Article

Not peer-reviewed version

U-Plan: An Integrated Framework for the Coordination and Real-Time Supervision of Heterogeneous Unmanned Aerial Systems

Ehsan Kouchaki *, Miguel Ángel de Frutos Carro, José Ramiro Martínez-de Dios, Anibal Ollero

Posted Date: 14 May 2026

doi: 10.20944/preprints202605.0937.v1

Keywords: multi-UAS mission planning; multi-UAS coordination frameworks; applications

 doi:10.20944/preprints202605.0937.v1

Article

U-Plan: An Integrated Framework for the Coordination and Real-Time Supervision of Heterogeneous Unmanned Aerial Systems

Ehsan Kouchaki [1,*], Miguel Ángel de Frutos Carro [2], José Ramiro Martínez-de Dios [1] and Aníbal Ollero [1]

[1] GRVC Robotics Laboratory, University of Seville, 41092 Seville, Spain
[2] Centro de Automática y Robótica (UPM-CSIC), Universidad Politécnica de Madrid-Consejo Superior de Investigaciones Científicas, 28006 Madrid, Spain
* Correspondence: ekouchaki@us.es; Tel.: +34-672-92-2801

Highlights

What are the main findings?

- Design, development, and validation of U-Plan, an integrated framework for the coordination, planning, real-time monitoring, and replanning of multiple heterogeneous Unmanned Aerial Systems (UAS) capable of coping with real-world operational conditions, including UAS kinematic constraints, wind effects, and airspace constraints, and dynamic changes in UAS, mission, and environment conditions.

What is the implication of the main finding?

- Validation of U-Plan and its components integrated with professional-grade Visionair Ground Control Station (GCS) software and VECTOR-SIL Software-in-the-Loop autopilot simulator, and comparison with main existing methods show significant efficacy, efficiency, scalability, and real-time adaptability to dynamic changes.

Abstract

Despite the large amount of successful existing methods and frameworks for planning sets of multiple Unmanned Aerial Systems (UAS), there are still lack of coordination frameworks capable of coping with real-world operational conditions. This paper presents U-Plan, an integrated management framework for the coordination of multi-UAS missions. U-Plan is designed to plan, schedule, monitor, and replan a heterogeneous set of UAS to complete Points of Interest (PoIs) visiting missions while ensuring that all generated trajectories are safe, feasible, and compliant with the required PoIs' arrival times, UAS kinematics and energetic constraints, and the existing 3D No-FLy Zones (NFZs). U-Plan is designed as a practical tool for strongly dynamic missions, and is built upon three core components: 1) an NFZ-aware route computation method that explicitly accounts for NFZs prior to the Vehicle Routing Problem (VRP) optimization, resulting in shorter NFZ-safe routes; 2) a trajectory planning module that ensures the generation of kinematically-feasible trajectories for fixed-wing UAS; and 3) a mission supervision module for real-time monitoring and replanning in case of changes in UAS, mission, wind speed, or airspace restrictions. It was implemented and validated by interfacing with professional-grade Visionair Ground Control Station Software and the VECTOR-SIL Software-in-the-Loop simulator, which realistically replicates the behavior of certified fixed-wing autopilots under various weather conditions. The validation shows U-Plan's capacity to efficiently satisfy complex mission requirements with strong scalability. Due to its high computational efficiency, U-Plan enables online mission replanning, allowing UAS fleets to seamlessly adapt to changes typical of real-world operational scenarios.

Keywords: multi-UAS mission planning; multi-UAS coordination frameworks; applications

 doi:10.20944/preprints202605.0937.v1

1. Introduction

The use of Unmanned Aerial Systems (UAS) has become established as an effective solution for tasks such as inspection [1–3], monitoring [4–6], and surveillance [7,8] in complex, dynamic, and unstructured environments. However, many applications require covering multiple objectives or tasks under time, energy, and safety constraints. In this context, the use of multi-UAS systems increases efficiency by reducing total mission time, providing redundancy against individual failures, and exploiting the synergies resulting from the cooperation of UAS with heterogeneous capabilities. Despite these advantages, the operation of multi-UAS systems introduces additional challenges that go beyond individual platform control. The coordination of multi-UAS systems requires integrating efficient task allocation and trajectory planning into a common framework capable of efficiently generating flight plans for each UAS that satisfy its particular kinematic constraints, energy constraints, and communication range.

We are interested in a multi-UAS mission management framework for real-world missions, capable of efficiently considering the most relevant practical effects present in every-day multi-UAS management systems, such as wind conditions, airspace constraints caused by obstacles and No-Fly Zones (NFZs), and UAS kinematic constraints and energy consumption effects. The framework should have the possibility of managing heterogeneous UAS with different kinematic models, energy consumption models, and initial states, and should have the capability to efficiently adapt to the dynamic changes that often appear during mission execution, such as failures in UAS, dynamic updates in the mission definition, or changes in the environment conditions or airspace restrictions, among others.

A wide variety of successful methods and frameworks have been developed in multi-UAS coordination. Many of them focus on mathematical optimality and often make simplifying assumptions that disregard practical effects such as UAS kinematic constraints, wind effects, energy considerations, and airspace restrictions [9–12]. Other methods address some of these effects but involve a very high computational cost that severely constrains multi-UAS replanning, also resulting in poor scalability with large swarms sizes or complex missions [13–15]. Other frameworks focus solely on the initial multi-UAS planning and do not consider the dynamic adaptation of the flight plans due to the lack of autonomous mission monitoring and replanning functionalities or due to computational constraints [16,17]. Despite the large amount of successful existing methods and frameworks, there are lack of effective and efficient multi-UAS coordination systems capable of coping with real-world conditions.

This work proposes U-Plan, a multi-UAS management framework for the coordination, motion planning, and monitoring of multi-UAS missions. U-Plan is designed to plan, schedule, monitor, and replan a heterogeneous set of UAS to complete a mission consisting in minimizing the time performed in visiting a set of Points of Interest (PoIs) while ensuring that all generated trajectories are safe, feasible, and compliant with the required PoI's arrival times, UAS kinematics and energetic constraints, and the existing 3D NFZs. U-Plan is designed as an efficient and practical tool for multi-UAS coordination in real-world missions. It is structured in three main components. **Muti-UAS Route Computation** solves a Vehicle Routing Problem (VRP) to determine the route that is assigned to each UAS, considering NFZs, wind effects, and UAS energy consumption, models. The VRP is solved by Google's OR-Tools library, selected for its capability to generate near-optimal solutions with high computational efficiency and scalability –a critical requirement for enabling dynamic, real-time replanning. Traditional approaches treat NFZ avoidance as a post-processing step after route generation, which often leads to unnecessarily long UAS routes. Instead, our approach integrates NFZ avoidance prior to the VRP optimization by calculating the cost matrix of the VRP, explicitly accounting for NFZs, hence obtaining significantly shorter routes. Furthermore, it explicitly incorporates wind-aware constraints to ensure the resulting routes remain energetically feasible under real-world aerodynamic conditions. The second component, **Trajectory Smoothing** transforms the routes assigned to each UAS into kinematically feasible trajectories using a geometry-based smoothing method that considers the minimum turning radius of that UAS, making a kinematically feasible maneuver, while reducing the deviation w.r.t.

 doi:10.20944/preprints202605.0937.v1

the original path and maintaining NFZ compliance. Finally, **Mission Supervision** validates the UAS flight plans against the mission requirements, monitors the mission execution, and predicts mission fulfillment, triggering mission replanning if necessary. The three components are integrated in a seamless closed-loop structure under the supervision of the human operator.

U-Plan was implemented, interfacing with Visionair GCS Software, a commercial Ground Control Station for management of UAS and fleets, and using VECTOR-SIL, a Software-in-the-Loop simulator that realistically replicates the behavior of fixed-wing UAS autopilots under all weather conditions. With this set-up, the full U-Plan, and also its components, have been extensively validated and compared with the main existing approaches, showing its efficiency, scalability, and adaptability to changes. As required, U-Plan requires very low mission planning computation times, which enables its use for dynamic mission replanning even with large UAS fleet sizes and high numbers of PoIs, being capable to adapt to the frequent changing conditions that can be found in many complex multi-UAS scenarios.

The main contributions of the work are:

- U-Plan, a multi-UAS management framework for the coordination, planning, monitoring, and replanning of complex and dynamic multi-UAS missions;
- Multi-UAS management methods: (i) NFZ-aware route allocation that explicitly accounts for NFZs prior to the VRP optimization, resulting in shorter NFZ-safe routes; (ii) trajectory planning that ensures short kinematically-feasible trajectories; and (iii) mission supervision and monitoring;
- Validation of U-Plan (and its components) in realistic multi-UAS missions, interfacing with commercial products VECTOR-SIL and Visionair GCS Software.

The reminder of this work is as follows. The main related work is summarized in Section 2. The formulation of the addressed problem and design of the proposed U-Plan framework is described in Section 3. The methods used to implement U-Plan components and methods are presented in Section 4. Section 5 extensively validates both the proposed methods individually as the full U-Plan proposed. Finally, Section 6 summarizes the conclusions and main future work.

2. Related Work

A wide variety of good multi-UAS path planning methods have been proposed along the years. Only the main related research directly related to the proposed methods and framework are summarized.

A high number of methods can solve the full multi-UAS path planning problem and can generate a safe, kinematically feasible path for each UAS to fulfill the mission, see e.g., [9–11]. However, most of these methods assume simplifications in the UAS and airspace constraints, or disregard critical effects such as wind or energy consumption, which often constrain their practical feasibility. Integrating in these methods complex, real-world conditions, such as non-linear UAS energy consumption and dynamic wind fields, would result in too high computational costs for practical feasibility, also impacting scalability. To overcome these computational bottlenecks while ensuring flexibility to cope with realistic problems, U-Plan adopts a decoupled approach: 1) first, generating optimal piecewise-linear routes for each UAS considering NFZs and wind effects, and 2) second, generating a smoothed trajectory from each UAS piece-wise route, ensuring kinematic feasibility while reducing the deviations of the trajectory w.r.t. the original route. This decoupled approach strongly reduces the computational cost and adds modularity and flexibility to the proposed multi-UAS framework.

Multi-UAS route generation has been often addressed as a Vehicle Routing Problem (VRP) [18,19]. Many methods solve VRP using exact algorithms predominantly based on Mixed-Integer Linear Programming (MILP) using optimization tools such as Gurobi [20,21] or CPLEX [5,7], among others. Although they guarantee optimal routes, exact algorithms have high computational cost and have poor scalability with the problem size. On the other hand, swarm intelligence-based methods including Genetic Algorithms (GA) [3,8,15], Simulated Annealing (SA) [22,23], or Ant Colony Optimization (ACO) [24,25] are more computationally efficient, and can find near-optimal solutions in significantly

 doi:10.20944/preprints202605.0937.v1

lower times than exact methods. However, as the mission scale increases, these metaheuristics often face scalability challenges, exhibiting higher variance in solution quality and a sharp increase in computation times that can hinder real-time reactivity in mission replanning. Selecting a suitable and computationally efficient solver may be not straightforward in many complex problems. For these cases, several software suites such as PicatSAT [26], Choco-solver [27], SICStus Prolog [28], and OR-Tools [29], have been developed to encompass solvers for different types of optimization problems. Google OR-Tools, an open-source suite that provides very efficient solvers for linear programming, mixed-integer programming, constraint programming, and routing optimization problems, has attracted significant interest. OR-Tools has obtained excellent results in the MiniZinc Challenge[1] (an annual competition of constraint programming solvers), being the winner in 30 of the 37 challenges proposed since 2018. Although OR-Tools has been proposed with good results for UAS and multi-UAS routing, see e.g., [14,17], it has not been integrated nor validated in full multi-UAS coordination systems.

