Technische Universität München
TUM School of Engineering and Design

Deterministic and Contingency-Aware Motion Planning for UAVs over Congested Areas in VLL Airspace

Markus Ortlieb

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr. – Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Sophie Armanini

Prüfer*innen der Dissertation:

1. Prof. Dr.-Ing. Florian Holzapfel
2. Prof. Dr. ir. Maarten Uijt de Haag

Die Dissertation wurde am 05.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 30.04.2023 angenommen.

Eidesstattliche Erklärung

Hiermit erkläre ich, Markus Ortlieb, gegenüber dem Lehrstuhl für Flugsystemdynamik der Technischen Universität München, dass ich die vorliegende Dissertation selbstständig und ausschließlich unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen angefertigt habe.

Die Arbeit wurde in gleicher oder ähnlicher Form an keiner anderen Hochschule oder Universität vorgelegt.

Garching, 03. Januar 2023

Markus Ortlieb

Kurzfassung

Die vorliegende Arbeit entwickelt und beschreibt eine funktionale Bahnplanungsarchitektur, welche die Anforderungen, die sich aus bereits veröffentlichten und in der Entwicklung befindlichen Standards für den pilotierten (e)VTOL- und UAV-Betrieb außerhalb der Sichtweite (BVLOS) und über städtischem Gebiet in Europa ergeben, adressiert. Die identifizierten Anforderungen werden auf den automatisierten Flugbetrieb übertragen und Konzepte zur automatisierten Missionsplanung für UAVs und (e)VTOLs erarbeitet. Die Architektur zerlegt eine komplexe Planungsaufgabe in mehrere kleinere Planungsaufgaben geringerer Komplexität, für die dedizierte Planungsalgorithmen entwickelt und bereitgestellt werden. Die Lösung der Gesamtplanungsaufgabe ergibt sich aus der Kombination der Lösungen der Teilprobleme. Basierend auf der zuvor beschriebenen Architektur werden Bahnplanungsverfahren für UAVs in bodennahem Luftraum über städtischem oder anderweitig risikobehaftetem Terrain abgeleitet. Die Komplexität und Dimension des Planungsproblems wird durch eine umfangreiche Vorverarbeitung des Planungsraumes und unter Anwendung eines dem Planungsproblems zugrunde liegenden Regelwerks reduziert. Hierbei wird sichergestellt, dass alle Lösungen der im Planungsraum verbleibenden Lösungsmenge nach operationellen und dynamischen Kriterien sicher sind, wodurch die Machbarkeit der Planungslösung garantiert wird. Durch Ausnutzung der Vorausplanung kann innerhalb des reduzierten Lösungsraums eine Onlineplanungsfähigkeit erreicht werden. Ziel der beschriebenen Algorithmen und Verfahren ist, im Rahmen der geltenden Regulatorik nachweislich regelkonforme und fliegbare Trajektorienmengen zu identifizieren, welche auch unter Einbezug unvorhergesehener Ereignisse jederzeit eine sichere Missionsdurchführung erlauben.

Abstract

This work develops and describes a functional path planning architecture, which addresses the requirements that arise from recently published and ongoing rulemaking activities for piloted (e)VTOL and UAV operations beyond the visual line of sight (BVLOS) and over congested areas in Europe. It further projects these requirements into the field of automated vehicle operation to enable automated mission planning for UAVs and (e)VTOLs under EASA's SC-VTOL Enhanced and certified category for UAS operations. The architecture breaks a complex planning task into multiple smaller and less complex planning tasks, for which it provides dedicated planning algorithms. The solution to the overall problem is generated from the combination of the subtasks' solutions. From the developed framework, motion planning algorithms for use in very low level airspace over congested areas are derived. The complexity and dimension of the planning task is reduced by extensive preprocessing of the planning space using a set of underlying rules to the planning problem. It is ensured that all solutions in the set of remaining solutions in the planning space are safe according to operational and dynamic criteria, by which the feasibility of the final planning solution is guaranteed. The approach exploits the prior offline-planning phase to achieve online planning capabilities within the reduced solution space. The described algorithms and procedures aim to identify verifiably rule-compliant and flyable trajectory sets within the scope of an applicable regulatory framework and to enable safe mission execution even when unforeseen events during flight are considered. The developed methods are validated in planning scenarios, which reflect potentially realistic future UAV operations in terms of complexity of the environment, network size, and spatial dimension.

Table of Contents

List of Figures

List of Tables

Table of Acronyms

Acronym	Description
ACAS	Aerial Collision Avoidance System
ADS-B	Automatic Dependent Surveillance - Broadcast
AGL	Above Ground Level
AMC	Acceptable Means of Compliance
API	Application Programming Interface
ARC	Air Risk Class
AROC	Automated Reachset Optimal Control
ARP	Aerospace Recommended Practise
ATC	Air Traffic Control
ATS	Air Traffic Service
BVLOS	Beyond Visual Line of Sight
C2	Command and Control
ConOps	Concept of Operations
CS	Certification Specification
CSFL	Continued Safe Flight and Landing
CU	Control Unit
DAA	Detect and Avoid
DAL	Design Assurance Level
DB	Database
DOF	Degree of Freedom
DSM	Digital Surface Model
EASA	European Union Aviation Safety Agency
EUROCAE	European Organisation for Civil Aviation Equipment
EVLOS	Extended Visual Line of Sight
FDAL	Functional Design Assurance Level
FTE	Flight Technical Error
FL	Flight Level
(i)GRC	(intrinsic) Ground Risk Class
ICAO	International Civil Aviation Organization
IFR	Instrument Flight Rules
IRIS	Iterative Regional Inflation by Semidefinite programming
JARUS	Joint Authorities for Rulemaking on Unmanned Systems
MOC	Means of Compliance
NSE	Navigation System Error
OSO	Operational Safety Objective
PDF	Probabilistic Density Function

Acronym	Description
RMT	Rule-Making Task
RPAS	Remotely Piloted Aerial System
SC	Special Condition
SERA	Standardized European Rules of the Air
SAE	Society of Automotive Engineers
SAIL	Specific Assurance and Integrity Level
SORA	Specific Operations Risk Assessment
TMPR	Tactical Mitigation Performance Requirements
TUM	Technische Universität München
TSE	Total System Error
UA	Unmanned Aircraft
UAM	Urban Air Mobility
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
USSP	U-Space Service Provider
VFR	Visual Flight Rules
VLL	Very Low Level
VLOS	Visual Line of Sight
VP	Vertiport
(e)VTOL	(electric) Vertical Take-off and Landing

Table of Symbols

Latin Letters		
Symbol	**Unit**	**Description**
A	$-$	geometric arc
$\mathcal{C}$	$-$	configuration space
C	$-, -, - , -$	abstract configuration in the configuration space, risk composite function, total cost, image matrix
E	$-, -$	set of edges, ellipsoid
F	$-$	(vertiport) funnel
$\mathcal{G}$	$-$	graph
H	$-, -$	continuous state space, altitude profile
J	$-$	cost function
K	$-$	neighbourhood threshold
L	$m, -, -, -$	absolute length, (abstract) extension length, left Dubins primitive, data layer
M	$-, -$	geometric centre, obstacle map
$\mathcal{N}$	$-$	navigation function
N	$-, -$	set size, (tree) branch factor
$\mathcal{O}$	$-$	obstacle set
$\mathcal{P}$	$-$	set of waypoints
P	$-, -$	probability, (geometric) point
Q	$-$	motion library
R	$m, -$	radius, right Dubins primitive
Re	$-$	reward function
$\mathcal{S}$	$-, -$	search space, planning surface
S	$-, -$	(local) evaluation strategy, straight Dubins primitive
$\mathcal{T}$	$-, -$	set of (partial) trajectories
T	$-, -$	tree of trajectories, transition probability
U	$-$	set of neighbouring states
V	$-, -, -$	visibility domain, set of vertices
W	$-$	set of wavefront states
X	$-$	set of states
a	$-, \frac{m}{s^2}, -$	randomly sampled configuration, acceleration, action
b	m	half wingspan
c	$-, m, m$	(item) cost, resolution, obstacle clearance
d	m, m	distance, translational vector
e	$-$	(graph or path) edge
fp	$-$	fork point

Latin Letters		
Symbol	**Unit**	**Description**
g	$\frac{m}{s^2}$	gravity constant
h	$-$, m	continuous state, altitude
i	$-$	index counter
j	$-$, $\frac{m}{s^3}$	index counter, jerk
k	$-$, $-$, $-$, $-$, $-$	neighbour vertex, guard vertex, scaling factor, safety factor, path length constraint factor
l	m	(item) length
m	kg, $-$, $-$	mass, manoeuvre, index counter
n	$-$, $-$, $-$, $-$	dimension, load factor, expansion node, index counter
p	m, $-$	position/waypoint, penalty factor
q	$-$, $-$	graph vertex, trim state
r	$-$, m	risk, radius
s	$-$	(discrete) state
t	s	time
u	$-$, $-$	neighbouring state, control input vector
v	$\frac{m}{s}$	velocity
w	$-$, $-$	(normalized) weight factor, disturbance vector
x	m, m, $-$	coordinate in the spatial x-dimension, location, system state vector
y	m	coordinate in the spatial y-dimension
z	m	coordinate in the spatial z-dimension
$\bar{z}$	m	flight altitude heuristic

Greek Letters		
Symbol	**Unit**	**Description**
Γ	$-$	manoeuvre cost
Δt_c	s	contingency planning interval
Φ	rad, $-$	roll angle, manoeuvre map
Ψ	rad	heading
α	rad, rad	generic (manoeuvre) angle, first Dubins angle
β	rad, rad	second Dubins angle, free vertiport approach angle
γ	rad, $-$, rad, $-$	path angle, trim cost, third Dubins angle, discount factor
δ	$-$	state transition
ϵ	$-$	acceptance / termination criterion
η	$-$	set of trim primitive parameters
λ	$-$, rad	non-negative coefficient, heading angle towards a goal state
μ	$-$	(stochastic) expectation
ϕ	$-$	potential function
σ	$-$	standard deviation
θ	$-$	generic clock state
τ	$-$, s	path, coasting time
χ	rad	azimuth
ζ	$-$	set of manoeuvre parameters

1 Introduction

1.1 Motivation and Background

The search for new modes of transportation of people and goods to relief ground traffic and reduce congestion in metropolitan areas in recent years has led to the development of a new class of air vehicles and their manufactures, aiming to enable and provide aerial transport across and between cities. These so called (electric) vertical take-off and landing vehicles ((e)VTOLs) typically feature battery-electric propulsion systems and are designed to cover short to medium distances with very small noise footprints also during take-off and landing, such that they can be deployed to transport passengers or cargo into inner cities or residential areas. Current rulemaking activities in Europe and the U.S. foresee (e)VTOL vehicles to be operated in very low level (VLL) airspace at altitudes from 300ft to 1000ft above ground level (AGL) and between so-called vertiports. Vertiports provide a controlled landing environment and the required infrastructure for payload ground handling and battery charging. First urban air mobility (UAM) passenger services between few hubs are expected to be introduced in dedicated cities in the first half of this decade with a human pilot controlling the vehicle on a pre-determined flight path. Similarly, logistics and time-critical delivery of goods create an increasing number of use cases for aerial cargo transport. Early operations will be limited to few profitable business cases by high operating cost and the number of available pilots. The potential of (e)VTOL operations in both passenger and cargo scenarios, however, can only be exploited, if the service can be scaled to a larger vehicle fleet and a growing network of vertiports. With targeted fleet sizes of several 100.000 (e)VTOLs in the 2030s, the automation of vehicles is expected to become an inevitable enabler to scale UAM and (e)VTOL cargo operations.

This new use case for automated air vehicles in the urban environment also introduces new challenges and boundary conditions on the motion planning problem to guide these vehicles. The regulatory framework for automated UAM operations in Europe is described in the certified UAS category of the 'European Aviation Safety Agency (EASA) concept for regulation of Unmanned Aircraft Systems (UAS) operations and Urban Air Mobility'. So called type #2 operations (see [59]) require vehicles to operate on pre-defined routes and fulfil the same safety standards that apply to commercial airliners. Dense operation of many automated aerial vehicles will further require a path planning method to enable the vehicle to react to contingency events at flight time, while continuing to ensure efficiency of the operation. Equally important and a critical component to the success of the UAM concept will be, how a motion planning method can provide seamless integration with existing air traffic and air space structures and guarantee compliance with an applicable set of rules. As seen in different fields, where tasks formerly executed by humans are transferred to an automated or even autonomous control system, the risk perceived by a non-expert consumer plays a critical role for the public acceptance of a new technology, regardless of the system's actual failure rate. The success of automated air transport as a means to exploit the third dimension in urban travel and daily commute may, hence, largely depend on the ability of engineers to design a guidance system, which is not only transparent to the competent authority but also provides sensible communication of the safety aspect to the interested public.

From a technical perspective, the sensor and computational technology to enable safe and efficient automated urban air mobility exists, with clear advantages of an automated system over a human-piloted aircraft expected in terms of safety, operating cost and availability. The functional gap to enable automated flight over congested terrain lies in the current state of UAV motion planning. It lacks algorithms and planning methods, which guarantee that a feasible and regulation-compliant solution to a motion planning problem in a partially known environment will be found in sufficiently short time during flight. This dissertation addresses the identified gap with a planning method, which uses self-imposed limitations and boundary conditions to reduce the complexity of the planning problem. The solution to the simplified planning task combines properties of guaranteed safety to ground and air parties with online planning capabilities.

1.2 State of the Art: Path Planning for UAVs under Consideration of Regulatory Constraints and Risk

This section gives an overview of the literature and the state of the art in UAV motion planning in the specific case of operations under risk and consideration of regulatory boundary conditions. A generic survey of motion planning algorithms and methods for UAVs can e.g. be found in [26] and [12] and shall not be the primary focus of this chapter.

The development of path planning methods designed for UAVs over high-risk areas and in strictly regulated air space is a relatively young research topic. The scope of of this research field is defined by recently published and ongoing regulatory work by EASA in Europe. Major limitations on the design of an appropriate planning method are imposed by EASA's certified category for UAS operations ([59]) and the special condition for small-category vertical take-off and landing aircraft (SC-VTOL, [68], [56]). Vehicles operated under this framework are mandated to follow preplanned trajectories at all times. Further, any landing outside a predetermined and prepared area is defined as a catastrophic event by the regulator.

Suggestions for system architectures for the management of UAM operations on ecosystem level have been made and described in [75] and [8], whereas the challenge of strategic deconfliction between multiple air traffic parties in a network of vertiports are addressed in [28]. While these works focus on UAM operations on ecosystem level, to the best of the author's knowledge at the time of writing this dissertation, there exist no publications targeting a functional architecture for SC-VTOL-compliant path planning on aircraft level, except for the author's own work [47].

Risk-based certification approaches, similar to SORA for the Specific Category ([20] and [33]), but with additional levels of rigour from the existing Part-21 rules in commercial aviation ([19]) are foreseen in current rulemaking activities for the certified UAS category ([59]). This category will be the certification baseline for unmanned flight beyond the visual line of sight (BVLOS) over congested areas. A number of approaches to model different aspects of risk on ground and in the air and incorporate such risk metrics into the path planning have been presented in [74] and [1]. Most publications in this field follow the idea to either combine a risk metric into some sort of cost function ([72], [13] and [77]) or execute the planning algorithm directly on the risk map ([54], [53]). All of the above mentioned leave the question unaddressed, how the computation of large risk maps can be integrated into a path planning architecture such that the planning task always uses an up-to-date risk map while ensuring sufficiently short execution times. This work presents a modular risk modelling approach, which decouples the risk map computation from the planning task.

Methods specifically developed to provide viable solutions to the path planning problem for unmanned vehicles operating in very low level (VLL) air space are described and demonstrated in [62] and [71]. Both publications introduce a vertical discretization of the configuration space in terms of flight levels, which they combine with a three-dimensional roadmap in the former and individual planning graphs on each flight level in the latter case. Roadmaps and flight levels are computed in an offline preprocessing phase and searched at flight time for a path minimizing a cost function. Regulatory constraints are not treated and respected by design of the method but can be considered as boundary conditions or cost items. This design choice affects the methods' properties such, that they cannot guarantee feasibility and rule compliance of a solution by design. An approach, which purely focuses on cost functions to implement a path planning method for remotely piloted aerial systems (RPAS) in compliance with existing regulation for a specific application scenario in Italian airspace is presented in [29]. In extension to the above-mentioned approaches, the authors of [61] introduce a graph structure, which equally spans across different flight levels, but possesses a structure similar to road networks. Where a regular graph would have a single vertex at the intersection of multiple edges, a directed structure resembling round-abouts is proposed, in which two agents meeting at a graph vertex can deconflict. While enabling simple deconfliction between multiple vehicles, the approach is limited in scale by the maximum number of aircraft per round-about to maintain a well-clear state. It therewith enforces flight paths with non-optimal efficiency in terms of energy and time. This work proposes a deconfliction approach for segregated corridors following the principle of road lanes, instead. No publications related to the herein presented approach to decouple the vertical from the horizontal planning task by computing a flight altitude profile and leveraging this profile to reduce the dimension of the horizontal planning task could be found at the time of writing.

Where a single precomputed flight path may be sufficient for the operation of a single or countably few vehicles in segregated airspace, dense vehicle operations in non-segregated air space require options for action to react to unforeseen contingency events and other air traffic. Planning solutions, which react to changing conditions and provide a trajectory to alternate landing sites at flight time are discussed extensively and demonstrated in the literature in the context of online planning methods. With the above-mentioned restrictions imposed by SC-VTOL and the certified UAS category, the following review of research shall focus on work, which uses at least partially precomputed methods to achieve the described task.

Several publications propose to define alternative landing sites prior to flight, however, differ in the extend to which landing trajectories are computed offline and stored onboard the aircraft. In [31], a planning approach is presented, which computes emergency trajectories to predetermined emergency landing sites. The nearest emergency landing site is selected from a database onboard the UAV and an emergency landing flight path is generated at flight time. The authors of [27] extend the pre-computation phase in a scenario for commercial airliners to the computation of a Voronoi map representing precomputed contingency flight paths. In case of a contingency event, the map is searched for an appropriate flight path to an alternate airport at flight time and an online trajectory generation method converts the identified path into a flyable trajectory. A similar effort, however, in the context of smaller UAVs, is presented in [4], where contingency landing paths to alternate landing sites are computed prior to flight in an offline computation step. From each waypoint of the nominal flight path, a contingency landing path is computed using an A^* algorithm on a discretized terrain map. Although demonstrated in a rural scenario of limited extent, this approach is a first example of SC-VTOL-compliant and contingency-aware flight planning, which computes a sparse tree of trajectories rather than a single flight path. The state of the art focuses on methods, which provide options for safety landings and neglects scenarios, in which the destination remains reachable on an alternative path. Further, the approach described in [27] assumes that the vehicle can reach the nearest boundary of its current Voronoi cell with a straight line at all times, whereas the method presented in [4] fails when a sequence of contingency events occurs.

A prominent example of a system, which leverages partially precomputed planning solutions to simplify an otherwise complex online planning task is the recent collision avoidance systems ACAS X ([34]) and its derivative for unmanned aircraft systems ACAS Xu ([43]). A library of precomputed avoidance manoeuvres is paired with the in-flight evaluation of the aircraft state and selection of appropriate avoidance manoeuvres based on an equally offline-optimized Markov Decision Process.

All the above methods yield a planning solution, which is transparent and deterministic to a varying extent, however, they are all limited in resolution and the number of options for action by the memory resources available on the respective system. One way to address this issue is to increase the safety of online planning algorithms by seeking performance guarantees for the execution of a specific control strategy. Reachability analysis has found its way into the online planning of collision-free paths mainly in one of two applications. One direction of research, of which [23] and [5] are representatives, aims to derive safety margins from the system's reachable set in an offline computation. The system will operate safely in any obstacle environment, which has been augmented with the applicable safety margins. The other group of works synthesizes control laws, which minimize the reachable set of the system when executing a partial trajectory or motion primitive. A library of such controllers enables a system to plan and execute collision-free paths within a specified disturbance envelope through combinations of reach-set-optimized motion primitives. Aggressive manoeuvring of a small UAV using this approach is demonstrated in [25] and [42], whereas various different approaches for the syntheses of controllers with formal guarantees are discussed in [66], [64] and [65].

These approaches can guarantee, that once a path was found, it can be executed safely within the defined limits of the reachability analysis. However, the approach has no effect on the properties of the underlying planning algorithm, nor can it guarantee that a solution is found. Offline-computed navigation functions, which encode the optimal action as a function of the current vehicle state can partially solve this issue, as they enable a greedy online implementation to find a deterministic and near-optimal solution at predictable execution time. A well-known representative of this class of online algorithms is the optimal manoeuvre automaton described in [21].

The review of the literature and state of the art shows, that different solutions exist, which satisfy different subsets of the requirements of the urban path planning problem. This is especially true in the fields of performance guaranteed online planning and risk modelling. However, a research gap exists, where path planning methods are required to enable vehicles to react to unexpected contingency events, while mandated to operate on precomputed flight paths. The present work will extend the scope of previous research in the fields of deterministic and contingency-aware path planning using precomputed paths and partially precomputed online planning methods. It will further present methods, which ensure compliance with regulation by design and combine the individual planning efforts into a UAM planning framework, which satisfies the requirements of the certified UAS category and SC-VTOL.

1.3 Objectives of this Dissertation

This dissertation develops and describes a functional path planning architecture, which addresses the requirements that arise from recently published and ongoing rulemaking activities for piloted (e)VTOL and UAV operations beyond the visual line of sight (BVLOS) and over congested areas in Europe. It further projects these requirements into the field of automated vehicle operation to enable automated mission planning for UAVs and (e)VTOLs under EASA's SC-VTOL Enhanced and certified category for UAS operations. The architecture breaks a complex planning task into multiple smaller and less complex planning tasks, for which it provides dedicated planning algorithms. The solution to the overall problem is generated from the combination of the subtasks' solutions.

Current rulemaking activities for the early adoption of UAM follow two governing principles: First, the evaluation of the intended mission in terms of risk to the ownship and third parties in the air and on ground associated with said mission. And second, the operation of the vehicle exclusively on preplanned routes and between predetermined landing sites, so-called vertiports. In order to satisfy these requirements, the planning architecture divides the path planning problem into an extensive offline planning and a lean online planning phase. The results of the offline planning phase can be inspected before flight, meaning that undesired states or regions in the offline solution, which the vehicle shall not penetrate during the mission, can be removed from the solution prior to take-off. This architecture serves as the starting point for the development of deterministic and contingency-aware path planning methods.

The remainder of this work aims to develop planning algorithms to implement the presented architecture. It exploits the knowledge about the partially known operating environment and applicable rules and regulation to reduce the planning problem's complexity. Based on assumptions on the vehicle performance and operations, a rule-based planning approach is developed, which simplifies the planning task to a degree, that it can be represented as a quasi-two-dimensional path planning problem. During a preprocessing effort, which removes any rule-violating state, the search space is further reduced to only contain the set of valid solutions, such that any solution that is found is guaranteed to be rule-compliant and feasible. Hence, the approach achieves rule-compliance of the planning solution by design. Assurance of feasibility is decoupled from the planning step, which shall instead focus on optimality of the feasible solution.

Based on the governing principle of rule-based planning, this research develops contingency path planning methods and concepts for the conflict-free operation of multiple vehicles inside a narrow, segregated flight corridor. These methods are employed to compute large databases of feasible trajectories prior to flight. It is demonstrated that the urban path planning problem in scope of SC-VTOL and the certified category can, to a large extend, be solved with a decision logic, which selects trajectories from a precomputed database at flight time. The methods and algorithms to compute said database are presented in chapter 5 in this work. In cases, where a higher resolution or more options for action are required, preplanned databases tend to suffer from the curse of dimensionality, caused by exponentially growing memory requirements with increased resolution. To address this shortfall of database-focused planning methods and to provide options for action beyond the limits of memory complexity, a locally constraint online planning method is developed, which operates inside convex regions of obstacle free space. The set of obstacle-free regions is computed offline and stored onboard the aircraft prior to take-off.

The developed planning framework shall be validated in planning scenarios, which reflect realistic future operations in terms of complexity of the environment, network size, and spatial dimension. It is the prime objective of this work to present a methodology to address and solve the path planning problem in the urban environment and discuss how self-imposed limitations support the safety case to an extend that enables automated flight over high-risk environments.

1.4 Statement of Contributions

The main contribution of this dissertation is the development and analysis of a deterministic and contingency-aware motion planning framework for (e)VTOLS and UAVs over congested areas and in VLL airspace in compliance with EASA's SC-VTOL and certified UAS category. Within the implementation of said framework, the following individual contributions beyond the state of the art are made:

C-1: An SC-VTOL and EASA certified UAS category-compliant functional architecture for deterministic and contingency-aware flight planning in VLL airspace over congested areas.

a. Decomposition of the flight planning task into an offline and online phase. In the offline phase, the flight planning is conducted and validated before flight. The online phase collapses the planning task to a trajectory selection problem of lower dimension. Only solutions with guaranteed feasibility are provided in the online phase.

b. Time criticality of online planning tasks is removed by the offline pre-computation.

c. The pre-flight validation of the planning solution ensures compliance with all applicable rules and the target safety level.

C-2: A new method to enforce rule-compliant path planning results in compliance with SC-VTOL and the certified category through a reduction of the search space dimension.

a. Decomposition of the 3D environment into a rule- and regulation-compliant 3D surface that is compatible with the dynamics of the vehicle, and a 2D planning task executed on this surface. This step decouples the assurance of feasibility from the planning step, such that the latter can focus on optimality and will always yield a feasible solution.

b. Compared to the 3D continuum, the planning surface already collapses the search space to a subset of the free configuration space, eliminating one dimension.

c. Edges, which are non-compliant with the rule set, are removed from the search graph, effectively reducing the search space to a graph containing the subset of solutions with guaranteed feasibility.

C-3: A road lane concept for bidirectional operation of point-to-point connections in VLL airspace segregated flight corridors and early U-Space (U1/U2), and procedures for its utilization.

a. Starting from a uni-directional flight path, lanes are generated at safe horizontal and vertical distance, which reflect the vehicle dimensions and estimated total system error (TSE). A 2.5D planning assumption is made on the initial flight path. Flight directions and altitudes are assigned following existing rules of the air (semicircular rule).

b. Vertical separation is achieved using helix manoeuvres in immediate proximity to the vertiport and with a dedicated planning method. With the usage of flight surfaces and discrete flight levels as a discretization of the search space, this method integrates into the overall framework.

C-4: A method for pre-flight computation of a database of trajectories and motion primitives for decision-based 3D contingency management in line with SC-VTOL continued safe flight and landing (CSFL).

a. Computation of a tree of trajectories leading to both the destination (based on the original graph) and alternate vertiports (utilizing a wavefront-based navigation function). A local Dubins path planner is used to ensure that transitions between trajectories to destination and alternate vertiports are feasible by connecting respective exit and entry states.

b. The tree is branched in constant time intervals, providing a safe alternative path to any diversion vertiport.

c. The trajectory tree is extended with primitives, allowing for a change of flight level, velocity and local holding patterns to implement an additional local action layer before changing the flight path.

d. Structure the trajectory tree to allow continuous reduction to the remaining set of reachable trajectories at flight time.

C-5: An approach for protected contingency online planning in pre-planned convex and obstacle-free regions with a limited time horizon.

a. Integration of a deterministic online planning method inside pre-computed convex and obstacle-free areas into the UAM path planning framework.

b. A combination of pre-computed, locally obstacle-free spaces and deterministic online planning methods with performance guaranteed manoeuvres enable deterministic online planning under bounded uncertainty.

c. With no obstacles present, an obstacle-unaware planning method can be selected, which typically convergence more quickly than obstacle-aware methods. Using protected online contingency planning, the tree size for a database-focused planning approach can be reduced considerably, without reducing the number of available planning alternatives.

1.5 Outline

Starting from a review of applicable and ongoing regulatory activities, assumptions on the future operating environments of automated UAVs are derived and applied to set the scope of the addressed motion planning problem. Quantitative methods to assess and evaluate risk as an integral part of the solution process are presented in chapter 4. The developed risk modelling approach incorporates risk from different sources and decouples the risk assessment from the planning task. The system architecture for a regulation-compliant path planning framework is developed and implemented in chapter 5.

The presented method relies to a large extent on databases of pre-computed trajectories and manoeuvring spaces, from which appropriate actions are selected at flight time. The governing principle of search space reduction and flight phase-specific planning methods is explained and implementations for the individual planning functions are proposed. In this context, two operational scenarios are considered. A concept in which multiple vehicles operate in opposite directions inside a narrow air corridor is developed to support e.g. supply routes of remote locations. The second scenario addresses vehicle operations in a network of vertiports and under consideration of continued safe flight and landing requirements. For this purpose, distinct flight levels are introduced and tree-like trajectory structures implemented, which provide alternative routes to the destination or alternate vertiports at constant time intervals.

To provide planning capabilities beyond the limits of databases and reduce the database size for large missions, a protected online planning functionality is developed in chapter 5.7. A feasibility criterion to contain an online planning capability inside a pre-computed convex region of obstacle-free space is developed and combined with an existing toolbox to compute performance guaranteed control laws.

Simulation results to validate the above planning functions are presented and discussed in chapter 6. A random obstacle environment is used to demonstrate the approach in a dense obstacle environment with a near-uniform distribution of obstacles. In a second step, risk models for an urban vertiport network are generated and coupled with the planning framework to compute planning solutions in a realistic operating environment.

Chapters 7 and 8 discuss the implications of results from chapter 6 and provide an outlook into potential extensions to the presented planning framework and related research topics of interest.

2 Path Planning Algorithms

This chapter describes the path planning problem as it is defined for the scope of this work and introduces important terms and respective definitions as used throughout the following chapters. Further, the search space exploration and motion planning algorithms, which are relevant for this work are introduced and described in detail. The aim is to provide a comprehensive understanding of existing methods from the literature, which are applied in this dissertation.

2.1 General Aspects of the Path Planning Problem

The path planning problem addressed in this work is the problem of finding a path between a start and destination position in an urban environment, which is collision-free and respects a set of applicable rules. This rule set is derived from recently published regulation and ongoing rule-making tasks for the operation of UAVs and (e)VTOLs in very low-level airspace over congested terrain. Vehicles operating within the scope of UAM can only operate between and within a network of dedicated vertiports, which are mapped and known to all parties prior to a flight planning request. Once a network of vertiports exists, it is considered quasi-static over the duration of a mission. Although most urban environments change continuously, the rate of change of obstacle maps and the vertiport infrastructure is considered slow relative to the duration of a UAM mission. It is therefore assumed that vehicles travel between the same set of take-off and destination vertiports repeatedly and frequently. Explanations and a detailed description of applicable rules and the reasoning behind them is provided in chapters 3 and 5.2.

The planning environment consists of a partially known obstacle map in three spatial dimensions. Static obstacles such as structures or permanent no-fly zones are part of the known obstacle set, whereas other air traffic, temporary air space restrictions or third ground parties are members of an obstacle set, which is unknown prior to flight time and subject to change over time. Hence, time is treated as the fourth dimension of the planning environment.

To avoid misinterpretation in the following chapters, it shall be distinguished between the terms “motion” and “path”. The term “motion” will be used to describe a system’s transition from one state in the n-dimensional state space s_i^n to another n-dimensional state s_{i+1}^n, where time is one dimension of the state space. Continued motion of a system in n-dimensional state space results in an n-dimensional state space trajectory τ'. The term “path” on the other hand is used to refer to a sequence of spatial coordinates connecting a start and destination location. A path τ has no time information and is ignorant of the target system. The planning problem addressed in this work is therefore best described as a motion planning problem, as it considers time as a dimension of the planning environment.

2.2 Terminology

Beyond the distinction between the terms “motion” and “path” the following terms are introduced and defined for the scope of this work:

1. *Configuration space* $\mathcal{C}$: The vector space defined by a system’s degrees of freedom (DOF). This work treats the motion planning problem for a system with six states: position in three-dimensional space (x, y, z), the azimuth χ, the path angle γ and the velocity magnitude $\bar{v}$. The configuration space is therefore six-dimensional. Constraints on certain parameters may apply, such that the actual configuration space can represent a manifold in the DOF vector space. The free configuration space is denoted $\mathcal{C}_{free}$.
2. *Search space* $\mathcal{S}$: The feasible set of all possible solutions. The search space is a subset of the configuration space. In the present case, the search space is limited to the spatial dimensions of the configuration space and derives the remaining degrees of freedom from constraints on the vehicle performance and implications from a coordinated flight assumption.
3. *2.5 dimensional obstacle representation*: Obstacles are defined from ground to a specific altitude above ground level and without undercuts. This means that e.g. bridges are represented as solid obstacles from ground to the roadway, with no option to pass below.

2.3 Search Space Exploration Algorithms

The motion planning problem for UAM and cargo delivery over congested terrain is set in an environment of pre-defined take-off and landing locations, which are linked by a network of routes. In contrast to tree-type data structures, a roadmap-type search graph allows to service multiple queries for connection routes between different vertiport pairs on a single exploration result during the graph’s interval of validity. For this reason, the selection of exploration methods focuses on roadmap computation approaches.

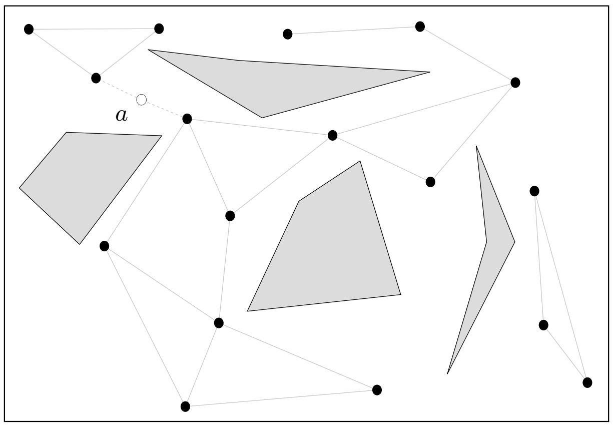


Figure 2-1: A roadmap graph with three unconnected components and a sample of the configuration space a with possible connecting edges (based on [40]).

2.3.1 Sampling-Based Roadmaps

Instead of computing an exact representation of the free configuration space $\mathcal{C}_{free}$, sampling-based methods generate a sufficiently accurate approximation of $\mathcal{C}_{free}$ from random sampling of the configuration space $\mathcal{C}$. This approach is particularly beneficial, when dealing with greater than two-dimensional configuration spaces or complex obstacle environments. The general framework of sampling-based roadmaps is introduced and presented in [36], however, paired with a probabilistic component which will be described in more detail in Section 2.3.1.2. This section is based on the description of the planning method in [40]. An example of a roadmap graph with three groups of connected vertices is provided in Fig. 2-1 along with a random sample of the free configuration space and possible connecting edges. Different groups of connected vertices in a graph are referred to as components. Sampling-based roadmap planners are probabilistically complete, i.e. the planner will find a solution to any query between two points in the free configuration space, using a sufficient number of samples and given that a solution exists. All roadmap methods share the notion of a construction phase and a query phase ([40]).

The construction phase:
The construction phase builds a topological graph $\mathcal{G}$ with a set of edges E and vertices V. Following the initialization of the graph $\mathcal{G}$ with empty vertex and edge sets, the main loop is repeated until the graph reaches a size of N vertices. Each iteration i samples a position a_i from the configuration space. If a_i does not lie in $\mathcal{C}_{free}$, a new sample is generated until a sample $a_i \in \mathcal{C}_{free}$ is obtained. Then it is added to the graph $\mathcal{G}$ as a new vertex q. As a new valid vertex was found, the index i is incremented, too. Different implementations to determine the neighbourhood of vertex q in $\mathcal{G}$ exist and will be addressed below. Regardless of the neighbourhood's definition, once a neighbourhood was obtained, the construction phase first checks each current neighbour q_n for its membership in a graph component other than that of vertex q. A collision check

on the solution of the local planning function *connect()* is only performed, if at least one neighbour q_n extends the graph's current connectivity. If the solution is collision-free, the edge $q - q_n$ is added to the graph $\mathcal{G}$.

The generic construction phase follows the algorithm illustrated in Algorithm 1, where different variants and derivatives of the sampling-based roadmap may use different implementations of the mentioned abstract functions.

Algorithm 1: A pseudo-code representation of the sampling-based roadmap construction phase. Note that i is only incremented if a vertex a is added to $\mathcal{G}$, such that the final roadmap will consist of exactly N vertices. Based on [40].

```
$\mathcal{G}$.init(), $\mathcal{C}_{free}$.init();
$q = \emptyset$, $i \leftarrow 0$;
while $i < N$ do
    if $a_i \in \mathcal{C}_{free}$ then
        $\mathcal{G}$.add_vertex($a_i$);
        $i \leftarrow (i + 1)$;
        foreach $q_n \in$ neighbourhood($a_i, \mathcal{G}$) do
            if (not $\mathcal{G}$.same_component($a_i, q_n$)) and connect($a_i, q_n$) then
                $\mathcal{G}$.add_edge($a_i, q_n$);
return $\mathcal{G}$
```

For cases, in which the objective is to obtain multiple, alternative solutions to a query, [40] suggests to replace the membership check with the degree of vertex q_n as a connection criterion, where the vertex degree represents the number of connections between the vertex and other vertices. A connection is established, if the vertex degree is below a threshold K. Different definitions and implementations to obtain a new vertex' neighbourhood exist. Among others, the standard literature ([40], [39]) identify a number of useful and efficient options:

Nearest K: Defines the neighbourhood of a vertex q as the set of K vertices closest to q. This is considered a good default implementation.

Component K: Defines the neighbourhood of vertex q as the set of up to K nearest vertices from each graph component. The threshold K should be selected lower than for the nearest K method to avoid an excessive number of neighbour candidates.

Radius: Chooses all vertices as neighbours, which are within a radius R of the current vertex q. An upper limit K for the neighbourhood size may be required with increasing density of vertices in the graph $\mathcal{G}$. With an increasingly uniform distribution of vertices, the neighbourhoods of a nearest K and radius implementation converge.

The query phase:
The query phase assumes that a sufficiently accurate and complete approximation of the configuration space was built in the construction phase. Based on a query to connect an initial position C_0 with a destination position C_1, the query phase tries to find connections from C_0 and C_1 to the graph $\mathcal{G}$, following lines 6-9 of Algorithm 1. If both positions can be connected to a vertex in $\mathcal{G}$, a planning solution can be obtained from a graph search on $\mathcal{G}$, for which e.g. Djikstra's algorithm ([16]) can be employed. A solution to the graph search on $\mathcal{G}$ corresponds to a solution to the query to connect C_0 and C_g in the free configuration space $\mathcal{C}_{free}$.

Like other sampling-based exploration methods (see e.g. rapidly-exploring random trees (RRT) ([41])), sampling-based roadmap approaches are only probabilistically complete. This means, that if no solution to a query is found on $\mathcal{G}$, it cannot be concluded, that no solution exists in $\mathcal{C}_{free}$, since the graph may only represent a subset of the free configuration space and may fail to represent sections of the configuration space, in which a solution exists. Sampling-based methods perform particularly poor on long and narrow passage ways, an issue to which the probabilistic roadmap method described in Section 2.3.1.2 seeks to provide a solution ([26]).

2.3.1.1 Sampling-Based Visibility Roadmaps

Sampling-based visibility roadmaps, first introduced in [67] and adopted into standard literature in [40], adhere to the notion of the construction and query phases described above, however, the method is different in the neighbourhood definition and acceptance criteria for a new vertex. This section on visibility roadmaps is based on [40]. The method divides nodes into two classes as illustrated in Figure 2-2:

- Guards: A sample a is considered a guard, if it cannot see other guards at the time of sampling. Guards cannot be exchanged for better guards at a later time. No other guards can exist in a guard's k visibility domain $V(k)$.
- Connecting nodes: A connecting node q must at least see two guards. For each connector, there exist two guards k_1 and k_2, such that $q \in V(k_1) \cap V(k_2)$, where $V(k_1)$ and $V(k_2)$ are the visibility domains of k_1 and k_2.

The main differentiation from previous methods lies in the strict connection criteria, a new vertex must fulfil, before it is added to the graph. While the neighbourhood (compare Algorithm 1, line 6) is defined as the entire graph, a vertex sample a can fall into one of three cases:

- Sample a cannot connect to any guard k within its visibility domain $V(a)$. It is added to the graph $\mathcal{G}$ as a new guard.

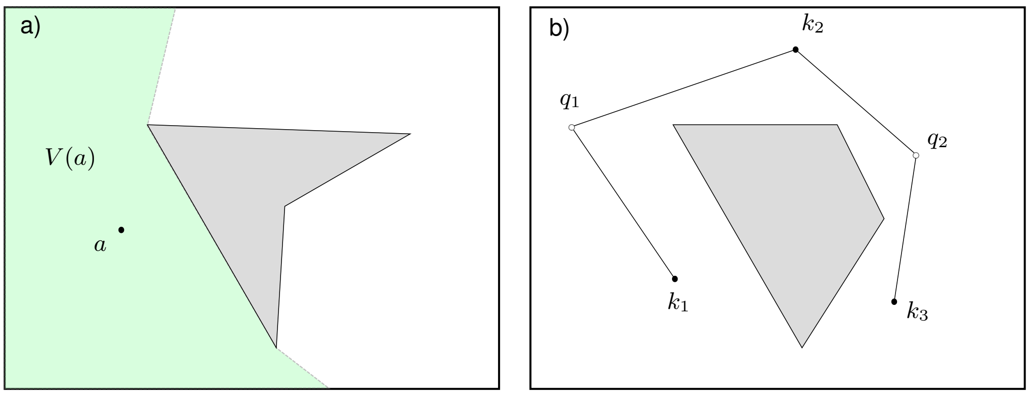


Figure 2-2: The sampling based visibility roadmap: a) defines the visible set $V(a)$ at sample a. Every configuration in $V(a)$ can be reached from a. The visibility roadmap in b) divides nodes into guards k_i and connecting nodes q_j. A guard's k_i visibility region $V(k_i)$ is empty of other guards, whereas a connecting node q_j must at least see two guards. Figure is based on [67].

- Sample a connects to at least two guards k_1 and k_2 from different connected components of $\mathcal{G}$. a is added to $\mathcal{G}$ as a connecting node with edges connecting a to guards k_1 and k_2.
- If neither of the above apply, a could only connect to guards of the same connected region. Sample a is discarded.

The combination of these criteria results in a strongly reduced number of vertices in the final roadmap, when compared to other sampling-based roadmap approaches. Despite being more time-expensive in the construction phase, the method is probabilistically complete while achieving the same coverage of the free configuration space as the base method with fewer nodes. The authors of [67] state that visibility roadmaps outperform the base method in narrow passages by a factor of 12 in terms of computation time. One disadvantage of the method is that its performance depends on the placement of guards, which cannot be deleted in favour of better guards, which may be found later in the construction phase. This specifically applies to pairs of guards with small intersections of respective visibility domains.

2.3.1.2 Probabilistic Roadmaps

Starting from the observation, that the construction phase described in Section 2.3.1 may lead to poorly connected graph components, where large regions of the free configuration space are connected by narrow passages, probabilistic roadmaps extend the framework of sampling-based search graphs with a probabilistic expansion step to improve the graph's connectivity ([36]). Following the construction phase, a probabilistic expansion step according to Algorithm 2 is inserted into the method.

Algorithm 2: A pseudo-code representation of the probabilistic expansion step.

```
1  $\mathcal{G}$.init(), $L$.init();
2  $\mathcal{G}$.assign_heuristic_weights($w$);
3  while true do
4      $q \leftarrow \mathcal{G}$.select_node($w$);
5      $n \leftarrow q$.expand_node($L$);
6      if $\mathcal{G}$.connection_exists($q, n, r$) then
7          $\mathcal{G}$.add_node($n$);
8          $\mathcal{G}$.add_ext_path($q, n$);
9          break
10 return $\mathcal{G}$
```

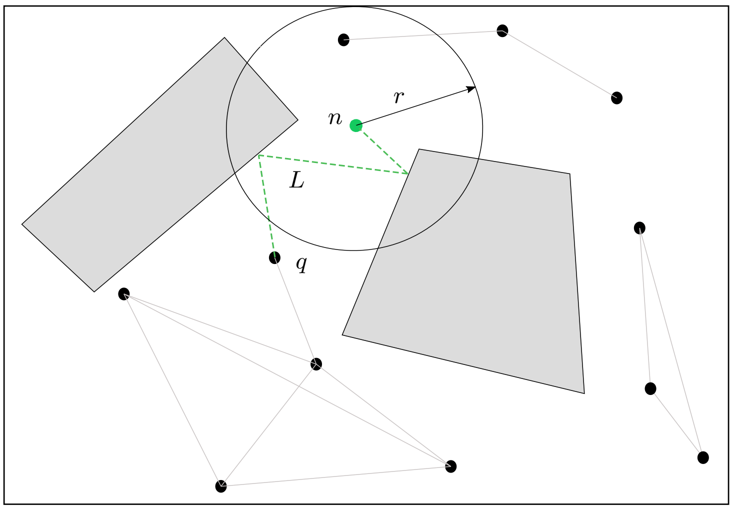


Figure 2-3: An illustration of the probabilistic expansion step (based on [36]). The expansion node q is expanded to the new node n. The neighbourhood definitions from the generic construction phase apply to find connections from n.

