Technical University of Munich

Institute of Flight System Dynamics

Development and Evaluation of a Full-Stack Path Planner for an Air Taxi Control Center

—

Entwicklung und Bewertung eines Full-Stack-Pfadplaners für ein Air-Taxi-Kontrollzentrum

Master's Thesis

Author: Sukhbir Singh

Matriculation Number: 03763741

Supervisors: Dominik Heimsch
Dr. Enno Nagel

Examiner: Prof. Dr.-Ing. Florian Holzapfel

2025-10-01

Statutory Declaration

I, Sukhbir Singh, declare on oath towards the Institute of Flight System Dynamics of Technical University of Munich, that I have prepared the present Master's Thesis independently and with the aid of nothing but the resources listed in the bibliography.

This thesis has neither as-is nor similarly been submitted to any other university.

Garching, 2025-10-01

Sukhbir Singh

Abstract

This thesis presents the design and implementation of a scalable and modular path planning system for airtaxi operations in structured urban airspace (U-space). Leveraging a microservice-based architecture, the system addresses the complexity of low-altitude corridor navigation through a global A path planner and a proposed local planner using potential fields for dynamic obstacle avoidance.*

Developed in Python and Java SpringBoot with standardized web interfaces, the system supports future extensions such as real-time obstacle fusion and adaptive replanning. A CesiumJS-based frontend enables 3D visualization of 4D trajectories. System performance is evaluated with a focus on scalability and latency, including a breakdown of service-level delays.

By focusing on real-world deployability and modularity, this work contributes to bridging the gap between U-space theoretical frameworks and operational airtaxi management systems, drawing on proven microservice principles from UAV domains.

Kurzfassung

Diese Masterarbeit stellt die Entwicklung eines skalierbaren und modularen Pfadplanungssystems für den Betrieb von Lufttaxis im strukturierten urbanen Luftraum (U-Space) vor. Mithilfe einer mikroservicebasierten Architektur wird die Komplexität der Navigation in niedrig gelegenen Luftkorridoren durch einen globalen Pfadplaner (A-Algorithmus) und einen vorgeschlagenen lokalen Planer auf Basis von Potentialfeldern adressiert.*

Die Implementierung erfolgt in Python und Java SpringBoot mit standardisierten Webschnittstellen und ermöglicht Erweiterungen wie Echtzeit-Hindernisfusion und adaptive Neuplanung. Ein mit CesiumJS entwickeltes Frontend visualisiert die vierdimensionalen Trajektorien in 3D. Die Systemleistung wird im Hinblick auf Skalierbarkeit und Latenz analysiert.

Durch den Fokus auf reale Einsatzfähigkeit leistet diese Arbeit einen Beitrag zur Überbrückung der Lücke zwischen theoretischen U-Space-Konzepten und operativen Lufttaxi-Systemen und greift bewährte Mikroservice-Prinzipien aus UAV-Anwendungen auf.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Table of Symbols

Latin Letters

Symbol	Description
V	set of vertiports (graph nodes)
E	set of corridors (graph edges)
s	starting vertiport (start node)
d	destination vertiport (goal node)
n	generic node in the graph
$d(u, v)$	geodesic (Haversine) distance between nodes u, v
$g(n)$	cumulative cost-to-come from s to n
$h(n)$	heuristic cost-to-go from n to d
$f(n)$	A* evaluation function $f(n) = g(n) + h(n)$
L	total path length (km)
T	total flight time (s or min)
V_{TAS}	true airspeed (m/s)
V_g	ground speed (m/s)
$V_{\text{takeoff}}, V_{\text{transition}}, V_{\text{cruise}}, V_{\text{approach}}$	nominal speed profile (m/s) after applying safety factor
k_{safety}	multiplicative safety factor for speed/altitude margins
H	altitude (m, MSL)
H_c	requested corridor height above ground level (AGL), fixed at $200\,\mathrm{m}$
H^{abs}	resulting absolute waypoint altitude $h_i + H_c$ (m MSL)
$\dot{H}$	climb or descent rate (m/s)
ψ	heading angle (deg)
$\dot{\psi}$	turn rate (deg/s)
ϕ	bank angle (deg)
a	longitudinal acceleration (m/s^2)
status	planning outcome code (200=resolved, 409=unresolved)

Greek Letters

Symbol	Description
φ	latitude (deg)
λ	longitude (deg)
τ	continuous time variable (s)
α	interpolation factor in [0,1] for waypoint generation
$\Delta\lambda_{\min}$	minimum separation in longitude (deg)
$\Delta\varphi_{\min}$	minimum separation in latitude (deg)
$\Delta H_{\min}$	minimum vertical separation (m)
$\Delta\tau_{\min}$	minimum temporal separation (s)

Indices and Sets

Symbol	Description
i, j	indices of nodes or waypoints
$P = [v_0, \ldots, v_n]$	path returned by A* (sequence of nodes)
$W = [w_1, \ldots, w_m]$	synthesized waypoints (with altitude/speed)
$\mathcal{V}(v_i)$	coordinate mapping for vertiport v_i
$\mathcal{T}$	generated 4D trajectory
$\mathcal{T}'$	sub-sampled trajectory
$\mathbf{S}$	separation minima vector in NDMap
E_{conflict}	set of corridor edges involved in a detected conflict

1 Introduction

Urban Air Mobility (UAM) is emerging as a key component of future transportation systems, promising rapid and sustainable aerial connections within and between cities. However, integrating large numbers of electric vertical take-off and landing (eVTOL) vehicles into low-altitude airspace poses significant operational challenges. To ensure safety, efficiency, and scalability in such environments, robust path planning and conflict-management capabilities are essential.

This thesis addresses these challenges by designing and evaluating a full-stack, microservice-based path-planning framework for U-Space air-taxi operations. The framework combines a global A* planner, a 4D trajectory-generation pipeline, and an embedded conflict-detection/resolution mechanism, complemented by a CesiumJS-based front end for real-time visualisation. It aims to provide insights into the scalability, latency, and practical deployment potential of microservice-oriented U-Space planning systems.

1.1 Background and Context

The rapid development of UAM systems has given rise to novel challenges in managing low-altitude airspace, especially in densely populated urban environments. U-Space, the European framework for Unmanned Traffic Management (UTM), proposes a structured, service-oriented approach to handle the safe and efficient integration of Unmanned Aerial Vehicles (UAVs) into controlled airspace [1]. A critical component of this framework is the path-planning system, which must be capable of generating conflict-free, efficient, and constraint-compliant trajectories in real time for a large number of concurrent missions.

Traditional monolithic path-planning systems can face challenges regarding modularity, scalability, and resilience. Recent advancements in microservices architecture offer a promising alternative by enabling loosely coupled, independently scalable services that can operate asynchronously [2]. By decoupling planning, verification, and visualisation, such architectures allow for better system maintenance and higher throughput in distributed UAV operations.

1.2 Problem Statement

Although significant efforts have been made in designing U-Space services, there is a lack of performance-oriented studies that examine the real-time behaviour of microservice-based UAV path-planning systems. In particular, the response time under high-load scenarios, the impact of integrated conflict detection and resolution, and service-level breakdowns remain underexplored.

Moreover, integrating 4D path planning (including altitude and time dimensions), vehicle-specific kinematic constraints, and conflict-resolution logic into a unified architecture poses consider-

able challenges—especially in corridor-based environments typical of U-Space implementations.

Therefore, this thesis designs and evaluates a modular path-planning system that leverages microservice principles for robust real-time operation and provides insights into its scalability, latency, and practical deployment potential.

1.3 Objectives

The main objectives of this thesis are as follows:

- Design and implement a full-stack path-planning framework based on microservice principles for U-Space air-taxi operations.
- Integrate a global A* planner with a 4D trajectory-generation pipeline and an embedded conflict-detection/resolution mechanism.
- Evaluate system performance in terms of scalability, end-to-end latency, and service-level breakdowns under different traffic loads.
- Visualise 4D trajectories and conflicts using a CesiumJS-based front end for improved situational awareness.
- Discuss future extensions such as richer resolution strategies (temporal shifts, altitude changes) and weather/risk-based corridor weighting.

1.4 Thesis Structure

The remainder of this thesis is organised into the following chapters:

- Chapter 3 presents the literature review, summarising existing U-Space concepts, UAV path-planning algorithms, conflict-management approaches, and microservice architectures.
- Chapter 4 describes the full-stack design of the system, including its main subsystems and their interactions.
- Chapter 5 outlines the scenario and data basis used in this work, focusing on the Hamburg vertiport network and corridor model.
- Chapter 6 details the implemented path-planning algorithms and the 4D trajectory-generation pipeline with integrated conflict detection and resolution.
- Chapter 7 presents the results of the evaluation, including illustrative examples of planned trajectories as well as the performance and conflict-handling analysis under different loads.
- Chapter 8 concludes the thesis and discusses future research directions.

2 Motivation

2.1 Importance of Scalable Path Planning in U-Space

Urban Air Mobility (UAM) concepts rely on safe and efficient trajectory planning within highly dynamic and dense low-altitude airspace. The European U-space framework envisions a digital and service-oriented architecture to enable such operations [3].

A key requirement for this environment is scalability: thousands of trajectory requests may need to be processed in real time, while ensuring separation minima and maintaining safe operations. Previous research has shown that trajectory-based operations require computationally efficient frameworks to handle large volumes of requests without degrading performance [4], [5].

2.2 Challenges in Real-Time Multi-UAV Coordination

Path planning interacts with several other services in the UAM ecosystem. The global planner provides a feasible route across the corridor network, but this must be complemented by flight management software capable of synthesizing flyable 4D trajectories [6], conflict detection modules such as NDMap [7], and path verification services [5].

If these services are tightly coupled in a monolithic architecture, the system risks reduced interoperability, higher technical debt, and limited extensibility. Given the diversity of programming tools and modeling approaches used in these modules, there is a strong need for a framework that allows independent implementations to interact seamlessly.

2.3 Role of Microservices in UTM Systems

A microservice-based design allows path planning to be implemented as an independent service, decoupled from trajectory generation, verification, and conflict detection. This promotes modularity, extensibility, and ease of maintenance [2], [8].

Such an approach is particularly relevant in the context of U-space, where real-time performance and scalability are critical. Evaluating this framework in terms of scalability and latency is essential to assess its viability for future UAM applications [4].

Finally, the usability of these systems also depends on visualization and stakeholder access. Different actors (e.g., operators, regulators, or airspace managers) may require different levels of visibility into the system. Determining which information should be accessible to which user group remains an open question of both technical and regulatory importance [9].

3 Literature Review

3.1 Existing U-Space Architectures

Across Europe, the SESAR Master Plan frames U-space as part of the Digital European Sky transition to trajectory-based operations (TBO) and a cloud-/data-driven, service-oriented delivery model. It sets deployment priorities (SDO5: TBO; SDO8: service-oriented delivery) and a dedicated U-space 2.0 roadmap, thereby motivating the use of 4D trajectory representations and interoperable services in this thesis [9].

A complementary comparison of SESAR and FAA NextGen clarifies how negotiated 4D operations use controlled/required times of arrival (CTA/RTA) and time windows to improve predictability and capacity. Critically for the evaluation in this work, it also catalogues key performance areas and metrics used to assess 4D concepts (e.g., safety, efficiency, capacity, predictability, controller workload) [10].

Within U-space tactical services, conflict management over 4D grids has been demonstrated: trajectories are discretized in space–time, and conflicts are detected and resolved iteratively to minimize deviation from originally filed paths while bounding processing time. These findings motivate the separation between 'monitoring' and 'resolution' adopted in the present framework and the preference to preserve operator-intended trajectories [11]. This evidence also motivates the use of a fixed corridor layer and time-based admission control for the Hamburg test network considered in this study.

3.2 Microservices in UAV Path Planning

Prototypes of U-space in-flight services show a modular, cloud-hosted stack with distinct components for U-space service management, tracking, monitoring, emergency management, and tactical deconfliction. Implementations frequently employ message-oriented middleware (e.g., ROS topics) to decouple services and support scalability. These patterns support the microservice-based architecture applied in this work, which isolates planning, conformance monitoring, and conflict handling behind stable interfaces [12].

Front-end & Visualisation (CesiumJS).