Most existing multi-UAS route generation methods treat obstacle and NFZ avoidance as a post-processing detour step after UAS routing: they generate an initial route first without considering NFZs and then, modify colliding segments through geometric refinements [13,30]. While post-processing simplifies the routing task, it frequently results in significantly longer routes because the initial optimization is solved based on inaccurate Euclidean distances. To solve this, U-Plan makes use of a NFZ-aware UAS routing mechanism that adapts the VRP cost matrix such that NFZ avoidance is performed naturally during the solving of the VRP, resulting in significantly shorter trajectories with a negligible additional computational cost. Similarly, few multi-UAS routing systems consider the effects of wind, which has strong impact on both UAS flight times and fly energy consumption. Neglecting these wind effects can lead to mission failure or premature energy exhaustion. In fact, recent research emphasizes the need for wind-aware routing to ensure that trajectories remain energetically feasible even in adverse weather conditions [31–33]. To cope with them, U-Plan incorporates wind effects constraints in the VRP. As a result, U-Plan naturally includes NFZ avoidance and wind effects into the multi-UAS routing components using OR-Tools, hence efficiently solving two critical practical aspects in multi-UAS routing with negligible additional computational cost.

Once the multi-UAS routes (piece-wise) have been generated, we need smoothing methods that: 1) ensure kinematic feasibility of the resulting trajectories, and 2) reduce the deviation of the trajectories w.r.t. to the original piece-wise routes, to help preserve the obstacle and NFZ compliance guaranteed for the routes by the prior VRP. Different route smoothing methods have been proposed, such as B-Splines [34], Dubins [12,35], or "Overfly" maneuvers (used in commercial autopilots [36]). However, they fulfill our requirements. B-Splines cannot inherently guarantee compliance with the UAS's turning radius constraints. Dubins curves results in non-uniform spatial deviations w.r.t. the piece-wise route that become severely skewed during sharp turns. "Overfly" maneuvers can induce strong overshoots and become geometrically infeasible for wide cornering angles. For solving those problems, in U-Plan we adopt a geometry-based smoothing method that considers the minimum turning radius of each UAS, making a kinematically feasible maneuver, while reducing the deviation w.r.t. the original path, helping preserve NFZ compliance.

Beyond initial planning, real-world missions are inherently dynamic and subject to disruptions such as hardware failures, changes in the mission (e.g., new PoIs to be visited), changes in environment conditions (e.g., wind), or changes in airspace restrictions (e.g., changes in existing NFZs or declaration of new ones). Coping with them requires supervision architectures capable of continuous monitoring and efficiently replanning the UAS fleet to adapt to changes in real-time. Several architectures have been proposed. Work [12] proposes a real-time collaborative planning method by coupling task assignment with Dubins path distances. It incorporates a dynamic replanning mechanism for new tasks and UAS failures through specific emergency strategies. However, airspace constraints and wind fields are not considered, which constraints its applicability in real environments. Work [17]

[1] https://www.minizinc.org/challenge/

 doi:10.20944/preprints202605.0937.v1

introduces a multi-UAS coordination framework for emergency medical logistics that integrates a digital-twin human-machine interface for mission monitoring through visualization of the telemetry and state of the different UAS. However, this framework does not consider the autonomous detection of missions deviations and autonomous fleet replanning, leaving it for future research. Work [16] developed a real-time network approach to generate kinematically feasible obstacle-free paths while handling pop-up obstacles for dynamic reassignment. However, it is limited to the routing of a single UAS, and assumes an idealized operational state with constant altitude and velocity.

3. General Description

The objective is to develop a multi-UAS mission management framework for the robust motion planning and real-time monitoring of cooperative heterogeneous multi-UAS missions. U-Plan is designed to plan, schedule, monitor, and replan (if required) a heterogeneous set of UAS to complete a mission consisting in minimizing the time performed in visiting a set of Points of Interest (PoIs) while ensuring that all generated trajectories are safe, feasible, consistent with wind effects, and compliant with the required PoIs' arrival times, UAS kinematics and energetic constraints, and the existing 3D No-FLy Zones (NFZs). Every PoI –defined by its 3D coordinate (latitude, longitude, altitude)– must be visited exactly once by one UAS of the fleet and the maximum arrival time for each PoI should be lower than the given values. The set of UAS flight plans should minimize the *makespan*, defined as the time at which the last UAS completes its mission and arrives at an end depot. This optimization is subject to the constraint that all trajectories should be safe, feasible for each UAS both kinematically and also in terms of energy consumption, and compliant with all NFZs avoidance requirements.

U-Plan is devised to be as realistic as possible:

- The mission includes a defined start depot for each UAS (the initial UAS location), but a number of potential end depots, a shared pool of landing sites, e.g., airfields where the UAS can land.
- Each UAS is characterized by its specific kinematic and energetic constraints, including maximum roll angle, maximum airspeed, and its aerodynamic specifications, including the aerodynamic reference area and the drag coefficient.
- The wind field in the environment is used for computing the UAS required airspeed and energy consumption. The Digital Elevation Model (DEM) of the environment is used to check for trajectories safety and terrain collisions.
- The airspace is assumed realistic with potential NFZs, defined as 3D volumes that UAS must strictly avoid. NFZs can be declared dynamically at any time during the mission, and can have different shapes, including cylindrical (circular footprint), prismatic (polygonal footprint), or composite shapes formed by a combination of linear and curved boundaries.
- The mission definition, environment conditions, and airspace restrictions can change dynamically, and if required, U-Plan recomputes the flight plans to adapt to the new conditions.

U-Plan is designed as a practical tool, and keeps the human in the loop. U-Plan solves the optimization problem, outputs a feasible flight plan for each UAS, and provides it to the operator for approval together with the predicted mission completion data. The operator may accept the solution given by U-Plan, or modify the mission definition and requirements and re-run U-Plan. In addition, U-Plan includes a module that dynamically supervises the fulfillment of the mission and the changes/updates in the mission definition, environment conditions, and airspace restrictions, and also predicts the fulfillment or not of the mission in the new conditions. In case of nonfulfillment or by operator request, U-Plan triggers replanning. It has been designed and developed to meet the following principles:

- **Efficient planning**. U-Plan must produce close-to-optimal safe UAS flight plans to fulfill the mission efficiently in terms of *makespan* satisfying the mission, UAS, environment, and airspace requirements and constraints.
- **Kinematically and energetically feasible flight plans**. The generated flight plan for each UAS must be feasible, considering the UAS kinematics and energy consumption.

 doi:10.20944/preprints202605.0937.v1

- **Dynamic reconfigurability**. U-Plan must efficiently adapt to changes in mission definition, UAS, environment, and airspace constraints. U-Plan must be capable of ingesting updates and efficiently generating new, safe, and feasible flight plans.
- **Scalability**. U-Plan must scale well with the problem size (e.g., number of UAS, number of PoIs , number of NFZs) without a significant performance degradation.

The architecture of U-Plan consists of three main components, see Figure 1. **Multi-UAS Route Computation** solves a Vehicle Routing Problem (VRP) to determine the route that is assigned to each UAS. Conversely to most existing approaches, which treat NFZ avoidance as a post-processing often leading to longer UAS routes, our approach integrates NFZ avoidance within the VRP. Module *NFZ-Aware Cost Generation* calculates the cost matrix of the VRP explicitly accounting for NFZs to ensure that no route violates existing NFZs. Module *Wind-Related Constraints* incorporates the impact of the wind field on the mission, translating wind effects into feasibility constraints regarding maximum airspeed limits and battery energy consumption. The cost matrix resulting from *NFZ-Aware Cost Generation* together with the wind-related constraints are used by *Multi-UAS VRP Solver* to solve the multi-UAS route computation. For this role, we employ Google's OR-Tools [29], selected for its ability to produce high-quality, near-optimal solutions with high computational efficiency and good scalability. This component is described in Section 4.1.

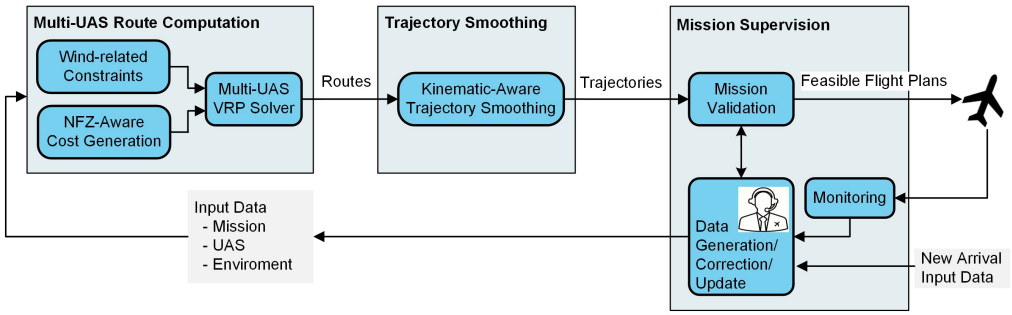


Figure 1. General scheme of the U-Plan architecture

Trajectory Smoothing, see Section 4.2, transforms the piece-wise routes assigned to each UAS into kinematically feasible trajectories. In general, the routes' vertices do not respect the minimum turning radii of fixed-wing UAS. To solve it, our method inserts waypoints in the UAS routes, generating smooth turns that satisfy the minimum turning radius of that UAS while minimizing the deviation w.r.t. the original piece-wise route to help preserve compliance of NFZs. The output is an ordered sequence of waypoints (with temporal information) for each UAS that represents a 3D kinematically feasible trajectory.

Mission Supervision, see Section 4.3, is responsible for the validation and monitoring of the generated flight plans, considering the mission requirements and the mission updates. *Mission Validation* checks the UAS trajectories against the mission requirements. It re-validates that the maximum airspeed and PoI maximum arrival time constraints are still satisfied after trajectory smoothing, ensuring that the final path adheres to all mission deadlines and kinematic limits. It also checks the fulfillment of safety operational requirements, including: minimum and maximum allowable distances between consecutive waypoints, maximum permissible number of waypoints per trajectory, Radio Line-of-Sight (RLOS) range to assess potential communication loss, potential inter-UAS collision, and the DEM of the environment to check for terrain collisions. If all the constraints are satisfied, the final flight plans, comprising 3D waypoints and specific airspeeds, are sent to the UAS. Otherwise, specific errors are flagged and reported to the operator for correction. *Monitoring* oversees the mission continuously, checking for mission updates, critical anomalies, and mission fulfillment prediction and alerts the operator to abort the mission or trigger a replanning sequence. Upon detecting a trigger event, such as failure in any UAS, changes or new NFZs, changes or new PoIs, or fleet modifications, it prompts

 doi:10.20944/preprints202605.0937.v1

the operator to initiate replanning. To ensure mission continuity, the input data is updated to reflect the current state of the mission: active UAS locations are treated as new start depots, and remaining unvisited targets are consolidated into a fresh VRP instance. This allows U-Plan to generate a seamless extension of the ongoing mission without requiring a manual mission abort.

4. Methods

Table 1 summarizes the adopted notation. Let $\mathcal{P}$ be the set of PoIs, each defined with its latitude, longitude, altitude and maximum required arrival time τ_{max}. $\mathcal{S}$ is the set of start depots, which are taken as the initial location of each UAS. $\mathcal{E}$ is the set of end depots, the pool of depots where the UAS can finish their trajectories. $\mathcal{N}$ is the set of all nodes considered in the problem, PoIs and start and end depots, i.e., $\mathcal{N} = \mathcal{P} \cup \mathcal{S} \cup \mathcal{E}$. $\mathcal{V}$ is the set of all UAS, ϕ^v_{max} is the maximum roll angle, $V^v_a|_{max}$ is the maximum airspeed, E^v_{ini} is the initial energy level, a^v is the aerodynamic reference area, and c^v_d is the drag coefficient of UAS v. A mission is initiated by the operator providing the following:

- **Mission data**: the 3D coordinates of all PoIs in $\mathcal{P}$ and end depots in $\mathcal{E}$, as well as τ^i_{max} for each PoI $i \in \mathcal{P}$,
- **UAS data**: the number of all UAS ($|\mathcal{V}|$), the kinematic constraints (ϕ^v_{max} and $V^v_a|_{max}$), the aerodynamical coefficients (a^v and c^v_d), and the initial state of each UAS v (start depot $s^v \in \mathcal{S}$ and E^v_{ini}),
- **Environmental and airspace data**: The definition of all 3D NFZs denoted by $\mathcal{Z}$, the DEM for terrain constraints, and the horizontal wind field.

Table 1. Nomenclature and variable definitions used in the mathematical formulation.