A heuristic weight w of “difficulty” is associated with each node, which describes the probability of the node lying in a narrow, hence difficult, region. The set of all weights is normalised, such that

$$\sum_{i=0}^{N} w_i = 1. \tag{2-1}$$

The heuristic to assign weights w_i is described in detail in [36]. While the termination criterion is not met, the method randomly selects a node q for expansion, where the probability that a node q is selected corresponds to its "difficulty" $w(q)$. The node is expanded within an expansion length L and the new vertex n is treated as a node of the same component as the expansion node q. Figure 2-3 expands the expansion node q with a random bounce method, which repeatedly picks a random direction and moves in this direction until a collision occurs or the extension length L is reached. Where the expansion path collides with a surface, a new random direction is chosen. If the vertex and new configuration n can be connected to a new component of graph $\mathcal{G}$ within radius r following Algorithm 1, ll. 6-9, the vertex n and edge (q, n) are added to the graph. Neighbourhood definitions to connect n to other components are adopted from the generic construction phase in section 2.3.1. For non-deterministic expansion methods, the path between q and n must be saved explicitly. If the vertex and new configuration n can be connected to a new component of graph $\mathcal{G}$, the vertex n and edge (q, n) are added to the graph.

Different difficulty heuristics and expansion strategies are discussed in [35]. An efficient method for holonomic robots is the node extension with a random bounce method described in Fig. 2-3. Probabilistic roadmaps yield solutions to configuration spaces of arbitrary complexity and dimension, however, at the cost of slow convergence in two- and three-dimensional applications ([26]).

2.3.2 Voronoi Roadmaps

Unlike the above methods, which sample the configuration space to build a graph representation of $\mathcal{C}_{free}$, Voronoi roadmaps build a search graph from the configuration space's Voronoi diagram. A comprehensive mathematical description of Voronoi diagrams as well as algorithms to compute a region's Voronoi diagram are provided in [14]. Voronoi diagrams are built from a finite number of seed points in a plane. Each seed point defines a region of space, where the distance between any point within the region and the region's seed point is less, than to any other seed in the plane. In the two-dimensional case illustrated in Figure 2-4, the Voronoi diagrams for two different seed environments are given. The boundaries between Voronoi regions compose a skeleton of the plane's configuration space. When interpreting region boundaries as edges and intersection points as vertices, a graph maximizing the distance to seed points is obtained from the Voronoi diagram.

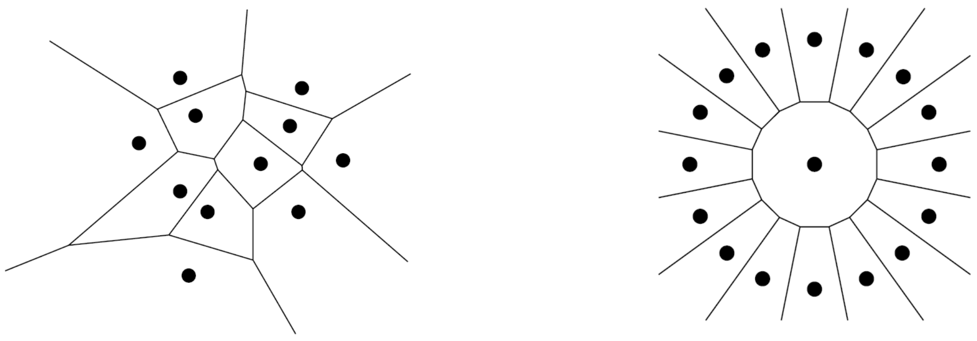


Figure 2-4: The Voronoi diagrams of two different environments of seed points in two-dimensional space. From [14].

Pruning the Voronoi graph to remove edges and vertices, which collide with the obstacle set, will yield a complete graph representation of the free configuration space (compare to the left-hand side of Figure 2-4). The roadmap solution derived from the Voronoi decomposition is therefore complete and runs in $\mathcal{O}(NlogN)$ time ([26]). Although the classical Voronoi decomposition is a two-dimensional algorithm, extensions to the three-dimensional case have been proposed (see e.g. [70] and [44]) and implemented in e.g. [60].

2.3.3 The Wavefront Expansion Algorithm

Wavefront expansion algorithms are a handy tool to compute optimal action plans for robots in discrete environments and are well-covered in standard literature [40] and [39]. The environment, in which a robot operates in two dimensions is discretised using a uniform grid, and a target location set X_T. An optimal action plan will provide the optimal action the robot can take to reach the target set X_T from any discrete state in the free configuration space using a local evaluation strategy S. Action plans can be implemented using a discrete function $\phi : X \mapsto [0, \infty]$ on which the local evaluation strategy S defines the optimal strategy in every state x as

$$S = argmin(\phi(u)), \text{ with } u \in U, \tag{2-2}$$

where U is the neighbourhood of the state x. Throughout the remainder of this dissertation, the definition of a navigation function given in [40] will be used, which refers to an action plan with three distinct properties. An action plan that is implemented as a function ϕ is called a navigation function if:

- $\phi(x) = 0$ for all $x \in X_T$.
- $\phi(x) = \infty$ if no state in X_T can be reached from x.

- For each state $x \in X \setminus X_T$, the evaluation strategy S returns a state x' with $\phi(x') < \phi(x)$.

A navigation function is considered an optimal navigation function, if the evaluation function incorporates a cost metric for the transition between two states x and u and satisfies the optimality criterion

$$\phi(x) = min(c(x,u) + \phi(u)), \text{ with } u \in U, \tag{2-3}$$

where $c(x,u)$ represents the cost to reach state x from u. The wavefront algorithm is a simplification of Dijkstra's algorithm ([16]) to compute optimal navigation functions in the case that each motion has unit cost c_0. The boundary states of the target set X_T form the initial wavefront W_0. Any neighbouring state on W_0 can be assigned the optimal cost value of 1 and is organised in the new wavefront W_1. Neighbouring states of W_1 receive the cost value 2 and are organised in the wavefront W_2. An inductive repetition of this step will reach all reachable states in the configuration space on the i-th wavefront W_i in $\mathcal{O}(n)$ time complexity. Unreached states are assigned a cost value of ∞. Algorithm 3 describes the wavefront expansion algorithm with unit cost as pseudo code.

Running the evaluation strategy S in (2-2) on the navigation function $\mathcal{N}$ computed from the wavefront expansion algorithm will result in an optimal control policy towards X_T from any point in the free configuration space on $\mathcal{N}$ (compare to Figure 2-5).

Algorithm 3: A wavefront expansion algorithm with unit cost c_0 for each motion. Algorithm is based on [40].

```
$c_0$.init();
$\mathcal{W}_0 \leftarrow X_T, i \leftarrow 0$;
do
    foreach $x \in \mathcal{W}_i$ do
        $\phi(x) = i \cdot c_0$;
        foreach $q \in x$.unexplored_neighbours() do
            $\mathcal{W}_{i+1}$.add($q$);
    $i \leftarrow (i+1)$;
while $\mathcal{W}_i \neq \emptyset$;
return $\mathcal{W}$
```

2.4 Finite State Motion Models

Finite state motion models are hybrid control systems, which discretize a vehicle's continuous action envelope into a finite set of discrete motions, so-called motion primitives.

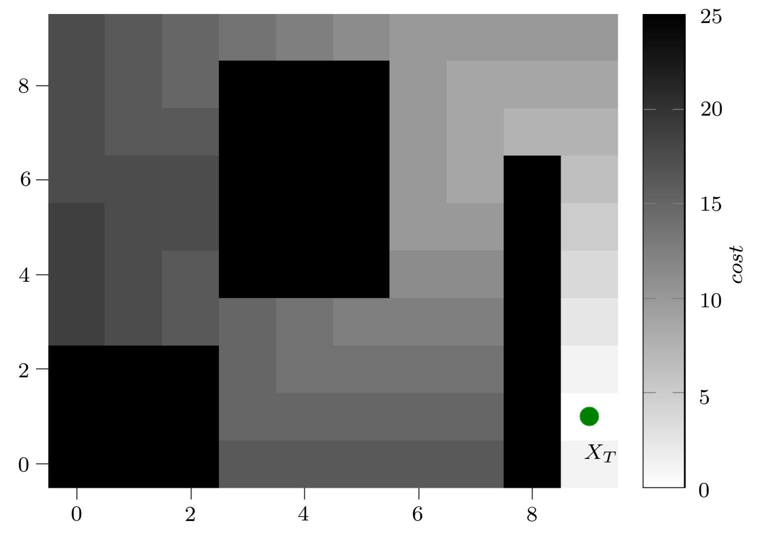


Figure 2-5: An optimal navigation function built from a wavefront expansion algorithm starting in X_T and based on unit cost for each motion. Obstacles are depicted in black.

Transferred to e.g. a fixed-wing aircraft's lateral motion, this means that instead of allowing turns at an arbitrary bank angle within the vehicle envelope, a finite state motion model defines countably few bank angles across its envelope, at which the vehicle can turn. This notion helps to reduce the complexity and computational cost to generate a near-optimal and feasible path between two states in space considerably.

2.4.1 The Optimal Manoeuvre Automaton

Several examples of motion planning frameworks similar to the manoeuvre automaton can be found in e.g. [26], whereas the first strict definition and analysis of manoeuvre automata was presented in [21]. This description of the manoeuvre automaton is partially adapted from [49]. A manoeuvre automaton discretizes the vehicle's continuous motion envelope into a finite set of trim states and manoeuvres. The set of trim states Q_T contains states of steady control input parameters, such as hover, level flight at constant velocity or steady turns. Manoeuvres are a set of motion primitives Q_M, which describe transitions between two trims, i.e. a manoeuvre is a state, in which the derivative of at least one control input parameter is non-zero. The union of trim states Q_T and manoeuvres Q_M is referred to as a motion library Q. A graphical illustration of a small manoeuvre automaton for a forward-moving and curvature-constrained robot in two dimensions, is given in Figure 2-6. The robot's movement on the xy-plane is described by a velocity v and azimuth rate $\dot{\Psi}$.

The initial description in [21] defines the manoeuvre automaton as a control system with the below properties and is applied to the manoeuvre automaton of Fig. 2-6:

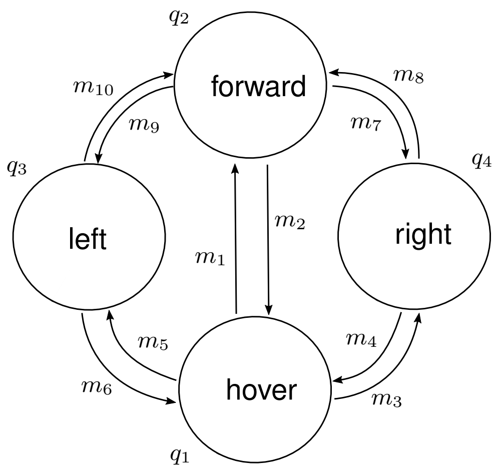


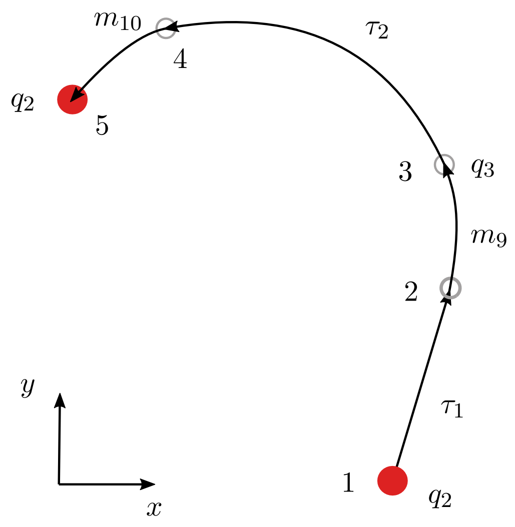
Figure 2-6: An illustration of a simple manoeuvre automaton with four trim states q_1 to q_4. Arrows indicate transitions between trim states using manoeuvres m_1 to m_{10}. Figure is based on [49].

- A finite motion library $Q = Q_T \cup Q_M$, where Q_T represents a set of trim primitives and Q_M a set of manoeuvres.
- A finite set of trim primitive parameters η_q, with $q \in Q_T$. In the above example $\eta_q = \left\{v, \dot{\Psi}\right\}$ applies.
- A finite set of manoeuvre parameters ζ_m, with $m \in Q_M$. In the above example $\zeta_m = \left\{\dot{v}, \ddot{\Psi}\right\}$ applies.
- The mappings Previous : $Q_M \mapsto Q_T$, and Next : $Q_M \mapsto Q_T$ such that Previous(m) and Next(m) return the trim state in which the manoeuvre m starts and ends, respectively.
- A discrete state $s \in Q_T$, i.e. for the curvature-constrained robot $s \in \{\text{hover}, \text{forward}, \text{right}, \text{left}\}$.
- A continuous state $h \in H$, representing the vehicle position, orientation and speed of the robot $h = [x, y, \Psi, v]^T$.
- A clock state $\theta \in \mathbb{R}$, which is reset after each switch of s.

The manoeuvre automaton generates a trajectory between two configurations in space using a combination of manoeuvres and trim states. From the above definition, it becomes evident, that for each trim state, the planning solution must provide a duration, which defines how long the vehicle stays in the particular trim state before executing the subsequent manoeuvre. This duration is denoted the coasting time τ. The hybrid control strategy (m, τ) defines a manoeuvre m to enter a trim state q and the time τ spent in q. Hence, it implements a bijective map $(m, \tau) \leftrightarrow (s, h)$ describing a time-continuous trajectory by two discrete parameters. Figure 2-7 implements a time-continuous trajec-

	state	movement	duration
1	q_2	forward	τ_1
2	m_9	enter left turn	$t(m_9)$
3	q_3	left turn	τ_2
4	m_{10}	exit left turn	$t(m_{10})$
5	q_2	forward	0

(a) List and durations of manoeuvres and trim states.



(b) The resulting trajectory in two-dimensional space.

Figure 2-7: A sequence of manoeuvres and trim states from the automaton of Fig. 2-6 implementing a left turn.

tory of a left turn using a sequence of manoeuvres and trim states from the automaton of Fig. 2-6. The movement begins in position 1 with the agent executing a straight forward motion q_2 for the duration τ_1. In position 2, the manoeuvre to enter a steady left turn m_9 is initiated. The manoeuvre has a predetermined duration $t(m_9)$. From position 3, where the transition manoeuvre ends and the agent is in a state of a steady left turn q_3, the state q_3 is maintained for the duration τ_2 until the manoeuvre to exit the turn m_{10} starts in position 4. A straight forward motion q_2 is re-established in position 5.

To assess the quality of a trajectory and derive an optimality condition, the cost of a trajectory connecting two states needs to be defined. For this purpose, the manoeuvre-end map $\Phi_{(m,\tau)}(s,h)$, describing the state which results from the execution of the hybrid control strategy (m,τ) in the current state (s,h), is introduced. Following Bellman's principle of optimality implies that the optimal cost function $\tilde{J}^*(s,h)$ of a trajectory on the manoeuvre automaton satisfies

$$\tilde{J}^*(s,h) = \min_{(m,\tau)} \left\{ \gamma_s \tau + \Gamma_m + \tilde{J}^* \left[\Phi_{(m,\tau)}(s,h) \right] \right\}, \tag{2-4}$$

where γ_s is the cost per time of trim trajectory s and Γ_m the cost of manoeuvre m.

An approximation of the optimal cost J^* can be obtained by means of a value iteration

$$J_i(s,h) = \min_{(m,\tau)} \left\{ \gamma_s \tau + \Gamma_m + J_{i-1} \left[\Phi_{(m,\tau)}(s,h) \right] \right\}. \tag{2-5}$$

To be used in the above value iteration, a discrete approximation of the continuous state $h \in H$ must be found. This can be achieved with a finite number of representative states $h_i,\ i = 1...N$. The initial implementation [21] proposes a piece-wise linear approximation of the set H into a collection of simplices with disjoint interiors. Any continuous state h will be a member of one simplex with vertices $h_{i,1}, h_{i,2}, ...h_{i,n}$, such that the cost function at h can be written as

$$\hat{J}(s,h) = \sum_{j=1}^{n+1} \lambda_{i_j} \hat{J}^{i_j}(s), \tag{2-6}$$

with

$$h = \sum_{j=1}^{n+1} \lambda_{i_j} h_{i_j}, \tag{2-7}$$

where λ_{i_j} are non-negative coefficients that are selected to describe h as a linear approximation of the representative states h_i. The parameter n represents the dimension of H.

The principle working pattern of the value iteration (2-5) is explained using the strongly simplified example of Fig. 2-8: A grid of dimension 4x5 is considered, on which an agent starting from any position on the grid shall reach the bottom left corner using as few moves as possible. Further, the agent's movement is constrained to the movement of a knight on a chess board, i.e. it moves two cells vertically and one cell horizontally, or two cell horizontally and one cell vertically. The grid is initialised with zeros. The cost of a move Γ_m is 1. After the first iteration (Fig. 2-8a)), all cells except the target cell have value 1, this is because from each cell except the target, any move ends up in a cell of value 0. After the second iteration (Fig. 2-8b)), cells, from which the destination is reached with one move remain at value 1, all other cells increment to value 2. This pattern repeats in iterations 3 and 4 (Fig. 2-8c) and d)), until no cell's increment between two iterations violates a termination criterion ϵ and convergence is achieved in iteration 5 (Fig. 2-8e)). In the presented example, the termination criterion is selected as $\epsilon < 1$. The final iteration's grid values represent the optimal cost function $\tilde{J}^*(s,h)$ to reach the target state from anywhere on the grid and under the given constraint.

If the optimal cost function J^* is approximated in an offline computation step the execution of a greedy policy on J^* in the online phase results in a near-optimal and deterministic control strategy within the limits of the motion library Q and under the assumption of small disturbance of the actuated system. An example of an optimal cost function for the manoeuvre automaton of Fig. 2-6 is illustrated in Fig. 2-9. Considerations on the manoeuvre automaton's robustness against uncertainty in the system and external disturbances, which lead to the development of the robust manoeuvre automaton framework are e.g. described in [22].

2.4.2 Dubins Paths

The notion of Dubins curves, initially introduced in [17] and adopted into standard literature in e.g. [40], discretizes the vehicle envelope more aggressively than the ma-

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	1	1	1

(a) Iteration 1

2	2	2	2	2
2	1	2	2	2
2	2	1	2	2
0	2	2	2	2

(b) Iteration 2

3	2	3	2	3
2	1	3	3	2
3	3	1	2	3
0	3	2	3	2

(c) Iteration 3

4	2	3	2	3
2	1	4	3	2
3	4	1	2	3
0	3	2	3	2

(d) Iteration 4

5	2	3	2	3
2	1	4	3	2
3	4	1	2	3
0	3	2	3	2

(e) Iteration 5

Figure 2-8: Convergence of the optimal cost function of a constrained agent operating on a grid and targeting the bottom left corner in five iterations.

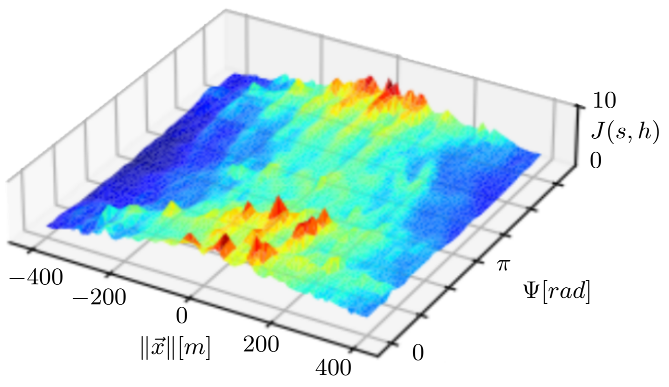


Figure 2-9: A slice of the approximation $J(s,h)$ of the optimal cost function $J^*(s,h)$ of the manoeuvre automaton of Fig. 2-6. $\|\vec{x}\|$ represents the translational distance from the target state. Ψ denotes the heading angle towards the target. From [49].

Table 2-1: Dubins curve motion primitives.

Description	Sign	Curvature
straight	S	0
left	L	-1
right	R	1

noeuvre automaton. It assumes a vehicle, which can only travel forward and has a constraint on the maximum tracking curvature. The term Dubins curve then refers to the shortest path between two points in the Euclidean plane, which is computed from a combination of maximum curvature arcs and straight segments. The state space discretization of a Dubins path planner can therefore be described using the primitives in Tab. 2-1. While ten combinations of these primitives exist, the author of [17] showed, that only the sequences

$$\{L_\alpha R_\beta L_\gamma,\ R_\alpha L_\beta R_\gamma,\ L_\alpha S_d L_\gamma,\ L_\alpha S_d R_\gamma,\ R_\alpha S_d R_\gamma,\ R_\alpha S_d L_\gamma\} \tag{2-8}$$

can possibly be a representation of an optimal path, where α, β, γ are turning angles and d is the straight line distance. For any two configurations in the Euclidean plane, Dubins curves can be described geometrically using the two circles tangent to the vehicle heading in each state and an additional support circle for sequences consisting of only turns (compare to (2-8)). Figure 2-10 illustrates Dubins curves for two sequences and different start and target configurations. Similar to coasting times in the manoeuvre automaton framework, the Dubins solution requires to compute the angles (α, β, γ) and distance d, which the vehicle shall travel on each arc or straight segment, respectively. This problem can be understood and solved as a geometric optimization problem.

Dubins paths are resolution complete, however, produce paths of discontinuous curvature at the interface between two primitives. Different approaches to smooth the transition between primitives and improve a vehicle's tracking performance on Dubins paths have been proposed, of which Bezier curves ([3]) and clothoids ([51]) are among the best known. Often, Dubins paths are used with graph-based exploration methods to provide a local planning method in regions, which are already known to be obstacle-free.

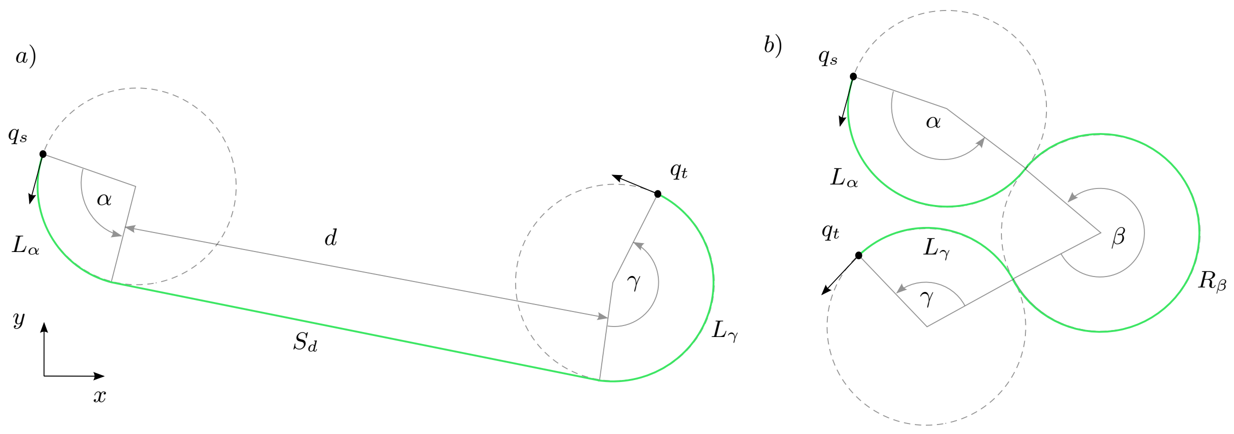


Figure 2-10: Dubins curves in two-dimensional space between different start and target configurations q_s and q_t and using different sequences: a) illustrates a $L_\alpha S_d L_\gamma$ path between configurations q_s and q_g, whereas b) shows a $L_\alpha R_\beta L_\gamma$. Figure is based on [40].

3 European Regulation for UAV and (e)VTOL Operations

This chapter summarizes the status of existing European regulation and ongoing rule-making tasks in the field of VTOL aircraft until June 2021. Due to the highly dynamic nature of these activities, the regulation baseline presented here can only be considered a snapshot at the time of writing. However, the governing principles described, and general notions presented are expected to persist and be found in the final versions of regulation documents.

3.1 Special Condition for Vertical Take-Off and Landing Vehicles (SC-VTOL)

The special condition for small-category vertical take-off and landing aircraft (SC-VTOL, ([68])) was published by EASA in 2019 to support ongoing certification endeavours of multiple VTOL aircraft manufactures. SC-VTOL defines objective-based certification requirements in order to provide the necessary flexibility for different VTOL configurations to become certified, given that they can meet these requirements, but regardless of the vehicle design. It is based on the certification specification CS-23 for Normal, Utility, Aerobatic and Commuter Aeroplane ([10]) and integrates elements from the certification specification CS-27 for small rotorcraft ([11]). New elements are proposed, where neither of the previous seem appropriate. The scope of applicability is limited to aircraft with a maximum certified take-off mass of 3175kg and a seating capacity for nine or fewer passengers. SC-VTOL is expected to be the baseline for a certification specification for VTOL vehicles.

The special condition distinguishes between two certification categories according to the intended mission of the aircraft as defined in SC-VTOL VTOL.2005 and AMC VTOL.2510:

- Category Enhanced: Applies to "[...] aircraft intended for operations over congested areas or for Commercial Air Transport operations of passengers[...]" ([68]). Any failure condition that would prevent continued safe flight of the aircraft and landing at a vertiport in cases of off-nominal operation is considered catastrophic.
- Category Basic: Applies to all VTOL aircraft and operations in scope of SC-VTOL, for which the enhanced category does not apply. Any failure condition that would prevent a controlled emergency landing of the aircraft even outside a vertiport is considered catastrophic.

The notion of continued safe flight and landing (CSFL) is only introduced for the enhanced category, which has a significant impact on how aircraft manufacturers and operators shall approach the automation of VTOL vehicle operation in this category.

SC-VTOL VTOL-2000 defines CSFL as an aircraft's capability to perform "[...] continued controlled flight and landing at a vertiport, possibly using emergency procedures, without requiring exceptional piloting skill or strength" ([68]). This means that any event within the scope of CSFL must not lead to a landing outside a prepared vertiport or similar landing site. Proposed means of compliance (MOC) for SC-VTOL ([56]) state explicitly, that minimum performance and obstacle clearance requirements apply to CSFL as do for the nominal envelope and that a landing in the scope of CSFL shall not cause additional damage to the aircraft.

Table 3-1 illustrates enhanced and basic categories including subcategories and failure condition classifications. In accordance with the above, VTOL operations over congested areas fall into the enhanced category, regardless of the mission objective being cargo or passenger transport. This has three important implications on the motion planning task:

1. The planning task for nominal and contingency operations is constrained to queries between vertiports.
2. The planning problem's configuration space is partially under the control of the vertiport operator.
3. Emergency situations, outside the scope of CSFL, and therefore planning methods applied in such scenarios, are beyond the scope of the certification baseline. As a consequence, planning algorithms for use in emergency operations are not addressed in this thesis.

A detailed analysis and derivation of boundary conditions on the urban planning task from EASA regulation is provided in Chapter 5.

3.2 EASA Regulation for UAS Operations in the Specific Category

While the previous section and summary of the special condition for VTOL aircraft refer to a new class of aircraft, they are piloted conventionally with a human pilot onboard the aircraft and no means to control the aircraft from an external site. This section briefly summarizes the existing regulation, applicable for unmanned aircraft systems in the specific UAS category at the time of writing of this dissertation. Current UAS regulation explicitly excludes operations of unmanned, passenger-carrying vehicles, where no operator is present onboard the aircraft. However, this overview is considered relevant as ongoing rulemaking activities for the certified UAS category (RMT.0230, [59]) evolve from existing standards.

Table 3-1: SC-VTOL certification categories and associated failure condition classifications. Quantitative safety objectives are expressed per flight hour. From [68].

		Failure Condition Classification			
	Maximum Pax Seating Configuration	**Minor**	**Major**	**Hazardous**	**Catastrophic**
Enhanced Category	any	$\leq 10^{-3}$ FDAL D	$\leq 10^{-5}$ FDAL C	$\leq 10^{-7}$ FDAL B	$\leq 10^{-9}$ FDAL A
Basic Category	7-9 Pax	$\leq 10^{-3}$ FDAL D	$\leq 10^{-5}$ FDAL C	$\leq 10^{-7}$ FDAL B	$\leq 10^{-9}$ FDAL A
	2-6 Pax	$\leq 10^{-3}$ FDAL D	$\leq 10^{-5}$ FDAL C	$\leq 10^{-7}$ FDAL C	$\leq 10^{-8}$ FDAL B
	0-1 Pax	$\leq 10^{-3}$ FDAL D	$\leq 10^{-5}$ FDAL C	$\leq 10^{-6}$ FDAL C	$\leq 10^{-7}$ FDAL C

3.2.1 The UAS Ecosystem

Unlike conventional aircraft, which combine all functions relevant to control and operate the aircraft onboard a single system component (the aircraft), unmanned aerial systems can have several components. According to EASA regulation [58] and [59], a UAS includes at least the following two subsystems:

The unmanned aircraft (UA): The UA is an aircraft and aviation product and can be a powered-lift, fixed-wing or hybrid system designed for different missions and use cases. The UA can be piloted manually from a remote location and/or implement different levels of automation.

The command unit (CU): According to the Basic Regulation ([58]), the CU is an equipment to control the unmanned aircraft remotely. One CU may control several UAs. The scope and implementation of the CU heavily depends on the UA, the intended operation and mission and level of automation. The implementation of a CU can range from a handheld device to structural installations in buildings or vehicles and potentially include ground navigation aids or other equipment which is critical to the safe operation of the UA. The CU is expected to be specified as part of the UA type design.

Figure 3-1 provides an illustrative example of a UA and different CU layers for a UAS operation as expected in the scenario of flights over urban terrain and in scope with the remainder of this work. Depending on the mission, there may exist more than one vertiport layer in the UAS with different vertiport types providing different services.

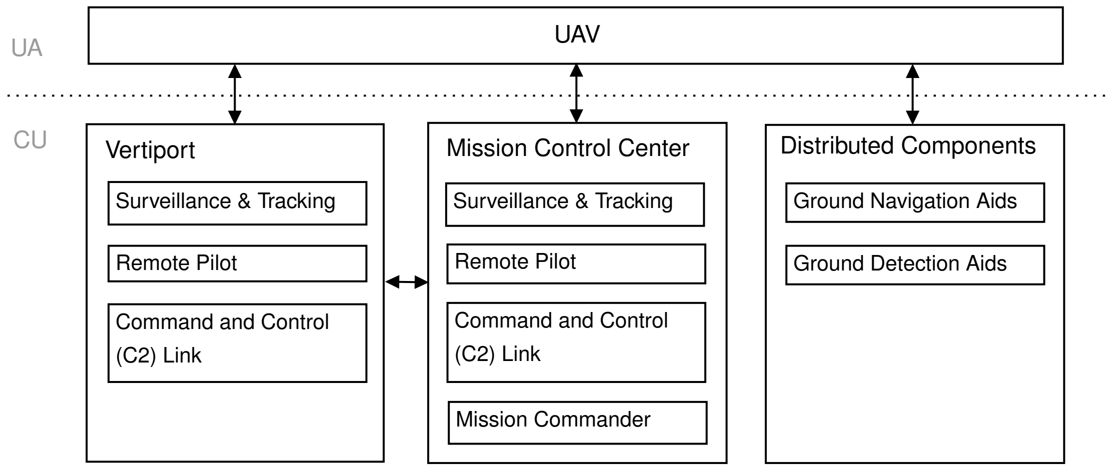


Figure 3-1: A UAS with multiple command unit layers as expected in the urban use case. Depending on the mission, there may exist more than one vertiport layer in the UAS. Different vertiport types may provide different services.

3.2.2 Risk Assessment as an Enabler of UAS Operations in the Specific Category

All UAS operations, which violate one or more criteria for operation in the EASA open category as defined in Article 4 of [20] are required to operate in the specific category. Operators in the specific category shall apply for an operational approval from the competent authority and perform a risk assessment during the application process. The identified risk in combination with appropriate mitigation measures decide if and under which condition an approval is granted. Until the time of writing this work, the only Acceptable Means of Compliance (AMC) to conduct such risk assessments is the specific operations risk assessment (SORA) according to [20], Article 11, and developed by the Joint Authorities on Rulemaking for Unmanned Systems (JARUS) in [33].

SORA provides a framework to demonstrate that an intended UAS mission can be conducted safely to both the operator and competent authority. The concepts of risk and robustness are key to the SORA process. The notion of risk as it is used in SORA follows the SAE's ARP 4754A / EUROCAE ED-79A definition as “[...] the combination of the frequency (probability) of an occurrence and its associated level of severity [...]” ([38]). It uses a holistic risk model, which defines the probabilityP_{harm} that an operation causes harm to a third party as the product of the probabilities of three independent aspects

$$P_{harm} = P_{ooc} P_{strike|ooc} P_{harm|strike}, \tag{3-1}$$

where P_{ooc} is the probability of an out-of-control state, $P_{strike|ooc}$ is the probability of a strike in an out-of-control state and $P_{harm|strike}$ the probability that a strike causes harm to third parties or critical infrastructure. Robustness in the SORA context combines the level of integrity and assurance at which a property is achieved. Guidance and reference materials to evaluate the extent, to which an operator claim fulfils these concepts are provided in [20]. In addition to the concepts of risk and robustness, SORA uses

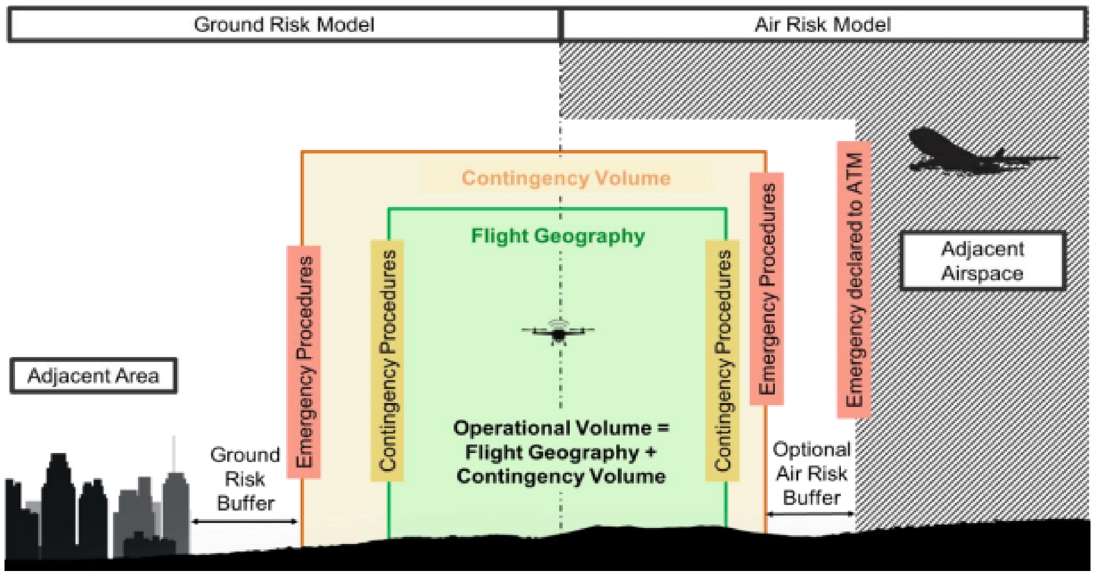


Figure 3-2: The semantic SORA model defining terminology and operational volumes as used throughout the SORA process. From [20].

the terminology and definitions of operational volumes provided in Figure 3-2, which illustrates the semantic SORA model.

The SORA process to assess the risk associated with the operation of a specific UAS in a specific environment and under specific boundary conditions embraces the above notions and follows ten distinct phases described in the SORA process flow chart in Figure 3-3. Based on a concept of operations (ConOps), in which the operator describes the UAS, its operation and the operator's operational safety culture, an analysis of the ground risk class (GRC) is conducted. The intrinsic GRC (iGRC) describes the risk of a person on ground being struck be the UAS. While a generic metric to assess the iGRC based on vehicle dimensions, kinetic energy and operational scenario categories is provided in [20], JARUS are working towards detailed models to allow a quantitative assessment of the iGRC and account for different vehicle designs (see ongoing work on [2]). The iGRC can be controlled and reduced with appropriate mitigation means, where mitigation strategies implemented with high robustness lead to greater reduction of the iGRC and therefore lower the final GRC. The GRC is evaluated on an integer scale from 1 to 10, with 1 representing the lowest risk. The process to assess the air risk class (ARC) follows the same notion as the evaluation of the GRC. The initial ARC corresponds to the intrinsic risk of a mid-air collision, which can be controlled and reduced using strategic mitigations (e.g. ADS-B or FLARM) to obtain the residual ARC. The ARC is discretized into ARC categories a-d, with category a being the lowest. On top of strategic mitigation strategies, SORA implements tactical mitigation performance requirements (TMPR), allowing to reduce the final GRC and residual ARC further. For operations within the visual line of sight (VLOS), the see-and-avoid concept can be fulfilled by the operator, however, additional conditions may apply to

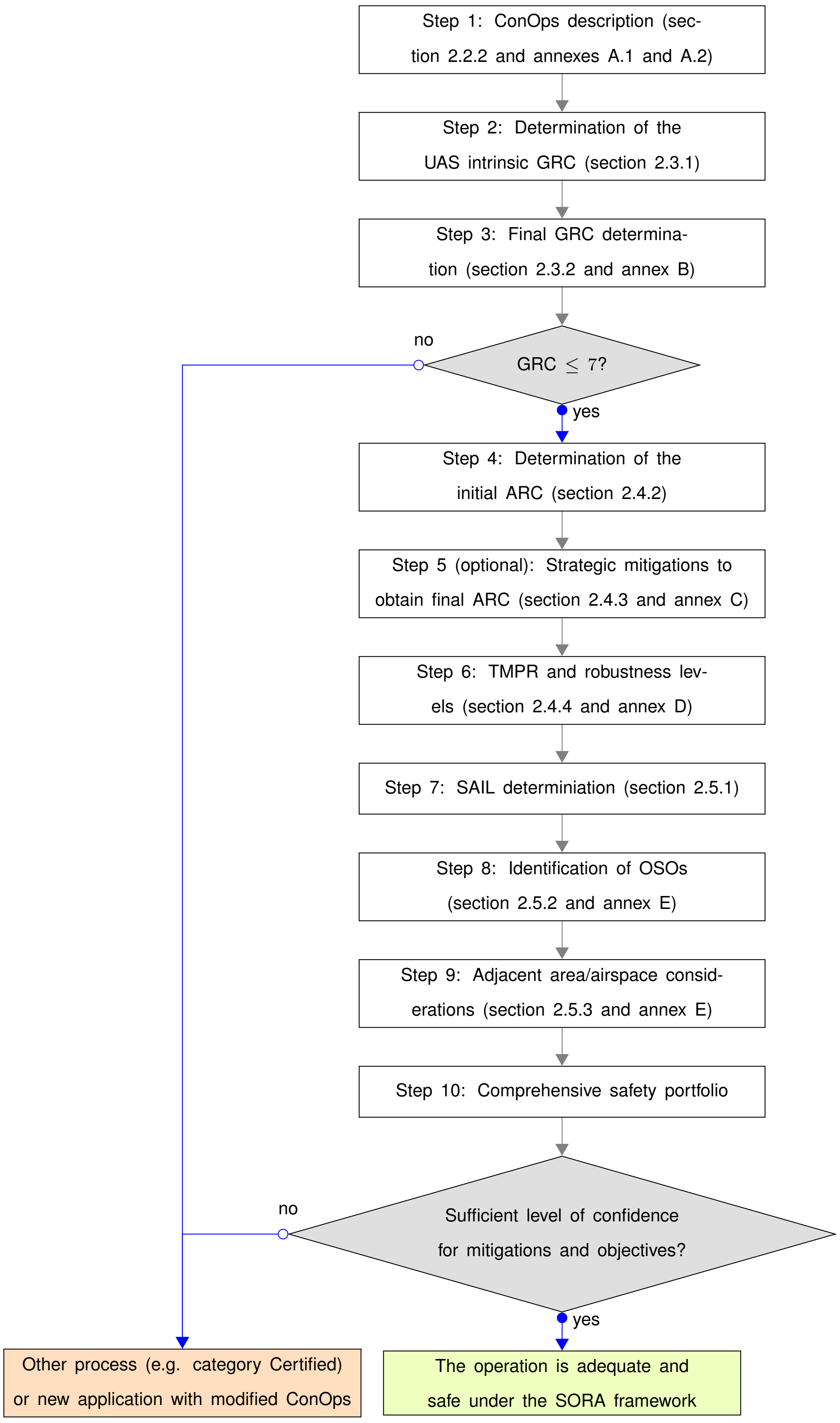

Step 1: ConOps description (section 2.2.2 and annexes A.1 and A.2)
Step 2: Determination of the UAS intrinsic GRC (section 2.3.1)
Step 3: Final GRC determination (section 2.3.2 and annex B)
GRC ≤ 7?
no
yes
Step 4: Determination of the initial ARC (section 2.4.2)
Step 5 (optional): Strategic mitigations to obtain final ARC (section 2.4.3 and annex C)
Step 6: TMPR and robustness levels (section 2.4.4 and annex D)
Step 7: SAIL determiniation (section 2.5.1)
Step 8: Identification of OSOs (section 2.5.2 and annex E)
Step 9: Adjacent area/airspace considerations (section 2.5.3 and annex E)
Step 10: Comprehensive safety portfolio
Sufficient level of confidence for mitigations and objectives?
no
yes
Other process (e.g. category Certified) or new application with modified ConOps
The operation is adequate and safe under the SORA framework

Figure 3-3: A flow chart of the ten phases of the SORA process (based on [20]). Operations outside the SORA scope may become subject to the certified UAS category.

operations under extended VLOS (EVLOS). UAS operated under BVLOS conditions must fulfil additional TMPRs, which demand the use of detect-and-avoid (DAA) systems or other systems, which indicate the presence of other air traffic participants and the need for an avoidance manoeuvre to the operator.

The final GRC and residual ARC are consolidated in the specific assurance and integrity level (SAIL), which indicates the confidence that the described operation will remain under control. In a subsequent step, the SAIL is used to determine the recommended level of robustness, with which additional safety mechanisms, so-called operational safety objectives (OSOs) should be implemented. A list of proposed OSOs is provided in Annex E of [20].

The final step in the evaluation of the risk associated with a specific UAS operation concerns the probability that an out-of-control state will cause the UA to infringe the adjacent area or airspace. SORA requires the operator to substantiate that "[...] no probable failure of the UAS or any external system supporting the operation should lead to operation outside the operational volume" ([20]). For operations with greater risk (see Article 11 of [20] for details), additional level of rigor is applied, which requires that "[...] (1) the probability of the UA leaving the operational volume should be less than $10^{-4}\frac{1}{fh}$; and (2) no single failure of the UAS or any external system supporting the operation should lead to its operation outside the ground risk buffer" ([20]).

UAS operations with a final GRC of 7 or greater, and for which safety objectives cannot be met with a sufficient level of confidence and robustness require to repeat the SORA application with e.g. a limited ConOps or may become subject to the certified UAS category.

3.3 EASA Concept for the Regulation of UAS Operations in the Certified UAS Category

The scope of the the EASA certified UAS category defines three types of operations, which are considered relevant for the future operation of unmanned and/or passenger-carrying VTOL aircraft ([59]):

1. Operations type #1: Operations under instrument flight rules (IFR) of cargo-carrying UAS in air spaces A-C, which operate between EASA-regulated aerodromes.

2. Operations type #2: UAS operations over congested or non-congested areas in U-space air space for the purpose of passenger or cargo transport. This includes unmanned VTOL aircraft.

3. Operations type #3: Operations of manned, passenger-carrying VTOL aircraft inside or outside U-Space airspace.

Although type #3 operations include a pilot onboard the aircraft and are therefore not UAS operations, type #3 is considered for the similarity in the operating environment and UAM mission, which lead to similar regulatory requirements on the U-Space and infrastructure. This work develops methods for the automation of type #2 operations, which is why this section will focus on regulation relevant for unmanned operations in U-Space air space around and between adequate aerodromes. However, the methods presented could be adapted to support type #3 operations if applicable changes and limitations are introduced, which account for the presence of a pilot onboard the aircraft.

According to [59], early operations of unmanned UAS accept the fact that detect and avoid (DAA) systems, which are developed in accordance with a set of validated standards accepted throughout Europe, do not exist yet. Strategic and tactical mitigation means need, hence, to be put in place by the U-Space service provider to guarantee safe operations. Current rule-making activities for near-term type #2 operations assume the presence of U-Space services, which support strategic and pre-tactical deconfliction of aircraft or DAA-capabilities, where applicable. In general, these assumptions apply to UAS operations at very low level (VLL), below the minimum altitude for operations under visual flight rules (VFR). Where U-Space airspace expands beyond VLL, [59] requires coordination procedures between the U-Space provider and air traffic service (ATS) unit to guarantee appropriate deconfliction between VFR manned and VLL unmanned traffic. A detailed description of U-Space services can be found in Annex IV of [59].

