AIRSHIP

Funded by the European Union's
Horizon Europe programme under
the grant agreement No 101096487

D7.1 Report on GNC system and simulations

WP7 Mission avionics

Disclaimer

The content of this deliverable reflects only the author's view. The European Commission is not responsible for any use that may be made of the information it contains.

Document Information

Grant Agreement / Proposal ID	101096487
Project Title	AUTONOMOUS FLYING SHIPS FOR INTER-ISLAND AND INLAND WATERS TRANSPORT
Project Acronym	AIRSHIP
Project Coordinator	Universidad Politécnica de Madrid
Project starting date (duration)	1st January 2023 (48 months)
Related Work Package	WP7
Related Task(s)	T7.1
Lead Organisation	Luxemburg University
Contributing Partner(s)	LU, UPM, INESC TEC
Due Date	30/06/2026
Submission Date	30/06/2026
Dissemination Level	PU

Revision History

Revision	Revision Date	Name	Position
1.0	17/06/2026	Pedro Lima, Matteo El Hariry (LU)	Researcher
2.0	25/06/2026	Zorana Milosevic, Sergio Dominguez (UPM)	Researcher

TABLE OF CONTENTS

1 Introduction 7
1.1 Purpose of the Deliverable . . . 7
1.2 Document Structure . . . 7

2 Guidance, Navigation, Control (GNC) system 9

3 Path planning 10
3.1 Global path planning . . . 10
3.2 Local path planning . . . 12

4 Waypoint guidance 16
4.1 Waypoint acceptance criterion . . . 16
4.2 Experimental validation . . . 17
4.2.1 Results and discussion . . . 19

5 Learning-Based and Closed-Form Guidance 21
5.1 Guidance architecture . . . 21
5.2 Coordinated turns and the turn-radius trade . . . 21
5.3 Waypoint following . . . 22

6 Conclusion and Future Work 26

References 27

List of Tables

Table 1 Abbreviation and Acronyms . . . 5

List of Figures

Figure 1 a) Elevation map of Noja beach in Spain (Map obtained via (cnig, 2018)). b) Elevation map of a lake in Mänttä in Finland (Map obtained via (NLS, 2024)). . . . 11
Figure 2 Example of a global path in Tampere (Map obtained via (NLS, 2024)) . . . 11
Figure 3 Case of path avoiding an area of high waves (presented in pink) in Noja beach (The proposed area of high waves is created and not obtained from real data). . . . 12
Figure 4 Local planner finding the shortest path (in red) to go back to the original path (in blue) in the case of the lake in Mänttä. . . . 13
Figure 5 Calculation of the offset required to safely fly over an obstacle (represented in yellow). . . . 13
Figure 6 Calculation of the offset required to safely manoeuvre around an obstacle (represented in yellow). . . . 14
Figure 7 Example of local obstacle avoidance. The original trajectory is shown in blue, the replanned trajectory in red, and the enlarged obstacle regions in yellow. . . . 14
Figure 8 Overall waypoint guidance flowchart. . . . 16
Figure 9 Square 2D waypoint mission. . . . 18
Figure 10 Figure-eight 2D waypoint mission. . . . 18
Figure 11 3D low altitude waypoint mission (top view). . . . 19

Figure 12 3D low altitude waypoint mission. The axes represent latitude and longitude of the waypoints. The mission starts at GPS coordinate $(40.40°, -18.40°)$, a region of the North Atlantic Ocean. The WIG craft is tasked to simultaneously control position and height. 19
Figure 13 Coordinated-turn tracking on the A0-S: one episode of eight alternating $\pm 10°$ bank commands (top-left, commanded vs. achieved bank), the altitude held during the turns (bottom-left), and the per-turn settled bank-error distribution over 512 episodes (right). 22
Figure 14 Coordinated-turn robustness and accuracy versus commanded bank on the two airframes. Bank angle sets the turn radius; the slow A0-S turns tightly but loses robustness past $\sim 13°$, while the fast V7 turns on wide radii with very steady tracking. 23
Figure 15 Single-waypoint guidance on the A0-S (gate: ≤ 10 m horizontal, ≤ 4 m vertical). Left: example ground tracks (colour encodes altitude). Centre: the capture funnel against the arrival gate (no run fails catastrophically). Right: arrival rate vs. leg distance, $\sim 84\%$ on short legs decaying gracefully to a $\sim 36\%$ aggregate. 24
Figure 16 Control-quality limits on the V7. Left: turn robustness versus turn duration (tracking stays tight; only rare late divergence). Centre: waypoint arrival versus capture gate (fail $= 0$ throughout, i.e. precision-limited). Right: loosening the vertical gate recovers arrival far more than the horizontal one — altitude lag is the binding constraint. 24
Figure 17 Waypoint-sequence guidance over distant legs (40–100 m, 10 m capture radius) on the two airframes. Each panel is one rollout; the policy re-anchors the leg on each capture and chains as far as the heading drift allows. Well-aligned sequences complete $(4/4)$; the harder chains time out on the long legs. 25

Abbreviations and Acronyms

Acronym	Description
CFD	Computational Fluid Dynamics
DoF	Degrees-of-freedom
GNC	Guidance, Navigation, and Control
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
L1	Lookahead (L1) nonlinear guidance
MAE	Mean Absolute Error
MaxErr	Maximum Error
OMPL	Open Motion Planning Library
PID	Proportional–Integral–Derivative controller
RL	Reinforcement Learning
RMSE	Root Mean Square Error
ROS	Robot Operating System
UWV	Unmanned WIG Vehicle
WIG	Wing-in-ground

Table 1: Abbreviation and Acronyms

Executive Summary

This deliverable reports the work carried out under Task 7.1 — Guidance and Navigation — of Work Package 7 (Mission Avionics), spanning M6–M42 of the AIRSHIP project. Task 7.1 develops the higher-level motion control that sits above the low-level actuator control of Work Package 5: given an estimate of the vehicle's state and the active mission, the guidance layer decides where the wing-in-ground (WIG) craft should go and emits the setpoints that the inner control loop tracks.

The work builds directly on the dynamic modelling of Task 5.1 and the control development of Tasks 5.2 and 5.3, and is written so as not to repeat that material: the controllers are treated as a fixed, validated inner loop, and only the guidance layer above them is described. The deliverable also reflects the interdependence between WP5, WP7 and the vehicle design activities of WP2. Building accurate simulations, control algorithms and validation mechanisms depends strongly on the evolving vehicle design, which went through many iterations; in particular, the impossibility of obtaining a complete aerodynamic model for vehicles still being designed and built (intractable CFD, incomplete wind-tunnel coverage) led the partners responsible for the GNC (UPM and the University of Luxembourg) to develop a manifold of alternative solutions, and a robust software infrastructure that enables thorough testing and simplified modelling of fixed-wing and WIG vehicles.

The defining constraint of the application is speed: a WIG craft cruises fast and close to the surface, so the guidance layer must react within a tight compute budget on embedded flight hardware. Because the over-water environment is obstacle-sparse and tolerates loose tracking tolerances, the chosen strategy trades a small amount of precision for guidance laws that are either closed-form (constant-time) or learned (a single forward pass), avoiding heavy iterative planners. The work carried out in parallel by UPM and the University of Luxembourg has some overlap in scope but complementary functionality, arising from the similar nature of the assigned tasks and the different domain expertise of each group; both approaches are reported here rather than forcing a single workflow.

Concretely, the deliverable presents a global/local path-planning pipeline that computes cost-optimal, ground-effect-preferring paths over an elevation map and re-plans around deviations and obstacles; a waypoint-guidance and path-following law that autonomously completes 2-D and 3-D waypoint missions while holding low altitude above the water; and a complementary set of learning-based and closed-form guidance laws — coordinated turns and a learned waypoint policy — evaluated on two airframes of different scale to illustrate the versatility of the framework. Together these form the waypoint, trajectory and path-following foundation on which the remaining guidance and navigation activities, and the validation tasks T7.2 and T7.3, build.

1 Introduction

The AIRSHIP project develops an environmentally friendly wing-in-ground (WIG) effect vehicle for autonomous transport over inter-island and inland waters. Once a vehicle can be stabilised and commanded at the actuator level, the next capability it needs is *guidance and navigation*: the higher-level motion control that decides where the craft should go and produces the setpoints the low-level controllers track. This deliverable reports the work of Task 7.1 — Guidance and Navigation — carried out within Work Package 7 (Mission Avionics) over the period M6–M42, led by the University of Luxembourg with the participation of UPM and INESC TEC.