For situational awareness and planner debugging, CesiumJS was adopted as a web-native 3D geospatial engine supporting global WGS-84 rendering, time-dynamic entities (CZML), and 3D Tiles. In the proposed stack it acts as a thin client for *4D* trajectory visualisation (animated timelines) and live conformance indicators, complementing the microservices without coupling visual logic to back-end services [13].

3.3 Path Planning Algorithms (A*, Potential Fields, etc.)

Search-based planners remain strong baselines for low-altitude small Unmanned Aircraft Systems (sUAS). A two-stage planner that seeds with A* and then relaxes the path using a chain of mass–spring–damper elements in a potential field produces collision-free, flyable 3D paths and naturally embeds timing to avoid co-temporal conflicts—a property leveraged in the conflict-costing of this work [14].

Evidence for A* design trade-offs in UTM contexts shows that 3D A* can reduce travelled distance compared to stacked-2D variants ($\approx$6% on a 200 m $\times$ 200 m $\times$ 120 m map), but at substantially higher runtime (9$\times$ on average), underscoring the decision in this study to use 3D search selectively and cache heuristics [15]. Accordingly, this work employs a 2D A* over a fixed-altitude corridor graph for global planning and delegates vertical and timing constraints to a separate 4D trajectory generator (Sec. 6.2), achieving a balance between runtime and trajectory fidelity.

To handle evolving threats, hybrid global–local methods combine learning and reactive fields. One dynamic approach uses improved Q-learning for global routing and calls an artificial potential field locally when unknown threats emerge between waypoints, successfully re-planning around new hazards in simulation—supporting the "planner + local-reactor" design choice in the present study [16].

Finally, recent surveys categorize UAV planners (grid/graph: Dijkstra, A*, D*/LPA/IDA*; sampling: PRM/RRT; optimization/MIP/MPC; bio-inspired/metaheuristics; geometric/Voronoi/DT; game-theoretic) and map them to environment assumptions (static vs. dynamic), aiding algorithm selection and benchmarking design in this thesis [17].

3.4 Performance Metrics in Distributed Systems

Because the planner runs inside a distributed U-space stack, both *algorithmic* and *system* metrics are evaluated. From 4D-TBO literature, safety, predictability, and capacity metrics (e.g., losses of separation, fulfilled time windows, CTA/RTA accuracy, sector throughput) are adopted to quantify operational impact [10]. From U-space architectures, service-level metrics such as planner latency and throughput under load, queuing delays, backpressure behaviour, availability/MTBF, and tactical deconfliction processing time inform the performance evaluation [11], [12].

On the path-planning side, the reported measures include computation time versus instance size, path length as an energy proxy, smoothness/curvature bounds, re-plan time on dynamic updates, and conflict- or constraint-violation rates under injected traffic—aligning with A* versus 3D-A* trade-offs and real-time reactivity observed in prior work [14], [15].

These literature-derived metrics guided the evaluation of the Hamburg case (Secs. 7.4 and 7.5), where latency, path length, and conflict-resolution success rates were measured under different traffic loads.

4 Overview of the Full-Stack Design

The system comprises four logical functions that together implement the proposed framework: (i) a WebServer that provides authoritative aeronautical topology (vertiports and the air-corridor network), (ii) a Planner that computes candidate 4D trajectories over that topology and performs iterative conflict checks, (iii) an internal conflict-checking module embedded within the Planner that screens candidate routes against active traffic, and (iv) a U-space Service Provider (USSP) that validates, authorizes, and supervises the flight-plan lifecycle.

Figure 4–1 illustrates the interactions between these functions. In the present implementation the conflict-checking logic is not deployed as a separate service but is tightly integrated with the Planner. The Planner retrieves vertiport and corridor data from the WebServer, incorporates constraints and active traffic from the USSP, synthesizes a candidate route, and iteratively checks it against existing traffic. If a conflict is detected, the Planner recomputes the route using the lateral re-routing strategy described in Sec. 6.4 until a conflict-free trajectory is obtained. The resulting trajectory is then submitted to the USSP for validation and activation.

4.1 Subsystems of the Design

WebServer (Topology Registry)

Provides the authoritative source of vertiport locations and the corridor graph used for route generation. It stores vertiport identifiers, names, geographic footprints, elevations, capacities, and corridor geometries including altitude bands, speed limits, directionality, and optional waypoints or holding points. The WebServer content corresponds to the vertiport/corridor dataset described in Sec. 5.

Planner with Integrated Conflict Checking

Generates feasible 4D paths on the corridor network subject to aircraft performance and operational constraints, while also performing iterative conflict checks. It takes as input the topology from the WebServer, constraints and active traffic from the USSP, and mission specifications (origin, destination, time window). It produces conflict-free candidate trajectories with proposed routes, timestamps, and performance annotations. Conflict detection is implemented using NDMap (Sec. 6.3) embedded within the Planner. Only conflicts with other routes are considered in this study; geofences and other constraints are outside the present scope.

USSP (Authorization and Supervision)

Serves as the authoritative decision point for flight-plan validation, authorization, activation, modification, and termination. It performs strategic deconfliction, policy enforcement, state transitions, supervision signals, and audit logging. It maintains the authoritative state of active plans and disseminates updates or notifications that can trigger replanning.

4.2 Operational Flow Summary

1. The Planner fetches topology from the WebServer and constraints/traffic from the USSP.
2. The Planner computes a candidate plan.
3. The Planner's internal conflict-checking module (NDMap) evaluates the candidate against active traffic; if a conflict is detected, lateral re-routing is applied until no conflict remains.
4. The conflict-free plan is submitted to the USSP for validation and activation.
5. Updates to constraints or deviations trigger notifications, prompting replanning as needed.

This architecture isolates authoritative aeronautical data (WebServer) from operational authorization (USSP), while integrating conflict checking directly into the Planner to reduce inter-service latency. It enables repeatable, policy-grounded evaluation of the path planner while preserving component independence and scalability.

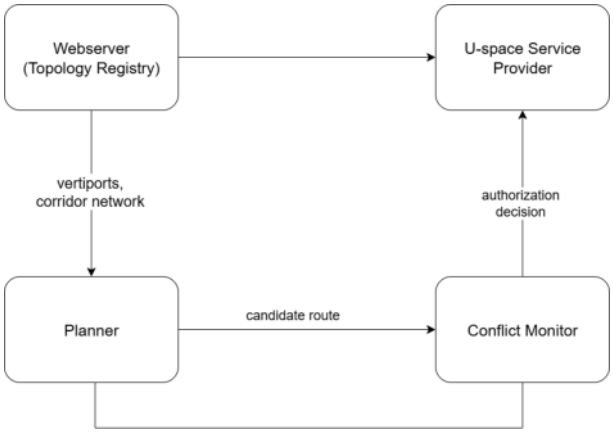


Figure 4–1: High-level architecture of the full-stack design. In the present implementation, the conflict-checking logic is embedded in the Planner.

5 Scenario and Data Basis

The path planning framework developed in this work is evaluated in a representative urban environment: the metropolitan area of Hamburg. This scenario provides a dense network of potential vertiports connected via corridors and reflects the kinds of challenges expected in future urban air mobility (UAM) operations.

5.1 Vertiports

Vertiports represent the take-off and landing sites for eVTOL aircraft. In this study, a set of anonymized vertiport locations within Hamburg is used. Each vertiport may contain one or more landing pads; for analysis, the average latitude and longitude of all pads is taken as the node coordinate. Figure 5–1 illustrates the spatial distribution of these vertiports across the city.

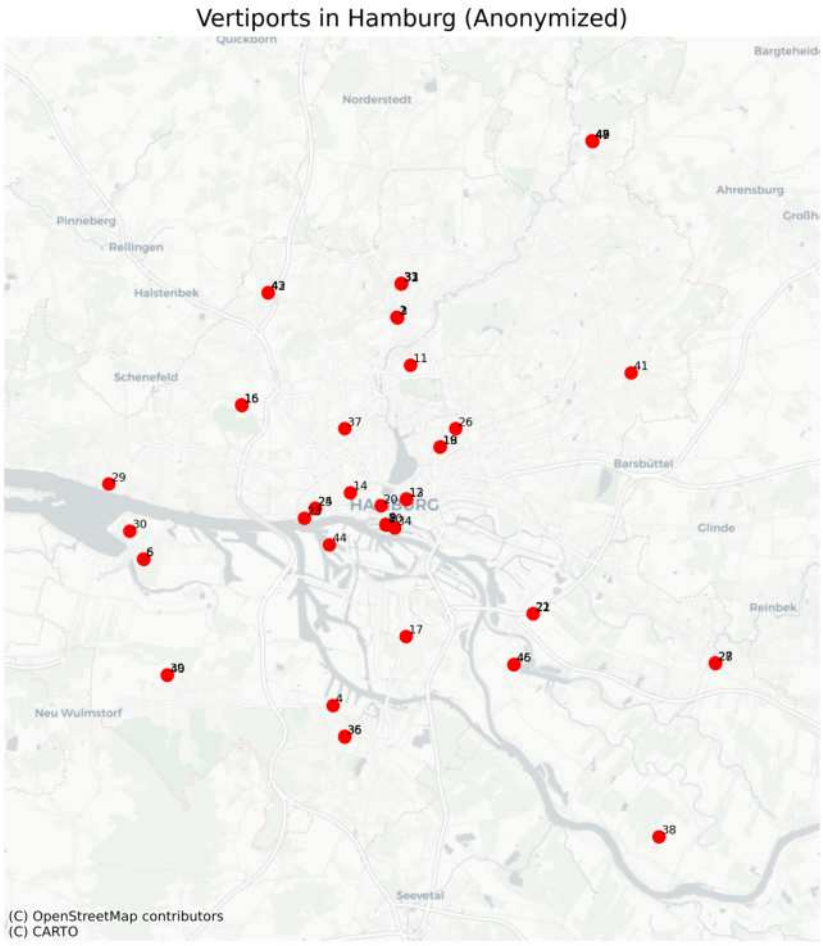


Figure 5–1: Vertiports in Hamburg (anonymized).

Corridors

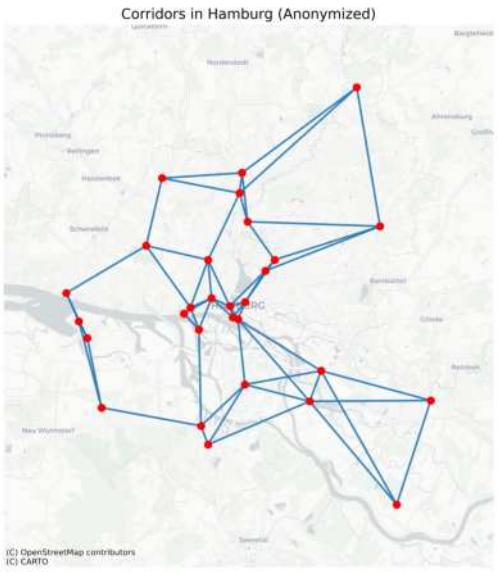
Corridors define the available airspace connections between vertiports and are modeled as *undirected* edges in the graph $G = (V, E)$. Geometrically, each corridor is represented as a straight-line segment between two vertiports, and the edge weight $d(u, v)$ corresponds to the geodesic (great-circle) distance between their coordinates.

For the vertical structure, each corridor is placed at a fixed altitude layer. According to the European Commission's U-space Blueprint [3], a reference altitude of $500\,\mathrm{ft}$ (approximately $152\,\mathrm{m}$) was adopted for low-level airspace operations. In this work, a constant altitude of $200\,\mathrm{m}$ above ground level is used for the construction of corridors. Each corridor is assigned a uniform width of $20\,\mathrm{m}$, representing a lateral safety buffer around the nominal centerline.

For visualization and operator interaction, the corridor graph was also rendered in an interactive Cesium front-end (Fig. 5–2, right). This allows the user to inspect the 3D geometry of the corridor network, vertical layering, and vertiport locations directly within a web-based interface.

Thus, the corridor model is characterized by the following specifications:

- **Shape:** straight line between two vertiports;
- **Graph representation:** undirected edge with weight equal to geodesic distance;
- **Altitude:** constant $200\,\mathrm{m}$ above ground level;
- **Width:** $20\,\mathrm{m}$.