It is expected that early operations over congested areas will be restricted to a finite set of pre-defined routes and corridors, for which the operator must assure proper mitigation of the ground and air risk, as well as compliance with minimum heights addressed in SERA.3105 ([69]), to the competent authority or U-Space service provider. The proposed process foresees the operator to submit a flight plan to the U-Space service provider for authorization. The service provider is responsible to analyse conflicts between authorized flights and other request and based on the results of this analysis authorizes the flight plan or proposes an alternative pre-defined route. The feasibility and availability of such pre-defined routes need to take into account the availability and existence of appropriate landings sites (i.e. vertiports) for take-off and landing under regular conditions but also contingency operations. For each mission, alternate vertiports must be pre-planned to allow for a diversion from the initial destination vertiport, in the event that a landing at this vertiport is no longer advisable or possible. Pre-planned alternate vertiports are divided into the following categories:

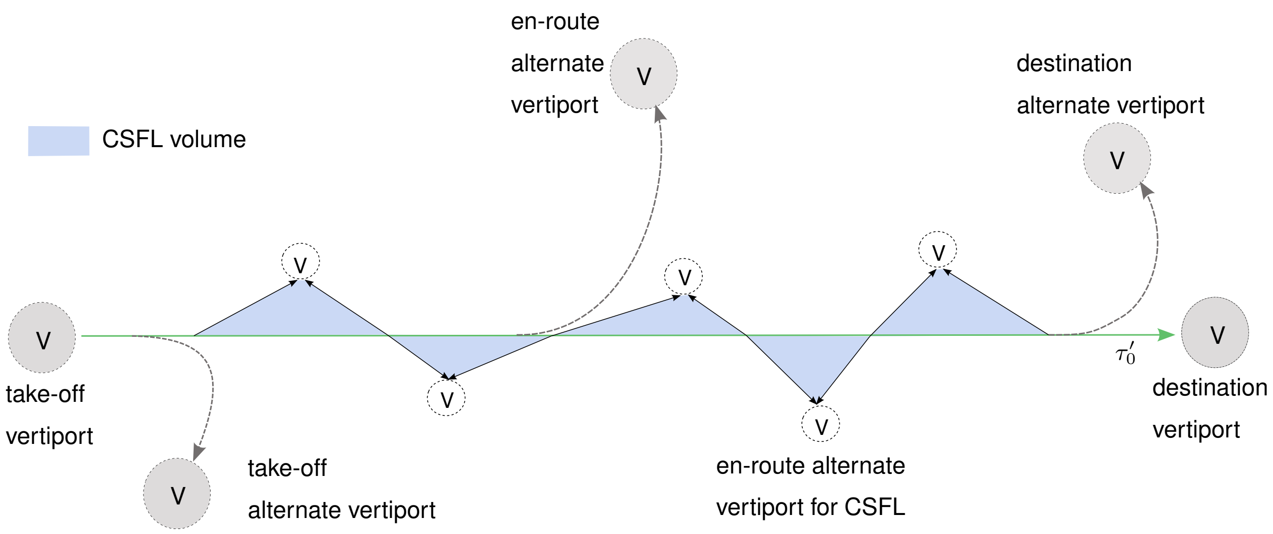


Figure 3-4: Categories of vertiports and their allocation along a predefined trajectory τ_0'. Flight volumes highlighted in blue color represent areas, within which the respective alternate vertiport for CSFL will be selected in a CSFL event. Vertiports for different purposes can be subject to different requirement sets and provide different services. Figure is based on [59].

- Take-off alternate vertiports are alternate vertiports for VTOL capable aircraft near the take-off location. They serve as alternate landing sites in cases of aborted take-offs or when the mission is aborted shortly after take-off.
- En-route alternate vertiports are vertiports, at which the vehicle can land with normal performance at the current weather, obstacle and Command and Control Link (C2Link) conditions if a need to divert from the destination is identified during en-route flight.
- Destination alternate vertiports are alternate vertiports near the intended destination.

Additional pre-planned vertiports must be provided for the case of continued safe flight and landing. Such vertiports will only be used in the unlikely event of a safety-critical situation and may therefore fulfil a reduced set of requirements. The different types and structure of pre-planned vertiports are illustrated and explained in Fig. 3-4. Ongoing rulemaking activities focused towards the operation from and to vertiports are based on the existing regulation for operations near aerodromes ([32]) and are extended to account for VTOL-specific properties in [59]. The latest draft version for take-off and landing performance requirements at vertiports is provided in [57].

4 Risk Modelling as an Enabler for UAM Motion Planning

The evaluation of risk is present in all aerospace sectors and provides guidance and reasoning for strategic decisions. European regulation for UAV operations continues to follow the same notion. However, vehicles operating under this new regulation fly at significantly lower altitudes and - in the case of UAM - over congested terrain. This chapter develops a risk evaluation method of modular risk models, which leverages distributed geo-spatial information from dissimilar sources to provide a decision baseline for strategic mission planning for UAVs in urban or other high-risk environments. The framework will be used throughout the remainder of this dissertation to consider risk as a parameter in the planning process and generate cost functions for the evaluation of flight path cost. The following sections are based on [50], in which the method was first published.

4.1 Modular Modelling of Environmental Risk

The analysis and evaluation of the planning environment with respect to the risk, that a UAV operation poses to third ground and air parties is an integral part of UAV operations under the EASA Specific category. Where existing means of compliance may fail over congested areas due to qualitative exclusion criteria and the applied level of abstraction, UAV operations in the certified UAS category create a need for risk modelling methods, which provide high-resolution risk maps of congested areas and a quantitative assessment of a mission's associated risk. Such models require the availability of large geo-spatial datasets, based on which risk models can be developed to describe different aspects of the overall risk. Higher-dimensional risk models may consider and combine multiple risk layers. Such models can only be maintained and deployed efficiently, if the processes of data collection and risk assessment can be automated. This also requires the implementation of independent update cycles for each layer.

The herein developed risk modelling framework, initially introduced in [50], describes each data layer as a discretised risk map of normalised values on the interval $[0, 1]$, where 0 indicates no risk and 1 very high risk. Risk maps are three-dimensional point clouds, in which the resolution can vary with the use case and spatial dimension. For every risk type and layer, a data point in the risk map describes the risk that a UAV passing through a cuboid in space poses to a third party. Risk types and layers refer to a specific base data set, such as an area's population, land use or airspace map. Each risk layer is generated from a two-dimensional impact probability pattern and impact severity map. The impact probability pattern describes the impact zone of a UAS in an out-of-control state and descending towards ground as a probabilistic splash pattern. The impact probability pattern is, hence, described as a probabilistic density function

(PDF). Impact severity maps "[...] quantify the severity of an impact at a specific location [on ground], [...] the quantification of which [...] depends on the operational and regulatory context of the vehicle and mission" ([50]). The convolution of impact severity maps and an impact probability pattern is called a risk map that assigns a risk value to every point in three-dimensional space (see Fig. 4-2 and Fig. 4-3).

Based on the two-dimensional risk map, the third map dimension can be computed from a vertical risk degression model, which extrapolates the initial risk value up to a maximum flight altitude h_{max}. The application of a vertical degression model implies that risk is assumed to decrease, the "[...] higher the UAV flies above the ground surface, due to longer mitigation time. A *linear* degression model yields a [three-dimensional] risk map, where the risk decreases linearly with altitude from the initial risk r_0 towards a residual overflight risk r_r" ([50]). While different valid implementations of degression models may exist, this implementation proposes a linear function $r_{i,d}$ with input parameters d_{zmin}, "[...] [the relative altitude] to the ground surface where risk starts to decrease, and [d_{zmax}, the altitude at which] risk [has] reduced to a residual risk value $r_{i,r}$" ([50]). Equation (4-1) describes the linear degression risk r_d of a ground risk item i at height $h_i \in [d_{zmin}, d_{zmax}]$ above ground as

$$r_{i,d} = r_{i,0} - \frac{h_i - d_{i,z_{min}}}{d_{i,z_{max}} - d_{i,z_{min}}}(r_{i,0} - r_{i,r}), \tag{4-1}$$

where h_i is the altitude above the ground risk item r_i.

4.1.1 The Impact Probability Model

The two-dimensional impact probability model is intended to capture uncertainty in the prediction of the vehicle crash site. It describes the probability distribution of potential impact zones around the predicted crash site. While arbitrarily complex and comprehensive stochastic models may be implemented to provide accurate probabilistic predictions of a vehicle crash site (see e.g. [13]), this work limits the scope of uncertainty modelling to a normal distribution around the reference crash site. This limitation is chosen deliberately and considered sufficient for the purpose of demonstrating the risk modelling framework.

The impact probability at crash sites x is defined as

$$P_{imp}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{4-2}$$

where, for demonstration purposes, the standard deviation σ is set to 3 times the characteristic vehicle dimension to account for debris at impact. An analysis of the actual vehicle crash behaviour may provide a more accurate number. The expectation μ represents the position of the reference crash site x_{ref}. Figure 4-1 represents the discretised approximation of a normally distributed impact probability pattern with a critical

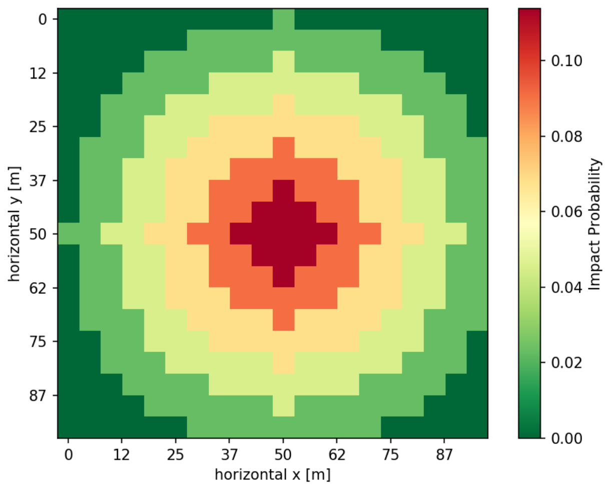


Figure 4-1: Impact probability pattern with a critical impact radius $R_{crit} = 16.5m$, an overall impact radius $R_{imp} = 49.5m$ and resolution $c = 5m$. From [50].

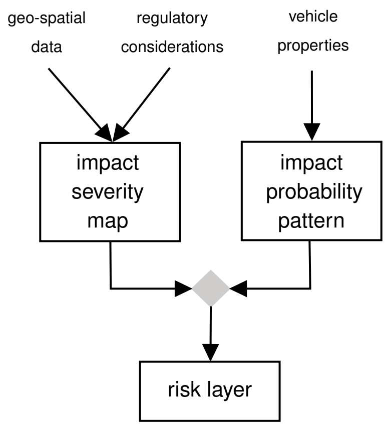


Figure 4-2: The risk map generation process and input parameters.

impact radius $R_{crit} = \sigma = 16.5m$ and overall impact radius $R_{imp} = 3\sigma = 49.5m$ around the reference crash site x_{ref}. The resolution is $c = 5m$. The cumulated impact probability P_{imp} at all points x_n within R_{imp} of x_{ref} is approximated with its 3σ-environment, such that

$$P_{imp} = \sum_{n=0}^{N} P_{imp}(x_n) \approxeq 1, \text{ with } \{x_n \in \mathbb{R}^2 \mid |\vec{x}_n - \vec{x}_{ref}| \leq R_{imp}\} \tag{4-3}$$

holds, where N is the total number points in the discretised impact probability pattern. The impact probability pattern is implemented as a convolution matrix of size $m \times m$, with $m \in \mathbb{N}$ and where $m = \text{int}(2R_{imp}/c)$.

4.1.2 The Impact Severity Model

The impact severity model quantifies the severity of a vehicle's ground impact at a certain location with respect to an underlying geo-spatial data set on the interval $[0, 1]$. The formulation of an impact severity model can be derived from regulatory boundary conditions, common practise or in its most simple implementation, conduct a linear mapping between two data points, which are considered zero and maximum risk. The result is a discretised map with an impact severity score assigned to each point.

Figure 4-3 illustrates how a risk map is obtained from the convolution of an impact severity map and an impact probability pattern. The impact probability pattern is not drawn to scale. While the raw data has no semantic meaning, the risk map connects a data set with knowledge about the vehicle and the operational context. Discrete risk items from individual sources are converted to a continuous representation of risk. For complex environments, risk maps and data layers from many different sources may be required to build a comprehensive risk assessment.

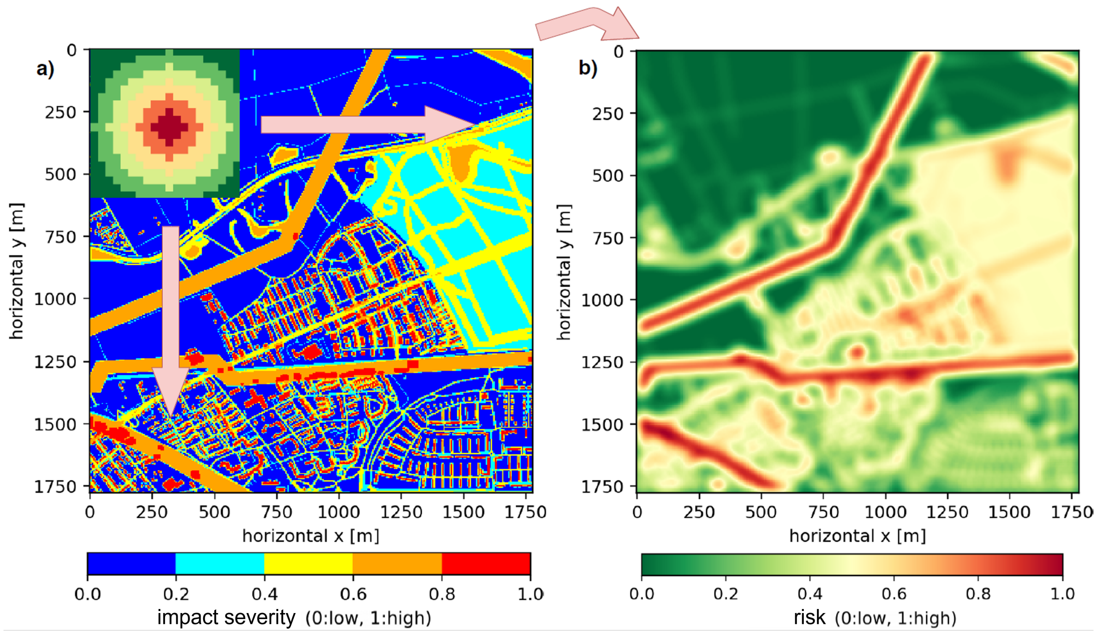


Figure 4-3: A convolution of an impact severity map and impact probability pattern is shown on the left. The impact severity map distinguishes discrete risk levels from different sources. The representation of the impact probability pattern is magnified for better observability. The processed risk map is displayed on the right. From [50].

4.2 Capturing Expert Knowledge in Risk Maps

With larger quantities of risk layers and different data sources, and an increasing automation of the risk map generation and update process, a risk modelling approach needs to develop methods to deal with uncertainty in the completeness and quality of the raw data set. A data set may be incomplete, outdated or corrupted and even if a data set or provider have completed a certification program, the available data may not be able to cover every risk-relevant aspect of an operation. On the other hand, pilot or other expert experience is a valuable asset in making flight operations safe and robust against e.g. changing meteorological conditions. The capability to capture expert knowledge and augment risk maps with this knowledge "[...] is therefore important and necessary, when building safety critical risk models" ([50]). Expert knowledge can also be used to complete incomplete data sets or account for recent changes in the environment, which are not yet captured in the data set. In the following, two different applications of expert knowledge are described.

Editing of data-sourced risk maps: In order to capture and mitigate erroneous or incomplete data sets in auto-generated risk layers and respective input (raw) data, a risk layer can be edited and modified in all three dimensions. One or multiple invalid data points can be overwritten with updated information by the expert. A simple graphic user interface (GUI) enables the expert to edit risk maps using a discrete grid of risk values without altering the raw data set. Since manual editing of risk maps can a have a severe safety impact on the vehicle operation, such experts may require special clearance by the vehicle operator or competent authority.

Generation of new expert risk maps: Beyond editing existing risk maps, the risk database can be augmented with additional layers to allow for a further contribution of expert knowledge. An expert can define high-risk areas and safe-to-fly environments as additional map layers. In the event that exclusive access to a specific flight corridor for use in a mission was granted by a responsible authority, the corridor's geographic shape can be added as a safe-to-fly environment by the authorised expert. Other applications, such as flight planners, may use this information to plan a flight path within the boundaries of the assigned air space. Expert risk maps are treated as independent risk layers, which can be merged with other risk layers to create a risk map composite.

4.3 Merging Risk Maps into Composites

The result of merging semantically related risk layers under consideration of the operational context is called a risk map composite. Risk map composites reduce the map dimension from a potentially large number of data layers to few semantic groups. In the presented framework, risk layers are distinguished and grouped by two criteria. A risk layer can either describe a ground risk or air risk item. "Ground risk is defined as an impact on third parties or critical infrastructure on the ground. Air risk is the risk of a mid-air collision between two parties in air traffics. Mid-air collisions are typically caused by a violation of existing rules of the air" ([50]). Further, a risk layer can display static or dynamic properties. Risk is considered to be static, when its change cycles are subject to days or weeks. Threats and conditions that change within minutes or hours are treated as aspects of dynamic risk and require shorter update cycles.

The computation of a risk map composite may use different implementations, depending on the selected method to reduce the risk model's dimension. A *max-value* function provides a simple but effective implementation to achieve dimensional reduction and results in a conservative evaluation of the operating environment. Each data point in the resulting risk map composite will correspond to the highest risk value found across all input risk layers at this specific location. Similar to the modelling of impact severity, more complex merge functions can be developed and implemented.

Composites computed from static risk layers prior to operation can support the strategic planning process to identify flight routes that avoid known high-risk areas described by the static risk layers. A composite of dynamic risk layers can be computed or updated immediately before take-off. Knowledge of the static and dynamic risk environment supports an informed decision making and enables an operator to select the safest route from a set of options based on the current time and prevailing situation.

4.4 An Architecture for an Automated Risk-Management Framework

For any given environment, a large number of different data sets needs to be collected from different sources and risk maps computed based on an impact severity model, which takes into account regulatory boundary conditions, and a vehicle-specific impact probability pattern. In [50], an application of the framework is described, where five different data types are queried from public databases and processed to generate an automated risk model of an operating environment. It is reasoned, that risk models of approximately $100km^2$ in size can be computed on consumer hardware in about one hour. It is obvious, that the time to provide comprehensive risk models must be reduced by at least two orders of magnitude, when the risk model is coupled with a path planning framework, which computes viable flight paths for on-demand mission in urban environments. For this purpose, a framework is developed, which manages risk layers in a modular setting and decouples the risk map computation from the actual planning query, shown in Fig. 4-4.

The risk map generation pipeline is divided into three distinct modules. The raw data module automates interfaces with a number of available commercial or public databases via APIs. An acquired data set is either forwarded to the risk processing and storage phase directly, or it is combined with another data set into a so-called feature layer. Feature layers are data sets, which are composed of cut sets of one or more raw data types and contain meta information beyond each individual layer's context. A practical example of a feature layer is a cut set of a digital surface model (DSM) and vegetation data, from which potential emergency landing sites can be identified. In this specific example, it is evident, that the DSM alone would not be sufficient to select a landing location, since an area may be flat and free of structures, while heavily vegetated. Raw data sets, relevant for the risk assessment of an operational environment may include but are not limited to the overview provided in Table 4-1.

The risk processing and storage layer implements the risk map generation method described in Section 4.1 and saves each risk map in the risk layer database. Following the principle described in Fig. 4-2, each raw data set or feature layer requires a dedicated risk model, consisting of an impact severity model and impact probability pattern.

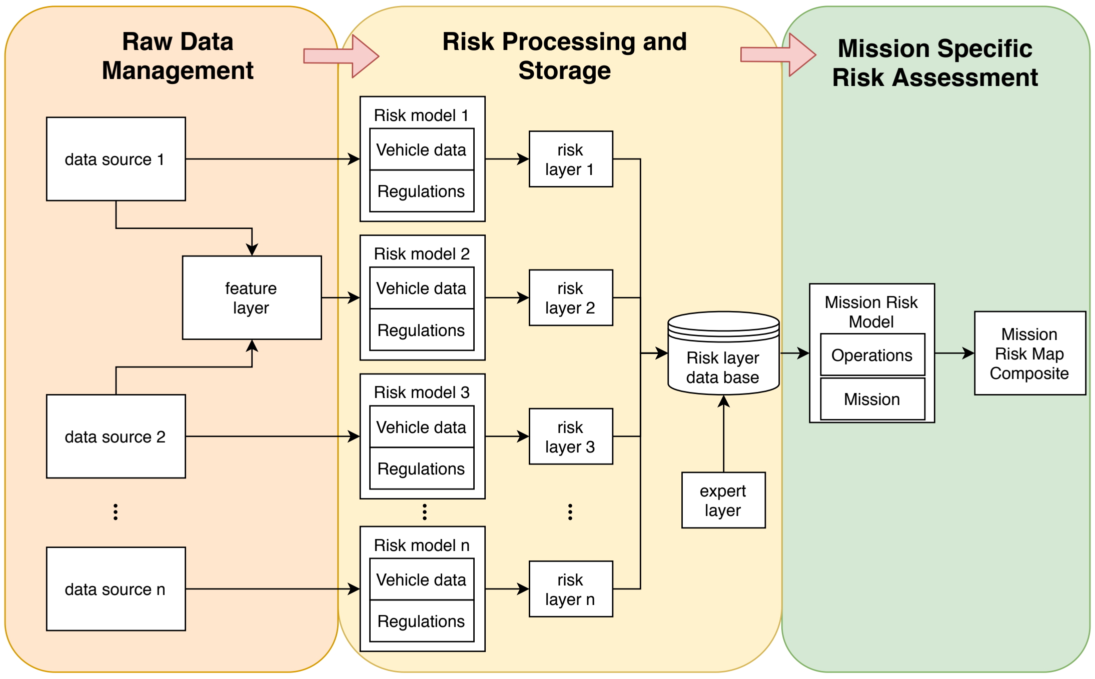


Figure 4-4: Required data sets are queried automatically from public and commercial databases. From this data, risk layers specific to a vehicle and regulatory context are generated and saved in a data base. Mission-specific risk models implement a comprehensive analysis of a mission's risk. The analysis of mission risk is decoupled from the acquisition of raw data to enable shorter response times. From [50].

They transform the raw data set into a semantic description of the risk associated with the input data under consideration of the vehicle and regulatory context. The risk layer database contains all auto-generated risk models and expert-generated risk layers and allows expert users to inspect, edit or correct any risk layer in the database. Since each raw data interface and risk model are implemented as independent modules, each risk layer can be updated and recomputed according to the expected rate of change of the underlying data set. Hence, the database can be configured to contain a sufficiently recent version of each risk map at all times without unnecessary re-computation cycles of risk maps, which are still valid.

While the previous two phases of the risk modelling framework are managed and triggered independently from a consumer request, the mission specific risk assessment phase decouples the query-specific risk assessment from the risk modelling and computationally expensive risk map computation. Where each risk map in the risk layer database is specific to a vehicle and regulatory framework, it is agnostic towards the exact mission and ConOps, under which a mission shall be executed. The mission specific risk assessment incorporates mission- and operation specific parameters into the risk assessment and implements a mission risk model, which uses a subset of the risk layer database to conduct a risk assessment at the time of incoming queries. It merges risk layers into risk map composites of semantic groups and superposes all applicable

Table 4-1: A list of potentially relevant raw data sets and classification into semantic groups.

	static risk	**dynamic risk**
ground risk	terrain vegetation structural obstacles roads population density land use	public events commuter movement traffic congestion
air risk	permanent no-fly-zones air space structure airport approach routes	temporary no-fly-zones air traffic

composites to obtain a comprehensive risk assessment in a single final mission risk composite. Mission and operation parameters are considered in the form of weighting factors $w_{l,i}$ and $w_{c,i}$, respectively. The factor $w_{l,i}$ represents the risk layer weight in a composite, whereas the factor $w_{c,i}$ describes the composite weight in the final mission risk map. For each semantic group of Tab. 4-1, risk maps of applicable data layers L_i, with $i = 1, 2, ..., n$, are therefore merged into a mission-specific risk map composite C_m using a weighted *max-value* function over all data points $x \in L_i$

$$C_m(x) = max(w_{l,1}L_1(x), w_{l,2}L_2(x), ..., w_{l,n}L_n(x)). \tag{4-4}$$

Mission-specific parameters, which may affect the weighting of risk layers, include but are not limited to the type of the vehicle payload or mission objectives, which increase the probability of a collision with the terrain or structures. Given the expectation that vehicles operating in UAM scenarios under the EASA certified UAS category will be certified to the same level of rigour as commercial aircraft, a weighted max-value function is considered suitable to prefer areas of moderate risk across a range of risk types over areas with a single high-risk item. Following the same notion, the final mission risk map C_{final}, is computed as a composite of mission-specific risk map composites $C_{m,j}$, with $j = 1, 2, ..., p$, such that

$$C_{final}(x) = max(w_{c,1}C_{m,1}(x), w_{c,2}C_{m,2}(x), ..., w_{c,p}C_{m,p}(x)), \text{ with } x \in C_{m,j} \tag{4-5}$$

considers operational factors in the risk assessment. Operational parameters of interest can be weather conditions or characteristics of the operation, such as the availability of data services, occlusion of a C2-link or limited satellite coverage, flight in segregated airspace or over special events. "With a central database containing the most recent version of each risk layer, large numbers of mission risk models can be handled and maintained efficiently, as the update cycles of individual risk layers [L_i] and the final mission risk model [C_{final}] are decoupled. Mission specific risk models are not limited to UAS operations but can be transferred to [different kinds] of aerial missions beyond the limits of qualitative risk evaluation methods, [such as SORA]" ([50]).

Within the scope of this dissertation the following chapters will build on the presented risk evaluation method to translate known aspects of an operating environment into a quantitative risk metric and integrate risk as a cost factor in the assessment of flight path cost.

5 Development of a Motion Planning Framework for Safe UAM Operations

This chapter describes a motion planning framework and planning methods, which solve the planning problem for UAM operations over congested areas in a deterministic and verifiable manner. The framework proposes to distinguish between a set of nominal routes, from which a decision making entity onboard the vehicle or on the ground can select a sequence of paths towards the destination vertiport and contingency planning methods, which implement a verifiable notion of SC-VTOL's continued safe flight and landing. Emergency situations, which require an immediate landing outside a pre-determined landing site, are not within the scope of current regulatory activities and neither subject to consideration in the set of proposed planning methods. However, the framework may be extended with planning methods designed for the specific case of emergency landings. A definition of the urban planning problem is provided and applicable assumptions for the simplification of the planning problem explained. Furthermore, it is reasoned how this work's contributions extend the identified state of the art (see Sections 1.4 and 1.2, respectively) to enable safe UAM operations.

5.1 Challenges in the Automation of Urban Air Mobility

The path planning task for UAM applications is set in an environment, which is challenging from both a safety and regulatory perspective. UAM vehicles operate at expected altitudes between 300ft and 1000ft AGL in obstacle-dense environments, where the effect of a ground impact is severe. The environment is partially known and partially controlled by the vertiport operator. Static obstacles such as structures and permanent airspace restriction are required to be available as map data or digital surface models. Since current EASA rulemaking limits the operational scope of UAM missions to connections between pre-determined vertiports, critical flight manoeuvres during take-off and landing will be contained in a controlled environment, unless an emergency event occurs.

By the time of the introduction of first commercial UAM services, availability of U-Space services is expected (for reference see [73] and [46]), which will take a role similar to ATC in controlled airspace. U-Space will support the operation of multiple aerial vehicles in the urban air space with a growing number of basic aviation services, among which are the four mandatory U-Space services

- network identification service,
- geo-awareness service,

- traffic information service and
- UAS flight authorization service.

The presence of U-Space services ensures the safe integration and strategic deconflicting with other U-Space air traffic based on the evaluation and scheduling of flight plans, which are proposed to the U-Space provider by the respective vehicle operator. In order to facilitate strategic traffic deconfliction, the EASA certified UAS category requires UAVs to follow preplanned routes in segregated corridors and mandates the U-Space provider to approve each route prior to flight. With increasingly dense UAM operations, however, it will become more and more difficult to ensure safe and efficient operations without a capability that enables in-flight decision making to handle unexpected events and tactical deconfliction tasks. Under the existing regulatory framework, such in-flight decision making methods will be required to display deterministic properties, be transparent to the competent authority and allow inspection of the solution space prior to flight.

On the other hand, public acceptance plays a role to the success of UAM. Beyond the frequently discussed topics of noise and privacy concerns, whether or not automated UAM operations are perceived as a safe technology, is expected to be an important factor to the acceptance of UAM services. The flight planning process should therefore be designed such that the process itself and the results it produces, provide performance guarantees and can be explained to the interested public.

The flight planning framework developed in this chapter leverages extensive pre-processing of the partially known environment and pre-computation of partial solutions to the online planning problem to reduce the complexity of the planning task in flight. It further guarantees to provide a feasible solution to the planning problem at flight time within bounded runtime and memory requirements, given that the pre-computed partial solution has passed the pre-flight inspection.

5.2 Assumptions on the Motion Planning Problem

The following list of items derived from Chapter 3 and Section 5.1 describes the underlying assumptions, on which the planning framework is built, and defines prerequisites, which are required to make applicable simplifications to the planning task.

A-1: The obstacle environment is considered quasi-static over the duration of a mission. Although subject to constant change in most urban environments, the structural obstacle map's rate of change is considered slow relative to the duration of a UAM mission. Structures are expected to evolve within time intervals of days or weeks, whereas a typical mission duration t_m is assumed to be $t_m < 30min$. The same logic applies to airspace restrictions, which are announced prior to effect. The static obstacle map is therefore considered known at the time of planning and over the course of a mission.

A-2: Vehicles operating within the scope of UAM can only operate between and within a network of dedicated vertiports, which are mapped and known to all parties prior to a flight planning request. Once a network of vertiports exists, the same logic as for structural obstacles is applied regarding its rate of change. It is therefore assumed that vehicles travel between the same set of take-off and destination vertiports repeatedly and frequently.

A-3: Vertiports are under the control of the vertiport operator. Availability of a vertiport can be confirmed prior to the final landing approach.

A-4: Continued safe flight and landing applies to all UAM operations. A vehicle landing outside a pre-determined and controlled vertiport is considered a catastrophic event, which must not occur at a probability $P_{cat} > 10^{-9}\frac{1}{fh}$. All off-nominal operations within the scope of CSFL are considered contingency operations. Catastrophic occurrences, which require a landing at an unprepared site, are referred to as emergency operations and are beyond the scope of this work.

A-5: The mandatory U-Space services network identification, geo-awareness, traffic information and flight authorization are available. Once a mission is approved, the flight path can only change within the limits of the approved flight plan.

A-6: The vehicle has knowledge of dynamically moving obstacles such as other air traffic. This information is either provided via U-Space or acquired by the vehicle using an on-board perception capability.

A-7: Obstacles are defined from ground to a specific altitude above ground level and without undercuts. This means that e.g. bridges are represented as solid obstacles from ground to the roadway, with no option to pass below. This assumption is referred to as planning in 2.5 dimensions.

5.3 An Architecture for Deterministic UAV Path Planning in Urban Environments

The general approach in this dissertation to solve the path planning problem in a complex high-risk environment and under consideration of regulatory and operational constraints follows two governing principles: First, the method uses prior knowledge about the ConOps and intended mission profile (compare assumptions A-1 to A-5) to reduce the problem scope and divide the overall task into a number of smaller sub-tasks. Second, the fact that the planning environment is partially known and controlled is exploited to reduce the computational load at flight time. According to contribution C-1, the flight

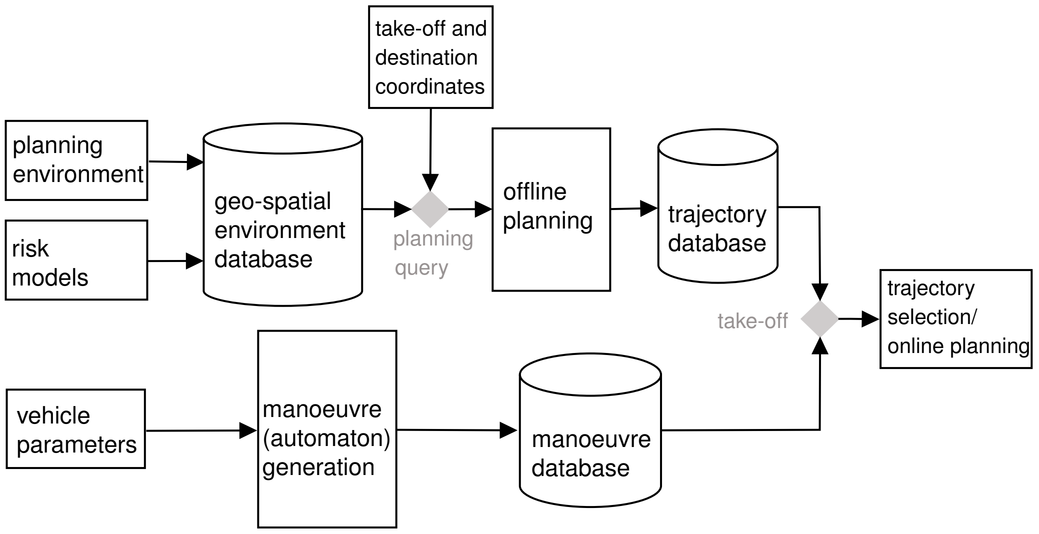

take-off and destination coordinates
planning environment
risk models
geo-spatial environment database
planning query
offline planning
trajectory database
take-off
trajectory selection/ online planning
vehicle parameters
manoeuvre (automaton) generation
manoeuvre database

Figure 5-1: Macroscopic flow chart of the mission planning process. Geo-spatial data sets of the environment and aircraft parameters are pre-processed prior to a planning request. The pre-planning phase reduces the computational effort at flight time.

planning task can be decomposed into an offline and online planning phase, where the offline phase covers all planning steps, for which the required input data are known prior to flight. In a network of vertiports with a known obstacle map, flight planning for nominal and contingency scenarios can be handled entirely in the pre-planning phase, such that the online planning problem can be collapsed to a trajectory selection problem. This comes with the advantage that the time-criticality of online planning tasks is removed in this approach and that a precomputed flight plan can be validated for safety and compliance with applicable rules before flight.

The macroscopic structure of the proposed planning architecture is illustrated in Fig. 5-1. If the planning environment is (partially) known prior to a mission-specific query, risk models for this environment can be generated following the process described in chapter 4 and provided to the flight planning method as inputs in extension to available unprocessed environmental data sets such as surface models and geographical maps. Similarly, a manoeuvre library of trajectory segments and manoeuvres, so-called motion primitives, for later use in an online-planning method can be computed if the physical properties and performance envelope of the vehicle are known. If also the operational envelope is known, control laws can be derived from each motion primitive. These control laws should minimise the vehicle's reachable set within the boundaries of the maximum disturbance corresponding to its operational envelope. Within the scope of this dissertation, the term reachable set is used to refer to the envelope of the maximum deviation from a reference state over the execution of a primitive or trajectory, whereas the operational envelope describes environmental conditions, in which the vehicle can operate. An incoming planning request triggers the pre-planning phase. The planning result is transferred to a database onboard the aircraft and used to se-

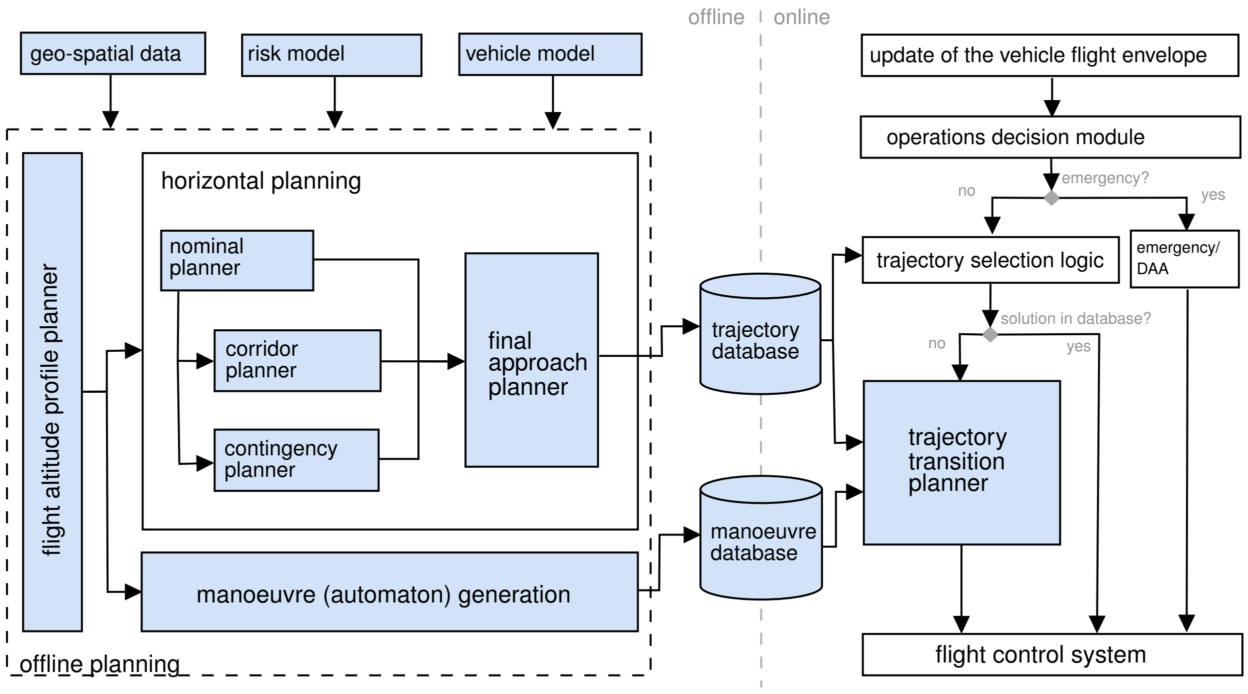


Figure 5-2: Microscopic structure of the planning framework. Flight-phase and operations-specific planners are allocated to the offline and online planning phase. A trajectory and manoeuvre database are generated in the offline phase and leveraged in the online flight planning.

lect the best action from a number of pre-planned path options at flight time. If events or contingency situations occur, for which no suitable action exists in the pre-planned trajectory database, an online planning algorithm restores a safe flight state using the equally pre-calculated manoeuvre database. A safe flight state can be a state on any pre-planned trajectory in the database. The online planning method is described in detail in section 5.7.

Figure 5-2 structures the planning process into multiple sequential planning methods treating different operational states and flight phases. The operational state refers to the vehicle operating under nominal, contingency or emergency conditions, whereas flight phases describe different sections of the overall mission, such as take-off, cruise or landing. At the interface between any two planners, the flight path's time derivatives must be continuous. This is ensured through appropriate boundary conditions. The actual mission planning in the offline phase is preceded by the previously described preprocessing of aircraft-related and environmental data. This step relies on the assumption of a predominantly controlled environment in the case of nominal operation of the aircraft. For the trajectory planning in non-safety-critical operating conditions, the flight altitude profile is computed in the flight altitude profile planner based on the environment's topology and a set of planning rules. The method follows the process described in section 5.4.1. The flight altitude planner produces a three-dimensional planning surface, which represents the intersection of the configuration space and the flight altitude profile. Hence, the definition of the flight altitude profile is decoupled from the planning in the horizontal plane. On the resulting three-dimensional surface, different quasi-two-dimensional planning approaches are used, depending on the respective planning objective. Based on initial valid solutions for nominal operations, a corridor planner can be deployed to derive corridors of segregated airspace and provide flight paths in opposite directions at safe distances to enable efficient commuter scenarios. This approach is called the rule-based planning approach, which is described in detail in Sections 5.4 and 5.5.

For nominal operations, the horizontal planning can e.g. use a graph-based approach paired with a cost function, which maximises operational safety and economic efficiency. In the case of contingency operations, the provision of as many and safe response options as necessary becomes a contingency planning method's main objective. Different planning approaches for different contingency scenarios can be implemented in parallel to obtain sets of solutions, which cover a large spectrum of vehicle degradations and operational hazards. The offline nominal, corridor and contingency planning methods are considered as planners, which are specific to the operational state. They are paired with flight-phase specific planners to ensure compliance with procedures and additional safety objectives during critical flight phases. The near-vertiport planner ensures safe operations during the approach onto a vertiport and in the immediate proximity of vertiports during take-off and landing. If required, these two planning objectives may be split into separate planning methods. All planning solutions, which are computed during the pre-planning phase are stored in a trajectory database and transferred onboard the aircraft. This database can be checked to provide a sufficient number of solutions to the planning task within the operational scope and relevant regulatory framework. Similarly, all trajectories in the database can be validated to comply with all applicable rules and target safety levels.

Although most operational cases can be covered within the scope of the pre-planning phase and the assumption of a partially controlled environment, such methods can only reduce the number of scenarios, in which true online planning methods are required. Following the notion to simplify an online planning task through offline computations of partial solutions, the framework proposes to generate motion libraries, which reflect the vehicle performance, prior to flight. The motion library and, optionally, navigation functions derived from the library are saved onboard the aircraft, too.

During flight, a decision logic classifies the respective flight condition on the basis of information provided by e.g. a runtime monitoring system and selects between two planning approaches. A logic module collapses the planning task for nominal and contingency operations to a trajectory selection task at flight time, reducing the time-criticality of online-planning tasks substantially. Emergency situations, which affect the flight safety or manoeuvrability of the aircraft significantly and are already outside the permissible range of regulations, are considered in an online planning approach. This emergency planner has the task to restore a safe operating condition or, if necessary, terminate the mission with minimal damage to the aircraft and persons involved. Comfort, efficiency or operational considerations are irrelevant in this case. It is desirable to implement emergency planners as manoeuvre-based methods, which use the pre-computed motion library to reduce the time required for the calculation of a valid solution and ensure short response times. Furthermore, the approach allows to take manoeuvring limitations into account by excluding the affected manoeuvres from the subset of admissible manoeuvres.

An additional contingency online planning module is proposed for large and/or complex planning environments, which lead to large trajectory databases. In such cases, a contingency online planning method can enable transitions between pre-computed trajectories beyond a set of intersections or fork points. The method provides more options for action with a constant database size. Alternatively, the database size can be reduced, if the required number of options remains constant. In order to not violate the objective to provide a deterministic and transparent solution at all times, this planning method must follow pre-determined patterns and be contained within a pre-determined, bounded volume. A method complying with these requirements is presented in Section 5.7.

The remainder of this chapter focuses on proposing implementations for the individual planning steps of the proposed framework, which are within the scope of EASA regulation for UAM. The relevant modules are highlighted in Fig. 5-2 in light blue colour.

5.4 A Method to Enforce Rule-Compliant Path Planning Results Through a Reduction of the Search Space Dimension

This section describes a rule-based approach for flight planning similar to helicopter VFR/IFR operations and was initially presented in [47]. An early feasibility assessment of the concept in a reduced scope can also be found in [9]. The section is a partial excerpt from the original publications [47] and [50] and covers the flight altitude profile planner, the nominal planner and near-vertiport planner listed in Fig. 5-2. Referring to contribution C-2, it enforces a rule-compliant path planning result in compliance with applicable regulation [68] and [59] through a reduction of the search space dimension. To lower the validation barrier for the proposed process, rule-based planning implements a formal approach which derives planning rules from applicable boundary conditions. The set of rules is then used to define strict altitude profiles and limit the planning task's search space. The method provides predictable planning results at comparably low complexity.

Following a vertical take-off, the method defines a path angle γ for all flight phases that include a climb or descend. Depending on the vehicle and environment and inspired by EASA air operations for helicopters ([18]), the path angle varies between $6°$ and $15°$. Further, concepts of vertical deconflicting, that are well-established in commercial and general aviation are applied to the urban scenario. The UAV shall maintain a constant altitude in cruise phases and shall not deviate more than a maximum value from the cruise altitude h_c. "The cruise altitude is selected from a discrete grid of pre-defined altitudes [$h \in H$] similar to flight levels and the semi-circular rule ([55]). While this approach covers only vertical deconfliction between airspace participants, [...] additional definitions for the minimal vertical clearance from static obstacles [are required]. Mandatory heights above applicable ground infrastructures (e.g. following [suggestions in [32] and [18]]) are enforced through augmentation of the respective obstacle height with the to-be-enforced clearance [$d_{z,min}$]" ([47]). The basic rule set for UAM operations in the scope of this dissertation contains the following items:

1. UAV operation is limited to a set of n discrete flight altitudes $h_{c,i}$, where $0 \leq i < n$.

2. Climb and descend phases are restricted to funnels F_j with angle γ_j around each vertiport VP_j.

3. The final cruise altitude h_c is adjusted for the semi-circular rule, using an offset $0 \leq d_{z,sc} \leq d_{z,sc,max}$, where the parameter $d_{z,sc,max}$ describes the distance between two semi-circular flight levels of the same flight direction. Reasonable distancing between flight levels shall be selected in accordance with the intended operation.