A WIG craft imposes an unusual combination of constraints on its guidance layer. It cruises fast and very close to the water, so the guidance must produce updated commands within a tight time budget on an embedded flight computer, and it must do so while respecting the altitude and bank limits that keep the vehicle safely inside ground effect. At the same time, the operating environment — obstacle-sparse open and inland water — is comparatively benign and tolerates loose tracking tolerances. This combination motivates guidance laws that deliberately trade a small amount of tracking precision for speed and low computational cost: closed-form (constant-time) laws and learned (single forward-pass) policies, in preference to heavy iterative planners.

The work reported here builds directly on the dynamic modelling of Task 5.1 and the low-level control of Tasks 5.2 and 5.3, and it is deliberately written to avoid repeating that material: the controllers are treated as a fixed inner loop, and only the guidance layer that sits above them is described. A second feature of the task is that two partners (UPM and the University of Luxembourg) addressed overlapping guidance objectives with complementary methods — a classical planning-and-PID pipeline and a learning-based/closed-form approach developed in a custom simulation environment. Rather than forcing a single workflow, the deliverable presents both, since the nature of learning-based development in a bespoke environment does not always allow the same analysis pipeline to be reused.

1.1 Purpose of the Deliverable

The purpose of this deliverable is to report the guidance and navigation capabilities developed for the AIRSHIP WIG vehicle under Task 7.1, focused on activity (a) of the task description: *waypoint and trajectory planning and path following*. It documents the global and local path-planning pipeline, the waypoint-guidance and path-following laws, and their validation on the project's 6-DoF simulation environment and high-fidelity flight simulators. The guidance layer emits setpoints — a planned path, an ordered set of waypoints, or a bank/turn-rate command — that the inner control loop of Tasks 5.2/5.3 tracks; the same setpoint interface is used in simulation and on the deployed autopilot, so a guidance law validated in the digital twin transfers to the vehicle with no change. The work provides the guidance foundation on which the remaining Task 7.1 activities (basic manoeuvres) and the validation tasks T7.2 and T7.3 build.

1.2 Document Structure

This document is organised as follows:

- Section 2 introduces the overall Guidance, Navigation and Control (GNC) architecture and the role of the guidance layer within it.

- Section 3 presents the path-planning pipeline: the global planner that computes a cost-optimal path over an elevation map, and the local planner that re-plans around deviations and obstacles.
- Section 4 describes the waypoint-guidance and path-following architecture, the Haversine-based waypoint logic, and the 2-D and 3-D tracking results with their performance metrics.
- Section 5 reports a complementary family of learning-based and closed-form guidance laws — coordinated turns and a learned waypoint policy — evaluated on the A0-S and Airship_V7 airframes.
- Section 6 summarises the main achievements, lessons learned, and planned next steps.

2 Guidance, Navigation, Control (GNC) system

The autonomy of the AIRSHIP vehicle is organised as a layered Guidance, Navigation and Control (GNC) architecture, in which each layer operates at a different rate and level of abstraction and feeds the layer below it.

Navigation estimates the vehicle's state — position, velocity, attitude and height above the surface — by fusing the onboard sensors (GNSS, IMU, airspeed and altimeter) into a single consistent estimate. This state is the common input consumed by both the guidance and the control layers, and its quality bounds what the upper layers can achieve.

Guidance is the subject of this deliverable. Given the estimated state, the active mission and the surrounding environment, it decides *where the vehicle should go* and emits the corresponding setpoints: a planned path over the operating area, an ordered list of waypoints to capture, or an instantaneous bank/turn-rate and altitude command. Guidance runs as the slow outer loop of a two-rate cascade, which keeps it lightweight and portable: the same guidance law can sit on top of an in-house controller in simulation or on the vehicle autopilot's attitude loop on hardware.

Control is the fast inner loop developed under Tasks 5.2 and 5.3. It takes the guidance setpoints and computes the actuator commands (elevator, ailerons, rudder and throttle) that realise them, while rejecting disturbances and respecting the actuator and flight-envelope limits. Because the control layer is treated here as a fixed, validated inner loop, this deliverable focuses exclusively on the guidance layer and its interface to control.

This separation of concerns is what allows the guidance methods presented in the following sections — a global/local path planner, a waypoint-following law, and a complementary set of learning-based and closed-form laws — to be developed, swapped and validated independently, while reusing the same navigation estimate below them and the same control loop beneath that.

3 Path planning

Autonomous navigation of the AIRSHIP vehicle requires the capability to generate safe and dynamically feasible trajectories between the current vehicle position and a desired destination while accounting for the operational environment and the vehicle's flight constraints. To achieve this, a hierarchical path planning architecture has been developed, consisting of two complementary modules: a Global Path Planner and a Local Path Planner.

The global planner is responsible for generating the mission-level trajectory before or during the mission using prior information about the environment, including terrain elevation, restricted areas, and known obstacles. The local planner continuously monitors the planned trajectory together with the current state of the vehicle and information received from the perception and situational awareness systems. Whenever necessary, it locally modifies the reference path to compensate for deviations from the nominal trajectory or to safely avoid newly detected obstacles while preserving the overall mission objective.

Both planners have been developed as ROS 2 Humble nodes and communicate with the rest of the AIRSHIP software architecture using standard ROS 2 interfaces. Path planning is based on the *Open Motion Planning Library* (Guo et al., 2026) (OMPL), along with the *grid_map library* (Fankhauser & Hutter, 2016) and the *grid_map_geo library* (Lim, 2022), which provide efficient map representation and geographic coordinate handling. All calculated paths, both from the global and the local planner, follow Dubins curves (Owen, Beard, & McLain, 2015), using the minimum turning radius and the maximum pitch the craft is capable of.

The planning architecture is distributed between the ground station and the onboard companion computer. The global planner executes on the ground station, where computational resources and mission information are readily available, and generates the initial mission trajectory before take-off. It may also be executed again whenever significant changes occur, such as the appearance of a new restricted area or a modification of the mission objectives. The resulting trajectory, together with the corresponding map information, is transmitted to the onboard local planner. During flight, the local planner continuously evaluates the current vehicle state and the surrounding environment, adapting the trajectory whenever necessary to ensure safe and reliable navigation under changing operating conditions.

3.1 Global path planning

The global path planner is responsible for generating an initial collision-free trajectory between the mission start and destination before the flight begins. Since it has access to prior information about the operating environment, the planner considers static terrain, elevation data, restricted areas, and any other known constraints when computing the optimal route. The environment is represented using digital elevation maps derived from GeoTIFF files. Besides the terrain elevation, these files also contain the corresponding EPSG reference system, allowing the planner to accurately transform between global geographical coordinates and the local coordinate frame used internally by the planning algorithms. Figure 1 illustrates two examples of elevation maps used: Noja beach in Spain (a) and Lake Mänttä in Finland (b).

To guarantee safe operation, the original elevation map is extended by generating two additional layers defining the minimum and maximum allowable flight altitudes. These layers define the navigable volume available to the planner while ensuring adequate clearance from the terrain.

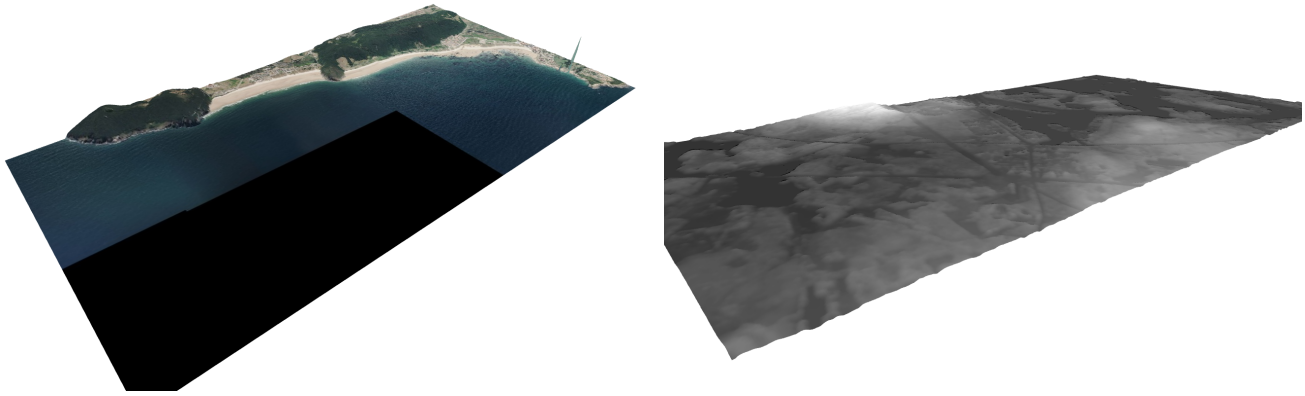


Figure 1: a) Elevation map of Noja beach in Spain (Map obtained via (cnig, 2018)). b) Elevation map of a lake in Mänttä in Finland (Map obtained via (NLS, 2024)).