Symbol	Description
$\mathcal{P}$	Set of all PoIs
$\mathcal{E}$	Set of end depots (shared pool)
τ^i_{max}	Maximum required arrival time of PoI i ($i \in \mathcal{P}$)
$\mathcal{V}$	Set of all UAS
$\mathcal{S}$	Set of start depots
$V^v_a\|_{max}$	Maximum airspeed of UAS v
E^v_{ini}	Initial energy level of UAS v
ϕ^v_{max}	Maximum roll angle of UAS v
c^v_d	Drag coefficient of UAS v
a^v	Aerodynamic reference area of UAS v
$\mathbf{W}$	Wind speed vector
$\mathcal{Z}$	Set of all 3D NFZs
$\mathcal{N}$	Set of all nodes, $\mathcal{N} = \mathcal{P} \cup \mathcal{S} \cup \mathcal{E}$
$\mathcal{R}$	Set of VRP generated routes
V^v_g	Ground speed of UAS v
$V^v_{a,ij}$	Airspeed of UAS v along segment (i, j)
c^v_{ij}	Travel time (cost) for UAS v from node i to node j
D	Distance matrix $D = [d_{ij}]$
R^v_{min}	Minimum turning radius of UAS v
T^v	Total flight time of UAS v
T_{max}	Makespan (maximum flight time across the fleet)
u^v_i	Continuous variable for node i and UAS v (MTZ subtour elimination)
x^v_{ij}	Binary variable; 1 if UAS v travels directly from node i to node j, 0 otherwise

 doi:10.20944/preprints202605.0937.v1

4.1. Multi-UAS Route Computation

This component assigns the routes to each UAS of the fleet. It solves an optimization problem consisting in determining the routes for a set of heterogeneous UAS that minimize the makespan in visiting a set of PoIs while ensuring that: i) the routes comply with existing NFZs, ii) the required maximum PoI's visiting times are satisfied, iii) the UAS travel with an airspeed is not higher than its maximum value, and iv) every UAS has enough energy to complete its route. Conversely to traditional approaches, our approach includes NFZs avoidance and wind effects in the VRP optimization, which requires specific mechanisms.

Solving a VRP calls for a cost matrix between all nodes. In VRP problems that minimize makespan, the cost matrix reflects time, and in most cases is static. In U-Plan, as UAS can have different velocities, the cost matrix cannot be pre-calculated as a static matrix. To cope with that, the cost matrix used in *Multi-UAS VRP Solver* is not expressed in terms of travel time. Instead, it uses a distance matrix D, which entry d_{ij} reflects the distance between every pair of nodes i and j, and the VRP solver uses D to dynamically compute the specific time required for each UAS v to travel from i to j as $c_{ij}^{v} = d_{ij}/V_{g}^{v}$, where V_{g}^{v} is the ground speed of UAS v. In standard VRP approaches, the cost matrix is computed using the Haversine distance, i.e., the shortest distance between two nodes on the Earth's sphere. This straightforward approach is not suitable for U-Plan. First, the straight Haversine distance paths between nodes are not always feasible due to the presence of NFZs, and second, addressing NFZs avoidance in a separate post-processing stage often leads to longer routes. To cope with NFZs, our approach includes module *NFZ-Aware Cost Generation*, which adapts the VRP cost matrix to ensure that the routes don't collide with NFZs, circumventing NFZs if the straight Haversine route generate collisions. In addition, traditional VRP do not consider the wind effects. Wind speed influences the UAS airspeed and the UAS energy consumption, hence affecting the capability of the UAS to perform a trajectory without energy exhaustion. To cope with them, module *Wind-related Constraints* manages these wind effects and incorporates them as constraints in the VRP.

4.1.1. NFZ-Aware Cost Generation

Instead of addressing obstacles and NFZs in a posterior route planning process, U-Plan considers them in the routing planning problem by using an adapted version of the distance matrix, constructed as shown in Algorithm 1.

 doi:10.20944/preprints202605.0937.v1

Algorithm 1 NFZ-Aware Distance Matrix Generation

Input: $\mathcal{N} = \{1, 2, ..., n\}, \mathcal{Z} = \{1, 2, ..., z\}$
Output: $D = [d_{ij}]$
$D \leftarrow$ new $n \times n$ matrix, initialized to 0
for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 1$ to n **do**
        if $i = j$ **then continue**
        end if
        $Z_{sides} \leftarrow \{\emptyset\}$, $Z_{vertices} \leftarrow \{\emptyset\}$
        for k in $\mathcal{Z}$ **do**
            if Straight segment from node i to node j intersects k **then**
                Construct a polygon on a 2D geodesic envelope points around k
                $Z_{sides} \leftarrow Z_{sides} +$ Sides of the polygon
                $Z_{vertices} \leftarrow Z_{vertices} +$ Vertices of the polygon
            end if
        end for
        if $Z_{sides} \neq \{\emptyset\}$ **then**
            Build a bidirectional graph on nodes i and j and $Z_{vertices}$ such that no edge
            intersects each $l \in Z_{sides}$
            $d_{ij} \leftarrow$ The shortest path's length from i to j by running A^* over the graph
        else
            $d_{ij} \leftarrow$ Haversine distance from i to j
        end if
    end for
end for

Let $\mathcal{N} = \{1, 2, ..., n\}$ be the set of considered nodes, which includes the PoIs and depots (start and end depots), and let $\mathcal{Z} = \{1, 2, ..., z\}$ be the set of 3D NFZs. For each pair of nodes (i, j), it is checked if the straight geodesic path between i and j intersects with any NFZ. If no intersection occurs, the Haversine distance between i and j is computed and stored as d_{ij}. If an intersection with a NFZ exists, we have to find a route from i to j circumventing the NFZ. For that purpose, we define a 2D geodesic envelope around the NFZ considering a safety margin, and construct a graph that includes the start and end nodes (i and j) and a set of points on the 2D geodesic envelope. The points are selected considering the NFZ geometry: vertices are selected for polygonal segments, while a set of uniformly distributed points is generated to approximate curved sections, ensuring accurate envelope representation for circular, polygonal, and composite shapes alike. The graph is constructed with bidirectional edges between any pair of points whose geodesic straight path does not intersect any NFZ, and the cost of each edge is taken as the Haversine distance between its points. Then, the shortest path between i and j in this graph is found. The cost of the shortest path is stored as d_{ij} in the distance matrix D. This process is illustrated in Figure 2.

 doi:10.20944/preprints202605.0937.v1

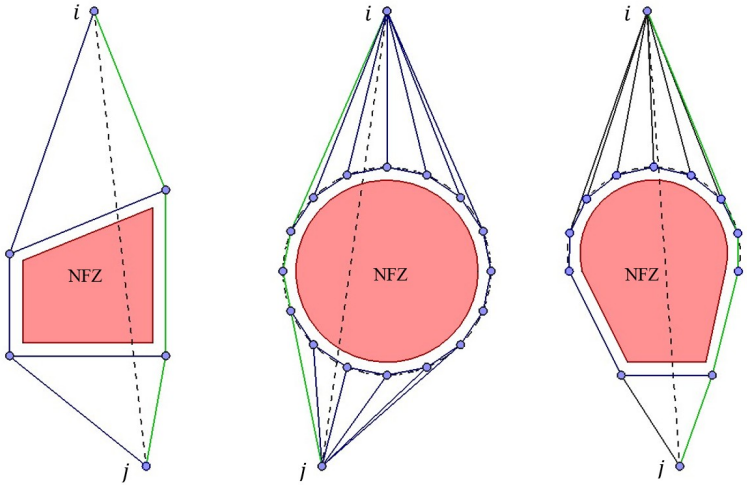


Figure 2. Graph construction for A^* No-Fly Zone (NFZ) avoidance. The graph includes the start/end points i and j, and the points on the NFZ envelope considering the safety margin. A^* finds the shortest path (in green colour) by exploring the edges of this graph around both polygonal (left), circular (center), and composite (right) NFZs.

In the proposed method, the resulting D includes the cost of feasible paths between nodes in $\mathcal{N}$. The adopted method is simple, efficient, and can obtain notable performance gains, as is shown in Section 5. The experimental results presented in the paper were obtained using A^* to find the shortest path in the constructed graph. It was selected due to its optimality, completeness, efficiency, and simplicity of implementation.

4.1.2. Wind-Related Constraints

Wind strongly affects the flight speed and energy consumption of UAS. First, the wind field modifies the required UAS airspeed for achieving a desired ground-velocity vector, and the maximum airspeed is an intrinsic characteristic for each UAS, which cannot be exceeded. In addition, wind-induced changes in the UAS airspeed have a direct impact on propulsion power and energy consumption, impacting on the capability of the UAS to perform a trajectory without exhausting its initial energy. This subsection describes how both constraints are incorporated into **Multi-UAS Route Computation**.

We assume that the wind field is provided as a discretized wind map $\mathbf{W}$ with a given spatial resolution, where each map cell includes a 2D vector that represents the wind speed and direction. This data can be retrieved using online meteorological services and APIs, such as Open-Meteo [37] or Meteomatics [38]. The wind vector at the cell that corresponds to the location of node i is represented by $\mathbf{W}_i$. The UAS fly at medium altitudes, where wind is more stable than on the surface. Hence, we can assume that the wind vector is similar at nearby locations. When a UAS traverses segment (i, j), the wind in the segment can be approximated by vector $\overline{\mathbf{W}}_{ij}$, the mean of the wind vectors along the segment.

In **Multi-UAS Route Computation** component, each UAS is commanded to follow its assigned route with a constant ground speed magnitude. The airspeed experienced by each UAS in each segment depends on the wind component along the segment. Denoting by $W_{\parallel,ij}$ the projection of the mean wind vector $\overline{\mathbf{W}}_{ij}$ onto the direction of the segment (i, j), the scalar velocity relation linking ground velocity, airspeed, and wind is:

$$V^v_{a,ij} = V^v_g - W_{\parallel,ij}, \tag{1}$$

where $V^v_{a,ij}$ is the airspeed of UAS v along the segment (i, j). This expression determines the aerodynamic effort the UAS must provide to maintain the desired motion. Because V^v_g is fixed by design in our routing module, variations in V^v_a arise from the wind field.

 doi:10.20944/preprints202605.0937.v1

Wind also has important effect on UAS energy consumption. For fixed-wing UAS conducting steady cruise flight, the dominant aerodynamic loss is the parasitic drag, which balances thrust (T_{ij}) in steady flight, and grows quadratically with the airspeed, then:

$$T_{ij} = \frac{1}{2}\rho a^v c_d^v \left(V_{a,ij}^v\right)^2, \tag{2}$$

where ρ is the air density, and a^v and c_d^v are respectively the aerodynamic reference area and drag coefficient of UAS v. Let η denote the propulsive efficiency relating the aerodynamic power, defined as $P_{a,ij} = T_{ij}\, V_{a,ij}^v$, to the corresponding electrical power $P_{e,ij}$ drawn from the battery. The electrical power drawn from the battery for propulsion is then $P_{e,ij} = P_{a,ij}/\eta$. Consequently, the total battery energy consumed due to propulsion over a segment of duration Δt is:

$$E_{ij}^v = \int_0^{\Delta t} P_{e,ij}(t)\, dt = \frac{1}{2\eta}\rho a^v c_d^v \left(V_{a,ij}^v\right)^3 \Delta t. \tag{3}$$

Substituting the airspeed–wind relationship (1) and the segment duration $\Delta t = c_{ij}^v$, we obtain the wind-aware propulsive energy consumption model for segment (i, j):

$$E_{ij}^v = \frac{1}{2\eta}\rho a^v c_d^v \left(V_g^v - W_{\parallel,ij}\right)^3 c_{ij}^v. \tag{4}$$

Equation (4) explicitly reveals the nonlinear effect of wind on propulsive energy consumption. This relationship provides a quantitative link between the travel time of a segment, the wind conditions, and the propulsive energy consumed by each UAS along its assigned route. This energy model is directly integrated into the VRP formulation in the following subsection as a constraint, where it is checked against the usable battery capacity allocated for propulsion (total capacity minus safety margin and non-propulsive loads).

4.1.3. VRP Formulation for Heterogeneous UAS

With the NFZ-aware distance matrix D providing the geometric basis for travel costs, we can now formally define the routing problem. The problem is formulated as a multi-depot heterogeneous VRP. Consistent with the system architecture, each UAS is assigned a specific start depot, while end depots are selected by the solver from a shared pool. Our objective is to optimize the fleet for the fastest overall mission completion. Therefore, we aim to minimize the makespan i.e., the time of the longest individual UAS route. The complete mathematical model is detailed below.

