Realistic Trajectory Generation in Simulated Environments for U-space Systems Assessment

Alex Sanchis, Abraham Garcia, Silvia Esparza, Juan V. Balbastre
Air Navigation Systems (SNA)
Universitat Politècnica de València (UPV)
Valencia, Spain
{asanmar4, agariba, silesbe1, jbalbast}@upv.edu.es

***Abstract*—As unmanned aircraft systems (UAS) gain traction in civil airspace, the ability to generate and analyze realistic flight trajectories is critical for advancing U-space services and ensuring safe integration. This paper outlines an upcoming study on creating high-fidelity UAS trajectories using the PX4 autopilot software-in-the-loop (SITL) environment with Gazebo as the simulation platform. The objective is to replicate flight behaviors that closely approximate real-world conditions, accommodating variables such as wind effects, data link disruptions, sensor/system failures, RC link loss, and battery dynamics, all of which are essential for a comprehensive assessment of UAS operational resilience.**

To support large-scale trajectory dataset generation, the proposed framework integrates a parallelized execution strategy using Docker-based fast-time simulation. This implementation significantly improves computational efficiency, achieving up to a tenfold increase in processing speed compared to conventional methods. The study further analyzes the impact of environmental conditions on processing time, demonstrating that additional perturbations do not introduce significant computational overhead.

Additionally, we will explore initial proposals for data structures, databases, and interoperability frameworks that facilitate seamless integration with existing conflict detection systems, ensuring that future U-space services can be efficiently developed. Aligned with Eurocontrol's BADA (Base of Aircraft Data) model for UAS, this work seeks to contribute a robust dataset and methodology to support enhanced airspace management, conflict resolution, and safety metrics within U-space. The ultimate aim is to provide insights into the operational viability of simulated UAS trajectories and inform future U-space applications, such as urban air mobility, surveillance, and logistical operations.

***Index Terms*—UAS trajectory simulation, PX4 SITL, U-space, fast-time simulation.**

I. Introduction

Integrating Unmanned Aerial Systems (UAS) into civil airspace presents significant challenges for trajectory prediction, a fundamental requirement for ensuring safe and efficient U-space operations. Unlike conventional aircraft, which rely on standardized performance models such as the Base of Aircraft Data (BADA), UAS lack a widely accepted reference model. This absence complicates the precise forecasting of drone trajectories, particularly under dynamic environmental conditions such as wind disturbances, leading to increased uncertainty in airspace management.

CIAICO/2022/044 - Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital of the Generalitat Valenciana

This study proposes a methodology for generating realistic UAS trajectories using a high-fidelity Software-in-the-Loop (SITL) simulation environment based on PX4. By leveraging this approach, it is possible to model complex drone behaviors while accounting for external perturbations, enabling a more accurate representation of real-world flight execution. Additionally, the ability to execute SITL-generated trajectories on physical drones with minimal deviation ensures that the proposed framework closely mirrors real-world operations, making it highly relevant for regulatory assessments and airspace optimization.

To enable large-scale trajectory dataset generation, a parallelized processing framework has been implemented using Docker-based fast-time simulation. This allows the execution of multiple flight plan simulations in parallel, significantly improving computational efficiency. The developed system achieves a tenfold increase in simulation speed compared to traditional sequential execution, reaching a processing efficiency ratio of up to $\times 100$ while maintaining accuracy. Furthermore, the study evaluates how different flight conditions, including wind disturbances, impact computational performance, concluding that environmental perturbations do not significantly alter execution time.

The datasets generated through this methodology provide valuable insights for strategic deconfliction, regulatory assessments, and trajectory-based airspace management. The study also demonstrates that simulated flight plans can be directly transferred to real drones with high fidelity, reducing the uncertainty in trajectory prediction and potentially minimizing reserved 4D airspace volumes for UAS operations. This work contributes to the ongoing development of U-space services, with direct applications in projects such as U-AGREE and CREATE, supporting the implementation of trajectory validation and flight plan approval systems.

This paper is structured as follows: Section II discusses the need for realistic UAS trajectories and reviews existing trajectory generation methodologies. Section III details the proposed framework, covering flight plan generation, SITL simulation, and parallelized execution. Section IV presents the implementation architecture and system optimization strategies. Section V evaluates computational performance, trajectory deviations, and processing efficiency under different conditions. Finally, Section VI summarizes key findings and

outlines future research directions.

II. Background

A. The Need for Realistic UAS Trajectories

Ensuring a safe and efficient airspace for UAS operations requires highly accurate trajectory predictions. Unlike traditional aircraft, drones lack a standardized performance model such as BADA, making it challenging to estimate their real-world flight behavior. Consequently, trajectory modeling for UAS has primarily relied on simplified approaches that introduce significant limitations in their applicability for regulatory and operational studies.

Previous studies, such as those conducted within the BUBBLES [1] and U-AGREE [2] projects, have employed basic trajectory models based on uniform motion with an added random directional error. These models assume a drone follows a straight-line trajectory between waypoints with a constant velocity, introducing minor deviations to account for potential environmental disturbances. While this approach provides a first-order approximation of trajectory uncertainty, it fails to capture the complex interactions between drones and atmospheric conditions, particularly in autonomous operations where external disturbances such as wind significantly impact flight paths.