The global trajectory is generated using OMPL while minimizing a cost function that combines travelled distance and flight altitude. Since Wing-in-Ground vehicles achieve their highest efficiency when operating close to the water surface, the planner penalizes unnecessary altitude changes. Consequently, higher flight altitudes are only selected when required to safely avoid terrain or other constraints. In Figure 2, an example of a path obtained by the global planner in a lake in Tampere is shown.



Figure 2: Example of a global path in Tampere (Map obtained via (NLS, 2024))

The planning framework has also been extended to incorporate environmental information, such as wave or swell maps. These environmental layers can be integrated into the planning cost function, allowing the planner to avoid areas where operating conditions would be unfavorable, for example, high waves. Figure 3 presents an example in which regions with high wave intensity receive a higher traversal cost, resulting in a safer trajectory.

After a global trajectory has been computed, the planner transmits the following information to the onboard local planner:

- the processed elevation map together with its additional safety layers;
- the EPSG code for local to global transformations;
- The planned path expressed in global coordinates.



Figure 3: Case of path avoiding an area of high waves (presented in pink) in Noja beach (The proposed area of high waves is created and not obtained from real data).

3.2 Local path planning

While the global planner computes the nominal mission trajectory, the local planner is responsible for adapting this trajectory during flight according to the current state of the vehicle and changes in the surrounding environment. The local planner continuously receives the global trajectory together with the latest navigation estimates and obstacle information generated by the perception and situational awareness modules. Whenever the nominal trajectory can no longer be safely followed, a new local trajectory is generated while preserving the original mission objective.

Two situations require replanning.

Trajectory recovery The first occurs when the vehicle deviates significantly from the planned trajectory due to disturbances, navigation errors, or external influences. In this case, the planner computes the shortest dynamically feasible path from the current vehicle position back to the nominal trajectory.

The closest point on the original trajectory is first identified using Dubins geometry, after which a new trajectory segment is generated while considering the terrain map and all currently known obstacles. Once the vehicle rejoins the original trajectory, normal path following resumes. An example is shown in Figure 4.

Obstacle avoidance The second replanning scenario occurs when previously unknown obstacles intersect the planned trajectory. In this case, the local planner identifies the sections of the original path that collide with the detected obstacles and computes new trajectory segments that safely reconnect the unaffected portions of the original path while avoiding the obstacles.

To guarantee that a feasible avoidance manoeuvre can always be generated, replanning is not performed using the detected obstacle geometry directly. Instead, each obstacle is enlarged by an appropriate safety margin before collision checking. This expansion ensures that sufficient clearance exists for the vehicle to execute the avoidance manoeuvre while respecting its kinematic constraints. Without this safety offset, situations could arise where an obstacle is detected too late or too close to the planned trajectory, making it impossible to compute a dynamically feasible avoidance path.

The required offset can be computed in two different ways, depending on whether the preferred avoidance strategy is to fly over the obstacle or to manoeuvre around it.

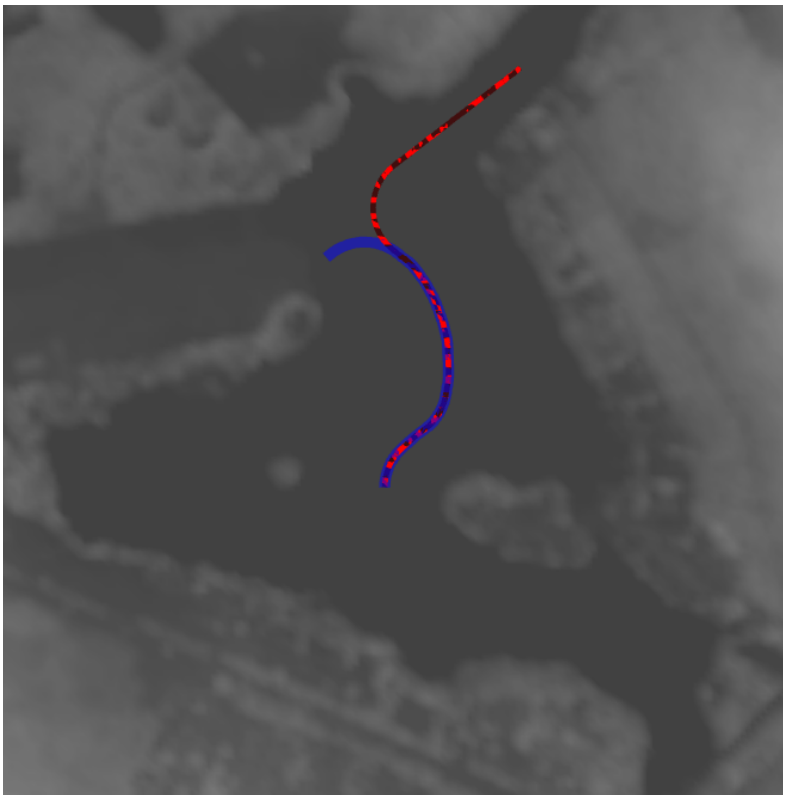


Figure 4: Local planner finding the shortest path (in red) to go back to the original path (in blue) in the case of the lake in Mänttä.

For vertical avoidance, the minimum horizontal distance required to safely overfly the obstacle is computed according to Figure 5, as:

$$Offset = \frac{H + d}{\tan \theta} \tag{1}$$

where (θ) is the maximum climb angle, H is the obstacle height, and d is the required safety clearance.

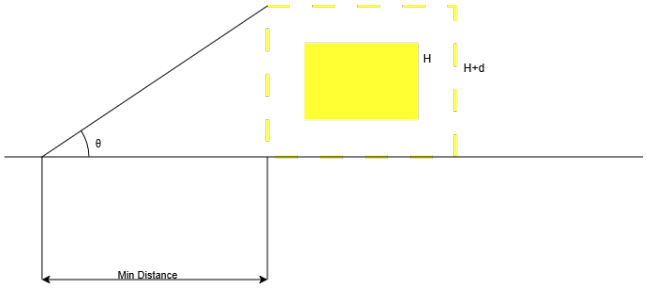


Figure 5: Calculation of the offset required to safely fly over an obstacle (represented in yellow).

Alternatively, the planner may generate a lateral avoidance manoeuvre. In this case, the required offset depends on the minimum turning radius of the vehicle. Following the geometry illustrated in Figure 6, the offset is computed as:

$$Ox = -R \cdot \cos \arcsin \frac{L2 + d}{2} + \frac{L1 + d}{2} \tag{2}$$

$$\gamma = \arccos \frac{R - Ox}{2 \cdot R} \tag{3}$$

$$Offset = 2 \cdot R \cdot \sin \gamma \tag{4}$$

where R is the minimum turning radius of the vehicle.

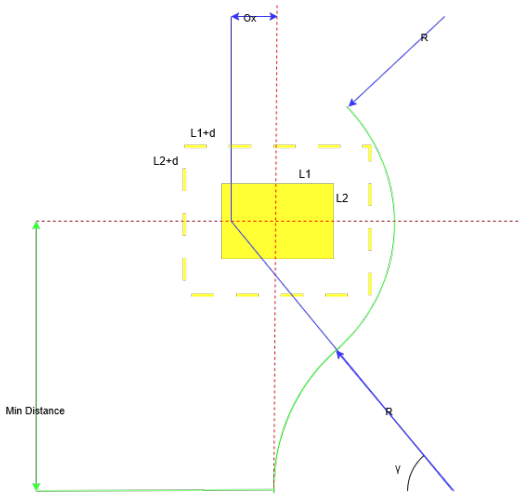


Figure 6: Calculation of the offset required to safely manoeuvre around an obstacle (represented in yellow).