4. A minimum vertical distance to obstacles $d_{z,min}$ must be maintained at all times. The value of $d_{z,min}$ may be specific to the flight phase.

5. A minimum horizontal obstacle clearance $d_{xy,min}$ must be maintained at all times. The value of $d_{xy,min}$ may be specific to the flight phase.

6. During nominal operations, there must be a landing site within a contingency landing radius R_s at all times.

A typical flight altitude profile connecting two vertiports VP_1 and VP_2 and complying with the above conditions 1.-4. is provided in Fig. 5-3.

According to the general notion to limit commercial air operations to flights between aerodromes, the presented approach allows vehicles to take-off and land only at dedicated vertiports. The following section explains how the defined rule set is exploited to simplify a path planning problem between two vertiports by reducing the search space dimension.

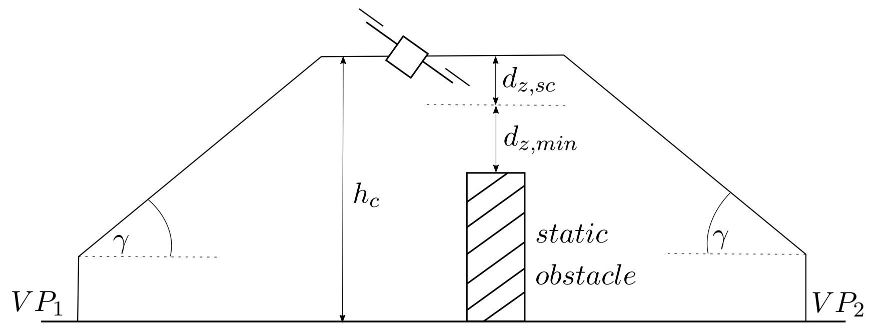


Figure 5-3: An altitude profile between two vertiports VP_1 and VP_2. Regulatory and kinematic limits are respected according to the described rule set. Obstacles are passed at cruise altitude h_c with a minimal clearance $d_{z,min}$ and directional altitude adjustment $d_{z,sc}$. During climb and descend a path angle γ is enforced. From [47].

5.4.1 A Method to Reduce the Search Space Dimension

The flight altitude profile presented in the previous section can be formalised into a flight altitude profile planner (compare Fig. 5-2), which derives the vertical vehicle movement across the mission from a rule set. This separation of the vertical planning phase from the horizontal planning phase allows to reduce the three-dimensional path planning problem to a problem in only two dimensions. Since the altitude profile computed in the flight altitude profile planner represents applicable regulation and rules, solutions from consecutive planning steps executed on the altitude profile comply inevitably with the same rule set. This reduces the valid solution space to a three-dimensional surface.

Approach and departure funnels $F_{VP,i}$, where $i \in \{1, 2\}$, and with a constant descend and climb angle γ are defined around each vertiport VP_i according to Fig. 5-3 and Fig. 5-4. The flight altitude profile planner further defines a cruise altitude h_c, which will be applied throughout all consecutive planning steps during the cruise phase. According to this method, a connection between any two vertiports VP_1 and VP_2 is described by a well-defined altitude profile

$$H = f(F_{VP,1}, F_{VP,2}, h_c). \tag{5-1}$$

A deviation from this altitude profile H between two vertiports can only occur in the cruise phase if different flight levels are selected for missions. The altitude profile is defined by the lower boundary of the union of vertiport funnels $F_{VP,1}$ and $F_{VP,2}$ and the cruise altitude h_c (see Fig. 5-4), such that

$$H(x,y) = min \begin{cases} F_{VP,1}(x,y) \\ F_{VP,2}(x,y) \\ h_c \end{cases}. \tag{5-2}$$

Models of no-fly zones and restricted air space near critical ground infrastructure, which affect the admissible flight altitude can be considered in the altitude profile in

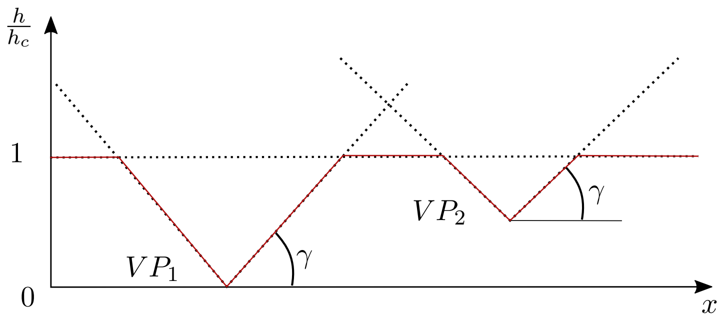


Figure 5-4: The cross section of a representative altitude profile (red) between vertiports VP_1 and VP_2 is derived from Fig. 5-3 according to (5-2). From [47].

a similar manner to vertiport approach funnels. Such no-fly zones would appear as additional terms on the right-hand side of (5-2).

While many different approaches to determine the cruise altitude h_c may be implemented, this demonstration of the search space reduction method computes it from the expected obstacle environment. The subset of obstacles $\mathcal{O}_g$ on a straight line g between the take-off and destination vertiport is identified from the obstacle environment $\mathcal{O}$. The cruise altitude h_c is selected as a heuristic

$$\bar{z} = \frac{1}{N} \sum_{n=0}^{N} z(\mathcal{O}_g^n) \tag{5-3}$$

of the average obstacle height on the identified line and augmented with a safety margin $d_{z,min}$. According to the semicircular rule for General Aviation aircraft, which corrects a selected flight altitude based on the aircraft's heading, this altitude is additionally adjusted with an offset $d_{z,sc}$ accounting for the primary flight direction and is subject to an upper bound on the cruise altitude $h_{c,max}$, representing the operations ceiling, such that

$$h_c = \min(h_{c,max}, \bar{z} + d_{z,min}) + d_{z,sc} \tag{5-4}$$

holds. The configuration space can be sliced along the altitude profile $H(x, y)$ to restrict the search space to the remaining valid solution space, which is represented by the three-dimensional surface $\mathcal{S}^*$ and obtained as described in Fig. 5-5.

The obstacle environment of a digital surface model, in which structures have been placed randomly is displayed in Fig. 5-5a). The top view of this obstacle map is shown in Fig. 5-5b), which corresponds to the obstacle environment at ground level. The three-dimensional model is sliced along the altitude profile $H(x, y)$ in Fig. 5-5c) (compare to Fig. 5-4) to obtain the flight surface $\mathcal{S}^*$. The flight surface $\mathcal{S}^*$ represents the altitude profile between vertiports VP_1 and VP_2 described by (5-2) and complies with conditions 1.-4. of the introduced rule set. "The final planning surface $\mathcal{S}$ is generated as the projection of the surface $\mathcal{S}^*$ into the horizontal xy-plane. Equation (5-5) describes the mapping from the initial [three-dimensional] flight surface model to the [two-dimensional planning] surface $\mathcal{S}$" ([47]).

$$\mathcal{S} : \mathbb{R}^3 \to \mathbb{R}^2,\, (x, y, z) \mapsto (x, y) \mid z = H(x, y) \tag{5-5}$$

The new surface $\mathcal{S}$ provides a reduced and rule-compliant search space for the path planning problem. The reduced search space's remaining obstacle environment is displayed in Fig. 5-5d). The number of obstacles on $\mathcal{S}$ that need to be considered in the path planning problem is considerably lower than in the initial obstacle situation in Fig. 5-5a) and Fig. 5-5b). This is due to the effect, that obstacles below the altitude profile $H(x, y)$ disappear from the set of obstacles, as the aircraft will pass them with safe vertical clearance.

The described procedure results in a planning in 2.5 dimensions. Truly three-dimensional structures such as overhangs and passages under structures are treated as members of the obstacle set. This means, that the approach requires a single distinct altitude limit for each coordinate, above which it is a member of the free space. This limitation is chosen deliberately to facilitate the implementation of contingency planning methods in the following sections. The consideration of airspace classes in the computation of the flight altitude profile $H(x, y)$ (see (5-2)) is required by the regulatory context and is therefore a necessary exception to the rule. Constraints on the vehicle dynamics and performance such as the climb or pitch rate, which only exist in 3D, are considered as boundary conditions in the subsequent trajectory generation. Depending on the respective flight phase, different trajectory generation methods may be applied. The described method to reduce the search space dimension requires that pre-processed maps are available for each pair of vertiports in the service network as outlined in Fig. 5-1 and according to the process described in chapter 4. Based on the assumption that UAM operations will be introduced as reoccurring flights between vertiports of a network, memory requirements for the environment representation can be reduced considerably when few discrete flight levels are used. For each of these flight levels the planning surface $\mathcal{S}$ is stored as well as the coordinates of each vertiport and the 2D-projection of the approach funnel until the highest flight level or the maximum cruise altitude. Once acquired, the planning surface remains valid and does not need to be recomputed until a change of the operational boundary conditions or obstacle environment occurs. The planning surface for any vertiport pair in the network is obtained from the superposition of the planning surfaces of the respective flight level and vertiport funnels.

5.4.2 Rule-Based Planning in Quasi-2D-Space

A variety of different path planning methods exist, which are suitable to connect two states in the free configuration space on the projected two-dimensional flight planning

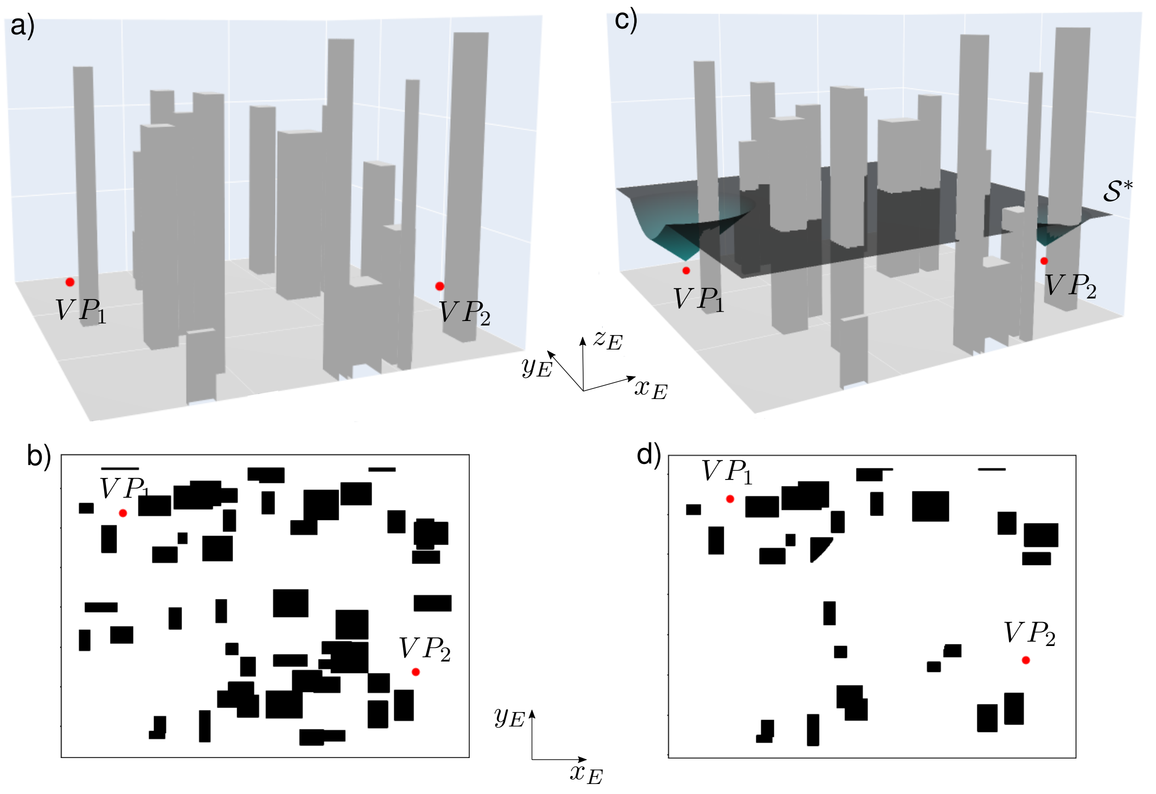


Figure 5-5: Simplification of the search space: An obstacle environment with two vertiports is shown in a). b) displays the top view of the same environment. In c), the obstacle environment is sliced along the flight altitude profile (see Fig. 5-4), which results in a flight surface $\mathcal{S}^*$. d) displays the obstacle map of the final planning surface $\mathcal{S}$. From [47].

surface. In the present case of a network of vertiports, each of which requires substantial investment and infrastructure, it is assumed, that the planning environment and the set of possible take-off and destination locations change slowly over time. It is therefore considered favourable to select a roadmap method, which allows to reuse a once computed search graph for later queries. Relying on e.g. [52] for the description of wind conditions near structures, design goals for a planning method change in the proximity of a vertiport and during take-off and landing manoeuvres. During these flight phases, velocities over ground are comparably low, which means that the impact of wind on the resulting vehicle motion can be substantial even at low wind speeds. Hence, the motion planning approach needs to be more sensitive to wind than in other flight phases. The global planning method is therefore extended with a local planner around each vertiport. The explanation of the global rule-based planner and final approach planner are based on the original work [47] and [50].

5.4.2.1 Global Planning with a Maximum Clearance Roadmap

Following the notion of flight safety to maximise the distance between the graph's edges and obstacles, a maximum-clearance roadmap $\mathcal{G}$, computed from the configuration space's Voronoi diagram, is implemented. A comprehensive overview of roadmap algorithms and Voronoi diagrams is provided in Chapter 2. Once the maximum clearance roadmap $\mathcal{G}$ is obtained, edges that violate a minimum horizontal clearance requirement $d_{xy,min}$ are removed from the graph according to rule 5. If the removal of such non-compliant graph edges generates disconnected components of $\mathcal{G}$, the largest remaining subcomponent is selected to become the new graph $\mathcal{G}^*$. To ensure that a contingency landing site is always within reach along a flight path and to exclude unsafe operational scenarios from the solution space, a radius R_s is defined, within which a landing site must always be available. Graph edges that are outside R_s around at least one vertiport are deleted, too (see rule 6). A final step connects all vertiports to the graph. Vertiports, for which there exists no rule-compliant connection to the graph are discarded.

A Dijkstra algorithm ([16]) is used on the roadmap of remaining rule-compliant edges to search the free configuration space $\mathcal{C}_{free}$ for the most cost efficient and collision-free path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ between the start and destination state and according to a cost function C. The obtained optimal flight path on $\mathcal{G}$ is smoothed using the strategy presented in Fig. 5-6. Starting with the initial position 1, the current graph node is kept and nodes skipped until the direct link between the current and the new node is more expensive than linking the two nodes via the previous node or a collision with the obstacle set occurs. Skipped nodes are removed from the path. Then, the current node is updated to the last node to which a collision-free short-cut exists. This process repeats until the end of the path is reached. Due to the non-uniform distribution of graph nodes

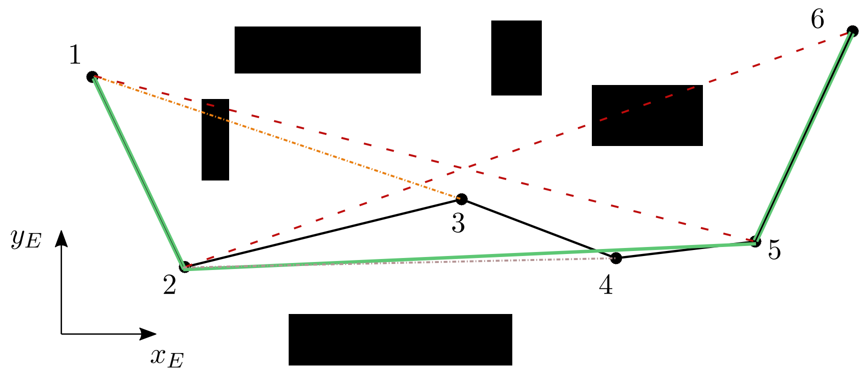


Figure 5-6: The initial path (black) is smoothed starting in position 1, from which nodes are skipped until the direct link to the new node is more expensive than linking the two nodes via the previous node. No simplification exists between 1 and 3 (orange). A short-cut is found between 2 and 5, whereas the backward search from position 6 achieves no further simplification (red). The final path is depicted in green. From [47].

and obstacles, the smoothing result can often be improved further with an additional smoothing step being applied backwards. New edge weights are computed under consideration of the applicable cost function C, such that the cost of the smoothed flight path τ is consistent with the chosen cost metric. Every edge generated during the smoothing process is checked for violations of the applicable rule set and penalised with infinite cost when invalid, such that the smoothed path is rule-compliant.

The definition of a cost function C is largely dependent on the objective, that the cost function shall achieve. Since the rule-based approach ensures feasibility of the solution in the pre-processing of the configuration space, the matter of feasibility is decoupled from the planning step. The latter focuses on optimality of the solution by minimizing a cost function, however, any solution, even if suboptimal, will be feasible. In the application of UAM, it seems favourable to define optimality in an operations and safety context. Beyond the risk, which is associated with the presence of an aircraft at a certain position in space, environmental conditions such as wind are factors to be considered in the evaluation of a flight path. The function describing an edge's total cost C_{tot} is proposed to consider the risk cost C_r, the edge length L_e and a wind penalty factor p_w, such that C_{tot} is expressed as

$$C_{tot} = (p_w L_e)^{w_e} \cdot {C_r}^{w_r}, \tag{5-6}$$

where w_e and w_r are empirical weight factors representing energy efficiency and risk, respectively. The risk cost C_r is computed following the approach described in chapter 4. The wind penalty factor p_w describes how energy cost increases under detrimental

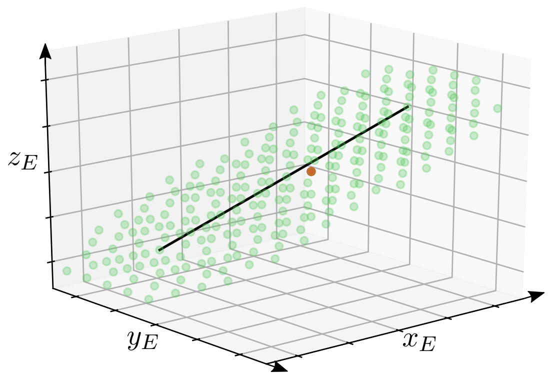


Figure 5-7: The highest risk value (orange) in a cylindrical volume with radius r_R around the edge is used to define an edge's risk value. The radius r_R corresponds to the vehicle dimension. From [50].

wind conditions. It is defined as a function of the wind speed along a graph edge $v_{w,e}$ and the maximum airspeed $v_{A,max}$ that the vehicle can achieve

$$p_w = \begin{cases} max\left(0.5, 1 - 0.5\frac{v_{w,e}}{v_{A,max}}\right) \text{ if } v_{w,e} > 0 \\ e^{\frac{1}{1+\frac{v_{w,e}}{v_{A,max}}}-1} \text{ if } -v_{A,max} < v_{w,e} \leq 0 \\ \infty \text{ else} \end{cases} . \tag{5-7}$$

According to (5-6) and (5-7), the edge cost becomes infinitely large when the headwind $v_{w,e}$ approaches the vehicle's maximum airspeed $v_{A,max}$. In favourable tailwind scenarios, the distance term $p_w L_e$ is reduced to half its original value. Most cost-aware path planning methods obtain the risk associated with a graph edge by integrating risk along the edge in terms of distance or time. This work uses the highest risk value in a cylindrical volume with radius r_R around the edge to define the edge's risk cost C_r (see Fig. 5-7), where the radius r_R corresponds to a typical vehicle dimension such as the rotor radius R or half wingspan b augmented by a scaling factor k to account for imperfect tracking performance. As UAM operations will be subject to SC-VTOL in piloted operations and the EASA certified UAS category for optionally piloted or unpiloted operations, vehicles will be certified to the same level of rigour as conventional aircraft. The chosen approach is, hence, selected to prefer airspace volumes with continuous moderate risk over volumes with few high-risk items in a low-risk environment. Depending on the operational context and mission, weights of individual risk layers in the mission-specific risk model vary according to chapter 4.

Partial planning results, which maintain their validity across multiple queries, can be reused until boundary conditions change and require an update of the respective solution. The sequence of function calls in the rule-based planning approach are described in Algorithm 4. The altitude profile H and flight surface $\mathcal{S}$ are computed for each point-to-point connection in a network of vertiports VP_i and specific to an aircraft type. As long as the altitude profile remains valid, a change in the obstacle map on $\mathcal{S}$ triggers

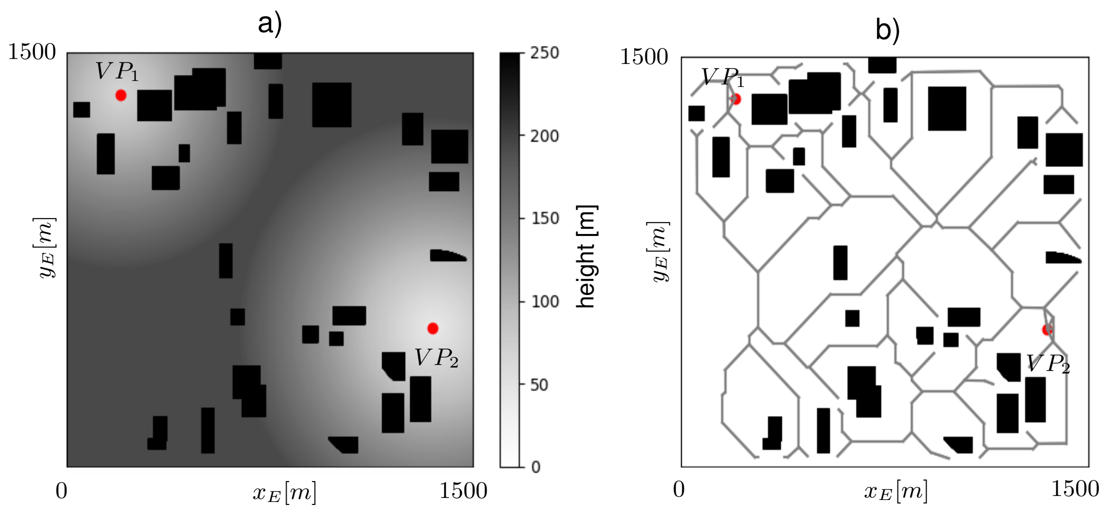


Figure 5-8: Rule-based planning: The reduced obstacle map of a flight surface $\mathcal{S}^*$ is displayed in a). b) shows the pruned roadmap $\mathcal{G}$ computed on the obstacle map of a). Figure is based on [47].

the roadmap generation (see section 2.3.1) and a consecutive pruning step, which reduces the roadmap $\mathcal{G}$ to its subset of rule-compliant edges. The most inner loop checks for a change in the risk model, which does not affect the underlying roadmap, such as changes in the wind direction. In this case, the edge weights w_e in $\mathcal{G}$ are recomputed according to the new cost function C_{tot} and a new flight path τ is obtained from a search on the updated graph. The partial solution from each planning step can be reviewed by the competent authority.

Algorithm 4: A pseudo code representation of rule-based planning update steps.

```
1  $\mathcal{G}$.init(), $VP$.init(), vehicle.init(), rules.init(), riskmodel.init()
2  while mission active do
3      $H \leftarrow$ generate_altitude_profile(vehicle, rules)
4      $\mathcal{S} \leftarrow$ generate_flight_surface($H$)
5      while $H$ valid do
6          compute_roadmap($\mathcal{G}$, vehicle, rules)
7          prune($\mathcal{G}$, rules)
8          connect_vertiports($\mathcal{G}$, $VP$, rules)
9          while $\mathcal{G}$ valid do
10             if riskmodel.update() then
11                 update_edge_weights($\mathcal{G}$, riskmodel)
12                 $\tau \leftarrow$ search_graph($\mathcal{G}$)
13                 smooth_path($\tau$)
```

The rule-based planning process is summarised in Fig. 5-8. The obstacle map of a planning surface connecting two vertiports is illustrated in a). The planning surface is computed following the procedure described in Section 5.4.1 and is unaware of obstacles below its flight altitude profile (see Fig. 5-4 and (5-2)). Figure 5-8b) shows the pruned maximum-clearance roadmap, from which invalid edges have been removed. The prototypical implementation of the roadmap $\mathcal{G}$ computes the graph as the skeleton of a bitmap representation of the obstacle environment using *Python*'s scikit-image implementation of bitmap skeletons based on [78]. Edge weights are assigned to the roadmap $\mathcal{G}$ according to (5-6) and Fig. 5-7 and the weighted roadmap can be exploited to compute a flight path. The final flight path τ is obtained as the waypoint sequence that results from a graph search on $\mathcal{G}$ and smoothed following the procedure described in Fig. 5-6.

The conceptual implementation to derive flyable trajectories τ' from a sequence of waypoints $\mathcal{P}$ on the flight path τ relies on a simple, Dubins-inspired trajectory generation method, which connects two adjacent path legs a and b with a circular arc A. The centre M_A of the arc A and the arc radius r_A are selected such, that r_A is maximised under the boundary conditions illustrated in Fig. 5-9:

- The arc radius r_A must be greater or equal to a minimum radius $r_{A,min}$, which is defined as the vehicle's coordinated turn radius at cruise airspeed $v_{A,c}$ and the maximum passenger load factor $n_{L,pax}$

$$r_{A,min} = \frac{v_{A,c}^2}{g\tan(\Phi)}, \text{ using } \Phi = \arccos\left(\frac{1}{n_{L,pax}}\right). \tag{5-8}$$

- The minimum obstacle clearance $d_{xy,min}$ must not be violated.

- The arc A may only span a length l_A on legs a and b of lengths l_a and l_b, which must not exceed half the length of the shorter path leg

$$l_A \overset{!}{\leq} \frac{min(l_a, l_b)}{2}, \text{ with } l_A = \frac{r_A}{\tan\left(\frac{\alpha_{ab}}{2}\right)} = r_A \cot\left(\frac{\alpha_{ab}}{2}\right). \tag{5-9}$$

If the above boundary conditions are respected, every two adjacent path legs can be connected independently from any previous or following leg, while both the incoming and outgoing arc on each leg are tangent to the leg's initial velocity vector. Hence, a sequence of waypoints $\mathcal{P}$ is converted into a sequence of arc and straight line segments $\mathcal{T}$. The trajectory's velocity profile is computed from prior knowledge of the vehicle's performance envelope and the maximum admissible load factor $n_{L,pax}$ in each turn in a successive processing step. This method accepts discontinuities in the path curvature and does not claim optimality of any kind. It may be replaced with a more sophisticated trajectory generation method providing continuity of the path in higher derivatives (see e.g. [3] and [51]), when a product development is intended or high tracking performance is required.

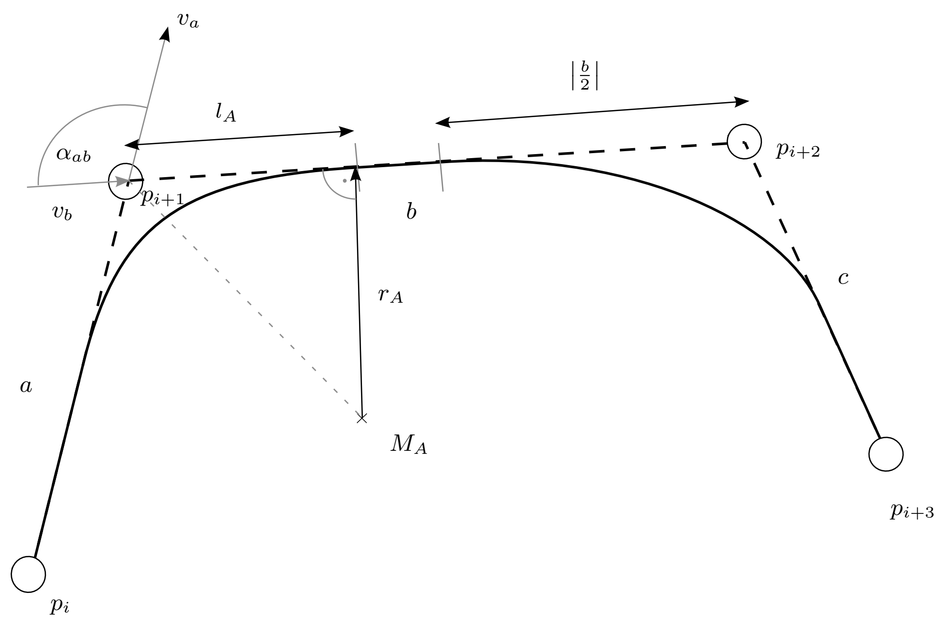


Figure 5-9: Adjacent path legs a, b and c are connected using circular arcs A of radius r_A.

5.4.2.2 A Near-Vertiport Planner for Take-Off and Landing Manoeuvres

The near-vertiport planner (see Fig. 5-2) runs on the planning surface $\mathcal{S}$'s approach funnel and defines a final approach and departure zone with radius r_{ls} around the vertiport. Inside r_{ls}, the planning strategy is adjusted towards take-off and landing procedures, which may differ from the global planning objective. When the global planner intersects the final approach and departure zone around the vertiport, a local Dubins path planner is used to align the vehicle with the wind direction during take-off and landing. The radius r_{ls} therefore depends on the vehicle capabilities. When different vehicles operate on a vertiport it should be selected to consider the least manoeuvrable vehicle. The near-vertiport planner determines the largest obstacle-free element in the final approach and departure zone, described by the angle β_{free} (see left-hand side of Fig. 5-10). Then, the planner is restricted to operate within β_{free} to enforce an approach from the direction with the largest margin on manoeuvring space and to increase robustness against potentially harmful wind conditions and gusts. For details on the implementation of a Dubins path planner, see Chapter 2.4.2.

If the wind direction is not contained inside the free manoeuvring angle section β_{free}, the planner chooses the approach or departure direction closest to the wind direction, while still within the obstacle-free segment. Similar to planning surfaces, the free manoeuvring angle interval is computed under consideration of a minimum vertical distance to ground obstacles $d_{z,min}$ and following the ICAO definition for maximum obstacle heights around an aerodrome and so-called approach and take-off climb surfaces (see [32]).

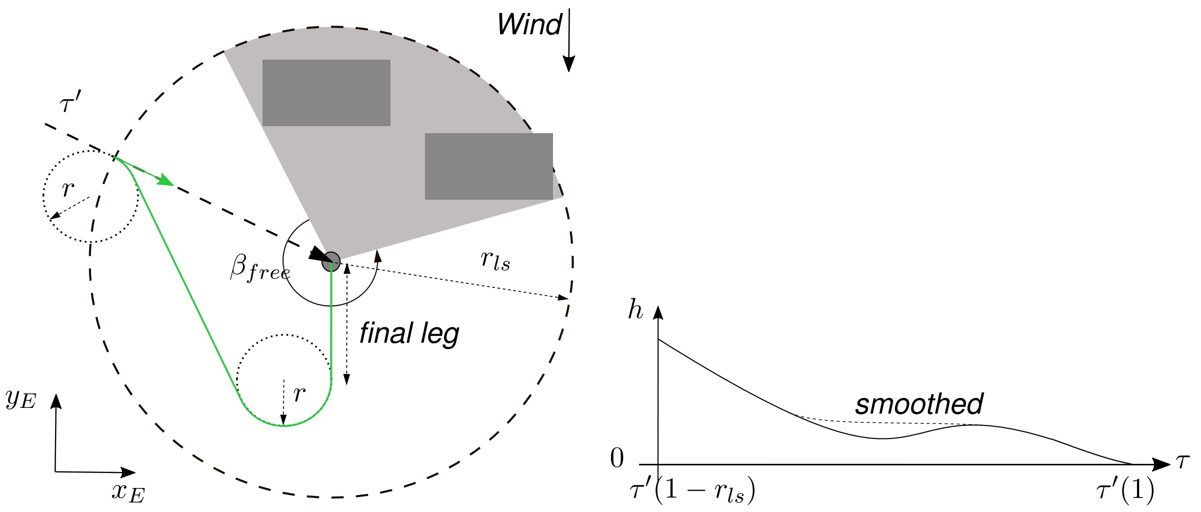


Figure 5-10: The near-vertiport planner operates on the obstacle-free angular interval β_{free} and within a radius r_{ls} around the vertiport to align a vehicle with the wind direction during take-off and landing. Unintended climb-descend patterns near the vertiport are avoided. Figure is based on [47].

Both take-off and landing flight paths are planned from the vertiport towards the intersection of the global trajectory τ' with a circle of radius r_{ls} around the vertiport. Approach and departure trajectories are distinguished by the direction of flight, where the vehicle intersects the radius r_{ls}. As a consequence of the flight surface definition, take-off and landing trajectories are planned on funnel-shaped surfaces, which causes the altitude to correlate with the Euclidean distance to the vertiport. This way, the underlying funnel shape introduces unintended climb-descend patterns when the Euclidean distance to the vertiport reaches a local minumum. The planner can force the flight path to maintain a monotone vertical profile within r_{ls} around the vertiport, if the vertical distance to any no-fly zone or restricted airspace above the vertiport is large enough. This is typically the case for aerodromes and vertiports. A flight path planned from the vertiport towards the global solution must not have negative gradients on the altitude profile (see right-hand side of Fig. 5-10). Appropriate boundary conditions on the vehicle heading Ψ and velocity vector $\vec{v}$ ensure continuity of the global trajectory τ' at the interface between the global and near-vertiport planner.

5.5 A Road Lane Concept for Bi-Directional Operations in Segregated Corridors

The rule-based planning approach for the operation of a single vehicle in a network of vertiports can be extended to account for multiple vehicles, which operate between the same two vertiports repeatedly and in opposite directions at the same time. Early use cases of such operations may be a cargo UAV providing supplies to a remote location or commuter services between a downtown area and an airport. Early U-Space implementations U1 and U2 (for reference see [73] and [46]), where limited inter-vehicle communication and tactical deconfliction methods are available, demand segregated air corridors for UAV missions within U-Space airspace. Based on this assumption, it is considered favourable to contain the operation of multiple vehicles between the same two vertiports within a single segregated corridor to reduce efforts for path planning and traffic management. In reference to contribution C-3, a deconfliction method is proposed, which extends the rule-based planning approach towards scenarios, in which multiple vehicles operate in opposite directions on a single route. This section is based on and a partial excerpt from the original paper [47].

A feasible and cost-optimal flight path τ between two vertiports VP_1 and VP_2 is computed on the planning surface $\mathcal{S}$. Assuming that the operational envelope is defined such, that the vehicle only operates in wind conditions, where the wind speed $v_w << v_{A,max}$, the global roadmap solution of this flight path is agnostic to the direction of flight, except for the impact of wind on the energy consumption. Hence, flight paths in opposite directions can be derived from the same initial planning solution, similar to road lanes. Each lane of flight path τ is shifted horizontally on $\mathcal{S}$ by an offset $\frac{d_{xy,rl}}{2}$ to achieve a safe road lane separation distance $d_{xy,rl}$ between opposite lanes. The parameter $d_{xy,rl}$ is specific to the vehicle, the mission and the environment, in which the operation takes place. It can be defined as the system's total system error (TSE) multiplied by a safety factor k

$$d_{xy,rl} = k \cdot TSE = k(FTE + NSE), \tag{5-10}$$

where FTE and NSE are the system's flight technical error and navigation system error, respectively. The path definition error is neglected.

In addition to the horizontal separation of lanes, the method exploits the 2.5D assumption introduced in Section 5.4.1. This assumption implies that a flight path is safe and collision-free on and above the flight surface $\mathcal{S}^*$, on which the original path was planned. A flight path's lanes can therefore be distinguished by the global direction of flight and be assigned different flight altitudes, where the lower lane must be allocated to the altitude flight profile of $\mathcal{S}^*$ and the vertical separation $d_{z,rl}$ is defined following the notion of (5-10). Flight altitudes are assigned to the respective road lane based on the global heading and following an abstraction of the semicircular rule. The manoeuvre to establish the altitude offset $d_{z,rl}$ is subject to two contradicting requirements. On the one hand side, it shall maximise the distance along the flight corridor, on which separation of lanes is achieved. On the other hand, the operational path angle interval shall not be exceeded to avoid a significant reduction of the vehicle's energy reserve due to excessive climb manoeuvres. A helix manoeuvre at a path angle $\gamma \leq \gamma_{max}$ is proposed, which spans the vertical separation in immediate proximity to the vertiport. This comes with the advantage, that the manoeuvre can be maintained within the near-vertiport planner's extension and does not require to rerun the global planning method. Figure

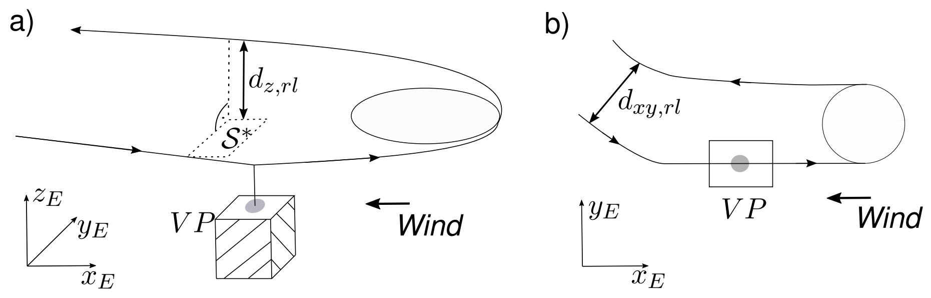


Figure 5-11: A concept for multi-vehicle operations in segregated corridors: Vertical separation $d_{z,rl}$ between traffic in opposite directions is achieved using a helix manoeuvre in a). A top view of the same situation is provided in b). From [47].

5-11 illustrates the definition and separation of road lanes in the horizontal and vertical dimension and shows that the near-vertiport planner maintains the alignment with the wind vector during take-off and landing also with the multi-vehicle operation.

While the introduction of a vertical separation is coherent with the rule-based planning approach, due to the 2.5D assumption, the concept cannot guarantee that road lanes, which are shifted horizontally from the rule-compliant path, are still rule-compliant as the distance to obstacles changes. To account for the change in obstacle clearance from the lane generation process, the lane separation $d_{xy,rl}$ must be considered in the roadmap pruning function (compare Alg. 4, line 7). The rule for obstacle clearance must be adjusted, such that the minimal obstacle for bi-directional corridor operations

$$d^*_{xy,min} = d_{xy,min} + \frac{d_{xy,rl}}{2} \tag{5-11}$$

is enforced on every graph edge. On a side note, it should be emphasised that the road lane concept with vertical and horizontal separation is designed for automated UAV operations. Piloted operations may prefer a road lane concept, in which the pilot has an unobstructed view of the on-coming traffic. They may therefore give up the vertical separation of lanes and rely on the pilot's ability to avoid other corridor traffic at the same flight level.

5.6 Contingency Planning with Databases of Trajectories

While the rule-based planning approach provides rule-compliant solutions to a planning task between two vertiports, cases, in which the occurrence of a contingency event requires a change of the guidance strategy, demand an extension of the planning method. Instead of computing a mitigation strategy at flight time, a contingency planning method is developed according to contribution C-4, which shifts computationally expensive planning steps into an offline processing phase. This reduces the complexity of the online planning task. The method computes a large database of trajectories, each of which represents a safe and feasible solution to the initial problem, such that the online planning task collapses to a selection task, which picks from a set of pre-computed feasible solutions. Deliberately accepting a reduced flexibility when compared to stand-alone online planning algorithms, a trajectory database can be verified and approved prior to take-off and provides a transparent discretization of the solution space. Should a trajectory contain undesired states, it may even be deleted from the database or blocked temporarily.

The proposed method structures trajectories in search trees, on which each branch represents an option for action and a logical contingency management module can select the most cost-efficient option according to a cost function C at each point in time. A proposal of a cost function C is given in (5-6). The database distinguishes between a tree of trajectories connecting take-off and destination vertiports and trajectories for continued safe flight and landing (for reference, see Section 3.1), which enable the vehicle to divert to an alternate vertiport (compare Fig. 3-4), should a contingency event require a change of the initial flight plan. This section is based on and a partial excerpt from the original paper [48], in which this approach was published initially. It covers the contingency planner and trajectory database listed in Fig. 5-2.

5.6.1 Computing a Tree of Trajectories to the Destination Vertiport

Based on the maximum clearance roadmap $\mathcal{G}$ generated on the planning surface $\mathcal{S}$ in Section 5.4.2.1, alternative flight paths between a take-off and destination configuration p_{start} and p_{goal} can be computed (compare contribution C-4a). This roadmap is already pre-processed to comply with the defined rule set, such that any alternative solution on this roadmap will be equally rule-compliant. Alternative solutions to the optimal flight path τ_{opt} obtained from Section 5.4.2.1 are generated by running a graph search algorithm on the roadmap $\mathcal{G}$ repeatedly, where the graph's cost structure is updated between every two iterations. To divert the path search from previous solutions, two types of penalty edge weights are introduced ([48]):

- A penalty weight $w_{p,i}$ is assigned to edges e_i that are a member of at least one existing path and
- a penalty weight $w_{rj,i}$ applied to edges e_i joining an existing path, i.e. ending in a vertex that is a member of at least one existing path.

In addition to the described penalty weights, the following parameters are defined to describe the trajectory tree computation method:

- A flight path τ consisting of a set of waypoints $\mathcal{P} = \{p_i | i \in (0, 1, 2, ..., n)\}$.
- Fork points $fp_i \in \mathcal{P}$ on τ, in which alternative paths start.

- A minimum distance $d_{fp,min} > 0$ between fork points fp.
- A branch factor $N \in \mathbb{N}^{+}$, defining the desired number of alternative paths in each fork point.
- A path length constraint factor $k > 1$, which limits the maximum path length L of an alternative path τ to $L_{\tau} \leq kL_{\tau_{opt}}$.

The algorithm to compute a tree of alternative paths to the destination vertiport on $\mathcal{G}$ is provided in Algorithm 5 and illustrated in Fig. 5-12. Following the initial search on $\mathcal{G}$ to obtain the optimal path τ_{opt}, the fork point fp is initialised at the initial vertex p_{start}. Penalty weights $w_{p,i}$ and $w_{rj,i}$ are applied to edges e_i on and leading to τ_{opt}, such that a repeated graph search on $\mathcal{G}$ in the function *replan()* yields an alternative path τ_{alt}. If the path τ_{alt} passes the path length constraint implemented in *path_valid()*,

$$l_{f,i} + l_{P\tau_{alt},i} \overset{!}{\leq} k \cdot L_{\tau_{opt}}, \tag{5-12}$$

where $l_{f,i}$ is the path length from the initial position p_{start} to the fork point fp_i and $l_{P\tau_{alt},i}$ the path length from the fork point fp_i to the goal p_{goal}, the path is added to the trajectory database DB. For each path to the destination vertiport, contingency paths leading to alternate vertiports are computed in *plan_contingencies()*. An implementation of this function is proposed in Section 5.6.2. Paths starting in p_{start} are computed and added to the trajectory database DB until either the number of path options in the current fork point N_{curr} is equal to the branch factor N or a path violates (5-12). In this case, the fork point is updated to the next waypoint $p_1 \in \mathcal{P}(\tau_{opt})$. To avoid clustering of alternative paths in obstacle-dense regions with narrowly spaced waypoints, the function *branch_dist_ok*() checks that a new fork point lies at a distance greater or equal $d_{fp,min}$ from the previous fork point. No alternative paths are computed from waypoint closer than $d_{fp,min}$ to the last fork. If passed, the computation of alternative paths repeats until a termination criterion is met, otherwise the next fork point candidate is checked for compliance. The process terminates if a tree of the required branch factor N is obtained, all path options on the graph $\mathcal{G}$ are exhausted or if the tree reaches a pre-determined size.

When the fork point update step reached the end of a path τ_i, the next path in the database DB, τ_{i+1}, is selected. For each path except the initial optimal path τ_{opt}, the fork point is initialised to the second waypoint $p_1 \in \mathcal{P}_{\tau_{i+1}}$, since each alternative path shares the first waypoint with its parent (compare Fig. 5-12). The algorithm terminates when all paths in DB have been visited, i.e. all paths obtained in a repeated graph search are either subsets of already identified paths or violate the length constraint (5-12). In this case the database DB containing a tree of rule-compliant trajectories between the take-off configuration p_{start} and destination p_{goal} is returned.

