreal Engine 3D visualisation tool, and consolidates a digital environment with a custom Python-based architecture that facilitates the deployment and testing of both procedural and intelligent multi-agent algorithms.

© 2023 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/DASC58513.2023.10311333.
Please refer to any applicable publisher terms of use.

B. Simulation Platforms and Digital Twins

Several simulation platforms have been developed for UAS operations. Notably, Microsoft released Project AirSim in 2023 for high-fidelity large-scale testing of autonomous solutions [6]. This platform, however, is decoupled from the software needed to create custom digital worlds, integrate custom vehicle models and deploy or test custom algorithms. Additionally, it does not readily support mixed-reality capabilities. Moreover, Quantum 3D [7] and MIT [8] have developed UAS pilot simulators to train UAS stakeholders across a range of mission profiles. Nonetheless, these software packages are not suitable for deploying and testing large-scale multi-agent simulations. This leaves a gap for an integrated, well-documented and easy-to-use platform that supports the integration of custom worlds, vehicle models and algorithms with a high degree of flexibility and scalability.

DTs have been employed to better reflect physical entities and environments. Notably, a DT introduces automatic bidirectional data flow between the physical world and its virtual representation, as illustrated in Fig. 1 [9]. Such platforms have been used in numerous industries with varying levels of maturity, as discussed in [5]. No comprehensive DT architecture, however, has yet been developed to test multi-agent UAS and AAM operations in a high-fidelity digital world, with automatic bidirectional data flow to its physical counterpart.

C. System Requirements

A robust DT platform benefits several stakeholders within the AAM and UAS industries. To satisfy the needs of these stakeholders, a comprehensive list of system requirements was derived for a DT prototype, summarised in Fig. 2 and listed in Table I. These were initially validated by a consortium of industrial and academic partners [10].

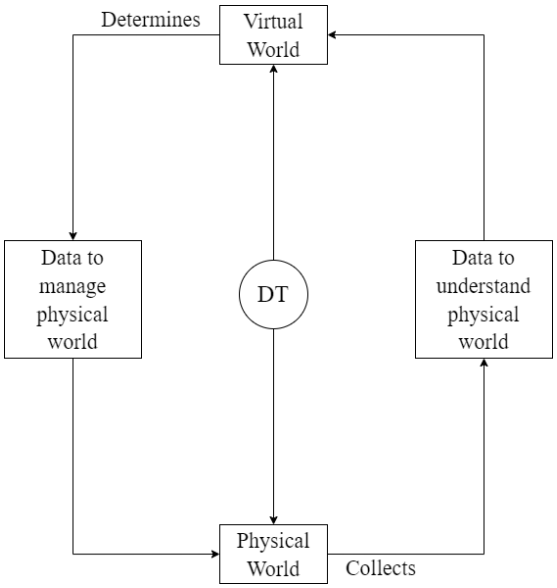


Fig. 1. Generic DT architecture.

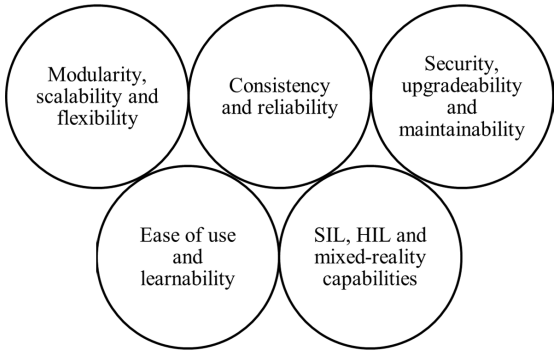


Fig. 2. Summary of key stakeholder requirements.

D. Contributions

This work designs, implements, tests and evaluates a DT architecture for testing procedural and intelligent multi-agent UAS and AAM operations. It delivers a DT prototype, built on Unreal/AirSim and adapted for UAS/AAM applications, that supports independent multi-agent simulations in a flexible, customisable, scalable and reliable environment with static and dynamic entities and enhanced hardware-in-the-loop (HIL) and mixed-reality features.

The main contributions of this work are as follows:

- a unified, modular and scalable DT architecture for UAS and AAM operations is proposed, prototyped and evaluated, with enhanced HIL and mixed-reality capabilities;
- a custom Python-based code architecture is developed to complement the DT, supporting a secure object-oriented framework and the inclusion of user-customisable scripts in a highly modular plug-and-play fashion;
- detailed methodologies for 3D indoor and outdoor digital world creation and custom algorithm deployment within the DT are comprehensively detailed and demonstrated, in line with the foundational work presented in [5]; and
- several demonstrative use cases are deployed to highlight the ability of the DT to support the development and testing of multi-agent systems and UTM ecosystems.

Notably, this paper builds on the work carried out in [5] and paves the way for the demonstrations and simulations conducted in [11].

E. Paper Structure

The remaining part of this paper is organised as follows: Section II outlines the proposed architecture for the DT prototype; Section III discusses the development of a digital world with digitally simulated elements; and Sections IV and V detail the use cases used to demonstrate the capabilities of the implemented prototype. In each case, existing work is reviewed, the utilised design methodologies are highlighted, and all obtained results are critically evaluated. Finally, Section VI summarises the conclusions and contributions of this work, together with areas meriting further research.

TABLE I
PROPOSED REQUIREMENTS FOR THE DT SYSTEM PRESENTED IN THIS PAPER.