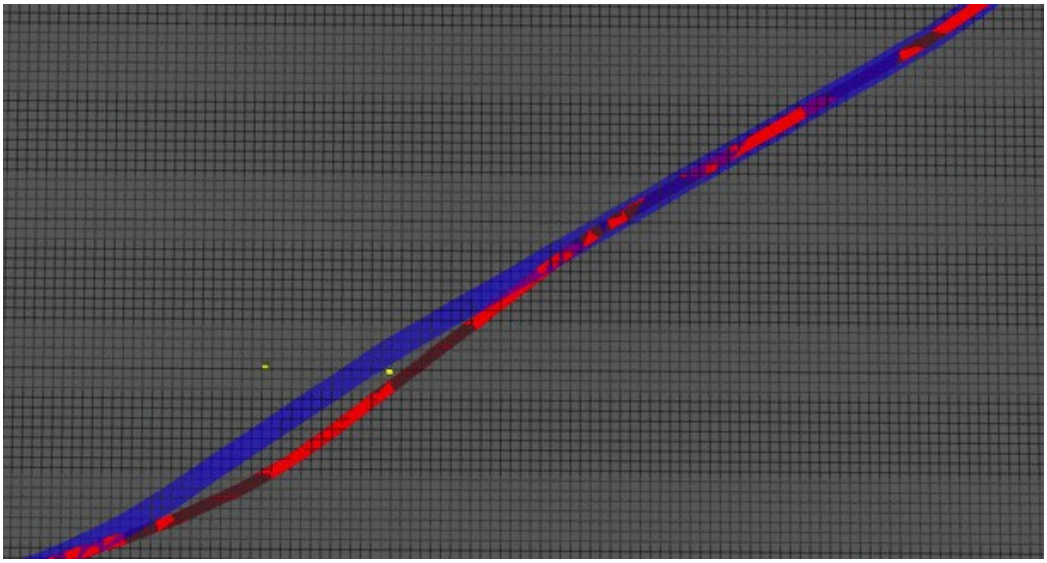


Figure 7: Example of local obstacle avoidance. The original trajectory is shown in blue, the replanned trajectory in red, and the enlarged obstacle regions in yellow.

The larger of the two computed offsets is selected and used to enlarge the obstacle boundaries before replanning. These expanded obstacles are then employed to identify the portions of the nominal trajectory that require modification. New path segments are subsequently generated to safely bypass the obstacles while preserving the overall mission trajectory and respecting the dynamic constraints of the WIG vehicle.

An example of the resulting obstacle avoidance behaviour is shown in Fig. 7. The original trajectory (blue) is locally modified (red) to safely avoid the enlarged obstacle regions (yellow).

4 Waypoint guidance

The waypoint guidance module is responsible for transforming the global trajectory generated by the path planning system into reference commands that can be executed by the low-level flight controller. While the global and local planners determine the optimal path from the current position to the mission objective while considering terrain, restricted areas, and obstacles, the waypoint guidance ensures that the vehicle progresses smoothly through the sequence of waypoints that define the planned route.

The waypoint guidance implementation developed within AIRSHIP builds upon the low-level control architecture presented in Deliverable D5.2. The guidance module generates position, heading, and altitude references, while the flight controller is responsible for stabilizing the vehicle and tracking these references. This separation between guidance and control enables different control strategies to be employed without modifying the navigation layer and provides a modular software architecture suitable for both simulation and real-flight deployment.

Figure 8 presents the overall waypoint guidance logic implemented in the AIRSHIP guidance system. The guidance module continuously receives the vehicle state from the navigation system together with the current mission plan. Based on the active waypoint, it computes the desired heading and altitude references and transmits them to the flight controller. Once the current waypoint has been reached, the next waypoint in the mission sequence becomes active until the final destination is reached.

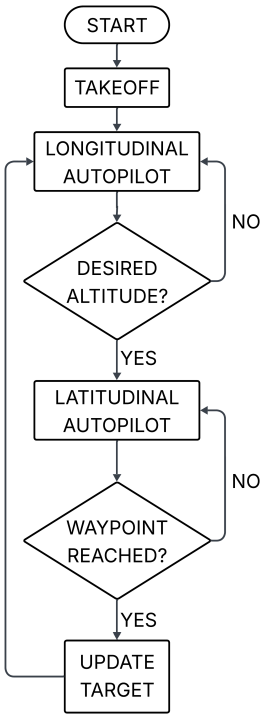


Figure 8: Overall waypoint guidance flowchart.

4.1 Waypoint acceptance criterion

The waypoint guidance module determines whether the active waypoint has been reached by computing the great-circle distance between the current vehicle position and the target waypoint using

the Haversine formulation (Robusto, 1957). Since the mission is represented by GPS coordinates, the Haversine equation provides an efficient and sufficiently accurate approximation of the shortest distance between two points on the Earth's surface while remaining computationally inexpensive.

To calculate the great-circle distance, the Haversine formula uses an intermediate value, a, representing the squared half-chord length between the points and accounting for the latitude and longitude differences. The great-circle distance d is then computed by applying the inverse sine function to the square root of a, scaled by the Earth's radius.

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{5}$$

$$d = 2r \arcsin\left(\sqrt{a}\right) \tag{6}$$

$$\Delta\varphi = \varphi_2 - \varphi_1 \tag{7}$$

$$\Delta\lambda = \lambda_2 - \lambda_1 \tag{8}$$

where φ_1, φ_2 are the latitudes of the two points (in radians), λ_1, λ_2 are the longitudes of the two points (in radians). r is the radius of Earth ($\approx$ 6371 km) and d is the great-circle distance between the two given points.

A waypoint is considered reached when the calculated distance falls below a predefined acceptance radius. In the current implementation, a threshold of 100 m is employed. Once this condition is satisfied, the guidance module automatically activates the next waypoint in the mission sequence and updates the reference commands transmitted to the low-level flight controller.

4.2 Experimental validation

The waypoint guidance was validated using the Software-in-the-Loop simulation environment described in Deliverable D5.3. Three representative mission scenarios were considered to evaluate the guidance system under increasing levels of complexity:

- a square waypoint mission,
- a figure-eight trajectory,
- a three-dimensional waypoint mission with varying altitude.

The objective of these experiments was to verify the ability of the guidance module to correctly manage waypoint transitions and generate appropriate reference commands. The evaluation of the low-level controller responsible for tracking these references has already been presented in Deliverable D5.2 and is therefore not repeated here.

Square 2D waypoint mission

The first experiment consisted of a simple square trajectory defined by four waypoints. Starting from the initial position, the guidance system successfully activated each waypoint in sequence and generated smooth heading references throughout the mission. The resulting trajectory closely follows the planned path, demonstrating stable waypoint transitions and reliable navigation.

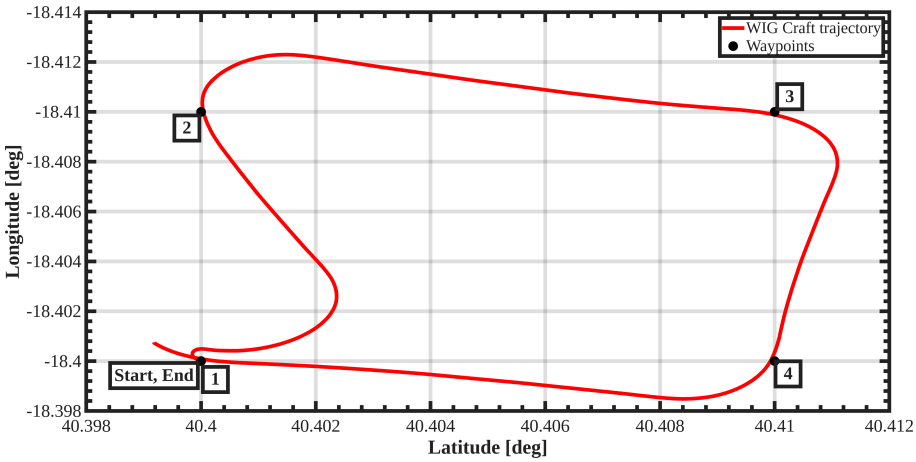


Figure 9: Square 2D waypoint mission.

Figure-eight 2D mission

The second experiment increased the trajectory complexity by defining eight waypoints arranged in a figure-eight pattern. Continuous heading changes and successive waypoint transitions were successfully managed by the guidance module without introducing oscillatory behaviour or unstable manoeuvres. The resulting trajectory demonstrates the capability of the proposed guidance algorithm to execute more demanding missions while maintaining smooth path following.