(a) Constructed corridor network (static map view).

(b) Cesium front-end view of the same corridor network (3D visualization).

Figure 5–2: Comparison of the corridor network in static map view (left) and interactive Cesium front-end (right).

5.2 Graph Representation

The resulting dataset is an *undirected*, weighted graph in which nodes correspond to vertiports and edges represent corridors. Edge weights are given by geodesic distances between node coordinates. This graph serves as the basis for all path planning experiments in this thesis; the global A* planner (Sec. 6.1) operates on this graph and uses the notation s (start), n (current node), and d (destination).

6 Path Planning Algorithms

6.1 Global A* Algorithm in Air Corridors

For global path planning between vertiports, a graph search approach is adopted. The urban airspace is modeled as an undirected, weighted graph where nodes represent vertiports and edges represent available air corridors. The edge weights are given by the geodesic distance between the connected vertiports, computed using the Haversine formula. The planning objective is to determine the shortest feasible route between a designated start and destination vertiport.

Problem Setup

Let

- $s \in V$ denote the starting vertiport,
- $d \in V$ denote the destination vertiport, and
- $n \in V$ denote a generic node (current vertiport) under expansion during the search.

The graph $G = (V, E)$ consists of a finite set of vertiports V and undirected edges $E \subseteq V \times V$, each edge $(u, v) \in E$ carrying a weight $d(u, v)$ equal to the corridor distance between u and v.

Illustration of the Search Graph

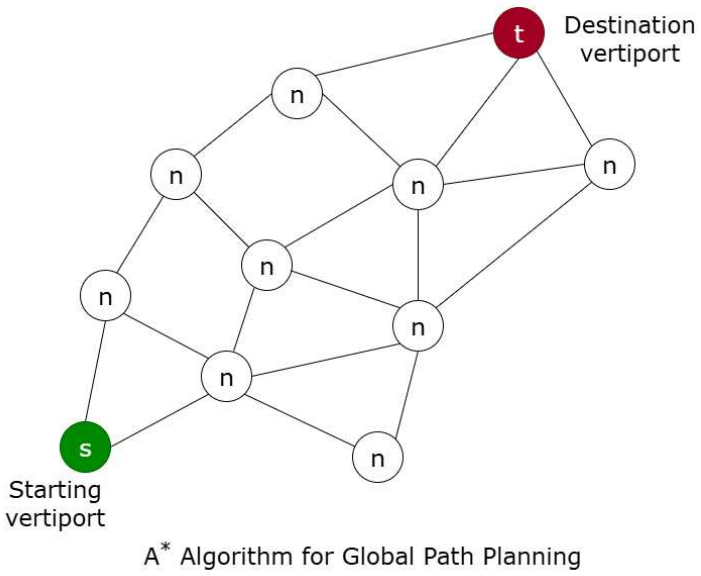


Figure 6–1: Illustration of the A* search on an undirected vertiport corridor graph. The starting node s and destination node d are connected through intermediate nodes n.

Heuristic

To guide the A* search, the geodesic (Haversine) distance between the current vertiport n and the destination d is used as the heuristic function $h(n)$. Because the Haversine distance represents the shortest path on the Earth's surface, it is admissible: the straight-line distance on the sphere is always less than or equal to the length of any feasible path along the corridor network.

Let (λ_1, φ_1) and (λ_2, φ_2) denote the longitude and latitude (in radians) of the two points respectively. Define $\Delta\varphi = \varphi_2 - \varphi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$. Then the Haversine distance is given by

$$d_{\text{hav}} = 2R \arctan\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right), \quad \text{with} \quad a = \sin^2\frac{\Delta\varphi}{2} + \cos\varphi_1 \cos\varphi_2 \sin^2\frac{\Delta\lambda}{2}, \tag{6–1}$$

where R is the Earth's radius (here $R = 6371\,\text{km}$).

Accordingly, the heuristic function becomes

$$h(n) = d_{\text{hav}}(\text{coord}(n), \text{coord}(d)), \tag{6–2}$$

where coord(v) denotes the geographic coordinates (λ, φ) of vertiport v.

Evaluation Function

The total evaluation score for each node n is defined as

$$f(n) = g(n) + h(n), \tag{6–3}$$

where $g(n)$ is the cumulative cost of the best-known path from s to n, and $h(n)$ is the heuristic estimate of the remaining cost to reach d.

In this context, the cost corresponds to the geodesic distance traveled along the air corridors. Since the corridor network is assumed to operate at a fixed altitude layer, the planning problem reduces to two dimensions, and the distance is computed purely in terms of latitude and longitude. Thus, $g(n)$ accumulates the 2D distances of all corridor segments from the start to the current node, while $h(n)$ provides the straight-line (Haversine) estimate from the current node to the destination. The evaluation score $f(n)$ therefore balances the actual distance traveled so far with an optimistic estimate of the remaining distance, ensuring efficient convergence toward the destination.

Algorithm

The algorithm used is the classical A* search, adapted to the vertiport corridor graph. It maintains a priority queue of nodes to be expanded, ordered by their $f(n)$ value.

Algorithm 6–1 A* Path Planning in Air Corridors

Require: Start vertiport s, destination vertiport d, undirected graph $G = (V, E)$ with edge distances $d(u, v)$, coordinates coord(v)

Ensure: Sequence of vertiports representing the planned route

$open \leftarrow \{s\}$ ▷ priority queue with key $f(v)$
$came\_from \leftarrow \emptyset$
$g(s) \leftarrow 0$
$f(s) \leftarrow h(s) =$ Haversine(coord(s), coord(d))
while $open$ not empty **do**
  $current \leftarrow \arg\min_{v \in open} f(v)$
  if $current = d$ **then**
    return Reconstruct path from $came\_from$
  end if
  Remove $current$ from $open$
  for all $neighbor \in$ neighbors($current$) **do**
    $tentative\_g \leftarrow g(current) + d(current, neighbor)$
    if $tentative\_g < g(neighbor)$ **then**
      $came\_from[neighbor] \leftarrow current$
      $g(neighbor) \leftarrow tentative\_g$
      $h(neighbor) \leftarrow$ Haversine(coord$(neighbor)$, coord(d))
      $f(neighbor) \leftarrow g(neighbor) + h(neighbor)$
      Add $neighbor$ to $open$
    end if
  end for
end while
return failure (no path found)

Output

The algorithm returns the ordered list of vertiports

$$\{s, v_1, v_2, \ldots, d\}, \tag{6–4}$$

that the aircraft should traverse. These nodes represent intermediate waypoints along the air corridor network and form the global reference trajectory for subsequent local planning and trajectory generation.

6.2 Trajectory Generation Pipeline

This section describes the generation of four–dimensional (4D) urban air mobility (UAM) trajectories (latitude, longitude, altitude, time).This pipeline interfaces with the C++-based CFMS [6] system while the planner is implemented in Python, demonstrating the microservice concept of this work. The pipeline transforms a discrete path over the vertiport network into a kinematically consistent, phase–annotated trajectory by (i) synthesizing geometric waypoints, (ii) assigning vertical and speed constraints, and (iii) invoking a modular flight management and 4D trajectory generation system (CFMS) [6] to densify and time–parameterize the path.

6.2.1 Notation and Conventions

The following symbols and units are used throughout this work:

- **Path over the graph:** $P = [v_0, v_1, \ldots, v_n]$ is the ordered list of vertiports yielded by the global A* planner (Sec. 6.1), with $v_0 = s$ (start) and $v_n = d$ (destination).
- **Coordinates:** each vertiport v_i is associated with WGS–84 coordinates $\mathcal{V}(v_i) = (\varphi_i, \lambda_i, h_i)$ with latitude φ_i (degrees), longitude λ_i (degrees), and pad elevation h_i (meters, MSL).
- **Altitude variables:** In this study, corridors are modelled at a fixed absolute altitude layer of $H_{\text{corridor}} = 200\,\text{m}$ above mean sea level, independent of individual pad elevations. All cruise waypoints are therefore generated at this fixed altitude:

$$H^{\text{abs}} \;=\; H_{\text{corridor}} \tag{6–5}$$

  while take-off and landing waypoints use the actual pad elevation h_i, and intermediate climb/descent waypoints interpolate between h_i and H_{corridor}.
- **Time:** continuous time is denoted by τ (seconds). This symbol is used consistently in equations to represent elapsed time along a trajectory.
- **Geodesic distance:** $d(u, v)$ is the great-circle distance between two coordinates (Haversine).
- **Linear interpolation:** $\text{lerp}(a, b, \alpha) \coloneqq a + (b - a)\,\alpha$, for $\alpha \in [0, 1]$.

6.2.2 Inputs

The following inputs are used for the trajectory generation stage:

- Path $P = [v_0, v_1, \ldots, v_n]$ from A* (Sec. 6.1).
- Vertiport map $\mathcal{V} : v_i \mapsto (\varphi_i, \lambda_i, h_i)$ (deg, deg, m MSL).
- Requested cruise height H_c (AGL), default $200\,\text{m}$.
- UAM speed profile with safety margin.

To ensure that the generated 4D trajectory remains flyable even under moderate disturbances (e.g. wind gusts, small deviations, or delayed responses), the nominal speeds used in this

study are derived from the aircraft's maximum performance values multiplied by a safety factor $s_{\mathrm{f}} \in (0, 1)$. For the reference UAM vehicle the maximum performance limits are

$$V_{\text{takeoff,max}} = 0\,\mathrm{m/s}, \quad V_{\text{transition,max}} = 20\,\mathrm{m/s},$$
$$V_{\text{cruise,max}} = 30\,\mathrm{m/s}, \quad V_{\text{approach,max}} = 20\,\mathrm{m/s}.$$

Applying a safety factor of $s_{\mathrm{f}} = 0.75$ yields the nominal speeds adopted in the planning process:

$$V_{\text{takeoff}} = s_{\mathrm{f}}\, V_{\text{takeoff,max}} = 0\,\mathrm{m/s}, \qquad V_{\text{transition}} = s_{\mathrm{f}}\, V_{\text{transition,max}} = 15\,\mathrm{m/s},$$
$$V_{\text{cruise}} = s_{\mathrm{f}}\, V_{\text{cruise,max}} = 25\,\mathrm{m/s}, \qquad V_{\text{approach}} = s_{\mathrm{f}}\, V_{\text{approach,max}} = 15\,\mathrm{m/s}.$$

These safety-factored speeds are used as the target values for the `UAM_takeoff`, `cruise` and `UAM_landing` phases when annotating the waypoints.

6.2.3 Waypoint Synthesis

Given P, the waypoint generator produces an ordered list $W = [w_1, \ldots, w_m]$ where each waypoint is

$$w_k \;=\; \{\varphi_k, \lambda_k, H_k, \texttt{id}, \texttt{(optional) speed\_reduction}\}. \tag{6–6}$$

The construction logic is as follows:

1. **Initial takeoff point:** at $(\varphi_0, \lambda_0, h_0)$.
2. **Mid–climb point:** along the initial bearing $v_0 \to v_1$ at $\alpha = 0.3$ of the segment, with altitude

$$H_{\text{climb}} \;=\; h_0 + 0.6\,(H^{\text{abs}} - h_0). \tag{6–7}$$

3. **Cruise nodes at internal vertiports:** for each v_i, $i = 1, \ldots, n-1$, altitude is set to H^{abs}.
4. **Pre–landing descent point:** along $v_{n-1} \to v_n$ at $\alpha = 0.7$ with

$$H_{\text{descent}} \;=\; h_n + 0.7\,(H^{\text{abs}} - h_n). \tag{6–8}$$

   This intermediate waypoint provides a clear transition from cruise altitude to the terminal approach. It produces a two-stage vertical profile (cruise $\to$ descent $\to$ landing) that smooths vertical-rate changes, yields more realistic trajectories, and leaves adequate horizontal distance for deceleration and stabilization before touchdown. The value $\alpha = 0.7$ is a simple, tunable heuristic chosen to begin the descent early enough for comfortable flare and speed control while preserving lateral fidelity.
5. **Final landing point:** at $(\varphi_n, \lambda_n, h_n)$.