The decision variable is x_{ij}^v, a binary variable that equals to 1 if UAS v travels directly from node i to node j, and 0 otherwise. c_{ij}^v represents the cost (travel time) for UAS v to traverse the segment (i, j), derived specifically from the NFZ-aware distance matrix and the UAS's ground speed. The objective is to minimize the makespan (T_{max}), which favors that the mission workload is balanced among the different UAS of the fleet:

$$\min\ T_{\max} \tag{5}$$

The objective function is subject to the following constraints:

- **Makespan Constraint**

$$T_{max} \geq T^v \quad \text{where} \quad T^v = \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}, i\neq j} c_{ij}^v x_{ij}^v \qquad \forall v \in \mathcal{V} \tag{C1}$$

where T^v denotes the total flight time for UAS v. Constraint (C1) links the individual mission times with the makespan T_{max} by defining it as being greater than or equal to every individual UAS's travel

 doi:10.20944/preprints202605.0937.v1

time, enabling the optimization objective to minimize the completion time of the entire fleet.

• **PoI Visiting Constraints**

$$\sum_{v\in\mathcal{V}}\sum_{i\in\mathcal{N},i\neq p} x_{ip}^{v} = 1 \qquad \forall p \in \mathcal{P} \tag{C2}$$

$$\sum_{i\in\mathcal{N},i\neq p} x_{ip}^{v} = \sum_{j\in\mathcal{N},j\neq p} x_{pj}^{v} \qquad \forall p \in \mathcal{P}, \forall v \in \mathcal{V} \tag{C3}$$

Constraint (C2) ensures that each PoI is visited exactly once by one UAS. Constraint (C3) is the flow conservation constraint, stating that if a UAS v arrives at a PoI p, it must also depart from it.

• **Multi-depot constraints**

$$\sum_{j\in\mathcal{P}\cup\mathcal{E}} x_{s^v j}^{v} = 1 \qquad \forall v \in \mathcal{V} \tag{C4}$$

$$\sum_{e\in\mathcal{E}}\sum_{i\in\mathcal{P}\cup\{s^v\}} x_{ie}^{v} = 1 \qquad \forall v \in \mathcal{V} \tag{C5}$$

$$\sum_{i\in\mathcal{N}} x_{si}^{v} = 0 \qquad \forall v \in \mathcal{V}, \forall s \in \mathcal{S}\backslash\{s^v\} \tag{C6}$$

$$\sum_{i\in\mathcal{N}} x_{is^v}^{v} = 0 \qquad \forall v \in \mathcal{V} \tag{C7}$$

$$\sum_{j\in\mathcal{N}} x_{ej}^{v} = 0 \qquad \forall v \in \mathcal{V}, \forall e \in \mathcal{E} \tag{C8}$$

Constraint (C4) forces each UAS v to depart from its specific start depot s^v. Constraint (C5) ensures that each UAS ends its route at one end depot chosen from the shared pool. Constraint (C6) prevents any UAS v from launching from a start depot assigned to another UAS k, unless they share the same physical location. Finally, constraints (C7) and (C8) prevent a UAS from returning to its own start node or departing from an end node.

• **Subtour elimination constraints (MTZ)**

Let u_i^v be the auxiliary continuous positive variable representing the sequence order in which node i is visited by UAS v. Additionally, $|\mathcal{P}|$ denotes the total number of PoIs.

$$u_i^v - u_j^v + |\mathcal{P}| \cdot x_{ij}^v \leq |\mathcal{P}| - 1 \qquad \forall i,j \in \mathcal{P}, i \neq j, \forall v \in \mathcal{V} \tag{C9}$$

$$\sum_{j\in\mathcal{N},j\neq i} x_{ji}^v \leq u_i^v \leq |\mathcal{P}| \sum_{j\in\mathcal{N},j\neq i} x_{ji}^v \qquad \forall i \in \mathcal{P}, \forall v \in \mathcal{V} \tag{C10}$$

This set of constraints prevent the formation of disconnected subtours on the PoI nodes. Constraint (C9) is the primary MTZ constraint, which enforces sequential order. Constraints (C10) links the u variables to the x variables, ensuring that u_i^v is greater than 0 if and only if PoI i is visited by UAS v.

• **Feasibility Constraints (Energy, Speed, Time)**

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N},j\neq i} E_{ij}^v x_{ij}^v \leq E_{avail}^v \qquad \forall v \in \mathcal{V} \tag{C11}$$

$$x_{ij}^v = 0 \qquad \text{if } V_{a,ij}^v > V_a^v|_{max} \tag{C12}$$

$$t^i \leq \tau_{max}^i \qquad \forall i \in \mathcal{P} \tag{C13}$$

Constraint (C11) ensures that the total propulsive energy consumed by UAS v does not exceed the available propulsive energy budget E_{avail}^v. This budget is derived from the UAS's initial energy state (E_{ini}^v), adjusted to reserve the required safety margin and account for non-propulsive loads (e.g., avionics). Constraint (C12) enforces the maximum airspeed limit by forbidding any edge (i,j) where the required airspeed exceeds the UAS's maximum airspeed. Finally, constraints (C13) enforces time windows, where t^i represents the arrival time at node i and τ_{max}^i is the required maximum time for visiting PoI i.

 doi:10.20944/preprints202605.0937.v1

This Mixed-Integer Linear Programming (MILP) formulation is implemented and solved using Google's OR-Tools routing library. The selection of OR-Tools is based on the trade-off between high-quality solutions and very short computation times, which is essential for meeting the efficient planning/replanning and scalability requirements. While exact solvers become very slow as problem size increases, OR-Tools makes use of heuristics to efficiently produce high-quality solutions for large-scale problems. A detailed scalability comparison validating this choice is presented in Section 5.

4.2. Trajectory Smoothing

Multi-UAS Route Computation assigns for each UAS a NFZs-compliant piece-wise route that can include turns that may violate the kinematical constraints of the UAS (assumed fixed-wing in our research). This module smooths these piece-wise routes to ensure the kinematic feasibility requirement while reducing the deviation w.r.t. the original piece-wise routes (necessary to preserve the NFZs compliance of the route).

Many methods have been proposed in the literature for trajectory smoothing, including Cubic Splines [39] and Dubins curves [40]. While Spline-based techniques generate continuous and differentiable paths that hit every target, they cannot inherently satisfy the UAS kinematic requirements and may generate curves with instantaneous curvatures smaller than the UAS's minimum turning radius. Dubins curves can respect kinematic limits, but forcing the path to pass through the vertex with a fixed orientation often results in rigid geometries that can deviate noticeably from the original route, risking NFZ compliance. In addition, many commercial autopilots employ an "Overfly" logic to ensure the UAS passes exactly over the waypoint [41]. In this configuration, the UAS crosses the vertex before initiating a turn to align with the next leg. To recover the path efficiently, the guidance system typically enforces a steep intercept angle (commonly 45°) to converge back to the desired track [36]. However, these maneuvers cause post-vertex overshoots (which grow rapidly with the magnitude of the turn) that could make the UAS violate a NFZ. Furthermore, this approach implies a strict geometric limitation: given the requirement to settle onto the path with a specific fillet curvature (R^v_{min}) from a 45° approach angle, it can be shown that it becomes geometrically infeasible when the turning angle at the vertex exceeds approximately 2 radians = 114.5°.

In U-Plan we adopt a geometry-based smoothing method that considers the minimum turning radius of that UAS, making a kinematically feasible maneuver, while reducing the deviation w.r.t. the original path and helping preserve NFZ compliance. Unlike Overfly, which reacts to the turn after passing the vertex, our approach anticipates the maneuver by treating the turn geometry as a constraint. The process is shown in Algorithm 2.

 doi:10.20944/preprints202605.0937.v1

Algorithm 2 Proposed Trajectory Smoothing Algorithm

Input: $\mathcal{R}, \mathcal{Z}, V_g = \{V_g^v : v \in V\}, \phi_{max} = \{\phi_{max}^v : v \in V\}$
Output: $\mathcal{T}$ (Set of kinematically feasible trajectories)
$\mathcal{T} \leftarrow \{\emptyset\}$
for each route $r \in \mathcal{R}$ corresponding to UAS v **do**
    $R_{min}^v \leftarrow Equation6$, $waypoints \leftarrow \{\emptyset\}$, $z_{old} \leftarrow \{\emptyset\}$
    for $i \leftarrow 2$ to length(r) - 1 **do**
        $\vec{v}_{in} \leftarrow r[i] - r[i-1]$, $\vec{v}_{out} \leftarrow r[i+1] - r[i]$
        $\vec{b} \leftarrow$ Bisector of angle between $\vec{v}_{in}$ and $\vec{v}_{out}$
        $C_i \leftarrow r[i] + R_{min}^v \cdot \vec{b}$
        $z_{new} \leftarrow \text{Circle}(\text{center} = C_i, \text{radius} = R_{min}^v)$
        $Z_{total} \leftarrow Z \cup \{z_{new}\} \cup \{z_{old}\}$
        Run A^* pathfinder from $r[i-1]$ to $r[i]$ avoiding Z_{total}
        $Segment_{wp} \leftarrow$ Resulting A^* path points
        Append $Segment_{wp}$ to $waypoints$
        Remove z_{old} from Z_{total}
        $z_{old} \leftarrow z_{new}$
    end for
    Run A^* pathfinder from $r[i]$ to $r[i+1]$ avoiding Z_{total}
    $Segment_{wp} \leftarrow$ Resulting A^* path points
    Append $Segment_{wp}$ to $waypoints$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{waypoints\}$
end for

Let $\mathcal{R}$ be the set of computed routes by **Multi-UAS Route Computation**. First, for every route $r \in \mathcal{R}$ corresponding to UAS v, we calculate the minimum turning radius R_{min}^v based on the kinematic limits of UAS v:

$$R_{min}^v = \frac{V_g^{v2}}{g \tan \phi_{max}^v}, \tag{6}$$

where ϕ_{max}^v is the UAS maximum roll angle and g is the gravitational constant. For each point i in route r, to construct the smooth turn, we identify the bisector of the angle formed by the incoming segment (from point $i-1$) and the outgoing segment (to point $i+1$). We then define the center point C_i located on this bisector at a distance R_{min}^v from vertex i, such that the circle of radius R_{min}^v centered at C_i has vertex i on its perimeter. This circle is treated as an artificial NFZ. We then apply the A^* path finder to generate a local path that connects the incoming and outgoing segments while avoiding this artificial obstacle. By positioning the circle on the bisector tangent to the vertex, A^* naturally obtains the shortest flyable arc that hugs the perimeter of the circle. This ensures that the trajectory comes as close as kinematically possible to the vertex without requiring an impossible sharp turn.

Figure 3 shows an example comparing the same turning geometries solved with the Overfly approach (left) and the proposed approach (right). Our method generate shorter paths compared to Overfly, which requires a large overshoot to intercept the next leg. As analyzed in Section 5.2, the adopted approach consistently generated shorter paths than methods based on Overfly, Splines and Dubins in all the experiments we performed.

 doi:10.20944/preprints202605.0937.v1

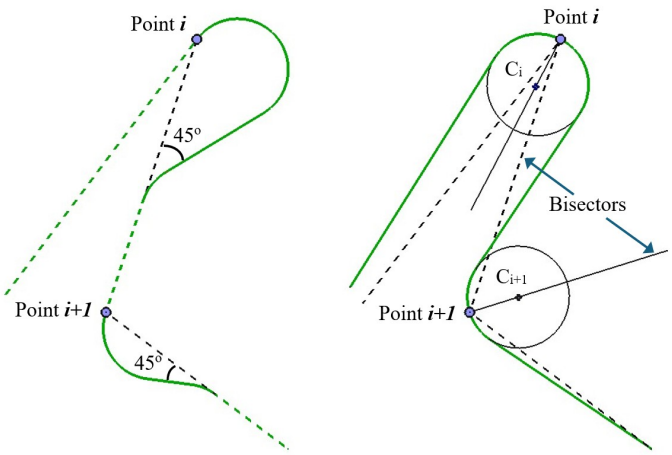


Figure 3. Example comparing the adopted trajectory smoothing method (right) and the Overfly maneuver (left). The Overfly maneuver results in large overshoots, especially in sharp angles that can cause significant deviations from the original path (dashed lines). Our method generates smooth, kinematically feasible turns around artificial NFZs constructed tangent to the vertices, reducing deviations and ensuring safety.