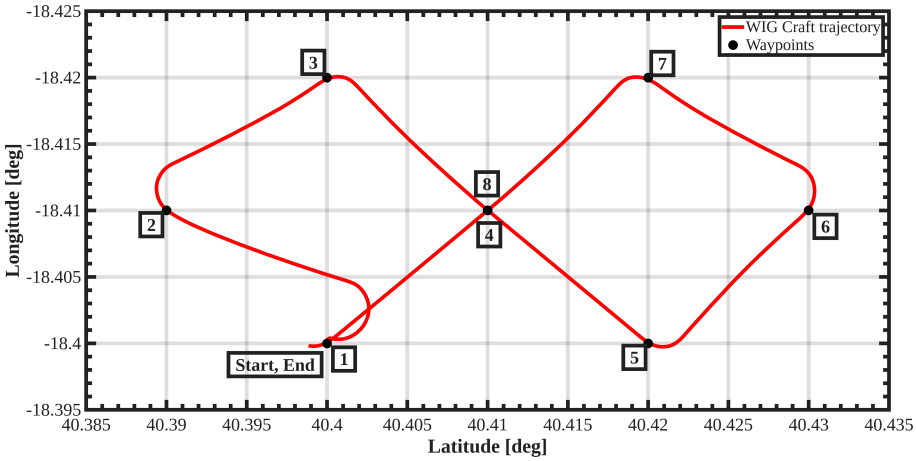


Figure 10: Figure-eight 2D waypoint mission.

3D low altitude mission

The final experiment extended waypoint guidance to three dimensions by assigning altitude references to each waypoint. The resulting mission required simultaneous horizontal navigation and vertical guidance while maintaining stable flight. The generated trajectory confirms that the guidance module successfully manages both horizontal and vertical reference generation, enabling smooth altitude transitions together with accurate waypoint progression.

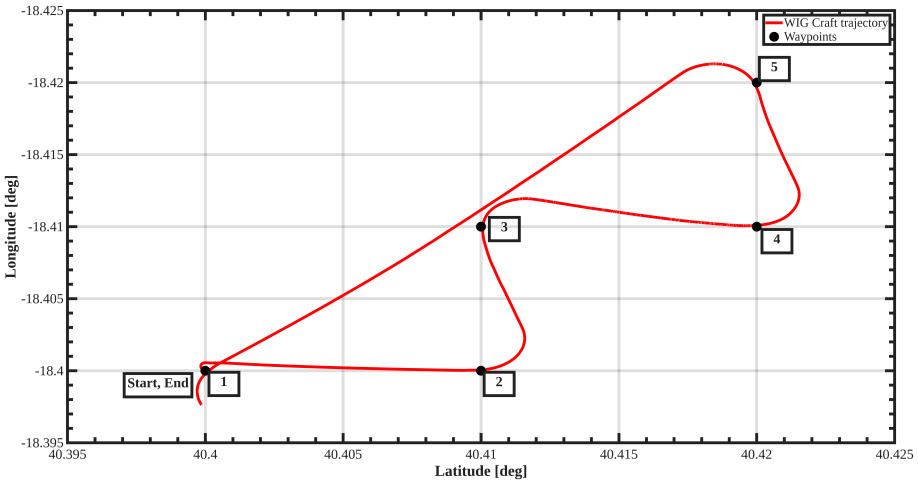


Figure 11: 3D low altitude waypoint mission (top view).

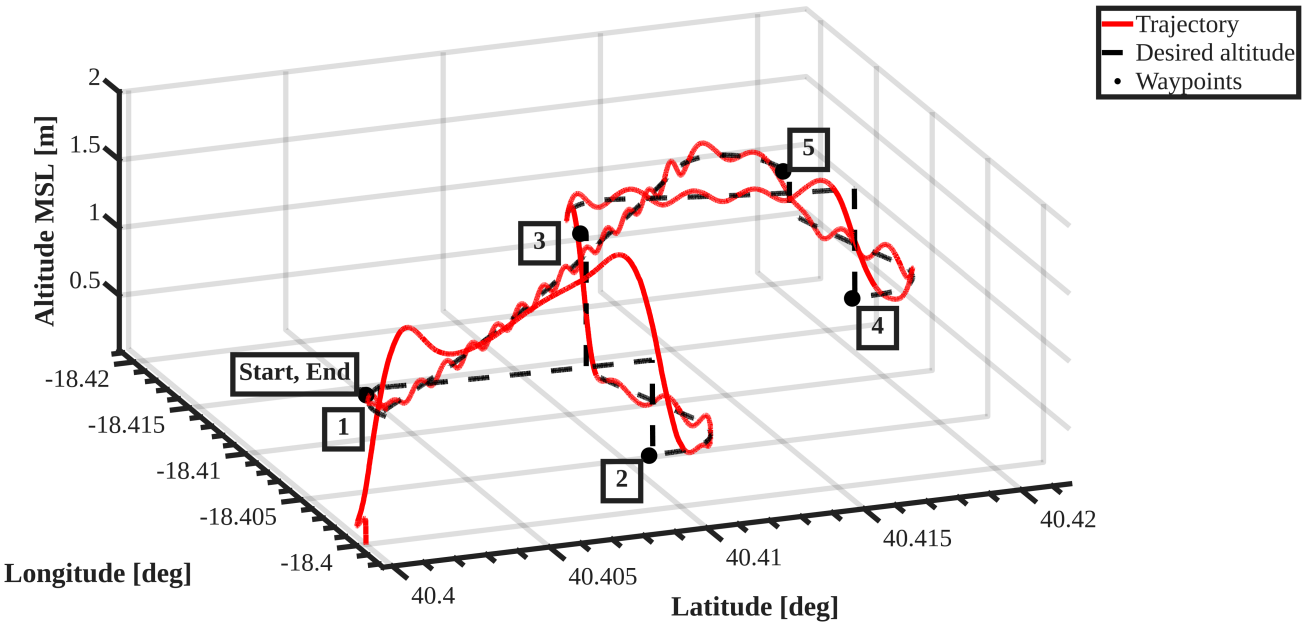


Figure 12: 3D low altitude waypoint mission. The axes represent latitude and longitude of the waypoints. The mission starts at GPS coordinate $(40.40°, -18.40°)$, a region of the North Atlantic Ocean. The WIG craft is tasked to simultaneously control position and height.

4.2.1 Results and discussion

The simulation results demonstrate that the proposed waypoint guidance module successfully performs autonomous waypoint navigation across all evaluated mission scenarios. In every case, the guidance system correctly managed waypoint sequencing, generated dynamically feasible reference commands, and completed the assigned missions without navigation failures.

The modular separation between guidance and low-level control allows the same waypoint guidance algorithm to operate independently of the specific controller implementation. Consequently, different flight control strategies developed within WP5 can be integrated without modifying the navigation layer,

facilitating software reuse and simplifying future system upgrades.

The presented waypoint guidance module also provides the interface between the path planning algorithms described in the previous section and the autonomous manoeuvring functions presented in the following sections of this deliverable. Together, these components establish the foundation for autonomous navigation of the AIRSHIP WIG vehicle.

A detailed quantitative evaluation of the low-level controller, including transient response characteristics, tracking accuracy, and time-domain performance metrics, is provided in Deliverable D5.2 and is therefore not repeated here.

5 Learning-Based and Closed-Form Guidance

The guidance results in the preceding section rely on a deterministic PID cascade with a line-of-sight waypoint update. In parallel, and because part of the guidance work was developed inside a custom learning environment, the University of Luxembourg pursued a complementary family of guidance laws that are either *closed-form* (constant-time) or *learned* (a single forward pass at inference). The motivation is the same speed/compute constraint that defines the WIG application, but the workflow differs: a learning-based solution trained in a bespoke 6-DoF environment cannot always reuse the same analysis pipeline, so it is reported here on its own terms. This section presents that work and, in doing so, exercises the framework on *two* airframes — the small A0-S scale prototype and the larger Airship_V7 — which makes the effect of airframe scale on controllability directly visible.

5.1 Guidance architecture

The guidance layer is the outer loop of a two-rate cascade: it emits setpoints (a waypoint, or a bank/turn-rate command) at a low rate, and the inner controller of Task 5.2/5.3 tracks them at a high rate. Two primitives are used. For following a leg, a closed-form lookahead (L1) law selects a "carrot" at distance L_1 ahead of the vehicle's projection onto the path and commands the lateral acceleration needed to chase it,

$$a_{\text{cmd}} = \frac{2\,V^2}{L_1}\,\sin\eta,$$

where V is the airspeed and η is the angle to the carrot (Johansen & Fossen, 2020; Kai, Hamel, & Samson, 2018). The lateral acceleration maps to a coordinated bank command; the single knob L_1 trades convergence speed against overshoot, and the law is constant-time on embedded hardware. For flying to and through 3-D waypoints, a learned full-6-DoF policy observes the geometry relative to the active leg (cross-track and along-track distance, heading error, altitude error, attitude, body rates, airspeed error) and outputs the guidance command directly; the network is small and evaluates as a single forward pass. Both primitives were developed on the project's 6-DoF environment (Towers et al., 2024; Raffin et al., 2021) on the A0-S and Airship_V7 airframes.