To enable the trajectory generation method presented in Section 5.4.2.1 to yield smooth transitions from one trajectory to another, boundary conditions at the start of the child

Algorithm 5: A pseudo code representation of alternative path planning update steps.

```
$\mathcal{G}$.init(), $DB$.init($\tau_{opt}$), $p_{goal}$.init(), $N$.init()
$\tau \leftarrow \tau_{opt}$, previous_fork $\leftarrow \emptyset$, current_fork $\leftarrow p_0(\tau_{opt})$
$N_{curr}, i, j$, DB_size $\leftarrow 0$
penalise_edges($\mathcal{G}$, $\tau$, $w_p$, $w_{rj}$)
while true do
    if branch_distance_ok($d_{fp,min}$, current_fork, previous_fork) then
        if $N_{curr} < N$ then
            $\tau_{alt} \leftarrow$ replan($\mathcal{G}$, current_fork, $p_{goal}$)
            if path_valid($\tau_{alt}$) then
                $\mathcal{T}_{CSFL} \leftarrow$ plan_contingencies($\tau_{alt}$)
                $DB$.add_paths($\tau_{alt}, \mathcal{T}_{CSFL}$)
                DB_size $+= 1$
                penalise_edges($\mathcal{G}$, $\tau_{alt}$, $w_p$, $w_{rj}$)
                $N_{curr} += 1$
                continue
            else
                previous_fork = current_fork
        else
            previous_fork = current_fork
    $i += 1$
    current_fork $\leftarrow p_i(\tau)$
    $N_{curr} = 0$
    if current_fork == $p_{goal}$ then
        $j += 1$
        if $j <$ DB_size then
            $\tau \leftarrow DB$.get_path($j$)
            $i = 0$
            previous_fork $\leftarrow \emptyset$
            current_fork $\leftarrow p_0(\tau)$
        else
            break
return $DB$
```

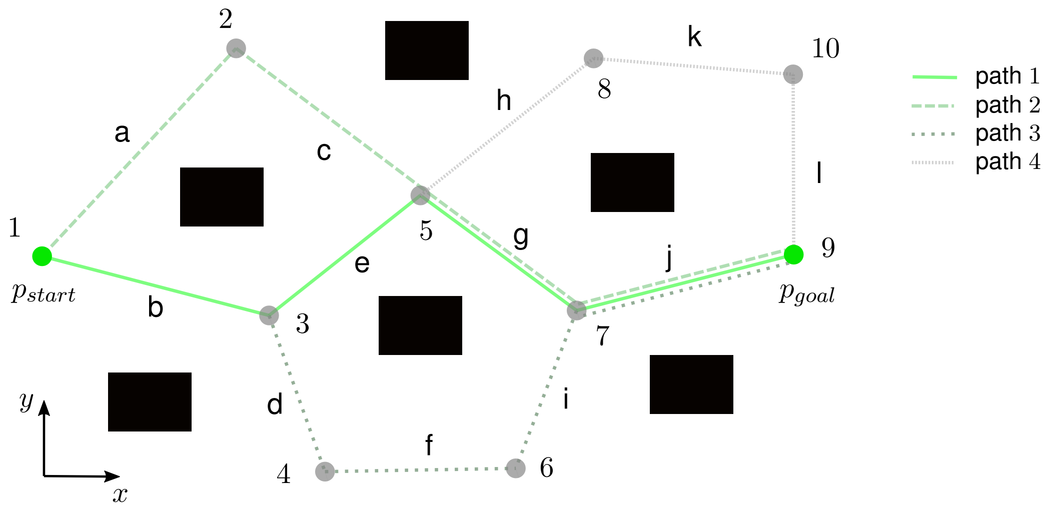


Figure 5-12: A path tree with branch factor $N = 2$ is generated on a graph between nodes p_{start} and p_{goal}. Every iteration, penalty weights $w_{p,i}$ and $w_{rj,i}$ are applied to edges e_i that are members of or joining an existing path, respectively, to identify new alternative solutions. From [48].

trajectory must be matched to the parent trajectory. This can be achieved through consideration of an additional support waypoint on the parent trajectory to ensure smooth transitions between the parent and child trajectories. Based on the defined limits for placement of an arc to connect two adjacent path legs in Fig. 5-9, the transition between a parent trajectory τ_i' and child trajectory τ_j', which forks in the parent trajectory's waypoint $p_{i,k}$, will be continuous in position and the velocity profile if a support waypoint

$$p_{j,-\frac{1}{2}} = p_{i,k-1} + \frac{p_{i,k} - p_{i,k-1}}{2} \tag{5-13}$$

on the parent path's leg $(p_{i,k} - p_{i,k-1})$ is added to the child path. The placement of the support waypoint $p_{j,-\frac{1}{2}}$ and continuous transitions between parent and child trajectories τ_i' and τ_j' are illustrated in Fig 5-13.

5.6.2 Contingency Paths for Continued Safe Flight and Landing

This section proposes a database-centric approach to the *plan_contingencies()* function in line 10 of Algorithm 5. The list of nominal trajectories leading to the destination vertiport is extended with equally four-dimensional contingency trajectories. These flight paths provide pre-planned diversion trajectories to all nominal and alternate vertiports, which are available and within a radius R_{CSFL} around the current position. A method according to contribution C-4a and C-4b is proposed, which provides a flight path to an alternate vertiport at constant time intervals Δt_C. The presented approach was initially developed to compute guidance laws, which guide an aircraft from any location in the flight area to a safe landing site and without the need to pre-compute and save explicit trajectories. To achieve compliance with the SC-VTOL Enhanced

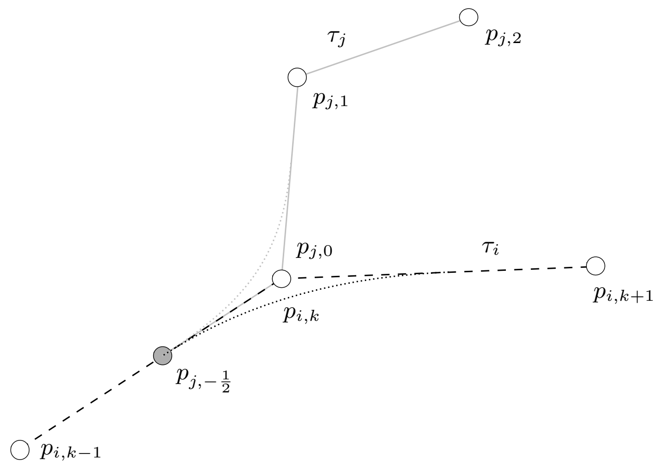


Figure 5-13: The placement of a support waypoint $p_{j,-\frac{1}{2}}$ on the parent trajectory τ_i' to obtain a smooth transition onto the child trajectory τ_j'.

category's CSFL requirements for early UAM operations, contingency trajectories are planned on the obtained navigation functions explicitly and added to the trajectory database. Should the requirement for pre-planned trajectories be lifted in future UAM operations, a vehicle may navigate directly on the navigation function.

5.6.2.1 Discretisation of the Contingency Planning Space

In order to plan safe contingency trajectories across a mission's altitude profile, different altitudes need to be considered for navigation functions to alternate vertiports. Hence, the discretization of the vertical airspace into flight levels (compare Section 5.4, rule 1) is also applied to the planning surfaces of contingency trajectories to be coherent with the discrete flight levels of the rule-based planning approach and provide a safe vertical separation between aircraft. Flight levels are defined between the minimum flight altitude h_{min} and service ceiling h_{max} at separation Δh_{FL}, where Δh_{FL} is defined such that it reflects relevant properties of air traffic participants, among which are the navigation performance and rotor downwash. When operating on different flight levels, two vehicles' horizontal flight paths can cross, without affecting the respective other. Flight levels can therefore be considered a valid extension of the road lane concept of section 5.5 for multiple vehicles on a single route, towards the operation of many vehicles on many routes.

For the purpose of contingency planning with databases, flight surfaces $\mathcal{S}^*_{m,n}$ are generated for each vertiport m and flight level FL_n, according to (5-2) and (5-5). These flight surfaces expand horizontally into the configuration space at the flight level's altitude h_{FL_n} and for each vertiport share an approach funnel F_m. Obstacle maps of higher flight levels are typically populated more sparsely than obstacle maps of lower flight levels. Flight level surfaces and approach funnels can be computed indepen-

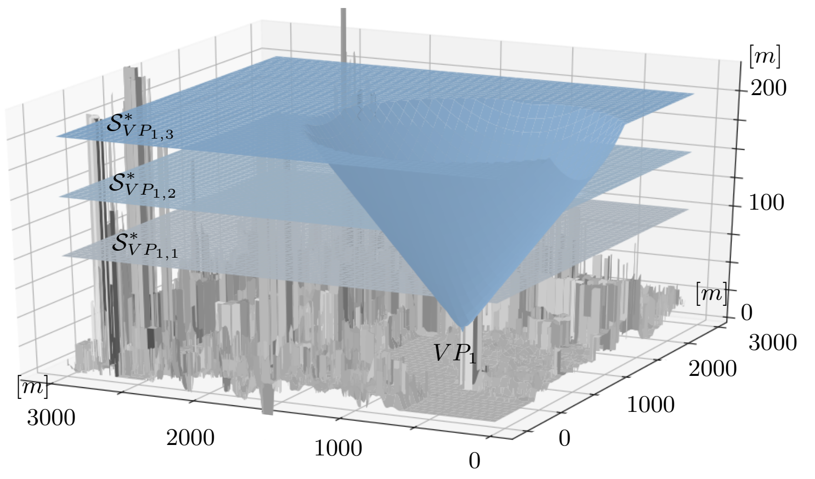

$\mathcal{S}^*_{VP_{1,3}}$
$\mathcal{S}^*_{VP_{1,2}}$
$\mathcal{S}^*_{VP_{1,1}}$
VP_1
$[m]$
200
100
0
$[m]$
3000
2000
1000
0
$[m]$
3000
2000
1000
0

Figure 5-14: An urban obstacle environment with contingency planning surfaces $\mathcal{S}^*_{m,n}$, $m = 1$ vertiports and $n = 3$ flight levels. The approach funnels of each planning surface share a common path angle to descend into vertiport VP_1. From [48].

dently for each flight level and vertiport, respectively, and then intersected to obtain a vertiport-specific flight surface $\mathcal{S}^*_{m,n}$. The projection of $\mathcal{S}^*_{m,n}$ into the horizontal plane (see (5-5)), the planning surface $\mathcal{S}_{m,n}$, is saved in memory for each vertiport and serves as the baseline to compute navigation functions on each surface. Contingency flight surfaces $\mathcal{S}^*_{VP,n}$ (with $n = 1...3$) with a representative obstacle environment are depicted in Fig. 5-14. Each surface connects to the vertiport VP_1 from a different flight level. The vertiport's approach funnel F_{VP_1} ensures, that a vehicle approaches the vertiport VP_1 at the same path angle γ, regardless of the flight level, from which it descends.

In a consecutive computation, optimal navigation functions $\mathcal{N}_{\mathcal{S}_{m,n}}$ (for reference see criterion (2-3) and Fig. 2-5) towards the vertiport are generated on each planning surface $\mathcal{S}_{m,n}$, using the wavefront expansion approach described in Algorithm 3. This approach ensures that the vehicle cannot get trapped in local minima, despite the implementation of a potential function and given that the vertiport can be reached from the initial state. The wavefront expansion is agnostic to the selected cost function C. This implementation uses a simple distance cost metric

$$C(\vec{x}_1, \vec{x}_2) = |\vec{x}_2 - \vec{x}_1|, \tag{5-14}$$

with two states in the free configuration space $\vec{x}_1$ and $\vec{x}_2$. However, the metric can be extended to more complex cost metrics. The computation of the navigation function $\mathcal{N}_{\mathcal{S}_{m,n}}$ is deterministic and for each planning surface the navigation function's compliance with a rule set can be verified prior to flight. A pre-planned trajectory on such a navigation function will display equally compliant behaviour. The obtained navigation function is specific to a vertiport, but not to a mission. This means that inside a network of vertiports, contingency navigation functions are valid for missions between any two vertiports in the network until the obstacle map, on which it was computed changes. Each planning surface $\mathcal{S}_{m,n}$ is saved as a structure with properties

- vertiport ID,
- flight level FL,
- altitude profile H,
- obstacle map M and
- navigation function $\mathcal{N}$.

Beyond serving as a basis for the generation of a database of contingency trajectories, precomputed navigation functions add an additional safety layer as they enable direct online vehicle guidance, when paired with e.g a steepest descend gradient method.

5.6.2.2 Computing Contingency Trajectories for Continued Safe Flight and Landing

Contingency trajectories leading from a discrete state on a trajectory τ' in the tree of trajectories T to the destination vertiport (see section 5.6.1) to n_{al} available alternate vertiports are computed in the offline planning phase and added to the trajectory database. A discrete state, in which a trajectory branches off another trajectory, is referred to as a fork point fp. Fork points are sampled at a contingency planning time interval Δt_C along each four-dimensional trajectory $\tau' \in T$.

In each fork point fp on $\tau' \in T$ at altitude h_{fp} and for each vertiport in the set of available alternate vertiports $v \in VP_{al}$, the *plan_contingencies()* function (see Algorithm 5, line 10) selects the planning surface $\mathcal{S}$ corresponding to the closest lower flight level. If there does not exist a path to the alternate vertiport v on this level, the vertiport is discarded. A contingency trajectory τ'_C is computed using a steepest descend on the navigation function $\mathcal{N}_\mathcal{S}$ and added to the trajectory tree T. This procedure is repeated for each $v \in VP_{al}$ within the contingency landing radius R_{CSFL} around the fork point fp and for each fp on τ'. Then the trajectory is updated to the next trajectory τ'_{i+1} in the tree. If the new trajectory τ'_{i+1} is a contingency trajectory, new contingency trajectories $\tau'_{C,i}$ are planned to all available alternate vertiports $v \in VP_{al}$ except destination sites of parent trajectories. This implies, that if a vertiport becomes unavailable during a mission, it will no longer be considered as an option in the resolution of consecutive contingency events. Hence, the algorithm produces a finite number of contingency trajectories in T and reduces the number of options at each fork point with increasing distance from the tree root (i.e. the take-off vertiport). The expansion of a tree branch terminates when the destination can be reached in shorter time than the branching interval Δt_C, there are no available alternate sites within R_{CSFL} or there are no more fork points on the current trajectory τ' that are above the lowest contingency flight level.

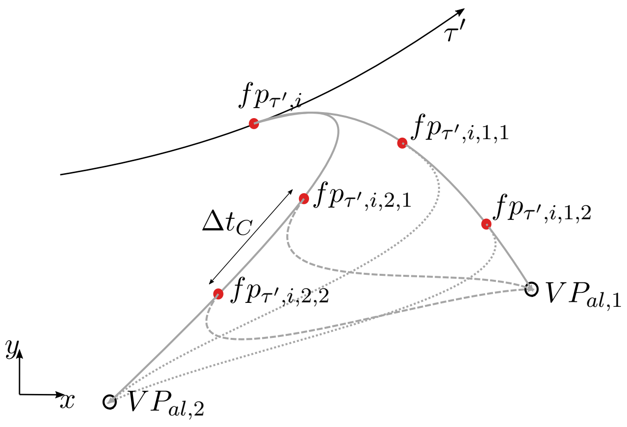


Figure 5-15: A contingency path branch is expanded towards alternate vertiports $VP_{al,1}$ and $VP_{al,2}$ from a fork point $fp_{\tau',i}$ on the trajectory τ'. From [48].

Figure 5-15 illustrates the expansion of a contingency branch to alternate vertiports from a fork point fp on the nominal trajectory τ' and with two available alternate landing sites $VP_{al,1}$ and $VP_{al,2}$. The algorithmic method for contingency planning towards alternate vertiports is described in Algorithm 6 using a recursive implementation of *plan_contingencies()*.

Algorithm 6: A recursive implementation of the contingency trajectory expansion towards alternate landing sites.

```
1  Function plan_contingencies($\tau'$, $\Delta t_C$, initial_valid_VPs, path_list):
2      for $fp$ on $\tau'$ do
3          valid_VPs = initial_valid_VPs
4          for $v$ in valid_VPs do
5              $\mathcal{S} \leftarrow$ get_nearest_surface($fp, v$)
6              if $\mathcal{S}$ then
7                  $\tau'_C \leftarrow$ plan_path($fp, \mathcal{S}$)
8                  path_list.add($\tau'_C$)
9                  new_valid_VPs $=$ valid_VPs.remove($v$)
10                 if new_VP_list not empty then
11                     path_list $\leftarrow$ plan_contingencies($\tau'_C$, $\Delta t_C$, new_valid_VPs, path_list)
12     return path_list
```

The proposed method plans contingency trajectories on a navigation function $\mathcal{N}$, which is unaware of the kinematic vehicle state on the parent trajectory in each fork point. Hence, a local planning step is required to ensure smooth and flyable transitions between the parent and child trajectory. A Dubins path planner can be employed to connect the vehicle state in the fork point fp on τ' with a state on the contingency trajectory τ'_C. According to Fig. 5-16, the turn radius of the Dubins primitives is selected

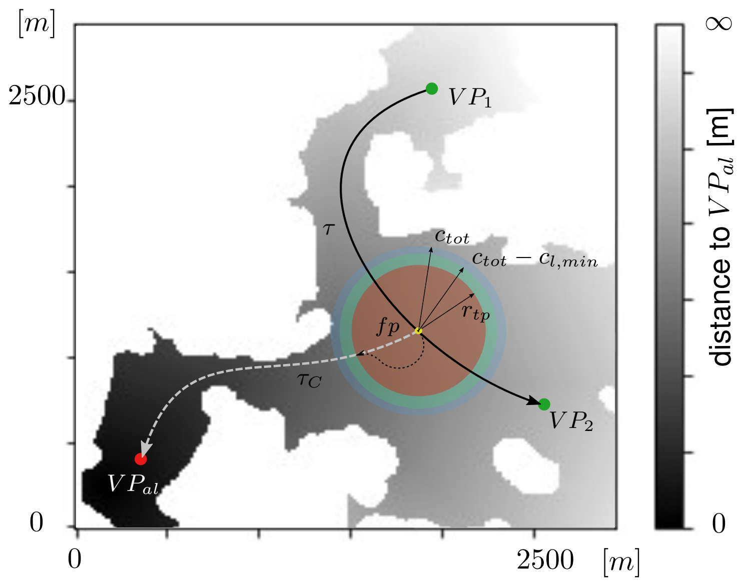


Figure 5-16: Continuous transitions between the trajectory τ' and the contingency trajectory τ'_C leaving τ' in the point fp are planned using a local Dubins path planner. Figure is based on [48].

such, that the vehicle can plan a transition inside a safe transition planning radius r_{tp} around fp, while the minimum lateral obstacle clearance $c_{l,min}$ is maintained

$$r_{tp} \leq c_{tot}(fp) - c_{l,min}. \tag{5-15}$$

Transitions are planned towards the intersection of the new contingency trajectory τ'_C with the circle of radius r_{tp} around fp. Contingency trajectories, to which a safe transition cannot be found within r_{tp}, are discarded. Similar to the argumentation of section 5.4.2.1, discontinuities in the path curvature are accepted in this conceptual implementation and the local planning method may be replaced for better tracking performance in future work.

5.6.2.3 Local Execution of Motion Primitives

When the operation of UAVs is extended into non-segregated air space, it is expected that a considerable portion of contingency events will be related to air traffic or occur as a consequence of vertiports being temporarily unavailable, due to e.g. vehicles moving on ground. Traffic-related events may therefore be resolved by postponing the point in time, at which a vehicle occupies a certain point in space, or by changing its flight altitude to avoid other vehicles. The horizontal trajectory profile can be maintained in both cases. The database of nominal and contingency trajectories is therefore extended with motion primitives (compare contribution C-4c), which allow the vehicle to change flight levels without leaving the horizontal trajectory profile and a holding pattern to enable the vehicle to loiter at constant altitude. Respecting the notion of de-

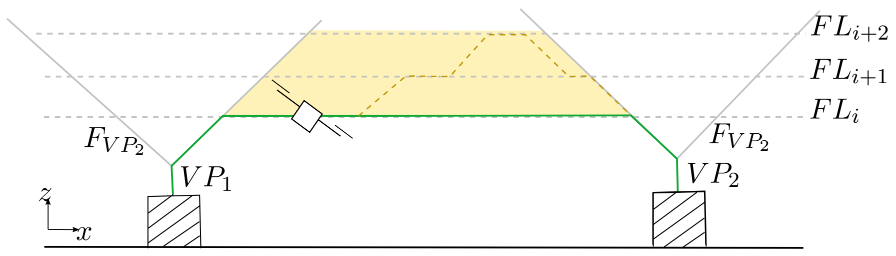


Figure 5-17: The manoeuvre space (yellow) between discrete flight levels. During the cruise phase, a vehicle can move freely between the initial flight level FL_i and higher flight levels (dashed orange). From [48].

coupled planning in the horizontal and vertical dimension (compare contribution C-2), the shape of a single primitive is restricted to either plane. Holding patterns are considered more energy efficient than hovering in cases where a longer temporal delay is required, that cannot be achieved by slowing down without entering highly inefficient flight states. Motion primitives for both loitering and altitude changes are pre-defined in velocities and geometric dimensions.

As a result to the 2.5D property of the rule-based planning method, the trajectory τ_i' is collision-free on and above its planning surface $\mathcal{S}_i$. Hence, a vehicle can change its flight altitude from a flight level FL_i, to a higher flight level FL_{i+1} and back. The horizontal trajectory profile can be maintained as long as the altitude does not fall below the trajectory's initial planning altitude. The vehicle has to return to the altitude of the trajectory's planning surface, when entering the destination vertiport's approach funnel, since it would otherwise violate the maximum descend angle. Figure 5-17 explains the vertical manoeuvre space for a mission between two vertiports VP_1 and VP_2 in an airspace with three flight levels.

In order to execute holding patterns safely along a trajectory, holding pattern entry point candidates are distributed along the trajectory at the holding pattern interval Δt_{hp}. In each holding pattern entry point candidate, the lateral obstacle clearance c_l between the trajectory and the closest obstacle is evaluated and saved. Equation (5-16) states that a holding pattern of radius r_{hp} can be executed, if the entry point candidate's lateral clearance $c_l(p_i)$ is greater or equal to the holding pattern's diameter $2r_{hp}$ augmented with a minimum distance to obstacles $c_{l,min}$

$$c_l(p_i) \overset{!}{\geq} 2r_{hp} + c_{l,min}. \tag{5-16}$$

This criterion prohibits the execution of holding patterns if the obstacle clearance on either side of the trajectory is small and regardless of large obstacle-free regions on the other. The obstacle clearance of entry point candidates along a trajectory τ', at which the execution of a holding pattern of radius r_{hp} is allowed, is displayed in green colour in Fig. 5-18. For these candidates (5-16) holds. Obstacle clearances of entry point candidates with insufficient obstacle clearance are depicted in red. In the illustrated

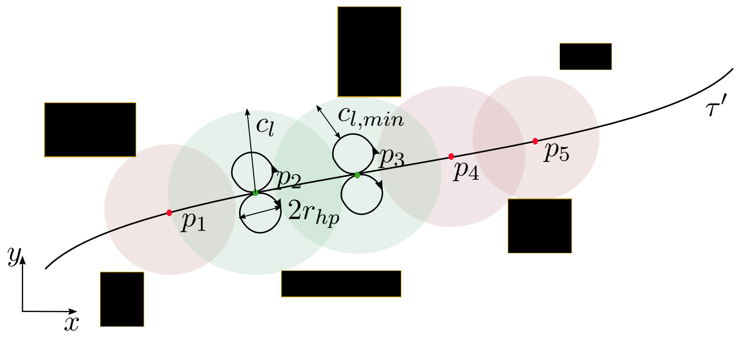


Figure 5-18: An illustration of the holding pattern execution criterion. Holding patterns with radius r_{hp} are allowed in the position p_i if the obstacle clearance $c_l(p_i)$ satisfies (5-16). From [48].

case, (5-16) holds in candidates p_2 and p_3. In candidates p_1, p_4 and p_5, the criterion is not fulfilled, although p_4 neighbours a large obstacle-free region in the positive direction of the y-axis.

The implementation of a trajectory database of nominal and contingency trajectories, motion primitives and navigation functions allows to respond to an unforeseen event with the action, which has the lowest impact on the mission objective. For dense vehicle operations it may also be beneficial to define a contingency action scheme, which prioritises actions. Within the scope of UAM and under the assumption of mission-specific flight corridors, a change of destination and the horizontal flight profile appear to be the least favourable option, when multiple vehicles operate in a network of vertiports. A prioritization scheme may therefore define to select a contingency mitigation action in the following order:

1. Adapt velocity
2. Change of flight level
3. Execution of holding pattern
4. Alternative trajectory to destination
5. Alternative trajectory to alternate vertiport
6. Immediate steepest descend on navigation function.

Different prioritization schemes may be defined according to the operational context and regulatory boundary conditions in different planning environments. Regardless of the prioritization of actions, the described contingency planning method provides N path options in each fork point on the tree of trajectories to the destination. Within

Table 5-1: The estimated database memory complexity with respect to different planning parameters.

Parameter	Sign	Memory Complexity
branch factor	N	$\mathcal{O}(n^m)$
flight time	t	$\mathcal{O}(n)$
path length restriction factor	k	$\mathcal{O}(n^3)$
number of alternate vertiports	$n_{VP,al}$	$\mathcal{O}(n^m)$
contingency branch interval	Δt_C	$\mathcal{O}(m^n)$

the longer of the holding pattern and contingency planning intervals Δt_{hp} and Δt_C, respectively, the method provides further n_p path options

$$n_p = n_{FL}(1 + n_{VP,al,av}), \tag{5-17}$$

with the number of available alternate vertiports in each fork point $n_{VP,al,av}$. Depending on the intended mission, the operational environment and the nature of expected contingency events, the number of flight levels n_{FL}, the number of available alternate vertiports $n_{VP,al,av}$, the branch factor N as well as the intervals Δt_{hp} and Δt_C can be selected to reflect the mission needs and support safe operation. The trajectory database maintains an entirely deterministic planning approach, which can be verified prior to take-off. The expected memory complexity of the proposed database is reflected in Tab. 5-1 with respect to different planning parameters.

5.7 Online Trajectory Transitions Within Convex Regions of Obstacle-Free Space

In the previous sections, a framework of planning methods is introduced, which solves the urban path planning problem with large databases of pre-computed trajectories and motion primitives. The online planning task can then be solved as trajectory selection problem. Data-base driven methods, however, need to find a balance between resolution and memory requirements, as the required memory increases exponentially with the tree resolution according to Tab. 5-1. When operating in environments, where sequences of multiple contingency events may require to provide alternative trajectories repeatedly and with potentially different mitigation strategies, the storage of sufficiently many trajectories in a trajectory database becomes difficult to handle or even intractable. With an increasing density of air traffic participants in the UAM en-

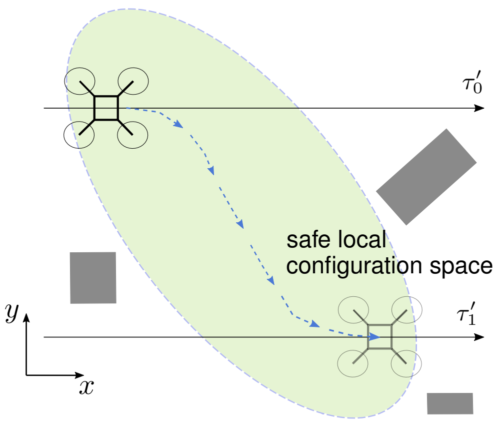


Figure 5-19: A transition between two pre-computed trajectories τ_0' and τ_1' is planned online and inside an obstacle-free region. From [49].

vironment, methods to enable in-flight replanning, will be required to at least a limited extent (compare to Fig. 5-2). The trajectory transition planning method was published in [49] originally and leverages pre-computed solutions to partial problems of the planning task, each of which can be verified or examined by a human prior to flight. The method computes transitions between two trajectories from a database inside equally pre-computed regions of obstacle-free space according to contribution C-5. This section is based on and a partial excerpt from the original paper [49] and covers the trajectory transition planner, manoeuvre generation and manoeuvre database listed in Fig. 5-2.

The task to plan a transition trajectory between two states on pre-computed trajectories can be divided into three individual problems. In a first step, convex obstacle-free regions are computed in the configuration space, inside which the absence of obstacles simplifies the planning task. Identified regions are linked with the trajectory database and for each region, intersecting trajectories are found. Second, an online planning method displaying deterministic properties and providing performance guarantees is required to contain the solution inside said regions. Finally, a decision module identifies transition candidates inside the current region at flight time. Feasible start and target positions on the current and targeted trajectory are identified and the transition process is triggered. A top view of the problem setting for two trajectories τ_0' and τ_1' intersecting the same region is illustrated in Fig. 5-19.

5.7.1 Computing Convex, Obstacle-Free Regions in the Configuration Space

An online planning function is considered safe for the task at hand, if it is guaranteed to contain any solution within an obstacle-free region. Further, there must exist predefined entry and exit trajectories to enter and leave a region. If the region is convex, this can be achieved even more easily than with concave regions as the shortest connecting path between any two points inside the region will also lie inside the region. The method to compute convex, obstacle-free regions in this work is based on the semidefinite programming approach described in [15] and the derived IRIS toolbox[1]. The goal of this computational step is the identification of save transition regions in the configuration space (see contribution C-5a). The IRIS toolbox is used to compute ellipsoids E of maximal area around an initialisation point P_s on a polygon representation of the two-dimensional planning surface's obstacle map. The ellipsoid E is described as an image of the unit sphere

$$E(C,d) = \{x = C\tilde{x} + d \mid \|\tilde{x}\| \leq 1\}, \tag{5-18}$$

with a square matrix C and translational vector d. The algorithm initialises the ellipsoid E_0 in the point P_s and iterates through the list of obstacles. Tangent hyperplanes are generated, which separate each obstacle from the ellipsoid. Within the polytope that is defined by the ellipsoid's neighbouring hyperplanes, a new maximal ellipsoid E_1 can be found. The process terminates, when the ellipsoid's area growth converges. For a detailed description of the algorithm the reader shall be referred to the original paper [15].

To find obstacle-free ellipsoids in the two-dimensional configuration space, which intersect with ideally many trajectories in the trajectory database, the point P_s is initialised at the take-off location $P_{0,i}$ of a nominal trajectory τ_i'. The resulting ellipse E_0 is saved and the initialisation point P_s updated to a point $P_{1,i}$ on τ_i' outside E_0, which lies at a distance d from the boundary of E_0. The method is described in Fig. 5-20. The process terminates if the n-th ellipse E_n contains the trajectory's destination vertiport, or the distance from the intersection of τ_i' and the ellipse E_n to the destination is less than d. The point P_s is reinitialised at the take-off location of the next trajectory τ_{i+1}' in the database. For any $P_{k,i+j}$ on a trajectory τ_{i+j}', a new region is only computed if the current start point $P_{k,i+j}$ does not lie inside any previous ellipse. By the time the computation of ellipses terminates, a state s on a trajectory τ_i' in the database is either inside at least one obstacle-free transition region or closer than d to either a transition region or the destination vertiport. The smaller the parameter d is selected, the more transition regions are generated and fewer trajectory sections lie outside any transition region. In the edge case $d = 0$, any state s on any trajectory τ_i' will be a member of at least one region. A post-processing step iterates through all transition regions and identifies intersecting trajectories. This is necessary, as also trajectories other than the one, on which the region was initialised may intersect with it. The unique ID of each intersecting trajectory as well as coordinates of the trajectory's entry and exit points are saved as a property of each transition region. Hence, the method produces a list of obstacle-free transition regions including intersecting trajectories and the intersection interval of each intersecting trajectory.

[1] https://github.com/rdeits/iris-distro

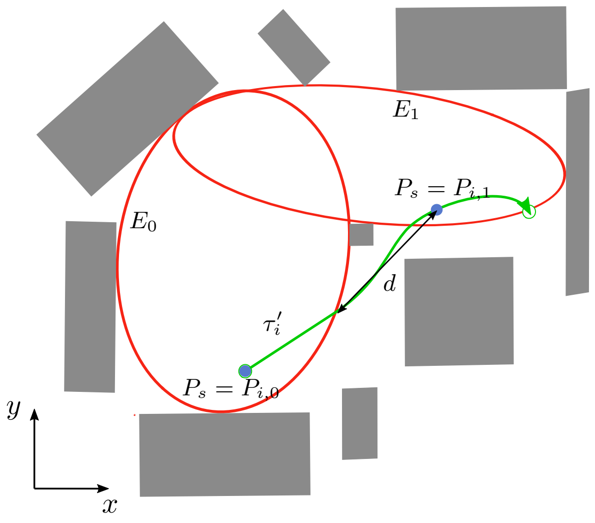


Figure 5-20: Obstacle-free regions along a trajectory τ_i' are obtained by inflating ellipses on the obstacle map. After each iteration i the method reinitialises on τ_i' at a distance d from the ellipse's boundary. The process terminates if the n-th ellipse E_n contains the destination vertiport or E_n is closer than d to the destination. From [49].

5.7.2 Deterministic Online Planning Inside Transition Regions

If a database of trajectories and obstacle-free transition regions is stored onboard an aircraft, this aircraft can run an obstacle-unaware planning method to transition between any two trajectories inside the same region, if the planning method contains the transition path inside said region. The second partial solution to the transition problem between trajectories, hence, addresses the selection of the online planning method. In the attempt to reduce the computational complexity of the online planning task, an obstacle-unaware and resolution complete planning method is proposed. The method shall further support the governing principle to compute partial solutions of the planning problem offline in order to simplify the online planning task in flight. Therefore, the manoeuvre automaton framework presented in [21] is used to generate transitions between pre-computed trajectories online and in flight. An illustration of a simple manoeuvre automaton with four trim states can be found in Fig. 2-6. An outline of the general method and more specifically the notion of optimal manoeuvre automata is given in Chapter 2.4.1.

A vehicle's performance envelope can be described by a motion library Q consisting of a set of trim states Q_T and manoeuvres Q_M, where the vehicle can switch trim states using manoeuvres. This motion library can also be validated to respect potential operational limitations. Trim states $q \in Q_T$ are defined on the two-dimensional planning surface $\mathcal{S}$ as a tuple of the kinematic velocity v_k and the load factor n_L

$$q = (v_k, n_L), \tag{5-19}$$

from which the turn radius r and azimuth rate $\dot{\Psi}$ can be derived as

$$r = \frac{{v_k}^2}{g \tan \Phi}, \text{ with } \Phi = \arccos(\frac{1}{n_L}) \tag{5-20}$$

and

$$\dot{\Psi} = \frac{v_k}{r}. \tag{5-21}$$

Under the assumption of coordinated flight and neglecting the effect of wind, the course rate $\dot{\chi}$ is equivalent to the azimuth rate $\dot{\Psi}$. Manoeuvres m are computed as a transition between two trim states q_0 and q_1 with the flat state transition

$$\delta s = [\Delta \dot{\Psi}_{0,1}, \Delta v_{k0,1}]^T \tag{5-22}$$

based on the kinodynamic steering method described in [7]. The required transition time is computed independently for each flat state under the consideration of boundary conditions $[\ddot{\Psi}_{max}, \dddot{\Psi}_{max}, a_{max}, j_{max}]^T$, with a_{max} and j_{max} the maximum translational acceleration and jerk, respectively. The state transition with the longest transition time defines the manoeuvre duration t_m, with the remaining output being adjusted such that a monotonic behaviour of the load factor n_L is achieved over the manoeuvre duration. Each manoeuvre and trim state are assigned unit cost.

Further, an optimal control policy (for reference see Fig. 2-9) can be approximated as an optimal cost function $\tilde{J}^*$ based on the value iteration described in (2-4) and (2-5). The function $\tilde{J}^*$ describes the optimal cost of reaching a target state s_t, when the action $a \in Q$ is executed in the current state

$$s = [q, h]^T, \text{ with } h = [\Psi, v_k, p_x, p_y] \tag{5-23}$$

and a trim state $q = (v_k, n_L)$, the heading Ψ, kinematic velocity v_k and position (p_x, p_y). The general notion of the value iteration ([6]) to quantify a state's value $\tilde{J}^*(s)$ as the minimum cost to reach a target state by the execution of an available action $a \in Q(s)$ is represented as

$$\tilde{J}^*(s) = \min_{a \in Q(s)} \sum_{s'} T(s, a, s')(\Gamma_a + \tilde{J}^*(s')), \tag{5-24}$$

with the cost Γ_a of executing action a in state s and the transition probability $T(s, a, s')$ defining the probability with which an action a leads to a target state s'. If deterministic behaviour of state transitions is assumed, the transition probability $T(q_0, a, q_1)$ that an action a executed in an initial trim state q_0 will lead to a new trim state q_1 can be considered 1, simplifying (5-24) to

$$\tilde{J}^*(s) = \min_{a \in Q(s)} (\Gamma_a + \tilde{J}^*(s')). \tag{5-25}$$

Depending on the nature of action a, the action cost Γ_a corresponds to the cost γ_q of following a trim trajectory for a coasting time τ or the manoeuvre cost Γ_m

$$\Gamma_a = \begin{cases} \gamma_q \tau \text{ if } a \in Q_T \\ \Gamma_m \text{ if } a \in Q_M \end{cases}. \tag{5-26}$$

An approximation of the optimal control policy $\tilde{J}^*(s)$ can be obtained iteratively over all states $s = [q, h]^T$ such that

$$J_i^*(s) = \min_{a \in Q(s)} (\Gamma_a + J_{i-1}^*(s')). \tag{5-27}$$

To be used in the above value iteration, a discrete approximation of the continuous state $h \in H$ must be found. This can be achieved with a finite number of representative states $h_i,\ i = 1...N$. A piece-wise linear approximation of the set H into a collection of simplices with disjoint interiors is used in this implementation (see 2.4.1 and [21], [49]). Any state h will be a member of one simplex with vertices $h_{i,1}, h_{i,2}, ...h_{i,n}$, such that the cost function at h can be written as

$$\hat{J}(q, h) = \sum_{j=1}^{n+1} \lambda_{i_j} \hat{J}^{i_j}(q), \tag{5-28}$$

with

$$h = \sum_{j=1}^{n+1} \lambda_{i_j} h_{i_j}, \tag{5-29}$$

where λ_{i_j} are non-negative coefficients that are selected to describe h as a linear approximation of the representative states h_i. The parameter n represents the dimension of H.

If the optimal cost function J^* is approximated in an offline computation step (for reference and a simplified example of a value iteration see 2.4.1 and Fig. 2-8) the execution of a greedy policy on J^* in the online phase results in a near-optimal and deterministic control strategy within the limitations of the motion library Q and under the assumption of small disturbance of the actuated system. These properties fulfil contribution C-5b.

The application of the above optimal manoeuvre automaton will result in collision-free transitions between trajectories, given that the deterministic assumption on the transition probability between two trim states $T(q_0, a, q_1)$ holds. To extend this property to situations, in which the vehicle is subject to bounded uncertainty and disturbance, the AROC toolbox[2] is leveraged to synthesise optimal manoeuvre controllers. These controllers minimise a system's reachable set under bounded external forces acting on the system and with uncertainty in the system's state estimation. For the purpose of controller synthesis, a two-dimensional kinematic vehicle model $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{w})$, which corresponds to the definition of the manoeuvre automaton, is introduced. The system state

$$\vec{x} = [\Psi, v_k, p_x, p_y]^T + \delta\hat{\vec{x}} \tag{5-30}$$

contains the heading angle Ψ, kinematic velocity v_k and the vehicle position (p_x, p_y) in the two-dimensional configuration space and can be measured with an uncertainty

[2]https://tumcps.github.io/AROC/

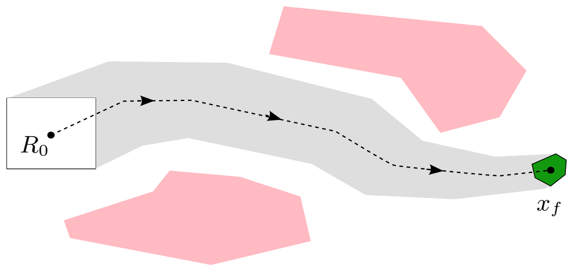


Figure 5-21: A representation of a reach-avoid problem between an initial set R_0 and final set x_f. Figure is based on [37].

$\delta\hat{\vec{x}}$. Bounded control inputs are the azimuth rate $\dot{\Psi}$, with $|\dot{\Psi}| \leq \dot{\Psi}_{max}$, and translational acceleration a_k, with $|a_k| \leq a_{k_{max}}$. Further, a bounded disturbance

$$\vec{w} = [w_{\dot{\Psi}}, w_{a_k}]^T, \text{ with } |w_{\dot{\Psi}}| \leq w_{\dot{\Psi},max}, |w_{a_k}| \leq w_{a_k,max} \tag{5-31}$$

on the control input is considered, such that the vehicle dynamics can be expressed as

$$\dot{\vec{x}} = \begin{bmatrix} \dot{\Psi} \\ \dot{v}_k \\ \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} \dot{\Psi} + w_{\dot{\Psi}} \\ a_k + w_{a_k} \\ \cos(\Psi) v_k \\ \sin(\Psi) v_k \end{bmatrix}. \tag{5-32}$$

The dynamic vehicle model (5-32) with states $\vec{x}$, control inputs $\vec{u}$ and disturbance $\vec{w}$ is used to synthesise reach-set optimal controllers using AROC. In this dissertation, the AROC toolbox is applied to a given problem and without providing deeper insights into the implemented control and reachability algorithms. For details on the AROC implementation, control algorithms and controller synthesis, the reader is referred to the relevant publications [64], [65], [66] and the AROC manual [37].

Based on the vehicle dynamics (5-32) and the reference motion primitives describing trim states and manoeuvres in the motion library Q, the AROC toolbox can be used to express the motion library as a set of optimal controllers. These controllers are vehicle-specific and generated offline, prior to flight. In flight, a trajectory is generated from a sequence of controllers representing manoeuvres and trim states. Two controllers can be executed consecutively if the current controller's final set is contained in the next controller's initial set. If a transition trajectory is built from primitives, of which no reachable set violates the transition region, a transition is guaranteed to be safe within the boundaries of considered disturbance (see Fig. 5-21).

5.7.3 A Feasibility Criterion for Trajectory Transitions Inside Convex Regions

The online planning method described in the previous section requires an external decision system to request a transition from the current trajectory onto another. This decision system is assumed to be available on the aircraft or to communicate with the motion planner from the ground through a data link. Transitions between trajectories inside transition regions require the initial and target state to be located inside the transition region such, that the near-optimal control policy will not violate the respective region boundaries. A transition request is divided into the following steps ([49]):

1. Receive a target trajectory τ_t' inside the current region E_0.
2. Select the target state s_t on τ_t'.
3. Test if the transition is feasible.
4. Compute a manoeuvre sequence from a greedy policy on the optimal cost function J^*.
5. Execute the manoeuvre controller sequence.

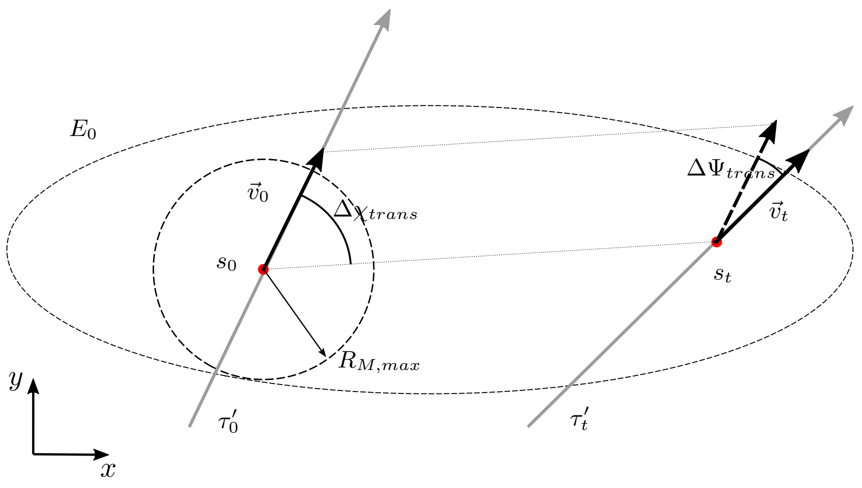


Figure 5-22: The feasibility criterion for a transition between an initial state s_0 and target state s_t on trajectories τ_0' and τ_t' inside the region E_0. The transition is feasible if $R_M(max(\Delta\chi_{trans}, \Delta\Psi_{trans}), v_0) \leq R_{M,max}$ is fulfilled. From [49].

This work implements a simple target state selection, which defines the target state s_t on the target trajectory τ_t' at a distance $d_{\delta E_0}$ inside the region's boundary δE_0. A feasibility criterion is developed, which takes the vehicle's initial and target states s_0 and s_t into account, and assesses if a transition between these states will be contained in E_0. A transition is feasible, if the manoeuvre radius R_M to achieve the required course change is less than the clearance $c_{l,\delta E_0}(s_0, s_t)$ to the region boundary δE_0 in states s_0 and s_t. Figure 5-22 illustrates the feasibility criterion for a transition between two states. The transition starts in state s_0, in which the vehicle has velocity $\vec{v}_0$ and terminates in state s_t with velocity $\vec{v}_t$. The angle between the velocity $\vec{v}_0$ and the direct line between s_0 and s_t is referred to as the transition course change $\Delta\chi_{trans}$ and the angle between $\vec{v}_0$ and $\vec{v}_t$ as the heading change $\Delta\Psi_{trans}$. The manoeuvre angle α is defined as the larger of these angles

$$\alpha = max(\Delta\chi_{trans}, \Delta\Psi_{trans}). \tag{5-33}$$

When the manoeuvre angle α is known, the manoeuvre radius R_M can be computed as

$$R_M = \sqrt{\sin(\alpha)^2 + |\cos(\alpha) - 1|^2}\frac{|\vec{v}_0|^2}{\tan(\Phi_{max})g}, \tag{5-34}$$

where g is gravity and Φ_{max} the roll angle, at which the maximum load factor $n_{L,max}$ is achieved. With the manoeuvre radius R_M, the feasibility criterion is defined as

$$c_{l,\delta E_0}(s_0, s_t) \overset{!}{\geq} kR_M(\alpha, \vec{v}_0), \tag{5-35}$$

with a safety factor $k \geq 1$. Similar to the holding pattern criterion in (5-16), this criterion can be improved in a way that it is aware of the direction of the manoeuvre angle α and nearby region boundaries in the opposite direction will not prevent safe transitions. If a set of initial and target states (s_0, s_t) satisfy the feasibility criterion, the online planning method generates a transition trajectory and executes the corresponding controller sequence. A successful planning result of a safe transition between pre-planned trajectories using the presented online planning method is shown in Fig. 5-23.