Category	Requirements
Modularity, scalability and flexibility	• Vehicle models, 3D environments and simulated algorithms should be loosely coupled, such that they may be independently developed, tested and integrated into the system, if they adhere to a defined protocol and/or interface. • Static and dynamic objects may be independently developed, tested and included within the 3D environment at specified locations and, where applicable, with specified dynamic behaviour, if they adhere to a defined protocol and/or interface. • The properties and behaviour of objects and 4D regions within the environment should be readily defined and modified. • Sensors may be independently developed, tested and included within the 3D environment or vehicle models, and their relationship with the simulation environment may be customised. • The DT should be easily used with more complex algorithms, larger maps, a larger number of simulated vehicles and/or a wider variety of different static/dynamic objects, assuming the underlying processing and memory requirements are met. • The environment should be easy to simulate at different times of day, lighting conditions, temperatures, pressures and other physical conditions.
Consistency and reliability	• In the absence of stochastic processes, the simulation should return consistent and integral results under identical conditions. • The environment should return statistical test data over multiple simulations, that can be readily used to support research findings or demonstrate the reliability of an algorithm in safety-critical situations. • The environment should be suitable to train intelligent algorithms in preparation for real-world deployment. • Exceptions during run-time should return the appropriate errors, without displaying erroneous (seemingly correct) results.
Security, upgradability and maintainability	• Data links between internal software modules should be secure and clearly defined. • Access rights may be readily customised for different software modules and variables. • Interfaces with external software and/or hardware should adhere to data integrity and cybersecurity standards, or support the inclusion of such features in future system updates. • The software should be easily modified and upgraded by future developers, and follow standard design and coding practices. • The entire system should be well documented, including any known bugs and fixes. • The code must be easily maintained as future bugs and issues arise when the product is used.
Ease of use and learnability	• The product manual should clearly describe all necessary steps to install and use the DT environment, highlighting all system dependencies and version requirements. • A specific coding or technical background should not be needed to use the system, excluding the knowledge and skills required to develop custom algorithms, environments and models when required. • A user-friendly interface should make system modifications, module interfacing and product usage as intuitive as possible. • Both a graphical and command-line interface should be included to cater for different user requirements.
Software-in-the-loop (SIL), HIL and mixed-reality capabilities	• A simulated model may be connected to its physical counterpart for real-time data interchange. • The DT platform should support SIL and HIL testing for a wide variety of different applications and devices. • Mixed-reality simulations may be realised within the DT across the entire mixed-reality spectrum. • Physical tests in remote or controlled environments may be enhanced using virtual obstacles, entities and environments without significantly increasing the associated safety risk.

II. DIGITAL TWIN ARCHITECTURE

A. Software Development Environment

Unreal Engine is used to realise the digital environment within the developed prototype, owing to its compatibility with AirSim. This is a widely used game engine and 3D visualisation tool, designed to offer a highly flexible and robust platform for building and deploying video games across multiple platforms [12]. The cross-platform development software offers a wide range of features, including a robust visual scripting system, a powerful level editor, advanced particle and lighting systems, and an extensive set of tools for creating realistic physics simulations.

The functionality of Unreal is enhanced through Cesium, a versatile geospatial platform that provides high-quality geospatial visualisation capabilities [13]. Notably, the Cesium Ion API allows developers to integrate different 3D content, including photogrammetry, building information modelling (BIM) data, and other 3D data objects.

Microsoft AirSim is further used as a plugin within Unreal to introduce digital UAS and sensor elements within the digital environment. It is an open-source, cross-platform simulation platform that readily incorporates a variety of sensor and vehicle models, supporting the quick deployment of UAS simulations [14]. This is complemented by a high degree of support for custom digital models, custom weather conditions, and a physics engine that accurately simulates the real world. The software also supports HIL simulations, by allowing users to connect physical flight controllers to the simulated environment. These features make AirSim ideal for the DT under consideration, offering a realistic and flexible simulation tool to model emerging AAM vehicles and operations. Nonetheless, modelling complex aerodynamics and other physical phenomena remains a challenging task, and AirSim still struggles to accurately simulate all the nuances of real-world systems. Consequently, the platform may suffer from simulation bias, whereby modelling inaccuracies lead to over-optimistic or over-pessimistic evaluations.

B. Architecture Overview

A unified and modular architecture is proposed to realise the DT system, while managing the interactions between all underlying software applications. A high-level overview of this architecture is illustrated in Fig. 3. This comprises three main parts, namely the real-world elements and functions, the simulated environment realised through Unreal, Cesium and AirSim, and the user-defined scripts, settings and files.

The first part of this architecture involves a flight simulation realised within a custom 3D environment. The underlying static environment can be readily created through Unreal Engine and integrated with the DT. Additionally, the simulated weather can be configured within the AirSim settings file and dynamically varied throughout the simulation, using the default AirSim interface or application programming interfaces (APIs). Notably, live weather data can be used to update the dynamic weather conditions in real time.

Dynamic objects and entities can be simulated within the static environment. When not part of the agents being simulated, such entities can be implemented using C++ code within Unreal. These elements are thereby decoupled from AirSim, enforcing a more robust simulation architecture. Road traffic and people, for instance, can be introduced within the environment for enhanced realism, but do not need to be accurately controlled and monitored throughout the simulation. Similarly, dynamic entities like commercial, business, and general aviation aircraft can be integrated within the environment, allowing the UAS and AAM operations to be tested in non-segregated airspace, while eliminating the need to continually control such traffic throughout a simulation.

Multi-agent simulations can be subsequently realised within the DT through the functionality offered by AirSim, to independently simulate and control multiple UASs. These vehicles can be generated on startup by appropriately configuring the AirSim settings file, or dynamically spawned using the respective APIs. The latter, however, does not currently support the inclusion of a custom set of sensors on board the spawned vehicle. Within the proposed architecture, the set of simulated UASs are realised as a database of Python objects. Each vehicle is an instance of a UAV class, which can be controlled through its appropriate attributes. This enforces a consistent user experience and reduces the risk of coding errors, by eliminating the need for users to become familiar or interact with AirSim and other external APIs.

The simulated agents can be controlled through a set of Python scripts running in an independent thread. These are managed by a user-customisable main file termed the *ScriptCore*. This object-oriented approach enables a user to dynamically switch between different classes with the same class and function names, but different function definitions.

This architecture thereby offers a highly modular system, whereby user-defined scripts can be introduced in a plug-and-play fashion. Additionally, all architectural elements are loosely coupled, allowing for the independent development and integration of 3D worlds, digital entities, digital models and simulation scripts. The interactions between these elements are streamlined and abstracted from the user for better compatibility and ease-of-use. Additionally, the Tello drone discussed in [5] and its associated API are considered as representative physical entities of this DT.