5.2 Coordinated turns and the turn-radius trade

The lateral primitive underlying both the L1 law and the waypoint policy is the coordinated banked turn: a commanded bank that holds altitude while turning. On the A0-S, an alternating-turn sequence (8 turns/episode over 512 episodes, $\pm 10^\circ$ command) tracks the bank reference while keeping the altitude error within roughly ± 3 m throughout the turns (Fig. 13), with a median settled bank error of $0.5^\circ/1.3^\circ$ (right/left) at 100% survival. Because bank angle sets the turn radius ($R \approx V^2/(g\tan\phi)$), sweeping the commanded bank exposes the precision/robustness trade directly (Fig. 14). On the A0-S (trim 13 m/s) the turn is robust and accurate up to about 13° of bank — survival stays at 100% and the settled bank RMSE grows roughly linearly with bank angle — after which robustness falls off as the tighter, more aggressive turns approach the airframe's limits. On the larger Airship_V7 (trim 28 m/s) the same law holds altitude to 1.6 m and bank to 1.0° at 100% survival, with the higher cruise speed yielding correspondingly larger radii (e.g. $-9^\circ \to R \approx 506$ m, $-14^\circ \to 321$ m, $-20^\circ \to 220$ m). The comparison quantifies how airframe scale and trim speed reshape the controllable envelope: the small, slow A0-S turns tightly but loses margin at high bank, whereas the large, fast V7 turns on much wider radii but with markedly steadier tracking.

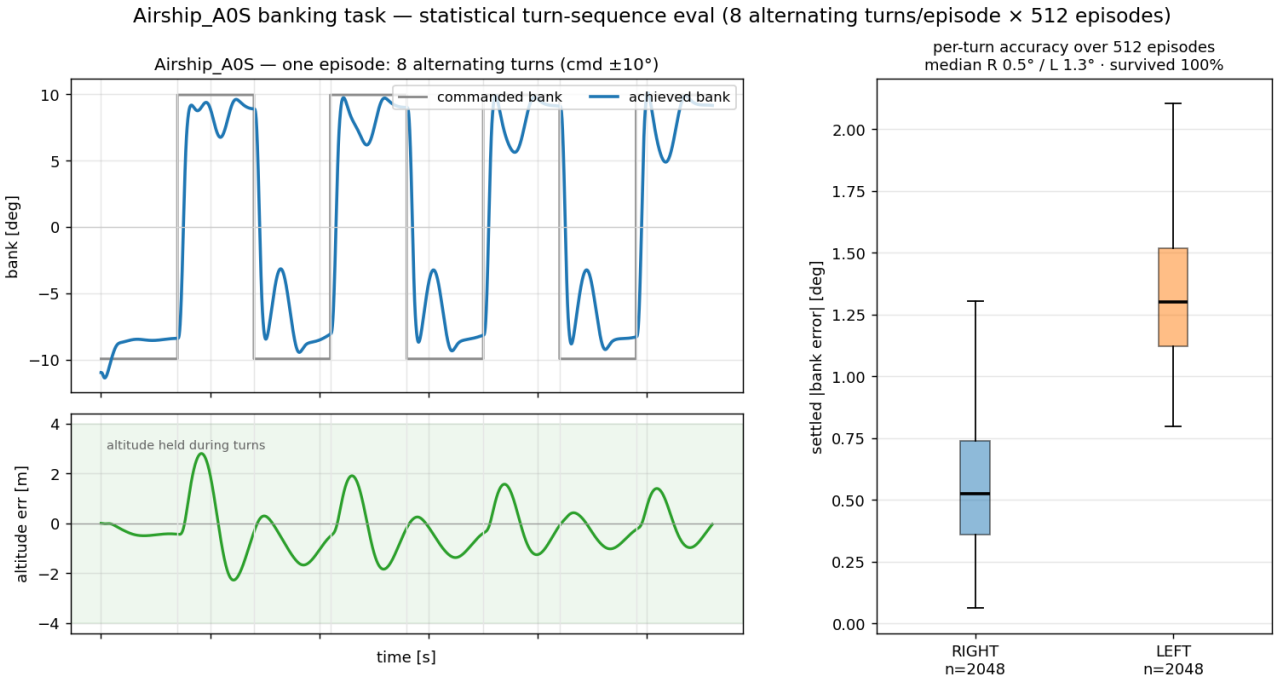


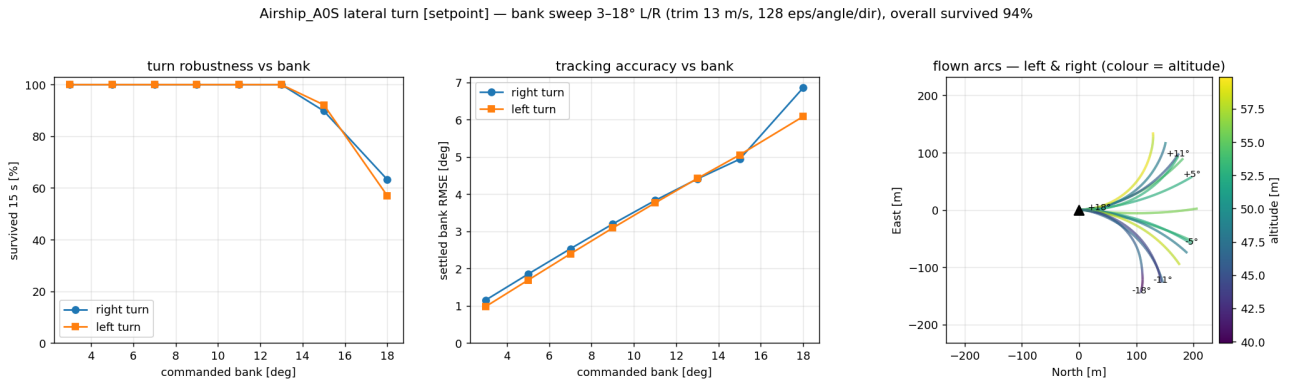
Figure 13: Coordinated-turn tracking on the A0-S: one episode of eight alternating $\pm 10^{\circ}$ bank commands (top-left, commanded vs. achieved bank), the altitude held during the turns (bottom-left), and the per-turn settled bank-error distribution over 512 episodes (right).