4.3. Mission Supervision

U-Plan incorporates a supervision module that functions as the final dynamic manager of the mission to ensure operational safety and robustness and adapt to updates. *Mission Validation* checks the generated flight plan against the mission requirements. *Monitoring* continuously checks for mission updates, critical anomalies (e.g., UAS or hardware failures), and mission nonfulfillment prediction to trigger mission replanning.

4.3.1. Mission Validation

Once the smoothed trajectories are generated, they undergo a validation process before uploading the flight plans to the UAS. This step functions as a safety gate, ensuring that the planned waypoints satisfy all physical and operational constraints defined in the static mission acceptance parameters. The system checks for the following critical errors:

- **Segment Length Violations:** The Haversine distance between any two consecutive waypoints must fall within the range $[d_{min}, d_{max}]$, the minimum and maximum distances for the autopilots of the UAS. Segments shorter than d_{min} may cause guidance instability, while segments longer than d_{max} may lead to path tracking errors over long distances.
- **Maximum Number of waypoints**: The total number of waypoints generated for each UAS is checked against a maximum permissible threshold.
- **Terrain Collision:** The Above Mean Sea Level (AMSL) of every waypoint must be strictly higher than the underlying terrain elevation plus a safety distance. This check utilizes a DEM to ensure that no waypoint is located underground or dangerously close to the terrain surface.
- **Communication Range:** The distance between every waypoint and the Ground Control Station (GCS) is calculated to ensure it remains within the effective Radio Line-of-Sight (RLOS) range, preventing communication loss.
- **Maximum Airspeed Compliance:** The smoothed trajectory is re-evaluated against the wind field to ensure that no segment requires an airspeed exceeding the UAS limit, accounting for any path modifications introduced by the smoothing process.
- **Maximum Arrival Time Compliance:** The arrival times at all PoIs are recalculated based on the final trajectory length to verify that all PoI maximum arrival times (τ^i_{max}) are still met, as the smoothing process can slightly alter the trajectory distances.
- **Inter-UAS Collision:** It ensures that the distance between any two UAS never falls below a predefined safety threshold at any moment during the mission, by evaluating the continuous trajectories of the fleet.

 doi:10.20944/preprints202605.0937.v1

If any of these constraints is violated, the specific error is flagged to the human operator and the plan is rejected. Using that flag, the human operator can modify the specific mission inputs and requirements (e.g., PoI location, assigned altitude, fleet selection, or modifying the altitude of one UAS at the predicted inter-UAS collision point as in [42]) to resolve the conflict, and re-launch the computation of the mission with the new inputs.

4.3.2. Monitoring

This module subscribes to the operator's command stream, external data feeds, and UAS telemetry to continously monitor the mission execution and updates in order to detect risks for the successful completion of the mission, including:

- **Mission Updates:** Changes such as the change or addition of new PoIs to the visiting list, changes in the end depots, change in required arrival time for any PoI, fleet expansion via the introduction of new UAS, or environmental changes like the definition of new NFZs and changes in the wind field.
- **Critical Anomalies:** Emergency situations detected via telemetry, such as a UAS hardware failure, loss of link, or battery levels dropping below the critical threshold during flight.
- **Mission Fulfillment Prediction:** Proactive evaluation of mission progress to predict potential failures before they occur. This includes identifying if a waypoint has been missed, forecasting if the projected arrival time at any subsequent PoI will exceed the required deadline (τ_{max}), or calculating if the remaining energy is insufficient to reach a valid end depot under current environmental conditions.

When any of these events are detected, the system alerts the operator to stop the mission and trigger an immediate replanning sequence. The sequence proceeds as follows:

- **Step 1: Loiter and Hold.** All active UAS are commanded to stop their current trajectory advancement and enter a loiter pattern describing circular trajectories of radii R^v_{min} centered at their current GNSS location. This freezes the mission safely while U-Plan computation occurs. This step could be avoided for low-size problems due to the low re-computation times of U-Plan.
- **Step 2: Start Depot Update.** The current GNSS location of every UAS is captured and designated as the new start depot for the replanning problem. If new UAS are added in the new mission plan, their launch sites are added as start depots.
- **Step 3: Parameters Update.** The **Mission Data**, **UAS data**, and **Environmental and airspace data** are updated to reflect the new state. This involves: (i) removing all already visited PoIs from the target list; (ii) updating the coordinates and arrival time of any existing PoIs if changed; (iii) adding any newly defined PoIs; (iv) updating the set of available end depots if the pool has changed, (v) updating the UAS fleet if there is any failure or new addition; and (vi) ingesting any new environmental constraints, such as new NFZ definitions or updated wind field data.
- **Step 4: Re-computation. Multi-UAS Route Computation** and *Trajectory Planning* are executed from scratch using this updated **Mission Data**, **UAS data**, and **Environmental and airspace data**.

Once the new plan has been generated, its suitability has been confirmed by *Mission* Validation, and it has been sent to the UAS in the fleet, the UAS exit the loiter pattern and starts executing the new flight plan. This capability allows U-Plan to adapt seamlessly to dynamic missions and emergencies without interruption.

5. Validation

Validating solutions for managing UAS swarms is highly complex due to the simultaneous interaction of multiple factors: environmental characteristics (topography, presence of obstacles), operational requirements (number of UAS, their speeds, altitudes, separation distances, etc.). In this context, simulations are an essential tool, providing a safe, controlled, and reproducible environment for analyzing performance. U-Plan was validated, integrated in a professional setup comprising

 doi:10.20944/preprints202605.0937.v1

Visionair GCS Ground Control Station Software and VECTOR-SIL Software-in-the-Loop simulator from UAV Navigation [43], a Spanish company specialized in UAV guidance, navigation, and control systems. This setup realistically replicates the behavior of certified fixed-wing UAS autopilots under all weather conditions. Utilized by Tier 1 aerospace manufacturers, this environment accounts for real-world avionics constraints, latencies, and communication protocols to ensure a safe and reproducible framework for assessment. U-Plan was implemented in Python, making use of Google Maps via the Map JavaScript API for the graphical user interface and trajectory visualization. It was executed on a standard desktop PC equipped with an 11th Gen Intel® Core™ i7 processor with 32 GB of RAM. Through this integrated setup, the full U-Plan architecture and its individual components have been extensively validated and compared with existing approaches, demonstrating their capacity to satisfy complex mission requirements with short execution times and low UAS energy consumption.

For the validation of U-Plan, we adopted a bottom-up approach. First, the validation of each individual module is presented and analyzed independently. Next, the overall U-Plan architecture is evaluated in complete missions, validating the seamless interaction between all components.

5.1. Multi-UAS Route Computation

First, we validate the effectiveness of the proposed NFZ-aware cost generation strategy, which integrates obstacle avoidance cost during the VRP optimization. We compare it to the traditional post-VRP detour approach. We designed an experiment involving two UAS with the same ground speed ($V_g = 120$ km/h), 6 PoIs, and the start and end depots separated by approximately 45 km. The environment includes one composite NFZ formed by a circular zone intersecting a rectangular zone, and another circular NFZ. Figure 4 shows an example that compares the resulting routes by using: (a) VRP solver ignoring NFZs; (b) traditional post-VRP detour method; and (c) result of our NFZ-aware method.

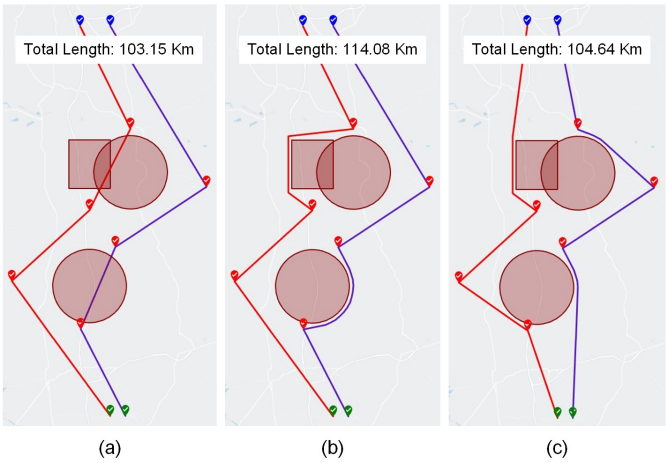


Figure 4. Example comparing the routes resulting from different routing strategies: (a) VRP ignoring NFZs (calculated using simple Haversine distances without considering NFZs); (b) traditional post-VRP detour method in which invalid segments are modified using A^* to avoid NFZs; (c) the proposed method. PoIs are marked in red color. Start depots, in blue. End depots, in green.

The VRP solver, ignoring NFZs, generated routes with a total distance of 103.15 km, which were operationally infeasible as both traversed the NFZ. The traditional post-VRP detour method modified the invalid segments to avoid the NFZ, forcing the UAS to take long detours around the NFZs, resulting in a total length of 114.08 km. Our method generated valid routes with a significantly shorter total route length of 104.64 km. By encoding the NFZ into the cost matrix prior to the optimization, the VRP solver detects that the direct connection is costly and selects a completely different visiting sequence.

A large number (> 300) of randomized missions with different number and locations of PoIs and positions/sizes of the NFZ were performed. In all of them the proposed NFZ-aware cost update

 doi:10.20944/preprints202605.0937.v1

method obtained shorter or equal route lengths compared to the post-VRP detour approach. Table 2 presents the results for 10 of these experiments. The pure VRP results are included in the table as a theoretical lower bound, but they include invalid routes. Our method obtained route length reductions ranging up to 12.263 km, with an average reduction in total route length of 6.28 km. In addition, the improvement over traditional post-VRP detour increased with the complexity and number of NFZs in the environment.

Table 2. Comparative analysis of route lengths for 10 randomized experiments with varying PoI and NFZ configurations. The last column shows the total reduction of route lengths of our method over traditional post-VRP detour.

Number of PoIs	VRP ignoring NFZs (km)	Post-VRP detour (km)	Our NFZ-aware method (km)	Length reduction (km)
9	92.462	100.511	96.412	4.099
10	89.265	99.243	91.853	7.39
9	91.194	98.346	92.062	6.284
12	105.732	109.186	106.717	2.469
7	93.298	98.258	95.098	3.16
6	99.524	109.761	100.858	8.903
12	88.687	101.402	89.139	12.263
6	91.594	105.601	95.328	10.273
10	98.872	104.041	101.262	2.779
11	94.441	102.763	97.582	5.181

Next, we evaluate the performance of OR-Tools [29] and compare it to other existing well-known and widely-used exact and metaheuristic optimization methods. Gurobi (v9.5) [44] and CPLEX (v22.1) [45] were chosen as state-of-the-art MILP solvers. For the metaheuristic category, we selected PyGAD (v3.0) [46], representing a canonical Genetic Algorithm (GA), and Ant Colony Optimization (ACO) [47]. The comparison focuses on two key metrics: the computation time and makespan. All methods were executed on the same computer and solved the same set of randomized missions with different numbers of UAS and different numbers of PoIs distributed on an area with a radius of 10 km. For comparison fairness, NFZs and wind constraints were not considered in the mission definition. Accordingly, the methods' performance directly reflects the impact of problem size rather than environmental complexity. Each mission was solved by performing 10 independent runs (the results were averaged) to ensure statistical reliability of the results.

The results, in Table 3, show the suitability of OR-Tools as the main solver in U-Plan. The results show a clear trade-off between optimality and computational efficiency. While Gurobi and CPLEX obtained optimal solutions for small instances, their computation time degraded rapidly as the problem size increased, becoming computationally infeasible for moderate-size mission scales. In fact, in the computer used, they were not able to solve missions with $\geq$ 16 PoIs in a reasonable time, making them unsuitable for dynamic or time-critical contexts, such as mission replanning. Conversely, PyGAD and ACO provided feasible solutions for larger problems but exhibited longer computation times and slightly longer routes than OR-Tools. In contrast, OR-Tools obtained a good balance between computation time and routes lengths, yielding near-optimal results (comparable to MILP solvers) while maintaining sub-second runtimes even for moderate problem sizes. This computational efficiency makes it particularly advantageous for on-board or real-time mission replanning, and justifies the selection of OR-tools as the VRP solver in U-Plan.

 doi:10.20944/preprints202605.0937.v1

Table 3. Computation time and makespan for Gurobi [44], CPLEX [45], PyGAD [46], ACO [47], and OR-Tools [29] in randomized missions with different number of UAS and of PoIs.