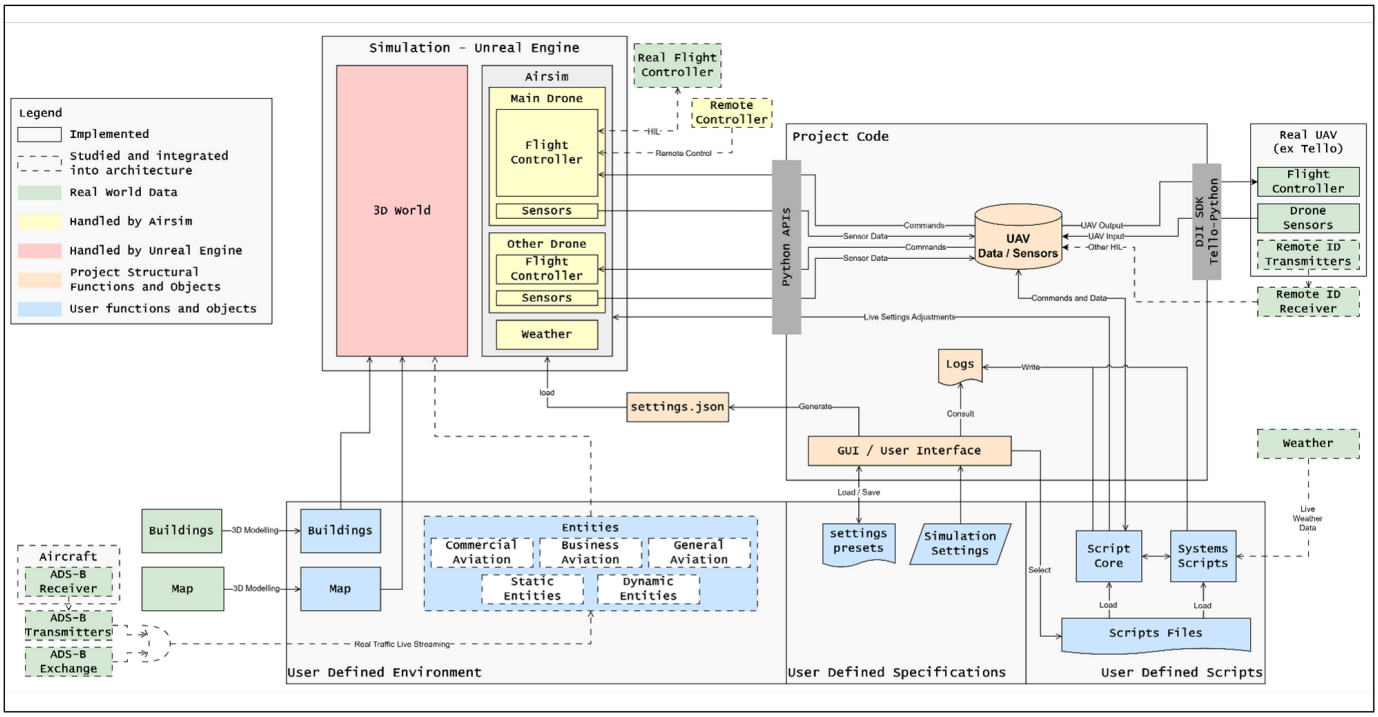


Fig. 3. High-level overview of the proposed architecture.

C. Code Structure

Within the simulation thread, a user can readily introduce custom scripts to realise a wide variety of different simulation scenarios. Notably, these scripts can be written in any programming language, provided that the respective file can be called and executed in Python, and can appropriately interact with the database of UAV objects.

In line with the proposed architecture, the developed code is illustrated in Fig. 4 and involves a database of UAV and Sensor Python objects, instantiated when launching a simulation thread. Each UAV object is assigned a set of fixed attributes on instantiation, including its name, initial pose, list of on-board sensors and whether it represents a virtual or physical UAS. Similarly, each Sensor object is assigned a fixed name, type and pose. Additionally, each object comprises a set of attributes that reflect the current vehicle state or sensor reading. These are dynamically updated throughout program execution to reflect the most recent data obtained from the physical or virtual environments. Each UAV object further contains a set of dynamically assigned attributes that reflect the next action each vehicle must take during the subsequent simulation time-step. This object-oriented architecture allows user-customisable scripts to be decoupled from interactions with external software environments and UAS platforms. This enhances the security of the DT and facilitates the use of the overall system, by eliminating the need to understand and interpret a variety of different interface protocols and programming libraries.

For demonstration purposes, Tello drones are used to realise the mixed-reality demonstrations discussed in [5]. The ability to interact with these UASs is thereby integrated within the developed prototype. In particular, interactions with external interfaces are handled through a set of custom scripts, namely:

- *AirSim Input*: used to extract the vehicle state and sensor data of all virtual entities at each simulation timestep, and update the attributes within the respective Python objects.

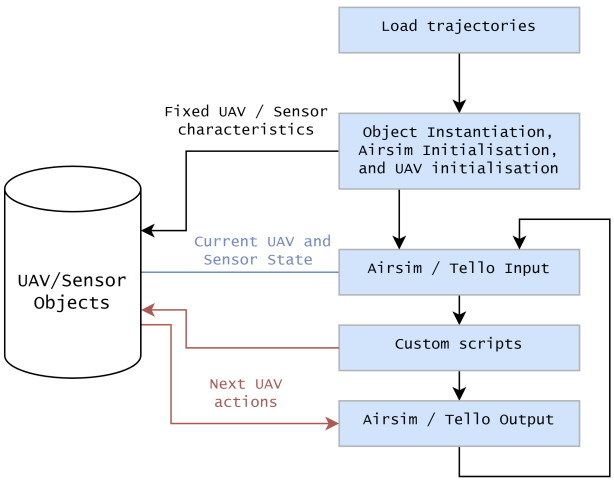


Fig. 4. Structural overview of the developed code.