5.3 Waypoint following

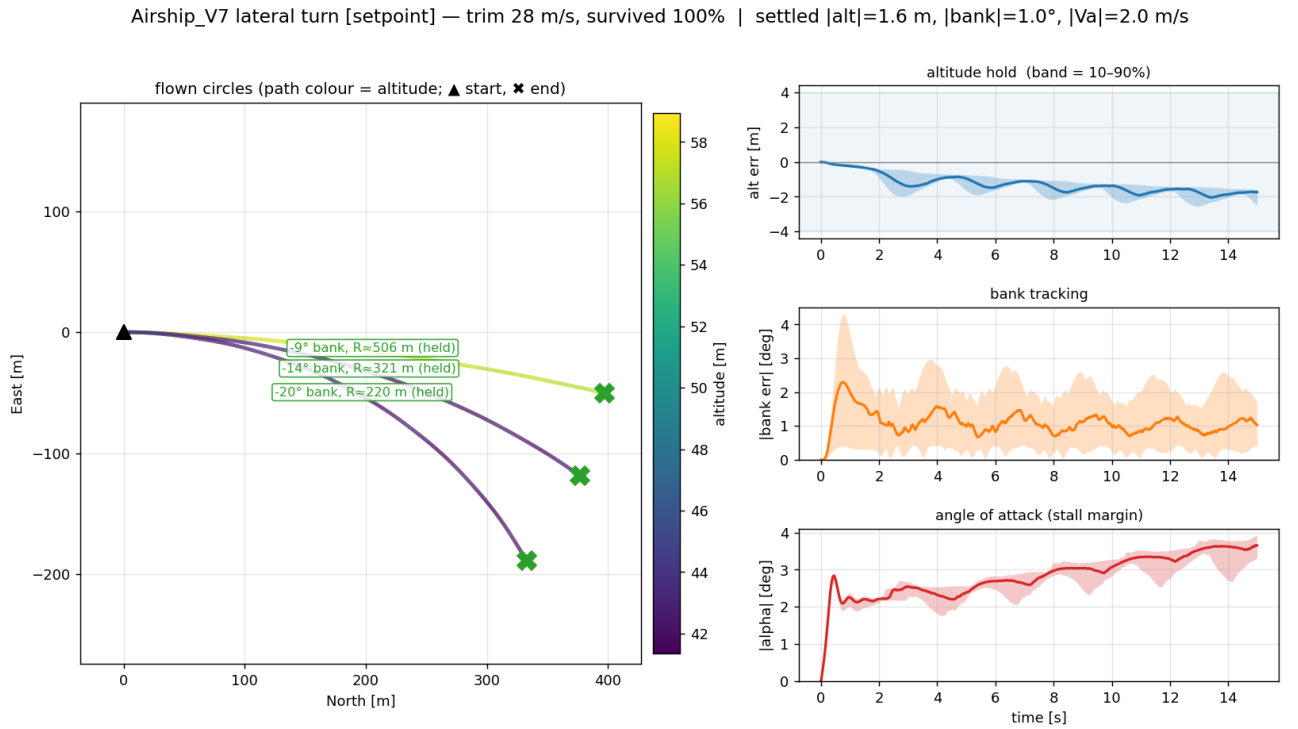
The learned guidance policy was evaluated on single 3-D waypoints. A waypoint is counted as captured only when the vehicle is simultaneously inside a horizontal (≤ 10 m) and a vertical (≤ 4 m) arrival gate. Aggregated over randomly placed waypoints the policy captures $\sim 36\%$ of targets with *zero* catastrophic failures (Fig. 15); crucially, the arrival rate rises to $\sim 84\%$ on the shortest legs and falls smoothly with leg distance. The misses are timeouts at the gate, not loss of control — the open-loop heading drift over a long leg simply grows faster than the policy re-corrects within the arrival radius.

The same binding constraint is confirmed by an evaluation sweep on the V7 (Fig. 16): failure is zero everywhere, the horizontal channel is effectively nailed, and it is the *altitude lag* that limits capture — loosening the vertical gate recovers the arrival rate far more than loosening the horizontal one. The behaviour is therefore precision-limited, not safety-limited.

The sequence variant chains legs by re-anchoring the leg origin on each capture. On a demanding chain of distant legs (40–100 m, 10 m capture radius) it captures on average 1.49 of 4 waypoints on the A0-S and 1.70 of 4 on the larger V7 (Fig. 17); short, well-aligned sequences complete fully, while the long-leg heading drift terminates the harder chains. Closing this gap by feeding the closed-form lookahead carrot into the waypoint policy is the immediate next step.



(a) A0-S (trim 13 m/s): bank sweep 3–18°. Robustness (left), settled bank RMSE (centre), flown arcs colour-coded by altitude (right).



(b) Airship_V7 (trim 28 m/s): flown circles with the bank-to-radius mapping (left) and the held altitude, bank error and angle-of-attack margin (right).

Figure 14: Coordinated-turn robustness and accuracy versus commanded bank on the two airframes. Bank angle sets the turn radius; the slow A0-S turns tightly but loses robustness past $\sim 13°$, while the fast V7 turns on wide radii with very steady tracking.

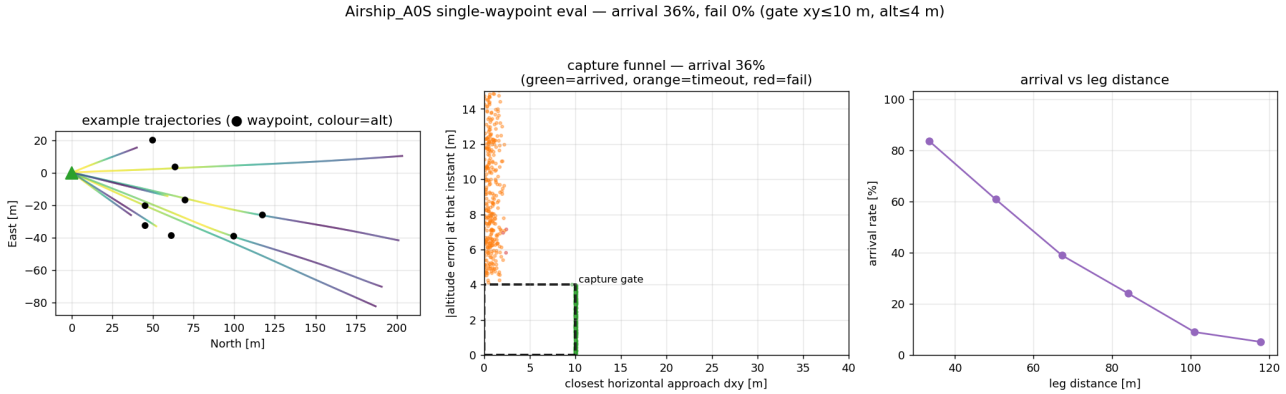


Figure 15: Single-waypoint guidance on the A0-S (gate: ≤ 10 m horizontal, ≤ 4 m vertical). Left: example ground tracks (colour encodes altitude). Centre: the capture funnel against the arrival gate (no run fails catastrophically). Right: arrival rate vs. leg distance, $\sim$84% on short legs decaying gracefully to a $\sim$36% aggregate.

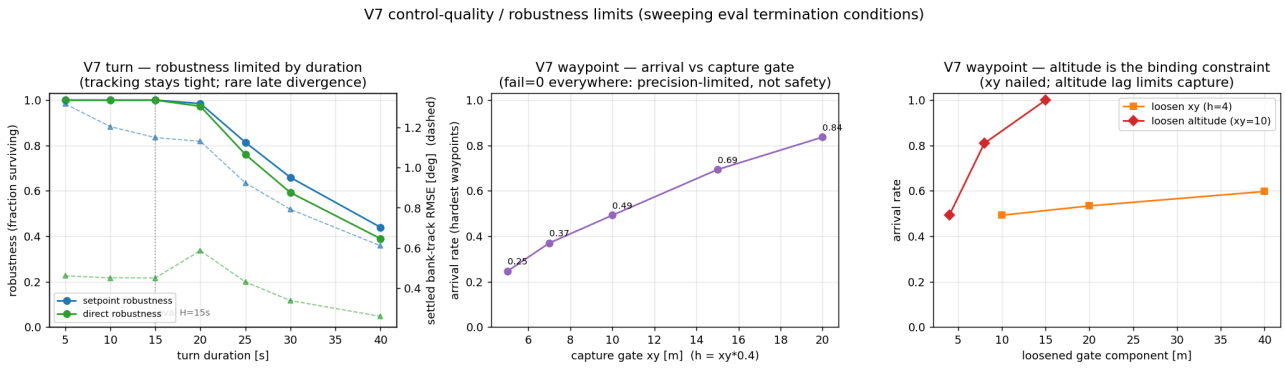
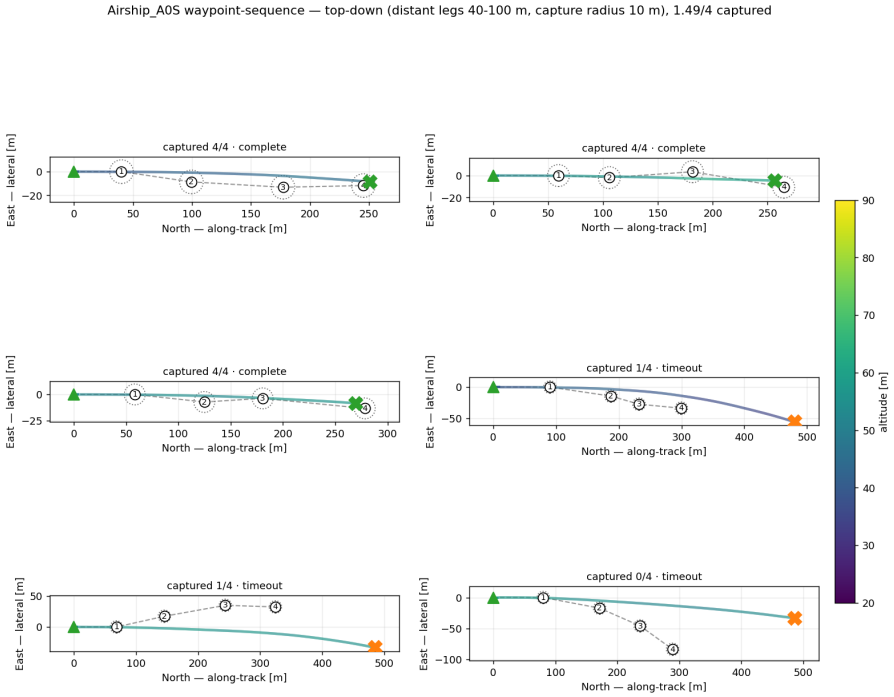
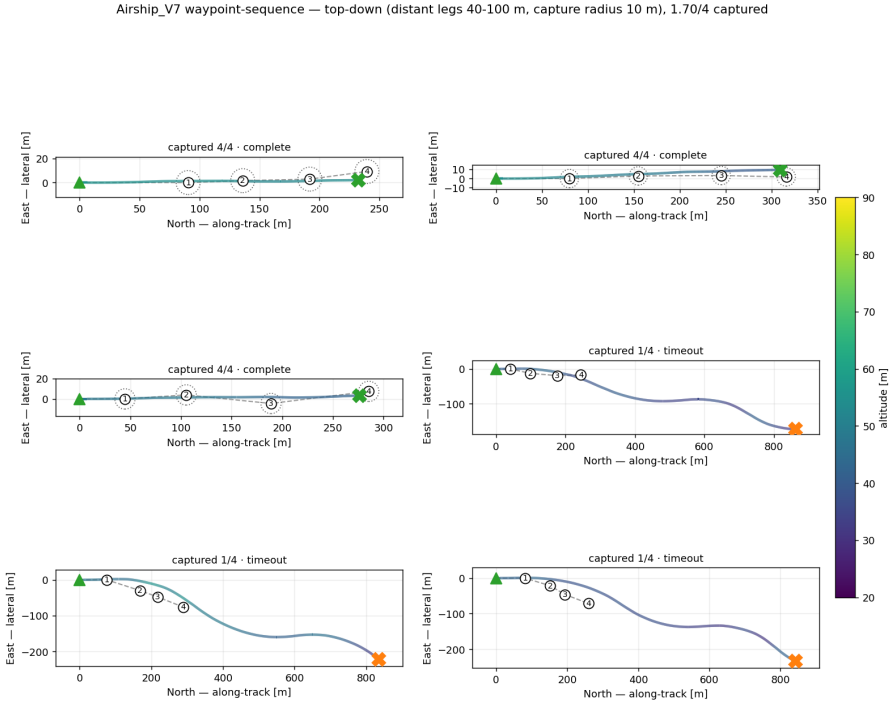