Number of UAS	Number of PoIs	Computation time (s)					Makespan (s)				
		Gurobi	CPLEX	PyGAD	ACO	OR-Tools	Gurobi	CPLEX	PyGAD	ACO	OR-Tools
2	10	2.63	5.18	4.08	0.68	**0.03**	**926**	**926**	**926**	952	**926**
	12	5.69	17.8	5.29	1.18	**0.04**	**933**	**933**	988	968	**933**
	14	18.37	12.09	4.62	1.06	**0.05**	**963**	**963**	1032	1074	**963**
	16	170.74	520.2	5.35	1.37	**0.05**	**1074**	**1074**	1211	1163	1114
	20	-	-	5.54	1.64	**0.09**	-	-	1311	1200	**1144**
	40	-	-	5.46	4.63	**0.84**	-	-	2906	2027	**1484**
	100	-	-	19.65	24.08	**8.26**	-	-	6661	2901	**1932**
3	10	28.34	88.5	3.45	0.73	**0.04**	**921**	**921**	**921**	976	**921**
	12	327.37	975.02	4.16	0.97	**0.07**	**866**	**866**	**866**	954	**866**
	20	-	-	4.96	2.95	**0.11**	-	-	1271	1219	**1114**
4	10	96.96	253.05	4.02	0.88	**0.06**	**855**	**855**	**855**	906	**855**

5.2. Trajectory Smoothing

The proposed trajectory smoothing module is compared against three well-known methods: B-Spline interpolation, Dubins curves, and the Overfly logic implemented by many UAS autopilots. First, we show the resulting smoothed trajectories for one example route comprised of 6 waypoints (P1–P6) distributed in a plane, which segments have varied turn geometries. The UAS was assumed to have a minimum turning radius (R^v_{min}) of 200 meters, a ground speed of V^v_g=120 km/h, and a maximum roll angle of ϕ^v_{max}=29.6°.

The resulting smoothed trajectories obtained are depicted in Figure 5. As shown, the trajectory generated by B-Spline has continuous curvature but deviates significantly from the original route. This is a strong disadvantage in our case as the resulting smoothed trajectory can potentially enter NFZs. In addition, interpolation-based methods like Splines do not guarantee kinematic feasibility. While the Dubins method generates kinematically feasible paths, it relies on rigid geometric constructions that result in a non-uniform deviation around the vertices. In wide turns, the Dubins method exhibit similar performance to the proposed method, but in sharp angles, it fails to distribute the deviation uniformly, often skewing the trajectory significantly to one side of the vertex. The Overfly method, while effective for standard turns, exhibits critical limitations in wide cornering geometries, as e.g., vertex P5, where it fails to generate a valid intercept trajectory because the required geometry violates the method's convergence constraints. In contrast, our trajectory smoothing method successfully handles all vertex geometries. It distributes the deviation uniformly around the vertex, ensuring a kinematically feasible and safe turn even in sharp angles, confirming its better robustness and lower deviation than the other methods.

 doi:10.20944/preprints202605.0937.v1

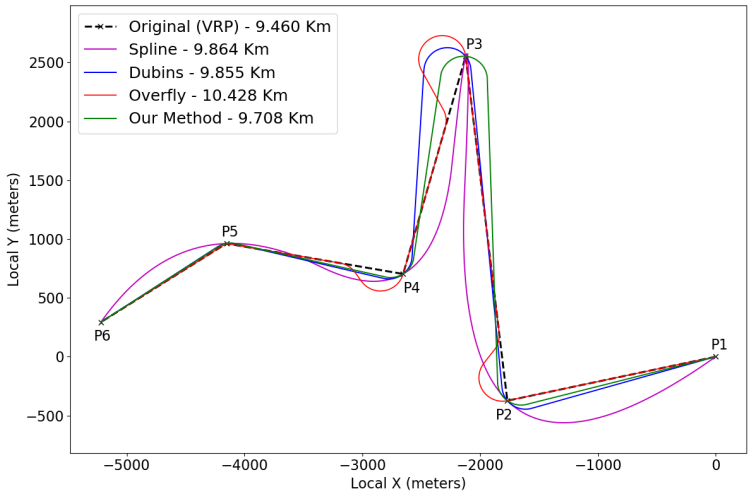


Figure 5. Comparison of smoothed trajectories obtained by B-Spline, Dubins, Overfly, and the proposed trajectory smoothing method in one example route.

In addition, the proposed method generated the shortest path of 9.708 Km. Overfly produced the longest path (10.428 Km), mainly due to the large overshoot required to recover the track after sharp turns. Spline and Dubins methods generated trajectories of 9.864 Km and 9.857 Km, respectively. As in this research, each UAS is commanded to maintain a constant ground speed throughout the mission, our method also obtained the lowest mission makespan.

This analysis was repeated with 300 randomized missions with different numbers of UAS and PoIs at random locations, confirming these results. The proposed smoothing method consistently outperformed the counterparts, obtaining in all tested cases the shortest total length. Table 4 summarizes the results for 10 of these missions. The table also includes (in the "Straight-line distance" column) the sum of the lengths of the segments between two consecutive waypoints in all routes (theoretical baseline for the minimum possible path), showing the very low smoothing overhead of the adopted method.

Table 4. Comparative analysis of the total path lengths for different smoothing methods across 10 randomized experiments.

Number of UAS	Number of PoIs	Straight-line length (km)	B-Spline (km)	Dubins (km)	Overfly (km)	Our smoothing method (km)
4	2	117.790	122.822	118.324	119.872	**118.266**
4	3	176.661	183.921	177.450	179.514	**177.360**
6	3	188.130	195.588	188.652	190.389	**188.595**
7	2	140.684	151.262	140.998	142.066	**140.934**
6	4	266.648	275.832	267.996	271.060	**267.672**
5	3	137.217	145.887	137.718	139.530	**137.625**
4	3	163.371	170.994	164.130	166.041	**163.959**
5	4	235.376	245.128	236.442	239.468	**236.312**
7	2	124.008	128.242	124.458	125.750	**124.398**
4	3	128.412	132.456	129.015	130.554	**128.856**

5.3. Full U-Plan

The complete U-Plan architecture integrated with VECTOR-SIL and Visionair GCS Software is validated in a realistic, large-scale scenario. The mission, set in the Andalusia region of Spain, consists in visiting 26 geographically distributed PoIs using 4 fixed-wing UAS, which start from defined depots and must finish at any one of 5 available landing sites (end depots) selected from a shared pool. To increase operational realism, the mission incorporates actual NFZs derived from the ENAIRE (primary air navigation manager in Spain) Drones Database [48], encompassing restricted areas and environmental protection zones. A constant wind field of 6.0 m/s blowing towards the East

 doi:10.20944/preprints202605.0937.v1

(90°) is applied to validate the energy-aware routing logic. While U-Plan supports spatially varying wind fields, a constant wind model was selected to clearly isolate the impact of wind direction on the energy consumption of routes with different orientations. The input parameters for the fleet and the environment are summarized in Table 5. All UAS share identical ground speed (V_g=130 km/h), maximum airspeed ($V_a|_{max}$=155 km/h), and initial energy level (3.6 MJ), and the arrival time deadline is 3.600 s for all PoIs.

Table 5. Input parameters for the full U-Plan validation scenario.

Parameter	Value	Unit
Number of PoIs	26	-
End depots (Pool)	5 (Huelva, Cadiz, Las Cabezas)	-
PoIs maximum arrival time (τ_{max})	3600	s
Number of UAS	4	-
Start depots	4 (Seville)	-
Ground speed (V_g)	130.0	km/h
Maximum airspeed ($V_a|_{max}$)	155.0	km/h
Initial energy (E_{ini})	3.6	MJ
Propulsive efficiency (η)	0.75	-
Ref. wing area (a)	0.5	m^2
Drag coefficient (c_d)	0.03	-
Air density (ρ)	1.2	kg/m^3
Wind intensity	6.0	m/s
Wind direction	90.0 (East)	deg

The trajectories resulting from U-Plan are depicted in Figure 6 and detailed in Table 6. All trajectories fulfill the UAS kinematic constraints and avoid the restricted airspace (NFZs), ensuring operational safety. The VRP solver leveraged the flexibility of the shared end-depot pool by selecting the 4 most efficient landing sites out of the 5 available options. The workload was distributed effectively among the fleet, with total travel times ranging from 47 to 48 minutes, resulting in a balanced distribution and low makespan. In terms of computation, **Multi-UAS Route Computation** required 2.73 seconds, and the total computation time devoted in **Trajectory Smoothing** and **Mission Supervision** was 2.41 seconds, confirming its feasibility for real-time operation.

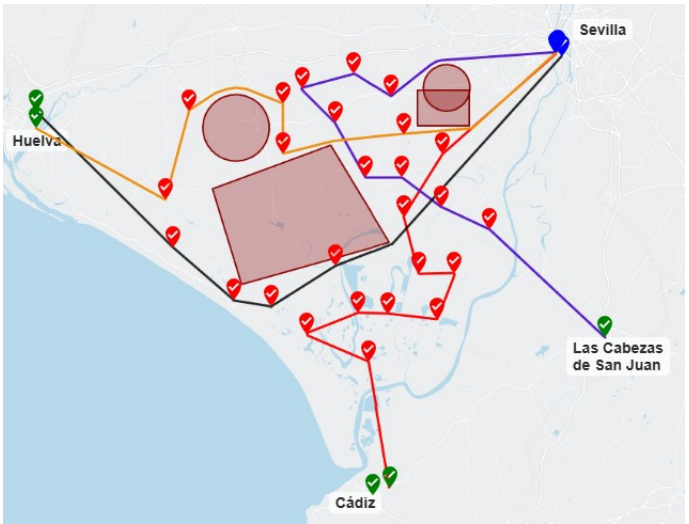


Figure 6. Trajectories computed by U-Plan for the considered mission. The trajectories for UAS 1 (Red), UAS 2 (Blue), UAS 3 (Green), and UAS 4 (Orange) successfully visit all 26 PoIs while avoiding real-world NFZs (shaded areas) and terminating at the designated shared end depots.

 doi:10.20944/preprints202605.0937.v1

Table 6. Summary of results of U-Plan for the multi-UAS mission shown in Figure 6.

UAS #	Number of PoIs	Length (km)		Mission time (s)	Energy (MJ)
		Straight routes	Smoothed trajectories		
1	9	105.03	105.49	48m 41s	1.75
2	8	101.50	102.58	47m 20s	1.5
3	4	104.71	105.28	48m 35s	2.16
4	5	103.63	104.22	48m 7s	2.18
Makespan				**48m 41s**	

It can be noticed in Table 6 that although the flight distances for all UAS are remarkably similar (ranging between 102.58 km and 105.49 km), the energy consumption shows a significant variance. For instance, UAS 2 consumes approximately 1.5 MJ, whereas UAS 4 consumes 2.18 MJ, a difference of nearly 45%. This discrepancy arises because the trajectory of UAS 4 involves segments flying predominantly Westward, incurring a high aerodynamic penalty ($V_a > V_g$), while UAS 2 benefits from tailwinds or crosswinds for most of its flight. Despite these variations, the planner successfully ensured that all vehicles remained well within safety energy margins, with the lowest remaining battery level at 39.4%. As shown, the proposed trajectory smoothing method introduces a maximum addition of the length of 1.08 km (from 101.50 km to 102.58 km), being capable of resolving kinematic constraints and ensuring smooth, flyable turns with very low deviation from the routes calculated by the VRP solver. Table 7 provides a breakdown of the predicted arrival time for every PoI. Each row corresponds to a specific UAS trajectory, listing the arrival times for each PoI visited by the UAS in sequential order. As shown, the arrival time constraint (3.600 s) is fulfilled for all PoIs, with the latest arrival time occurring at 2.290 s (UAS 4).

Table 7. Predicted arrival time for each PoI in the multi-UAS mission shown in Figure 6.