Additionally, within the BUBBLES project, real-world drone flights operated by human pilots were analyzed to assess the feasibility of detection of pairwise conflicts between UAS in urban airspace [1]. However, human-operated flights introduce additional uncertainty, as pilot reactions to external factors vary widely. This variability raises concerns about the reliability of such trajectories for regulatory purposes, particularly when determining separation minima or lateral buffer zones. If real-world deviations from planned trajectories are taken at face value, one might conclude that extensive airspace reservations are necessary to ensure safe drone operations. However, this assumption becomes problematic in a future scenario where thousands of autonomous drones share the airspace, as excessive separation requirements would severely limit operational efficiency.

The U-space framework requires highly realistic trajectory models to enable accurate airspace assessments, regulatory evaluations, and standardization efforts. Realistic trajectory generation is essential for designing operational constraints that balance safety and efficiency while avoiding unnecessarily large airspace reservations. By leveraging high-fidelity simulation environments that incorporate realistic flight dynamics, wind effects, and system degradations, U-space services can ensure that future regulatory frameworks are based on trajectories that closely represent autonomous drone behavior.

B. Simulation-Based Approach for Trajectory Generation

Given the absence of a predefined UAS performance model, a simulation-based approach becomes the most viable alternative for generating realistic drone trajectories. By leveraging a high-fidelity 3D flight simulator integrated with a Software-in-the-Loop (SITL) autopilot, it is possible to model how a drone interacts with its environment under various operational conditions. This methodology enables detailed analysis of trajectory deviations caused by external factors such as wind dynamics, ensuring a more accurate representation of real-world flight behavior.

One of the most advanced and widely adopted solutions for UAS simulation is PX4, an open-source flight control software designed for drones and other unmanned vehicles. PX4 provides a standardized ecosystem that supports a wide range of drone hardware and software, making it an ideal choice for trajectory generation and validation. Thanks to its modular architecture, PX4 allows seamless integration of different components while maintaining robustness and performance, ensuring that simulations closely mirror real flight scenarios.

A key advantage of PX4 is its compatibility with MAVLink, a widely used communication protocol in the UAS industry. This enables the loading and execution of predefined flight plans across most commercially available drone platforms, ensuring that simulations performed in SITL can later be tested and deployed in real-world operations. This interoperability makes PX4 an optimal solution for studying autonomous UAS behavior, as it allows for a direct transition from simulation to physical flight testing. Furthermore, PX4's extensive validation in real-world applications reinforces its reliability, with thousands of commercial UAS systems currently using PX4-based flight stacks and ongoing flight tests continuously refining the software's safety and performance.

Alternative simulation solutions were also considered. MATLAB's UAV Toolbox, in combination with X-Plane, provides a robust framework for developing and testing UAS autopilot algorithms [3]. This approach offers extensive flexibility, allowing for the customization of control laws and flight dynamics, making it well-suited for research in novel guidance and navigation algorithms. However, this flexibility also introduces significant complexity, requiring custom modeling of each UAS type. Moreover, while MATLAB facilitates algorithmic development, the integration with real-world drones is not as seamless as with PX4.

Another widely used autopilot is ArduPilot, which, combined with Mission Planner, offers an alternative for simulation and trajectory validation [4]. ArduPilot provides robust SITL and Hardware-in-the-Loop (HITL) capabilities, enabling users to test flight behavior in a simulation environment before executing the same mission on a physical drone. Studies such as [5] have demonstrated that trajectories generated in SITL closely match those observed in real flight tests, validating the effectiveness of this approach for trajectory prediction.

Ultimately, PX4 was selected as the most suitable platform for this study due to its modern architecture, widespread adoption, and continuous development by the open-source community. Unlike MATLAB-based solutions, which require full autopilot implementation, PX4 provides ready-to-use configurations for a wide variety of commercial drones, making it a practical choice for research focused on trajectory generation rather than low-level control algorithm development. Similarly,

while ArduPilot presents an alternative with HITL validation capabilities, PX4 offers a more intuitive interface, better community support, and a higher degree of maintainability, ensuring compatibility with existing U-space systems and future operational deployment.

C. The Need for Large-Scale Flight Plan Dataset Generation

As the density of UAS operations increases, the ability to generate large volumes of flight plans becomes crucial for evaluating the scalability and safety of U-space services. Unlike manned aviation, where structured flight routes and predefined airspace reservations dominate, drone operations introduce various flight behaviors, requiring flexible and extensive datasets to support regulatory and operational studies.

Flight plans define the intended trajectory of a drone by specifying a sequence of waypoints and operational parameters. To properly model real-world U-space scenarios, it is essential to consider multiple flight plan types that reflect the expected diversity of drone operations. These include:

- Survey and mapping (SCAN): Characterized by structured grid-like trajectories, commonly used for environmental monitoring, infrastructure inspection, and search-and-rescue missions.
- Delivery operations: Representing autonomous logistics applications where drones transport packages between predefined locations, often following optimized paths to minimize travel time and energy consumption.
- Fixed-route flights: Covering operations between established locations of interest, such as hospital networks, logistic hubs, or emergency response centers, requiring reliable and repeatable flight paths.
- Randomized and ad-hoc trajectories: Reflecting unstructured drone movements, including recreational flights, testing missions, or dynamically planned routes influenced by real-time conditions.