Figure 16: Control-quality limits on the V7. Left: turn robustness versus turn duration (tracking stays tight; only rare late divergence). Centre: waypoint arrival versus capture gate (fail $= 0$ throughout, i.e. precision-limited). Right: loosening the vertical gate recovers arrival far more than the horizontal one — altitude lag is the binding constraint.



(a) A0-S — $1.49/4$ captured



(b) Airship_V7 — $1.70/4$ captured

Figure 17: Waypoint-sequence guidance over distant legs (40–100 m, 10 m capture radius) on the two airframes. Each panel is one rollout; the policy re-anchors the leg on each capture and chains as far as the heading drift allows. Well-aligned sequences complete ($4/4$); the harder chains time out on the long legs.

6 Conclusion and Future Work

This deliverable reported the guidance and navigation work carried out under Task 7.1 for the AIRSHIP wing-in-ground vehicle, concentrating on activity (a): waypoint and trajectory planning and path following. Within the layered GNC architecture, the guidance layer was developed as a lightweight outer loop that emits setpoints for the validated inner control loop of Tasks 5.2/5.3, keeping the same interface in simulation and on the vehicle.

Three strands of work were delivered. A global/local path-planning pipeline, built on ROS 2 with OMPL and Dubins curves, computes a cost-optimal path over an elevation map — preferring low-altitude flight to exploit ground effect — and re-plans locally around path deviations, restricted zones and obstacles, including the use of meteorological (swell) data to steer the craft away from high-wave areas. A waypoint-guidance and path-following law, based on a cascaded PID architecture with Haversine-based waypoint logic, autonomously flew the vehicle through 2-D "square" and "eight" missions and a fully 3-D waypoint profile, holding the target altitude near 1 m above the surface while respecting bank limits, and was characterised with standard time-domain and tracking performance metrics. Finally, a complementary family of learning-based and closed-form guidance laws — coordinated turns and a learned waypoint policy — was evaluated on two airframes of different scale, making the effect of airframe size and trim speed on controllability directly visible: the small, slow A0-S turns tightly but loses robustness at high bank, while the larger, faster Airship_V7 turns on much wider radii with markedly steadier tracking.

The common thread across all three strands is the deliberate trade of a small amount of tracking precision — which the obstacle-sparse over-water environment tolerates — for the speed and low computational cost the high-speed WIG envelope demands, using closed-form or single-forward-pass guidance in preference to heavy iterative planners.

The main residual limitation identified during evaluation is the open-loop heading drift on long waypoint legs, which is the binding constraint on continuous waypoint capture; the failure mode is precision-limited rather than safety-limited, with no catastrophic failures observed and the horizontal channel effectively nailed. Feeding the closed-form lookahead (L1) carrot into the learned waypoint policy to suppress this drift is the immediate next step. Beyond that, the guidance layer developed here feeds the remaining Task 7.1 activity (basic manoeuvres: take-off, landing, see-and-avoid, turning) and provides the input for the validation tasks T7.2 and T7.3, and it is exercised through the same Software-in-the-loop/Hardware-in-the-loop process reported for the control avionics in deliverable D5.3. The software will be progressively deployed and adapted from the A0-D/A0-S scale prototypes toward the full-scale vehicle.

References

cnig. (2018). *Instituto Geográfico Nacional centro de descargas.* Retrieved 2026-06-25, from https://centrodedescargas.cnig.es/CentroDescargas/home

Fankhauser, P., & Hutter, M. (2016). A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In A. Koubaa (Ed.), *Robot operating system (ros) – the complete reference (volume 1)* (chap. 5). Springer. Retrieved from http://www.springer.com/de/book/9783319260525 doi: 10.1007/978-3-319-26054-9{_}5

Guo, W., Tyrovouzis, T., Flores, E., Ramsey, C. W., Kingston, Z. K., Şucan, I. A., … Kavraki, L. E. (2026). *The Open Motion Planning Library 2.0.* Retrieved from https://arxiv.org/abs/2605.29301

Johansen, T. A., & Fossen, T. I. (2020). Guidance, Navigation, and Control of Fixed-Wing Unmanned Aerial Vehicles. In M. H. Ang, O. Khatib, & B. Siciliano (Eds.), *Encyclopedia of Robotics* (pp. 1–9). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2025-11-17, from http://link.springer.com/10.1007/978-3-642-41610-1_67-1 doi: 10.1007/978-3-642-41610-1_67-1

Kai, J.-M., Hamel, T., & Samson, C. (2018). *A unified approach to fixed-wing aircraft path following guidance and control.* arXiv. Retrieved from http://arxiv.org/abs/1803.05184 (arXiv:1803.05184 [cs]) doi: 10.48550/arXiv.1803.05184

Lim, J. (2022). *grid_map_geo github.* Retrieved 2026-06-25, from https://github.com/ethz-asl/grid_map_geo

NLS. (2024). *National Land Survey of Finland mapsite.* Retrieved 2026-06-25, from https://asiointi.maanmittauslaitos.fi/karttapaikka/tiedostopalvelu/laserkeilausaineisto_05?lang=en

Owen, M., Beard, R. W., & McLain, T. W. (2015). Implementing dubins airplane paths on fixed-wing uavs*. In K. P. Valavanis & G. J. Vachtsevanos (Eds.), *Handbook of unmanned aerial vehicles* (pp. 1677–1701). Dordrecht: Springer Netherlands. Retrieved from https://doi.org/10.1007/978-90-481-9707-1_120 doi: 10.1007/978-90-481-9707-1_120

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1-8. Retrieved from http://jmlr.org/papers/v22/20-1364.html

Robusto, C. C. (1957). The cosine-haversine formula. *The American Mathematical Monthly*, *64*(1), 38–40. Retrieved from http://refhub.elsevier.com/S0038-092X(18)30017-3/h0290

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., … others (2024). Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032*.