- *AirSim Output*: used to send commands to virtual UASs, according to the attributes of each respective Python object.
- *Tello Input*: used to extract the vehicle state and sensor data of all real Tello drones at each simulation timestep, and update the attributes within the respective Python objects.
- *Tello Output*: used to send commands to physical Tello drones, according to the attributes of each respective Python object.

To further highlight the modularity and flexibility of the proposed code structure, a set of three user-customisable scripts are developed, namely:

- *Script Core*: used to manage the flow of each simulation, by instantiating all Sensor and UAV objects; loading the appropriate UAS trajectories; instantiating the appropriate Flight Manager and Collision Avoider instances; and sequentially calling the appropriate input, output, Flight Manager and Collision Avoider functions at each simulation time-step.
- *Flight Manager*: used to signal whether each UAS has started or completed its assigned trajectory, and update the nextWaypoint attribute of each vehicle as the simulation progresses. For simplicity, UAS trajectories are defined within a JavaScript Object Notation (JSON) file as a sequence of waypoints, each identified by a 3D North, East, Down (NED) coordinate, the speed at which the UAS should travel to the waypoint, and the time of departure from each waypoint.
- *Collision Avoider*: used to determine the next action each UAS must take, according to the vehicle state, sensory data and mission requirements.

D. Simulation Flow

The flow of a typical simulation is illustrated in Fig. 5, whereby data from the physical and virtual worlds is gathered, processed, and used to determine the actions taken in the next simulation timestep in a closed-loop fashion. This assumes that the user has already generated the appropriate simulation settings, launched the simulation engine, and started the execution of the simulation scripts.

The dynamic equations embedded within Unreal and AirSim first compute the next pose of each simulated object, according to the previous state of the vehicle, the transmitted command and the state of its surrounding environment. Such poses allow the scene to be appropriately rendered and provide all necessary data to the digital sensors modelled in AirSim. The simulation scripts can subsequently collect data from the physical and virtual environments and appropriately transmit commands to the physical and virtual vehicles. During each loop, the scripts can also access a log file and append information to it. Finally, in the absence of HIL tests or custom flight controllers, the default AirSim flight controller executes the transmitted high-level commands. A joystick can instead be utilised to manually control the main UAS.

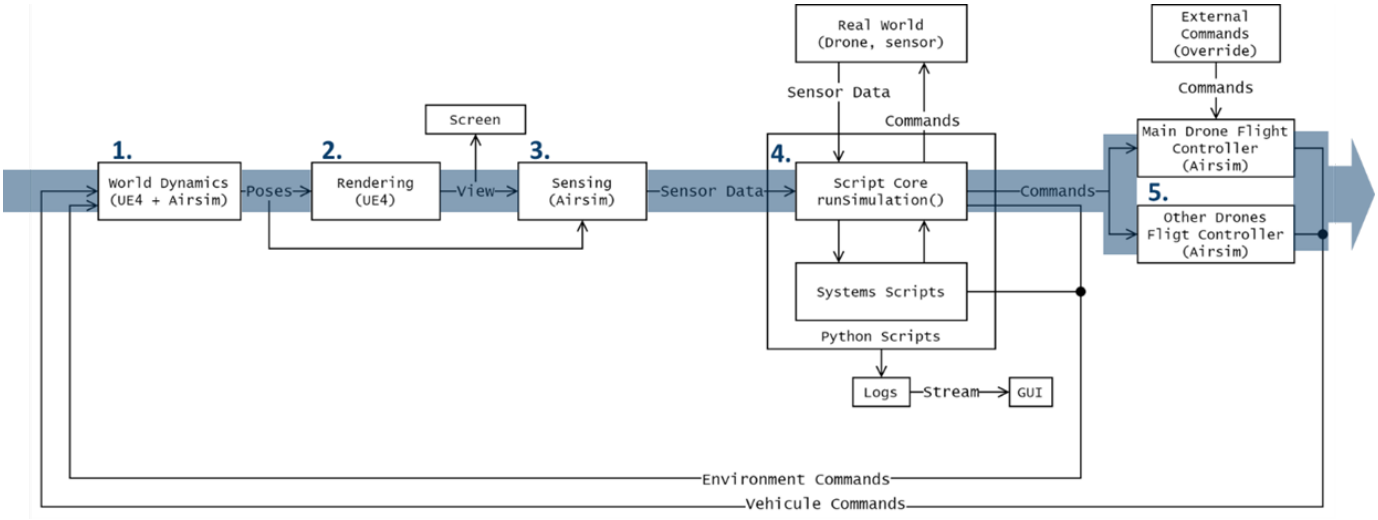


Fig. 5. Overview of a typical simulation flow using the developed DT.

III. Digital World vs. Physical World

A. Context

3D environments are central to the digital half of a DT, and reflect the world within which digital elements and vehicles are deployed. These are often built by superimposing a sequence of layers with increasing levels of complexity and functionality to realise complex outdoor environments, as discussed in [5]. Owing to the small size of many UASs, such vehicles have also been widely employed in indoor environments for inspection, surveillance or search and rescue missions. Moreover, indoor spaces typically offer a safer and more controlled test environment. Consequently, a robust DT should support the modelling of both outdoor and indoor digital environments and elements.

A review of outdoor environments within existing simulation platforms [15]–[17] suggests that outdoor digital worlds should respect the laws of physics while incorporating a high-degree of textural realism to support vision-based applications. Moreover, real-time environmental data can be incorporated for enhanced simulation capabilities. Additionally, the environment should be easily customisable and scalable for larger system deployments, whereby multiple digital elements can be independently realised and simulated. A review of indoor digital environments [18], [19] further suggests that fidelity and accuracy become more critical in indoor digital worlds, where UASs must typically execute tighter manoeuvres to avoid obstacles and respect spatial room constraints.