6.2.4 Phase Encoding and Speed Constraints

Each waypoint is annotated with a nominal flight phase and target speed:

`UAM_takeoff` Initial (pad) waypoint, executed at the takeoff speed V_{takeoff}.
`cruise` Mid–climb, internal cruise nodes, and descent–initiation nodes, with $V_{\text{transition}}$ applied at climb/descent anchors and V_{cruise} along the en-route segments.
`UAM_landing` Final (pad) waypoint, with the speed set to 0 m/s (stationary at landing).

6.2.5 CFMS Integration

The waypoint list is converted to the CFMS input structure:

- (φ, λ) converted from degrees to radians.
- Altitudes expressed in meters MSL (using H^{abs} for cruise waypoints).
- Phase and speed fields set as above; the global context specifies the aircraft type `CFMS_UAMcopter`, cruise height H_c, and cruise speed V_{cruise}.

CFMS [6] then generates a dense 4D trajectory $\mathcal{T} = \{(\varphi(\tau), \lambda(\tau), H(\tau), \tau, \dots)\}$ that satisfies the kinematic profile and phase schedule.

6.2.6 Temporal and Spatial Sampling

To reduce data volume, the CFMS output is uniformly sub–sampled to at most $M = 500$ points:

$$\text{index}_i = \left\lfloor \frac{i\,(N-1)}{M-1} \right\rfloor, \quad i = 0, \dots, M-1, \tag{6–9}$$

with N original samples; the first and last samples are always retained.

6.2.7 Exported Data Fields

Each trajectory is represented as a sequence of sampled data points of the form

$$\{\varphi, \lambda, H, \mathtt{t}, \mathsf{TAS}, V_g, \phi, \dot{H}, \dot{\psi}, \mathtt{flight\_phase}\}, \tag{6–10}$$

where:

- φ – latitude of the sample point (deg).
- λ – longitude of the sample point (deg).
- H – altitude above mean sea level (m).
- $\mathtt{t}$ – elapsed time since the start of the trajectory (s); denoted by τ in the equations.
- TAS – true airspeed (m/s): the speed of the vehicle relative to the surrounding air.
- V_g – ground speed (m/s): magnitude of the velocity vector relative to the ground.

- ϕ – bank angle (deg): roll about the longitudinal axis; zero is wings-level.
- $\dot{H}$ – climb rate (m/s): time derivative of altitude; positive in climb, negative in descent.
- $\dot{\psi}$ – turn rate (deg/s): rate of change of the ground-track angle. In coordinated flight it can be approximated by

$$\dot{\psi} = \frac{g \tan \phi}{V_g}, \tag{6–11}$$

  where g is gravitational acceleration (m/s^2).
- `flight_phase` – label indicating the nominal phase of flight: `UAM_takeoff` (departure and climb), `cruise` (en-route), `UAM_landing` (approach and landing), or `other` (any intermediate or internal phase).

6.2.8 Algorithm Summary

Algorithm 6–2 Trajectory Generation from A* Path

Require: Path $P = [v_0, \dots, v_n]$, vertiport map $\mathcal{V}$, cruise height H_c (AGL)
Ensure: Subsampled, phase–annotated 4D trajectory $\mathcal{T}'$

$H^{\text{abs}} \leftarrow h_0 + H_c$ ▷ constant cruise layer above departure pad
$W \leftarrow$ synthesize_waypoints$(P, \mathcal{V}, H^{\text{abs}})$
annotate phases and speeds in W (`UAM_takeoff`, `cruise`, `UAM_landing`)
$W_{\text{CFMS}} \leftarrow$ convert_to_CFMS(W) ▷ radians; altitudes in m MSL
$\mathcal{T} \leftarrow$ CFMS_generate_trajectory(W_{CFMS})
$\mathcal{T}' \leftarrow$ uniform_subsample$(\mathcal{T}, M = 500)$
export $\mathcal{T}'$ as JSON $\{\varphi, \lambda, H, \mathtt{t}, \dots\}$

Consistency with Global Planning.

The global A* planner (Sec. 6.1) minimizes 2D geodesic distance over the corridor graph at a fixed altitude layer. The trajectory pipeline respects this assumption by assigning a constant cruise height H_c (AGL) along the route and applying geodesic distances only for horizontal planning.

6.3 Conflict Detection with NDMap

The following section is a summarized version of the NDMap conflict detection method presented in [7], adapted to the requirements of this work.

Overview

In this work, conflicts are detected using NDMap configured in four dimensions $(\lambda, \varphi, H, \tau)$, i.e., longitude $[\mathrm{deg}]$, latitude $[\mathrm{deg}]$, altitude $[\mathrm{m}]$, and time $[\mathrm{s}]$. NDMap indexes submitted trajectories

in this 4D space and, for a newly inspected trajectory, efficiently identifies portions of previously submitted trajectories that come within user-defined separation minima. This enables very fast screening suitable for near real-time admission control.

Inputs and Configuration

Each trajectory is represented as an ordered sequence of sampled four-dimensional points

$$(\lambda, \varphi, H, \tau) \in [-180^\circ, 180^\circ] \times [-90^\circ, 90^\circ] \times [0, 100000]\ \text{m} \times [0, 86400]\ \text{s},$$

where λ is longitude (deg), φ is latitude (deg), H is altitude above mean sea level (m), and τ is time (s).

Separation requirements are encoded as a vector

$$\mathbf{S} = (\Delta\lambda_{\min},\ \Delta\varphi_{\min},\ \Delta H_{\min},\ \Delta\tau_{\min}), \tag{6–12}$$

which sets the minimum allowed spacing in each dimension and parameterizes the detector's resolution. In this study,

$$\mathbf{S} = (0.002^\circ,\ 0.004^\circ,\ 300\ \text{m},\ 90\ \text{s}),$$

corresponding respectively to minimum separations in longitude (deg), latitude (deg), altitude (m), and time (s). A minimum inter-conflict interval of 2 s is applied to merge very closely spaced detections.

Output

For each conflicting pair involving the inspected trajectory, NDMap returns one or more *conflict time intervals*

$$[\tau_{\text{start}},\ \tau_{\text{end}}],$$

expressed in seconds on the scenario time axis. These intervals constitute the primary output consumed by the deconfliction/scheduling logic in this work.

Performance Characteristics

According to the NDMap documentation [7], runtimes are on the order of milliseconds per trajectory on commodity hardware, with reported averages of a few milliseconds per object for national- to continental-scale datasets. The implementation exhibits near-linear scaling with dataset size and tunable trade-offs between runtime and memory via configuration parameters, which supports near real-time screening in the intended UAM use case.

Summary. NDMap consumes sampled 4D trajectories together with separation minima $\mathbf{S}$ and produces conflict time intervals $[\tau_{\text{start}}, \tau_{\text{end}}]$ for any violations. The detector's low latency and

favorable scaling make it appropriate for iterative, online conflict checks within the planning pipeline described in this thesis.

6.4 Conflict Resolution

Conflict detection with NDMap (Sec. 6.3) identifies spatio-temporal overlaps between a newly proposed candidate trajectory and the set of already submitted trajectories. To ensure conflict-free operation, conflict resolution is applied directly within the path planning loop of this work.

Resolution Strategy

In this study, conflict resolution is realized through **lateral re-routing**. The principle is to discourage the reuse of route segments that are involved in conflicts and to force the planner to search for an alternative path through the corridor network.

1. A candidate path is computed using the global A* planner (Sec. 6.1).
2. The corresponding trajectory is generated and checked against existing traffic using NDMap.
3. If a conflict is detected, the edges of the corridor graph that belong to the conflicting candidate route are penalized by assigning them effectively infinite weight, i.e.,

$$d(u, v) \coloneqq \infty, \quad \forall\, (u, v) \in E_{\text{conflict}},$$

   which removes them from further consideration by A*.
4. A* is re-executed on the modified graph to find an alternative route that avoids the discouraged edges.
5. This process is repeated iteratively until NDMap no longer reports conflicts.

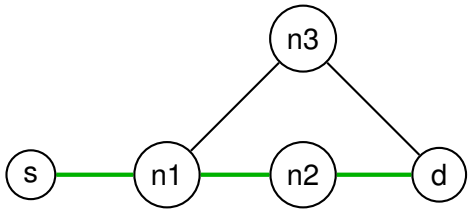
Properties

This integrated resolution approach has the following characteristics:

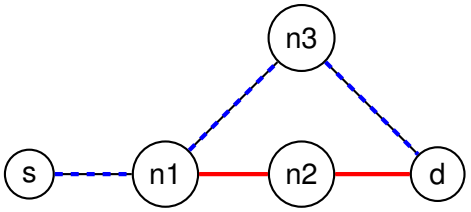
- **Simplicity:** implemented as an extension of the global A* search without requiring additional modules.
- **Determinism:** the iterative penalization guarantees that once all conflicting edges are discouraged, the resulting solution is conflict-free (if a feasible route exists).
- **Cost trade-off:** since conflicts are avoided by forbidding certain edges, the resulting route may be longer than the original candidate, reflecting a trade-off between efficiency and safety.
- **Scalability:** the conflict detection and re-routing loop can be repeated for successive trajectory submissions, enabling online admission control in multi-UAV settings.

Scope of this Work

Although other resolution mechanisms such as temporal shifting or vertical maneuvering are possible, the present work focuses exclusively on lateral re-routing within the corridor graph. This allows for a modular and scalable demonstration of how conflict detection (via NDMap) and path planning (via A*) can be tightly integrated in a unified framework.



(a) Initial candidate path (green).



(b) Conflict edges penalized (red), re-routed path (blue dashed).

Figure 6–2: Illustration of conflict resolution via lateral re-routing.

7 Results

This chapter presents the results of the path planning framework developed in this work. It reports representative trajectories between selected vertiports in the Hamburg scenario (Sec. 5), the associated kinematic profiles (true airspeed, turn rate, altitude, bank angle), geometric properties of the paths, and computational performance for planning and trajectory generation (Secs. 6.1 and 6.2). A dedicated section (Sec. 7.3) demonstrates conflict detection outcomes using NDMap (method in Sec. 6.3).

All trajectories and profiles shown in this chapter are *planned (reference) 4D trajectories* produced by the pipeline, i.e., the intended paths to be followed by the aircraft; they are not simulation runs.

7.1 Experimental Setup

7.1.1 Scenario and Data

The experiments are conducted on the Hamburg vertiport network described in Sec. 5, with corridors modeled as undirected edges at a constant altitude layer of $200\,\mathrm{m}$ AGL and width $20\,\mathrm{m}$. Nodes correspond to vertiports (averaged pad coordinates); edges are weighted by geodesic distance.

7.1.2 Planner and Generator Configuration

- **Global planner:** A* on the undirected corridor graph (Sec. 6.1), with Haversine heuristic $h(n)$ and evaluation function $f(n) = g(n) + h(n)$.
- **Trajectory generator:** CFMS-based 4D trajectory synthesis (Sec. 6.2) with cruise height $H_c = 200\,\mathrm{m}$ (AGL), speed profile $(V_{\text{takeoff}}, V_{\text{transition}}, V_{\text{cruise}}, V_{\text{approach}})$.

7.1.3 Evaluation Metrics

The following metrics are reported for each run:

- **Path geometry:** number of legs, total length L (km).
- **Timing:** total flight time T (s), average TAS.
- **Kinematics:** TAS profile, altitude compliance w.r.t. H_c, peak bank angle $\max \phi$, peak turn rate $\max \dot{\psi}$.
- **Compute:** A* runtime and trajectory generation time.

7.2 Case Studies: Sample Trajectories

This section presents representative trajectories between selected vertiport pairs. For each case, s denotes the starting vertiport and d the destination.

7.2.1 Representative Long-Leg Route

Trajectory A was selected as a representative case because it contains one of the longest legs in the Hamburg network. This results in a comparatively high total path length and flight time. Furthermore, the extended initial leg allows the UAM vehicle sufficient time to transition from take-off into the corridor network, stabilizing its climb and entering the cruise segment more gradually. This makes Trajectory A well-suited for evaluating the performance of the global planner and the consistency of the generated cruise profile.