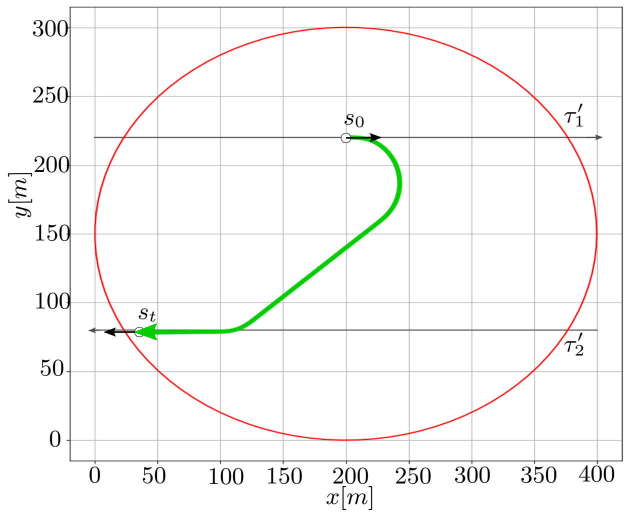


Figure 5-23: A successful transition between two states s_0 and s_t on trajectories τ_1' and τ_2'. The transition is executed if the feasibility criterion described in (5-35) and Fig. 5-22 holds. From [49].

6 Validation of the Developed Motion Planning Framework

The methods developed and described in chapter 5 are applied to compute databases of trajectories connecting vertiports in a randomised obstacle environment and an application scenario of realistic extent. It is shown how the selection of planning parameters affects memory requirements and the granularity of the planning solution. Based on the random obstacle map and the trajectory database obtained for a mission between two vertiports on this map, obstacle-free regions are identified. These regions are used to validate the online planning approach presented in chapter 5.7. For this purpose, an optimal manoeuvre automaton with reach-set optimised controllers is generated. The planning framework is then transferred to the UAM context and demonstrated in a realistic application scenario and considering risk as a cost parameter following chapter 4. The results are presented and discussed to outline strengths and limitations of the developed solutions. It is shown that the planning solutions to both scenarios respect the planning rules defined in section 5.4.

All planning results are computed using implementations of the described algorithms in *Python* and on a workstation with 32 GB RAM and an Intel Xeon W-2125 CPU running Windows 10.

6.1 Demonstration of the Planning Method in a Randomised Obstacle Environment

In the following, the offline planning phase of the proposed planning framework (see left-hand side of Fig. 5-2) is demonstrated in a randomised obstacle environment. The environment contains $n_{obs} = 1000$ obstacles of rectangular shape and uniformly distributed height between $0m$ and $500m$ on flat terrain. The maximum dimension of a single obstacle is $150m$x$150m$. A database of trajectories is computed on a flight planning surface connecting the departure and arrival vertiports VP_1 and VP_2 at a cruise altitude $h_{cruise} = 250m$ AGL. A summary of map properties and applied planner parameters is provided in Tab. 6-1. Planning results of different branch factors N, path length restriction factors k and contingency planning interval Δt_C are compared to assess the parameters' impact on the database size. The minimum vehicle turn radius r is determined under consideration of a maximum load factor $n_{L,max} = 1.50$ and cruise air speed $v_a = 18\frac{m}{s}$. Using

$$tan(\Phi) = \frac{{v_a}^2}{rg} \text{ and } n_L = \frac{1}{cos(\Phi)}, \tag{6-1}$$

the minimum turn radius results to $r = 30m$, where g represents gravity.

A top view of the randomised obstacle environment with four vertiports is shown in Fig. 6-1. The vertiport coordinates are listed in Tab. 6-2. It shall be noted that the obstacle map represents all obstacles on ground level and indicates the obstacle height according to the indicated colour code.

6.1.1 Planning on the Nominal Flight Planning Surface

The search space reduction method and planning approach presented in sections 5.4.1 and 5.4.2.1 are applied to the artificial planning environment. As a result of the uniform obstacle height distribution on the interval $[0, 500]m$ and neglecting adjustments from the semi-circular rule in this artificial test scenario, applying (5-4) yields a cruise altitude $h_{cruise} = 250m$. Under consideration of the path angle $\gamma = 7.5°$ and coordinates of the start and destination vertiports VP_1 and VP_2, the flight altitude profile H is computed following (5-2). The flight surface, which slices the obstacle environment along the flight altitude profile H, is computed across a rectangular map section that is defined by the vertiport coordinates and extended with a margin $M_{\mathcal{S}}$. An illustration of the flight surface $\mathcal{S}^*$ through vertiports VP_1 and VP_2 and using a map margin $M_{\mathcal{S}} = 500m$ is provided in Fig. 6-2a). As all subsequent planning activities are limited to such surfaces, the flight surface enforces rules 2. to 4. of section 5.4. In the next step, the flight surface $\mathcal{S}^*$ is projected according to (5-5) to obtain the planning surface $\mathcal{S}$. The new obstacle map on the planning surface $\mathcal{S}$ is illustrated in Fig. 6-2b). The obstacle density is reduced considerably compared to the environment's ground obstacle map in Fig. 6-1.

Once the planning surface's obstacle map is obtained, a maximum clearance roadmap $\mathcal{G}$ is generated and pruned on $\mathcal{S}$ following rules 5. and 6. of section 5.4 and under consideration of the parameters listed in Tab. 6-1. Fig. 6-3a) displays the resulting search graph on the three-dimensional flight surface, whereas Fig. 6-3b) shows the same graph on the projected planning surface. For each vertiport, the safety radius R_{CSFL} for continued safe flight and landing is depicted in red colour. Regions outside the union of the vertiports' safety radii are removed from the search graph such that any solution on $\mathcal{G}$ is guaranteed to remain within safe distance to a vertiport at all times. The consecutive planning steps are then run on this 2D roadmap, which represents the safe and rule-compliant, three-dimensional configuration space for the given planning task.

The shortest path connecting a start and goal location is identified from a search on the roadmap $\mathcal{G}$ using Dijkstra's graph search algorithm. The applied cost function follows (5-6). Due to the artificial nature of the demonstration scenario, the risk cost item C_r is neglected, reducing the expression of the total cost to

$$C_{tot} = p_w L_e, \tag{6-2}$$

with the edge length L_e and wind penalty factor p_w (see (5-7)).

Table 6-1: Map properties and flight planner settings applied to the demonstration scenario.

Parameter	**Representation**	**Value**	**Unit**
path angle	γ	7.5	$^\circ$
cruise altitude	h_{cruise}	250	m AGL
air speed	v_a	18	$\frac{m}{s}$
vertical obstacle clearance	$c_{v,min}$	11	m
lateral obstacle clearance	$c_{l,min}$	50	m
max. load factor	$n_{L,max}$	1.50	-
contingency planning interval	Δt_C	$20, 30, 60$	s
interval between holding pattern entries	Δt_{hp}	20	s
flight levels	FL_i	$150, 200, 250, 300$	m AGL
branch factor	N	$2, 3, 4$	-
path length restriction factor	k	$1.5, 2$	-
min. distance between fork points	d_{min}	250	m
CSFL radius	R_{CSFL}	3500	m
near-vertiport radius	r_{ls}	100	m
map dimension	-	10000 x 10000	m
map resolution	-	1	m
navigation function resolution	-	10	m
number of obstacles	n_{obs}	1000	-
number of vertiports	n_{VP}	4	-

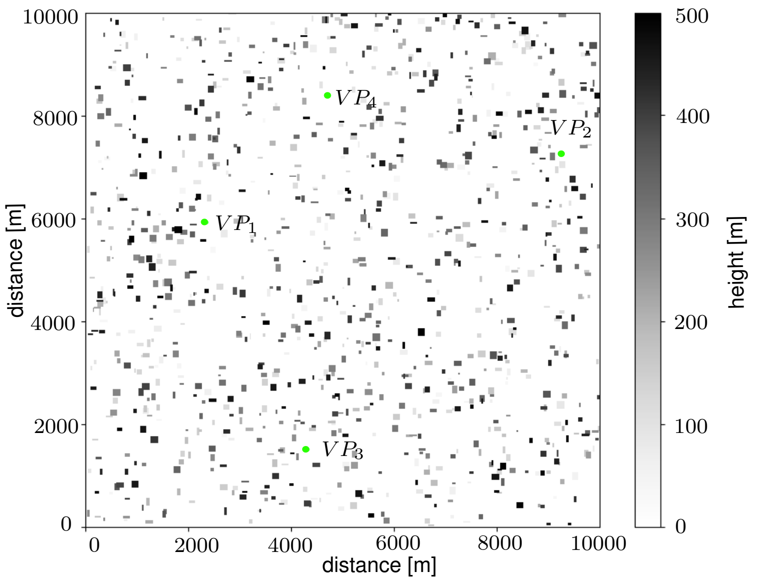


Figure 6-1: A top view of the ground obstacle map with vertiports VP_1 to VP_4.

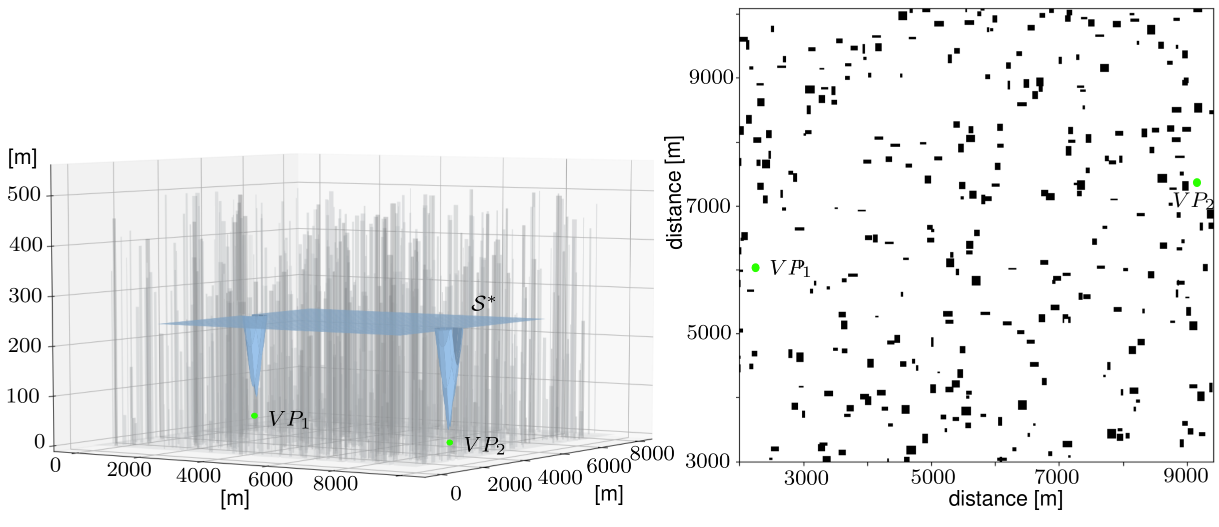


Figure 6-2: Flight surface $\mathcal{S}^*$ and planning surface $\mathcal{S}$ obstacle map.

Table 6-2: Vertiport coordinates.

Vertiport	Coordinates
1	(2358,5942,54)
2	(9270,7280,35)
3	(4321,1523,23)
4	(4681,8407,16)

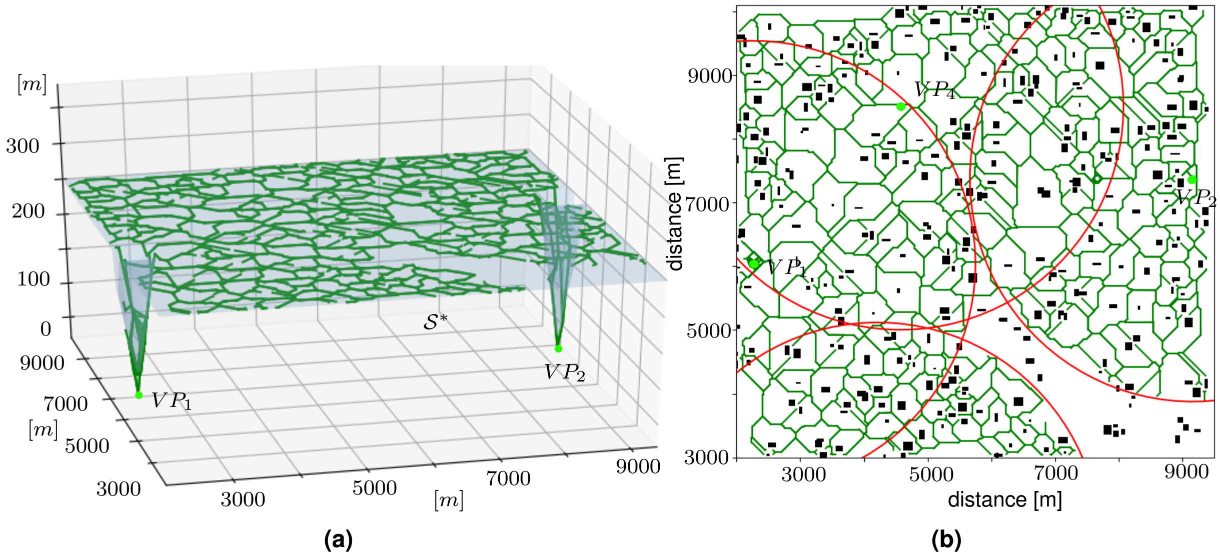


Figure 6-3: The pruned maximum clearance roadmap on $\mathcal{S}$ and as it spans across the flight surface $\mathcal{S}^*$.

6.1.1.1 Bi-Directional Corridor Operations

When multiple vehicles should be operated between two vertiports repeatedly and in opposite directions, the road lane method presented in section 5.5 ensures that oncoming traffic is separated safely while still maintained inside the same flight path corridor τ. A potential application scenario may e.g. be a supply route between a remote location and the nearest village. In order to achieve vertical and horizontal offsets of the flight path's lanes, the distance parameters $d_{xy,rl}$ and $d_{z,rl}$ for horizontal and vertical separation, respectively, are selected to account for the vehicle dimensions and TSE as documented in Tab. 6-3. For the purpose of this functional demonstration, it is considered acceptable to neglect a model and analysis of the actual TSE and proceed with an estimated value.

To ensure that all lanes for each flight path respect the minimum obstacle clearance requirements, the roadmap graph $\mathcal{G}$ is recomputed with the updated horizontal obstacle clearance

$$d^*_{xy,min} = d_{xy,min} + \frac{d_{xy,rl}}{2} = 80m. \tag{6-3}$$

The six resulting bi-directional flight paths connecting all four vertiports in the network are illustrated on the environment's ground obstacle map in the left-hand side of Fig. 6-4. The zoomed displays of the near-vertiport environments show the action radius $r_{ls} = 100m$ of the near-vertiport planner (see Section 5.4.2.2) and how incoming and outgoing trajectories are aligned with a randomly selected wind direction. Helix manoeuvres to gain or reduce vertical separation between two lanes in a flight corridor are injected when entering or before leaving the approach zone to avoid excessive manoeuvring close to the landing pad. This practise can temporarily reduce the obstacle clearance during the climb manoeuvre to a value $c_l < c_{l,min}$. As long as the helix turn radius $r_{h,rl}$ is smaller than the minimum obstacle clearance $c_{l,min}$ (compare Tab. 6-1 and Tab. 6-3), a helix turn will still be collision-free. This behaviour can be avoided and rule-compliance enforced if climb manoeuvres are accepted closer to the landing pad. The right-hand side of Fig. 6-4 shows the reciprocal values of the obstacle clearance for all bi-directional flight paths. Based on the minimum obstacle clearance $c_{l,min} = 50m$, values less than $\frac{1}{50m}$ indicate compliance with the requirement. The observed outlier refers to a climb manoeuvre at vertiport VP_2, which violates the minimum obstacle clearance following the described logic. All other paths and helix manoeuvres respect the given rule set.

Table 6-3: Road lane parameter settings.

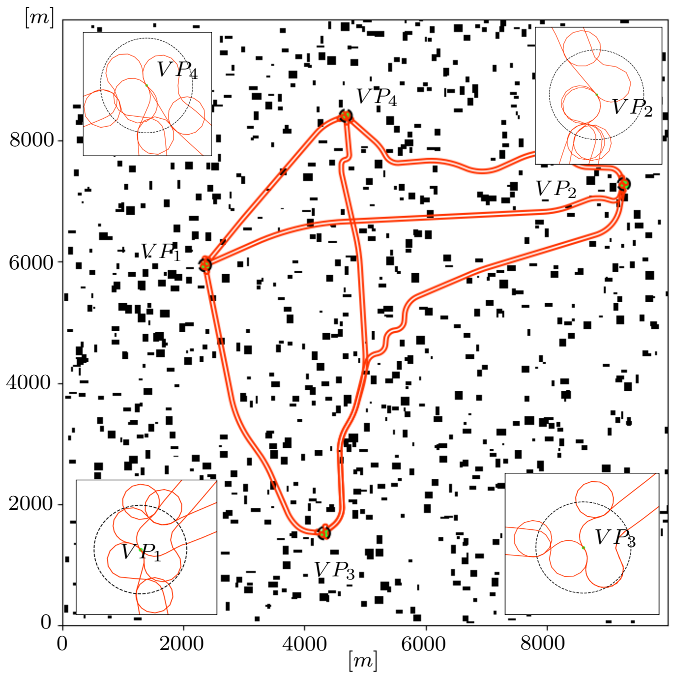
Parameter	Representation	Value
horizontal offset	$d_{xy,rl}$	$60m$
vertical offset	$d_{z,rl}$	$30m$
helix radius	$r_{h,rl}$	$40m$
helix path angle	$\gamma_{h,rl}$	$7.5°$

6.1.1.2 Computing a Tree of Trajectories to the Destination Vertiport

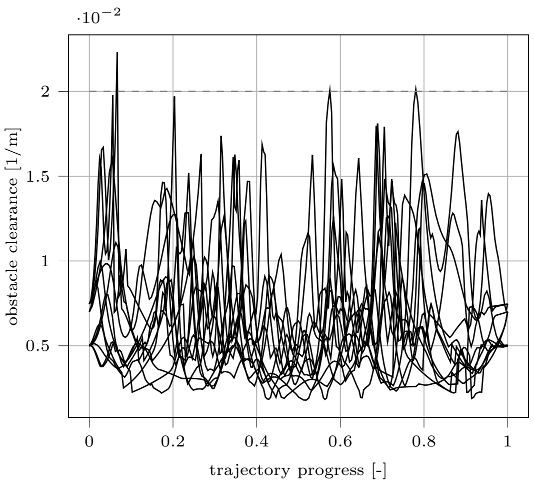
Beyond the case, where multiple vehicles are operated on a single flight path corridor, the generation method for different alternative flight paths between a start and destination vertiport as described in section 5.6.1 is validated. Starting from an initial optimal path τ_{opt} connecting vertiports VP_2 and VP_1, penalty weights $w_{p,i}$ and $w_{rj,i}$ are applied to edges e_i on or leading to paths from any previous iteration. The new total cost of a graph edge e_i can therefore be expressed as

$$C'_{tot,i} = (1 + B_p w_{p,i} + B_{rj} w_{rj,i}) C_{tot,i} = (1 + B_p w_{p,i} + B_{rj} w_{rj,i}) p_w L_{e,i}, \qquad \text{(6-4)}$$

with $B_p, B_{rj} \in \{0, 1\}$ being boolean indicators of the respective penalty case. Without further optimization of the parameter values, the penalty weights are chosen as $w_{p,i} = 3$ and $w_{rj,i} = 1.5$ empirically.



(a)



(b)

Figure 6-4: (a): Road lanes of flight path τ_i enabling multiple vehicles to operate in a network of vertiports and in opposite directions plotted on the ground obstacle map. Zoomed sections show altitude separation manoeuvres and final approaches. (b): Obstacle clearance as the reciprocal of the distance to the nearest obstacle for all bi-directional flight paths.

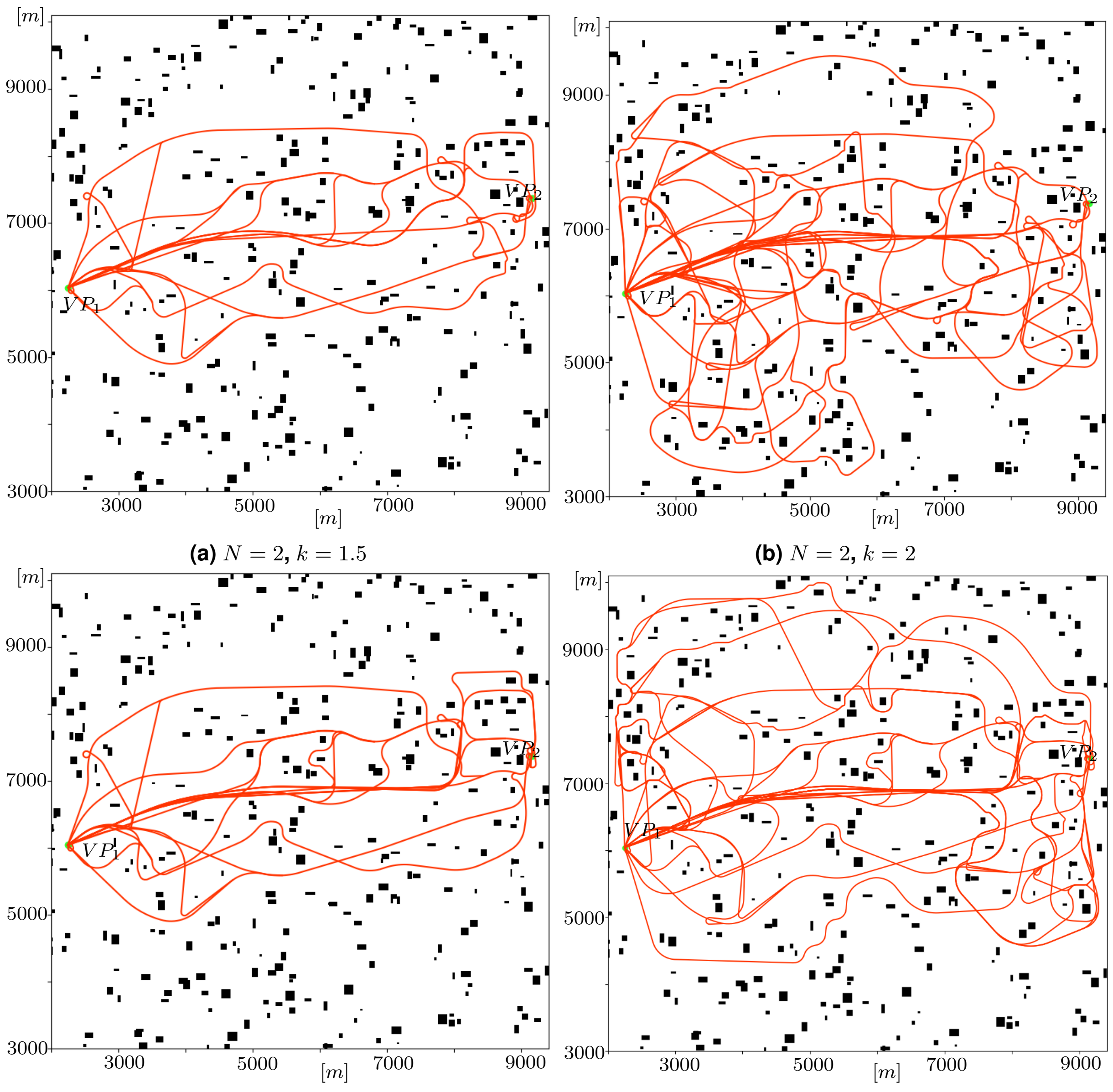
Trajectory trees $\mathcal{T}_{N,k}$ connecting vertiport VP_2 to vertiport VP_1 are shown in Fig. 6-5 for branch and path length restriction factors $N \in \{2,3\}$ and $k \in \{1.5,2\}$, respectively. When comparing $\mathcal{T}_{2,1.5}$ to $\mathcal{T}_{3,1.5}$ and $\mathcal{T}_{2,2}$, it can be observed that, despite the expectation that the number of paths grows polynomially in $\mathcal{O}(n^m)$ with the branch factor N (see Tab. 5-1 for reference), a relaxation of the path length constraint has a much larger impact on the number of alternative paths than the branch factor. This can be explained by the notion, that the implemented edge penalty scheme limits the number of attractive paths to an extent, that a lower branch factor already exploits a significant portion of the available solution space. Relaxing the path length constraint, however, increases the solution space and more valid alternative paths can be found. A quantitative analysis of a planner parameter's impact on the resulting database is presented in the following section in Tab. 6-4. Further investigations analysing the coupling of the path penalty scheme and planner parameters are not conducted in the scope of this validation case, as this work focuses on the introduction of the general motion planning framework rather than the optimization of the individual planning method's parameter set.

The reciprocal values of the obstacle clearance across the trajectory tree $\mathcal{T}_{2,2}$ are shown in Fig. 6-6. No violation of the minimum clearance requirement can be observed.

6.1.2 Contingency Planning on Contingency Flight Levels

To enable contingency planning according to CSFL requirements and following chapter 5.6, additional planning surfaces and navigation functions are computed for each alternate vertiport at altitudes $150m, 200m, 250m$ and $300m$ (see also Tab. 6-1). In this demonstration scenario, alternate vertiports are VP_2, VP_3 and VP_4. Figure 6-7 displays the contingency navigation functions of all alternate vertiports and flight levels. Dark colouring indicates a large distance to the vertiport, where brighter areas are closer to the vertiport. Obstacles are represented as white polygons with grey outline. It can be seen across all navigation functions that the obstacle density decreases with increasing altitude. This implies that upward flight level changes can always be executed, a downward change, however, requires a prior check if the transition will be collision-free (compare to Fig. 5-17).

A CSFL planning solution (blue) for the initial shortest path τ'_{opt} in the tree of trajectories $\mathcal{T}_{2,2}$ (red) with contingency planning interval $\Delta t_C = 30s$ and computed based on the navigation functions in Fig. 6-7 is shown in Fig. 6-8a). The transition between each contingency trajectory and its parent is ensured to be continuous using the Dubins transition planner described in section 5.6. With a contingency trajectory to every remaining alternate vertiport available every $30s$, 3231 contingency trajectories are gen-



(a) $N = 2$, $k = 1.5$ **(b)** $N = 2$, $k = 2$

(c) $N = 3$, $k = 1.5$ **(d)** $N = 3$, $k = 2$

Figure 6-5: Trajectory trees $\mathcal{T}_{N,k}$ connecting vertiports VP_2 and VP_1 with different branch and path length restriction factors N and k. The impact of the branch factor on a trajectory tree's growth is smaller than expected from Tab. 5-1.

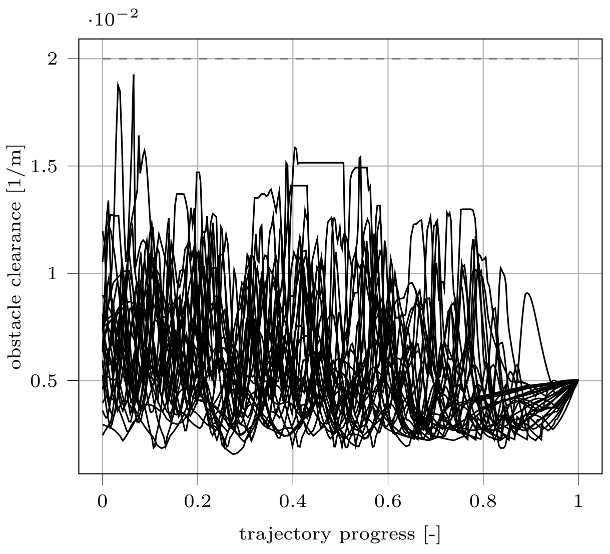


Figure 6-6: Obstacle clearance of the trajectory tree $\mathcal{T}_{2,2}$ connecting vertiports VP_2 and VP_1 with branch factor $N = 2$ and path length restriction factor $k = 2$.

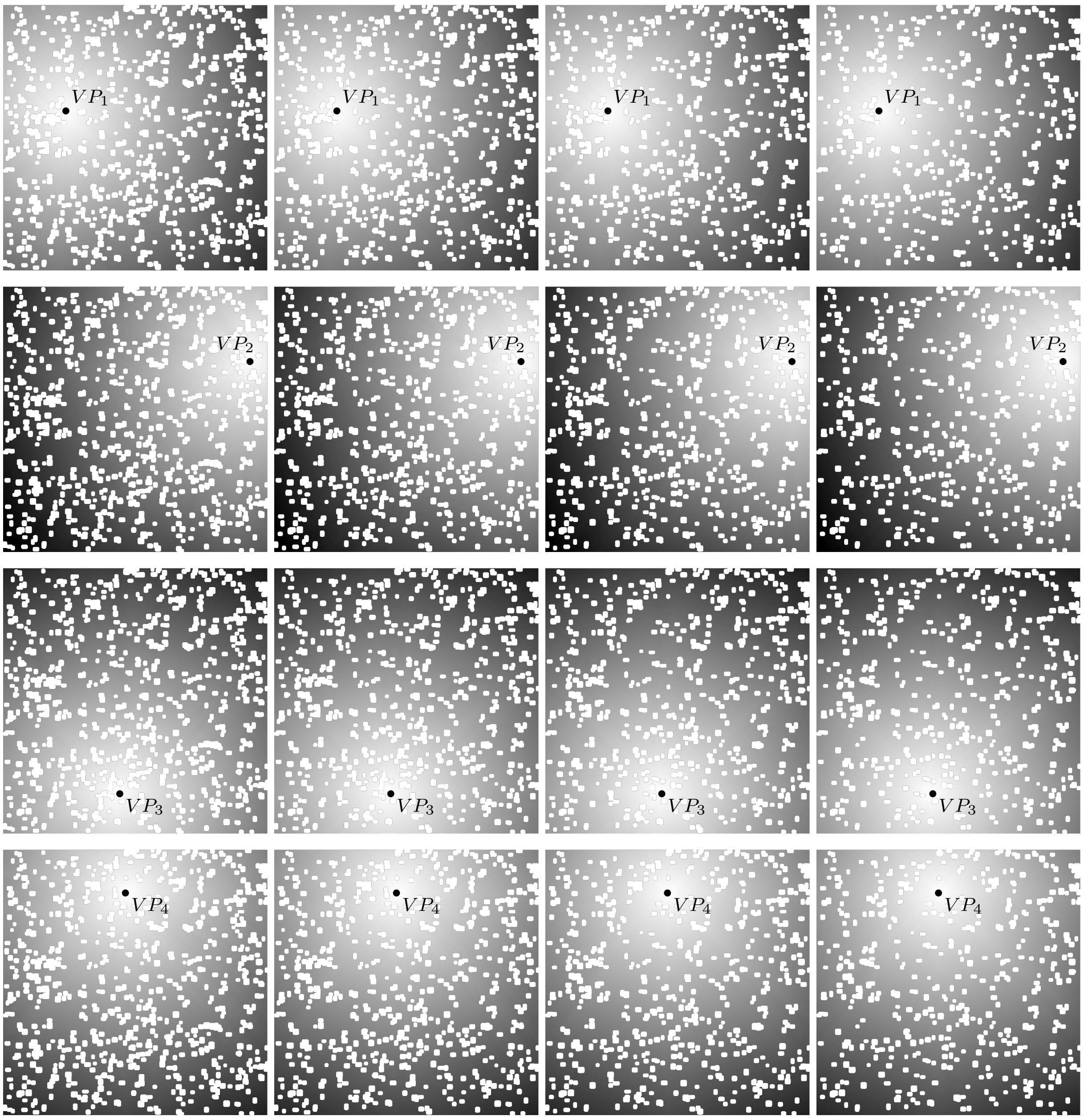
erated for the optimal path τ'_{opt}. To improve clarity and readability of the illustration only every second trajectory is displayed.

Identical planning efforts to the one for the optimal trajectory τ'_{opt} are made for all trajectories in the tree $\mathcal{T}_{2,2}$ and collected in a trajectory database. This database represents the action space for the mission connecting VP_2 with VP_1. Figure 6-8b) validates that all contingency trajectories respect the minimum obstacle clearance requirement despite the limits of the navigation function's resolution. As navigation functions resolve at $c = 10m$, the minimal obstacle clearance $c_{l,min}$ must be adjusted conservatively to account for discretisation-implied inaccuracies. Using the actual obstacle clearance

$$c_l \geq c_{l,min} + \frac{\sqrt{2}c}{2} \tag{6-5}$$

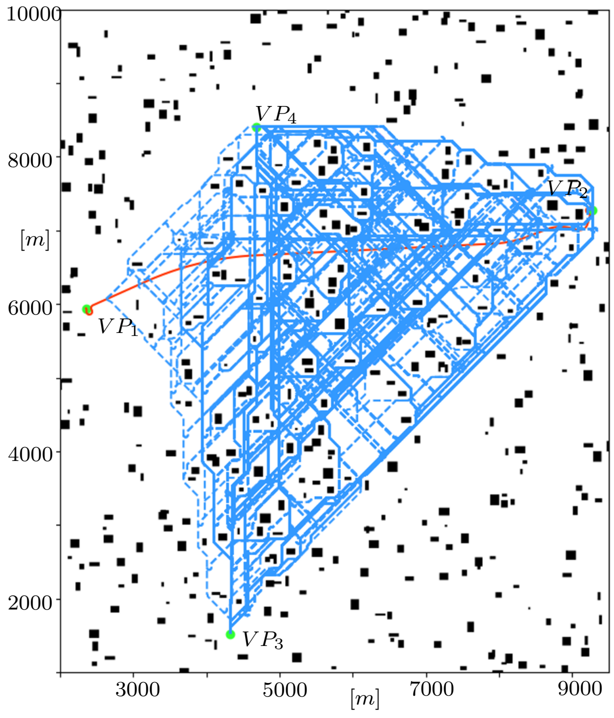
for planning contingency trajectories, rule-compliance is enforced.

Table 6-4 analyses the overall database sizes and differences in memory requirements between planning solutions using different sets of parameters. The impact of the branch factor N on the database size is over-estimated considerably in Tab. 5-1. As stated previously, this is likely due to effects of the applied penalisation scheme. Approximately cubic growth of the database can be observed with the path length restriction factor k, which corresponds to the expectation that the database grows quadratically with the larger solution space and linearly with increased trajectory length. Exponential growth of the number of trajectories in the final database is expected with decreasing contingency planning intervals Δt_C, which is confirmed by the obtained planning results. A detailed analysis of the impact that different penalisation strategies and parameters may have on planning results is not considered in the scope of this work.

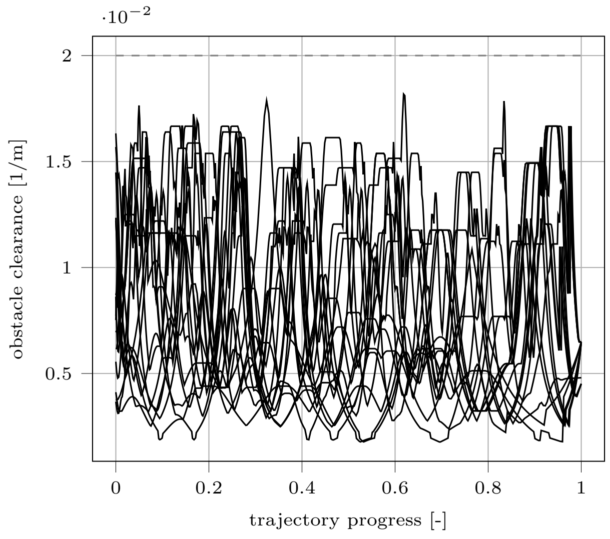


(a) altitude $150m$ **(b) altitude** $200m$ **(c) altitude** $250m$ **(d) altitude** $300m$

Figure 6-7: Navigation functions for vertiports VP_1 **to** VP_4 **(top to bottom) across all contingency flight levels. Bright regions indicate that the alternate vertiport is near. Obstacles are represented as white polygons with grey outline. All navigation functions have quadratic dimensions of** $10km$***x***$10km$**.**



(a)



(b)

Figure 6-8: (a): Contingency trajectories for a single nominal trajectory τ' with contingency planning interval $\Delta t_C = 30s$. Trajectories are plotted on the obstacle map of the contingency planning surface $\mathcal{S}_{c,250}$. (b): Reciprocal of obstacle clearance of contingency trajectories over the normalised progress.

Table 6-4: Offline planning results for a mission connecting VP_2 and VP_1 in the random obstacle environment.

run	N	k	Δt_C	trajectories to destination	trajectories to alternate	size on memory
1	2	1.5	$30s$	17	45404	5.04 GB
2	2	2.0	$30s$	34	115994	13.00 GB
3	3	1.5	$30s$	20	56215	6.24 GB
4	3	2.0	$30s$	39	141952	15.70 GB
5	2	1.5	$20s$	17	160253	20.40 GB
6	2	2.0	$20s$	34	394627	38.02 GB
7	3	1.5	$20s$	20	198854	20.70 GB
8	3	2.0	$20s$	39	481043	45.06 GB
9	2	1.5	$60s$	17	7138	0.82 GB
10	2	2.0	$60s$	34	21772	2.48 GB
11	3	1.5	$60s$	20	10699	1.20 GB
12	3	2.0	$60s$	39	22475	2.52 GB

6.1.3 Conclusion

The earlier sections show that it is feasible and practical to preprocess and reduce the search space for a motion planning task such, that it forces any solution in said search space to comply with an applicable set of rules. When the search space omits non-compliant regions, solutions will only differ in efficiency (or optimality) but not in feasibility. The VLL airspace can be structured similar to higher airspace using distinct flight levels. This structure facilitates to deviate from an initial trajectory without the need for a replanning at flight time.

Concepts for bi-directional operation in narrow corridors are presented. Further, it is shown that even missions in complex obstacle environments can be covered with databases of trajectories from a preplanning phase. Figures 6-4 and 6-6 indicate that the performance of individual planning methods could be optimised with additional effort. A solution to place vertical offset manoeuvres between trajectories on opposite directions, such that obstacle clearance and distance to the landing site are maintained at the same time may be found. Similarly, an optimised penalty strategy to compute alternative trajectories to the destination vertiport could result in a more uniform distribution of trajectories across the search space. It is found that the time interval, in which alternative trajectories are provided at each point of the mission, is the most significant impact parameter for the database size. The developed approach allows an aircraft to react to unforeseen events at flight time, while the union of reachable flight states can be assessed and verified before the flight. A decision making logic, which evaluates the current situation in flight and selects the appropriate action from the database can be implemented e.g. following propositions made in [76]. The memory requirements for trajectories could further be reduced with more advanced data management. For instance, trajectory segments, that are shared by multiple trajectories, may only be saved once and referenced by each related trajectory.

The decision, whether or not a database has sufficiently high resolution to conduct a mission is expected to lie with the competent authority or USSP. In both cases, models and methods to evaluate and assess appropriate database parameter settings will be required. The method cannot ensure deconfliction between vehicles during take-off and final approach in the immediate proximity to vertiports. Ensuring sufficient time between movements for aircraft to clear the vertiport remains within the responsibility of the vertiport operator.

6.2 Protected Online Planning in a Randomised Obstacle Environment

This section evaluates obstacle-free volumes in the above obstacle environment and validates the protected online planning method described in section 5.7 using the plan-

Table 6-5: Trim states of the optimal manoeuvre automaton.

Description	Yaw Rate $\dot{\Psi}$	Ground Speed v	Load Factor n_L
hover	$0\ ^\circ/s$	$0\frac{m}{s}$	1.00
forward	$0\ ^\circ/s$	$15\frac{m}{s}$	1.00
left turn	$24.87\ ^\circ/s$	$15\frac{m}{s}$	1.20
right turn	$-24.87\ ^\circ/s$	$15\frac{m}{s}$	1.20

ning results from section 6.1.2. An optimal manoeuvre automaton is defined and expressed as a set of reach-set optimised controllers following section 5.7.2. Obstacle-free transition areas are computed and linked with the trajectory database. Transitions between randomly selected states on pre-planned trajectories of the initial database and inside the same obstacle-free region are computed to validate the online planning method.

6.2.1 Generation of an Optimal Manoeuvre Automaton

For the purpose of validating the online planning method inside pre-computed convex regions, a manoeuvre automaton of four trim states is generated. These "[...] trim states represent the minimal operational envelope of a fictional VTOL vehicle in two spatial dimensions. In addition to a hover state, a forward flight at a single velocity v as well as left and right turns at that velocity v and at a selected load factor n_L [...]" ([49]) are defined. Manoeuvres enable transitions between trim states. An exception are transitions between left and right turns, which can only be achieved via an intermediate state (compare to Fig. 2-6). The vehicle state

$$s = [q, h]^T, \text{ with } h = [\Psi, v_k, p_x, p_y] \tag{6-6}$$

is described as a trim state q and the global state h, containing the heading Ψ, kinematic velocity v_k and position (p_x, p_y) in two-dimensional space. It is assumed that the vehicle achieves changes in direction using coordinated turns. The load factor n_L is, hence, defined by the velocity and azimuth rate. Trim states and manoeuvres in the manoeuvre automaton's motion library Q are described comprehensively in Tab. 6-5 and Tab. 6-6. Figure 6-9 displays the spatial displacement of each manoeuvre in the library subset Q_M.

From the above definition, an optimal manoeuvre automaton is derived using the value iteration and state space discretisation methods presented in Sections 2.4.1 and 5.7.2. If the optimal cost function is approximated in an offline computation step, a greedy policy in the online phase results in a near-optimal and deterministic control strategy.

Table 6-6: State transitions of the manoeuvre automaton.

Description	Heading Change $\Delta\Psi$	Movement $\Delta\vec{x}$
hover / forward	$0°$	$[61.5, 0]^T\, m$
hover / right	$-78.10°$	$[35.4, -39.0]^T\, m$
hover / left	$78.10°$	$[35.4, 39.0]^T\, m$
forward / hover	$0°$	$[58.5, 0]^T\, m$
forward / right	$-20.28°$	$[20.7, -2.4]^T\, m$
forward / left	$20.28°$	$[20.7, 2.4]^T\, m$
right / hover	$-83.19°$	$[31.4, -40.0]^T\, m$
right / forward	$-14.55°$	$[20.6, -3.9]^T\, m$
left / hover	$83.19°$	$[31.4, 40.0]^T\, m$
left / forward	$14.55°$	$[20.6, 3.9]^T\, m$

The value iteration is set up under the assumption of deterministic state transitions and following (5-27) as

$$J_i^*(s) = \min_{a \in Q(s)} (\Gamma_a + J_{i-1}^*(s')). \tag{6-7}$$

For the functional validation of this method unit cost $\Gamma_a = 1$ is applied to every trim state primitive and manoeuvre execution, simplifying (6-7) to

$$J_i^*(s) = \min_{a \in Q(s)} (1 + J_{i-1}^*(s')). \tag{6-8}$$

A trajectory computation will therefore produce the shortest possible manoeuvre sequence within the limitations of the motion library. This sequence, however, may not necessarily be time-optimal.