B. Design Methodologies

Several approaches can be taken to model a 3D digital environment, depending on the resources and data available, as presented in [5]. Apart from the co-simulation capabilities discussed in [11], each proposed design technique is investigated throughout this prototype, and used to implement digital environments of several locations at Cranfield University and Cranfield Airport.

The methodology proposed and used to model the outdoor environment of Cranfield University is illustrated in Fig. 6. The simplest approach to create such a digital world involves taking advantage of the 3D data within Google Earth, which boasts very high-fidelity models of many geographical locations. This data can be extracted, imported into Blender as a 3D mesh for fine-tuning, and exported to Unreal Engine. No such 3D data, however, is currently available for the area surrounding Cranfield University and Airport. Similarly, no existing 3D model of the area has yet been developed.

In the absence of 3D models or data, the digital environment must be manually created within a development engine like Unreal, possibly using plugins like Open Street Maps (OSM) and Cesium to access existing databases of information. Initially, community-developed assets can be used to integrate standard features of an outdoor environment. When a specific asset is unavailable, 3D modelling software can be used to modify similar readily available assets and better represent the true physical entity. If no similar asset is available, the entity must be manually modelled in software such as Blender, possibly using photogrammetric or measurement data for better accuracy.

Laser or photogrammetric-based modelling becomes more feasible in enclosed indoor spaces. Notably, a hybrid approach is proposed and used to model an indoor lab within Cranfield University, according to the methodology depicted in Fig. 7. This suggests that an appropriate LiDAR scanner and photogrammetric camera must first be selected according to the system requirements, and used to collect the necessary data points. For this prototype, standard applications were used with an iPhone14/iPadPRO LiDAR camera to demonstrate such a process. The data must be subsequently processed and manipulated within a 3D modelling software such as Blender to clean the underlying mesh, possibly using custom textures for greater realism. Finally, the mesh and textures can be imported into Unreal Engine and integrated within the DT.

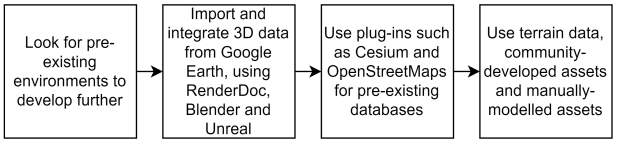


Fig. 6. Methodology used to create the outdoor digital environment.

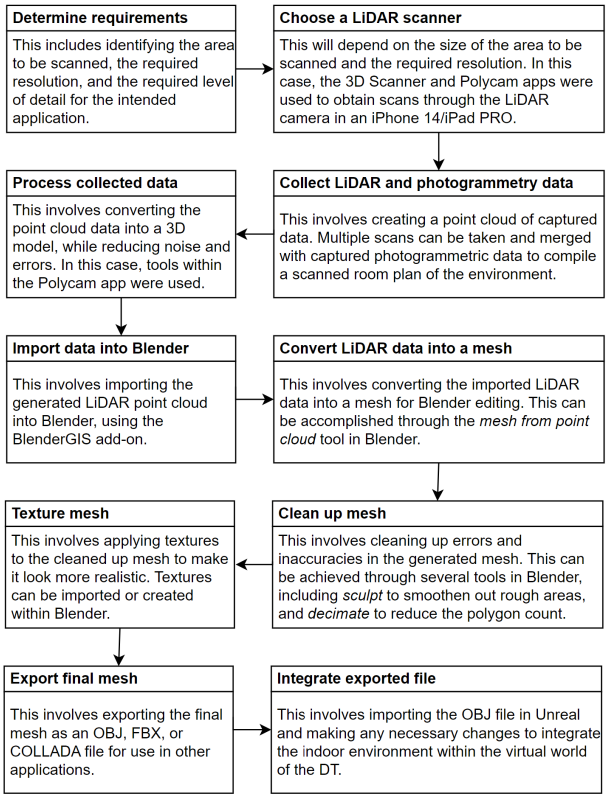


Fig. 7. Methodology used to create the indoor digital environment.

C. Implementation Outcome

The outdoor environment modelled using Unreal, Cesium and OSM is shown in Fig. 8, and suggests that a preliminary outdoor digital replica of Cranfield University and Airport was successfully created. Notably, several assets and features, such as lampposts, fences, trees, windows and aircraft were successfully introduced within the environment and a plane was successfully programmed to periodically take-off and land at Cranfield Airport. This suggests significant developments from the initial digital world developed in [5].

Similarly, the modelled lab and neighbouring corridor are shown in Fig. 9. A high-fidelity indoor environment was successfully created using LiDAR scanning and photogrammetry techniques, including all key elements within the two indoor spaces. High-fidelity replicas of an indoor flight arena at Cranfield University and a Custom UAS mesh were also developed using 3D modelling techniques and utilised for initial mixed-reality demonstrations. These, however, were presented in [5] and are not reproduced in this paper.



Fig. 8. Modelled features and assets within the outdoor environment.



Fig. 9. Modelled indoor environment.

D. Evaluation and Discussion

The accuracy of the digital worlds was quantitatively evaluated by considering the root mean square error (RMSE) and mean absolute error (MAE) between distance measurements in the physical and virtual worlds. Within the outdoor environment, mobile GPS measurements were used to determine the true location between several key locations at Cranfield University. These were compared to corresponding distances in the digital world, yielding the results shown in Fig. 10. This suggests an RMSE of $63\,\mathrm{m}$ and an MAE of $76\,\mathrm{m}$, likely owing to the use of OSM and satellite imagery to position buildings within the environment, having a high associated inaccuracy in sparsely populated environments like Cranfield. This error is significantly improved in [11], when updating the outdoor map using high-fidelity models based on a high-accuracy computer aided design (CAD) model of Cranfield University.

Similarly, the accuracy of the indoor model was evaluated by comparing measurements within the digital world to those taken using a laser distance meter in the physical environment, yielding the results shown in Fig. 10. This suggests an RMSE of $0.22\,\mathrm{m}$ and an MAE of $0.114\,\mathrm{m}$, confirming that a precise digital indoor environment was successfully created. Nonetheless, a qualitative analysis suggests that further work is required to improve textural realism and achieve a photo-realistic representation of the indoor space.