Route Overview

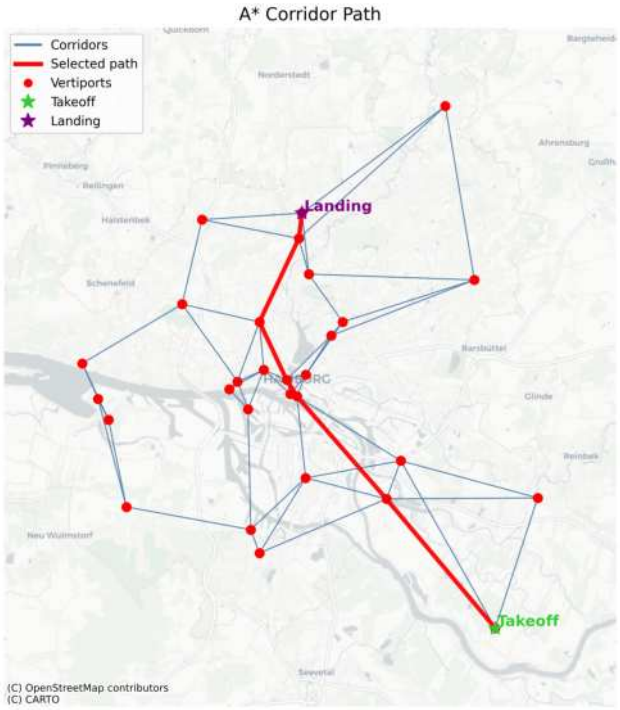
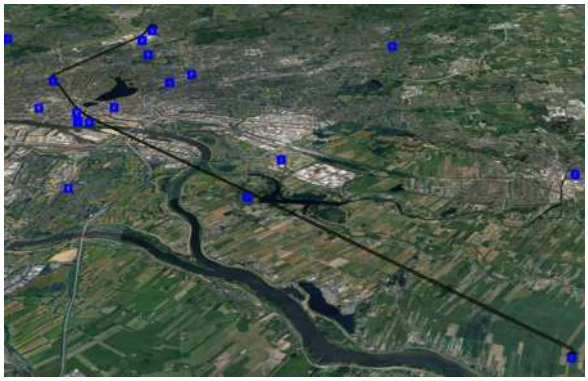


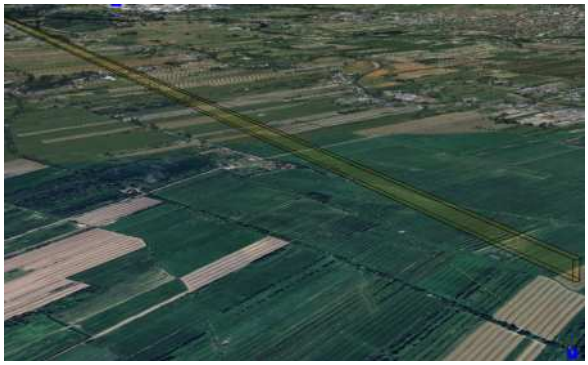
Figure 7–1: Planned reference trajectory (Trajectory A) from s to d over the Hamburg corridor network.

Front-end Visualization (Cesium)

Figure 7–2 shows the planned Trajectory A rendered in the Cesium-based web client. The left view displays the full 3D corridor context of the route, while the right view zooms in on the departure phase to highlight the take-off and initial climb segment.



(a) Planned route (Trajectory A): full 3D corridor context.



(b) Planned route (Trajectory A): zoom on take-off and initial climb.

Figure 7–2: Cesium-based visualization of the planned Trajectory A in the front end. Left: full route; right: take-off segment.

Kinematic Profiles

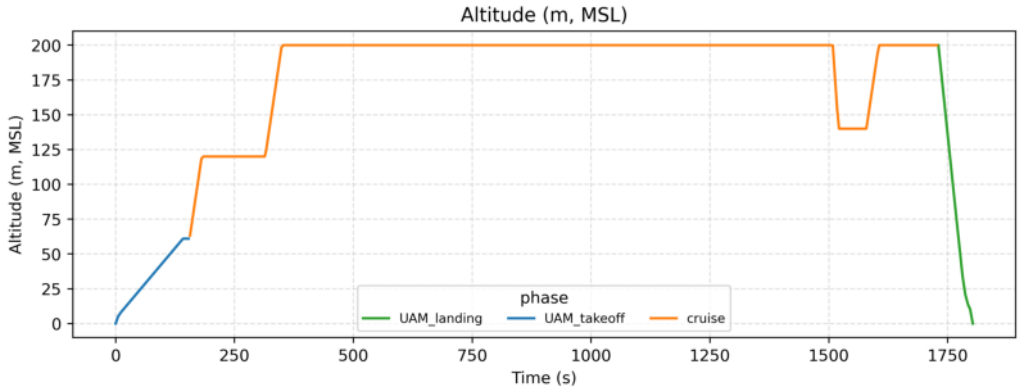


Figure 7–3: Planned altitude profile $H(\tau)$ for Trajectory A (cruise layer near $200\,\mathrm{m}$ AGL).

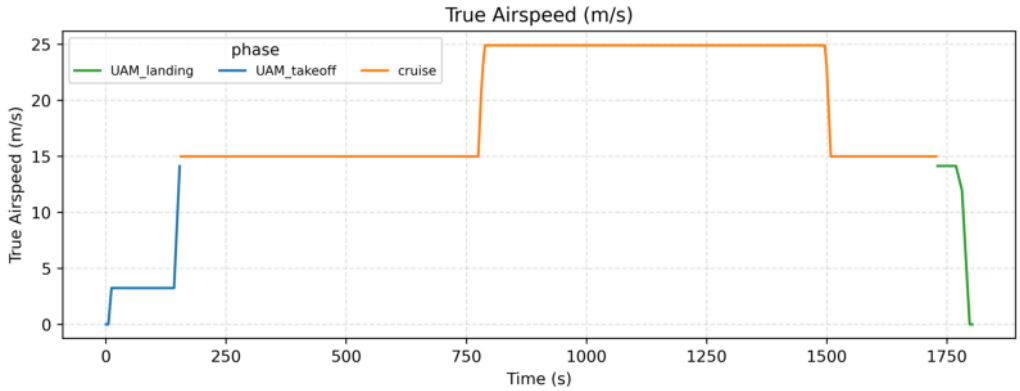


Figure 7–4: Planned true airspeed (TAS) profile for Trajectory A.

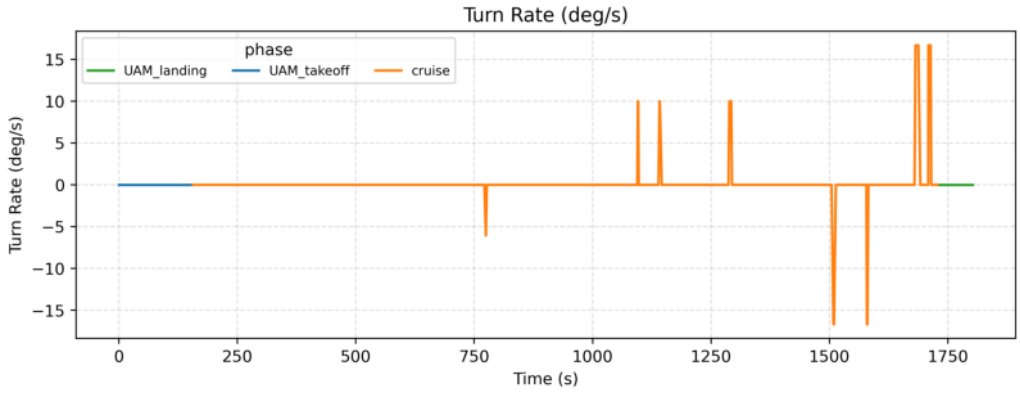


Figure 7–5: Planned turn-rate profile $\dot{\psi}(\tau)$ for Trajectory A (deg/s).

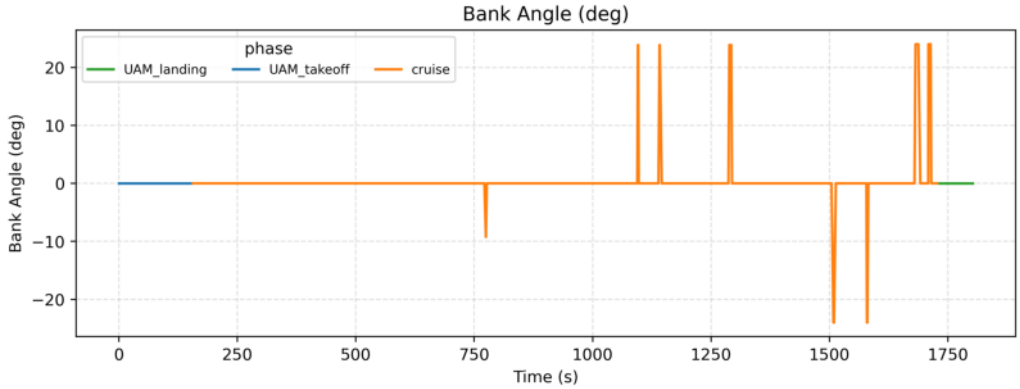


Figure 7–6: Planned bank-angle profile $\phi(\tau)$ for Trajectory A (deg).

Interpretation of bank angle and turn-rate profiles (CFMS).

At each turn waypoint, the inbound and outbound ground tracks define the required course change $\Delta\chi$ and the turn direction. CFMS then synthesizes a dynamically feasible turn using the aircraft performance limits (maximum load factor and bank-rate limit) at the current *ground-speed* V_g (speed over ground, i.e., the magnitude of the ground-relative velocity vector) [6]. The constructed turn has three phases: bank-up, constant-bank arc, and bank-down. A bank-rate bound $\dot{\phi}_{\max}$ shapes the transients; if the transients would exceed the geometric corner, CFMS reduces the allowable bank to fit the fillet.

The bank ceiling is set by the maximum load factor $n_{\max}$ under coordinated-turn assumptions:

$$\phi_{\max} \;=\; \arccos\!\left(\frac{1}{n_{\max}}\right). \tag{7–1}$$

The instantaneous turn rate follows coordinated-turn kinematics:

$$\dot{\psi}(t) \;=\; \frac{g\,\tan(\phi(t))}{V_g(t)}\,, \tag{7–2}$$

so, for a given V_g, the largest achievable turn rate is

$$\dot{\psi}_{\max} \;=\; \frac{g\,\tan(\phi_{\max})}{V_g}\,. \tag{7–3}$$

Consistency check for reported peaks. Using the study's typical value $n_{\max} = 1.1$ (small UAS),

$$\phi_{\max} \;\approx\; \arccos\!\left(\frac{1}{1.1}\right) \;\approx\; 24.6^{\circ}, \tag{7–4}$$

which is consistent with the observed $\max\phi \approx 24.0^{\circ}$. At a representative transitional groundspeed $V_g \approx 15\ \mathrm{m/s}$,

$$\begin{aligned}\dot{\psi}_{\max} \;&\approx\; \frac{9.81\,\tan(24^{\circ})}{15}\ \mathrm{rad/s}\\ &\approx\; 0.29\ \mathrm{rad/s} \;\approx\; 16.7^{\circ}/\mathrm{s},\end{aligned} \tag{7–5}$$

matching the reported $\max\dot{\psi} \approx 16.68^{\circ}/\mathrm{s}$. (When CFMS's simplified point-to-point routine is used, a capped constant turn rate may be applied instead of the full filleted turn.)

Summary Statistics

Table 7–1: Summary statistics for the planned reference Trajectory A.

Points	500
Total time T (s)	1802.71
Total time T (min)	30.05
Total distance L (km)	32.13
Mean TAS (m/s)	17.88
Peak bank angle $\max\phi$ (deg)	24.01
Peak turn rate $\max\dot{\psi}$ (deg/s)	16.68
Peak climb rate $\max\dot{H}$ (m/s)	6.10

7.2.2 Representative Short-Leg Route

Trajectory B was chosen to illustrate the behavior of the system when the initial leg is relatively short. In this case, the UAM vehicle has limited horizontal distance available during the first segment, requiring it to execute a circular turning maneuver in order to gain sufficient altitude and align with the required waypoint. This scenario highlights the interaction between climb dynamics and network geometry, making it a complementary test case to Trajectory A.