UAS #	Arrival times (s) to each PoI in sequential order								
1	629	937	1183	1331	1534	1742	1865	2098	2382
2	737	917	1143	1341	1603	1751	1957	2180	
3	1304	1624	1782	2120					
4	766	1273	1481	1907	2290				

Next, we evaluate the computational time of the total pipeline across randomized scenarios with increasing problem complexity (number of PoIs and fleet size). These tests utilized random PoI distribution within an area of 10 km radius, random NFZs and wind fields, and results averaged over 10 independent runs. Figure 7-a shows the results for the *Multi-UAS VRP Solver* module alone, while Figure 7-b presents the total computation time for the full U-Plan. The results show that although the computation time increases in missions with numbers of PoIs and UAS, the method exhibits good scalability. The growth in computation time remains moderate, ensuring real-time replanning capabilities for the fleet sizes and PoIs densities envisioned in typical multi-UAS missions.

 doi:10.20944/preprints202605.0937.v1

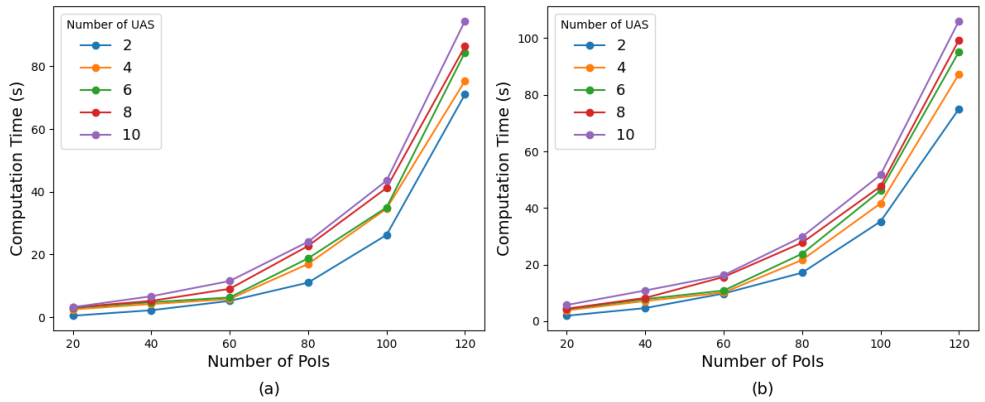


Figure 7. Scalability analysis showing the impact of fleet size and number of PoIs on computation time: (a) computational time of *Multi-UAS VRP Solver* alone. (b) total computational time of the complete U-Plan architecture.

Finally, we evaluate the effectiveness of the **Mission Supervision** through dynamic mission replanning. The mission involves 4 UAS assigned to visit 30 PoIs, operating with the same UAS specifications defined in Table 5. We simulate that UAS 4 has a critical failure at time $T = 1.250$ s. Figure 8 shows the mission state at the moment of the failure (left) and the subsequent replanned trajectories (right).

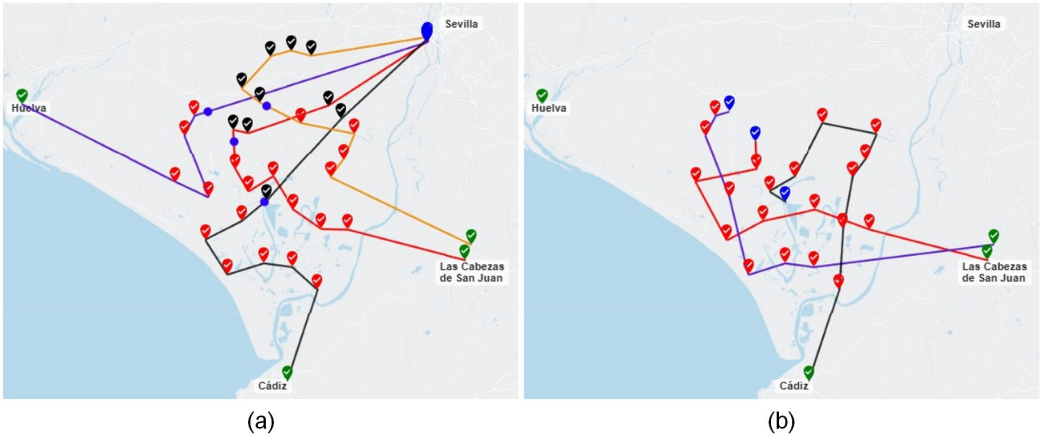


Figure 8. Dynamic replanning in response to a critical anomaly (UAS 4 failure at $T = 1.250$ s). (a) Snapshot of the mission state at the moment of failure: black markers indicate visited PoIs; blue dots represent the instantaneous positions of UAS 1, 2, and 3; red markers denote the remaining unvisited PoIs. (b) The replanned trajectories showing the redistribution of the remaining workload among the three UAS, ensuring mission completion despite the fleet reduction. In both figures the trajectories for UAS 1 are in red color, UAS 2 (blue), UAS 3 (green), and UAS 4 (orange).

After detecting the failure signal from UAS 4, the current state of the mission execution is immediately captured and the replanning mission is defined. Since the size of the problem is small, the replanning process is done without **Step1: Loiter and Hold**. The already visited PoIs, shown in black color in Figure 8-a, are removed from the target list. The remaining unvisited PoIs (marked in red), including those originally assigned to the failed UAS 4, form the PoI set for the new mission. The current positions of the surviving vehicles (UAS 1, 2, and 3), shown as blue dots in Figure 8-a, are designated as the new start depots for the replanning instance. The mission is then replanned to redistribute the pending workload among the three remaining UAS by executing **Multi-UAS Route Computation**, **Trajectory Smoothing**, and **Mission Validation**. As shown in Figure 8-b, U-Plan successfully generates valid trajectories that ensure that all remaining targets are visited fulfilling the mission, UAS, and scenario constraints. The total computation time for this replanning process was 1.64 seconds. This short response time confirms U-Plan's capability to handle emergency contingencies,

 doi:10.20944/preprints202605.0937.v1

seamlessly transitioning from a 4-UAS to a 3-UAS mission configuration without significant disruption of the operational continuity.

An interesting observation from the replanning results is the dynamic reassignment of end depots. In the replanned mission UAS 2 was redirected to land at an end depot different from that assigned in the initial plan. This shows the ability to leverage the shared pool of end depots dynamically, re-evaluating the end depot for each UAS based on its new starting position and updated workload rather than adhering to fixed pre-assignments. This flexibility is critical for ensuring efficient energy and mission management in real-world conditions, which are prone to frequent unexpected contingencies.

This analysis was repeated >100 randomized missions with different types of changes (e.g., new NFZs, new PoIs, addition of UAS, wind speed, among others), confirming the replanning capacity of U-Plan.

5.4. Experimental Validation with Professional Software-in-the-Loop Simulators

To validate the practical applicability of the generated flight plans, we conducted Software-in-the-Loop (SIL) validation campaigns using the Visionair GCS Software integrated with VECTOR-SIL simulators, products from UAV Navigation that are commercialized all over the world. This setup includes the actual autopilot hardware (flight control computer, Attitude and Heading Reference System (AHRS), and air data system) interfaced with a high-fidelity flight dynamics model, ensuring that the validation accounts for real-world avionics constraints, latencies, and communication protocols.

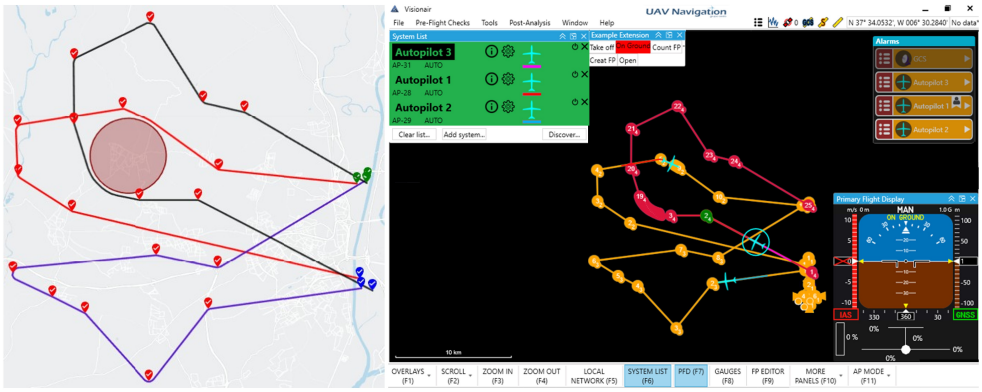
The objective of the experiments was to verify that the trajectories generated by U-Plan are accepted by the certified commercial autopilot and that the estimated flight time matched the telemetry data from the high-fidelity VECTOR-SIL simulator. We used three VECTOR-SIL devices, each simulating the navigation of a UAS following the trajectory assigned by U-Plan. They were connected to a host computer via Ethernet ports and ran simultaneously within the Visionair GCS Software environment, as shown in Figure 9. The integration between U-Plan and Visionair GCS Software was achieved via a custom SDK extension developed in C#. This interface allows U-Plan to export the calculated trajectories as flight plans, which are automatically detected, parsed, and loaded by Visionair GCS Software for execution on VECTOR-SIL.

Figure 9. Pictures of the SIL simulation environment provided by UAV Navigation, featuring the VECTOR-SIL and Visionair GCS Software.

The mission consisted in using a fleet of 3 UAS to visit 19 PoIs distributed to cover a broad geographical area, with start and end depots in the surroundings of the city of Seville. There was a moderate constant wind (3 m/s from East) in the considered area and a circular NFZ. The fleet comprised three heterogeneous UAS with distinct values of ground speed: UAS 1 ($V_g = 140$ km/h), UAS 2 ($V_g = 150$ km/h), and UAS 3 ($V_g = 145$ km/h), all with a maximum airspeed of 165 km/h. Figure 10-a presents the trajectories computed by U-Plan. The execution times of the flight plans computed by U-Plan for each UAS are: 1035 s, 1013 s, and 1037 s, respectively. Figure 10-b shows the

 doi:10.20944/preprints202605.0937.v1

trajectories resulting from the simulation of the flight plans computed using Visionair GCS Software and VECTOR-SIL, in which the UAVs completed their flight plans in 1033 s, 1008 s, and 1034 s, respectively.



(a) (b)

Figure 10. Results of trajectories computation in U-Plan and Visionair GCS Software. (a) Trajectories generated by U-Plan. UAS 1 (Red), UAS 2 (Blue), UAS 3 (Greeen). (b) Flight plan execution using Visionair GCS Software. The image shows the flight plans and their navigation by UAS in the Visionair GCS Software environment.

U-Plan provided high fidelity in the flight plan execution times w.r.t. those obtained using SIL realistic simulation. The maximum temporal deviation observed across all trajectories is lower than 5 seconds (UAS 2: 1013 s predicted versus 1008 s simulated), involving an error margin of less than 0.5%. This low error level was confirmed in all of the experiments with randomized missions (> 30 in total) that were performed. This close alignment confirms that the mathematical models used in U-Plan for time estimation are accurate representations of the real-world flight dynamics modeled by the SIL autopilot. It also validated that the flight plans generated by U-Plan are fully compatible with and executable by certified autopilots without requiring manual adjustment or intervention.

6. Conclusions and Future Work

This paper presented U-Plan, a comprehensive mission management framework designed for the effective and efficient coordination, planning, real-time monitoring, and replanning of heterogeneous multi-UAS systems in real-world operational conditions. Our approach successfully integrates mission optimization with physical and environmental constraints, providing a robust solution for planning safe and feasible flight paths.

The U-Plan framework is organized by three main components: 1) **Multi-UAS Route Computation**, which accounts for NFZs and wind speed within VRP to produce short NFZ-safe and wind-aware routes; 2) **Trajectory Smoothing**, which ensures the generation of kinematically-feasible trajectories for fixed-wing UAS while reducing the deviations w.r.t. VRP-generated piece-wise routes; and 3) **Mission Supervision**, which validates the flight plans to the UAS, real-time monitors the mission, detects deviations, and triggers replanning in case of changes in UAS, mission, wind speed, or airspace restrictions.

By encoding restricted airspace and NFZs into the initial cost matrix, we demonstrated that the VRP solver can identify more efficient visiting sequences, resulting in significant reduction in total route length compared to traditional post-processing detour methods. Furthermore, our wind-aware energy model explicitly accounts for the nonlinear impact of wind direction on aerodynamic drag and energy consumption, ensuring that flight plans remain energetically feasible. The kinematic-aware trajectory smoothing method consistently generates shorter, kinematically-feasible trajectories than established methods like B-Splines, Dubins curves, and the "Overfly" logic used in commercial autopilots, all while maintaining a lower spatial deviation from the optimized VRP route. Finally, the mission supervision

 doi:10.20944/preprints202605.0937.v1

provides the necessary reactivity for real-world operations, enabling the detection of anomalies and execute dynamic replanning.