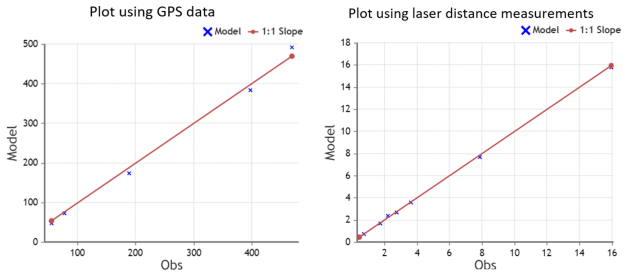


Fig. 10. Distance measurements obtained from the digital model against the true measurement observations for outdoor (left) and indoor (right) environments.

IV. Multi-agent Case Study

A. Setup

Multi-agent systems (MASs) are a class of distributed systems composed of multiple agents that interact with each other to achieve a common goal. This offers numerous advantages over single-agent systems, including increased scalability, robustness, and flexibility. Notably, in AAM and UAS operations, MASs allow vehicles to adapt to changing environmental conditions or system requirements, while accomplishing tasks that would be impossible for a single vehicle to perform alone. Such multi-agent frameworks would benefit from the capabilities of a robust DT for the deployment and testing of their underlying algorithms.

In general, MASs can be procedural or intelligent. Procedural approaches are often the simpler, and will likely be used during the initial stages of AAM roll-out. These promote co-operative coordination and information sharing amongst agents, to cohesively work towards a common goal. Conversely, intelligent algorithms are often required for complex and densely populated environments.

B. Experimentation

The ability to readily deploy multi-agent algorithms within the developed DT is demonstrated through a collaborative surveillance use case. A number of virtual drones are instructed to autonomously fly along pre-defined trajectories, with each drone periodically capturing images using a downward-facing camera. Captured images are stitched together in real-time to give a larger panoramic view of the region being surveyed. For simplicity, the trajectories are assumed to ensure that any two images being stitched together have the same orientation and an overlapping image segment.

Computer vision algorithms are used to stitch newly captured images to the current panoramic view in real time, according to the methodology outlined in Fig. 11. Common points amongst successive images are located, and used to calculate the homography matrix that relates the two images. This is then used to warp and blend the images, through readily available functions from open-access Python libraries. The stitched aerial images of an area are thereby produced and can be used for a range of surveillance or mapping tasks.

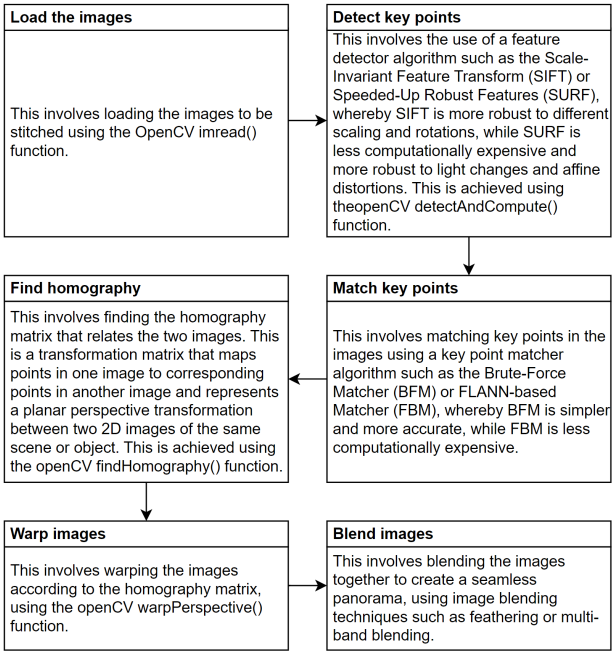


Fig. 11. Methodology used for image stitching during collaborative surveillance.

A significant limitation of this implementation is that the ability to stitch images depends on the frequency at which images are taken, and the need for common features between successive images. In fact, the algorithm is likely to fail in regions with few distinctive features. Nonetheless, this approach was deemed sufficient for a preliminary proof of concept demonstration, and fine-tuning the image stitching algorithm falls beyond the scope of this work.

C. Results and Discussion

Raw images taken by the UASs were successfully combined into a single stitched image in real-time with minimal distortion, as shown in Fig. 12. Despite only implementing a basic stitching algorithm, this suggests that more complex techniques can be readily implemented for higher resolution and more reliable results. Notably, the drones can be instructed to fly at higher altitudes for greater coverage, and a more high-fidelity environment could be used to facilitate the identification of distinct features in successive images. In fact, the algorithm occasionally failed within a low-fidelity environment, owing to insufficient distinct features across multiple images. Moreover, the camera feeds can be augmented with positioning data to better handle non-overlapping images. In general, however, this proves that procedural multi-agent algorithms can be readily integrated within the DT.

Similarly, trained intelligent multi-agent algorithms can be easily deployed in a similar fashion. Notably, AI-based systems can also be trained within the developed prototype during the initial stages of development. Additionally, the ability to introduce mixed-reality features within such demonstrations was demonstrated in [5].

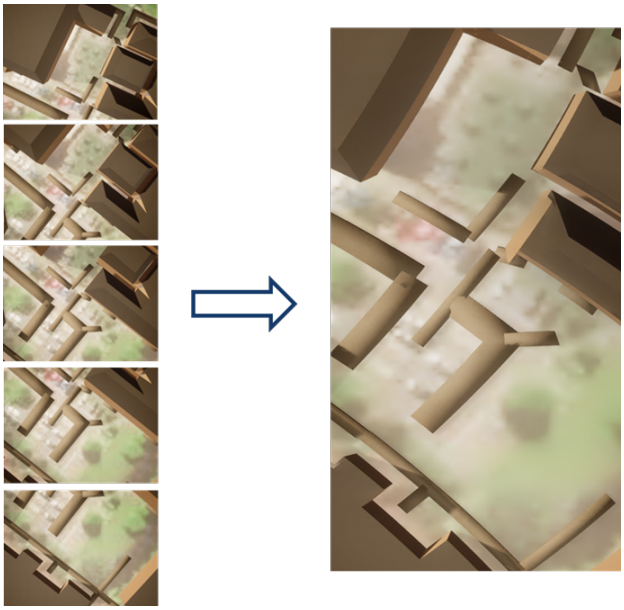


Fig. 12. Combination of raw images into a single stitched image.