To assess the impact of drone operations on airspace efficiency and safety, large-scale dataset generation is necessary to simulate high-traffic conditions and analyze potential conflicts. The capability to generate diverse and representative flight plans ensures that airspace models can capture the variability of UAS behavior and provide meaningful insights for future regulatory frameworks.

Furthermore, evaluating the interaction of these flight plans in dense airspace environments allows for identifying potential bottlenecks, optimal separation criteria, and the effectiveness of traffic management solutions. Developing realistic and scalable flight plan generation methodologies will ensure safe and efficient drone integration into shared airspace as U-space matures.

D. Use Cases and Applications

The need for realistic UAS trajectories extends beyond theoretical modeling, as such datasets are actively being utilized in various research and operational studies. For instance, realistic trajectory data has been employed for UAV positioning in urban environments leveraging 5G networks [6] and in strategic flight plan approval and conflict detection systems using multi-USSP architectures [7]. These studies highlight the growing demand for accurate trajectory representations to enhance U-space safety and efficiency.

Realistic trajectory datasets are fundamental to multiple U-space services. Among others, they play a key role validating designs of Traffic Information Services (TIS) [8] and Flight Authorization Services (FAS) [7]. The FAS, which ensures compliance with airspace constraints and regulatory frameworks, benefits from realistic trajectories to validate conflict detection algorithms and improve authorization accuracy [7]. Additionally, U-space separation management projects such as SPATIO [9] rely on accurate trajectory modeling to assess strategic and tactical deconfliction mechanisms.

Another critical application is within the U-plan Preparation Service (UPPS), defined in the U-space CONOPS by CORUS [10]. UPPS is envisioned as a tool for UAS operators to develop flight plans and submit them for authorization, potentially as a service provided by USSPs. While the internal workings of UPPS are outside the scope of CORUS, its effectiveness hinges on the ability to generate realistic flight plans that reflect real-world operational constraints.

Realistic trajectory datasets are not limited to TIS, FAS, or UPPS but have broader applications across all U-space services, enhancing airspace management, risk assessment, and operational validation. Fig. 1 illustrates the U-space service catalog defined by EASA, outlining the ecosystem in which trajectory modeling improves automation, scalability, and safety.

Notice in Fig. 1 that services are optional or compulsory from the point of view of the authority designating the U-space airspace. When that authority considers that any optional services are needed for safety, that service becomes mandatory for USSPs and UAS operators.

III. Methodology

A. Flight Plan Generation for Simulation

To ensure a comprehensive evaluation of UAS trajectory execution, four distinct types of flight plans have been designed, each representing a different operational scenario. These plans serve as input for trajectory simulations and are designed to reflect realistic drone missions. Fig. 2 illustrates the various types of flight plans considered in this study.

The flight plans are generated within a 50 km² area centered on Valencia, allowing for the simulation of various operational scenarios in an urban environment. The developed tool enables customization of these geographic boundaries and other parameters, such as waypoint distance and heading constraints. A seed number is also used to initialize the random number generator in MATLAB to ensure result reproducibility, guaranteeing consistent outcomes for the same input parameters.

All flight plans are executed at a constant altitude, randomly selected between 25 m and 75 m, ensuring a realistic vertical profile for different operational scenarios. Additionally, maximum speeds for each drone are randomly assigned, ranging from 4 m/s to 12 m/s, simulating various types of drones

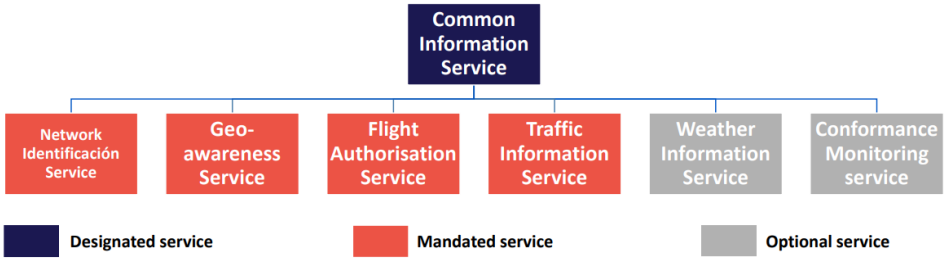


Fig. 1. U-space service catalog as defined by EASA [11].

with varying capabilities. These diverse mission types enable a thorough evaluation of UAS trajectory prediction models, accounting for structured and unstructured drone operations.

1) Random Waypoint Flight Plan: In this configuration, the drone takes off from a randomly assigned position and follows a sequence of randomly generated waypoints. To maintain realistic motion constraints, each waypoint is generated within a predefined distance range of 400 m to 800 m from the previous waypoint, with a maximum allowable heading change of 135°. This scenario represents unstructured drone flights, such as recreational operations or exploratory missions.

2) Fixed Waypoint Flight Plan: This flight plan models structured routes between predefined locations. The drone takes off from a randomly selected fixed point within a list of known waypoints, which in this case correspond to hospitals in the city of Valencia. It then flies directly to another randomly selected fixed waypoint and lands. This scenario simulates drone operations in medical logistics, such as transporting urgent medical supplies between healthcare facilities.

3) Delivery Flight Plan: Designed to simulate logistics operations, the delivery flight plan involves a two-segment trajectory. The drone takes off from a fixed waypoint corresponding to a logistics center, flies directly to a randomly selected delivery point, lands for five seconds to simulate a package drop-off, then takes off again and returns to its starting location. This flight plan emulates real-world parcel delivery services using UAS.