U-Plan was implemented and validated integrated in a commercial setup comprising Visionair GCS Software integrated with the VECTOR-SIL Software-in-the-Loop fized-wing autopilot simulator from the company UAV Navigation. The validation of each individual component and comparison with main existing methods together with the validation of the full framework show significant efficacy, efficiency, scalability, and adaptability to dynamic changes in realistic complex multi-UAS missions.

As future work, although the validation using VECTOR-SIL and Visionair GCS provides high-fidelity real-world flight dynamics and includes aerodynamic disturbances, sensor noise, and communication latencies, conducting experimental flight campaigns would enable assessing the framework's robustness against actual noise and perturbations. In addition, while potential inter-UAS collisions are currently managed through the supervision module, future work will explore proactive spatiotemporal deconfliction directly within the trajectory smoothing phase through time-parameterized 4D collision avoidance mechanisms.

Author Contributions: Conceptualization, E.K. and J.R.M.; methodology, E.K. and J.R.M.; software, E.K.; validation, E.K., M.A.F. and J.R.M.; formal analysis, E.K. and J.R.M; investigation, E.K. and J.R.M.; resources, E.K.; data curation, E.K.; writing—original draft preparation, E.K.; writing—review and editing, J.R.M.; visualization, E.K.; supervision, J.R.M. and A.O; project administration, J.R.M.; funding acquisition, J.R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by grant PID2023-149683OB-I00, "RAISE: Robots Aéreos Inteligentes en Cooperación Estrecha con Sistemas IoT para la Inspección Avanzada de Viaductos", funded by MICIU/AEI/10.13039/501100011033 and ERDF, EU and by the 2024 FPI Program of the Spanish Ministerio de Ciencia, Innovación y Universidades (PREP2023-001631) in the subprogram Subprograma Estatal de Formación del Programa Estatal para Desarrollar, Atraer y Retener Talento, en el marco del Plan Estatal de Investigación Científica, Técnica y de Innovación 2021-2023. Partial funding has been obtained from UAV Navigation-Grupo Oesía SLU under contract "U-Plan: Arquitectura Jerárquica para la Planificación de UAS Heterogéneos" within the framework of project "U-SCUAR: Investigación Avanzada de UAS en el Ámbito de la Categoría Específica", funded by the CDTI through the 2022 Aeronautical Tech. Plan (PTA).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author, the data are not publicly available due to privacy restrictions.

Acknowledgments: The authors gratefully acknowledge Laura García-Junceda del Río from UAV Navigation for her technical assistance and the use of the company's facilities during the experimental validation with the VECTOR-SIL autopilot and Visionair GCS.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAS	Unmanned aerial System
NFZ	No-Fly Zone
VRP	Vehicle Routing Problem
MILP	Mixed-Integer Linear Programming
SA	Simulated Annealing
ACO	Ant Colony Optimization
GA	Genetic Algorithms
PoI	Points of Interest
DEM	Digital Elevation Model

 doi:10.20944/preprints202605.0937.v1

AMSL	Above Mean Sea Leve
GCS	Ground Control Station
RLOS	Radio Line-of-Sight

References

1. Abid, M.; El Kafhali, S.; Amzil, A.; Hanini, M. Optimization of UAV flight paths in multi-UAV networks for efficient data collection. *Arabian Journal for Science and Engineering* **2025**, *50*, 7207–7232.
2. Zammit, C.; Van Kampen, E.J. Comparison between A* and RRT algorithms for 3D UAV path planning. *Unmanned Systems* **2022**, *10*, 129–146.
3. Chen, C.M.; Lv, S.; Ning, J.; Wu, J.M.T. A genetic algorithm for the waitable time-varying multi-depot green vehicle routing problem. *Symmetry* **2023**, *15*, 124.
4. Ozkan, O.; Kilic, S. UAV routing by simulation-based optimization approaches for forest fire risk mitigation. *Annals of Operations Research* **2023**, *320*, 937–973.
5. Corberán, T.; Plana, I.; Sanchis, J.M.; Segura, P. The multidepot drone general routing problem with duration and capacity constraints. *International Transactions in Operational Research* **2025**, *32*, 3756–3779.
6. Al-kabi, H.; Mazinani, S.M. DNCS: New UAV navigation with considering the no-fly zone and efficient selection of the charging station. *Ain Shams Engineering Journal* **2021**, *12*, 3669–3676.
7. Stodola, P.; Kutěj, L. Multi-depot vehicle routing problem with drones: Mathematical formulation, solution algorithm and experiments. *Expert Systems with Applications* **2024**, *241*, 122483.
8. Euchi, J.; Sadok, A. Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. *Physical Communication* **2021**, *44*, 101236.
9. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Applied Soft Computing* **2021**, *112*, 107796.
10. Yu, X.; Jiang, N.; Wang, X.; Li, M. A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. *Expert Systems with Applications* **2023**, *215*, 119327.
11. Guo, J.; Gan, M.; Hu, K. Cooperative path planning for multi-UAVs with time-varying communication and energy consumption constraints. *Drones* **2024**, *8*, 654.
12. Liu, C.; Lu, Y.; Xie, F.; Ji, T.; Zheng, Y. Dynamic real-time multi-UAV cooperative mission planning method under multiple constraints. *Drones* **2026**, *10*, 132.
13. Jeong, H.Y.; Song, B.D.; Lee, S. Truck-drone hybrid delivery routing: Payload-energy dependency and No-Fly zones. *International Journal of Production Economics* **2019**, *214*, 220–233.
14. Ramasamy, S.; Reddinger, J.P.F.; Dotterweich, J.M.; Childers, M.A.; Bhounsule, P.A. Coordinated route planning of multiple fuel-constrained unmanned aerial systems with recharging on an unmanned ground vehicle for mission coverage. *Journal of Intelligent & Robotic Systems* **2022**, *106*, 30.
15. Wang, X.; Liu, Z.; Li, X. Optimal delivery route planning for a fleet of heterogeneous drones: A rescheduling-based genetic algorithm approach. *Computers & Industrial Engineering* **2023**, *179*, 109179.
16. Myers, D.; Batta, R.; Karwan, M. A real-time network approach for including obstacles and flight dynamics in UAV route planning. *The Journal of Defense Modeling and Simulation* **2016**, *13*, 291–306.
17. Sadeghi Esfahlani, S.; Simanjuntak, S.; Sanaei, A.; Fraess-Ehrfeld, A. Advanced Drone Routing and Scheduling for Emergency Medical Supply Chains in Essex. *Drones* **2025**, *9*, 664.
18. Mor, A.; Speranza, M.G. Vehicle routing problems over time: a survey. *Annals of Operations Research* **2022**, *314*, 255–275.
19. Adewumi, A.O.; Adeleke, O.J. A survey of recent advances in vehicle routing problems. *International Journal of System Assurance Engineering and Management* **2018**, *9*, 155–172.
20. Jung, S. MILP-based cost and time-competitive vehicle routing problem for last-mile delivery service using a swarm of UAVs and UGVs. *Journal of Air Transport Management* **2025**, *124*, 102736.
21. Poikonen, S.; Wang, X.; Golden, B. The vehicle routing problem with drones: Extended models and connections. *Networks* **2017**, *70*, 34–43.
22. Aydınalp, Z.; Özgen, D. Solving vehicle routing problem with time windows using metaheuristic approaches. *International Journal of Intelligent Computing and Cybernetics* **2023**, *16*, 121–138.
23. Rossit, D.G.; Toncovich, A.A.; Fermani, M. Routing in waste collection: A simulated annealing algorithm for an Argentinean case study **2021**.
24. Li, B.; Qi, X.; Yu, B.; Liu, L. Trajectory planning for UAV based on improved ACO algorithm. *IEEE Access* **2019**, *8*, 2995–3006.

 doi:10.20944/preprints202605.0937.v1

25. Omar, A.; Omar, Y.; Solayman, M.; Mansour, H. Comparative Analysis of Ant Colony Optimization and Google OR-Tools for Solving the Open Capacitated Vehicle Routing Problem in Logistics. In Proceedings of the 2025 Intelligent Methods, Systems, and Applications (IMSA). IEEE, 2025, pp. 552–557.
26. Zhou, N.F.; Kjellerstrand, H.; Fruhman, J. *Constraint Solving and Planning with Picat*; Springer, 2015.
27. Prud'homme, C.; Fages, J.G. Choco-solver: A Java library for constraint programming. *Journal of Open Source Software* **2022**, *7*, 4708. https://doi.org/10.21105/joss.04708.
28. Carlsson, M.; Mildner, P. SICStus Prolog—the first 25 years. *Theory and Practice of Logic Programming* **2012**, *12*, 35–66.
29. Furnon, V.; Perron, L. OR-Tools Routing Library. https://developers.google.com/optimization/routing/, 2025. Google, Version 9.12.
30. Liu, Y.Q.; Han, J.; Zhang, Y.; Li, Y.; Jiang, T. Multivisit drone-Vehicle Routing Problem with Simultaneous Pickup and Delivery Considering No-Fly Zones. *Discrete Dynamics in Nature and Society* **2023**, *2023*, 1183764.
31. Incekara, H.; Selek, M. Wind-Effected Dynamic Quadrotor Route Planning with Metaheuristic Methods in Different Weather Conditions. *Advances in Electrical & Computer Engineering* **2021**, *21*.
32. Ito, S.; Akaiwa, K.; Funabashi, Y.; Nishikawa, H.; Kong, X.; Taniguchi, I.; Tomiyama, H. Load and wind aware routing of delivery drones. *Drones* **2022**, *6*, 50.
33. Tran, T.B.; Kolmanovsky, I.; Biberstein, E.; Makke, O.; Tharayil, M.; Gusikhin, O. Effect of wind on electric vehicle energy consumption: Sensitivity analyses and implications for range estimation and optimal routing. *Journal on Autonomous Transportation Systems* **2024**, *1*, 1–31.
34. He, W.; Qi, X.; Liu, L. A novel hybrid particle swarm optimization for multi-UAV cooperate path planning. *Applied Intelligence* **2021**, *51*, 7350–7364.
35. Zhen, Z.; Xing, D.; Gao, C. Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm. *Aerospace Science and Technology* **2018**, *76*, 402–411.
36. UAV Navigation. Fixed-Wing Auto Mode: Overfly Configuration. https://www.uavnavigation.com/support/kb/autopilot-software/fixed-wing-software/fixed-wing-auto-mode. Accessed: 24 November 2025.
37. Open-Meteo Contributors. Open-Meteo Weather API. https://open-meteo.com, 2024. Accessed: 2026-01-XX.
38. Meteomatics AG. Meteomatics Weather API. https://www.meteomatics.com, 2024. Accessed: 2026-01-XX.
39. Song, B.; Wang, Z.; Zou, L. On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm. *Cognitive Computation* **2017**, *9*, 5–17.
40. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Hoshino, Y.; Peng, C.C. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors* **2018**, *18*, 3170.
41. Park, H.; Deyst, J.; How, J.P. A New Nonlinear Guidance Logic for Trajectory Tracking. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Providence, RI, USA, 2004.
42. Chee, K.; Zhong, Z. Control, navigation and collision avoidance for an unmanned aerial vehicle. *Sensors and Actuators A: Physical* **2013**, *190*, 66–76.
43. UAV Navigation-Grupo Oesía. Company Profile. https://www.uavnavigation.com/company, 2024. Accessed: 2026-01-28.
44. Gurobi Optimization, LLC. Gurobi Optimizer. https://www.gurobi.com, 2025. Accessed: 2025-11-08.
45. IBM. IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio, 2025. Accessed: 2025-11-08.
46. Gad, A.F. PyGAD: An Intuitive Genetic Algorithm Python Library. *Multimedia Tools and Applications* **2021**, *80*, 29207–29220.
47. Dorigo, M.; Stützle, T. *Ant Colony Optimization*; MIT press, 2004.
48. ENAIRE. ENAIRE Drones: Mapa interactivo de zonas geográficas UAS en España. https://drones.enaire.es, 2025. Accessed: 16 January 2026.