Route Overview

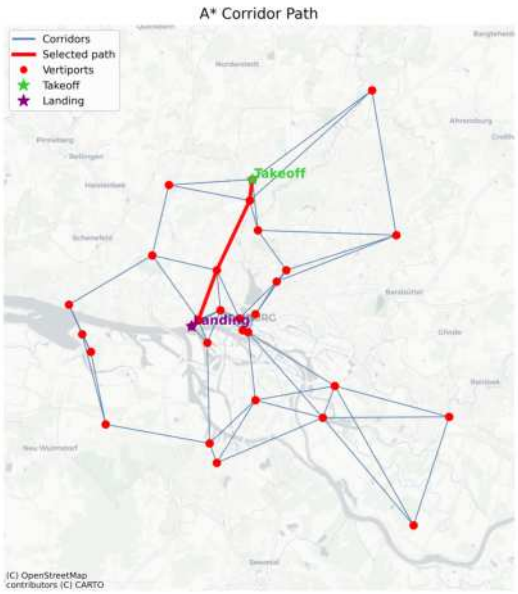
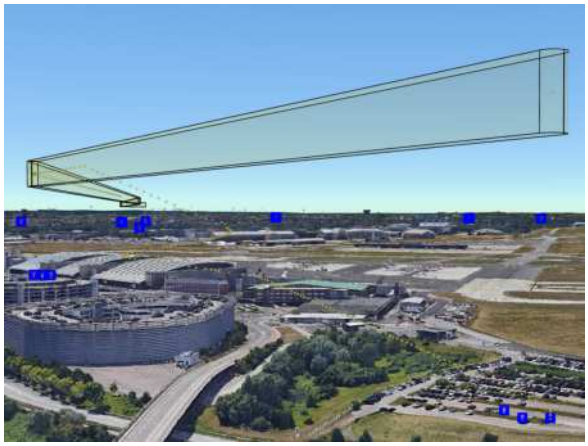


Figure 7–7: Planned reference trajectory (Trajectory B) from s to d over the Hamburg corridor network.

Front-end Visualization (Cesium)

Figure 7–8 shows the planned Trajectory B in the Cesium-based front end. The left panel displays the full 3D route, while the right panel zooms into the departure phase, illustrating the short initial leg and the early turning manoeuvre required to reach the first waypoint.

(a) Planned route (Trajectory B): full 3D corridor context.



(b) Planned route (Trajectory B): zoom on take-off and initial turning manoeuvre.

Figure 7–8: Cesium-based visualization of the planned Trajectory B in the front end. Left: full route; right: take-off segment.

Kinematic Profiles

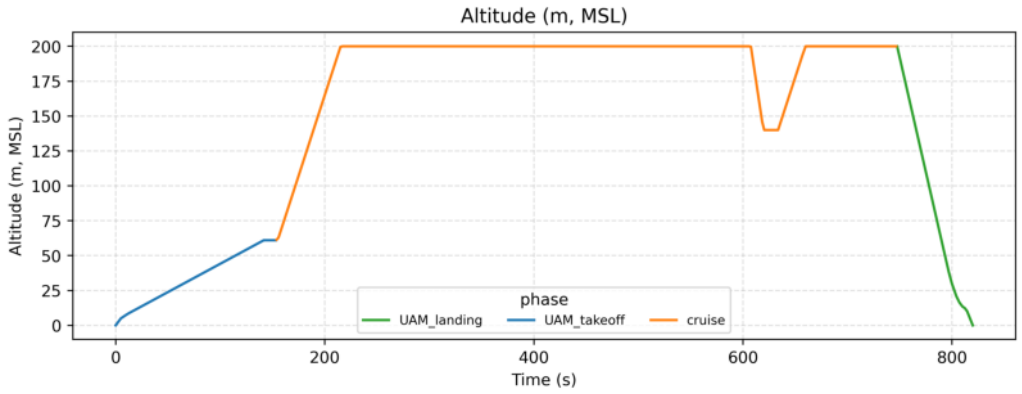


Figure 7–9: Planned altitude profile $H(\tau)$ for Trajectory B (cruise layer near $200\,\mathrm{m}$ AGL).

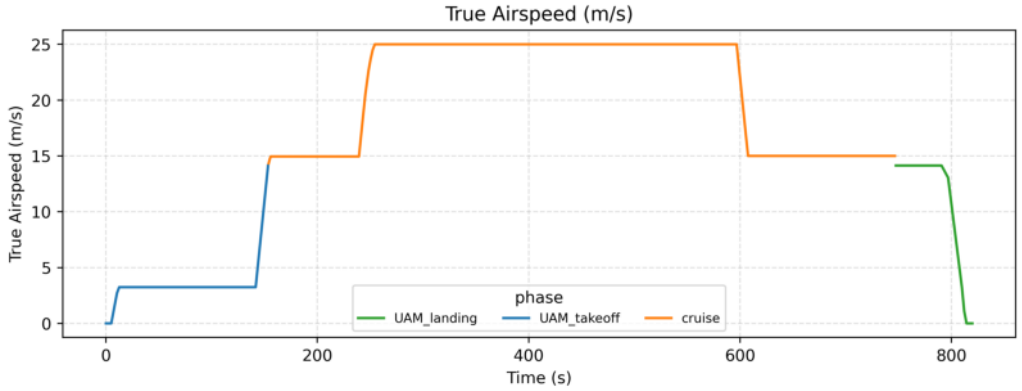


Figure 7–10: Planned true airspeed (TAS) profile for Trajectory B.

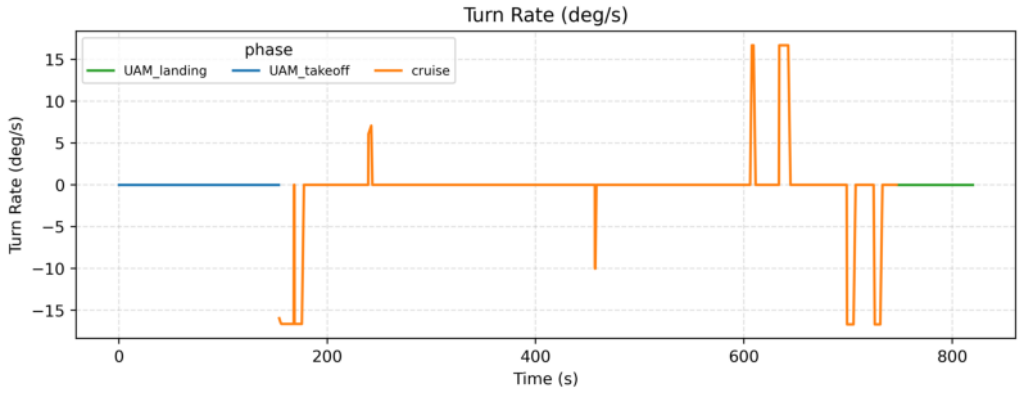


Figure 7–11: Planned turn-rate profile $\dot{\psi}(\tau)$ for Trajectory B (deg/s).

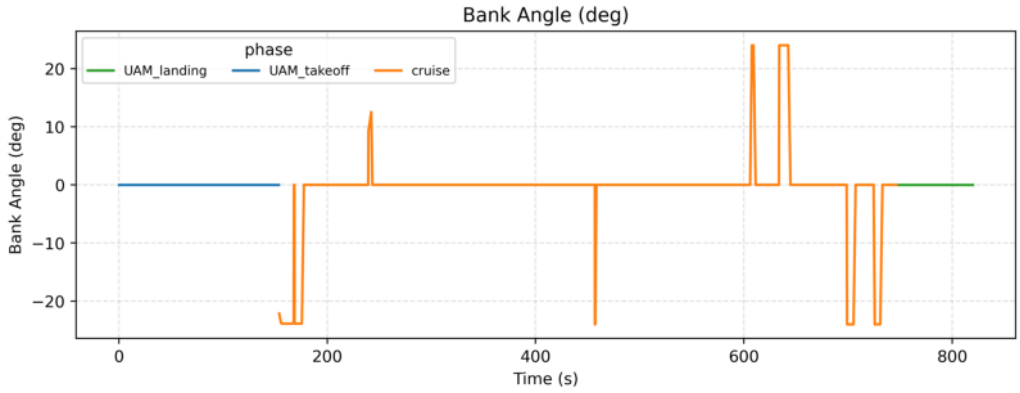


Figure 7–12: Planned bank-angle profile $\phi(\tau)$ for Trajectory B (deg).

Summary Statistics

Table 7–2: Summary statistics for the planned reference Trajectory B.

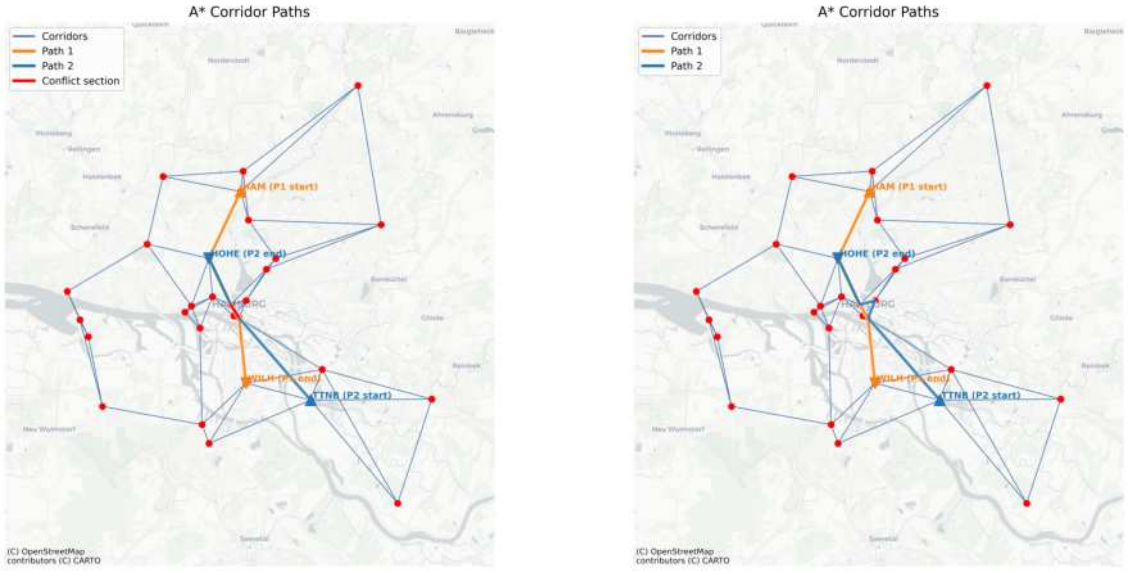
Points	500
Total time T (s)	819.81
Total time T (min)	13.66
Total distance L (km)	13.79
Mean TAS (m/s)	16.67
Peak bank angle $\max \phi$ (deg)	24.01
Peak turn rate $\max \dot{\psi}$ (deg/s)	16.68
Peak climb rate $\max \dot{H}$ (m/s)	6.63

7.3 Conflict Handling

In this section, the ability of the framework to detect and resolve conflicts is demonstrated. Two representative trajectories were selected within the Hamburg corridor network (trajectory `HAM` $\rightarrow$ `WILH` and trajectory `TTNB` $\rightarrow$ `HOHE`), whose initially planned routes overlap spatially and temporally. This setup provides a suitable test case for illustrating both the detection of a conflict and its subsequent resolution through lateral re-routing.

7.3.1 Conflict Detection

The candidate routes were submitted to the NDMap conflict detection module (Sec. 6.3), which evaluates trajectory pairs in four-dimensional space (longitude, latitude, altitude, time). Figure 7–13a shows the initially planned routes, with the overlapping edge segment highlighted in red. NDMap reported a conflict interval $[\tau_{\text{start}}, \tau_{\text{end}}] = [654.2\,\text{s}, 664.8\,\text{s}]$, indicating that both UAM trajectories would occupy the same corridor segment within a temporal overlap of approximately $10.6\,\text{s}$.



(a) Initial planned routes for HAM → WILH and TTNB → HOHE; the overlapping planned corridor segment (conflict) is marked in red.

(b) Re-planned conflict-free route for TTNB → HOHE; the revised planned path avoids the previously conflicting corridor.

Figure 7–13: Comparison of conflict detection and resolution using NDMap and rerouting on planned routes.