4) Scan Flight Plan: This trajectory is designed for aerial surveillance and mapping operations. The drone is assigned a randomly selected area to survey and follows a structured scanning pattern over the region. This type of plan is commonly used in applications such as environmental monitoring, search-and-rescue missions, and infrastructure inspections.

B. Trajectory Processing in PX4 SITL

Once a flight plan is defined, it is converted into a trajectory through a detailed simulation using PX4 SITL. This simulation environment enables high-fidelity trajectory generation by replicating real-world flight conditions, allowing the introduction of external perturbations such as wind variations. The SITL framework facilitates seamless integration with external software, enabling automated flight execution and telemetry recording.

Fig. 3 illustrates the structure of the PX4 SITL simulation environment. The system exchanges MAVLink messages between the simulated drone and PX4, processing sensor data while returning actuator commands. Communication is established via UDP, allowing PX4 to interact with the simulator and external developer APIs such as MAVSDK or ROS. The standard MAVLink module also enables connections to ground control stations, like QGroundControl.

A dedicated software tool has been developed to automate the simulation process using PX4 SITL. This tool, implemented in Python with the MAVSDK library, executes the SITL environment in the background while sequentially loading flight plans stored in a database. Each plan, formatted in QGroundControl's '.plan' structure, is autonomously uploaded to the simulated drone, executing the predefined waypoints. Telemetry data is continuously recorded throughout the flight, stored in a structured CSV format, and subsequently uploaded to the corresponding database for further analysis.

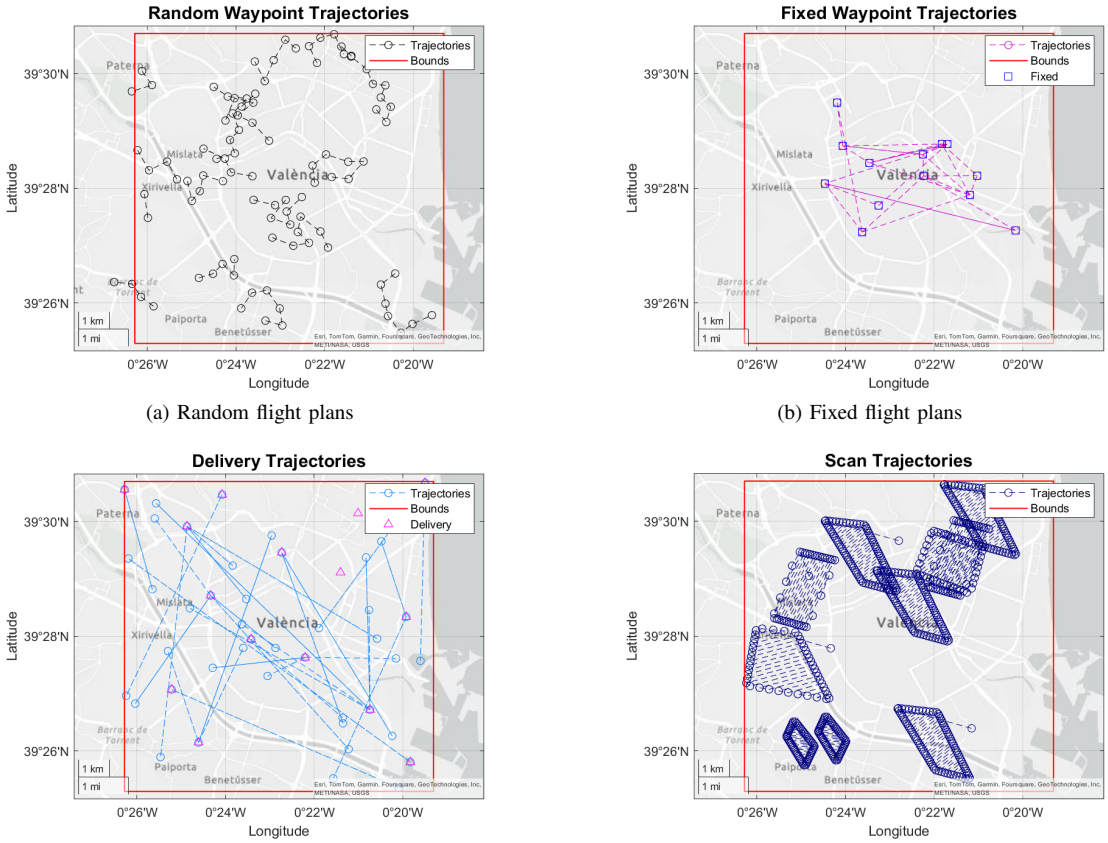
By leveraging this automated pipeline, large-scale trajectory datasets can be efficiently generated, ensuring consistency in flight execution and enabling robust validation of trajectory prediction models.

C. Parallelized Simulation Framework with Docker

Generating extensive datasets of realistic drone trajectories requires significant computational resources. A major limitation encountered during development was the inability to run multiple PX4 SITL instances on the same system due to fixed port allocations within the PX4 source code. Since these ports could not be dynamically reassigned, direct parallel execution was not feasible.