V. UTM Case Study

A. Setup

Several UTM architectures have been proposed to address the emerging needs of AAM. Nonetheless, no globally harmonised solution has yet been identified. DTs thereby offer an ideal platform to experiment with and develop emerging and conceptual traffic management systems. Most UTM services require a fully deployed system within non-segregated airspace to appropriately consider inter-UAS communication links and interactions with ATM systems. Such testing, however, cannot be reliably achieved in the early stages of development, even when supported by a DT platform. Nonetheless, simpler services can be readily deployed in segregated environments, and easily tested during the initial stages of UTM development. Notably, flight plan authorisation and conformance monitoring represent two core services that can be easily deployed.

B. Experimentation

Elements of a UTM framework can be readily introduced within the proposed architecture, owing to the high degree of modularity and flexibility within the underlying code structure. Notably, flight plans can be sequentially requested from the user to detect and display any potential conflicts. For simplicity, only two flight plans are considered in this demonstration. If a conflict is detected, the user is asked to re-input the second flight plan until the two are found to be strategically deconflicted. At each simulation timestep, the state of each UAS is subsequently analysed to determine whether or not it has violated its allocated operational volume. If the constraints have been violated, appropriate action may be taken to signal lack of conformance or take any necessary corrective actions.

Two different methods are identified to generate operational flight volumes and identify conflicts, namely:

- *A vectorial approach*: This involves defining operational volumes using the waypoint coordinates of each trajectory. Each segment of its operational volume is created as a rectangle, defined using the coordinates of its vertices. The four edge equations of each rectangle can be used to determine whether or not a specified coordinate falls within the enclosed segment. This method, however, is hard to implement and computationally expensive, owing to the large number of required vectorial checks.
- *A grid-based approach*: This involves a 2D map that contains all the submitted flight plans, such that image processing techniques can be used to detect any conflicts. The operational volumes are generated as a sequence of circles along the path, yielding an overall rounded rectangular shape. Binary operations are subsequently used to generate the union of all operational volumes, such that any strategic conflicts can be detected by analysing the resulting image, as illustrated in Fig. 13. During flight, the position of a UAS can be translated to a position on the respective map, and pixel-based techniques can be readily used to determine whether the vehicle falls inside or outside its assigned operational volume.

The grid-based method is used throughout this work, owing to the shorter development time and greater algorithm efficiency. Since an image discretises the underlying map into pixels, however, this method introduces a precision bias linked to the scale of the map in pixels per meter. Throughout this project, this scale is defined as 10 pixels per meter.

C. Results and Discussion

On launching a simulation, the user was successfully instructed to select trajectories for two UASs. If both trajectories were strategically deconflicted, the flight plans were authorised and the simulation could commence. If not, the conflicting paths were displayed as in Fig. 14, and the user was repeatedly asked to re-input the second trajectory until it did not conflict with the first authorised flight plan. This reflects a basic first-come-first-served prioritisation scheme, where the second flight plan must account for the restrictions imposed by the operational volume assigned to the first flight plan.

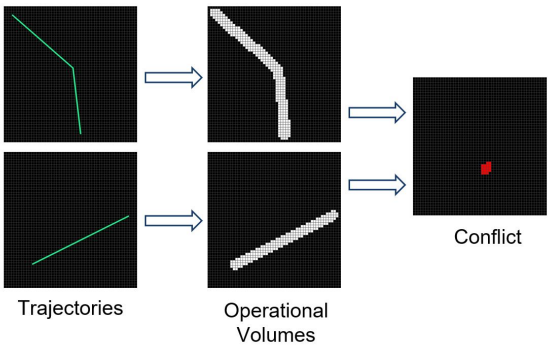


Fig. 13. Grid-based approach for strategic conflict detection.

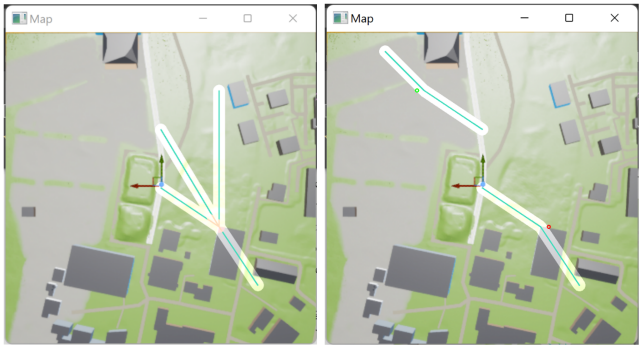


Fig. 14. Display showing a strategic conflict (left) and real-time display with a conforming UAS in the top operational volume and a non-conforming UAS in the bottom operational volume (right).

After successfully assigning two deconflicted flight plans to the simulated UASs, each vehicle was instructed to traverse its respective operational volume. A radar-like display was simultaneously displayed, showing the trajectories, operational volumes and real-time positions of the UASs. If a UAS exited its assigned volume, the corresponding radar plot turned red to signal lack of conformance, as highlighted in Fig. 14.

These demonstrations effectively showcase the ability of the developed prototype to simulate both pre-flight and in-flight UTM services. Moreover, more sophisticated algorithms can be implemented to demonstrate a more complete and realistic UTM framework. Notably the services can be readily extended to any number of UASs and flight plans, and more complex prioritisation schemes can be implemented. Time-based deconfliction can also be introduced, whereby flight plans can intersect if the UAS flights are scheduled to not enter the conflicting region simultaneously. This is particularly important to support the business case of emerging UAS operations like air taxis and on-demand deliveries, where operations cannot be scheduled in advance.