Based on empirical observations, the acceptance criterion to reach a given target state for (6-8) is defined as

$$\epsilon = \begin{bmatrix} \epsilon_{v_k} \\ \epsilon_{\Psi} \\ \epsilon_{\dot{\Psi}} \\ \epsilon_p \end{bmatrix} = \begin{bmatrix} 0.5\frac{m}{s} \\ 5° \\ 5\ °/s \\ 1m \end{bmatrix}. \tag{6-9}$$

From the disturbance and vehicle dynamic model described in (5-31) and (5-32), respectively, the vehicle's motion library is expressed as a set of reach-set optimised controllers. The control law synthesis uses the convex interpolation control approach implemented in the AROC tool box and first presented in [64]. The work leading to the development of and describing this toolbox is considered to provide sufficient verification of the employed methods and no further verification of the AROC implementation

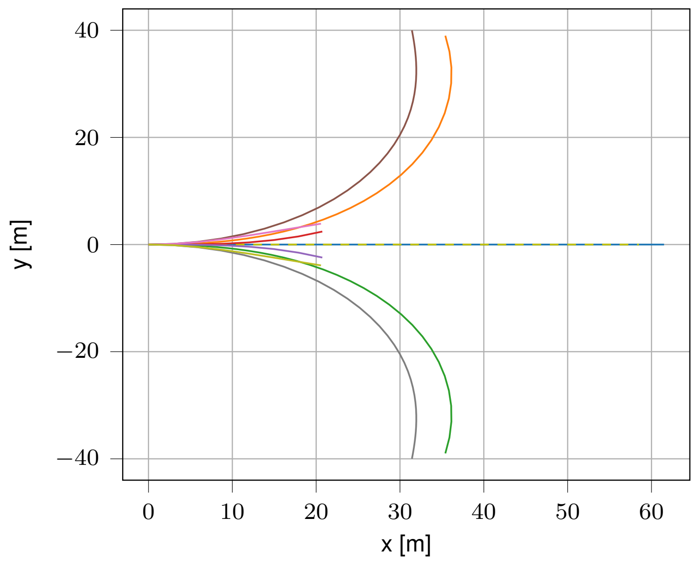


Figure 6-9: Displacements of the manoeuvre subset Q_M of the motion library Q. Accelerating and decelerating manoeuvres are longer than manoeuvres with only a change in the angular velocity.

is conducted at this point. The control input parameters acceleration a_k and azimuth rate $\dot{\Psi}$ are selected to account for the dynamics of a large and heavy UAV, such that

$$a_k \in \left(-4\frac{m}{s^2}, 4\frac{m}{s^2}\right) \text{ and } \\ \dot{\Psi} \in \left(-28.65\ ^\circ/s, -28.65\ ^\circ/s\right). \tag{6-10}$$

Uncertainties in the vehicle state and control input parameters from external disturbance and measurement errors are described in Tab. 6-7. Figures 6-10a) and b) illustrate the reach sets of synthesised control laws for the manoeuvre subset Q_M (see also Fig. 6-9) and trim state subset Q_T of the motion library Q. Transitions between the hover and moving states lead to significantly longer manoeuvres. The continuous coasting time τ of each trim state is discretised using

$$\tau^* = k \cdot 0.5s, \text{ with } k \in \mathbb{N}^+. \tag{6-11}$$

For increased readability, trim states with coasting time $\tau^* = 1.5s$ are displayed in Fig. 6-10b). Each trim state is therefore represented as a sequence of three times the respective control law.

6.2.2 Computation of Obstacle-Free Planning Spaces

In the simplest case, obstacle-free regions can be computed on the lowest flight level and applied to trajectory transitions on all higher flight levels. This comes at the sacrifice of a conservative assumption on the availability of obstacle-free spaces at greater altitudes. Alternatively, and to avoid limitations from overly conservative assumptions,

Table 6-7: Considered uncertainties on control inputs and vehicle state in the synthesis of control laws.

Parameter	Absolute Uncertainty	Unit
$\dot{v}$	± 0.2	$\frac{m}{s^2}$
$\dot{\Psi}$	± 1.45	$^\circ/s$
v	± 0.5	$\frac{m}{s}$
Ψ	± 1.15	$^\circ$
x_1	± 0.5	m
x_2	± 0.5	m

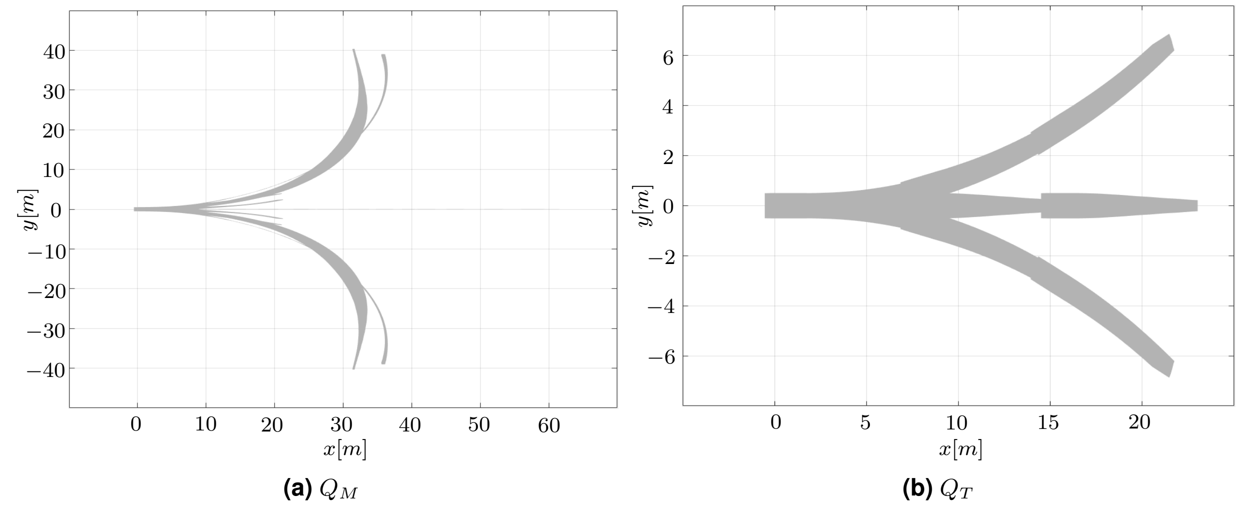


Figure 6-10: The motion library Q expressed in reach-set optimised control laws. (a): Controllers of the manoeuvre subset Q_M (see also Fig. 6-9). (b): Controllers of the trim state subset Q_T. Each trim state is plotted as a sequence of three primitives.

obstacle-free regions can be computed for each flight level independently and applied according to the actual flight altitude at each transition. The computation method for the individual flight level is identical in both cases. For this reason it is considered sufficient to validate the computation method on the lowest flight level at $h_{FL} = 150m$. In the i-th iteration, this implementation of the transition area search presented in section 5.7.1 updates the start point P_s at a distance $d = 100m$ from the previous region E_{i-1} .

Figure 6-11 displays 1524 obstacle-free regions computed based on CSFL trajectories of the initial trajectory tree's optimal solution presented in Fig. 6-8. To each transition area, the IDs and intersection intervals of intersecting trajectories are assigned. Since transition areas are obstacle-free by definition, transitions between trajectories can be planned at flight time inside each region with the above defined manoeuvre automaton, although the method is obstacle-unaware. It can be observed that many near-identical regions are computed in areas with a higher density of trajectories. This effect is particularly distinct in obstacle-dense environments, when start point candidates P_s on a

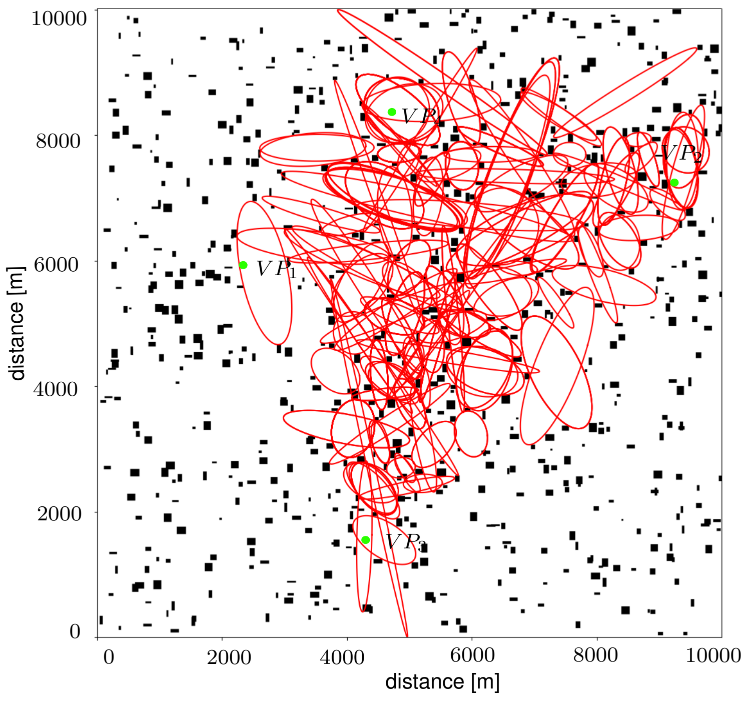


Figure 6-11: Obstacle-free ellipses in the obstacle environment at $150m$ above ground level. Transitions between two trajectories inside the same ellipse are safe, if the transition trajectory does not violate the ellipse's boundary. Transition regions are computed based on the CSFL planning solution in Fig. 6-8. Vertiports are indicated for orientation.

trajectory τ' are outside existing transition areas by only a small margin. In such cases, a new transition region is inflated around P_s, which is likely to be similar to the closest existing region. A filter function, identifying and pruning such clusters of areas, could help reduce the number of redundant transition regions significantly, with little impact on the accumulated transition area. The additional memory consumed by transition regions is small compared to a trajectory tree's memory consumption.

6.2.3 Online Transition Planning Between Preplanned Trajectories

Two transition regions E_0 and E_1 are selected from the set of obstacle-free regions shown in Fig. 6-11 to validate the online transition planning method presented in section 5.7.2. The validation planning method uses the optimal manoeuvre automaton defined in section 6.2.1. Adding this online planning capability to the trajectory database allows for a higher number of possible actions compared to an entirely pre-planned mission. The planning solution maintains deterministic properties. When combining the approach with the above verified control strategy, it guarantees that transitions between trajectories are maintained inside the respective region and that the transition path will end on a preplanned trajectory.

Figure 6-12 shows how the extension with an online planning method increases a pre-planned mission's action space. Simulation results of online transitions are depicted in green colour. Transitions are planned between randomly sampled states on pre-planned trajectories (black) inside E_0. An aircraft without an online planner that travels on a pre-planned trajectory can only divert to children of the current trajectory in equally predefined fork points. Once it has passed a fork point, it cannot return. The presented solutions show online planning results for $n = 10, n = 50$ and $n = 100$ trajectory transitions inside the same transition region in green colour and using a safety factor $k = 1$ in the feasibility criterion (5-35). Transitions between trajectories without a parent-child relation are not possible in the tree structure of the trajectory database. Transitions inside the second obstacle-free region E_1 are simulated and illustrated in Fig. 6-13. In contrast to Fig. 6-12, a safety factor $k = 1.5$ is used in the feasibility criterion (5-35). The comparison between planning results with different safety factors shows, that the margin towards the region boundary during manoeuvres initiating the transition grows with the safety factor. With both safety factors, transition trajectories may touch the region boundary in edge cases, but never violate the transition region.

Selected controller sequences of transition trajectories from Fig. 6-13a) with $n = 10$ are illustrated in Fig. 6-14. To account for the deviation of the controllers' reachable sets from the nominal manoeuvre trajectories, the safety factor $k > 1$ must be selected sufficiently large to ensure that the region's boundary is not violated by the disturbed system. From the illustration of manoeuvre and trim controllers in Fig. 6-10, it becomes evident that the expansion of reachable sets perpendicular to the direction of motion is small relative to the manoeuvre length. The reachable sets of straight-line primitives can hardly be observed in the provided controller sequences when drawn to scale. The selected safety factor $k = 1.5$ can be considered conservative in the given case and allows for safe transitions inside the obstacle-free region. As no controller reach set violates the transition area, all transitions are collision-free within the system's specified uncertainty. This, however, may change for different manoeuvre shapes and with greater uncertainty in the system. Larger safety factors k may be required in such cases.

Robustness of the method is provided by the selection of the geometric feasibility criterion, which ensures that a transition will not violate the region boundary. The required manoeuvre radius (see (5-34)) depends on the maximum turn manoeuvre defined in the manoeuvre automaton and is therefore vehicle-specific.

The *python* implementation of a greedy control policy on the manoeuvre automaton's precomputed optimal cost function completes a planning request even for long transitions in less than $1s$. Figure 6-15 plots execution times from 10000 transitions in the transition region E_1. It indicates that execution time scales linearly with the number of manoeuvres in a transition sequence. Execution times for shorter transitions are already considered sufficiently short to run the algorithm in a vehicle's guidance loop

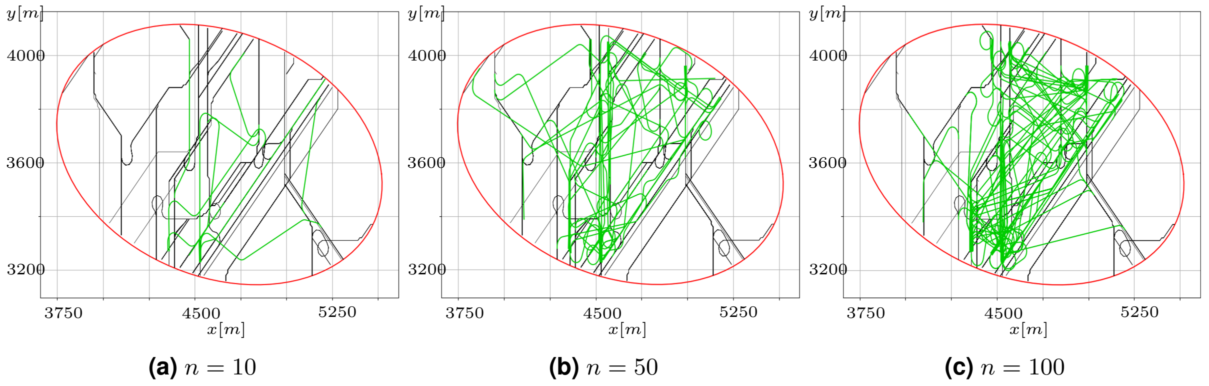


Figure 6-12: Simulation results of online transitions between randomly sampled states on pre-planned trajectories inside a transition region E_0 from Fig. 6-11, using a safety factor $k = 1$ in the feasibility criterion (5-35). n indicates the number of runs. Transitions may touch, but do not violate the region boundary.

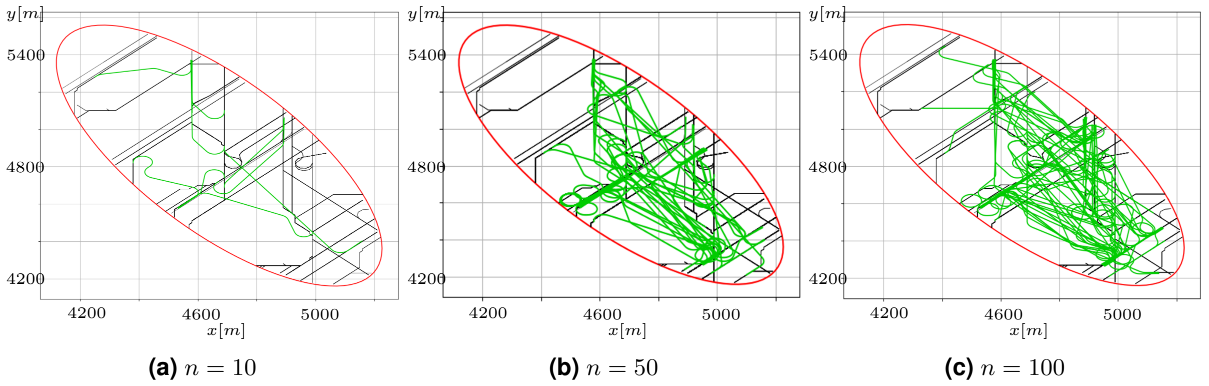


Figure 6-13: Simulation results of online transitions between randomly sampled states on pre-planned trajectories inside a transition region E_1 from Fig. 6-11, using a safety factor $k = 1.5$ in the feasibility criterion (5-35). n indicates the number of runs.

at flight time. Significantly quicker execution times are expected when the algorithm is translated to a compiled language and integrated into an embedded system.

6.2.4 Conclusion

The above sections show that manoeuvre-based planning methods allow to optimise motion primitives and cost functions in an offline computation effort, by which the planning problem's complexity at flight time is reduced significantly. Additional performance guarantees within a defined uncertainty envelope can be provided by synthesising control laws, which minimize the system's reachable sets during the execution of specific motion primitives. AROC continues to be updated with new control strategies (see e.g. [24] and [63]) to improve the performance of synthesised controllers and allow for larger disturbance envelopes. The required disk space for the controller library and the manoeuvre automaton's optimal cost function is within the range of few Gigabytes

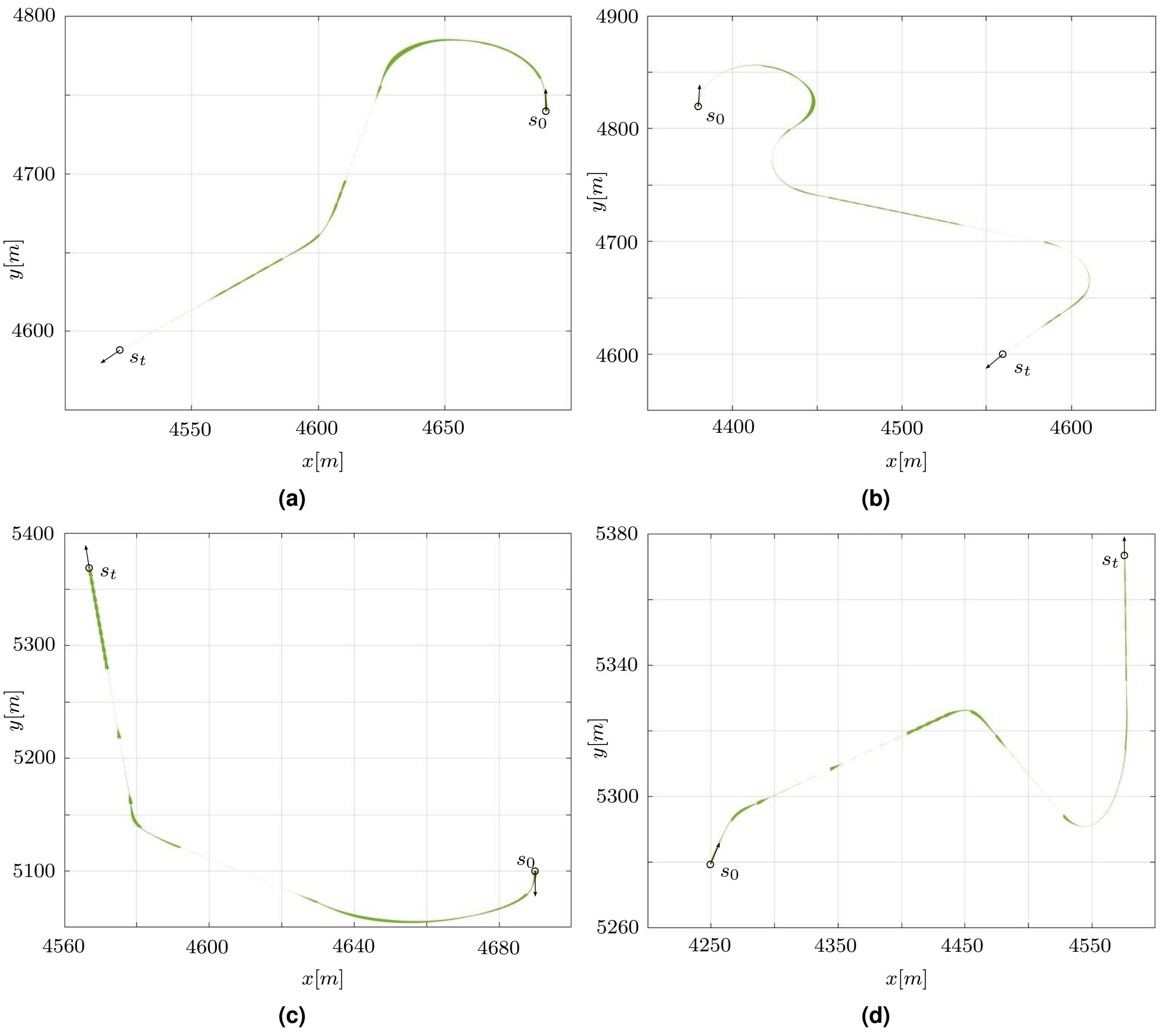


Figure 6-14: The reachable sets of controller sequences representing selected transition trajectories from Fig. 6-13a). A safety factor $k > 1$ must be selected such, that the maximum deviation of a controller reach set from the nominal manoeuvre is contained inside the transition region.

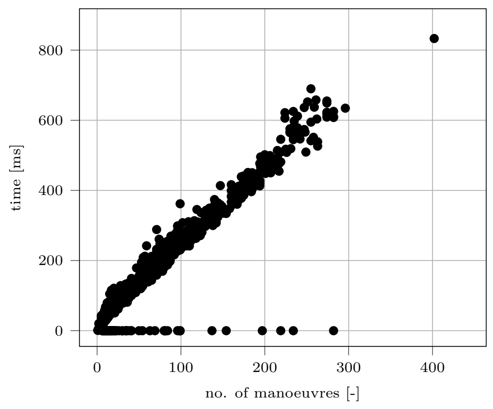


Figure 6-15: Computation time of online transitions over the number of manoeuvres in the transition trajectory from 10000 transition samples in E_1. Execution time increases linearly with the number of manoeuvres.

in the given case. More complex manoeuvre automata, potentially considering larger uncertainty in the system, may cause increased memory requirements. The presented results, however, imply that also larger automata with reasonable resolution of the vehicle performance can be computed and deployed within the limits of today's computing systems. The presented online planning method and related performance guarantees allow to define obstacle-free regions, inside which transitions between preplanned trajectories can be planned and executed at flight time. If these regions are convex, transitions can be contained inside the region with only the obstacle-unaware planning method.

It is shown that it is feasible and practical to compute such obstacle-free transition regions and the set of trajectories from a related database, which intersect with said regions. In cases, where a new region's start point candidate lies outside an existing region by small margins, the presented method can be further optimised to limit the number of near-redundant transition areas. A geometric criterion is developed and derived from the vehicle-specific motion library, which ensures that transition trajectories, which are planned and executed at flight time, do not violate the current transition region. It is further reasoned how the definition of appropriate values for the safety factor on the distance between a transition's start point and the region boundary help contain the transition inside the respective region even under external disturbance within the limits of the uncertainty envelope.

All algorithms executed at runtime are deterministic and allow to compute a worst case execution time. The results of the offline computation for the optimal cost function and controller libraries can be verified and released by a competent authority or the USSP before take-off. In contrast to the cases with the exclusive use of a trajectory database, not all possible flight paths during the mission are known exactly, when transition trajectories are planned online. However, the airspace volumes are known, within which a transition is executed. Hence, strategic deconfliction between several aircraft by a USSP can still be handled on the basis of predetermined flight plans.

6.3 Validation in a Realistic Urban Scenario

In order to validate the developed motion planning framework in the intended target environment, an urban air taxi mission over congested terrain is considered in the following. The mission is designed to be subject to the certified UAS category of EASA's future UAS regulation.

6.3.1 The Urban Application Scenario

The urban validation scenario (previously presented in [48]) is set in London, UK, and implements a UAM mission between Victoria station, where the express train connection from Gatwick airport terminates, and Cannon Street station in the City of London. Two alternate vertiports on the Eastern banks of Thames river between Lambeth and Westminster Bridge and on a roof top near Blackfriars station are defined for use in the case that a contingency event occurs. A list of vertiport coordinates and altitudes AGL is provided in Tab. 6-8. It is assumed that wind conditions are steady from a northern direction $\alpha_w = 0°$ and of speed $v_w = 8\frac{m}{s}$ in the considered environment . These parameters are selected randomly from the vehicle's operating conditions. Further, the operator is assumed to have been given clearance to operate a large UAV in the special use air spaces R157, R158 and R160 in the lower London air space. No temporary no-fly zones and NOTAMs were active in the relevant air space at the time of the risk map computation. The considered vehicle is a large UAV operated under the EASA certified UAS category with characteristic properties listed in Tab. 6-9. Planner settings and relevant parameters respect Tab. 6-1 unless explicitly stated otherwise in Tab. 6-11.

Table 6-8: Vertiport WGS84-coordinates of the urban scenario.

Vertiport	Coordinates	Altitude AGL
1	N$51.493498°$ E$-0.146207°$	$86m$
2	N$51.511153°$ E$-0.090315°$	$59m$
3	N$51.496942°$ E$-0.117017°$	$21m$
4	N$51.511123°$ E$-0.101090°$	$46m$

Table 6-9: Urban scenario UAV characteristics.

Parameter	Representation	Value
rotor radius	R	$5m$
mass	m	$500kg$
max. air speed	$V_{A,max}$	$20\frac{m}{s}$

6.3.2 Generation of Risk Maps

A three-dimensional risk map of the flight area is computed following the approach introduced in chapter 4. Individual risk layers are merged using the described *max-value* function and extrapolated vertically according to the linear degression model (4-1). The residual risk altitude is selected as $d_{z,max} = 100m$ AGL. The risk model incorporates risk layers from structures L_{struct}, roads L_{road}, land-use L_{land}, air space L_{air} and ground traffic L_{traf} using the application programming interfaces (API) of HERE[1], AirMap[2] and Overpass[3] for OpenStreetMap to obtain and process the most recent data sets. Since permanent air space restrictions over the city of London are lifted for this scenario and no temporary restrictions were active at the time of risk map computation, air space data has no effect on the resulting risk map.

The risk model resolution is defined to correspond to half the characteristic vehicle dimension $\frac{1}{2}D = R = 5m$ of Tab. 6-9. Two composites of static and dynamic risk $C_{st}(\vec{x})$ and $C_{dyn}(\vec{x})$ are generated from the individual risk layers (see (4-4)) and merge risk layers following

$$\begin{aligned} C_{st}(\vec{x}) &= max(w_{l,struct}L_{struct}(\vec{x}), w_{l,road}L_{road}(\vec{x}), w_{l,land}L_{land}(\vec{x})) \\ C_{dyn}(\vec{x}) &= w_{l,traf}L_{traf}(\vec{x}) \end{aligned} \tag{6-12}$$

and with the position $\vec{x} \in \mathbb{Q}^3$. Risk layer weights $w_{l,i}$, determining the weight of each risk type in the composite, are selected according to Tab. 6-10 and reflect the vehicle and mission properties in the risk modelling. Based on the weight, size and certification background of the considered vehicle, risk layer weights for risk associated with ground movements are reduced and the hazard class for public spaces lowered. On the other hand, the hazard class of structures is increased due to the effect a potential physical impact of a large and heavy vehicle may have on these structures. Further considerations on the selection of appropriate risk layer weights can be found in [50] or may be derived from e.g. [20] or [53]. Static and dynamic composites are weighted equally in the final risk map, such that

$$C_{fin}(\vec{x}) = max(w_{c,st}C_{st}(\vec{x}), w_{c,dyn}C_{dyn}(\vec{x})) \tag{6-13}$$

[1]https://developer.here.com/web/
[2]https://developers.airmap.com/web/
[3]http://overpass-api.de/web/

holds with weights $w_{c,st} = w_{c,dyn} = 1$. All weight factors are selected to represent reasonable values for the intended scenario. Different values may be selected during actual operations based on extended modelling or qualitative reasoning by the USSP (see e.g. [50] for reference).

Table 6-10: Risk layer weight factors.

Weight Factor	Value
$w_{l,struct}$	1.0
$w_{l,road}$	0.7
$w_{l,land}$	0.7
$w_{l,traf}$	0.7

Cross sections of the flight area's resulting final risk map C_{fin} at $40m, 60m, 80m$ and $100m$ AGL are shown in Fig. 6-16. Physical obstacles are displayed as areas of risk value 1. It can be observed that most risk features are related to structural obstacles and railway tracks with high-voltage power lines. With increasing altitude, the number of critical risk items of value 1 decreases significantly.

6.3.3 Simplification of the Urban Configuration Space

The nominal flight surface connecting vertiports VP_1 and VP_2 (see Tab. 6-8) is computed at the cruise altitude $h_{cruise} = 150m$ following the process described in chapter 5.4. Three contingency flight levels are defined around the cruise altitude at $100m$, $150m$ and $200m$ and contingency flight surfaces computed for each vertiport and flight level. Figure 6-17a) illustrates the nominal flight surface $\mathcal{S}^*$ between VP_1 and VP_2 in blue colour in the digital surface model of the flight area. Contingency flight surfaces $\mathcal{S}^*_{c,100}$, $\mathcal{S}^*_{c,150}$ and $\mathcal{S}^*_{c,200}$ and approach funnels around each vertiport are depicted in light gray colour. While the two alternate vertiports VP_3 and VP_4 as well as the take-off vertiport VP_1 can serve as CSFL sites in case of a contingency event, once abandoned, the initial destination vertiport VP_2 is no longer a valid alternate landing site. Hence, no contingency flight surfaces and navigation functions need to be computed for this vertiport.

Regardless of the vertiport type, the approach funnels of all vertiports end at a vertical distance above the vertiport. This means that the vertiport can only be reached or left with a final vertical descend or initial climb manoeuvre. Otherwise boundary conditions would be violated on the funnel surface. Alternatively, steeper path angles γ during climb and descend phases can be defined to reduce the length of or even avoid vertical manoeuvres. Rulemaking activities for vertiport take-off and landing procedures are currently under development. Reference [57] documents the current draft state of the related proposed MOC for SC-VTOL.

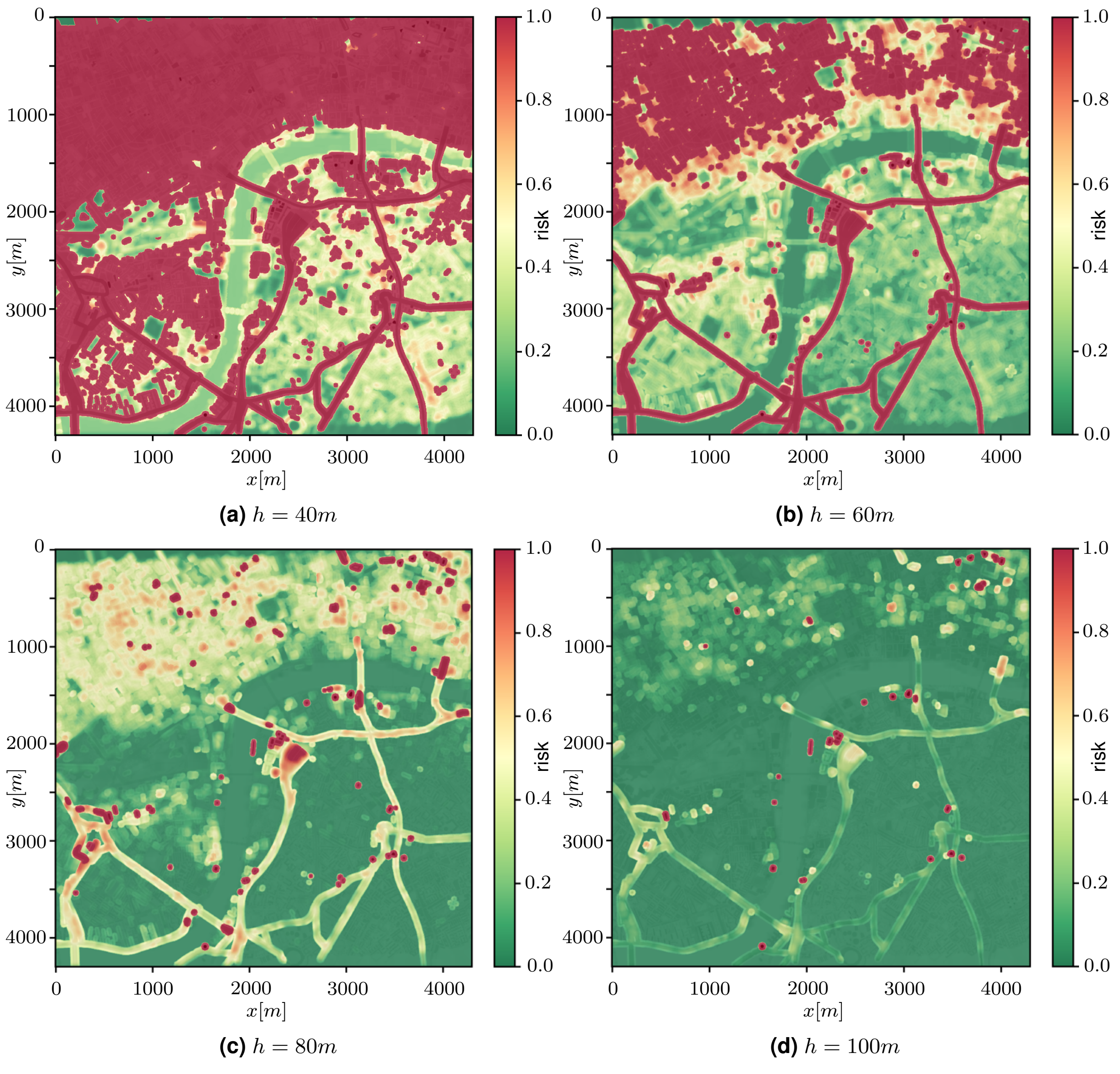


Figure 6-16: Cross sections of the urban scenario risk map at different altitudes above ground level. The risk values at $h = 100m$ are defined as the residual risk value for the vertical extrapolation of ground risk values.

Figure 6-17b) shows the reduced complexity of the two dimensional planning problem on the nominal planning surface compared to the complete obstacle set at ground level. The ground obstacle map is drawn in greyscales, whereas obstacles on the planning surface are depicted in black. Overall, more than $1.5 \cdot 10^4$ structural obstacles exist in the scenario configuration space (see also Tab. 6-11), of which only 29 remain on the planning surface $\mathcal{S}$. This corresponds to a reduction of the number of obstacles n_{obs} by three orders of magnitude and one spatial dimension.

Table 6-11: Updated flight planner settings in the urban scenario.

Parameter	Representation	Value	Unit
cruise altitude	h_{cruise}	150	m AGL
flight levels	FL_i	$100, 150, 200$	m AGL
branch factor	N	2	-
contingency planning interval	Δt_C	30	s
path length restriction factor	k	2	-
efficiency weight exponent	w_e	$\frac{2}{3}$	-
risk weight exponent	w_r	$\frac{1}{3}$	-
number of ground obstacles	n_{obs}	15303	-
wind speed	v_w	8	$\frac{m}{s}$
wind direction	α_w	0	rad

To prepare the consecutive mission planning step, a roadmap graph is computed on the nominal planning surface $\mathcal{S}$ and navigation functions computed for each contingency flight level and vertiport. The procedure follows the steps carried out in chapter 6.1.1 and Fig. 6-3 and Fig. 6-7. Due to the dimensions of the planning scenario, the CSFL radius R_{CSFL} has no impact on the roadmap generation.

6.3.3.1 Planning in the Simplified Configuration Space

The same planning process as to the artificial planning scenario in section 6.1 is applied to the simplified configuration space of the urban scenario. A trajectory database containing a tree of trajectories to the destination vertiport and additional trajectories to alternate vertiports in the scope of CSFL is computed. In contrast to the earlier scenario, risk is considered in the planner's cost function following (5-6) and (6-4), such that an edge's e_i penalised total cost $C'_{tot,i}$ is expressed as

$$C'_{tot,i} = (1 + B_p w_{p,i} + B_{rj} w_{rj,i}) C_{tot,i} = (1 + B_p w_{p,i} + B_{rj} w_{rj,i}) \cdot (p_w L_e)^{w_e} \cdot {C_r}^{w_r}, \quad \text{(6-14)}$$

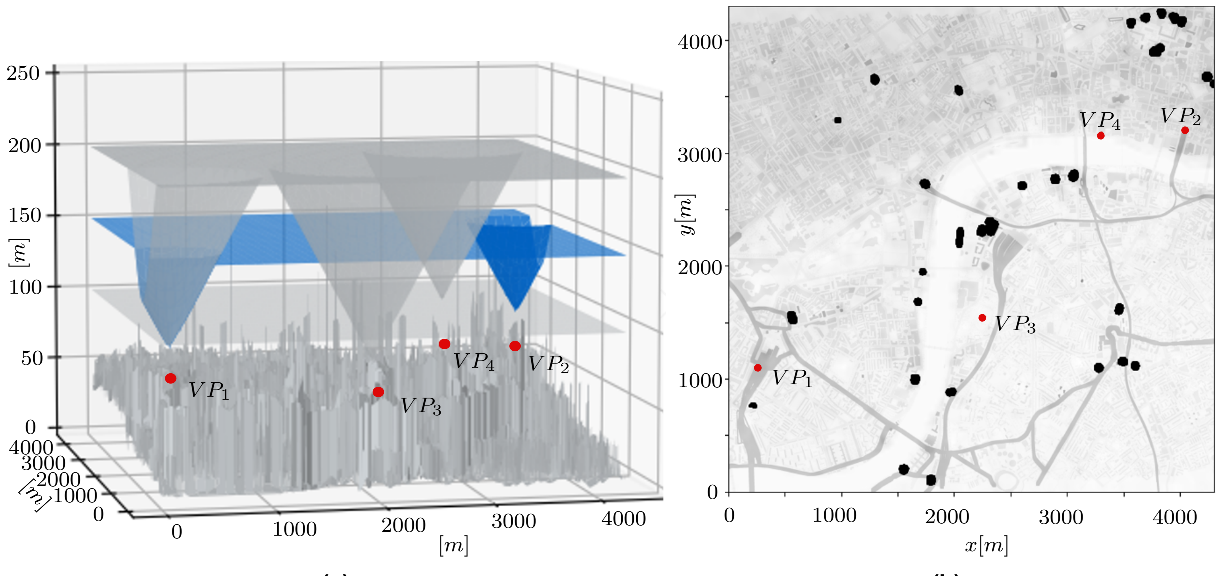


Figure 6-17: (a): Nominal and contingency flight surfaces of the urban scenario. Vertiports are indicated for orientation. (b): A comparison of the ground obstacle map (grey) and planning surface obstacle map (black). Vertiports are marked red for reference.

with the efficiency and risk weight exponents $w_e = \frac{2}{3}$ and $w_r = \frac{1}{3}$, respectively (see also Tab. 6-11). In actual operations, these parameters can be used to tune a mission's risk aversion and efficiency. In agreement with Fig. 5-7, the risk cost of an edge is defined as the highest risk value found in a cylindrical volume of radius R along the edge.

Using the risk maps generated in section 6.3.2 and the cost function (6-14), a tree of trajectories between the departure vertiport VP_1 and destination VP_2 is computed on the nominal planning surface $\mathcal{S}$. With a branch factor $N = 2$, contingency planning interval $\Delta t_C = 30s$ and path length restriction factor $k = 2$ (see also Tab. 6-11), the planning parameters are selected to produce a trajectory database, which can be visualised reasonably.

Trajectories leading to the destination vertiport VP_2 are plotted onto a cross section of the risk map at $h = 150m$ in Fig. 6-18a). With the selected parameter setting, a trajectory tree $\mathcal{T}_{2,2}$ of 102 trajectories is generated on the planning surface's roadmap graph. When looking at the directions of approaches to the destination vertiport, it can be observed that with increasing penalty weights approaches from a southern direction with higher risk become attractive. Reciprocal values of the distance to the closest obstacle over each trajectory's progress are plotted in Fig. 6-18b). The solution complies with the defined horizontal obstacle clearance $c_{l,min} = 50m$.

Figure 6-19 illustrates the CSFL solution for the cost-optimal path τ'_{opt} on the flight area's satellite map[4]. 283 trajectories to alternate vertiports are identified on the navi-

[4]https://earth.google.com/web/

gation functions of the respective contingency flight surfaces. Since the trajectory τ'_{opt} runs close to the alternate vertiports, comparably few CSFL trajectories are required to achieve the desired database time resolution Δt_C. Appendix A provides further illustrations of the planning solution on three-dimensional satellite maps. Contingency trajectories for CSFL are generated for all trajectories in $\mathcal{T}_{2,2}$, such that a total of 41823 trajectories to alternate landing sites exist in the final database. A summary of database parameters is provided in Tab. 6-12. Compliance of the CSFL planning solution with the horizontal obstacle clearance is validated in Fig. 6-19b).

Convex regions for online transitions between preplanned trajectories are computed on the contingency planning surface $\mathcal{S}_{c,150}$ closest to the nominal planning surface and plotted on the surface's obstacle map in Fig. 6-20. Similar to Fig. 6-11, many of the computed transition regions are near-redundant, emphasising the need to extend the computation method with a redundancy filter. The protected online planning problem is fully defined by the trajectory database and set of transition regions. It is therefore treated and validated adequately in section 6.2 and does not require repeated validation at this point.

Table 6-12: Properties of the urban scenario planning solution.

Parameter	Value
trajectories to destination	102
CSFL trajectories	41823
transition regions	231
database size	1.72 GB

6.3.4 Conclusion

The urban planning scenario demonstrates, that the developed framework and motion planning methods work as intended in the target environment. The generated databases are sized reasonably, such that an aircraft operating on a preplanned mission has many options for action even in cases of multiple consecutive contingency events. The USSP or competent authority on the other hand, maintains exact knowledge about the aircraft's decision space. When compared to the artificial planning scenario in section 6.1, the simplification of the configuration space leads to much sparser obstacle environments on the respective planning surfaces in the urban scenario, indicating that the method is well suited for environments, where the distribution of obstacle heights is skewed towards lower structures. For metropolitan areas, this is typically the case.

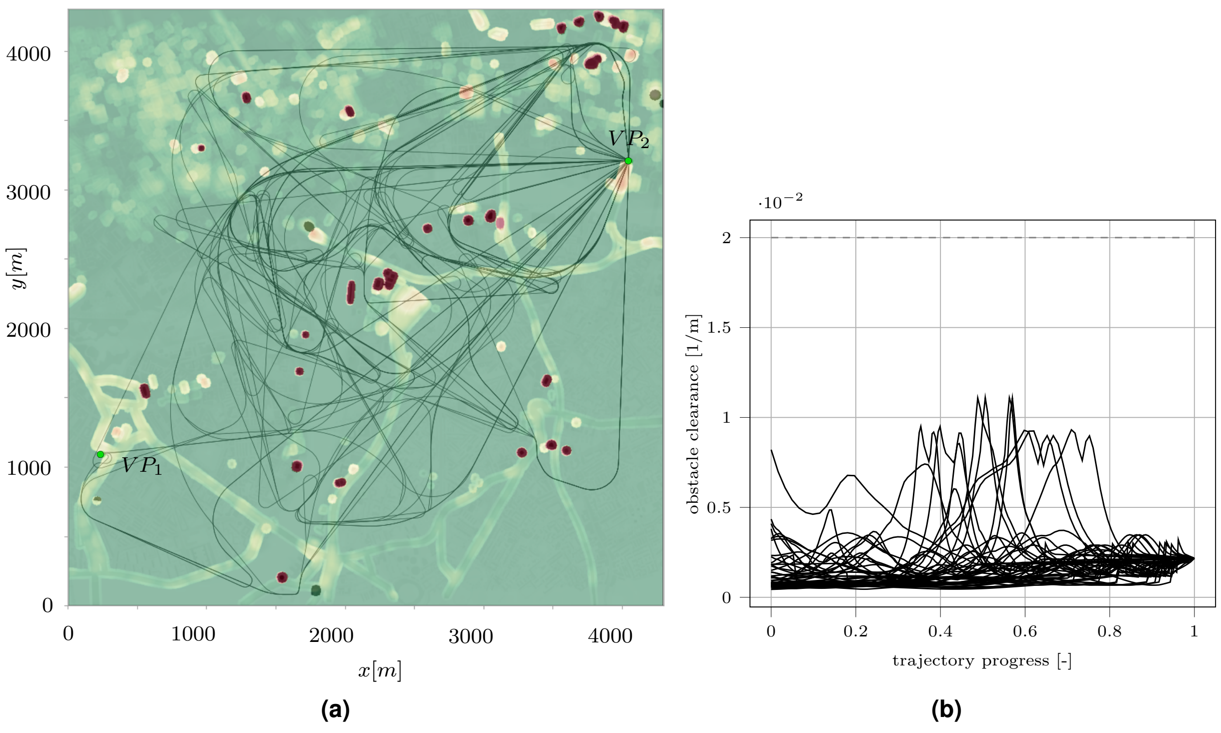


Figure 6-18: (a): Urban planning results to destination vertiport on a cross section of the risk map at $h = 150m$. (b): The reciprocal of obstacle clearance of trajectories to destination.