A containerized approach using Docker [13] was implemented to overcome this limitation, allowing multiple independent SITL instances to run in parallel. Each Docker container hosts a fully isolated PX4 SITL environment, ensuring numerous simulations can be executed simultaneously without conflicts. This enables efficient large-scale trajectory processing while maintaining system stability.

Fig. 4 illustrates the final software architecture used for trajectory analysis. Each Docker container is directly connected to the central database and registered as an available simulation instance. A task allocator continuously monitors available containers and assigns flight plans to be processed based on a priority-based scheduling algorithm. Once a container



(a) Random flight plans

(b) Fixed flight plans

(c) Delivery flight plans

(d) Scan flight plans

Fig. 2. Different types of flight plans used in simulations.

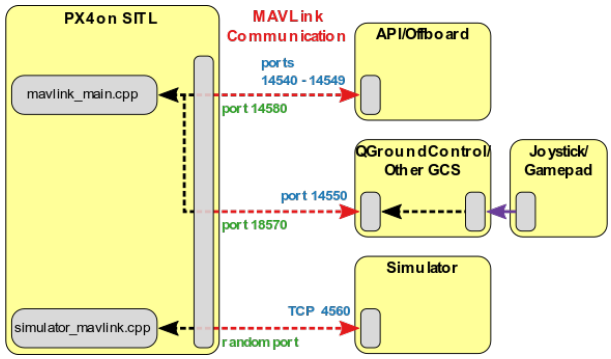


Fig. 3. PX4 SITL simulation architecture [12].

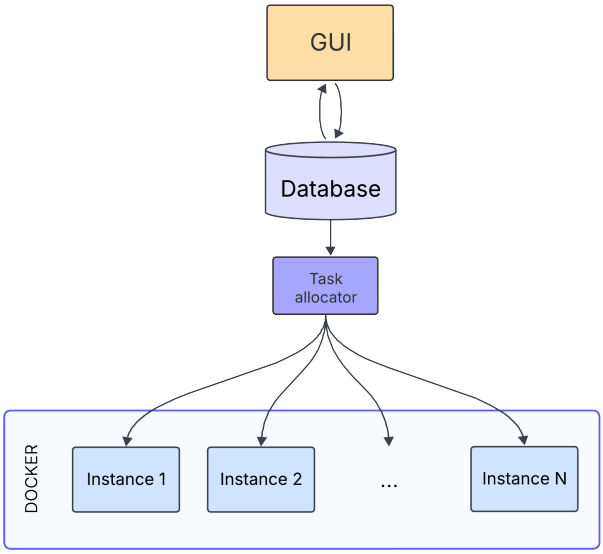


Fig. 4. System architecture for parallelized trajectory processing.

completes a simulation, it uploads the generated trajectory data back to the database and becomes available for new assignments.

A graphical user interface (GUI) has been developed to facilitate interaction with the system. The GUI allows users to:

- Upload and manage flight plans.
- Monitor and control active simulations.
- Download and analyze the results of executed trajectories.

This framework ensures efficient workload distribution and maximizes computational throughput, enabling processing thousands of flight plans in a reduced time frame. By leveraging Docker-based parallelization, the system effectively scales to meet the demands of realistic trajectory dataset

generation while maintaining accuracy and consistency in trajectory execution.

IV. Evaluation and Results

A. Computational Infrastructure and Parallelization Study

The simulation environment comprises a high-performance computing system configured to support large-scale trajectory simulations. The specifications of the system are as follows:

- Processor: AMD Ryzen Threadripper 7970X (32 cores, 64 threads)
- Graphics Card: NVIDIA RTX 4070 Ti Super
- Memory: 64 GB RAM
- Operating System: Windows Server

To determine the optimal parallelization strategy for trajectory processing, an experiment was conducted in which 200 identical flight plans were processed using varying numbers of Docker containers running PX4 SITL instances. The average execution time per trajectory (y-axis) was measured for an increasing number of parallel instances (x-axis), with results presented in Fig. 5.

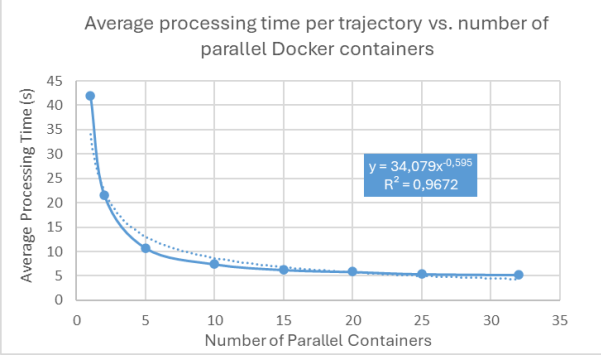


Fig. 5. Average processing time per trajectory vs. number of parallel Docker containers.

For clarity, the figure only displays results up to 32 parallel instances, as beyond this point, the system experiences a substantial degradation in performance. The relationship between execution time and the number of containers follows a power-law trend, which can be approximated using a potential function with an $R^2 = 0.9672$, indicating a firm fit.

Beyond 32 parallel instances, increasing the number of containers leads to a rapid increase in execution time. With 42 instances, the overhead introduced by excessive context switching and competition for shared resources starts to degrade performance significantly. At 64 instances, execution times grow exponentially, with frequent process failures likely caused by memory saturation and the inability of each instance to allocate sufficient threads for stable execution.