7.3.2 Conflict Resolution

To resolve the detected conflict, the integrated rerouting strategy within the path planning framework (Sec. 6.1) was applied. Edges involved in the conflict were temporarily penalized by assigning infinite cost in the corridor graph, effectively discouraging their selection in subsequent planning iterations. A new A* search was then performed for the affected trajectory (TTNB → HOHE), yielding a longer but conflict-free path. The outcome is shown in Fig. 7–13b, where the rerouted trajectory bypasses the overlapping corridor segment, thereby eliminating the temporal and spatial conflict.

Summary

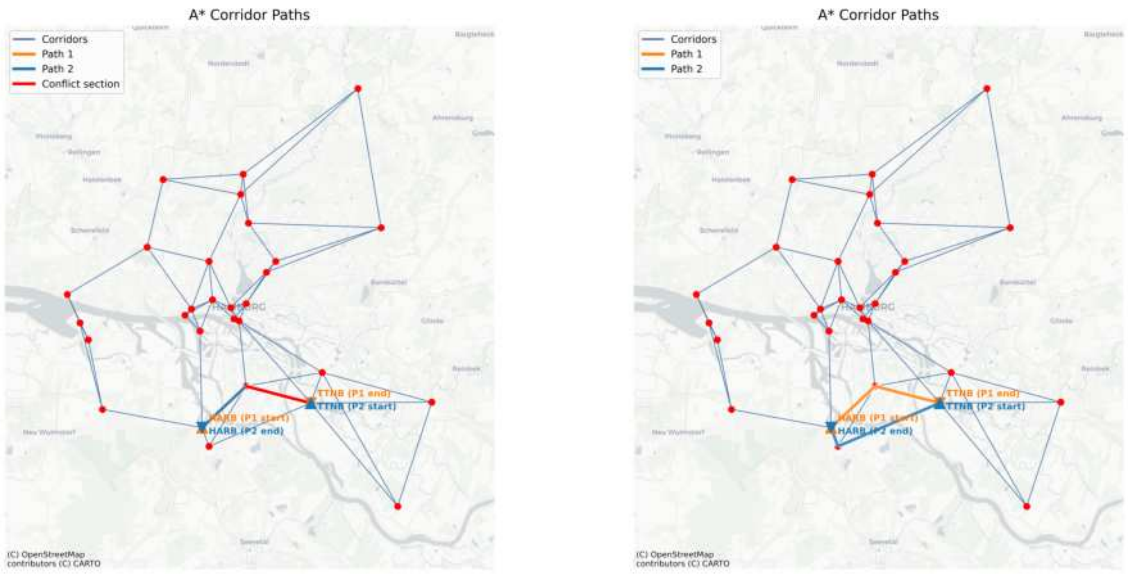
This case study demonstrates the integrated conflict handling approach of the framework: conflicts are detected via NDMap (Sec. 6.3), and if present, the route is adaptively recomputed by modifying corridor edge weights and re-running A* search (Sec. 6.1). Although this may increase the total distance and travel time, the resulting trajectory is conflict-free, supporting safe and scalable multi-UAM operations within the Hamburg scenario.

7.3.3 Conflict Handling with Interchanged Endpoints

A second case study considers two trajectories with interchanged start and end vertiports: trajectory HARB → TTNB and trajectory TTNB → HARB. In the initial planning stage, both trajectories select the same shortest route, leading to a complete overlap of the corridor segment between the two vertiports. This results in an unavoidable spatial and temporal conflict when both plans are submitted simultaneously.

Conflict Detection and Resolution

Figure 7–14 compares the initial conflicting routes (left) with the resolved case (right). NDMap (Sec. 6.3) flagged the overlapping corridor segment (red) as a conflict, with simultaneous occupancy intervals for both trajectories. To resolve this, the integrated rerouting strategy (Sec. 6.1) was applied: the conflicting edge was penalized by assigning it infinite cost, and a new A* search was run for the later-submitted trajectory (TTNB → HARB). The new solution yields the second-shortest path in the network, which avoids the overlapping corridor and ensures conflict-free operation.



(a) Initial planned routes for interchanged endpoints: both planned paths overlap completely on the shortest corridor segment (conflict).

(b) Re-planned conflict-free route for TTNB → HARB along the second-shortest corridor path, eliminating overlap.

Figure 7–14: Conflict handling for interchanged endpoints on planned routes. The later trajectory is re-planned to avoid the overlapping corridor.

Summary

This case demonstrates the robustness of the rerouting mechanism in situations where direct shortest paths are infeasible due to complete overlap. By enforcing the second-shortest path in the graph, the framework maintains safety while preserving operational feasibility in dense urban networks.

7.4 Scalability Evaluation with 100 Trajectories

To assess the computational scalability of the proposed framework, a batch of 100 synchronous trajectory requests was processed. In this setup, each request was completed before the next one was started. Independent pairs of take-off and landing vertiports were randomly selected, and successful trajectories were used for evaluation. Conflict detection and resolution mechanisms were disabled in this experiment to isolate the baseline computational performance.

Summary Statistics

The mean and standard deviation of the measured runtimes across the 100 test cases are summarized in Table 7–3. The 100 cases used distinct pairs of take-off and landing vertiports within the Hamburg vertiport network employed in this study, whose corridor graph was fully connected. Because conflict resolution was disabled for this evaluation and the graph connectivity guaranteed a path between any pair, all 100 planned trajectories were successfully generated.

Table 7–3: Runtime summary over 100 synchronous trajectory requests.

Metric	Mean (ms)	Std (ms)
Total time	162.92	38.38
Trajectory generation	151.43	38.06
A* path finding	0.41	1.21

Scatter Plot Analysis

Figure 7–15 shows the relationship between A* runtime and *path nodes (count)*. No meaningful correlation was observed (e.g., $r \approx 0.02$), confirming that A* search latency is essentially independent of the number of nodes traversed. In contrast, Fig. 7–16 demonstrates a strong positive correlation (e.g., $r \approx 0.75$) between *path nodes (count)* and trajectory generation time, consistent with the generator's per-segment workload.

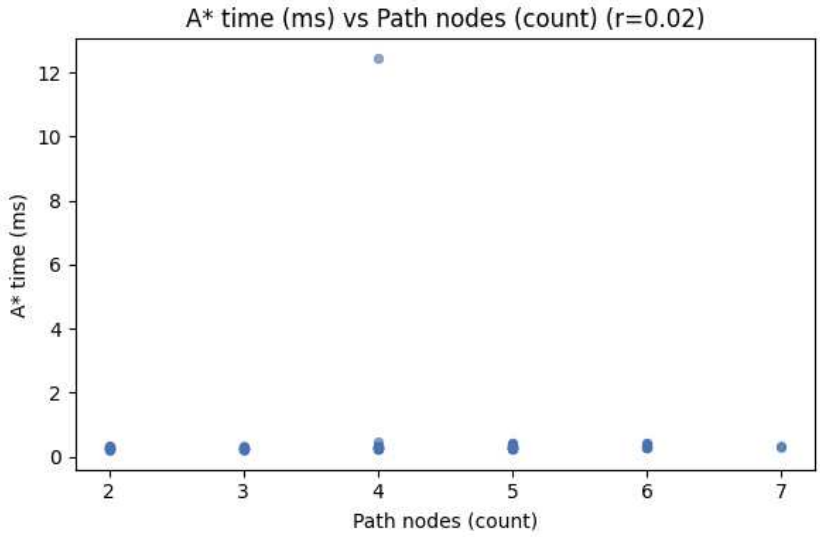


Figure 7–15: A* runtime vs. path nodes (count). Example correlation: $r \approx 0.02$.

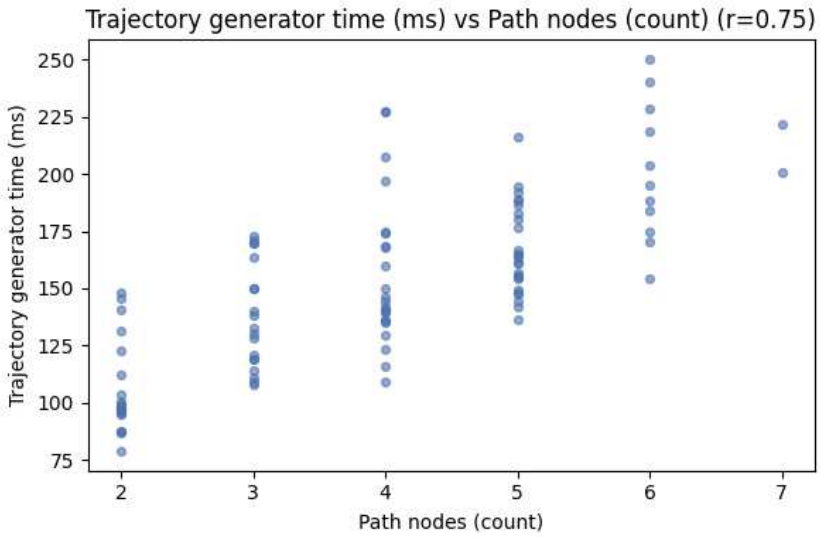


Figure 7–16: Trajectory generator runtime vs. path nodes (count). Example correlation: $r \approx 0.75$.

Runtime Distributions

The distribution of total processing time across the 100 runs is shown in Fig. 7–17, with values concentrated in the $120\,\mathrm{ms}$ to $260\,\mathrm{ms}$ range. The distribution of trajectory generation runtimes (Fig. 7–18) shows a similar pattern, confirming that trajectory generation is the dominant contributor to total latency.

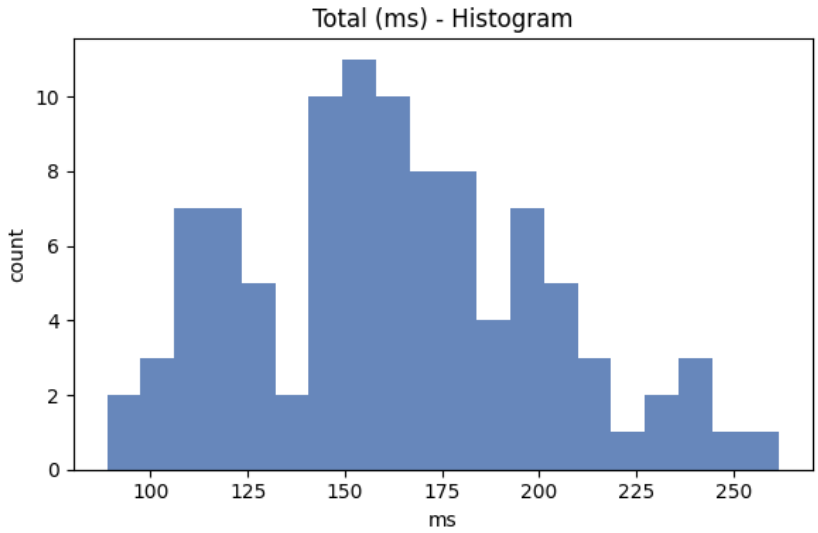


Figure 7–17: Histogram of total runtime per trajectory request.

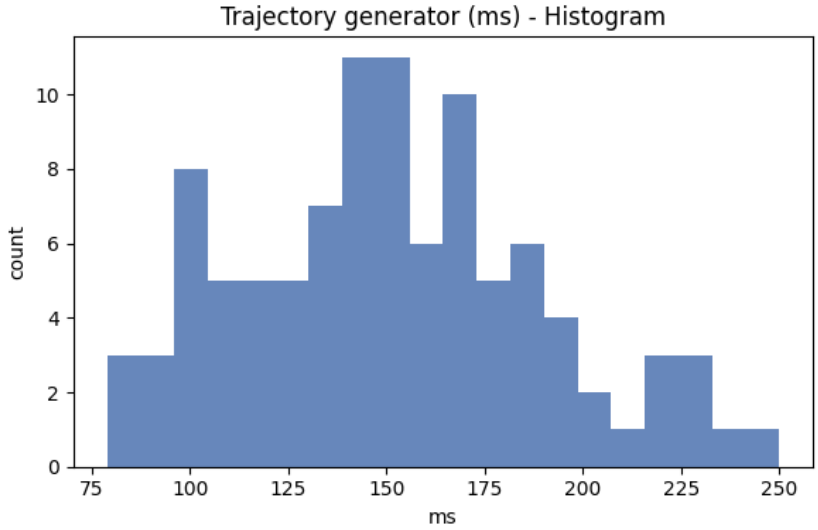


Figure 7–18: Histogram of trajectory generation runtime per request.