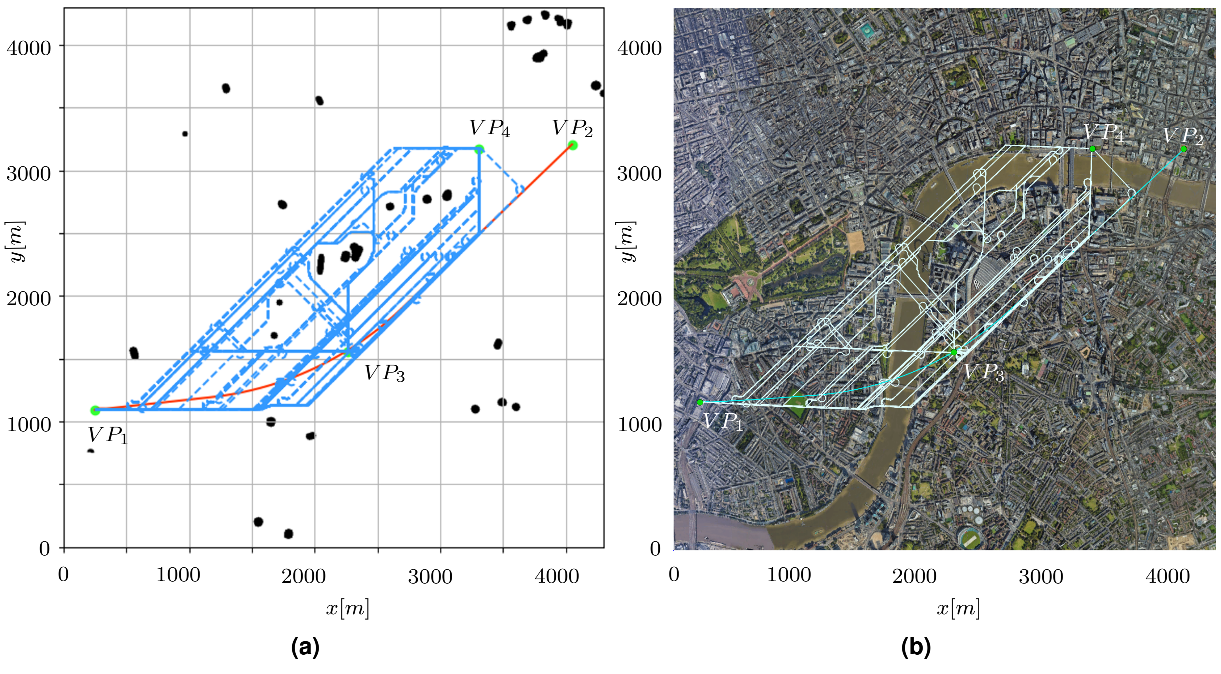
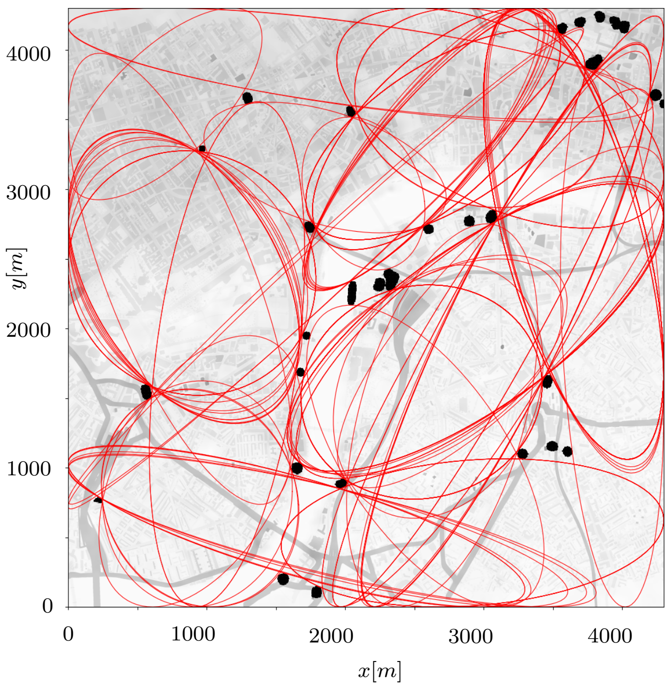
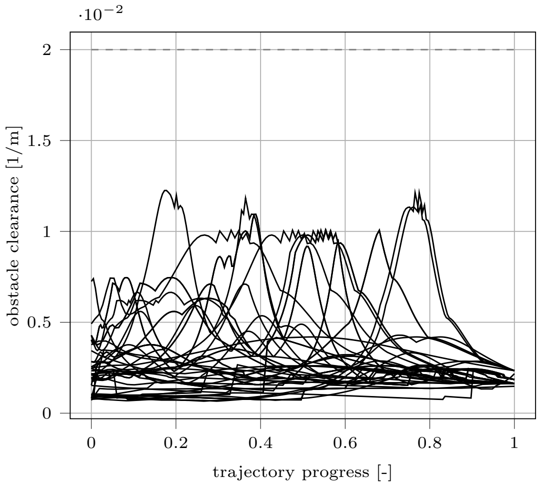


Figure 6-19: (a): CSFL planning solution for the optimal trajectory τ'_{opt} between VP_1 and VP_2 (red, compare to Fig. 6-18) on the obstacle map of the contingency planning surface $\mathcal{S}_{c,150}$. (b): CSFL planning solution for the optimal trajectory τ'_{opt} on the satellite map[4] of the planning region.



(a)



(b)

Figure 6-20: (a):Transition regions of the urban planning scenario, computed on the obstacle map of the contingency planning surface $\mathcal{S}_{c,150}$. The ground obstacle map is added for orientation. (b): Obstacle clearance of CSFL trajectories in Fig. 6-19.

As a result of the reduced obstacle-density, obstacle-free transitions regions also grow in size. This leads to more flexibility for a single aircraft to switch between trajectories of the database, however, comes at the cost of reduced USSP awareness of the vehicle state and control over the air space. Consequently, the air space must be operated more conservatively. Additional restrictions to limit the dimension of transition regions may therefore be considered in operational environments with few obstacles but dense air traffic.

The computation of planning surfaces in Fig. 6-17 is evidence that vertiports are difficult to approach and depart from at the defined path angle and obstacle clearances, unless positioned on an exposed structure. This creates the need for vertical climb and descend manoeuvres to connect the vertiport to its approach funnel. The issue is currently being addressed in rule-making activities for vertiport operation procedures. The generation of flight surfaces should be adjusted to account for these procedures, once available.

It is shown, how risk models are integrated into the motion planning process and exploited to extend the notion of qualitative risk assessment in the EASA specific category towards quantitative risk models of high resolution in the certified UAS category. Beyond the consideration of additional input parameters, such as weather, the fidelity of existing risk models can be increased further and redundant data sources introduced.

Weight factors in both the risk modelling and consideration of risk in the planner's cost function can be used to tune the planning solution to reflect risk-awareness and efficiency to the operator's preference.

7 Summary and Discussion

The planning methods developed in this dissertation aim to enable contingency-aware motion planning for UAVs under coming EASA regulations for operations over congested terrain and other environments, in which safety is critical. Considerations on the economic viability of envisioned UAM use cases demand further, that the framework shall enable dense operations of many vehicles in the same airspace, which is controlled by a USSP. This adds the need for a shared airspace structure between missions and deconfliction capabilities within the scope of available U-Space services.

Results from both validation scenarios of chapter 6 show that a significant simplification of the planning task can be achieved in the obstacle environment and search space dimension, when an appropriate preprocessing of the configuration space is applied. The rule-based approach further enforces any solution in the remaining valid configuration space to be feasible under the respective set of rules and the considered airspace structure. The actual planning task is reduced from finding a feasible solution in the configuration space to finding an optimal solution according to a defined cost function in the feasible solution space. According to contributions C-1 and C-4, the system is designed such that a database of rule-compliant trajectories and flight levels enables contingency handling, while the requirement to fly on predetermined routes is respected. Each partial solution can be updated independent from higher layers, meaning that e.g. the planning surface need not be recomputed, when the search graph is updated. Partial results can be stored and reused until environmental changes require an update, hence, reducing the time to respond to incoming planning queries. The planning framework was presented to an EASA panel for the evaluation of UAM automation capabilities without major concerns being raised by present domain experts.

The concept to simplify a complex configuration space and the motion planning task in this configuration space based on a defined set of rules according to contribution C-2 requires a flight area's obstacle environment to be mostly known and contain countably few, predetermined vertiports. The advantages of the method prevail in scenarios, where large portions of the risk- and obstacle environment change slowly over time. It is therefore particularly suitable for operational concepts in near-static environments, in which several vehicles commute frequently between the same set of vertiports. This characteristic is met by the design of intended air taxi (or UAM) networks in metropolitan areas as well as cargo or supply routes in rural areas and supported by the road lane concept of contribution C-3. Smaller changes in the planning environment's configuration space or temporarily restricted areas can be compensated by alternative routes already available in the planning solution (see contribution C-4) and without requiring a recalculation of the trajectory database or flight planning surface immediately.

In largely dynamic environments, however, the developed planning system loses this property. The advantage over conventional approaches is then reduced to the verifiability of the solution space, which is discretised by the trajectory database.

Further, the developed motion planning method and operational concept require the development and implementation of a logical component that selects the best possible alternative action from the pre-calculated trajectory and manoeuvre database based on the current aircraft and environmental state. The development of this contingency management module is not in scope of this dissertation. A conceptual development and prototypical implementation of a contingency management component that is specific to the approach and trajectory databases from this work is described in [76]. The reference also provides a proof of concept using a simulated operation of the module based on a simple mission. In addition to the trajectory selection, the functionality of loiter manoeuvres and primitives to change flight levels can only be demonstrated comprehensively when the associated contingency management module is available. Multiple flight levels of a trajectory database are shown in Appendix A. Outside the approach funnels of vertiports, an aircraft can move freely between different flight levels within the limitations of the obstacle environment. By implementing different prioritisation schemes, the aircraft's response to a contingency situation can be influenced significantly and changed to an operator's preference even when using the same database. A potential prioritisation sequence is described in Section 5.6.2.3.

Another aspect, which is not investigated in detail, is the effect of selected cost functions and risk models on the planning result. Results published separately in [50] indicate that a variation of vehicle parameters in the same risk model already has a significant effect on the most cost-efficient trajectory. Based on these findings and the observed impact of penalty terms, when the solution space is scaled with different planning parameters in Section 6.1.1.2, it is expected that the planning solution will react sensitively to changes in the selected cost function. Risk accounts for a significant portion of the path cost. For this reason, precise models of the environment and the design of appropriate risk models are considered to be critical for the practical implementation of the developed planning method. Insights into this related field of research and guidance regarding the design of risk models are provided by e.g. [13], [53] and [77].

Globally sub-optimal performance and efficiency that are caused by the limitations imposed on the configuration space are deliberately accepted and subordinate to the notion of safety. Trajectory databases that result from the reduced configuration space are designed to display great structural similarity to existing air space structures. This will facilitate the orientation and adaption to the new air space type for existing pilots. Additionally, existing rules of the air can be integrated into the operation of UAM networks, which allows potentially shared operations between aircraft and pilots of the UAM and General Aviation categories in VLL airspace. The developed methods may also be used to support early stages of UAM adaption by displaying flight paths to a human pilot on dedicated displays as so-called tunnels-in-the-sky. The design of efficient pilot assistant displays is a separate research topic and addressed in e.g. [30] and [45]. Specifically in early scenarios of manned UAM operations, when the availability of suitable pilots poses a potential thread to scale air taxi operations, it is important to keep entry barriers for existing commercial and recreational pilots as low as possible. Once flight operations in urban VLL airspace have been automated to a certain degree, this requirement is expected to become less significant.

When the operation of aircraft is automated and in conjunction with the earlier described contingency management module, the total quantity of trajectories that are reachable during a mission can be verified and further restricted if necessary. If trajectories in the offline generated planning solution intersect unwanted terrain or contain undesirable flight states, these trajectories can be removed by blocking them for a specific mission or deleting them from the database. The contingency management module may further limit the subset of actions available from the database at each point in time based on the current mission and aircraft state. In the event of deteriorated vehicle performance, for instance, certain trajectories that can no longer be flown with the reduced performance envelope, may be blocked for the remainder of the mission. Similar actions can be taken for the online transition planner's motion library (see contribution C-5), of which certain manoeuvres and trim states may be made unavailable for planning if a change in the vehicle performance occurs. In this case, it must be considered that the geometric feasibility criterion for transition planning inside convex regions may no longer be valid or require to be adjusted to the applicable subset of the motion library.

According to the current status of SC-VTOL ([68]) and the EASA concept for regulation of UAS in the certified UAS category ([59]), the criteria that a competent authority will apply to define or review the number and types of predefined actions in a database, required to execute a mission safely remain unaddressed. In the simplest scenario, the complexity of a mission and air traffic density that can be handled with a given trajectory database may be estimated conservatively and based on qualitative criteria. However, pursuing the objective to maximize the degree of utilisation in the UAM network, this approach is unfavourable and should be replaced with more accurate, quantitative methods. From a scientific perspective, comprehensive air space simulations and an analysis of potential edge cases will be required to derive evaluation criteria and metrics, which can be used to put database properties and achievable mission complexity into relation. Assessing the impact of transition regions, within which online transitions between preplanned trajectories are allowed, may prove particularly challenging in this context.

Based on the current status of regulatory and supporting documents, it can be expected that early implementations of this planning framework may have to refrain from enabling online planning capabilities and will be limited to database-centric planning methods only. It may also be discussed to which extend locally constraint, partial solutions of the offline solution can be recomputed online. Rerunning e.g. the near-vertiport planner during flight to re-evaluate the final approach within a constant radius around the destination vertiport would enable the aircraft to approach a vertiport on an updated landing trajectory, that takes the actual conditions at the time of arrival into account. Should no solution be found during the online re-computation, the offline solution can serve as a verified fallback. With extensions to the planning framework added in the future and based on the available computational power, it may become favourable to re-run also other flight phase-specific planning methods during flight.

The *Python* implementation that is used in the validation chapter of this dissertation serves the purpose to demonstrate the presented methods and investigate the plausibility of obtained results. It is not optimized for quick execution or minimizing runtime. Parts of the implementation, which handle the offline processing of the planning environment and pre-computation of trajectory databases, are not strictly required to be translated into a compiled language. Algorithms, however, which compute or select trajectories during flight, must be translated into an avionics programming language (preferably C or C++) in an effort to optimize runtime and prepare a potential future certification program. A significant reduction in execution time over the *Python* implementation is expected when online components are translated into C/C++ and implemented on embedded hardware. This applies in particular to the online computation of transition trajectories (see also Fig. 6-15).

Due to the continuous development of regulations and MOCs for the operation of UAS, the development of planning methods for this new field of aviation is inevitably subject to regulatory uncertainty. While the development of standards for the manned operation of eVTOLs and the operation of UAVs under the Specific Category is well advanced and can be considered mature in its basic features, a mature set of regulations for the automated operation of eVTOLs over urban terrain is still to be developed. This dissertation therefore relies on aspects of the SC-VTOL that are expected to be carried into the new certified UAS category as well as on draft versions that come out of the certified UAS category's rulemaking task and were available at the time of development. It cannot be excluded that underlying assumptions to this work will change or even be declared void as the rulemaking task progresses. Individual planning methods will therefore have to be updated continuously to keep up with the rulemaking progress (see e.g. vertiport take-off procedures in [57]). Nevertheless, it is expected that the methods and general approaches presented here, will (with adjustments) remain valid and applicable under the final version of the EASA concept for regulation of certified UAS operations.

8 Outlook

Beyond updates of individual modules, which are made necessary by the ongoing development of rulemaking activities, and improvements of planning algorithms discussed in chapter 6, the presented research should be continued and extended in three distinct categories to enable meaningful implementations of rule-based motion planning in UAM networks.

First and foremost, a contingency management module should be developed that acts as a decision maker during flight. It may monitor the current aircraft state and different environmental and mission-specific parameters, based on which it selects the appropriate action from a provided database at each point in time. Early implementations may include interactions with the responsible USSP for approval until entirely automated vehicle operations are enabled. Since such a component will be critical for the operational safety and is expected to face high certification barriers, the use of a formal specification language in a final implementation could be beneficial and may be considered in the development process.

Another aspect already discussed in chapter 7, is the need for reliable and accurate environment and risk models. To avoid that safety is affected by mapping errors, efforts should be made to enforce high data quality and potentially certify maps for specific UAM scenarios. However, efforts to obtain mapping data of high quality over the course of this dissertation could not identify academic or commercially available data sets of the resolution, precision and accuracy that would be required for commercial low-altitude operations in cluttered terrain. In the short term, it may therefore be more promising to achieve data quality through redundancy and overlay maps from multiple sources to identify map sections of high and low confidence. Based on the improved environment representation, more advanced risk models of higher fidelity can be developed and integrated into the planning framework. Such risk models could combine different raw data types to derive new information and incorporate models to evaluate the vehicle splash pattern based on the actual vehicle state and expected environmental conditions. This would allow the computation of highly accurate, risk-minimal flight corridors or trajectories, however, at the cost of a closer coupling between risk modelling and motion planning processes.

Lastly, metrics and criteria are required, to evaluate databases and their fitness to conduct certain missions under a set of boundary conditions. This issue may be addressed in future MOCs that are developed to support the primary certification specification. Unless restrictive assumptions on the type and environment of a mission are made to evaluate a database's fitness qualitatively, comprehensive air space simulations with varying parameter sets may be required to derive appropriate quantitative evaluation guidelines.

References

[1] Ancel, Ersin et al., “Real-time risk assessment framework for unmanned aircraft system (UAS) traffic management (UTM)”, in: *17th aiaa aviation technology, integration, and operations conference*, 2017, p. 3273.

[2] *Annex F: SORA Ground Risk Class Justification*, Joint Authorities for Rulemaking of Unmanned Systems, Quantitative Methods Group in WG 6, 2021.

[3] Askari, A et al., “A new approach in UAV path planning using Bezier–Dubins continuous curvature path”, in: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 230.6 (2016), pp. 1103–1113.

[4] Ayhan, Bulent et al., “Path planning for UAVs with engine failure in the presence of winds”, in: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2018, pp. 3788–3794.

[5] Bansal, Somil, Chen, Mo, and Tomlin, Claire J, “Safe and Resilient Multi-vehicle Trajectory Planning Under Adversarial Intruder”, in: *arXiv preprint arXiv:1711.02540* (2017).

[6] Bellman, R., *Dynamic Programming*, Princeton Landmarks in Mathematics, Princeton University Press, 1957, ISBN: 9780691146683.

[7] Boeuf, Alexandre, “Kinodynamic motion planning for quadrotor-like aerial robots”, PhD thesis, Institut national polytechnique de Toulouse (INPT), 2017, chap. 3.

[8] Bosson, Christabelle and Lauderdale, Todd A, “Simulation evaluations of an autonomous urban air mobility network management and separation service”, in: *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3365.

[9] Cantalloube, Félicien, *Manoeuvre library generation for the motion planning of an E-VTOL aircraft*, Master Thesis, ISAE-SUPAERO and Volocopter GmbH, 2019.

[10] *CS-23 Normal, Utility, Aerobatic and Commuter Aeroplanes*, European Aviation Safety Agency.

[11] *CS-27 Small Rotorcraft*, European Aviation Safety Agency.

[12] Dadkhah, Navid and Mettler, Berenice, “Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance”, in: *Journal of Intelligent & Robotic Systems* 65.1 (2012), pp. 233–246.

[13] Dalamagkidis, Konstantinos, Valavanis, Kimon P, and Piegl, Les A, “Evaluating the risk of unmanned aircraft ground impacts”, in: *2008 16th mediterranean conference on control and automation*, IEEE, 2008, pp. 709–716.

[14] De Berg, Mark et al., *Computational Geometry*, Springer-Verlag Berlin Heidelberg, 2008, ISBN: 978-3-540-77974-2.

[15] Deits, Robin and Tedrake, Russ, “Computing large convex regions of obstacle-free space through semidefinite programming”, in: *Algorithmic foundations of robotics XI*, Springer, 2015, pp. 109–124.

[16] Dijkstra, Edsger W et al., “A note on two problems in connexion with graphs”, in: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[17] Dubins, Lester E, “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents”, in: *American Journal of mathematics* 79.3 (1957), pp. 497–516.

[18] *EASA Air Operations: Annex IV (Part-CAT)*, European Aviation Safety Agency.

[19] *Easy Access Rules for Airworthiness and Environmental Certification (Regulation (EU) No 748/2012)*, European Aviation Safety Agency, 2021.

[20] *Easy Access Rules for Unmanned Aircraft Systems (Regulation (EU) 2019/947 and Regulation (EU) 2019/945)*, European Aviation Safety Agency, 2021.

[21] Frazzoli, Emilio, “Robust hybrid control for autonomous vehicle motion planning”, PhD thesis, Massachusetts Institute of Technology, 2001.

[22] Frazzoli, Emilio, Dahleh, Munther A, and Feron, Eric, “Robust hybrid control for autonomous vehicle motion planning”, in: *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 1, IEEE, 2000, pp. 821–826.

[23] Fridovich-Keil, David et al., “Planning, fast and slow: A framework for adaptive real-time safe trajectory planning”, in: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 387–394.

[24] Gassmann, Victor and Althoff, Matthias, “Verified Polynomial Controller Synthesis for Disturbed Nonlinear Systems”, in: *IFAC-PapersOnLine* 54.5 (2021), pp. 85–90.

[25] Gillula, Jeremy H et al., “Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice”, in: *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 1649–1654.

[26] Goerzen, Chad, Kong, Zhaodan, and Mettler, Bernard, “A survey of motion planning algorithms from the perspective of autonomous UAV guidance”, in: *Journal of Intelligent and Robotic Systems* 57.1 (2010), pp. 65–100.

[27] Grüter, Benedikt et al., “Emergency flight planning using voronoi diagrams”, in: *AIAA Scitech 2019 Forum*, 2019, p. 1056.

[28] Guerreiro, Nelson M et al., “Mission planner algorithm for urban air mobility–initial performance characterization”, in: *AIAA Aviation 2019 Forum*, 2019, p. 3626.

[29] Guglieri, Giorgio, Lombardi, Alessandro, and Ristorto, Gianluca, “Operation oriented path planning strategies for RPAS”, in: *American Journal of Science and Technology* 2.6 (2015), pp. 1–8.

[30] Gursky, Bianca I, Olsman, WFJ, and Peinecke, Niklas, “Development of a tunnel-in-the-sky display for helicopter noise abatement procedures”, in: *CEAS Aeronautical Journal* 5.2 (2014), pp. 199–208.

[31] Hovenburg, Anthony Reinier et al., “Contingency path planning for hybrid-electric UAS”, in: *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, IEEE, 2017, pp. 37–42.

[32] *ICAO Annex 14: Aerodromes Volume II, Heliports, Fourth edition*, International Civil Aviation Organisation, 2013.

[33] *JARUS guidelines on Specific Operations Risk Assessment (SORA)*, Joint Authorities for Rulemaking of Unmanned Systems, 2019.

[34] Jeannin, Jean-Baptiste et al., “Formal verification of ACAS X, an industrial airborne collision avoidance system”, in: *2015 International Conference on Embedded Software (EMSOFT)*, IEEE, 2015, pp. 127–136.

[35] Kavraki, Lydia E, “Random networks in configuration space for fast path planning”, PhD thesis, stanford university, 1995.

[36] Kavraki, Lydia E et al., “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, in: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[37] Kochdupmer, N. et al., *AROC 2020 Manual*, 2020, URL: `https://tumcps.github.io/AROC/data/Aroc2020Manual.pdf`.

[38] Landi, Alessandro and Nicholson, Mark, “ARP4754A/ED-79A-guidelines for development of civil aircraft and systems-enhancements, novelties and key topics”, in: *SAE International Journal of Aerospace* 4.2011-01-2564 (2011), pp. 871–879.

[39] Latombe, J.C., *Robot Motion Planning: Edition en anglais*, The Springer International Series in Engineering and Computer Science, Springer, 1991, ISBN: 9780792391296, URL: `https://books.google.de/books?id=Mbo\_p4-46-cC`.

[40] LaValle, Steven M, *Planning algorithms*, Cambridge university press, 2006.

[41] LaValle, Steven M et al., “Rapidly-exploring random trees: A new tool for path planning”, in: (1998).

[42] Majumdar, Anirudha and Tedrake, Russ, “Funnel libraries for real-time robust feedback motion planning”, in: *The International Journal of Robotics Research* 36.8 (2017), pp. 947–982.

[43] Manfredi, Guido and Jestin, Yannick, “An introduction to ACAS Xu and the challenges ahead”, in: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–9.

[44] Medvedev, Nikolai N et al., “An algorithm for three-dimensional Voronoi S-network”, in: *Journal of computational chemistry* 27.14 (2006), pp. 1676–1692.

[45] Mulder, Max, “Cybernetics of tunnel-in-the-sky displays”, PhD thesis, Technische Universiteit Delft, 1999.

[46] *Opinion No 01/2020: High-level regulatory framework for the U-space*, European Aviation Safety Agency, 2020.

[47] Ortlieb, Markus and Adolf, Florian-Michael, “Rule-based path planning for unmanned aerial vehicles in non-segregated air space over congested areas”, in: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, IEEE, 2020.

[48] Ortlieb, Markus, Adolf, Florian-Michael, and Holzapfel, Florian, “Computation of a Database of Trajectories and Primitives for Decision-Based Contingency Management of UAVs over Congested Areas”, in: *2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC)*, IEEE, 2021.

[49] Ortlieb, Markus, Adolf, Florian-Michael, and Holzapfel, Florian, “Protected Online Path Planning for UAVs over Congested Areas Within Convex Regions of Obstacle-Free Space”, in: *2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC)*, IEEE, 2021.

[50] Ortlieb, Markus, Konopka, Jannis, and Adolf, Florian-Michael, “Modular Modelling of Ground and Air Risks for Unmanned Aircraft Operations Over Congested Areas”, in: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, IEEE, 2020.

[51] Pinchetti, Federico, Joos, Alexander, and Fichter, Walter, “Efficient continuous curvature path generation with pseudo-parametrized algebraic splines”, in: *CEAS Aeronautical Journal* 9.4 (2018), pp. 557–570.

[52] Plate, EJ, Kiefer, H, and Wacker, J, “Wind and urban climates”, in: *Fifth Conference on Urban Environment*, 2004.

[53] Primatesta, Stefano, Rizzo, Alessandro, and Cour-Harbo, Anders la, “Ground risk map for unmanned aircraft in urban environments”, in: *Journal of Intelligent & Robotic Systems* 97.3 (2020), pp. 489–509.

[54] Primatesta, Stefano et al., “An innovative algorithm to estimate risk optimum path for unmanned aerial vehicles in urban environments”, in: *Transportation research procedia* 35 (2018), pp. 44–53.

[55] *Private Pilot Manual*, Jeppesen Sanderson, Inc, 2001.

[56] *Proposed Means of Compliance with the Special Condition VTOL, issue 1*, European Aviation Safety Agency, 2020.

[57] *Proposed Means of Compliance with the Special Condition VTOL, MOC VTOL.2115 Draft*, European Aviation Safety Agency, 2021, unpublished.

[58] *Regulation (EU) 2018/11392 (Basic Regulation)*, European Aviation Safety Agency.

[59] *RMT.0230: EASA concept for regulation of Unmanned Aircraft Systems (UAS) operations in ‘certified’ category and Urban Air Mobility - Issue 3.0*, European Aviation Safety Agency, 2021.

[60] Rycroft, Chris, *Voro++: A three-dimensional Voronoi cell library in C++*, tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2009.

[61] Sacharny, David, Henderson, Thomas C, and Cline, Michael, “Large-Scale UAS Traffic Management (UTM) Structure”, in: *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, 2020, pp. 7–12.

[62] Schopferer, Simon and Benders, Sebastian, “Minimum-risk path planning for long-range and low-altitude flights of autonomous unmanned aircraft”, in: *AIAA Scitech 2020 Forum*, 2020, p. 0137.

[63] Schuermann, Bastian and Althoff, Matthias, “Optimizing Sets of Solutions for Controlling Constrained Nonlinear Systems”, in: *IEEE Transactions on Automatic Control* 66.3 (2021), pp. 981–994, DOI: 10.1109/TAC.2020.2989762.

[64] Schürmann, Bastian and Althoff, Matthias, “Convex interpolation control with formal guarantees for disturbed and constrained nonlinear systems”, in: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 2017, pp. 121–130.

[65] Schürmann, Bastian and Althoff, Matthias, “Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space”, in: *IFAC-PapersOnLine* 50.1 (2017), pp. 11515–11522.

[66] Schürmann, Bastian and Althoff, Matthias, “Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems”, in: *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 2522–2529.

[67] Siméon, Thierry, Laumond, J-P, and Nissoux, Carole, “Visibility-based probabilistic roadmaps for motion planning”, in: *Advanced Robotics* 14.6 (2000), pp. 477–493.

[68] *Special Condition for small-category VTOL aircraft*, European Aviation Safety Agency, 2019.

[69] *Standardised European Rules of the Air (SERA)*, European Aviation Safety Agency.

[70] Tanemura, Masaharu, Ogawa, Tohru, and Ogita, Naofumi, “A new algorithm for three-dimensional Voronoi tessellation”, in: *Journal of Computational Physics* 51.2 (1983), pp. 191–207.

[71] Tang, Hualong and Zhang, Yu, “Airspace Design and Trajectory Planning for Urban Air Mobility (UAM) Traffic Management System”, in: ().

[72] Ten Harmsel, Alec J, Olson, Isaac J, and Atkins, Ella M, “Emergency flight planning for an energy-constrained multicopter”, in: *Journal of Intelligent & Robotic Systems* 85.1 (2017), pp. 145–165.

[73] *U-Space Blueprint*, SESAR Joint Undertaking, 2017, URL: https://www.sesarju.eu/sites/default/files/documents/reports/Uspace%20Blueprint%20brochure%20final.PDF.

[74] Usach, Hector, Vila, Juan A, and Gallego, Áurea, “Trajectory-Based, Probabilistic Risk Model for UAS Operations”, in: *Risk Assessment in Air Traffic Management*, IntechOpen, 2020, p. 125.

[75] Vascik, Parker D, “Systems analysis of urban air mobility operational scaling”, PhD thesis, Massachusetts Institute of Technology, 2020.

[76] *voloCHRIS Final Project Report*, mFUND, Bundesministerium für Verkehr und digitale Infrastruktur, 2020.

[77] Washington, Achim, Clothier, Reece A, and Silva, Jose, “A review of unmanned aircraft system ground risk models”, in: *Progress in Aerospace Sciences* 95 (2017), pp. 24–44.

[78] Zhang, Tongjie Y and Suen, Ching Y., “A fast parallel algorithm for thinning digital patterns”, in: *Communications of the ACM* 27.3 (1984), pp. 236–239.

Appendix

A. Extended Results Illustration for the Urban Validation Scenario

This appendix provides additional illustrations of the CSFL planning solution for the urban validation scenario (see Section 6.3). Selected extracts of the trajectory database are plotted on the Google Earth surface model of London, UK.

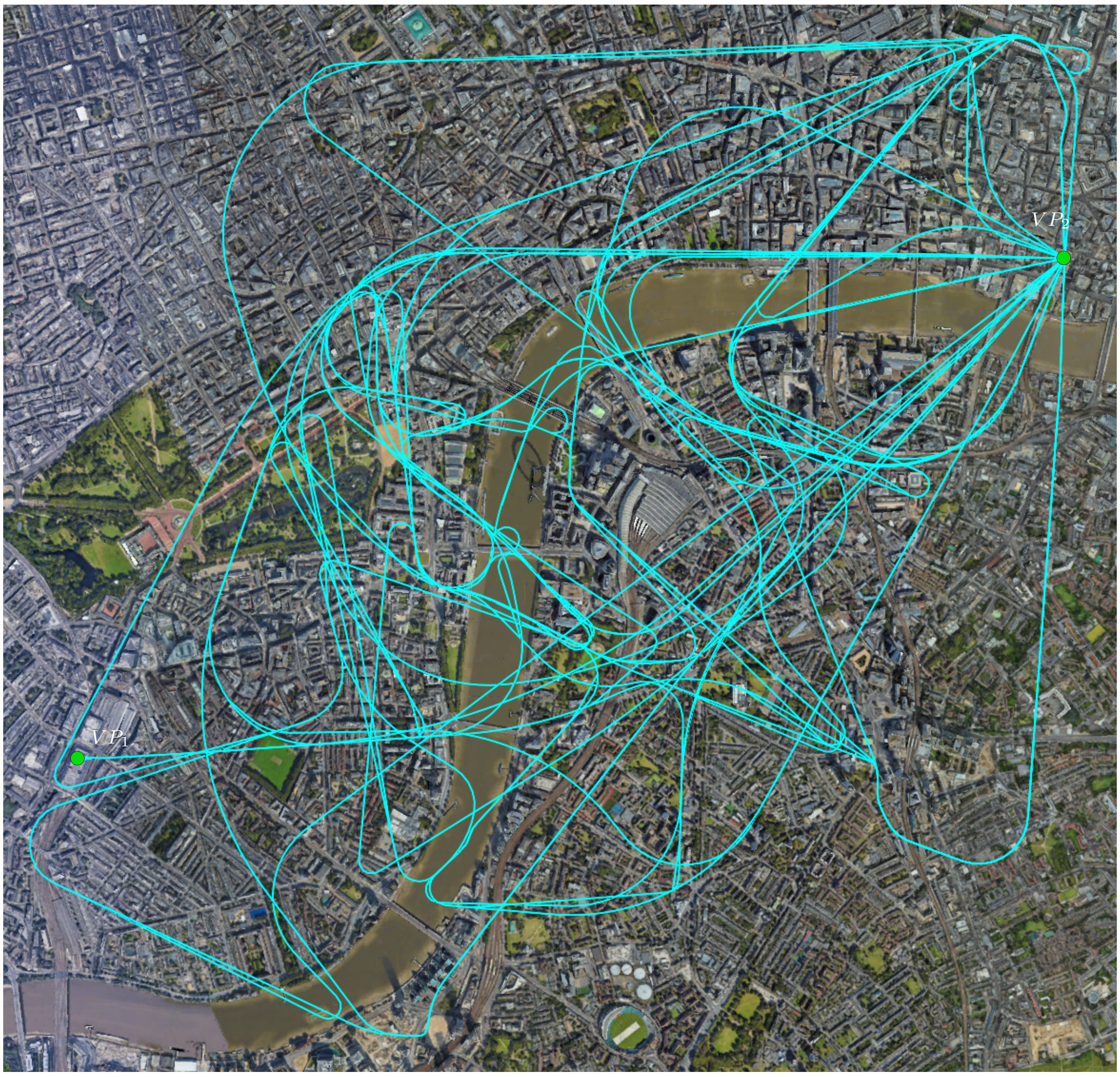


Figure A-1: A top view of the trajectory tree $\mathcal{T}_{2,2}$ on the satellite map. 102 trajectories connect VP_1 at Victoria station with VP_2 at Cannon Street. With increasing proximity to the destination, more trajectories coincide or near-coincide.

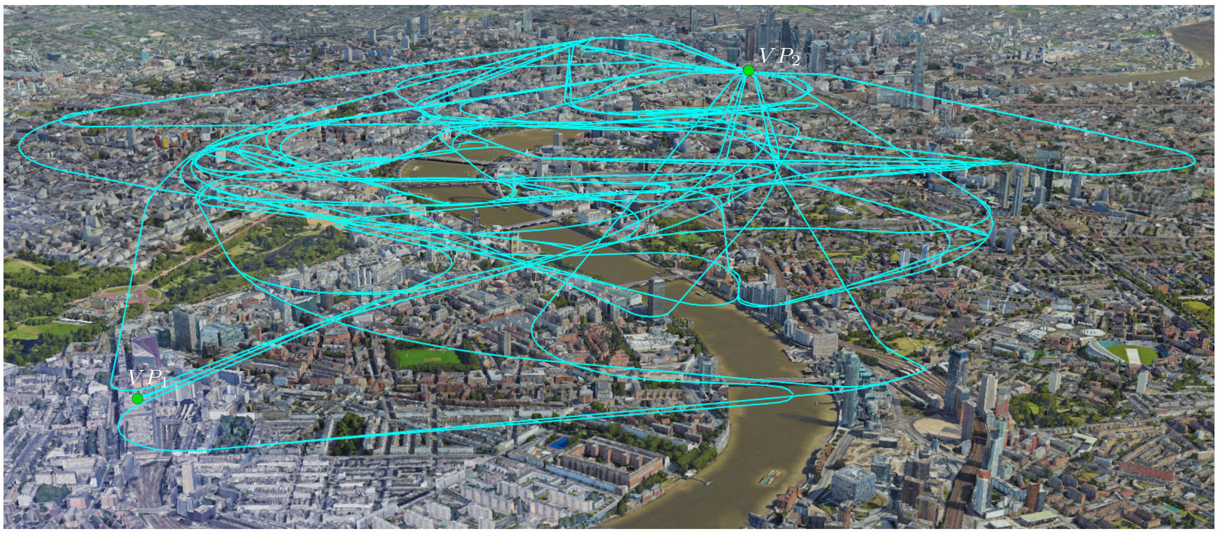


Figure A-2: A three-dimensional perspective of the trajectory tree $\mathcal{T}_{2,2}$. It can be observed, how trajectories descend into vertiport VP_2 on the approach funnel.

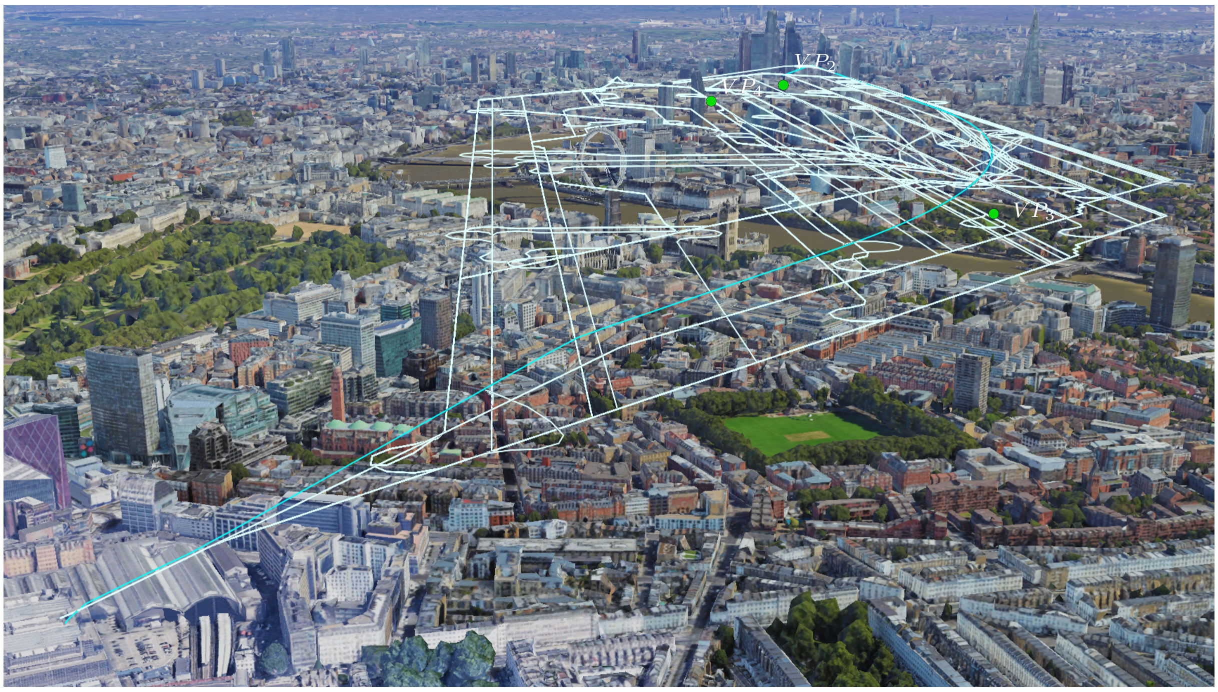


Figure A-3: A three-dimensional perspective on the CSFL solution (white) for the optimal trajectory τ'_{opt} (cyan) in $\mathcal{T}_{2,2}$ with $\Delta t_C = 30s$. The use of two distinct flight levels at $100m$ and $150m$ AGL for contingency planning is evident. Where a trajectory starts above the contingency planning surface, an initial descend onto the flight level is performed. The third flight level at $200m$ AGL is not illustrated, since no contingency trajectories is planned on this level. However, it is available for transitions from a lower flight level (and back).

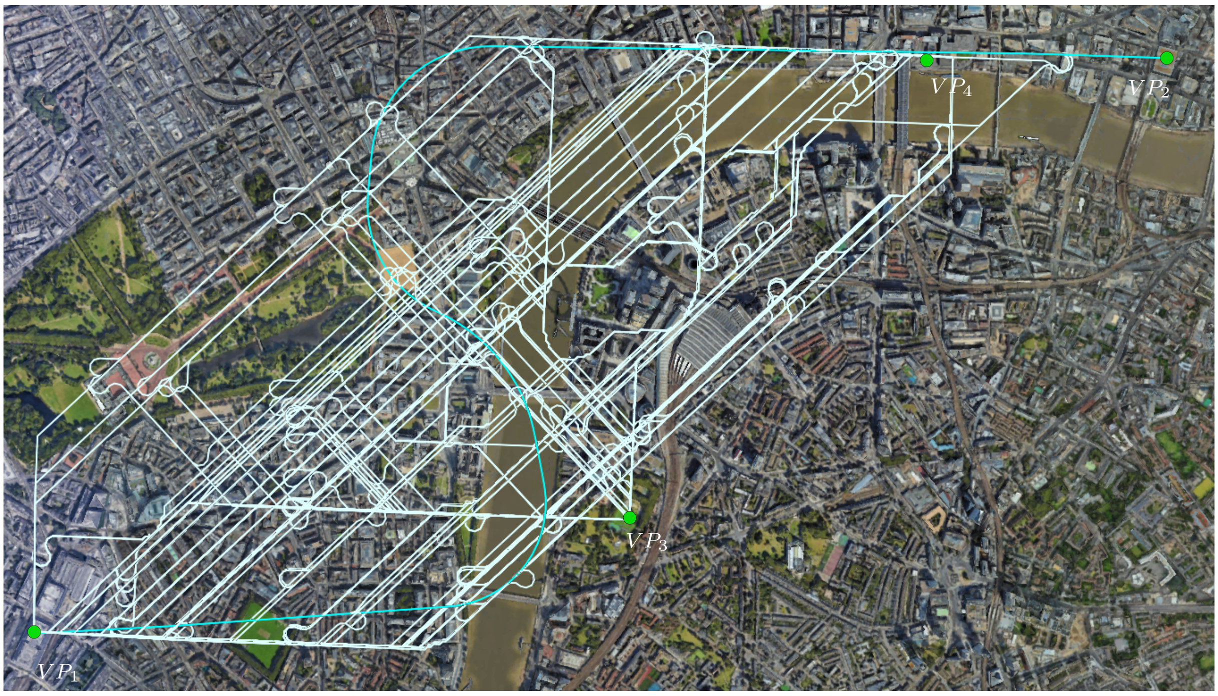


Figure A-4: A top view of an alternative path (cyan) in the trajectory tree $\mathcal{T}_{2,2}$ and respective CSFL solution (white) with $\Delta t_C = 30s$. 439 alternate trajectories are planned to alternate vertiports VP_1, VP_3 and VP_4. The London Eye ferris wheel and several high-rise building along the river are obstacles across all flight levels.

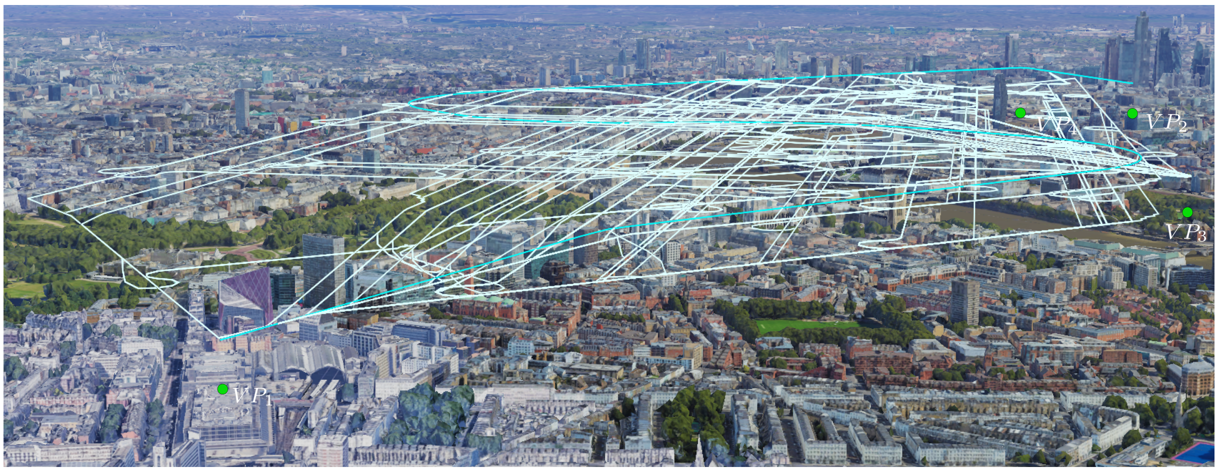


Figure A-5: Three-dimensional perspective of the alternative path and CSFL solution in Fig. A-4. Just as in Fig. A-3, the use of flight levels in the contingency planning can be observed. The approach funnels of vertiports VP_1 and VP_3 and the vertical descend manoeuvre to reach the vertiports from the funnel's bottom can be identified.

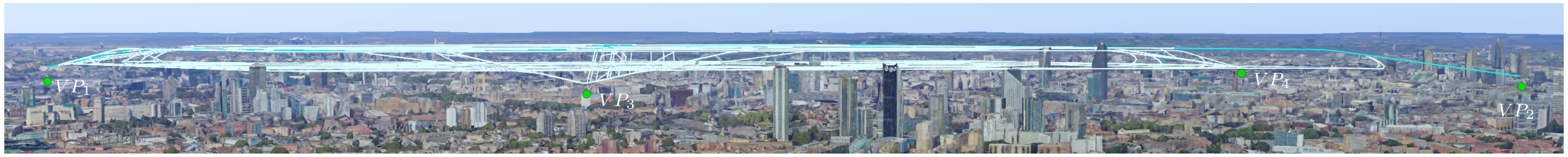


Figure A-6: The CSFL solution of Fig. A-4 and Fig. A-5 from a different perspective. The planning solution uses different flight levels and well-defined altitude profiles with funnels at each vertiport.