It is important to note that these results are specific to the hardware used in this study. With a more robust system, different optimal values could be obtained. However, the observed trend suggests that the optimal number of parallel processes is approximately half the number of available processing threads.

B. Baseline Processing Time Analysis

After determining the optimal parallelization strategy, five datasets were generated using different random seeds while maintaining identical conditions. Each dataset consisted of 500 flight plans, with a distribution of 20% random trajectories, 35% fixed waypoint trajectories, 35% delivery trajectories, and 10% scan trajectories. The primary goal of this analysis was to evaluate the computational performance of the system under standard operating conditions, precisely measuring the average processing time per trajectory, assessing overall execution time and system efficiency, and analyzing the relationship between simulated flight time and execution time.

Tab. I presents the mean processing time per trajectory type and its standard deviation. The significant variability observed, particularly in scan trajectories, is expected due to the random nature of flight plan generation, which leads to differences in waypoint distribution, average flight speed, and the number of trajectory corrections performed by the autopilot.

TABLE I
Average Processing Time per Flight Plan

Trajectory Type	Avg. Time (s)
Random	5.51 ± 0.25
Fixed	6.02 ± 0.60
Delivery	16.88 ± 1.41
Scan	47.07 ± 5.01

The total execution time required to process 500 flight plans was measured as $T_{\text{total}} = 6866.00 \pm 532.01$ s, where the deviation accounts for variations across different trials. In addition, the system utilization efficiency, defined as the fraction of total machine uptime spent actively processing trajectories, was calculated as $\eta = 75.68 \pm 5.79\%$; this value indicates there is room for optimization, as some instances fail during execution and remain inactive for the rest of the experiment, reducing overall efficiency. Additionally, resource consumption is not fully optimized when a small number of remaining trajectories are assigned to a limited subset of available instances. Future improvements, such as enhanced instance management and automated fault recovery mechanisms, could help increase utilization and further optimize processing efficiency.

To obtain a more consistent performance metric, an additional study was conducted to analyze the relationship between simulated flight time and execution time. This approach provides a more reliable measure of computational cost, as it accounts for differences in trajectory length and complexity. To better quantify the efficiency of trajectory processing relative to the simulated flight time, we define the Simulation Efficiency Ratio as the measured relationship between simulated flight hours and execution time for different trajectory types. This metric provides a normalized measure of how efficiently the system processes various trajectories. Tab. II presents the measured SER values for each trajectory type.

Given the observed relationship between execution time and simulated hours, it is possible to extrapolate the estimated processing time required to generate datasets of larger scales.

TABLE II
SIMULATION EFFICIENCY RATIO FOR DIFFERENT TRAJECTORY TYPES

Trajectory Type	Simulation Efficiency Ratio
Random	94.71 ± 6.74
Fixed	97.12 ± 7.62
Delivery	102.93 ± 8.93
Scan	109.75 ± 12.34
Mean	$\mathbf{103.61 \pm 8.59}$

Tab. III presents projected processing times for datasets containing 500, 1,000, 5,000, and 10,000 flight hours, computed using:

$$T_{\text{estimated}} = \frac{\text{Simulated Flight Time}(h)}{\text{SER}} \tag{1}$$

TABLE III
PROJECTED PROCESSING TIME FOR LARGE-SCALE DATASETS

Simulated Flight Time (h)	Estimated Processing Time (h)
500	4.83 ± 0.40
1,000	9.64 ± 0.80
5,000	48.19 ± 4.01
10,000	96.38 ± 8.02

These projections demonstrate that, while the current system can process moderate-scale datasets efficiently, generating datasets at the scale of thousands of flight hours remains computationally demanding. However, the architecture can be scaled by distributing Docker containers across multiple machines, significantly improving processing capabilities and reducing overall execution time.

C. Impact of Wind Conditions on Processing Time

To analyze the effect of external conditions on simulation performance, the same five datasets were processed under wind conditions using the Gazebo wind plugin. The following wind parameters were configured:

- Mean wind speed: 4 m/s
- Maximum wind speed: 20 m/s
- Wind direction: Consistent along the Y-axis

Processing times were then compared against the baseline simulations (without wind) to assess whether environmental perturbations significantly impact computational efficiency. Tab. IV presents the average processing time per flight plan and the simulation efficiency ratio (SER) for both baseline and wind-affected simulations.

TABLE IV
COMPARISON OF PROCESSING TIME AND SIMULATION EFFICIENCY RATIO UNDER WIND CONDITIONS

Trajectory Type	Avg. Time (s)	SER
Random	5.11 ± 0.62	102.24 ± 9.95
Fixed	5.73 ± 1.12	103.95 ± 16.23
Delivery	15.53 ± 1.25	114.43 ± 10.41
Scan	40.12 ± 4.23	127.68 ± 11.12
Total	6164.80 ± 634.77	115.93 ± 11.68

The total execution time required to process all flight plans was $T_{\text{total}} = 6164.80 \pm 634.77$ s. Additionally, the system utilization efficiency, defined as the fraction of total machine uptime spent actively processing trajectories, was calculated as $\eta = 83.82 \pm 6.16\%$. This indicates fewer parallel processing instances crashed compared to the baseline scenario, leading to more efficient execution and reduced total processing time.