Discussion

These results demonstrate that the A* stage contributes negligibly to total latency, with mean runtimes well below $1\,\mathrm{ms}$ across the sampled path sizes. In contrast, trajectory generation dominates the runtime and increases with the number of path nodes (and thus legs to synthesize). Overall processing times of roughly $160\,\mathrm{ms}$ per request indicate that the framework can support real-time urban air mobility (UAM) trajectory planning in a synchronous, single-request setting. Scaling to parallel, high-volume scenarios will require further evaluation under active conflict detection and resolution.

7.5 Evaluation under Active Conflict Resolution

In addition to the synchronous evaluation without conflict detection and resolution (Sec. 7.4), the framework was further assessed under active conflict handling. Fifteen random takeoff–landing pairs were selected from the Hamburg corridor network. For each path request, the conflict detection module (Sec. 6.3) and the lateral re-routing strategy (Sec. 7.3) were engaged. This setup allows measuring the combined latency of A* search, trajectory generation, and any re-planning iterations required to produce a conflict-free trajectory.

7.5.1 Processing Strategy

All 15 trajectory requests were processed synchronously, i.e. each request was completed before the next one started. This mode isolates the latency of the path-planning pipeline without concurrent scheduling effects. If NDMap detected a conflict, the edges involved were penalized and A* was re-run (Sec. 7.3), which may yield longer paths and increased runtimes. Two cases could not be resolved by lateral re-routing alone and remained in conflict, indicating that an alternative strategy (e.g. altitude change or departure delay) would be necessary.

7.5.2 Timing Results

Across the 13 successfully resolved trajectories, the total end-to-end latency (from path request to final 4D trajectory) exhibited a wide spread:

- **Count:** 13 trajectories,
- **Mean total time:** $5286.19\,\mathrm{ms}$,
- **Standard deviation:** $8003.16\,\mathrm{ms}$.

Two effects plausibly contribute to this variance: (i) the statistics are based on a small sample ($n = 13$), which makes the estimate of dispersion sensitive to a few long-running cases; and (ii) part of the measured duration includes inter-service I/O overhead, since trajectory samples are exchanged via intermediate files rather than through an in-process interface.

Figure 7–19 shows the scatter plot of total time versus path distance for these trajectories. A positive correlation $r = 0.76$ indicates that longer paths tend to require more processing time, reflecting both increased trajectory-generation cost and additional A* iterations in some cases.

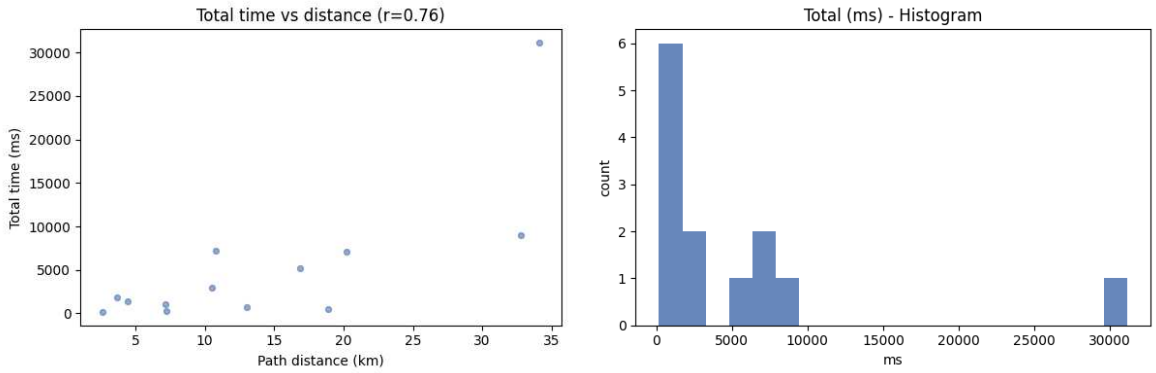


Figure 7–19: Left: total processing time versus path distance for 13 conflict-free planned trajectories. Right: histogram of total processing times across the same runs.

7.5.3 Discussion

Compared to the evaluation without conflict resolution (Sec. 7.4), both the mean latency and its variance are markedly higher under active conflict handling. This is expected, as each detected conflict triggers additional A* searches with modified edge weights (Sec. 7.3), potentially increasing path length and computation time. The two unresolved cases underscore the need for more flexible resolution mechanisms—such as altitude assignment, departure delay, or risk-map based edge re-weighting—to handle complex conflicts where lateral re-routing alone cannot find a feasible solution.

This evaluation thus highlights the trade-off between safety and responsiveness: while lateral re-routing enables automatic deconfliction in many scenarios, its runtime cost scales with graph complexity and the number of iterations required. The observed variance is further influenced by the small sample size and by the current file-based exchange between services; tighter in-process coupling (e.g., integrating CFMS into the planner) would plausibly reduce such overhead.

8 Conclusions and Future Work

8.1 Conclusions

This thesis has presented the design, implementation, and evaluation of a microservice-based path planning framework for urban air mobility (UAM) operations. By decomposing the system into independent services, the planner was able to integrate heterogeneous components implemented in different languages and tools—namely, the A* global planner (Python), the CFMS trajectory generator (C++), and the NDMap conflict detection module (C++). This modular design demonstrates the potential of microservices for building scalable and extensible architectures for U-space operations.

The framework was evaluated using a representative vertiport and corridor network modeled over the metropolitan area of Hamburg. Experiments focused on quantifying delays at different processing stages, including A* path finding, trajectory generation, and overall end-to-end response time. Results showed that:

- The A* path finding step contributed negligibly to the total delay, with runtimes consistently below 1 ms.
- Trajectory generation dominated the computational cost, with runtimes scaling linearly with path distance.
- The total processing time per trajectory request was on the order of 200 ms, demonstrating the feasibility of near real-time operation for single-request scenarios.
- Conflict detection and resolution were successfully demonstrated using NDMap, with lateral rerouting as the chosen strategy to avoid overlapping trajectories.

Overall, the system provides a primitive but functional prototype of a path planner for UAM operations, highlighting both the feasibility and the challenges of integrating multiple specialized modules into a unified framework.

8.2 Future Work

While the present study establishes a foundation, several extensions are necessary to bring the framework closer to realistic deployment:

Enhanced Conflict Resolution

The current work focused exclusively on lateral rerouting through penalization of conflicting edges in the corridor graph. Future extensions should include additional resolution strategies such as:

- **Temporal deconfliction:** delaying departure times to avoid simultaneous use of a corridor.
- **Vertical deconfliction:** assigning different altitude layers to conflicting trajectories.
- **Hybrid strategies:** combining lateral, temporal, and vertical maneuvers to balance efficiency and safety.

Complex Airspace Modeling

The current corridor network was static, uniform in altitude, and of fixed width. Future studies should incorporate:

- Multiple altitude layers with dynamic assignment,
- Variable corridor widths depending on traffic density or risk,
- More realistic urban scenarios with larger networks and higher traffic volumes.

Incorporation of External Factors

To better approximate real-world operations, future versions should integrate:

- **Weather effects:** wind, turbulence, and no-fly zones affecting edge weights and feasibility.
- **Risk maps:** penalizing corridors near sensitive areas (schools, hospitals, etc.) to reflect societal and regulatory constraints.
- **Dynamic re-weighting:** adjusting corridor costs in real time based on environmental and operational factors.

Aircraft Performance Models

The present study employed simplified kinematic constraints. For improved fidelity, future work should model:

- Performance envelopes specific to different eVTOL types,
- More detailed climb, descent, and turn dynamics,
- Energy consumption models for battery-electric aircraft.

Scalability and Multi-Agent Evaluation

The synchronous evaluation with 100 independent trajectory requests provided baseline performance characteristics. Future studies should investigate:

- Asynchronous and parallel trajectory requests at scale,

- End-to-end system latency under high traffic densities.

In summary, this thesis demonstrates that a microservice-based approach to UAM path planning is both technically feasible and computationally efficient for baseline scenarios. Expanding the framework with more advanced conflict resolution strategies, richer airspace and aircraft models, and larger-scale evaluations will be critical steps toward enabling safe, scalable, and realistic integration of UAM operations into the future air traffic management ecosystem.

References

[1] European Union Aviation Safety Agency (EASA), “Easy access rules for u-space — may 2024”, Tech. Rep., 2024. [Online]. Available: https://www.easa.europa.eu/en/downloads/139563/en (visited on 04/16/2025).

[2] L. Matlekovic, F. Juric, and P. Schneider-Kamp, “Microservices for autonomous uav inspection with uav simulation as a service”, *Simulation Modelling Practice and Theory*, vol. 119, p. 102 548, 2022. DOI: 10.1016/j.simpat.2022.102548. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22000466.

[3] “U-space blueprint”, European Commission, Tech. Rep., 2017. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e501ba2683&appId=PPGMS (visited on 08/03/2025).

[4] T. Prevot, J. Rios, P. Kopardekar, *et al.*, “Uas traffic management (utm) concept of operations to safely enable low altitude flight operations”, in *AIAA Aviation Forum*, 2016. DOI: 10.2514/6.2016-3292.

[5] SESAR Joint Undertaking, “Sesar concept of operations (conops 2019)”, Tech. Rep., 2019. [Online]. Available: https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c91e877d&appId=PPGMS (visited on 06/13/2025).

[6] F. Morscheck, “A modular experimental flight management and 4d trajectory generation system for unmanned multicopter, urban air mobility vehicles and other vtol vehicles”, in *IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–9. DOI: 10.1109/DASC52595.2021.9594290.

[7] A. Kuenz, *High Performance Conflict Detection and Resolution for Multi-Dimensional Objects* (DLR Forschungsbericht 31). Oct. 2015. [Online]. Available: https://elib.dlr.de/98476/.

[8] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, “Microservices: Yesterday, today, and tomorrow”, in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Springer International Publishing, 2017, pp. 195–216. DOI: 10.1007/978-3-319-67425-4_12. [Online]. Available: https://doi.org/10.1007/978-3-319-67425-4_12.

[9] “Sesar master plan 2025”, SESAR Joint Undertaking, Tech. Rep., 2025. [Online]. Available: https://www.sesarju.eu/sites/default/files/documents/reports/SESAR%20Master%20Plan%202025.pdf (visited on 09/16/2025).

[10] G. Enea and M. Porretta, “A comparison of 4d-trajectory operations envisioned for nextgen and sesar, some preliminary findings”, vol. 5, pp. 4152–4165, 2012.

[11] J. J. Acevedo, C. Capitán, J. Capitiin, *et al.*, “A geometrical approach based on 4d grids for conflict management of multiple uavs operating in u-space”, in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 263–270. DOI: 10.1109/ICUAS48674.2020.9213929.

[12] C. Capitán, H. León, J. Capitán, *et al.*, “Unmanned aerial traffic management system architecture for u-space in-flight services”, *Applied Sciences*, vol. 11, p. 3995, 2021. DOI: 10.3390/app11093995.

[13] *Cesiumjs documentation and tutorials (api reference)*, https://cesium.com/learn/cesiumjs/ref-doc/, 2025. (visited on 09/16/2025).

[14] L. R. Sahawneh, M. E. Argyle, and R. W. Beard, “3d path planning for small uas operating in low-altitude airspace”, in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 413–419. DOI: 10.1109/ICUAS.2016.7502528.

[15] C. A. Pötter Neto, G. de Carvalho Bertoli, and O. Saotome, “2d and 3d a* algorithm comparison for uas traffic management systems”, in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020, pp. 72–76. DOI: 10.1109/ICUAS48674.2020.9214028.

[16] X. Hou, F. Liu, R. Wang, and Y. Yu, “A uav dynamic path planning algorithm”, in *35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2020, pp. 127–131. DOI: 10.1109/YAC51587.2020.9337581.

[17] A. H. Ahmad, O. Zahwe, A. Nasser, and B. Clement, “Path planning algorithms for unmanned aerial vehicle: Classification, performance, and implementation”, in *3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 2023, pp. 1–6. DOI: 10.1109/ICECCME57830.2023.10252168.