VI. Conclusions and Further Work

This paper built on the work outlined in [5] to develop a flexible, modular and scalable DT architecture with enhanced SIL, HIL and mixed-reality capabilities, through a case study of Cranfield University and Airport. A custom Python-based code architecture was designed and implemented to complement the developed DT, supporting a secure object-oriented framework and the inclusion of user-customisable scripts in a highly modular plug-and-play fashion. Digital replicas of indoor and outdoor environments within Cranfield University and Airport were successfully created using Unreal, OSM and Cesium, integrated with the digital elements offered by AirSim, and consolidated into a unified Python-based DT architecture. The developed digital worlds were comprehensively evaluated and shown to exhibit good dimensional and textural accuracy. Moreover, several demonstrative use cases were deployed on the DT prototype, highlighting the ability of the DT to support the development and testing of multi-agent systems and UTM ecosystems.

Future work will aim to fine-tune the prototype and refine the implemented digital worlds to better reflect reality, possibly utilising photogrammetric data to enhance the accuracy of digital environments. Similarly, further tests will be conducted to demonstrate the co-simulation and HIL capabilities of the prototype. Additionally, more advanced multi-agent and mixed-reality simulations will be deployed and tested within the DT. Some of this work is conducted in [11].

VII. Acknowledgment

This research is funded by the UKRI Future Flight Challenge Phase 3 project HADO, 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] CoflightCouldServices, "A significant growth in Air Traffic by 2035," Tech. Rep., 2020.
[2] MarketsandMarkets, "UAV Market by Point of Sale, Systems, Platform (Civil Commercial, and Defense Government), Function, End Use, Application, Type (Fixed Wing, Rotary Wing, Hybrid), Mode of Operation, Mtow, Range Region - Global Forecast to 2027," Tech. Rep., 2021.
[3] EASA, "Specific Operations Risk Assessment," Tech. Rep., 2019.
[4] C. Dilmegani, "15 Digital Twin Applications/ Use Cases by Industry in 2023," AI Multiple, Tech. Rep., 2023. [Online]. Available: https://research.aimultiple.com/digital-twin-applications/
[5] J. Zhao, C. Conrad, Q. Delezenne, Y. Xu, and A. Tsourdos, "A Digital Twin Mixed-reality System for Testing Future Advanced Air Mobility Concepts: A Prototype," in *The 23rd Integrated Communications, Navigation and Surveillance Conference*, 2023.
[6] Microsoft, "Project AirSim for aerial autonomy," Tech. Rep., 2022. [Online]. Available: https://www.microsoft.com/en-us/ai/autonomous-systems-project-airsim?activetab=pivot1:primaryr3
[7] Quantum3D, "UAV Simulator: fixed wing Simulators," Tech. Rep., 2022. [Online]. Available: https://quantum3d.com/uav-simulator/
[8] S. Kaputsos, "UAV Pilot Simulator," MIT, Tech. Rep., 2022. [Online]. Available: https://www.media.mit.edu/projects/cloud-uav-sim/overview/
[9] C. Boje, A. Guerriero, S. Kubicki, and Y. Rezgui, "Towards a semantic Construction Digital Twin: Directions for future research," *Automation in Construction*, vol. 114, pp. 103–179, 2020.
[10] C. Conrad, Q. Delezenne, A. Mukherjee, A. Mhowwala, and M. Ahmed, "Developing a Digital Twin System to Test Intelligent Solutions for AAM Operations," Master's thesis, Cranfield University, 2023.
[11] J. Zhao, C. Conrad, R. Fremond, A. Mukherjee, Q. Delezenne, Y. Su, Y. Xu, and A. Tsourdos, "Co-simulation Digital Twin Framework for Testing Future Advanced Air Mobility Concepts: A Study with BlueSky and AirSim," in *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, 2023, unpublished.
[12] EpicGames, "Unreal Engine: The world's most open and advanced real-time 3D creation tool," Tech. Rep., 2023.
[13] Cesium, "The Platform for 3D Geospatial," Tech. Rep., 2023.
[14] Microsoft, "Airsim," Tech. Rep., 2021. [Online]. Available: https://microsoft.github.io/AirSim/
[15] CoppeliaRobotics, "Coppelia Sim from the creators of V-REP," Tech. Rep., 2023. [Online]. Available: https://www.coppeliarobotics.com/
[16] A. Mairaj, A. I. Baba, and A. Y. Javaid, "Application Specific Drone Simulators: Recent Advances and Challenges," *Simul. Model. Pract. Theory*, vol. 94, pp. 100–117, 2019.
[17] E. Capello, G. Guglieri, and F. Quagliotti, "UAVs and Simulation: an Experience on MAVs," *Aircraft Engineering and Aerospace Technology*, vol. 81, pp. 38–50, 2009.
[18] Y.-W. Lin and C. J. Spanos, "Developing a Digital Twin for Indoor Environments: A Case Study," Master's thesis, University of California at Berkeley, 2021.
[19] Y. Zou, F. Ye, A. Li, M. Munir, M. Munir, and E. Hjelseth, "A Digital Twin Prototype for Smart Parking Management," in *European Conference On Product And Process Modelling (ECPPM) 2022*, 2022.



https://dspace.org/

Developing a digital twin for testing multi-agent systems in advanced air mobility: a case study of Cranfield University and airport

Conrad, Christopher; Delezenne, Quentin; Mukherjee, Anurag; Mhowwala, Ali Asgher; Ahmed, Mohammad ...

2023-11-10
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/DASC58513.2023.10311333
Conrad C, Delezenne Q, Mukherjee A, et al., (2023) Developing a digital twin for testing multi-agent systems in advanced air mobility: a case study of Cranfield University and airport. In: IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC) 2023, 1-5 October 2023, Barcelona, Spain

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