To further investigate computational efficiency, we introduce a normalized execution time metric, which indicates the execution time with a 100% efficiency, defined as:

$$T_{\text{total, norm}} = T_{\text{total}} \times \eta \tag{2}$$

Tab. V presents the normalized execution times for nominal and wind-affected cases, demonstrating that they fall within the same range and confirming that computational cost remains independent of environmental conditions.

TABLE V
NORMALIZED EXECUTION TIME COMPARISON

Scenario	Normalized Total Time (s)
Nominal	5196.19 ± 402.63
With Wind	5167.34 ± 532.06

The results confirm that introducing wind conditions does not significantly impact processing times. Since the normalized execution times remain consistent across conditions, we conclude that computational cost is primarily influenced by system utilization efficiency rather than environmental perturbations introduced within the simulation.

To further refine this analysis, we compute the optimized Simulation Efficiency Ratio (SER), which represents the processing-to-simulated time ratio under ideal 100% efficiency conditions. By aggregating data from all nominal and wind-affected simulations, we obtain an optimized SER of 137.50 ± 4.26. This value provides a more stable reference for estimating execution times for large-scale dataset generation, demonstrating the robustness of the proposed simulation framework across different environmental conditions.

Achieving this efficiency would require further refinement of the developed software, which is proposed as future work. Enhancements in instance management and fault recovery mechanisms could contribute to maximizing system utilization and further optimizing the computational framework.

D. Trajectory Deviation Analysis: Nominal vs. Windy Conditions

This section evaluates the deviation between two 4D trajectories generated from the same flight plan under different environmental conditions:

- PX4 SITL nominal case (no wind)
- PX4 SITL windy case (wind conditions as defined in Section X)

The objective is to quantify how wind influences trajectory execution and identify systematic deviations throughout flight phases.

Fig. 6 illustrates the total deviation (in meters) between the two trajectories over time. The deviation exhibits a general increasing trend during cruise flight, with periodic dips aligning with waypoints. This behavior is expected, as waypoints act as reference corrections for the autopilot, minimizing lateral drift even under wind influence. However, minor temporal offsets prevent the error from reaching zero at these points.

The most significant trajectory deviation occurs during the landing phase, reaching a peak of 2.75 meters just before touchdown. This effect is attributed to the gradual descent under wind influence, causing a lateral shift in the final approach. Additionally, the takeoff phase presents the steepest initial deviation as the wind slows the initial ascent, leading to a more gradual climb than the nominal case.

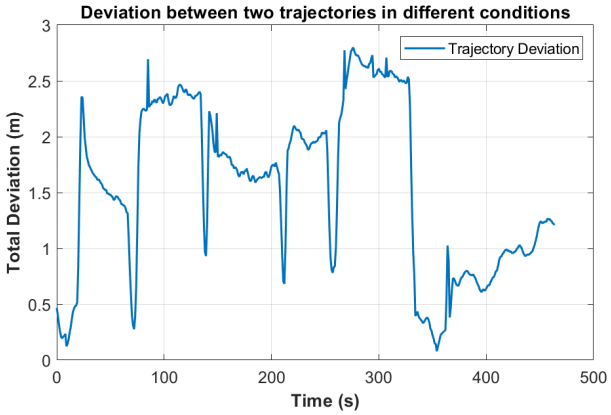


Fig. 6. Total deviation in meters between PX4 SITL trajectories with and without wind conditions.

Initially, this study aimed to include a comparison with BB-Planner, the trajectory generation tool used in the BUBBLES project [1]. However, such a comparison is not meaningful due to fundamental differences in trajectory computation methodologies. BB-Planner generates motion using a constant velocity model (MRU) with an added random deviation, whereas PX4 SITL employs a high-fidelity flight dynamics model, accounting for aerodynamic effects, wind interaction, and autopilot corrections. Consequently, deviations between PX4 and BB-Planner trajectories would be significantly more significant than those observed between nominal and windy PX4 cases, rendering direct comparisons impractical for regulatory validation.

This analysis confirms that wind conditions introduce systematic deviations throughout all flight phases but remain within predictable limits. This suggests realistic SITL-based trajectories could enhance 4D airspace planning accuracy by incorporating environmental effects. Future studies will explore how different wind profiles impact trajectory predictability and separation requirements.

V. Conclusions

A. Summary of Key Findings

This study has demonstrated the feasibility of generating realistic UAS trajectories using PX4 SITL simulations under various environmental conditions. The results validate the proposed methodology, highlighting its capability to model trajectory deviations due to wind disturbances and other operational factors. Additionally, the parallelized processing approach via Docker significantly reduces computation time, enabling large-scale dataset generation for further analysis and application.

With the developed software, the simulation speed has increased from a baseline ratio of up to $\times 10$ to a ratio of approximately $\times 100$ with the same computing system, achieving a +1000% improvement in processing efficiency using the same computational resources. Moreover, the software architecture supports distributed execution across multiple machines, allowing further scalability by connecting additional simulation nodes. This ensures that dataset generation can be adjusted dynamically based on computational availability, significantly reducing processing time when scaling the infrastructure. With an efficiency of 100%, the system could achieve simulation speeds of up to approximately $\times 140$ real-time. There is room for further optimization and crash minimization to maximize system performance.

Furthermore, this work has enabled the precise estimation of the required execution time for generating datasets with different trajectory distributions. By adjusting parameters such as the percentage of each trajectory type within a dataset, researchers and developers can now predict processing times with high accuracy, facilitating the design and deployment of large-scale simulations. Additionally, the analysis of wind influence has confirmed that introducing environmental perturbations does not significantly impact simulation performance. This finding suggests that the proposed methodology remains computationally efficient across various simulated environments, reinforcing its applicability for diverse U-space scenarios.

A key advantage of this approach is the seamless transition from simulation to real-world execution. Since flight plans generated within PX4 SITL follow the same control laws and navigation models as those executed by physical drones, these plans can be directly loaded onto real aircraft with minimal modification. This ensures that the simulated trajectories highly represent real-world flight behavior, significantly reducing the uncertainty in trajectory prediction. As a result, despite the increased computational cost associated with autopilot-based simulations, the reduction in simulation-to-reality deviation makes this methodology an ideal solution for safety-critical applications, such as strategic deconfliction and airspace management in U-space.

B. Future Work and Integration

Future research will focus on refining the simulation framework by incorporating additional environmental variables and further optimizing computational efficiency. The ability to load and execute flight plans within PX4 SITL ensures that the generated trajectories closely resemble real-world execution. This means simulated trajectories could be directly loaded onto physical drones, enabling real-world validation with minimal

deviation from their simulated counterparts. Such precision could contribute to reducing the volume of reserved 4D airspace for UAS operations, improving airspace utilization while maintaining safety.

Ongoing efforts are also directed toward integrating this simulation methodology into operational U-space services. In the U-AGREE project, the generated datasets are being used to support flight plan approval systems, where initial trajectories are evaluated for potential conflicts and adjusted to ensure regulatory compliance. Additionally, in CREATE, this work is being incorporated into the development of an integrated U-plan Preparation Service (UPPS) designed to provide users with an interface for creating, registering, simulating, and submitting flight plans for approval. This integration will allow operators to validate their missions in a realistic environment before deployment, enhancing strategic deconfliction capabilities within U-space.

By continuing to develop and integrate these trajectory generation tools, this research aims to contribute to advancing scalable and efficient UAS traffic management systems, bridging the gap between simulation and real-world implementation.

ACKNOWLEDGMENT

This research was conducted within the framework of the CREATE U-space project. CREATE U-space project (CIAICO/2022/044) has received funding from the Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital of the Generalitat Valenciana.

REFERENCES

[1] BUBBLES Project. (2020-2022) BUBBLES: A framework for U-space conflict management. [Online]. Available: https://www.sesarju.eu/projects/bubbles

[2] U-AGREE Project. (2022-2027) Integrated Risk Model for UAS Operations. [Online]. Available: https://www.sesarju.eu/projects/U-AGREE

[3] G. Anitha, E. Saravanan, J. murugan, and S. K. Nagothu, "Software in the loop (sitl) simulation of aircraft navigation, guidance, and control using waypoints," in *Modeling, Simulation and Optimization*, B. Das, R. Patgiri, S. Bandyopadhyay, V. E. Balas, and S. Roy, Eds. Singapore: Springer Nature Singapore, 2024, pp. 523–543.

[4] ArduPilot Development Team, "Mission Planner for ArduPilot," 2024. [Online]. Available: https://ardupilot.org/planner

[5] H. Qays, B. Jumaa, and A. Salman, "Design and implementation of autonomous quadcopter using sitl simulator," *Iraqi Journal of Computer, Communication, Control and System Engineering*, 05 2020.

[6] D. A. Maigualema-Quimbita, J. V. Balbastre Tejedor, D. Gomez-Barquero, and V. Monzonis Melero, "Positioning of unmanned aerial vehicles (UAVs) in urban environments using 5G networks: A hybrid approach based on multilateration and machine learning," in *2025 Integrated Communications, Navigation and Surveillance Conference (ICNS) (ICNS 2025)*, Brussels, Belgium, Apr. 2025, p. 9.

[7] S. Amarillo, A. Sanchis, B. Fraiz Freire, and J. V. Balbastre Tejedor, "Proposal of UAS strategic conflict detection concept with a centralised service in multi-USSP environment using an octree data structure," in *2025 Integrated Communications, Navigation and Surveillance Conference (ICNS) (ICNS 2025)*, Brussels, Belgium, Apr. 2025, p. 8.

[8] A. Garcia, S. Amarillo, S. Esparza, and J. V. Balbastre, "Proposal of uas strategic conflict detection concept with a centralised sevice in multi-ussp environment using an octree data structure," in *Proceedings of SESAR Innovation Days*, 2024.

[9] SPATIO Project. (2023-2026) U-space Separation Management. [Online]. Available: https://www.sesarju.eu/projects/SPATIO

[10] CORUS Project. (2017-2019) Concept of Operations for U-space. [Online]. Available: https://www.sesarju.eu/projects/CORUS

[11] European Commission. (2021) Commission Implementing Regulation (EU) 2021/664 of 22 April 2021 on a regulatory framework for the u-space. [Online]. Available: https://eur-lex.europa.eu/eli/reg_impl/2021/664/oj

[12] PX4 Development Team. (2024) Simulation — PX4 Guide (main). [Online]. Available: https://docs.px4.io/main/en/simulation

[13] Docker Inc. (2024) Docker: Accelerate How You Build, Share, and Run Modern Applications. [Online]. Available: https://www.docker.com/why-docker